

## Wprowadzenie do cyberbezpieczeństwa (WCYB)

### Laboratorium 1

#### Zadania do Tematu 1: Kali Linux | Podstawy sieci komputerowych

## Podstawy Basha

### Level 0

Przy użyciu komendy: "`ssh -p 2220 bandit0@bandit.labs.overthewire.org`" i wpisaniu nazwy użytkownika "bandit0", i wpisaniu hasła "bandit0" można dostać się do gry.

### Level 0 -> Level 1

Przy użyciu komendy: "`cat readme`" można dostać hasło do następnego etapu. Należy zakończyć dotychczasowe połączenie SSH i napisać "`ssh -p 2220 bandit1@bandit.labs.overthewire.org`" a następnie wpisać hasło.

### Level 1 -> Level 2

Przy użyciu komendy: "`cat ./-`" można dostać hasło do następnego etapu. Należy zakończyć dotychczasowe połączenie SSH i napisać "`ssh -p 2220 bandit2@bandit.labs.overthewire.org`" a następnie wpisać hasło.

### Level 2 -> Level 3

Przy użyciu komendy: "`cat "spaces in this filename"`" można dostać hasło do następnego etapu. Należy zakończyć dotychczasowe połączenie SSH i napisać "`ssh -p 2220 bandit3@bandit.labs.overthewire.org`" a następnie wpisać hasło.

## Podstawy sieci i analizy ruchu sieciowego

Spis "Podstawy sieci i analizy ruchu sieciowego":

1. Telnetlab
2. Network-basics
3. Routing-basics
4. Pcapanalysis
5. Wireshark-intro
6. Ogólne przemyślenia

## 1. Telnetlab

Ćwiczenie pokazuje użycie klienta telnet w celu uzyskania dostępu do zasobów na serwerze. Laboratorium ma na celu pokazanie podstawowej sieci klient-serwer oraz transmisji haseł w postaci zwykłego tekstu przez sieć przez telnet.

Telnetlab pokazuje również różnice między sesją telnet a sesją SSH. Telnet nie szyfruje pakietów, więc można podejrzewać wpisywanie hasła. Podczas sesji SSH wszystkie dane w pakietach są zaszyfrowane.

Wyszukujemy adres IP serwera przy użyciu komendy **“ifconfig”**. Adres IP serwera jest **“172.20.0.3”**

```
ubuntu@server:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:ac:14:00:03
          inet addr:172.20.0.3  Bcast:172.20.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:49 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:6662 (6.6 KB)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Rozpoczynamy sesję telnet. Używamy komendy **“telnet 172.20.0.3”** by uzyskać dostęp do zasobów serwera. Po podaniu ID użytkownika i hasła mamy dostęp do zasobów serwera. Możemy użyć komendy **“cat filetoview.txt”** by odczytać zawartość pliku **“filetoview.txt”**

```
ubuntu@client:~$ telnet 172.20.0.3
Trying 172.20.0.3...
Connected to 172.20.0.3.
Escape character is '^['.
Ubuntu 16.04.4 LTS
server login: ubuntu
Password:
Last login: Wed Nov  9 17:06:23 UTC 2022 from 172.20.0.2 on pts/3
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.15.0-20-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
ubuntu@server:~$ cat filetoview.txt
# Filename: filetoview.txt
#
# Description: This is a pre-created file for each student (telnet-server) container
#
# This file is modified when container is created
# The string below will be replaced with a keyed hash
My string is: 3dcb6b3bc3d7332f5d0f4810b3472553
ubuntu@server:~$ exit
logout
Connection closed by foreign host.
ubuntu@client:~$
```

Przy pomocy komendy **“exit”** kończymy sesję telnet. Na serwerze zaczynamy nasłuchiwanie ruchu sieciowego przy użyciu komendy **“sudo tcpdump -i eth0 -X tcp”**

```
ubuntu@server:~$ sudo tcpdump -i eth0 -X tcp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

Jeszcze raz zaczynamy sesję telnet

```
ubuntu@client:~$ telnet 172.20.0.3
Trying 172.20.0.3...
Connected to 172.20.0.3.
Escape character is '^['.
Ubuntu 16.04.4 LTS
server login: ubuntu
Password:

Login incorrect
server login:
Login timed out after 60 seconds.
Connection closed by foreign host.
ubuntu@client:~$ ssh 172.20.0.3
ubuntu@172.20.0.3's password:
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.15.0-20-generic x86_64)
```

