Stanisław Ciszkiewicz

Tymoteusz Malec

Laboratorium 1

ZADANIE 1 (OVERTHEWIRE)

LEVEL 0 - przygotowanie do gry

- wpisujemy do terminala komendę `ssh` (nazwa hosta, numer portu: `2220` i login: `bandit0`)
- wpisujemy hasło (`bandit0`) i przechodzimy do kolejnego poziomu.

```
└─$ ssh bandit.labs.overthewire.org -p 2220 -l bandit0
```

LEVEL 0 → LEVEL 1

- musimy wyszukać hasło, które pozwoli nam przejść do kolejnego poziomu gry (znajduje się ono w pliku „readme" na naszym komputerze) – w tym celu stosujemy polecenie `ls`, a następnie poleceniem `cat` odczytujemy hasło z zawartości pliku „readme",
- kopiujemy hasło, wprowadzamy komendę `exit` (kończymy proces)

```
bandit0@bandit:~$ ls
readme
bandit0@bandit:~$ cat readme
NH2SXQwcBdpmTEzi3bvBHMM9H66vVXjL
```

- następnie wprowadzamy bardzo podobną komendę do tej z zadania LEVEL 0, która przeniesie nas do okna wpisywania hasła:

```
┌──(stacis㉿kali)-[~/Desktop]
└─$ ssh bandit1@bandit.labs.overthewire.org -p 2220 -l bandit0
```

- wklejamy hasło odczytane z pliku „readme", zatwierdzamy i przechodzimy do kolejnego poziomu.

LEVEL 1 → LEVEL 2

- teraz w zadaniu jesteśmy poinformowani, że plik z hasłem do kolejnego poziomu znajduje się w pliku o nazwie „-" (używamy polecenia `ls`, aby sprawdzić, że faktycznie znajduje się w katalogu)
- w celu wczytania zawartości pliku oraz aby komputer właściwie zrozumiał znaczenie „-" w składni polecenia stosujemy poniższy zapis:

```
bandit1@bandit:~$ cat ./-
rRGizSaX8Mk1RTb1CNQoXTcYZWU6lgzi
```

- używamy komendy `exit`, a następnie otrzymane hasło wpisujemy do programu w ten sam sposób co poprzednio i przechodzimy do kolejnego poziomu.

LEVEL 2 → LEVEL 3

- tym razem plik z hasłem ma nazwę „spaces in this filename”,
- używamy polecenia `ls` do znalezienia pliku w katalogu,
- przez spacje w nazwie pliku nie możemy w tradycyjny sposób użyć polecenia `cat` (musimy zapisać nazwę pliku w cudzysłowie):

```
bandit2@bandit:~$ ls
spaces in this filename
bandit2@bandit:~$ cat "spaces in this filename"
aBZ0W5EmUfAf7kHTQeOwd8bauFJ2lAiG
```

- używamy komendy `exit`, wpisujemy hasło i kończymy pracę z zadaniem.

ZADANIE 2 (LABTAINERS)

Telnetlab

- po włączeniu programu labtainers w maszynie wirtualnej uruchamiamy ćwiczenie „telnet” (polecenie `labtainer telnetlab`)
- po uruchomieniu zadania pojawiają się nam dwa okna: server i client, a następnie wykonujemy kolejne zadania z listy:

a) W pierwszym zadaniu szukamy adresu IP serwera (w oknie server wpisujemy polecenie `ifconfig`. Z ekranu terminala odczytujemy adres IP serwera z linii `inet addr: 172.20.0.3`.

```
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:
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.18.0-15-generic x86_64)
```

b) W zadaniu drugim musimy dostać się z panelu klienta do serwera. W tym celu wpisujemy komendę `telnet 172.20.0.3`. Następnie system poprosi nas o login i hasło (obydwa `ubuntu`). W kolejnym kroku wpisujemy komendę `cat filetoview.txt` i oglądamy zawartość pliku. Następnie komendą `exit` opuszczamy sesję na komputerze klienta.

