Pourquoi apprendre Metasploit :

Comme je l'ai dit au début de ce tutoriel, Metasploit est un Framework (ensemble d'outils et de logiciels organisés qui simplifie une tâche complexe) qui va nous être utile dans les différentes étapes d'un test d'intrusion.

Durant le processus d'intrusion, on a besoin de *scanner*, *fuzzer*, *payload*, *encoder etc*… Au lieu d'être dispersé à gauche et à droite avec différents outils, Metasploit nous offre l'opportunité de regrouper tous ces outils sous son aile, avec un tas d'autres fonctionnalités et scripts dédiés au hacking.

Cela étant dit, le *hacker* utilisant Metasploit va optimiser en temps et en efficacité ses hacks. En plus du large spectre d'outils qu'offre ce Framework, la capacité de modifier voir créer des modules propres à nous, fait de lui une arme suisse pour les *hackers*.

Un peu d'histoire :

Metasploit a été développé par le fameux ***HD Moore***, ce dernier a constaté lors de son emploi dans une entreprise de sécurité qu'il passait une grande partie de son temps à la correction et la validation des exploits publiques. Il a eu l'idée de créer un petit Framework (à cette époque-là) pour la création et le développement d'exploits, il a lancé la première édition de Metasploit basé sur Perl en octobre 2003 avec un total de 11 exploits.

Le projet a pris de l'ampleur au fil du temps ainsi qu'un grand feed-back positif de la part de la communauté des pentesters, à tel point qu'en 2007, il est devenu un Framework indispensable pour tout spécialiste en sécurité informatique.

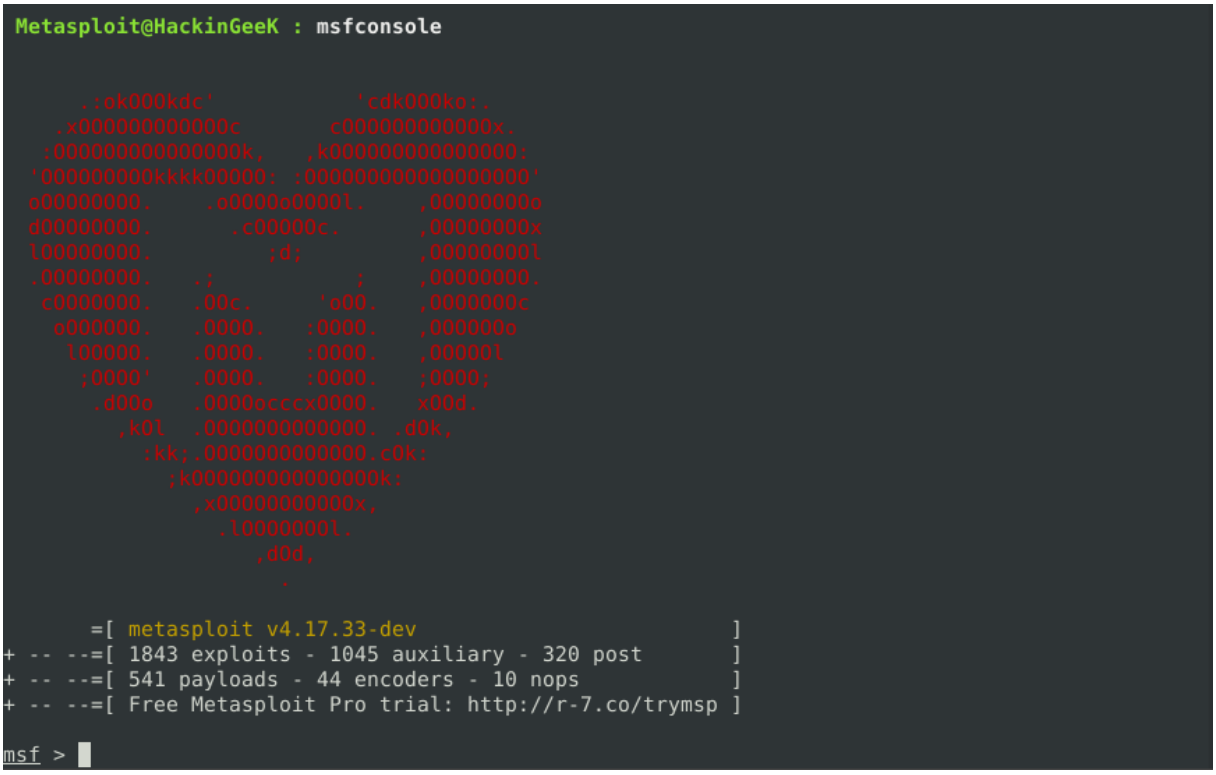
Durant la même année ce projet a été réécrit entièrement en Ruby et a gagné un total de 150 000 lignes de codes en plus, faisant la sortie de Metasploit 3.0.