Wraz z nasłuchiowaniem ruchu sieciowego jesteśmy w stanie odczytać wpisywane hasło. Hasło “mydoghasfleas” zostało podzielone na litery i każdą literę można znaleźć w pakiecie ack.

```
0x0020: 8010 00e5 5854 0000 0101 080a 507d ac9b ....XT.....P]..
0x0030: 820b 58ad ..X.
12:48:41.044673 IP telnetlab.client.student.some_network.46632 > server.telnet: Flags [P.], seq 118:119, ack 114,
win 229, options [nop,nop,TS val 1350414669 ecr 2181781677], length 1
0x0000: 4510 0035 7071 4000 4006 7214 ac14 0002 E..5pq@.@.r.....
0x0010: ac14 0003 b628 0017 836e 655c 9c42 1f72 .....(..ne\B.r
0x0020: 8018 00e5 5855 0000 0101 080a 507d b14d ....XU.....P].M
0x0030: 820b 58ad 6d ..X.m
12:48:41.088651 IP server.telnet > telnetlab.client.student.some_network.46632: Flags [.], ack 119, win 227, opti
ons [nop,nop,TS val 2181782923 ecr 1350414669], length 0
0x0000: 4510 0034 0bc4 4000 4006 d6c2 ac14 0003 E..4..@.@.....
0x0010: ac14 0002 0017 b628 9c42 1f72 836e 655d .....(B.r.ne]
0x0020: 8010 00e3 5854 0000 0101 080a 820b 5d8b ....XT.....].
0x0030: 507d b14d P].M
12:48:47.105388 IP telnetlab.client.student.some_network.46632 > server.telnet: Flags [P.], seq 119:120, ack 114,
win 229, options [nop,nop,TS val 2181782932 ecr 2181782923], length 1
0x0000: 4510 0035 7072 4000 4006 7213 ac14 0002 E..5pr@.@.r.....
0x0010: ac14 0003 b628 0017 836e 655d 9c42 1f72 .....(..ne\B.r
0x0020: 8018 00e5 5855 0000 0101 080a 507d c8fc ....XU.....P]..
0x0030: 820b 5d8b 79 ..].y
12:48:47.105402 IP server.telnet > telnetlab.client.student.some_network.46632: Flags [.], ack 120, win 227, opti
ons [nop,nop,TS val 2181788943 ecr 1350420732], length 0
0x0000: 4510 0034 0bc5 4000 4006 d6c1 ac14 0003 E..4..@.@.....
0x0010: ac14 0002 0017 b628 9c42 1f72 836e 655e .....(B.r.ne^
0x0020: 8010 00e3 5854 0000 0101 080a 820b 750f ....XT.....u.
0x0030: 507d c8fc P]..
12:48:48.103790 IP telnetlab.client.student.some_network.46632 > server.telnet: Flags [P.], seq 120:121, ack 114,
win 229, options [nop,nop,TS val 1350421731 ecr 2181788943], length 1
0x0000: 4510 0035 7073 4000 4006 7212 ac14 0002 E..5ps@.@.r.....
0x0010: ac14 0003 b628 0017 836e 655e 9c42 1f72 .....(..ne^B.r
0x0020: 8018 00e5 5855 0000 0101 080a 507d cce3 ....XU.....P]..
0x0030: 820b 750f 64 ..u.d
12:48:48.103833 IP server.telnet > telnetlab.client.student.some_network.46632: Flags [.], ack 121, win 227, opti
ons [nop,nop,TS val 2181789941 ecr 1350421731], length 0
0x0000: 4510 0034 0bc6 4000 4006 d6c0 ac14 0003 E..4..@.@.....
0x0010: ac14 0002 0017 b628 9c42 1f72 836e 655f .....(B.r.ne
0x0020: 8010 00e3 5854 0000 0101 080a 820b 78f5 ....XT.....x.
0x0030: 507d cce3 P]..
12:48:48.770673 IP telnetlab.client.student.some_network.46632 > server.telnet: Flags [P.], seq 121:122, ack 114,
win 229, options [nop,nop,TS val 1350422399 ecr 2181789941], length 1
0x0000: 4510 0035 7074 4000 4006 7211 ac14 0002 E..5pt@.@.r.....
0x0010: ac14 0003 b628 0017 836e 655f 9c42 1f72 .....(..ne^B.r
0x0020: 8018 00e5 5855 0000 0101 080a 507d cf7f ....XU.....P]..
0x0030: 820b 78f5 6f ..x.o
12:48:48.770686 IP server.telnet > telnetlab.client.student.some_network.46632: Flags [.], ack 122, win 227, opti
ons [nop,nop,TS val 2181790609 ecr 1350422399], length 0
0x0000: 4510 0034 0bc7 4000 4006 d6bf ac14 0003 E..4..@.@.....
0x0010: ac14 0002 0017 b628 9c42 1f72 836e 6560 .....(B.r.neb
0x0020: 8010 00e3 5854 0000 0101 080a 820b 7dc4 ....XT.....}.
0x0030: 507d d1b2 P]..
12:48:49.333305 IP telnetlab.client.student.some_network.46632 > server.telnet: Flags [P.], seq 122:123, ack 114,
win 229, options [nop,nop,TS val 1350422962 ecr 2181790609], length 1
0x0000: 4510 0035 7075 4000 4006 7210 ac14 0002 E..5pu@.@.r.....
0x0010: ac14 0003 b628 0017 836e 6560 9c42 1f72 .....(..nea^B.r
0x0020: 8018 00e5 5855 0000 0101 080a 507d d1b2 ....XU.....P]..
0x0030: 820b 7b91 67 ..{.g
12:48:49.333318 IP server.telnet > telnetlab.client.student.some_network.46632: Flags [.], ack 123, win 227, opti
ons [nop,nop,TS val 2181791172 ecr 1350422962], length 0
0x0000: 4510 0034 0bc8 4000 4006 d6be ac14 0003 E..4..@.@.....
0x0010: ac14 0002 0017 b628 9c42 1f72 836e 6561 .....(B.r.nea
0x0020: 8010 00e3 5854 0000 0101 080a 820b 7dc4 ....XT.....}.
0x0030: 507d d1b2 P]..
12:48:50.158358 IP telnetlab.client.student.some_network.46632 > server.telnet: Flags [P.], seq 123:124, ack 114,
win 229, options [nop,nop,TS val 1350423787 ecr 2181791172], length 1
0x0000: 4510 0035 7076 4000 4006 720f ac14 0002 E..5pv@.@.r.....
0x0010: ac14 0003 b628 0017 836e 6561 9c42 1f72 .....(..neaB.r
0x0020: 8018 00e5 5855 0000 0101 080a 507d d4eb ....XU.....P]..
0x0030: 820b 7dc4 68 ..}.h
12:48:50.158371 IP server.telnet > telnetlab.client.student.some_network.46632: Flags [.], ack 124, win 227, opti
ons [nop,nop,TS val 2181791907 ecr 1350423787], length 0
0x0000: 4510 0034 0bc9 4000 4006 d6bd ac14 0003 E..4..@.@.....
0x0010: ac14 0002 0017 b628 9c42 1f72 836e 6562 .....(B.r.neb
0x0020: 8010 00e3 5854 0000 0101 080a 820b 80fd ....XT.....}.
0x0030: 507d d4eb P]..
12:48:50.624780 IP telnetlab.client.student.some_network.46632 > server.telnet: Flags [P.], seq 124:125, ack 114,
win 229, options [nop,nop,TS val 1350424253 ecr 2181791907], length 1
0x0000: 4510 0035 7077 4000 4006 720e ac14 0002 E..5pw@.@.r.....
0x0010: ac14 0003 b628 0017 836e 6562 9c42 1f72 .....(..nebB.r
0x0020: 8018 00e5 5855 0000 0101 080a 507d d6bd ....XU.....P]..
0x0030: 820b 80fd 61 ....a
12:48:50.624799 IP server.telnet > telnetlab.client.student.some_network.46632: Flags [.], ack 125, win 227, opti
ons [nop,nop,TS val 2181792464 ecr 1350424253], length 0
0x0000: 4510 0034 0bca 4000 4006 d6bc ac14 0003 E..4..@.@.....
0x0010: ac14 0002 0017 b628 9c42 1f72 836e 6563 .....(B.r.nec
0x0020: 8010 00e3 5854 0000 0101 080a 820b 82d0 ....XT.....}.
0x0030: 507d d6bd P]..
12:48:51.120371 IP telnetlab.client.student.some_network.46632 > server.telnet: Flags [P.], seq 125:126, ack 114,
win 229, options [nop,nop,TS val 1350424749 ecr 2181792464], length 1
0x0000: 4510 0035 7078 4000 4006 720d ac14 0002 E..5px@.@.r.....
0x0010: ac14 0003 b628 0017 836e 6563 9c42 1f72 .....(..necB.r
0x0020: 8018 00e5 5855 0000 0101 080a 507d d8ad ....XU.....P]..
0x0030: 820b 82d0 73 ....s
12:48:51.120386 IP server.telnet > telnetlab.client.student.some_network.46632: Flags [.], ack 126, win 227, opti
ons [nop,nop,TS val 2181792960 ecr 1350424749], length 0
0x0000: 4510 0034 0bcd 4000 4006 d6bb ac14 0003 E..4..@.@.....
0x0010: ac14 0002 0017 b628 9c42 1f72 836e 6564 .....(B.r.ned
0x0020: 8010 00e3 5854 0000 0101 080a 820b 84c0 ....XT.....}.
0x0030: 507d d8ad P]..
```

```

0x0020: 8010 00e3 5854 0000 0101 080a 820b 84c0 ....XT.....
0x0030: 507d d8ad P]..
12:48:51.598660 IP telnetlab.client.student.some_network.46632 > server.telnet: Flags [P.], seq 126:127, ack 114,
win 229, options [nop,nop,TS val 1350425228 ecr 2181792960], length 1
0x0000: 4510 0035 7079 4000 4006 720c ac14 0002 E..5py@.@.r.....
0x0010: ac14 0003 b628 0017 836e 6564 9c42 1f72 ....(...ned.B.r
0x0020: 8018 00e5 5855 0000 0101 080a 507d da8c ....XU.....P]..
0x0030: 820b 84c0 66 ....f
12:48:51.598673 IP server.telnet > telnetlab.client.student.some_network.46632: Flags [.], ack 127, win 227, opti
ons [nop,nop,TS val 2181793438 ecr 1350425228], length 0
0x0000: 4510 0034 0bcc 4000 4006 d6ba ac14 0003 E..4..@.@.....
0x0010: ac14 0002 0017 b628 9c42 1f72 836e 6565 ....(...B.r.nee
0x0020: 8010 00e3 5854 0000 0101 080a 820b 869e ....XT.....
0x0030: 507d da8c P]..
12:48:51.981727 IP telnetlab.client.student.some_network.46632 > server.telnet: Flags [P.], seq 127:128, ack 114,
win 229, options [nop,nop,TS val 1350425611 ecr 2181793438], length 1
0x0000: 4510 0035 707a 4000 4006 720b ac14 0002 E..5pz@.@.r.....
0x0010: ac14 0003 b628 0017 836e 6565 9c42 1f72 ....(...nee.B.r
0x0020: 8018 00e5 5855 0000 0101 080a 507d dc0b ....XU.....P]..
0x0030: 820b 869e 6c ....l
12:48:51.981739 IP server.telnet > telnetlab.client.student.some_network.46632: Flags [.], ack 128, win 227, opti
ons [nop,nop,TS val 2181793821 ecr 1350425611], length 0
0x0000: 4510 0034 0bcd 4000 4006 d6b9 ac14 0003 E..4..@.@.....
0x0010: ac14 0002 0017 b628 9c42 1f72 836e 6566 ....(...B.r.nef
0x0020: 8010 00e3 5854 0000 0101 080a 820b 881d ....XT.....
0x0030: 507d dc0b P]..
12:48:52.376871 IP telnetlab.client.student.some_network.46632 > server.telnet: Flags [P.], seq 128:129, ack 114,
win 229, options [nop,nop,TS val 1350426007 ecr 2181793821], length 1
0x0000: 4510 0035 707b 4000 4006 720a ac14 0002 E..5p[.@.@.r.....
0x0010: ac14 0003 b628 0017 836e 6566 9c42 1f72 ....(...nef.B.r
0x0020: 8018 00e5 5855 0000 0101 080a 507d dd97 ....XU.....P]..
0x0030: 820b 881d 65 ....e
12:48:52.376884 IP server.telnet > telnetlab.client.student.some_network.46632: Flags [.], ack 129, win 227, opti
ons [nop,nop,TS val 2181794217 ecr 1350426007], length 0
0x0000: 4510 0034 0bce 4000 4006 d6b8 ac14 0003 E..4..@.@.....
0x0010: ac14 0002 0017 b628 9c42 1f72 836e 6567 ....(...B.r.neg
0x0020: 8010 00e3 5854 0000 0101 080a 820b 89a9 ....XT.....
0x0030: 507d dd97 P]..
12:48:52.859377 IP telnetlab.client.student.some_network.46632 > server.telnet: Flags [P.], seq 129:130, ack 114,
win 229, options [nop,nop,TS val 1350426489 ecr 2181794217], length 1
0x0000: 4510 0035 707c 4000 4006 7209 ac14 0002 E..5p[.@.@.r.....
0x0010: ac14 0003 b628 0017 836e 6567 9c42 1f72 ....(...neg.B.r
0x0020: 8018 00e5 5855 0000 0101 080a 507d df79 ....XU.....P].y
0x0030: 820b 89a9 61 ....a
12:48:52.859391 IP server.telnet > telnetlab.client.student.some_network.46632: Flags [.], ack 130, win 227, opti
ons [nop,nop,TS val 2181794699 ecr 1350426489], length 0
0x0000: 4510 0034 0bcf 4000 4006 d6b7 ac14 0003 E..4..@.@.....
0x0010: ac14 0002 0017 b628 9c42 1f72 836e 6568 ....(...B.r.neh
0x0020: 8010 00e3 5854 0000 0101 080a 820b 8b8b ....XT.....
0x0030: 507d df79 P].y
12:49:07.167808 IP telnetlab.client.student.some_network.46632 > server.telnet: Flags [P.], seq 130:131, ack 114,
win 229, options [nop,nop,TS val 1350440805 ecr 2181794699], length 1
0x0000: 4510 0035 707d 4000 4006 7208 ac14 0002 E..5p[.@.@.r.....
0x0010: ac14 0003 b628 0017 836e 6568 9c42 1f72 ....(...neh.B.r
0x0020: 8018 00e5 5855 0000 0101 080a 507e 1765 ....XU.....P~.e
0x0030: 820b 8b8b 73 ....s

```

Kończymy sesję telnet. Teraz użyjemy komendy “**ssh 172.20.0.3**” w celu uzyskania zasobów serwera. Tak jak poprzednio użyjemy komendy “**cat filetoview.txt**” by odczytać zawartość pliku “filetoview.txt”.

```

ubuntu@client:~$ ssh 172.20.0.3
ubuntu@172.20.0.3's password:
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.15.0-20-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
Last login: Fri Nov 11 12:46:03 2022 from telnetlab.client.student.some_network
ubuntu@server:~$ cat filetoview.txt
# Filename: filetoview.txt
#
# Description: This is a pre-created file for each student (telnet-server) container

# This file is modified when container is created
# The string below will be replaced with a keyed hash
My string is: 3dcb6b3bc3d7332f5d0f4810b3472553
ubuntu@server:~$