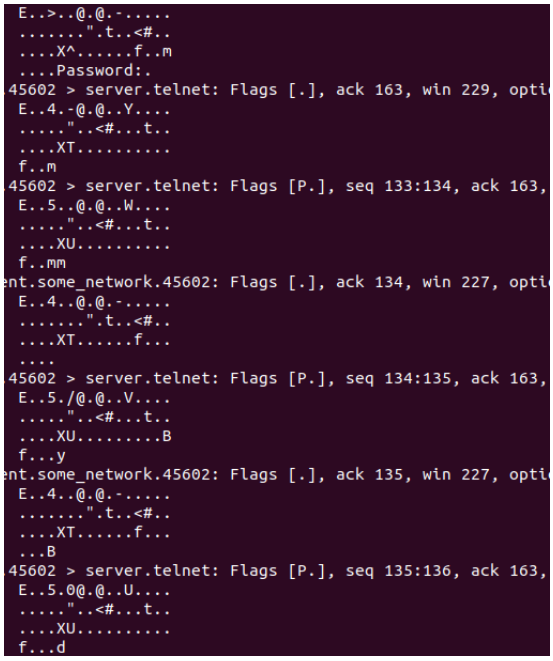
```
                                    ubuntu@server: ~
File  Edit  View  Search  Terminal  Help
ubuntu@server:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:ac:14:00:03
          inet addr:172.20.0.3  Bcast:172.20.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:56 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:7194 (7.1 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)
```

c) Na serwerze uruchamiamy podgląd ruchu na porcie TCP komendą:

```
ubuntu@server:~$ sudo tcpdump -i eth0 -X tcp
```

Następnie uruchamiamy kolejną sesję logowania z komputera klienta (wpisujemy login, jednak zamiast prawidłowego hasła wpisujemy: `mydoghasflees`). W momencie wpisywania hasła przez klienta na ekranie serwera widzimy pojawiające się kolejne raporty z ruchów na porcie TCP (także wpisywane hasło). W naszym przypadku widać je na sygnale ack 163, pomiędzy pojawiają się inne sygnały (poniżej trzy pierwsze litery hasła – „myd"). Dzięki obserwacji ruchu na porcie TCP właściciel serwera widzi jakie operacje są wykonywane na porcie (np. widzi wpisywane przez użytkownika hasło)



d) Wpisujemy komendę `ssh <IP>` z poziomu klienta, który umożliwi nam zarządzanie serwerem. Po wpisaniu hasła próbujemy zobaczyć plik „filetoview.txt". Dzięki ssh nie jesteśmy w stanie odczytać tekstu (jak robiliśmy to poprzednio) na porcie TCP z serwera. Kończymy pracę nad tym zadaniem poleceniem `stoplab telnetlab` z poziomu hosta zadania.

```
ubuntu@client:~$ ssh 172.20.0.3
The authenticity of host '172.20.0.3 (172.20.0.3)' can't be established.
ECDSA key fingerprint is SHA256:nFDnpYXdisAGpF1ZxOBv8Xc83CDp5qYU2frYQvB7Pt8.
Are you sure you want to continue connecting (yes/no)? yes
yes
Warning: Permanently added '172.20.0.3' (ECDSA) to the list of known hosts.
ubuntu@172.20.0.3's password:
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.18.0-15-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
Last login: Wed Nov  9 13:59:29 2022 from telnetlab.client.student.some_network
```

Network-basics

- Po uruchomieniu zadania pokazują nam się okna dwóch komputerów połączonych ze sobą.

a) W pierwszym kroku na obu urządzeniach wpisujemy komendę `ip addr`. Po jej wpisaniu na obu ekranach widzimy spis danych (min. adres MAC urządzenia – poniżej: `02:042:ac:00:00:03` oraz adres IPV4 urządzenia – poniżej: `172.0.0.3/24`).

```
eth0@if25: <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
```

b) W drugim kroku zadania na komputerze nr. 2 wpisujemy komendę `arp -a` (jest to włączenie protokołu sieciowego pozwalającego na uzyskanie adresu MAC innego urządzenia). Na ekranie nie obserwujemy żadnych zmian (komputery jeszcze nie wiedzą nic o sobie). Następnie na komputerze nr. 1 uruchamiamy śledzenie ruchu na protokole TCP komendą `sudo tcpdump -vv -n -e -i eth0`. W kolejnym kroku z komputera 2 pingujemy komputer 1 (sprawdzamy, czy istnieje połączenie między nimi – komenda `ping <IP>`). Na ekranie box1 widzimy ruch na protokole TCP.

```
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
20:51:53.710884 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-h
as 172.0.0.2 tell 172.0.0.3, length 28
20:51:53.711398 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
20:51:53.711726 02:42:ac:00:00:03 > 02:42:ac:00:00:02, ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 20876, 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
20:51:53.713698 02:42:ac:00:00:02 > 02:42:ac:00:00:03, ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 15789, offset 0, flags
```

Możemy zauważyć, że na początku adres MAC jest ukryty (litera f), protokół internetowy ARP prosi o dostęp do adresu MAC, następnnie go otrzymuje (reply). Wtedy komputery mogą wymieniać między sobą wartwy pakietu (ICMP), a widoczne nieco niżej `echo request` to po prostu wywołany przez komputer 2 ping. Następnie używamy ctrl-c i na obu komputerach wpisujemy `arp -a` - dzięki niemu na każdym urządzeniu widzimy adres MAC tego drugiego (poniżej przykład dla box1).