En 2009 ***Rapid7*** le leader dans le domaine de scan de vulnérabilités est devenu le sponsor officiel du projet Metasploit, ce qui a permis à HD Moore de se focaliser sur son projet à plein temps et d'embaucher des programmeurs pour l'aider. D'où Metasploit a connu une accélération exponentielle.

Après, Rapid7 a libéré deux versions commerciales de Metasploit **Express** et **pro** qui offrent des fonctionnalités plus avancées pour les spécialistes.

Préparation du lab :

Si vous utilisez une distribution Linux orientée sécurité, il est fort possible que Metasploit soit déjà préinstallé. Pour s'en assurer, il suffit de taper ***msfconsole*** sur le terminal et vous obtiendrez une image similaire comme celle-ci :



Si ce n'est pas le cas, la méthode la plus facile est de télécharger l'installeur directement du site officiel.

Quelques notions :

L'utilisation de Metasploit devient vite confuse si vous ne comprenez pas les notions de base utilisées fréquemment dans notre domaine. La compréhension de cette terminologie est indispensable pour pouvoir interagir facilement avec le Framework.

Voici une liste de termes que l'on va rencontrer souvent durant cette série de tutoriels :

Exploit : est une faille dans un système, application ou un service qui va permettre à un attaquant de l'exploiter afin de compromettre sa sécurité.

Payload : est un code que le système va exécuter pour avoir un résultat précis. Exemple : un *reverse shell* est un payload qui se charge de créer une connexion de la machine victime vers la machine de l'attaquant tandis qu'un *bind shell* est un payload qui *lie* (bind) l'invite de commande avec un port en écoute.

Shellcode : semblable aux payloads sauf qu'il est écrit en *assembly. Exemple : a*près avoir exploité une machine, un *meterpreter shell* est fourni pour interagir avec la machine compromise.

Module : un bout de code pouvant être importer. Souvent vous aurez besoin de scripts pour accomplir une tâche donnée, avec MSF vous pouvez l'importer comme un module externe, c'est l'une des beautés de ce Framework.

Listener : lorsque vous exploitez un système, ce dernier va tenter d'établir une connexion avec votre machine. Pour cela le *listener* va écouter jusqu'à l'établissement de la connexion.

Les interfaces de Metasploit :

Metasploit offre plusieurs interfaces d'interaction, je vais vous initier aux plus populaires :

msfconsole : est l'interface la plus utilisée et aussi la plus puissante, basée sur la ligne de commande ce qui lui confère une flexibilité et une richesse extrême ainsi qu'un support de la communauté. Depuis cette interface, vous pouvez exécuter un exploit, importer un module, créer un listener et beaucoup d'autres choses.

Armitage : c'est une interface graphique (GUI). Les débutants l'aiment bien parce qu'elle est assez intuitive (et riche mais pas autant que *msfconsole*). Pour comprendre et maîtriser Metasploit il faut faire travailler ses méninges, c'est vrai que ça peut faire peur mais une fois qu'on est habitué, ça devient une zone de confort 😜.

Utilitaire de Metasploit :

Il est injuste de parler de *msfconsole* sans parler de ***msfvenom.*** En fait, c'est un composant majeur de Metasploit qui vous permet de générer des *payloads*, *exécutables*, *shellcodes*, *apk* pour les utiliser dans vos exploitations.

Vous pouvez générer des shellcodes écrits en plusieurs formats C, *Ruby*, *pyth²on*, *JavaScript etc…* selon votre situation et vos besoins.

Cependant, ces shellcodes créés ne sont pas si effaces que ça en a l'air. Ils contiennent de mauvais caractères ce qui fait qu'un antivirus stupide peut les attraper facilement, c'est pourquoi vous aurez besoin d'une autre fonctionnalité de *msfvenom* qui est ***l'encodage***, ce dernier permet d'éviter ces mauvais caractères.