```

Przy ssh nie możemy odczytać danych w pakietach bo wszystkie dane w pakietach zostały zaszyfrowane.

```

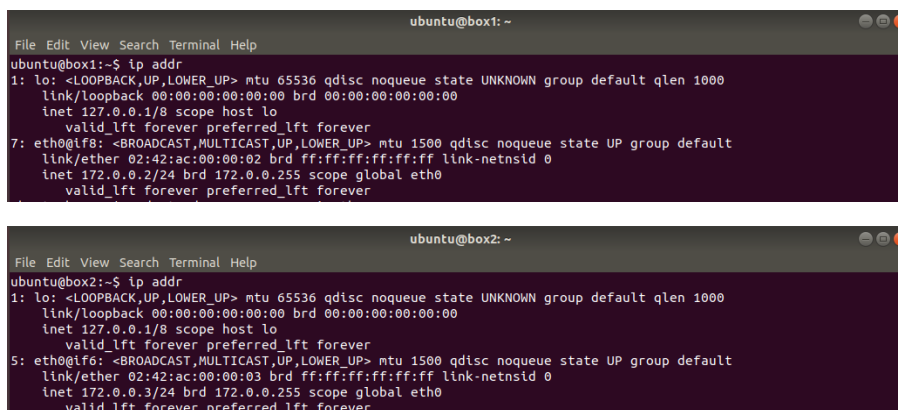
0x0050: 6753 d33e 6e80 1e74 gS.>n..t
15:00:10.767878 IP server.ssh > telnetlab.client.student.some_network.49216: Flags [P.], seq 3594:3630, ack 2838,
win 291, options [nop,nop,TS val 2425909904 ecr 2219212773], length 36
0x0000: 4510 0058 6797 4000 4006 7acb ac14 0003 E..Xg.@.z.....
0x0010: ac14 0002 0016 c040 25d7 cb61 70b8 5f57 .....@%.ap..W
0x0020: 8018 0123 5878 0000 0101 080a 9098 7290 ...#Xx.....f.
0x0030: 8446 7fe5 9fe6 6094 2512 7abc cd08 fc19 .F.....%.z.....
0x0040: 27b7 3d7e 44b8 22b8 941e b093 9c35 dd57 ',-D,".....S.W
0x0050: 137d ac46 3cd6 2c1c .).F<.,.
15:00:10.768031 IP telnetlab.client.student.some_network.49216 > server.ssh: Flags [P.], seq 2838:2874, ack 3630, option
s [nop,nop,TS val 2219212773 ecr 2425909904], length 0
0x0000: 4510 0034 8ea5 4000 4006 53e1 ac14 0002 E..4..@.S.....
0x0010: ac14 0003 c040 0016 70b8 5f57 25d7 cb85 .....@.p..WX...
0x0020: 8010 0122 5854 0000 0101 080a 8446 7fe5 ...XT.....F...
0x0030: 9098 7290 .....f.
15:00:11.007607 IP telnetlab.client.student.some_network.49216 > server.ssh: Flags [P.], seq 2838:2874, ack 3630,
win 290, options [nop,nop,TS val 2219213013 ecr 2425909904], length 36
0x0000: 4510 0058 8eae 4000 4006 53bc ac14 0002 E..X..@.S.....
0x0010: ac14 0003 c040 0016 70b8 5f57 25d7 cb85 .....@.p..WX...
0x0020: 8010 0122 5878 0000 0101 080a 8446 80d5 ...Xx.....F...
0x0030: 9098 7290 9117 0a23 c03a b712 fddc 8f62 .f...#.:....b
0x0040: 9f95 034e 2500 2a25 62f2 d777 7043 3320 ...N%.*%b..wpC3.
0x0050: 4040 996d 03df ac55 @!m..U
15:00:11.008138 IP server.ssh > telnetlab.client.student.some_network.49216: Flags [P.], seq 3630:3666, ack 2874,
win 291, options [nop,nop,TS val 2425910145 ecr 2219213013], length 36
0x0000: 4510 0058 8eae 4000 4006 53bc ac14 0002 E..X..@.S.....
0x0010: ac14 0003 c040 0016 70b8 5f57 25d7 cb85 .....@.p..WX...
0x0020: 8010 0122 5878 0000 0101 080a 8446 80d5 ...Xx.....F...
0x0030: 9098 7290 9117 0a23 c03a b712 fddc 8f62 .f...#.:....b
0x0040: 9f95 034e 2500 2a25 62f2 d777 7043 3320 ...N%.*%b..wpC3.
0x0050: 4040 996d 03df ac55 @!m..U
15:00:11.029105 IP telnetlab.client.student.some_network.49216 > server.ssh: Flags [P.], seq 3630:3666, ack 2874,
win 291, options [nop,nop,TS val 2425910145 ecr 2219213013], length 36
0x0000: 4510 0058 8eae 4000 4006 53bc ac14 0002 E..X..@.S.....
0x0010: ac14 0003 c040 0016 70b8 5f57 25d7 cb85 .....@.p..WX...
0x0020: 8010 0122 5878 0000 0101 080a 8446 80d5 ...Xx.....F...
0x0030: 9098 7290 9117 0a23 c03a b712 fddc 8f62 .f...#.:....b
0x0040: 9f95 034e 2500 2a25 62f2 d777 7043 3320 ...N%.*%b..wpC3.
0x0050: 4040 996d 03df ac55 @!m..U
15:00:11.029847 IP server.ssh > telnetlab.client.student.some_network.49216: Flags [P.], seq 3974:4050, ack 2874,
win 291, options [nop,nop,TS val 2425910166 ecr 2219213035], length 76
0x0000: 4510 0080 679a 4000 4006 7aa0 ac14 0003 E...g.@.z.....
0x0010: ac14 0002 0016 c040 25d7 ccdd 70b8 5f7b .....@%.p...{
0x0020: 8018 0123 58a0 0000 0101 080a 9098 7396 ...#X.....S.
0x0030: 8446 80eb dec8 9511 07d8 e8f5 d31f d149 .F.....I
0x0040: 0eef 6778 81e5 8161 ab4d a54e e438 4c84 ..gx...a.M.N.8L.
0x0050: 1a40 3c79 12f8 90bd 44d8 6082 5d7f 9cf1 .<y....D.'.)...
0x0060: de05 f96b 9631 89db e51a 1204 6498 fce6 ...k.....d...
0x0070: 8e00 c6f4 45bd 7f69 6870 7eba 78a4 34a7 ...E..thp-.x.4.
15:00:11.029913 IP telnetlab.client.student.some_network.49216 > server.ssh: Flags [P.], seq 3974:4050, ack 2874,
win 291, options [nop,nop,TS val 2425910166 ecr 2219213035], length 76
0x0000: 4510 0080 679a 4000 4006 7aa0 ac14 0003 E...g.@.z.....
0x0010: ac14 0002 0016 c040 25d7 ccdd 70b8 5f7b .....@%.p...{
0x0020: 8018 0123 58a0 0000 0101 080a 9098 7396 ...#X.....S.
0x0030: 8446 80eb dec8 9511 07d8 e8f5 d31f d149 .F.....I
0x0040: 0eef 6778 81e5 8161 ab4d a54e e438 4c84 ..gx...a.M.N.8L.
0x0050: 1a40 3c79 12f8 90bd 44d8 6082 5d7f 9cf1 .<y....D.'.)...
0x0060: de05 f96b 9631 89db e51a 1204 6498 fce6 ...k.....d...
0x0070: 8e00 c6f4 45bd 7f69 6870 7eba 78a4 34a7 ...E..thp-.x.4.

```

## 2. Network-basics

Ćwiczenie pokazuje podstawowe koncepcje sieciowe w środowisku Linux. Ćwiczenie krótko wyjaśnia do czego jest stosowany protokół ARP i pokazuje uzgadnianie trój-etapowe w sesji TCP. Skrypt do network-basics wyjaśnia też jak TCP zapewnia, że pakiety podczas sesji nie zostaną utracone i są ułożone we właściwej kolejności w celu dostarczenia do aplikacji.

Komenda “**ip addr**” wyświetla interfejs sieci



```

ubuntu@box1:~$ ip addr
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
7: eth0@if8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:00:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.0.0.2/24 brd 172.0.0.255 scope global eth0
        valid_lft forever preferred_lft forever

ubuntu@box2:~$ ip addr
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
5: eth0@if6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:00:00:03 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.0.0.3/24 brd 172.0.0.255 scope global eth0
        valid_lft forever preferred_lft forever

```

Gdy użyjemy komendy “**arp -a**” na box2 to nic się nie pokazuje, gdyż tablica ARP jest pusta.