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

c) Po raz kolejny uruchamiamy protokół TCP na komputerze nr. 1 (tym razem komendą `sudo tcpdump -vv -n -i eth0`. Na urządzeniu nr. 2 wpisujemy komendę `ssh <IP>`, a następnie obserwujemy zachowanie TCP na box1. Zuaważamy, że urządzenia łączą się na zasadzie uzgadniania trójetapowego. Najpier box 2 wysyła wiadomość do box1 (flaga [S]), potem następuje odpowiedź box1 (flaga [S.]). Powodzenie połączenia widzimy w 3 etapie: flaga[.], a seq i ack przyjmują wartość 1.

```
0x0000:  829c 9685 9858
21:26:25.638625 IP (tos 0x0, ttl 64, id 41753, offset 0, flags [DF], proto TCP (6), length 60)
    172.0.0.3.33092 > 172.0.0.2.22: Flags [S], cksum 0x5834 (incorrect -> 0x5946), seq 3889576182, win 29200, options
21:26:25.642240 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 60)
    172.0.0.2.22 > 172.0.0.3.33092: Flags [S.], cksum 0x5834 (incorrect -> 0x6bd6), seq 94931887, ack 3889576183, win
0
21:26:25.642733 IP (tos 0x0, ttl 64, id 41754, offset 0, flags [DF], proto TCP (6), length 52)
    172.0.0.3.33092 > 172.0.0.2.22: Flags [.], cksum 0x582c (incorrect -> 0x0ada), seq 1, ack 1, win 229, options [no
```

Routing-basics

- Po uruchomieniu ćwiczenia na ekranie pokazują nam się 3 terminale (na jednym z nich mamy widok czterech komputerów z sieci lokalnej).

a) W pierwszym ćwiczeniu należy wpisać na każdym urządzeniu polecenie `ifconfig`. Widzimy dzięki temu, że urządzenie ws1 i webserver znajdują się w innej podsieci niż komputer ws2 i ws3.

b) Na każdym urządzeniu ws wpisujemy komendę `route -n`. Dzięki temu pokazują się tablice trasowania, zawierające dane dla poszczególnych urządzeń (poniżej przykład dla ws1).

```
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
```

Jedyna różnica jest dla ws3 – nie ma on zapisanej w swojej tabelce informacji dotyczącej bramy sieciowej (gateway).

c) Następnie włączamy protokół śledzenia ruchu na porcie TCP na komputerze ws Gateway (`sudo tcpdump -i eth0 -n -vv`), a z urządzenia ws1 pingujemy urządzenie ws2 (widzimy, że między tymi komputerami, będącymi w tej samej podsieci występuje połączenie). Podjęcie próby połączenia ws1 z ws3 (będącymi w innych podsieciach) okazuje się niemożliwe – ws3 nie jest podłączone do bramy sieciowej.

```
larry@ws3:~$ ping 192.168.1.1 -c 3
ping: connect: Network is unreachable
```

d) Aby móc nawiązać połączenie między komputerami musimy podłączyć ws3 do bramy sieciowej (robimy to z terminala ws3 komendą: `sudo route add default gw [gateway IP]`). Po tej operacji dla sprawdzenia czy podłączenie zadziałało wpisujemy komendę `route -n` (pokazuje się nam zaktualizowana tabela).

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

Po tej operacji jesteśmy w stanie spingować ws1 z ws3 oraz połączyć z web server z ws3 (komenda: `wget [IP web server]`).

e) W kolejnym podpunkcie próbujemy połączyć ze stroną www.google.com z komputera ws2 i ws3 (komenda: `wget www.google.com`) – udaje nam się to z komputera ws2, jednak na ws3 wyświetla się błąd (ws3 nie ma skonfigurowanego DNS). Aby skonfigurować DNS używamy komendy `sudo nano /etc/resolv.conf` . Po otwarciu pliku wpisujemy dokładnie to samo co zawiera „etc/resolv.conf" na komputerze ws2 i zapisujemy zmiany. Następnie sprawdzamy, czy powyższe działanie wprowadziło zmiany w konfiguracji DNS.

```
larry@ws3:~$ wget www.google.com
--2022-11-10 11:15:21--  http://www.google.com/
Resolving www.google.com (www.google.com)... 142.250.186.196, 2a00:1450:401b:80d::2004
Connecting to www.google.com (www.google.com)|142.250.186.196|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: 'index.html.1'

index.html.1            [ <=>                ]  14.40K  --.-KB/s    in 0.006s

2022-11-10 11:15:21 (2.53 MB/s) - 'index.html.1' saved [14750]
```