Pour afficher tous les encodeurs que vous pouvez utiliser entrez cette commande :

```
msfvenom -l encoders
```

Et voici le résultat :

```
   mipsbe/byte_xori              normal     Byte XORi Encoder
   mipsbe/longxor                normal     XOR Encoder
   mipsle/byte_xori              normal     Byte XORi Encoder
   mipsle/longxor                normal     XOR Encoder
   php/base64                    great      PHP Base64 Encoder
   ppc/longxor                   normal     PPC LongXOR Encoder
   ppc/longxor_tag               normal     PPC LongXOR Encoder
   ruby/base64                   great      Ruby Base64 Encoder
   sparc/longxor_tag             normal     SPARC DWORD XOR Encoder
   x64/xor                       normal     XOR Encoder
   x64/xor_dynamic               normal     Dynamic key XOR Encoder
   x64/zutto_dekiru              manual     Zutto Dekiru
   x86/add_sub                   manual     Add/Sub Encoder
   x86/alpha_mixed               low        Alpha2 Alphanumeric Mixedcase Encoder
   x86/alpha_upper               low        Alpha2 Alphanumeric Uppercase Encoder
   x86/avoid_underscore_tolower  manual     Avoid underscore/tolower
   x86/avoid_utf8_tolower        manual     Avoid UTF8/tolower
   x86/bloxor                    manual     BloXor - A Metamorphic Block Based XOR Encoder
   x86/bmp_polyglot              manual     BMP Polyglot
   x86/call4_dword_xor           normal     Call+4 Dword XOR Encoder
   x86/context_cpuid             manual     CPUID-based Context Keyed Payload Encoder
   x86/context_stat              manual     stat(2)-based Context Keyed Payload Encoder
   x86/context_time              manual     time(2)-based Context Keyed Payload Encoder
   x86/countdown                 normal     Single-byte XOR Countdown Encoder
   x86/fnstenv_mov               normal     Variable-length Fnstenv/mov Dword XOR Encoder
   x86/jmp_call_additive         normal     Jump/Call XOR Additive Feedback Encoder
   x86/nonalpha                  low        Non-Alpha Encoder
   x86/nonupper                  low        Non-Upper Encoder
   x86/opt_sub                   manual     Sub Encoder (optimised)
   x86/service                   manual     Register Service
   x86/shikata_ga_nai            excellent  Polymorphic XOR Additive Feedback Encoder
   x86/single_static_bit         manual     Single Static Bit
   x86/unicode_mixed             manual     Alpha2 Alphanumeric Unicode Mixedcase Encoder
   x86/unicode_upper             manual     Alpha2 Alphanumeric Unicode Uppercase Encoder
   x86/xor_dynamic               normal     Dynamic key XOR Encoder
```

Notez que le meilleur classement est celui de l'encodeur ***x86/shikata_ga_nai*** (excellent) qu'on va utiliser dans un prochain exploit.

Metasploit… Collecte d'informations :

Maintenant que vous avez les bases, nous allons approfondir nos connaissances en respectant la méthodologie du hacking en commençant par : La collecte d'informations.

N.B : afin d'appliquer ce qu'on va voir dans les étapes suivantes nous avons besoin d'un environnement propice et légal pour hacker dans des conditions optimales.

La collecte d'informations se divise en deux grands paliers : *passive* et *active* nous allons détailler les deux procédures ainsi que les outils utilisés pour chacune.

Collecte d'informations passive :

La collecte d'informations *passive* ou *indirecte* vous permet de récolter des informations concernant votre cible sans interagir avec elle. Autrement dit sans inter-changement de paquets entre vous et votre cible, ce qui va éliminer totalement tous risques de détection de votre recherche.

Votre but de cette méthode est d'avoir une idée générale sur l'infrastructure de votre cible : l'hébergeur, les systèmes d'exploitation et leurs versions, les noms des serveurs…

Pour effectuer notre collecte, je vais utiliser **whois lookup.** C'est un outil qui contient une base de données immense sur les domaines publiques disponibles sur la toile, la simplicité de son utilisation et sa puissance le rend l'outil préféré des *hackers* quand ça concerne la collecte d'informations.