Na box1 zaczynamy obserwowanie ruchu sieciowego

```

ubuntu@box1:~$ sudo tcpdump -vv -n -e -i eth0
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes

```

Na box2 pingujemy box1

```
ubuntu@box2:~$ ping 172.0.0.2 -c 2
PING 172.0.0.2 (172.0.0.2) 56(84) bytes of data:
64 bytes from 172.0.0.2: icmp_seq=1 ttl=64 time=0.260 ms
64 bytes from 172.0.0.2: icmp_seq=2 ttl=64 time=0.060 ms

--- 172.0.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1014ms
rtt min/avg/max/mdev = 0.060/0.160/0.260/0.100 ms
```

Na box 1 widać pakiety. ARP request z box2 z pytaniem o adres MAC osoby obsługującej ruch do adresu IP box1. Zauważymy, że docelowy adres MAC w żądaniu ARP to ff:ff:ff:ff:ff:ff, co jest wiadomością rozgłoszeniową widzianą przez każdy interfejs Ethernet w sieci LAN. Zobaczymy ARP reply z box1. Gdy oba urządzenia skojarzą adresy IP z adresami MAC, mogą wymieniać między sobą pakiety warstwy sieciowej. W tym przypadku są to pakiety ICMP. Pierwszy pakiet to ICMP echo request z box2 do box1. To jest „ping”. Następny pakiet to ICMP echo request z box1

```
ubuntu@box1:~$ sudo tcpdump -vv -n -e -i eth0
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
14:17:28.416762 02:42:ac:00:00:03 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: Ethernet (len 6), IPv4 (len 4), Request who-has 172.0.0.2 tell 172.0.0.3, length 28
14:17:28.416778 02:42:ac:00:00:02 > 02:42:ac:00:00:03, ethertype ARP (0x0806), length 42: Ethernet (len 6), IPv4 (len 4), Reply 172.0.0.2 is-at 02:42:ac:00:00:02, length 28
14:17:28.416794 02:42:ac:00:00:03 > 02:42:ac:00:00:02, ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 52629, offset 0, flags [DF], proto ICMP (1), length 84)
    172.0.0.3 > 172.0.0.2: ICMP echo request, id 1, seq 1, length 64
14:17:28.416936 02:42:ac:00:00:02 > 02:42:ac:00:00:03, ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 21265, offset 0, flags [none], proto ICMP (1), length 84)
    172.0.0.2 > 172.0.0.3: ICMP echo reply, id 1, seq 1, length 64
14:17:29.431074 02:42:ac:00:00:03 > 02:42:ac:00:00:02, ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 52764, offset 0, flags [DF], proto ICMP (1), length 84)
    172.0.0.3 > 172.0.0.2: ICMP echo request, id 1, seq 2, length 64
14:17:29.431091 02:42:ac:00:00:02 > 02:42:ac:00:00:03, ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 21487, offset 0, flags [none], proto ICMP (1), length 84)
    172.0.0.2 > 172.0.0.3: ICMP echo reply, id 1, seq 2, length 64
```

Teraz użyjemy komendy “arp -a” w obu boxach, aby wyświetlić bieżącą zawartość tablicy ARP. Te wpisy ARP umożliwiają dwóm urządzeniom adresowanie się do siebie bez powtarzania request/response ARP.

```
ubuntu@box1:~$ arp -a
? (172.0.0.3) at 02:42:ac:00:00:03 [ether] on eth0
ubuntu@box1:~$
```

```
ubuntu@box2:~$ arp -a
? (172.0.0.2) at 02:42:ac:00:00:02 [ether] on eth0
ubuntu@box2:~$
```

Teraz chcemy zobaczyć jak wygląda “triple way handshake”. Użyjemy komendy “sudo tcpdump -vv -n -i eth0” na box1. Następnie na box2 zaczniemy sesję przy użyciu komendy “ssh 172.0.0.2”

```
ubuntu@box1:~$ sudo tcpdump -vv -n -i eth0
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

“triple way handshake”

```
14:20:32.053303 IP (tos 0x0, ttl 64, id 1118, offset 0, flags [DF], proto TCP (6), length 60)
    172.0.0.3.44628 > 172.0.0.2.22: Flags [S], cksum 0x5834 (incorrect -> 0xbb8b), seq 944564969, win 29200, options [mss 1460,sackOK,TS val 3219206364 ecr 0,nop,wscale 7], length 0
14:20:32.053368 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 60)
    172.0.0.2.22 > 172.0.0.3.44628: Flags [S.], cksum 0x5834 (incorrect -> 0x108f), seq 1354257294, ack 944564970, win 28960, options [mss 1460,sackOK,TS val 416673471 ecr 3219206364,nop,wscale 7], length 0
14:20:32.053386 IP (tos 0x0, ttl 64, id 1119, offset 0, flags [DF], proto TCP (6), length 52)
    172.0.0.3.44628 > 172.0.0.2.22: Flags [.], cksum 0x582c (incorrect -> 0xaf96), seq 1, ack 1, win 229, options [nop,nop,TS val 3219206364 ecr 416673471], length 0
```

### 3. Routing-basics

Lab routing-basics przedstawia ogólne wiadomości dotyczące routingu.

Lista adresów IP:

-ws1 IP: 192.168.1.1

-ws2 IP: 192.168.2.1

-ws3 IP: 192.168.2.2



## Routing wewnętrzny

Do każdej stacji roboczej (ws1, ws2, ws3) wpisujemy komendę “**route -n**”

```
harry@ws1:~$ route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
0.0.0.0 192.168.1.10 0.0.0.0 UG 0 0 0 eth0
192.168.1.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0

harry@ws1:~$

mary@ws2:~$ route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
0.0.0.0 192.168.2.10 0.0.0.0 UG 0 0 0 eth0
192.168.2.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0

mary@ws2:~$

larry@ws3:~$ route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
192.168.2.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0

larry@ws3:~$
```

Zauważmy, że ws1 i ws2 zawierają wpisy w tablicy routingu, które określają bramę jako bramę domyślną. To pozwala ws1 i ws2 na wzajemne adresowanie.

Możemy pingować ws2 z ws1.

```
harry@ws1:~$ ping 192.168.2.1
PING 192.168.2.1 (192.168.2.1) 56(84) bytes of data.
64 bytes from 192.168.2.1: icmp_seq=1 ttl=63 time=0.489 ms
64 bytes from 192.168.2.1: icmp_seq=2 ttl=63 time=0.073 ms
64 bytes from 192.168.2.1: icmp_seq=3 ttl=63 time=0.075 ms
64 bytes from 192.168.2.1: icmp_seq=4 ttl=63 time=0.082 ms
^C
harry@ws1:~$
--- 192.168.2.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3079ms
rtt min/avg/max/mdev = 0.073/0.179/0.489/0.178 ms
n
^C
harry@ws1:~$
```

Nie możemy pingować ws3 z ws1, gdyż ws3 nie ma wpisu w tablicy routingu definiującego, co zrobić z ruchem, który nie jest przeznaczony dla sieci LAN bezpośrednio podłączonej do ws3

```
harry@ws1:~$ ping 192.168.2.2
PING 192.168.2.2 (192.168.2.2) 56(84) bytes of data.
^C
harry@ws1:~$
--- 192.168.2.2 ping statistics ---
48 packets transmitted, 0 received, 100% packet loss, time 48218ms
n
^C
harry@ws1:~$
```

W ws3 zdefiniujemy komponent bramy jako bramę domyślną za pomocą polecenia “**sudo route add default gw 192.168.2.10**”

```
larry@ws3:~$ sudo route add default gw 192.168.2.10
larry@ws3:~$
```

Teraz możemy pingować ws3 z ws1

```
harry@ws1:~$ ping 192.168.2.2
PING 192.168.2.2 (192.168.2.2) 56(84) bytes of data.
64 bytes from 192.168.2.2: icmp_seq=1 ttl=63 time=0.185 ms
64 bytes from 192.168.2.2: icmp_seq=2 ttl=63 time=0.076 ms
64 bytes from 192.168.2.2: icmp_seq=3 ttl=63 time=0.073 ms
64 bytes from 192.168.2.2: icmp_seq=4 ttl=63 time=0.074 ms
^C
harry@ws1:~$
--- 192.168.2.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3079ms
rtt min/avg/max/mdev = 0.073/0.102/0.185/0.047 ms
n
^C
harry@ws1:~$
```

## Routing do Internetu

Możemy pingować [www.google.com](http://www.google.com) z ws2

```

mary@ws2:~$ ping www.google.com
PING www.google.com (142.250.203.196) 56(84) bytes of data.
64 bytes from waw02s22-in-f4.1e100.net (142.250.203.196): icmp_seq=1 ttl=53 time=171 ms
64 bytes from waw02s22-in-f4.1e100.net (142.250.203.196): icmp_seq=2 ttl=53 time=14.6 ms
64 bytes from waw02s22-in-f4.1e100.net (142.250.203.196): icmp_seq=3 ttl=53 time=13.2 ms
64 bytes from waw02s22-in-f4.1e100.net (142.250.203.196): icmp_seq=4 ttl=53 time=16.2 ms
^C
--- www.google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 13.212/53.708/170.847/67.638 ms
mary@ws2:~$ n

```

Nie możemy pingować [www.google.com](http://www.google.com) z ws3, ponieważ ws3 nie ma zdefiniowanego systemu nazwy domen (DNS). Gdybyśmy skonfigurowali plik `/etc/resolv.conf` moglibyśmy pingować [www.google.com](http://www.google.com) z ws3.

```

larry@ws3:~$ ping www.google.com
ping: www.google.com: Temporary failure in name resolution
larry@ws3:~$ 

```

### Korzystanie z translacji adresów sieciowych (NAT)

Komenda “**sudo iptables -L -v -t nat**” pokazuje przetłumaczone adresy źródłowe dla całego ruchu przeznaczony dla naszego zewnętrznego interfejsu sieciowego

```

harry@ws1:~$ sudo iptables -L -v -t nat
Chain PREROUTING (policy ACCEPT 9 packets, 866 bytes)
pkts bytes target      prot opt in      out     source      destination

Chain INPUT (policy ACCEPT 1 packets, 84 bytes)
pkts bytes target      prot opt in      out     source      destination

Chain OUTPUT (policy ACCEPT 4 packets, 336 bytes)
pkts bytes target      prot opt in      out     source      destination
0      0 DOCKER_OUTPUT all -- any    any     anywhere    127.0.0.11

Chain POSTROUTING (policy ACCEPT 4 packets, 336 bytes)
pkts bytes target      prot opt in      out     source      destination
0      0 DOCKER_POSTROUTING all -- any    any     anywhere    127.0.0.11

Chain DOCKER_OUTPUT (1 references)
pkts bytes target      prot opt in      out     source      destination
0      0 DNAT        tcp -- any    any     anywhere    127.0.0.11      tcp dpt:domain to:127.0.0.11:46711
0.11:46711
0      0 DNAT        udp -- any    any     anywhere    127.0.0.11      udp dpt:domain to:127.0.0.11:60220
0.11:60220

Chain DOCKER_POSTROUTING (1 references)
pkts bytes target      prot opt in      out     source      destination
0      0 SNAT        tcp -- any    any     127.0.0.11  anywhere    tcp spt:46711 to::53
0      0 SNAT        udp -- any    any     127.0.0.11  anywhere    udp spt:60220 to::53
harry@ws1:~$ 

```

Komenda “**sudo iptables -L -v**” pokazuje, czy przekazujemy ruch otrzymany z dwóch sieci LAN.