Zauważamy, że zmiany zostały wprowadzone – system rozumie zapytanie o domenę www.google.com.

f) Tym razem celem zadania będzie zrozumienie działania techniki NAT (Network Adress Translation). Pozwala ona na zmianę źródłowych lub docelowych adresów IP podczas ruchu sieciowego. Wpisujemy komendę `sudo iptables -L -v -t nat`, która pozwoli nam zobaczyć konfigurację NAT.

```
Chain POSTROUTING (policy ACCEPT 2 packets, 168 bytes)
 pkts bytes target     prot opt in     out     source               destination
    6   360 MASQUERADE  all  --  any    eth2    anywhere             anywhere
    1    60 SNAT       tcp  --  any    eth0    anywhere             192.168.1.2          tcp dpt:http to:192.168.1.10
```

W kolejnym kroku uruchamiamy tcpdump na gateway, a na komputerze ws1 wpisujemy komendę `wget www.google.com` (poniżej widok z terminala gateway).

```
11:46:50.038339 IP (tos 0x0, ttl 125, id 31418, offset 0, flags [none], proto TCP (6), length 1452)
    142.250.186.196.80 > 192.168.1.1.51626: Flags [P.], cksum 0x6277 (correct), seq 11297:12709, ack 142, win 6
11:46:50.038346 IP (tos 0x0, ttl 125, id 31419, offset 0, flags [none], proto TCP (6), length 1452)
    142.250.186.196.80 > 192.168.1.1.51626: Flags [P.], cksum 0x0e17 (correct), seq 12709:14121, ack 142, win 6
11:46:50.038424 IP (tos 0x0, ttl 64, id 5459, offset 0, flags [DF], proto TCP (6), length 40)
    192.168.1.1.51626 > 142.250.186.196.80: Flags [.], cksum 0x0b83 (incorrect -> 0x4c81), seq 142, ack 11297,
11:46:50.038435 IP (tos 0x0, ttl 64, id 5460, offset 0, flags [DF], proto TCP (6), length 40)
    192.168.1.1.51626 > 142.250.186.196.80: Flags [.], cksum 0x0b83 (incorrect -> 0x3b95), seq 142, ack 12709,
11:46:50.038441 IP (tos 0x0, ttl 64, id 5461, offset 0, flags [DF], proto TCP (6), length 40)
    192.168.1.1.51626 > 142.250.186.196.80: Flags [.], cksum 0x0b83 (incorrect -> 0x2c75), seq 142, ack 14121,
11:46:50.049596 IP (tos 0x0, ttl 125, id 31420, offset 0, flags [none], proto TCP (6), length 1095)
    142.250.186.196.80 > 192.168.1.1.51626: Flags [P.], cksum 0x2bd8 (correct), seq 14121:15176, ack 142, win 6
11:46:50.049644 IP (tos 0x0, ttl 64, id 5462, offset 0, flags [DF], proto TCP (6), length 40)
    192.168.1.1.51626 > 142.250.186.196.80: Flags [.], cksum 0x0b83 (incorrect -> 0x1d4e), seq 142, ack 15176,
11:46:50.050812 IP (tos 0x0, ttl 64, id 5463, offset 0, flags [DF], proto TCP (6), length 40)
    192.168.1.1.51626 > 142.250.186.196.80: Flags [F.], cksum 0x0b83 (incorrect -> 0x1d4d), seq 142, ack 15176,
```