Tout d'abord, installez *whois* depuis votre msfconsole avec la commande suivante :

```
apt install whois
```

Puis pour avoir une vue d'ensemble d'un domaine quelconque, tapez cette commande :

```
whois domaine.com
```

Vous aurez des informations de celui-ci dans une forme bien organisée :

```
msf > whois google.com
[*] exec: whois google.com

   Domain Name: GOOGLE.COM
   Registry Domain ID: 2138514_DOMAIN_COM-VRSN
   Registrar WHOIS Server: whois.markmonitor.com
   Registrar URL: http://www.markmonitor.com
   Updated Date: 2018-02-21T18:36:40Z
   Creation Date: 1997-09-15T04:00:00Z
   Registry Expiry Date: 2020-09-14T04:00:00Z
   Registrar: MarkMonitor Inc.
   Registrar IANA ID: 292
   Registrar Abuse Contact Email: abusecomplaints@markmonitor.com
   Registrar Abuse Contact Phone: +1.2083895740
   Domain Status: clientDeleteProhibited https://icann.org/epp#clientDeleteProhibited
   Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
   Domain Status: clientUpdateProhibited https://icann.org/epp#clientUpdateProhibited
   Domain Status: serverDeleteProhibited https://icann.org/epp#serverDeleteProhibited
   Domain Status: serverTransferProhibited https://icann.org/epp#serverTransferProhibited
   Domain Status: serverUpdateProhibited https://icann.org/epp#serverUpdateProhibited
   Name Server: NS1.GOOGLE.COM
   Name Server: NS2.GOOGLE.COM
   Name Server: NS3.GOOGLE.COM
   Name Server: NS4.GOOGLE.COM
   DNSSEC: unsigned
   URL of the ICANN Whois Inaccuracy Complaint Form: https://www.icann.org/wicf/
>>> Last update of whois database: 2019-01-20T18:11:47Z <<<
```

Dans cet exemple, la collecte d'informations concernant *google.com* nous divulgue que le nom du serveur dans lequel le nom de domaine est hébergé est : **NSX.GOOGLE.COM**, ce qui veut dire que le DNS (Domain Name System) est hébergé par l'organisation elle-même qui est **Google**. Un détail qui vous donne un aperçu sur la taille de l'organisation que vous ciblez.

Maintenant si vous voulez connaitre l'adresse IP de votre cible, vous pouvez utiliser un autre outil : ***nslookup*** (toujours depuis msf) :

```
msf > nslookup google.com
[*] exec: nslookup google.com

Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   google.com
Address: 172.217.18.46
Name:   google.com
Address: 2a00:1450:4006:807::200e
```

Actuellement, nous avons construit une vue générale sur notre cible incluant l'hébergeur, l'adresse IP, le pays… Et plusieurs d'autres informations qu'il ne faut absolument pas négliger, il se peut qu'un petit détail fera la différence entre le succès et l'échec de votre test d'intrusion.

Maintenant, on va devoir entamer la phase *active* pour avoir des informations plus poussées.

Collecte d'informations active :

Avec la collecte ***active***, l'interaction avec le système ciblé est **directe**, c'est-à-dire que vous êtes entrain d'envoyer et de recevoir des paquets avec votre cible. De cette façon vous obtiendrez des informations plus qualitatives (Application en exécution, version du système d'exploitation, port ouvert…) mais les risques d'être détecté par un **IDS** (Intrusion Detection System) ou un **IPS** (Intrusion Prevention System) sont élevés.

C'est pour cette raison qu'il faut maîtriser cette étape pour en extraire le plus d'informations avec le moindre bruit possible.

