

Enumerazione e correzione vulnerabilità in Metasploitable

Nell'esercizio di oggi, proveremo ad enumerare e successivamente correggere alcune vulnerabilità presenti su macchina virtuale Metasploitable. Per identificare le vulnerabilità, effettuiamo una scansione tramite software **Nessus**. A scopo didattico, ci concentreremo sulle vulnerabilità evidenziate nell'immagine sottostante, ad eccezione della seconda in quanto non presente sulla macchina presa in esame.

<input type="checkbox"/>	Sev ▼	Score ▼	Name ▲
<input type="checkbox"/>	CRITICAL	10.0 *	NFS Exported Share Information Disclosure
<input type="checkbox"/>	CRITICAL	10.0 *	rexecd Service Detection
<input type="checkbox"/>	CRITICAL	10.0	Unix Operating System Unsupported Version Detection
<input type="checkbox"/>	CRITICAL	10.0 *	VNC Server 'password' Password
<input type="checkbox"/>	CRITICAL	9.8	Bind Shell Backdoor Detection

Innanzitutto, effettuiamo la scansione tramite Nessus per avere un quadro generale della situazione; dopodiché effettuiamo una ulteriore scansione con **Nmap** sulle porte aperte nella macchina virtuale per aver a disposizione una maggiore quantità di dati e, eventualmente, dei bersagli su cui agire sia in caso di exploit che di azioni correttive.

```
(root@kali)-[/home/giuseppe]
# nmap -sV 192.168.1.101
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-27 13:18 CEST
Nmap scan report for 192.168.1.101
Host is up (0.000056s latency).
Not shown: 977 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
53/tcp    open  domain       ISC BIND 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind      2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec         netkit-rsh rexecd
513/tcp   open  login?
514/tcp   open  shell        Netkit rshd
1099/tcp  open  java-rmi     GNU Classpath grmiregistry
1524/tcp  open  bindshell    Metasploitable root shell
2049/tcp  open  nfs          2-4 (RPC #100003)
2121/tcp  open  ftp          ProFTPD 1.3.1
3306/tcp  open  mysql        MySQL 5.0.51a-3ubuntu5
5432/tcp  open  postgresql   PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc          VNC (protocol 3.3)
6000/tcp  open  X11          (access denied)
6667/tcp  open  irc          UnrealIRCd
8009/tcp  open  ajp13        Apache Jserv (Protocol v1.3)
8180/tcp  open  http         Apache Tomcat/Coyote JSP engine 1.1
MAC Address: 08:00:27:7E:BE:F3 (Oracle VirtualBox virtual NIC)
Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 52.56 seconds
```

Iniziamo adesso con la descrizione delle vulnerabilità e delle azioni correttive proposte.

1) BindShell Backdoor Detection

Come evidenziato dalla scansione, Nessus ha trovato un **bind shell** in ascolto sulla porta **tcp 1524**, come ulteriormente evidenziato dallo scan Nmap, dalla quale un antagonista potrebbe inserirsi per eseguire del codice malevolo senza necessità di credenziali. La soluzione che proponiamo è quella di **bloccare l'accesso alla porta** in esame, così da prevenire eventuali accessi indesiderati. Il procedimento prevede l'utilizzo di **iptables**, il firewall nativo di linux, ed è quanto riportato nelle immagini sottostanti per il nostro specifico caso:

- Creiamo una nuova regola per la **chain INPUT**, ossia quella che regola il traffico in entrata, con il seguente comando:

```
msfadmin@metasploitable:~$  
msfadmin@metasploitable:~$ sudo iptables -A INPUT -p tcp 1524 -j DROP
```

- Verifichiamo che la regola sia stata correttamente presa dal sistema con **sudo iptables -L -n**:

```
Chain INPUT (policy ACCEPT)  
target      prot opt source                destination          tcp dpt:ingreslock  
DROP        tcp  --  anywhere              anywhere  
  
Chain FORWARD (policy ACCEPT)  
target      prot opt source                destination  
  
Chain OUTPUT (policy ACCEPT)  
target      prot opt source                destination
```

- Infine controlliamo con uno scan **Nmap -sV** sul' IP bersaglio che la porta sia effettivamente non disponibile (evidenziamo il dettaglio della porta nell'immagine successiva):

```
1524/tcp filtered ingreslock
```

Andiamo così di fatto a bloccare qualsiasi connessione proveniente dall'esterno sulla porta vulnerabile, chiudendo quindi la strada ad un possibile exploit.