Następnie wyłączymy tcpdump na gateway i uruchamiamy go ponownie komendą `sudo tcpdump -i eth2 -vv -n`. Po ponownym wpisaniu `wget www.google.com` na komputerze ws1 obserwujemy ruch na porcie TCP z terminala gateway:

```
11:54:28.164718 IP (tos 0x0, ttl 63, id 2113, offset 0, flags [DF], proto TCP (6), length 40)
    203.0.113.10.51634 > 142.250.186.196.80: Flags [.], cksum 0x85e4 (incorrect -> 0x6289), seq 142,
 ack 14121, win 56940, length 0
11:54:28.169549 IP (tos 0x0, ttl 126, id 31452, offset 0, flags [none], proto TCP (6), length 1147)
    142.250.186.196.80 > 203.0.113.10.51634: Flags [P.], cksum 0x8b04 (correct), seq 14121:15228, ac
k 142, win 64240, length 1107: HTTP
11:54:28.169579 IP (tos 0x0, ttl 63, id 2114, offset 0, flags [DF], proto TCP (6), length 40)
    203.0.113.10.51634 > 142.250.186.196.80: Flags [.], cksum 0x85e4 (incorrect -> 0x52ce), seq 142,
 ack 15228, win 59860, length 0
11:54:28.172079 IP (tos 0x0, ttl 63, id 2115, offset 0, flags [DF], proto TCP (6), length 40)
    203.0.113.10.51634 > 142.250.186.196.80: Flags [F.], cksum 0x85e4 (incorrect -> 0x52cd), seq 142
, ack 15228, win 59860, length 0
```

Zauważamy, że zmieniły się porty wyjściowe w raporcie zmieniły się IP wyjściowe (z `192.168.1.1` na `203.0.113.10.51634`).

g) W zadaniu jesteśmy proszeni o wpisanie z konsoli remotews pingu urządzenia ws1. Jest to niemożliwe:

```
hank@remotews:~$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
From 172.16.0.10 icmp_seq=1 Destination Net Unreachable
From 172.16.0.10 icmp_seq=2 Destination Net Unreachable
From 172.16.0.10 icmp_seq=3 Destination Net Unreachable
From 172.16.0.10 icmp_seq=4 Destination Net Unreachable
From 172.16.0.10 icmp_seq=43 Destination Net Unreachable
From 172.16.0.10 icmp_seq=85 Destination Net Unreachable
```

Dzięki Nat niemożliwe jest zainicjowanie połączenia z wewnętrznym urządzeniem z poziomu internetu. W konsoli gateway po raz kolejny uruchamiamy iptables i analizujemy NAT (DNAT w sekcji prerouting i SNAT w sekcji postrouting).

```
Chain PREROUTING (policy ACCEPT 43 packets, 2994 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DNAT       tcp  --  eth2   any     anywhere             anywhere             tcp dpt:http to:192.168.1.2

Chain INPUT (policy ACCEPT 11 packets, 714 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 12 packets, 739 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 2 packets, 168 bytes)
 pkts bytes target     prot opt in     out     source               destination
   17  1039 MASQUERADE  all  --  any    eth2    anywhere             anywhere
    1    60 SNAT       tcp  --  any    eth0    anywhere             192.168.1.2          tcp dpt:http to:192.168.1.10
```

Jeszcze raz włączamy tcpdump na gateway – tym razem web server jest dostępny dla internetu (co tworzy potencjalne ryzyko).

```
12:29:09.707984 IP (tos 0x0, ttl 62, id 8295, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.1.10.58248 > 192.168.1.2.80: Flags [.], cksum 0x8383 (incorrect -> 0x704b), seq 140, ack
 452, win 237, options [nop,nop,TS val 1615070555 ecr 2289658535], length 0
12:29:09.728591 IP (tos 0x0, ttl 62, id 8296, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.1.10.58248 > 192.168.1.2.80: Flags [F.], cksum 0x8383 (incorrect -> 0x7036), seq 140, ac
k 452, win 237, options [nop,nop,TS val 1615070575 ecr 2289658535], length 0
12:29:09.731028 IP (tos 0x0, ttl 64, id 52324, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.1.2.80 > 192.168.1.10.58248: Flags [F.], cksum 0x8383 (incorrect -> 0x7020), seq 452, ac
k 141, win 235, options [nop,nop,TS val 2289658558 ecr 1615070575], length 0
12:29:09.731331 IP (tos 0x0, ttl 62, id 8297, offset 0, flags [DF], proto TCP (6), length 52)
```

Pcapanalysis

- zadanie polega na zaznajomieniu się i analizie plików PCAP
- po uruchomieniu zadania pojawia nam się jedno okno terminala o nazwie pcpanalysis

a) W pierwszym kroku w terminalu wpisujemy polecenie `man tshark` – pozwala ono na zaznajomienie się z podstawowymi opcjami dla komendy tshark. Następnie widzimy, że wpisanie komendy `tshark -T fields -e frame.number -e frame.time -e telnet.data -r telnet.pcap` pozwala na uzyskanie bardziej sprecyzowanych danych.