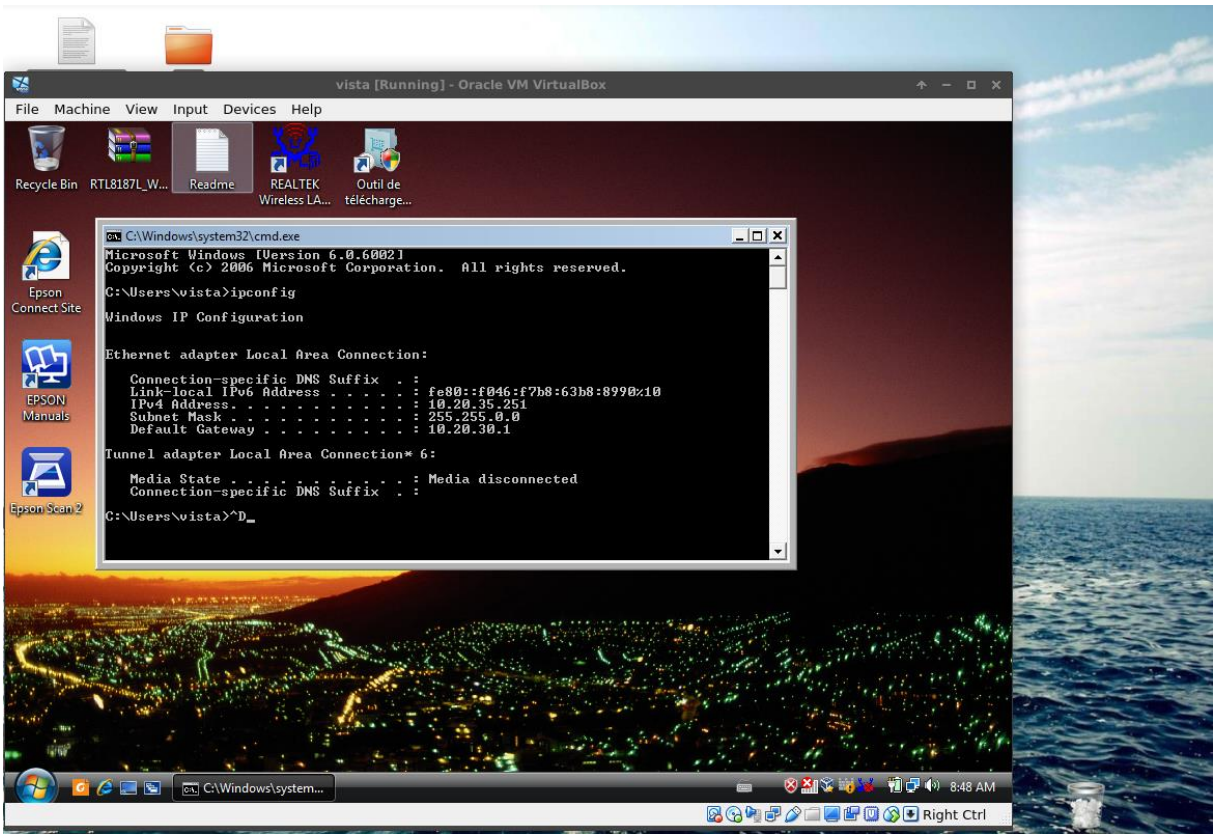
Nmap….. Scan de port :

Un port dans un système est une application utilisée comme une interface exécutant un service, par exemple : le port 80 est réservé aux applications web dont toutes les informations de cette application sont échangées depuis ce port.

Le fameux **Nmap** est un outil qui nous permet de scanner ces ports et d'identifier les services en exécution ainsi que beaucoup d'autres fonctionnalités avancées (NSE).

Le principe du scan est simple, Nmap envoie un paquet au système dans un port quelconque et si le système répond (n'importe quelle réponse) cela veut dire que le port est ouvert. Si le système ne répond pas cela veut dire le port est fermé ou bien filtré.

Afin de démontrer l'utilisation de Nmap je vais utiliser une machine virtuelle comme cible :



N.B : l'utilisation de Nmap sur des systèmes qui ne vous appartiennent pas est illégale, c'est pour cette raison que j'utilise une machine virtuelle.

Pour voir toutes les fonctionnalités qu'offre nmap entrez simplement la commande :

```
nmap -h
```

Une liste d'usage s'affiche décrivant toutes les variantes qu'on peut utiliser, il est très important de lire la documentation avant d'entamer votre scan.

```
[*] exec: nmap -h

Nmap 7.70 ( https://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sn: Ping Scan - disable port scan
  -Pn: Treat all hosts as online -- skip host discovery
  -PS/PA/PU/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
  -PO[protocol list]: IP Protocol Ping
  -n/-R: Never do DNS resolution/Always resolve [default: sometimes]
  --dns-servers <serv1[,serv2],...>: Specify custom DNS servers
  --system-dns: Use OS's DNS resolver
  --traceroute: Trace hop path to each host
```

Disons que je veux opter pour un scan basique, pour cela je ne lui passe aucun argument :

```
msf auxiliary(scanner/ip/ipidseq) > nmap 10.20.35.251
[*] exec: nmap 10.20.35.251

Starting Nmap 7.70 ( https://nmap.org ) at 2019-01-19 17:50 CET
Nmap scan report for 10.20.35.251
Host is up (0.0013s latency).
Not shown: 999 filtered ports
PORT     STATE SERVICE
5357/tcp open  wsdapi
MAC Address: 08:00:27:A4:FF:F8 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 13.54 seconds
```

Il y a quelques informations utiles dans ce screen-shot : nmap nous informe que 999 ports sont filtrés (et pas fermés), le port numéro 5 357 est ouvert et utilise le service **wsdapi**, puis nous avons l'adresse MAC du système ainsi que son type (system virtuel).