```

harry@ws1:~$ sudo iptables -L -v
Chain INPUT (policy ACCEPT 27 packets, 2178 bytes)
pkts bytes target      prot opt in      out     source      destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source      destination

Chain OUTPUT (policy ACCEPT 75 packets, 6210 bytes)
pkts bytes target      prot opt in      out     source      destination
harry@ws1:~$ 

```

## 4. Pcapanalysis

Lab pcapanalysis pokazuje przykładowe użycia komendy “**tshark**”.

Komenda “**man tshark**” pokazuje co można zrobić z tshark



```
TSHARK(1)                                The Wireshark Network Analyzer                                TSHARK(1)

NAME
  tshark - Dump and analyze network traffic

SYNOPSIS
  tshark [ -2 ] [ -a <capture autopstop condition> ] ... [ -b <capture ring buffer option> ] ...
  [ -B <capture buffer size> ] [ -c <capture packet count> ] [ -C <configuration profile> ]
  [ -d <layer type>==<selector>,<decode-as protocol> ] [ -D ] [ -e <field> ] [ -E <field print option> ]
  [ -f <capture filter> ] [ -F <file format> ] [ -g ] [ -h ] [ -H <input hosts file> ]
  [ -i <capture interface> ] [ -I ] [ -K <keytab> ] [ -l ] [ -L ] [ -n ] [ -N <name resolving flags> ]
  [ -o <preference setting> ] ... [ -O <protocols> ] [ -p ] [ -P ] [ -q ] [ -Q ] [ -r <infile> ]
  [ -R <read filter> ] [ -s <capture snaplen> ] [ -S <separator> ] [ -t a|ad|adod|d|dd|e|r|u|ud|udod ]
  [ -T fields|pdnl|ps|psml|text ] [ -u <seconds type> ] [ -v ] [ -V ] [ -w <outfile> ] [ - ]
  [ -W <file format option> ] [ -X ] [ -X <extension option> ] [ -y <capture link type> ]
  [ -Y <display filter> ] [ -z <statistics> ] [ --capture-comment <comment> ] [ <capture filter> ]

  tshark -G [ <report type> ]

DESCRIPTION
  TShark is a network protocol analyzer. It lets you capture packet data from a live network, or read
  packets from a previously saved capture file, either printing a decoded form of those packets to the
  standard output or writing the packets to a file. TShark's native capture file format is pcap format,
  which is also the format used by tcpdump and various other tools.
  Manual page tshark(1) line 1 (press h for help or q to quit)
```

Komenda “**tshark -T fields -e frame.number -e frame.time -e telnet.data -r telnet.pcap**” wyświetla określone pola z pliku telnet.pcap

```
ubuntu@pcapanalysis:~$ tshark -T fields -e frame.number -e frame.time -e telnet.data -r telnet.pcap
1   Sep 15, 2017 16:41:30.515574000 UTC
2   Sep 15, 2017 16:41:30.515582000 UTC
3   Sep 15, 2017 16:41:30.538873000 UTC
4   Sep 15, 2017 16:41:30.538990000 UTC
5   Sep 15, 2017 16:41:30.539635000 UTC
6   Sep 15, 2017 16:41:30.544517000 UTC
7   Sep 15, 2017 16:41:30.549115000 UTC
8   Sep 15, 2017 16:41:30.558409000 UTC
9   Sep 15, 2017 16:41:30.564036000 UTC
10  Sep 15, 2017 16:41:30.601214000 UTC
11  Sep 15, 2017 16:41:30.601262000 UTC
12  Sep 15, 2017 16:41:30.601268000 UTC
13  Sep 15, 2017 16:41:30.601397000 UTC
14  Sep 15, 2017 16:41:30.601469000 UTC
15  Sep 15, 2017 16:41:30.602942000 UTC
16  Sep 15, 2017 16:41:30.640945000 UTC
17  Sep 15, 2017 16:41:42.071615000 UTC
18  Sep 15, 2017 16:41:42.082132000 UTC
19  Sep 15, 2017 16:41:42.082208000 UTC
20  Sep 15, 2017 16:41:42.082303000 UTC
21  Sep 15, 2017 16:41:42.121338000 UTC
```

Pakiet nr 122 zawiera hasło podane przy próbie logowania użytkownika jako użytkownik „administrator”.

```
109  Sep 15, 2017 16:42:14.192950000 UTC
110  Sep 15, 2017 16:42:14.192976000 UTC      server login:
111  Sep 15, 2017 16:42:14.192995000 UTC      admin
112  Sep 15, 2017 16:42:17.238745000 UTC      admin
113  Sep 15, 2017 16:42:17.239814000 UTC      admin
114  Sep 15, 2017 16:42:17.239861000 UTC
115  Sep 15, 2017 16:42:17.242365000 UTC      Password:
116  Sep 15, 2017 16:42:17.242401000 UTC      admin-password
117  Sep 15, 2017 16:42:22.963166000 UTC
118  Sep 15, 2017 16:42:22.963971000 UTC
119  Sep 15, 2017 16:42:22.963993000 UTC
120  Sep 15, 2017 16:42:25.820267000 UTC
121  Sep 15, 2017 16:42:25.820306000 UTC
122  Sep 15, 2017 16:42:25.820469000 UTC      Login incorrect
123  Sep 15, 2017 16:42:25.820484000 UTC
124  Sep 15, 2017 16:42:25.821778000 UTC      server login:
125  Sep 15, 2017 16:42:25.821797000 UTC
```

Komenda “**tshark -Y frame.number==122 -r telnet.pcap**” wyświetla pakiet 122 z pliku telnet.pcap

```
ubuntu@pcapanalysis:~$ tshark -Y frame.number==122 -r telnet.pcap
122  46.304895  172.20.0.3 -> 172.20.0.2  TELNET 83 Telnet Data ...
ubuntu@pcapanalysis:~$
```

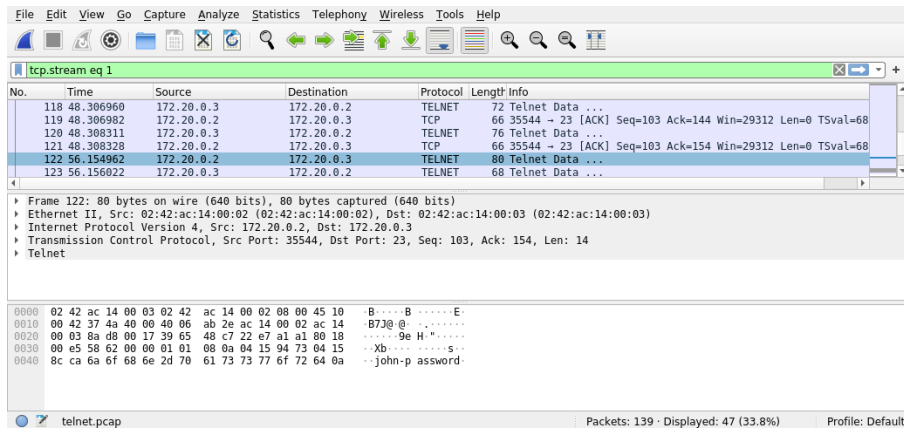
## 5. Wireshark-intro

Lab wireshark-intro przedstawia podstawowe narzędzia dostępne w wiresharku. Lab pokazuje jak zapisać jeden pakiet.

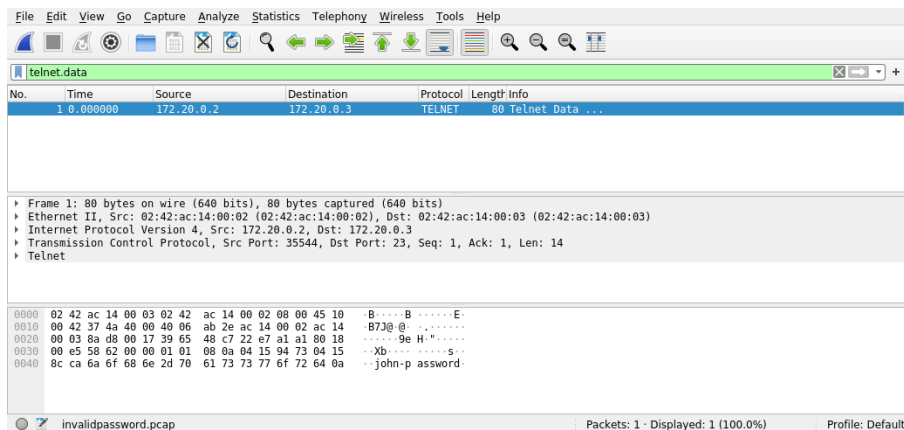
Wireshark odpalamy w terminalu przy użyciu komendy “**wireshark**”

W wiresharku otwieramy plik telnet.pcap

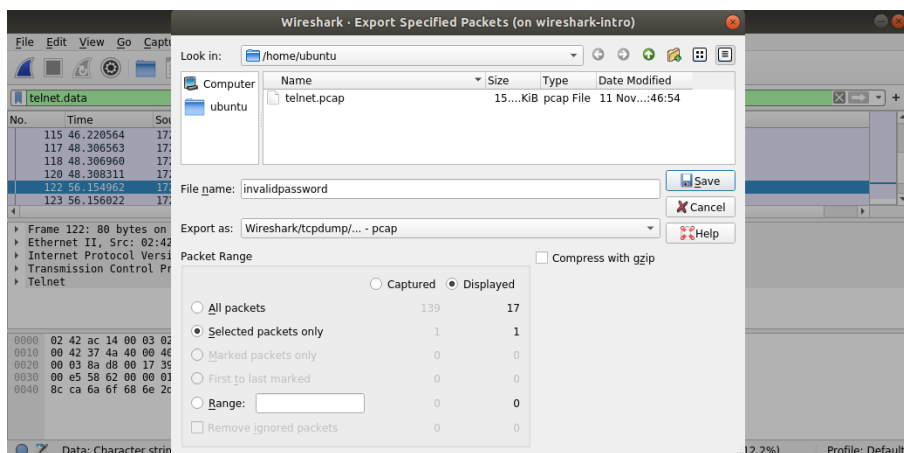
W celu znalezienia nieprawidłowe hasło użytkownika „john” używamy filtra “telnet.data”



W pakiecie 122 znajdujemy niepoprawne hasło kiedy użytkownik próbował zalogować się jako użytkownik “john”



Pojedynczy pakiet z nieprawidłowym hasłem użytkownika „john” zapisujemy jako “invalidpassword.pcap”



## 6. Ogólne przemyślenia

Lab'y dotyczące sieci i analizy ruchu pozwalają zrozumieć podstawowe koncepcje sieci. Lab'y Pcapanalysis i Wireshark-intro pozwala na wstępne zapoznanie się z narzędziami analizy ruchu

sieciowego poprzez realizację jednego pliku. Laby Telnetlab, Network-basics i Routing-basics pozwalają na zrozumienie podstaw sieci.

## Zadania podsumowujące

Spis “Zadania podsumowujące”:

1. Wireshark
2. tcpdump
3. Adresy IP
4. Skrypt dla domeny juniper.net
5. Skrypt w Bash

### 1. Wireshark

Po wejściu do Wiresharka wyznaczyliśmy przegląd zrzutu ruchu HTTP na podstawie pliku http.cap. Przefiltrowaliśmy protokoły, aby wyświetlały się jedynie TCP. Uzgodnienie sesji TCP (sekwencja flag SYN, SYN ACK, ACK) występuje na samym początku:

1	0.000000	145.254.160.237	65.208.228.223	TCP	62 3372 → 80 [SYN] Seq=0 Win=8760 Len=0 MSS=1460 SACK_PERM=1
2	0.911310	65.208.228.223	145.254.160.237	TCP	62 80 → 3372 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1380 SACK_PERM=1
3	0.911310	145.254.160.237	65.208.228.223	TCP	54 3372 → 80 [ACK] Seq=1 Ack=1 Win=9660 Len=0

Sesja zamykana jest również możliwa do wskazania, ponieważ występuje sekwencja flag TCP: FIN+ACK (serwer), ACK (klient), FIN+ACK (klient), ACK (serwer) w kolejnych segmentach.

39	5.017214	145.254.160.237	65.208.228.223	TCP	54 3372 → 80 [ACK] Seq=480 Ack=18365 Win=9236 Len=0
40	17.905747	65.208.228.223	145.254.160.237	TCP	54 80 → 3372 [FIN, ACK] Seq=18365 Ack=480 Win=6432 Len=0
41	17.905747	145.254.160.237	65.208.228.223	TCP	54 3372 → 80 [ACK] Seq=480 Ack=18366 Win=9236 Len=0
42	30.063228	145.254.160.237	65.208.228.223	TCP	54 3372 → 80 [FIN, ACK] Seq=480 Ack=18366 Win=9236 Len=0
43	30.393704	65.208.228.223	145.254.160.237	TCP	54 80 → 3372 [ACK] Seq=18366 Ack=481 Win=6432 Len=0

### 2. tcpdump

Na początku stworzyliśmy plik, który umożliwi nam przechwytywanie aktywności w sieci za pomocą komendy “**sudo tcpdump -w lab.pcap**”. Za pomocą komendy “**sudo tcpdump -nX -r lab.pcap**” odczytuje dane, które otrzymałem po logowaniu na skrzynkę pocztową. Dane, w którym momencie następuje uzgodnienie sesji TCP można było odnaleźć na początku przeglądu.