b) Tym razem musimy znaleźć pojedynczą ramkę zawierającą hasło podane przez użytkownika podczas próby logowania na konto administratora (mamy użyć Tsharka z komendą `-Y frame.number==N`). W tym celu do komendy z poprzedniego podpunktu dopisujemy kóncówkę: `-Y frame.number==140`.

```
ubuntu@pcapanalysis:~$ tshark -T fields -e frame.number -e frame.time -e telnet.data -r telnet.pcap -Y frame.number==140
140     Sep 15, 2017 16:59:41.963166000 UTC     admin-password
```

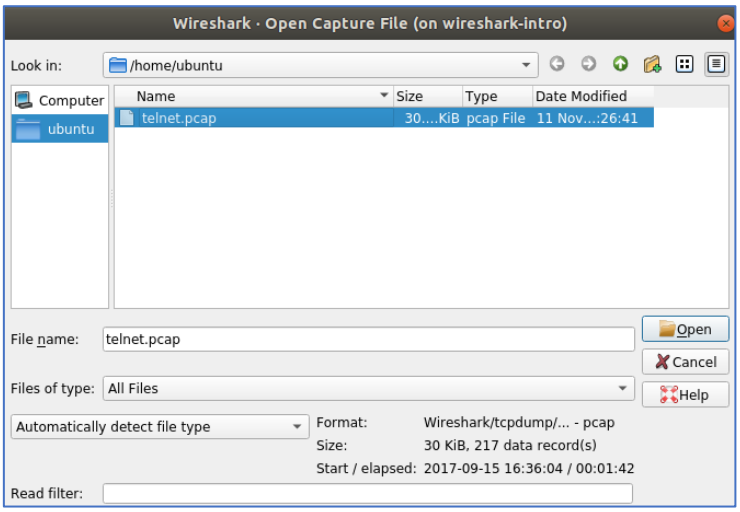
Wireshark-intro

- po uruchomieniu zadania pokazuje się nam jedno okno terminala o nazwie *wireshark*-intro

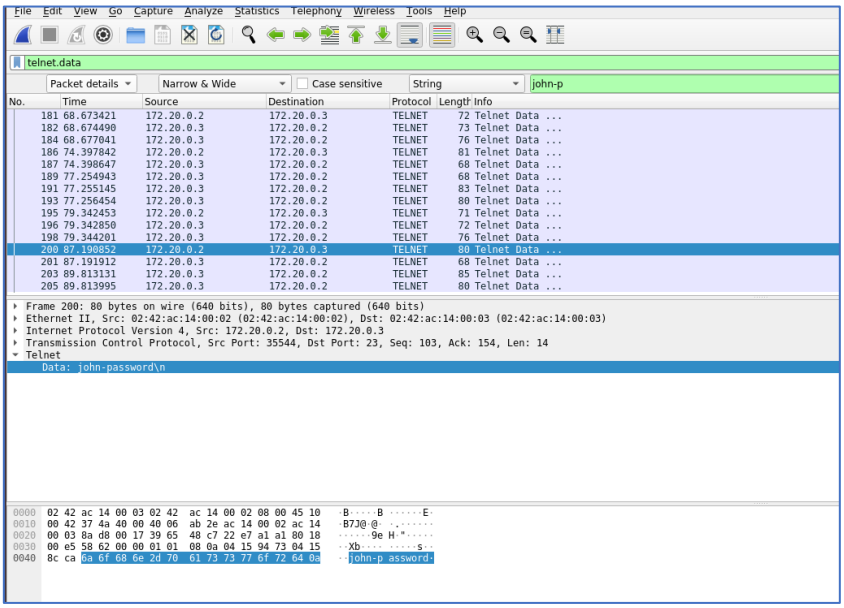
a) W terminalu wpisujemy komendę `ls -l`, aby zobaczyć zawartość zadania. Następnie wpisujemy komendę `file telnet.pcap`, żeby zobaczyć informacje o pliku.

```
ubuntu@wireshark-intro:~$ ls -l
total 32
-rw-rw-r-- 1 ubuntu ubuntu 31110 Nov 11 14:26 telnet.pcap
ubuntu@wireshark-intro:~$ file telnet.pcap
telnet.pcap: pcap capture file, microsecond ts (little-endian) - version 2.4 (Ethernet, capture length 262144)
```