N.B : si vous exécutez la même commande pour un système Linux en tant que cible, un message d'erreur se lève indiquant la déconnexion de votre cible alors qu'elle est belle et bien connectée. Dans ce cas la, il nous recommande d'utiliser l'argument **-Pn** afin d'éviter l'utilisation de ping (ICMP packets) pour extraire les informations que beaucoup de systèmes bloquent comme mesure de protection.

```
msf > nmap 10.0.2.15
[*] exec: nmap 10.0.2.15

Starting Nmap 7.70 ( https://nmap.org ) at 2019-01-20 22:15 CET
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.39 seconds
```

Vous devez dans ce cas adapter votre commande :

nmap -Pn IP

```
msf > nmap -Pn 10.0.2.15
[*] exec: nmap -Pn 10.0.2.15

Starting Nmap 7.70 ( https://nmap.org ) at 2019-01-20 22:15 CET
Nmap scan report for 10.0.2.15
Host is up (0.032s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE
53/tcp open  domain

Nmap done: 1 IP address (1 host up) scanned in 11.69 seconds
```

Revenons à notre exemple, vous pouvez utiliser l'argument **-O** pour détecter le système d'exploitation en marche :

```
msf auxiliary(scanner/ip/ipidseq) > nmap -O 10.20.35.251
[*] exec: nmap -O 10.20.35.251

Starting Nmap 7.70 ( https://nmap.org ) at 2019-01-19 17:47 CET
Nmap scan report for 10.20.35.251
Host is up (0.00087s latency).
Not shown: 999 filtered ports
PORT     STATE SERVICE
5357/tcp open  wsdapi
MAC Address: 08:00:27:A4:FF:F8 (Oracle VirtualBox virtual NIC)
Warning: OSScan results may be unreliable because we could not find at least 1 open and
1 closed port
OS details: Microsoft Windows Server 2008 or 2008 Beta 3, Microsoft Windows Server 2008
R2 or Windows 8.1, Microsoft Windows 7 Professional or Windows 8, Microsoft Windows Embe
dded Standard 7, Microsoft Windows Phone 7.5 or 8.0, Microsoft Windows Vista SP0 or SP1,
 Windows Server 2008 SP1, or Windows 7, Microsoft Windows Vista SP2, Windows 7 SP1, or W
indows Server 2008
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/
.
Nmap done: 1 IP address (1 host up) scanned in 11.27 seconds
```

Il nous dit que le système en marche est de type Windows (ce n'est pas assez précis 😕).

Si vous voulez plus d'informations, utilisez l'argument **-A** :

```
[*] exec: nmap -O -A 10.20.35.251

Starting Nmap 7.70 ( https://nmap.org ) at 2019-01-19 18:26 CET
Nmap scan report for 10.20.35.251
Host is up (0.0025s latency).
Not shown: 999 filtered ports
PORT     STATE SERVICE    VERSION
5357/tcp open  tcpwrapped
MAC Address: 08:00:27:A4:FF:F8 (Oracle VirtualBox virtual NIC)
Warning: OSScan results may be unreliable because we could not find at least 1 open and
1 closed port
Device type: general purpose|specialized|phone
Running: Microsoft Windows 2008|8.1|7|Phone|Vista
OS CPE: cpe:/o:microsoft:windows_server_2008::beta3 cpe:/o:microsoft:windows_server_2008
 cpe:/o:microsoft:windows_8.1 cpe:/o:microsoft:windows_7 cpe:/o:microsoft:windows cpe:/o
:microsoft:windows_vista::- cpe:/o:microsoft:windows_vista::sp1
OS details: Microsoft Windows Server 2008 or 2008 Beta 3, Microsoft Windows Server 2008
R2 or Windows 8.1, Microsoft Windows Embedded Standard 7, Microsoft Windows Phone 7.5 or
 8.0, Microsoft Windows Vista SP0 or SP1, Windows Server 2008 SP1, or Windows 7, Microso
ft Windows Vista SP2, Windows 7 SP1, or Windows Server 2008
Network Distance: 1 hop

TRACEROUTE
HOP RTT     ADDRESS
1   2.45 ms 10.20.35.251

OS and Service detection performed. Please report any incorrect results at https://nmap.
org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 41.66 seconds
```