```

0x0000: 0000 0010 0000 0238 0009 3a08 0000 0238 .....X.....X
14:45:23.405745 IP 10.0.2.15.54162 > 93.184.220.29.80: Flags [S], seq 1486725852, win 64240, options [mss 1460,sackOK,TS val 1209066175 ecr 0,nop,wscale 7], length 0
0x0000: 4500 003c ef8a 4000 4006 054d 0a00 020f E...@.@..M...
0x0010: 5db8 dc1d d392 0050 589d a2dc 0000 0000 ].....PX.....
0x0020: a002 faf0 4613 0000 0204 05b4 0402 080a ....F.....
0x0030: 4810 e2bf 0000 0000 0103 0307 H.....
14:45:23.420915 IP 34.102.187.140.443 > 10.0.2.15.33096: Flags [S.], seq 285632001, ack 2016030538, win 65535, options [mss 1460], length 0
0x0000: 4500 002c 6443 0000 4006 2c88 2266 b88c E...DC..@...f..
0x0010: 0a00 020f 01bb 8148 1106 6601 782a 2f4a .....H...f.x*/J
0x0020: 6012 ffff 0c96 0000 0204 05b4 0000 .....
14:45:23.420950 IP 10.0.2.15.33096 > 34.102.187.140.443: Flags [L.], ack 1, win 64240, length 0
0x0000: 4500 0028 e1e5 4000 4006 6ee9 0a00 020f E...@.@.n.....
0x0010: 2266 bb8c 8148 01bb 782a 2f4a 1106 6602 "f...H...x*/J..f.
0x0020: 5010 faf0 ea1b 0000 P.....

```

Za pomocą komendy **“sudo tcpdump -r | grep „Flags”**” znaleźliśmy flagi w danych. Sekwencja flag TCP: FIN+ACK (serwer), ACK (klient), FIN+ACK (klient), ACK (serwer) w kolejnych segmentach została odnaleziona na rysunku.

```

14:45:39.257491 IP 10.0.2.15.48714 > a104-87-160-216.deploy.static.akamaitechnologies.com.https: Flags [F.], seq 518, ack 1, win 64240, length 0
14:45:39.257802 IP a104-87-160-216.deploy.static.akamaitechnologies.com.https > 10.0.2.15.48714: Flags [L.], ack 519, win 65535, length 0
14:45:39.258488 IP 10.0.2.15.48698 > a104-87-160-216.deploy.static.akamaitechnologies.com.https: Flags [F.], seq 518, ack 1, win 64240, length 0
14:45:39.258730 IP a104-87-160-216.deploy.static.akamaitechnologies.com.https > 10.0.2.15.48698: Flags [L.], ack 519, win 65535, length 0

```

Za pomocą komendy **“sudo tcpdump -r | grep „login”**” znaleźliśmy wszystkie dane ze słowem „login” w danych.

```

(kali@kali)~$ sudo tcpdump -i lab.pcap | grep "login"
reading from file lab.pcap, link-type EN10MB (Ethernet), snapshot length 262144
14:45:28.938895 IP 10.0.2.15.56195 > arrissatton.domain: 518542 [RST] Seq=1000000000, Win=0, Len=0 (RST)
14:45:28.938976 IP 10.0.2.15.56195 > arrissatton.domain: 95894 AAAA? login.microsoftonline.com. (43)
14:45:28.964092 IP arrissatton.domain > 10.0.2.15.56195: 6185 11/0/0 CNAME login.mso.msidentity.com., CNAME ak.privatelink.msidentity.com., CNAME www.tm.ak.prd.aadg.trafficmanager.net., A 20.190.159.75, A 20.190.159.23, A 20.190.159.0, A 20.190.159.64, A 20.190.159.72, A 20.126.31.69, A 40.126.31.71, A 20.190.159.71 (206)
14:45:28.966805 IP arrissatton.domain > 10.0.2.15.56195: 9589 3/1/0 CNAME login.mso.msidentity.com., CNAME ak.privatelink.msidentity.com., CNAME www.tm.ak.prd.aadg.trafficmanager.net. (216)
tcpdump: pcap_loop: truncated dump file; tried to read 1514 captured bytes, only got 922

```

Następnie za pomocą komendy **“sudo tcpdump "port 443" -r lab.pcap**” użyliśmy filtra wyszukiwania jedynie ruchu, który pojawił się na porcie 443.