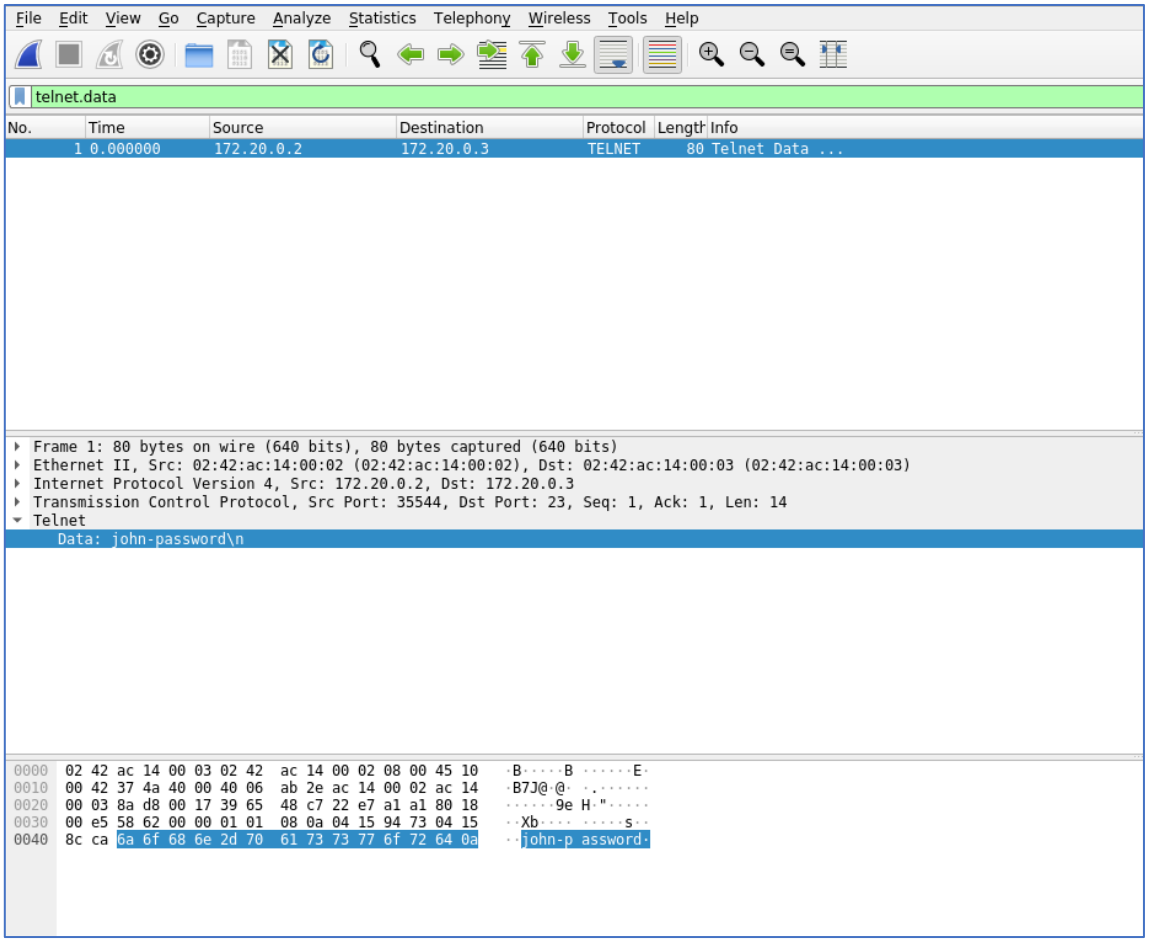
b) Następnie wpisujemy w konsoli `wireshark` – uruchamia się program wireshark – po czym otwieramy w nim plik „telnet.pcap”.



c) Po uruchomieniu programu wireshark próbujemy wyszukać w nim pakietu z nieprawidłowym hasłem wpisanym przez użytkownika user (john). W tym celu naciskamy na ikonę lupki w pasku narzędzi i wpisujemy w sekcji Packet details wyszukiwanie stringa: john-p.



Następnie według instrukcji eksportujemy znalezione pakiety, a nowy plik otwieramy, aby zobaczyć poprawność wykonanego zadania.



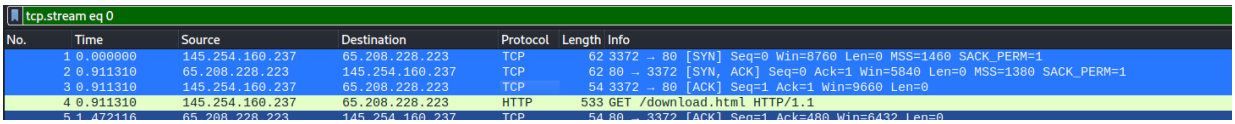
d) W ostatnim podpunkcie dalej zapoznajemy się z działaniem systemu wireshark – tym razem w sekcji Analyze wybieramy Follow, a następnie TCP stream (testujemy różne funkcje).