Nmap… Zombie scan :

Une des fonctionnalités avancées de Nmap est le ***idle scan*** ou scan par zombie, le principe est le suivant ; vous allez *spoofer* une adresse IP d'une machine qui se trouve dans le même réseau que vous. Puis vous allez effectuer votre scan en se cachant derrière (faire de lui un zombie) et qui va jouer le rôle d'intermédiaire entre vous et votre cible.

En fait, toute l'attaque repose sur l'*ID incrémentiel* qui est un nombre utilisé pour marquer chaque paquet envoyé du système, cela va facilite la prédiction de la réponse du système recevant le paquet.

Tout d'abord, il faut identifier un zombie. Pour cela, nous allons utiliser le module ***scanner/ip/ipidseq*** pour mapper tout le réseau à la recherche d'un hôte qui correspond à nos besoins :

```
msf > use auxiliary/scanner/ip/ipidseq
msf auxiliary(scanner/ip/ipidseq) > show options

Module options (auxiliary/scanner/ip/ipidseq):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   INTERFACE                   no        The name of the interface
   RHOSTS                      yes       The target address range or CIDR identifier
   RPORT      80               yes       The target port
   SNAPLEN    65535            yes       The number of bytes to capture
   THREADS    1                yes       The number of concurrent threads
   TIMEOUT    500              yes       The reply read timeout in milliseconds

msf auxiliary(scanner/ip/ipidseq) >
```

La commande *show options* vous affiche les options nécessaires au fonctionnement du module, certaines sont prédéfinies et d'autres non, c'est à vous de les définir.

La seule option qui nous intéresse ici, c'est ***RHOSTS***, c'est l'intervalle d'adresses qu'on veut scanner, il peut être de ce format (192.168.1.1-192.168.1.254) ou bien 192.168.1.1/24, c'est à vous de choisir, puis exécutez le module :

```
msf auxiliary(scanner/ip/ipidseq) > set RHOSTS 10.20.35.2-10.20.35.50
RHOSTS => 10.20.35.2-10.20.35.50
msf auxiliary(scanner/ip/ipidseq) > run

[*] 10.20.35.22's IPID sequence class: All zeros
[*] 10.20.35.7's IPID sequence class: All zeros
[*] 10.20.35.25's IPID sequence class: All zeros
[*] 10.20.35.8's IPID sequence class: All zeros
[*] 10.20.35.14's IPID sequence class: All zeros
[*] 10.20.35.23's IPID sequence class: All zeros
[*] 10.20.35.19's IPID sequence class: All zeros
[*] 10.20.35.21's IPID sequence class: All zeros
[*] 10.20.35.16's IPID sequence class: All zeros
[*] 10.20.35.28's IPID sequence class: All zeros
[*] 10.20.35.26's IPID sequence class: All zeros
[*] 10.20.35.11's IPID sequence class: All zeros
[*] 10.20.35.5's IPID sequence class: All zeros
[*] 10.20.35.13's IPID sequence class: All zeros
[*] 10.20.35.4's IPID sequence class: All zeros
[*] 10.20.35.15's IPID sequence class: All zeros
[*] 10.20.35.31's IPID sequence class: All zeros
[*] 10.20.35.17's IPID sequence class: All zeros
[*] 10.20.35.18's IPID sequence class: All zeros
[*] 10.20.35.9's IPID sequence class: All zeros
```

Nous avons une vingtaine d'hôtes que l'on ne peut **PAS** utiliser, car l'IP ID n'est pas incrémentiel. Du coup ils ne sont pas vulnérables à cette attaque 😜.

Si ce n'est pas le cas pour vous (présence de Windows XP dans votre réseau), vous pouvez continuer votre attaque en utilisant la commande suivante :

nmap -sI -A IP_SPOOFÉ IP_CIBLE

SSH ….scan :

Si durant votre scan vous trouvez une machine exécutant le protocole ssh (secure shell), vous devez savoir quelle est sa version. Il se peut que vous tombiez sur une version qui n'est pas mise à jour, ce qui augmente considérablement le risque d'avoir une faille dedans.