```

(kali@kali)~$ sudo tcpdump -n port 443 -i lab.pcap
reading from file lab.pcap, link-type EN10MB (Ethernet), snapshot length 262144
14:45:23.039718 IP 10.0.2.15.36042 > 52.39.126.109.443: Flags [S], seq 632800809, win 64240, options [mss 1460,sackOK,TS val 2735989944 ecr 0,nop,wscale 7], length 0
14:45:23.235768 IP 52.39.126.109.443 > 10.0.2.15.36042: Flags [S.], seq 285568001, ack 632800810, win 65535, options [mss 1460], length 0
14:45:23.235802 IP 10.0.2.15.36042 > 52.39.126.109.443: Flags [L.], ack 1, win 64240, length 0
14:45:23.240270 IP 10.0.2.15.36042 > 52.39.126.109.443: Flags [P.], seq 1:518, ack 1, win 64240, length 517
14:45:23.241295 IP 52.39.126.109.443 > 10.0.2.15.36042: Flags [L.], ack 518, win 65535, length 0
14:45:23.293119 IP 10.0.2.15.56336 > 34.102.187.140.443: UDP, length 1357
14:45:23.334933 IP 34.102.187.140.443 > 10.0.2.15.56336: UDP, length 1357
14:45:23.334970 IP 34.102.187.140.443 > 10.0.2.15.56336: UDP, length 1357
14:45:23.339495 IP 34.102.187.140.443 > 10.0.2.15.56336: UDP, length 1161
14:45:23.341382 IP 10.0.2.15.56336 > 34.102.187.140.443: UDP, length 87
14:45:23.485228 IP 10.0.2.15.33096 > 34.102.187.140.443: Flags [S], seq 2016030537, win 64240, options [mss 1460,sackOK,TS val 868663837 ecr 0,nop,wscale 7], length 0
14:45:23.420915 IP 34.102.187.140.443 > 10.0.2.15.33096: Flags [S.], seq 285632001, ack 2016030538, win 65535, options [mss 1460], length 0
14:45:23.420950 IP 10.0.2.15.33096 > 34.102.187.140.443: Flags [L.], ack 1, win 64240, length 0
14:45:23.422920 IP 10.0.2.15.33096 > 34.102.187.140.443: Flags [P.], seq 1:518, ack 1, win 64240, length 517
14:45:23.423145 IP 34.102.187.140.443 > 10.0.2.15.33096: Flags [L.], ack 518, win 65535, length 0
14:45:23.434107 IP 52.39.126.109.443 > 10.0.2.15.36042: Flags [L.], seq 1:2921, ack 518, win 65535, length 2920
14:45:23.434121 IP 10.0.2.15.36042 > 52.39.126.109.443: Flags [L.], ack 2921, win 62780, length 0
14:45:23.434502 IP 52.39.126.109.443 > 10.0.2.15.36042: Flags [P.], seq 2921:3507, ack 518, win 65535, length 586
14:45:23.434610 IP 10.0.2.15.36042 > 52.39.126.109.443: Flags [L.], ack 3507, win 62194, length 0
14:45:23.438066 IP 10.0.2.15.36042 > 52.39.126.109.443: Flags [P.], seq 518:644, ack 3507, win 62780, length 126
14:45:23.438375 IP 52.39.126.109.443 > 10.0.2.15.36042: Flags [L.], ack 644, win 65535, length 0
14:45:23.451408 IP 10.0.2.15.56336 > 34.102.187.140.443: UDP, length 100
14:45:23.451746 IP 10.0.2.15.56336 > 34.102.187.140.443: UDP, length 66
14:45:23.452828 IP 10.0.2.15.56336 > 34.102.187.140.443: UDP, length 292
14:45:23.463846 IP 34.102.187.140.443 > 10.0.2.15.33096: Flags [L.], seq 1:2921, ack 518, win 65535, length 2920
14:45:23.463863 IP 10.0.2.15.33096 > 34.102.187.140.443: Flags [L.], ack 2921, win 62780, length 0
14:45:23.467262 IP 34.102.187.140.443 > 10.0.2.15.33096: Flags [P.], seq 2921:3525, ack 518, win 65535, length 604
14:45:23.467292 IP 10.0.2.15.33096 > 34.102.187.140.443: Flags [L.], ack 3525, win 62780, length 0
14:45:23.472376 IP 10.0.2.15.36042 > 52.39.126.109.443: Flags [P.], seq 644:1269, ack 3507, win 62780, length 625
14:45:23.473298 IP 34.102.187.140.443 > 10.0.2.15.56336: UDP, length 506
14:45:23.473429 IP 52.39.126.109.443 > 10.0.2.15.36042: Flags [L.], ack 1269, win 65535, length 0
14:45:23.473447 IP 34.102.187.140.443 > 10.0.2.15.56336: UDP, length 95
14:45:23.473911 IP 34.102.187.140.443 > 10.0.2.15.56336: UDP, length 28
14:45:23.473933 IP 34.102.187.140.443 > 10.0.2.15.56336: UDP, length 28
14:45:23.473588 IP 34.102.187.140.443 > 10.0.2.15.56336: UDP, length 658
14:45:23.474910 IP 10.0.2.15.33096 > 34.102.187.140.443: Flags [P.], seq 518:582, ack 3525, win 62780, length 64
14:45:23.476572 IP 34.102.187.140.443 > 10.0.2.15.33096: Flags [L.], ack 582, win 65535, length 0
14:45:23.487135 IP 10.0.2.15.56336 > 34.102.187.140.443: UDP, length 37
14:45:23.487631 IP 10.0.2.15.33096 > 34.102.187.140.443: Flags [P.], seq 582:752, ack 3525, win 62780, length 170

```

Aby uruchomić ponownie przechwytywanie, stworzyliśmy nowy plik 443.pcap, na którym ponownie odwiedziliśmy wyszukiwarkę i zapisaliśmy jedynie dane (rysunek), które znajdowały się na porcie 443. Następnie, tym razem za pomocą filtra przechwytywania i komendy **“sudo tcpdump port 443 -w port443.pcap**” (rysunek) wyświetliły się jedynie dane znajdujące się na porcie 443.

```

(kali@kali)~$ sudo tcpdump port 443 -w port443.pcap

```

```

kali@kali:~$ sudo tcpdump port 443 -r port443.pcap
reading from file port443.pcap, link-type EN10MB (Ethernet), snapshot length 262144
15:54:29.976219 IP 10.0.2.15.36292 > 191.144.160.34.bc.googleusercontent.com.https: Flags [S], seq 4049120219, win 64240, options [mss 1460,sackOK,TS val 2021973754 ecr 0,nop,wscale 7], length 0
15:54:29.006931 IP 191.144.160.34.bc.googleusercontent.com.https > 10.0.2.15.36292: Flags [S.], seq 354560001, ack 4049120220, win 65535, options [mss 1460], length 0
15:54:29.006984 IP 10.0.2.15.36292 > 191.144.160.34.bc.googleusercontent.com.https: Flags [.], ack 1, win 64240, length 0
15:54:29.012602 IP 10.0.2.15.36292 > 191.144.160.34.bc.googleusercontent.com.https: Flags [P.], seq 15316, ack 1, win 64240, length 517
15:54:29.013105 IP 191.144.160.34.bc.googleusercontent.com.https > 10.0.2.15.36292: Flags [.], ack 518, win 65535, length 0
15:54:29.048629 IP 191.144.160.34.bc.googleusercontent.com.https > 10.0.2.15.36292: Flags [.], seq 112921, ack 518, win 65535, length 2920
15:54:29.048660 IP 10.0.2.15.36292 > 191.144.160.34.bc.googleusercontent.com.https: Flags [.], ack 2921, win 62780, length 0
15:54:29.048858 IP 191.144.160.34.bc.googleusercontent.com.https > 10.0.2.15.36292: Flags [P.], seq 202114339, ack 518, win 65535, length 1618
15:54:29.048869 IP 10.0.2.15.36292 > 191.144.160.34.bc.googleusercontent.com.https: Flags [.], ack 4539, win 61320, length 0
15:54:29.321402 IP 10.0.2.15.36292 > 191.144.160.34.bc.googleusercontent.com.https: Flags [P.], seq 5181582, ack 4539, win 62780, length 64
15:54:29.322221 IP 191.144.160.34.bc.googleusercontent.com.https > 10.0.2.15.36292: Flags [.], ack 582, win 65535, length 0
15:54:29.324112 IP 10.0.2.15.36292 > 191.144.160.34.bc.googleusercontent.com.https: Flags [P.], seq 5821752, ack 4539, win 62780, length 170
15:54:29.324217 IP 10.0.2.15.36292 > 191.144.160.34.bc.googleusercontent.com.https: Flags [P.], seq 7521037, ack 4539, win 62780, length 285
15:54:29.324315 IP 191.144.160.34.bc.googleusercontent.com.https > 10.0.2.15.36292: Flags [.], ack 752, win 65535, length 0
15:54:29.324325 IP 191.144.160.34.bc.googleusercontent.com.https > 10.0.2.15.36292: Flags [.], ack 1037, win 65535, length 0
15:54:29.338481 IP 191.144.160.34.bc.googleusercontent.com.https > 10.0.2.15.36292: Flags [P.], seq 453915893, ack 1037, win 65535, length 554
15:54:29.338500 IP 10.0.2.15.36292 > 191.144.160.34.bc.googleusercontent.com.https: Flags [.], ack 5893, win 62780, length 0
15:54:29.340567 IP 10.0.2.15.36292 > 191.144.160.34.bc.googleusercontent.com.https: Flags [P.], seq 103710868, ack 5893, win 62780, length 31
15:54:29.340879 IP 191.144.160.34.bc.googleusercontent.com.https > 10.0.2.15.36292: Flags [.], ack 1068, win 65535, length 0
15:54:29.342124 IP 191.144.160.34.bc.googleusercontent.com.https > 10.0.2.15.36292: Flags [P.], seq 509315124, ack 1068, win 65535, length 31
15:54:29.342146 IP 10.0.2.15.36292 > 191.144.160.34.bc.googleusercontent.com.https: Flags [.], ack 5124, win 62780, length 0
15:54:29.343931 IP 191.144.160.34.bc.googleusercontent.com.https > 10.0.2.15.36292: Flags [P.], seq 512410906, ack 1068, win 65535, length 5782
15:54:29.344087 IP 10.0.2.15.36292 > 191.144.160.34.bc.googleusercontent.com.https: Flags [.], ack 10906, win 61320, length 0
15:54:29.344346 IP 10.0.2.15.36292 > 191.144.160.34.bc.googleusercontent.com.https: Flags [P.], seq 106811107, ack 10906, win 62780, length 39
15:54:29.344544 IP 191.144.160.34.bc.googleusercontent.com.https > 10.0.2.15.36292: Flags [.], ack 1107, win 65535, length 0
15:54:29.672605 IP 10.0.2.15.52890 > waw07s03-in-f10.1e100.net.https: Flags [S.], seq 162576480, win 64240, options [mss 1460,sackOK,TS val 3597305931 ecr 0,nop,wscale 7], length 0
15:54:29.692900 IP waw07s03-in-f10.1e100.net.https > 10.0.2.15.52890: Flags [S.], seq 354688801, ack 162576481, win 65535, options [mss 1460], length 0
15:54:29.692950 IP 10.0.2.15.52890 > waw07s03-in-f10.1e100.net.https: Flags [.], ack 1, win 64240, length 0
15:54:29.694547 IP 10.0.2.15.52890 > waw07s03-in-f10.1e100.net.https: Flags [P.], seq 15318, ack 1, win 64240, length 517
15:54:29.695020 IP waw07s03-in-f10.1e100.net.https > 10.0.2.15.52890: Flags [.], ack 518, win 65535, length 0
15:54:29.735378 IP waw07s03-in-f10.1e100.net.https > 10.0.2.15.52890: Flags [P.], seq 11401, ack 518, win 65535, length 1400
15:54:29.735397 IP 10.0.2.15.52890 > waw07s03-in-f10.1e100.net.https: Flags [.], ack 1401, win 63080, length 0
15:54:29.735600 IP waw07s03-in-f10.1e100.net.https > 10.0.2.15.52890: Flags [P.], seq 140114682, ack 518, win 65535, length 3281
15:54:29.735670 IP 10.0.2.15.52890 > waw07s03-in-f10.1e100.net.https: Flags [.], ack 4682, win 61320, length 0
15:54:29.762832 IP 10.0.2.15.52450 > ec2-44-242-3-166.us-west-2.compute.amazonaws.com.https: Flags [S.], seq 44969445, win 64240, options [mss 1460,sackOK,TS val 510300950 ecr 0,nop,wscale 7], length 0
15:54:29.830223 IP waw07s03-in-f10.1e100.net.https > 10.0.2.15.52890: Flags [.], ack 582, win 65535, length 0
15:54:29.830535 IP 10.0.2.15.52890 > waw07s03-in-f10.1e100.net.https: Flags [P.], seq 5821752, ack 4682, win 62780, length 170
15:54:29.830594 IP 10.0.2.15.52890 > waw07s03-in-f10.1e100.net.https: Flags [P.], seq 7521361, ack 4682, win 62780, length 609
15:54:29.830659 IP waw07s03-in-f10.1e100.net.https > 10.0.2.15.52890: Flags [.], ack 752, win 65535, length 0
15:54:29.830978 IP waw07s03-in-f10.1e100.net.https > 10.0.2.15.52890: Flags [.], ack 1361, win 65535, length 0
15:54:29.844410 IP waw07s03-in-f10.1e100.net.https > 10.0.2.15.52890: Flags [P.], seq 468215296, ack 1361, win 65535, length 614
15:54:29.844444 IP 10.0.2.15.52890 > waw07s03-in-f10.1e100.net.https: Flags [.], ack 5296, win 62780, length 0
15:54:29.844756 IP 10.0.2.15.52890 > waw07s03-in-f10.1e100.net.https: Flags [P.], seq 13611392, ack 5296, win 62780, length 31
15:54:29.844905 IP waw07s03-in-f10.1e100.net.https > 10.0.2.15.52890: Flags [.], ack 1392, win 65535, length 0
15:54:29.847067 IP waw07s03-in-f10.1e100.net.https > 10.0.2.15.52890: Flags [P.], seq 529615277, ack 1392, win 65535, length 31
15:54:29.847105 IP 10.0.2.15.52890 > waw07s03-in-f10.1e100.net.https: Flags [.], ack 5327, win 62780, length 0

```

### 3 Adresy IP

#### 1. 1022 urządzeń sieciowych.

Pierwszy adres zarezerwowany dla identyfikatora sieci, ostatni to adres rozgłoszeniowy, maska 22 czyli 1024 adresów

#### 2. adres rozgłoszeniowy 192.168.3.255

Metoda obliczenia adresu rozgłoszeniowego:

- Na początek zamieniamy adresy IP i maski z systemu dziesiętnego na binarny
- Z adresu IP przepisujemy wszystkie bity na pozycjach, w których w adresie maski znajdują się jedyńki, pozostałe miejsca uzupełniamy jedynekami.
- Przeliczamy otrzymany adres z systemu binarnego na system dziesiętny

<http://www.korepetycjenowysacz.edu.pl/wyznaczanie-adresu-rozgloszeniowego/>

#### 3.

podsieć pierwsza: 62 urządzeń; zakres IP: 192.168.111.0 - 192.168.111.63

podsieć druga: 62 urządzeń; zakres IP: 192.168.111.64 - 192.168.111.127

podsieć trzecia: 62 urządzeń; zakres IP: 192.168.111.128 - 192.168.111.191

podsieć czwarta: 62 urządzeń; zakres IP: 192.168.111.192 - 192.168.111.255

Adres podsieci pierwszej: 192.168.111.0/26

Adres podsieci drugiej: 192.168.111.64/26

Adres podsieci trzeciej: 192.168.111.128/26

Adres podsieci czwartej: 192.168.111.192/26