2) VNC Server 'password' Password

Nessus ci informa che la password per il protocollo VNC, usato per l'accesso ed il controllo da remoto dell'host, risulta essere molto vulnerabile, essendo di fatto solo 'password'. Risolviamo questa vulnerabilità **cambiando la password** al servizio come illustrato di seguito:

- Eseguiamo i comandi per cambiare la password, digitandola quando richiesto: la inseriamo infatti una seconda volta per conferma.

```
root@metasploitable:/home/msfadmin# vncpasswd
Using password file /root/.vnc/passwd
Password:
Verify:
Would you like to enter a view-only password (y/n)? n
root@metasploitable:/home/msfadmin#
```

- Effettuiamo una prova da Kali per verificare l'effettiva efficacia del metodo usando il comando **vncviewer**: nel primo caso, prima di cambiare le credenziali, digitando 'password' ottenevamo l'accesso; nel secondo, dopo averle cambiate, l'accesso viene negato, risolvendo di fatto la vulnerabilità così come ci era stata proposta da Nessus.

```
(root@kali)-[/home/giuseppe]
# vncviewer
Connected to RFB server, using protocol version 3.3
Performing standard VNC authentication
Authentication successful
Desktop name "root's X desktop (metasploitable:0)"
VNC server default format:
  32 bits per pixel.
  Least significant byte first in each pixel.
  True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
Using default colormap which is TrueColor. Pixel format:
  32 bits per pixel.
  Least significant byte first in each pixel.
  True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0

(root@kali)-[/home/giuseppe]
# vncviewer
Connected to RFB server, using protocol version 3.3
Performing standard VNC authentication
Authentication failure
```

3) NFS Exported Share Information Disclosure

Nessus ci informa di una vulnerabilità presente a livello di **NFS**, ossia un protocollo la cui funzione è quella di permettere ad un client di accedere e condividere file e directory di un server remoto come se fossero locali. Nessus stesso, indagando il problema, ci reindirizza verso una pagina avente diverse **'best practices'** per questo protocollo, a sua volta indicandoci che il file che gestisce gli accessi ad esso è **/etc/exports**, il quale si presenta nella sua forma base come da immagine seguente:

```
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes      hostname1(rw,sync) hostname2(ro,sync)
#
# Example for NFSv4:
# /srv/nfs4       gss/krb5i(rw,sync,fsid=0,crossmnt)
# /srv/nfs4/homes gss/krb5i(rw,sync)
#
#               *(rw,sync,no_root_squash,no_subtree_check)
```

- Facendo ricerche riguardo questo file di testo, abbiamo infatti scoperto che è questo che regola il comportamento del protocollo con i client, ad esempio permettendogli di leggere e scrivere su file e directory (come indicato dal permesso **rw**). I permessi vengono garantiti dall'ultima riga di testo in quanto le altre sono commentate (e fungono da esempi): commentando anche quest'ultima riga, come da immagine sottostante, rimuoviamo di fatto tutti i privilegi di accesso ai client.

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


- Applichiamo i cambiamenti al server con il comando **sudo exportfs -a**, risolvendo la vulnerabilità.

Conclusioni

Vediamo nella prossima immagine come si presentava il report di Nessus del nostro Metasploitable prima dell'implementazione delle azioni correttive, spostando in particolare l'attenzione sulle criticità in comune con quelle prese in esame all'inizio di questo documento, ossia quelle che ci eravamo prefissati di risolvere, a conferma della loro presenza sulla nostra macchina.

<input type="checkbox"/>	Sev ▼	CVSS ▼	VPR ▼	Name ▲
<input type="checkbox"/>	CRITICAL	10.0 *	5.9	NFS Exported Share Information Disclosure
<input type="checkbox"/>	CRITICAL	10.0		Unix Operating System Unsupported Version Detection
<input type="checkbox"/>	CRITICAL	10.0 *		VNC Server 'password' Password
<input type="checkbox"/>	CRITICAL	9.8	9.0	Apache Tomcat AJP Connector Request Injection (Ghostcat)
<input type="checkbox"/>	CRITICAL	9.8		Bind Shell Backdoor Detection

Di seguito, invece, il report come si presenta dopo le aver implementato le correzioni qui discusse: possiamo notare l'assenza delle vulnerabilità risolte, confermando quindi l'efficacia delle nostre azioni correttive.

<input type="checkbox"/>	Sev ▼	CVSS ▼	VPR ▼	Name ▲
<input type="checkbox"/>	CRITICAL	10.0		Unix Operating System Unsupported Version Detection
<input type="checkbox"/>	CRITICAL	9.8	9.0	Apache Tomcat AJP Connector Request Injection (Ghostcat)
<input type="checkbox"/>	MIXED	 DNS (Multiple Issues)