Afin d'identifier la version de *ssh* on va devoir utiliser le module **scanner/ssh/ssh_version** :

```
msf > use auxiliary/scanner/ssh/ssh_version
msf auxiliary(scanner/ssh/ssh_version) > show options

Module options (auxiliary/scanner/ssh/ssh_version):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS   10.0.2.15        yes       The target address range or CIDR identifier
   RPORT    22               yes       The target port (TCP)
   THREADS  20               yes       The number of concurrent threads
   TIMEOUT  30               yes       Timeout for the SSH probe

msf auxiliary(scanner/ssh/ssh_version) > set RHOSTS 192.168.1.1
RHOSTS => 192.168.1.1
msf auxiliary(scanner/ssh/ssh_version) > run

[+] 192.168.1.1:22        - SSH server version: SSH-2.0-dropbear_0.48 ( service.version=0.4
8 service.family=Dropbear service.product=Dropbear service.protocol=ssh fingerprint_db=ssh.
banner )
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(scanner/ssh/ssh_version) >
```

Ftp…scan :

Le protocole **ftp** a connu beaucoup de difficultés dès son lancement, c'est un protocole complexe et pas suffisamment sécurisé, le module ***scanner/ftp/ftp_version*** peut nous donner quelques informations :

```
msf > use auxiliary/scanner/ftp/ftp_version
msf auxiliary(scanner/ftp/ftp_version) > show options

Module options (auxiliary/scanner/ftp/ftp_version):

   Name     Current Setting      Required  Description
   ----     ---------------      --------  -----------
   FTPPASS  mozilla@example.com  no        The password for the specified username
   FTPUSER  anonymous            no        The username to authenticate as
   RHOSTS                        yes       The target address range or CIDR identifier
   RPORT    21                   yes       The target port (TCP)
   THREADS  1                    yes       The number of concurrent threads

msf auxiliary(scanner/ftp/ftp_version) > set RHOSTS 192.168.1.1
RHOSTS => 192.168.1.1
msf auxiliary(scanner/ftp/ftp_version) > run

[+] 192.168.1.1:21        - FTP Banner: '220 FTP server (192.168.1.1) ready.\x0d\x0a'
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(scanner/ftp/ftp_version) >
```

Autre point très important, il faut avoir le réflexe de chercher des exploits sur exploitDB, quelle que soit la version que vous trouverez, des *zero days il* y en a tous les jours. Donc ne sous-estimez la possibilité d'avoir un exploit et ne soyez pas paresseux 😉.

N.B : si vous voulez voir la liste de tous les modules scanner, il suffit de taper use auxialiary/ + TAB (↹), vous aurez ainsi toute la liste des modules :

```
msf > use auxiliary/scanner/
Display all 554 possibilities? (y or n)
use auxiliary/scanner/acpp/login
use auxiliary/scanner/afp/afp_login
use auxiliary/scanner/afp/afp_server_info
use auxiliary/scanner/backdoor/energizer_duo_detect
use auxiliary/scanner/chargen/chargen_probe
use auxiliary/scanner/couchdb/couchdb_enum
use auxiliary/scanner/couchdb/couchdb_login
use auxiliary/scanner/db2/db2_auth
use auxiliary/scanner/db2/db2_version
use auxiliary/scanner/db2/discovery
use auxiliary/scanner/dcerpc/endpoint_mapper
use auxiliary/scanner/dcerpc/hidden
use auxiliary/scanner/dcerpc/management
use auxiliary/scanner/dcerpc/tcp_dcerpc_auditor
use auxiliary/scanner/dcerpc/windows_deployment_services
use auxiliary/scanner/dect/call_scanner
use auxiliary/scanner/dect/station_scanner
use auxiliary/scanner/discovery/arp_sweep
use auxiliary/scanner/discovery/empty_udp
use auxiliary/scanner/discovery/ipv6_multicast_ping
use auxiliary/scanner/discovery/ipv6_neighbor
use auxiliary/scanner/discovery/ipv6_neighbor_router_advertisement
use auxiliary/scanner/discovery/udp_probe
use auxiliary/scanner/discovery/udp_sweep
use auxiliary/scanner/dlsw/dlsw_leak_capture
```

Voilà, pour cette première partie de Metasploit