```
[kali@kali]# cat /dev/null && grep -r "href= index.html | cut -d '/' -f 3 | grep %,"  
##### juniper.net  
##### juniper.net  
##### juniper.net  
##### juniper.net  
##### juniper.net  
##### juniper.net  
##### juniper.net  
##### juniper.net  
##### juniper.net  
##### juniper.net  
##### juniper.net  
##### juniper.net  
##### juniper.net  
##### juniper.net  
##### juniper.net  
##### juniper.net  
##### juniper.net  
##### juniper.net  
##### juniper.net  
assets.adobe.com  
junipernetworks.it mtrdc.net  
d1xyc2-pkz.salesforceliveagent.com  
analytics.twitter.com  
consent.trustarc.com  
siteintercept.allgiantecotech.com  
connect.facebook.net  
scriptrs.demandbase.com  
api.demandbase.com  
unpkg.com
```

Listę można jeszcze było oczyścić za pomocą polecenia **"cut"** dla pierwszej kolumny.

```
(kali@kali)-[~]
└─$ grep "href=" index.html | cut -d "/" -f 3 | grep "\." | cut -d '"' -f 1
www.juniper.net
www.juniper.net
www.juniper.net
www.juniper.net
www.juniper.net
www.juniper.net
www.juniper.net
www.juniper.net
www.juniper.net
www.juniper.net
www.juniper.net
www.juniper.net
www.juniper.net
www.juniper.net
www.juniper.net
www.juniper.net
www.juniper.net
www.juniper.net
assets.adobedtm.com
junipernetworks.tt.omrdc.net
dlia3cr7nby.salesforce.com
```

Lista potrzebuje jeszcze jedynie oczyszczenia z duplikatów za pomocą polecenia sort z opcją "**unique(-u)**" i efekt będzie nas zadowalał.

```

root@kali:~# cat -n $(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 32 | grep '^.' | cut -d '.' -f 1 | sort -u) | grep "href=" index.html | cut -d '"' -f 3 | grep '\.' | cut -d '.' -f 1 | sort -u
1 analytics.twitter.com
2 api.demandbase.com
3 apps.juniper.net
4 assets.demandbase.com
5 blogs.juniper.net
6 careers.juniper.net
7 community.juniper.net
8 connect.facebook.net
9 consent.trustarc.com
10 content.juniper.net
11 de.html
12 8-1a3-c2-ph2-salesforceliveagent.com
13 en.html
14 entitlementsearch.juniper.net
15 es.html
16 events.juniper.net
17 fr.html
18 go.juniper.net
19 investor.juniper.net
20 it.html
21 ja.html
22 jsartnertraining.juniper.net
23 junipercommunity.force.com
24 junipernetworksxtm.ontrck.net
25 ko.html
26 learningportal.juniper.net
27 license.juniper.net
28 newroom.juniper.net
29 nl.html
30 partners.juniper.net
31 prresearch.juniper.net
32 pt.html
33 scripts.demandbase.com
34 siteintercept.allegiancetechnology.com
35 support.juniper.net
36 supportportal.juniper.net
37 threatlabs.juniper.net
38 tools.juniper.net
39 twitter.com
40 umhg.com
41 usoffregistration.juniper.net

```

Postanowiliśmy na koniec za pomocą polecenia `host` dla każdej nazwy domeny w utworzonym pliku tekstowym oraz utworzyliśmy pętlę `for`, aby zrealizować automatyzację tego zadania. Chcąc wyodrębnić jedynie adresy IP spośród wszystkich informacji kierujemy dane wyjściowe do polecenia **"`grep`"**. Poszukiwanym wyrażeniem jest `"has address"` . Dane wyjściowe są następnie wycinane i sortowane.

```
(kali@kali)~]
$ cat index.html | grep -o 'http://[^\"]*' | cut -d '/' -f 3 | sort -u > list.txt

(kali@kali)~]
$ for url in $(cat list.txt); do host $url; done
jpartnertraining.juniper.net is an alias for juniperpartners.mindtickle.com.
juniperpartners.mindtickle.com is an alias for k8s-alb-prod-eks-yitzpp-534652310.ap-southeast-1.elb.amazonaws.com.
k8s-alb-prod-eks-yitzpp-534652310.ap-southeast-1.elb.amazonaws.com has address 54.169.221.57
k8s-alb-prod-eks-yitzpp-534652310.ap-southeast-1.elb.amazonaws.com has address 52.76.20.203
schema.org has address 142.250.203.142
schema.org has IPv6 address 2a00:1450:401b:80e::200e
schema.org mail is handled by 5 alt2.aspmx.l.google.com.
schema.org mail is handled by 10 aspmx3.googlemail.com.
schema.org mail is handled by 1 aspmx.l.google.com.
schema.org mail is handled by 5 alt1.aspmx.l.google.com.
schema.org mail is handled by 10 aspmx2.googlemail.com.

(kali@kali)~]
$ for address in $(cat list.txt); do host $address; done | grep "has address" | cut -d " " -f 4 | sort -u
142.250.186.206
52.76.20.203
54.169.221.57
```

## 5. Skrypt w Bash

Na początku, aby w kolejności rosnącej zwrócić liczbę znaków znalezionych ścieżek plików mających w nazwie sshd, zapisaliśmy komendę `$ sudo find / -type f -name sshd* 2>/dev/null | grep 'share' > share.txt`, która znajduje wszystkie ścieżki, zawierające sshd i zapisuje output do pliku share.txt

```
(kali㉿kali)-[~]  
$ sudo find / -type f -name sshd* 2>/dev/null | grep 'share' > share.txt
```

Następnie trzeba było stworzyć polecenie, które umożliwi policzenie liczby wszystkich znaków w ścieżkach odnalezionych przez pierwszą komendę.

```
(kali㉿kali)-[~]  
$ for p in $(cat share.txt); do echo -n "$p" | wc -c; done | sort -n  
29  
30  
32  
33  
36  
37  
42
```