Przemyślenia

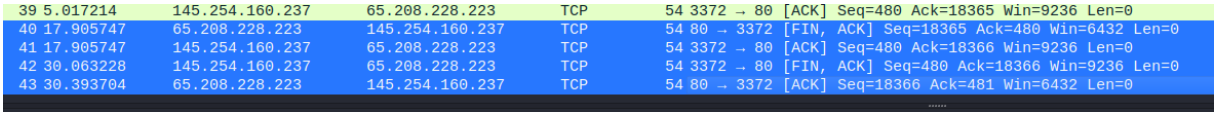
Wykonane powyżej ćwiczenia przedstawiają nam wiele sposobów na analizę ruchów na portach. W zależności od sytuacji możemy użyć rozwiązania z konsoli lub dedykowanej aplikacji (tj wireshark). Zadania pokazują, jak duże możliwości oferują nam te narzędzia, tłumaczą jak działa przepływ informacji między komputerami oraz jak porozumiewają się urządzenia między sobą (czego nie widać z poziomu użytkownika sieci).

ZADANIA PODSUMOWUJĄCE

Zadanie 1



a) Uzgodnienie sesji następuje w pierwszych trzech (flagi SYN, SYN i ACK, ACK)



b) Sesja zamykana jest w ostatnich czterech (flagi FIN i ACK, ACK, FIN i ACK, ACK)

Zadanie 2

```
┌──(kali㉿kali)-[~]
└─$ sudo tcpdump -w result.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
^C261 packets captured
261 packets received by filter
0 packets dropped by kernel
```

a) Zaczynam nasłuchiwanie i zapisuję do pliku komendą `sudo tcpdump -w result.pcap`

```
┌──(kali㉿kali)-[~]
└─$ tcpdump -r result.pcap
reading from file result.pcap, link-type EN10MB (Ethernet), snapshot length 262144
21:04:52.209795 IP 10.0.2.15.45558 > 40.99.217.130.https: Flags [.], ack 396410312, win 65535, length 0
21:04:52.210184 IP 40.99.217.130.https > 10.0.2.15.45558: Flags [.], ack 1, win 65535, length 0
21:04:52.471889 IP 10.0.2.15.53354 > 40.126.32.133.https: Flags [.], ack 392458519, win 65535, length 0
21:04:52.471928 IP 10.0.2.15.48450 > 40.79.141.153.https: Flags [.], ack 388423702, win 62780, length 0
21:04:52.471937 IP 10.0.2.15.44498 > 13.69.239.72.https: Flags [.], ack 396934767, win 62780, length 0
```

b) Wyświetlam zapisany ruch komendą `tcpdump -r result.pcap`

```
21:04:58.016124 IP 10.0.2.15.59220 > 40.126.32.136.https: Flags [S], seq 728534983, win 64240, options [mss 1
460,sackOK,TS val 3947065693 ecr 0,nop,wscale 7], length 0
21:04:58.054691 IP 40.126.32.136.https > 10.0.2.15.59220: Flags [S.], seq 400192001, ack 728534984, win 65535
, options [mss 1460], length 0
21:04:58.054752 IP 10.0.2.15.59220 > 40.126.32.136.https: Flags [.], ack 1, win 64240, length 0
```

c) Uzgodnienie sesji następuje na zdj powyżej (flagi S, S., .)

```
21:15:20.114163 IP a2-17-240-223.deploy.static.akamaitechnologies.com.https > 10.0.2.15.59244: Flags [F.], se
q 646, ack 966, win 65535, length 0
21:15:20.114389 IP 10.0.2.15.59244 > a2-17-240-223.deploy.static.akamaitechnologies.com.https: Flags [P.], se
q 966:1005, ack 647, win 63976, length 39
21:15:20.114682 IP 10.0.2.15.59244 > a2-17-240-223.deploy.static.akamaitechnologies.com.https: Flags [P.], se
q 1005:1029, ack 647, win 63976, length 24
21:15:20.114705 IP a2-17-240-223.deploy.static.akamaitechnologies.com.https > 10.0.2.15.59244: Flags [.], ack
 1005, win 65535, length 0
21:15:20.114752 IP 10.0.2.15.59244 > a2-17-240-223.deploy.static.akamaitechnologies.com.https: Flags [F.], se
q 1029, ack 647, win 63976, length 0
21:15:20.114885 IP a2-17-240-223.deploy.static.akamaitechnologies.com.https > 10.0.2.15.59244: Flags [.], ack
 1029, win 65535, length 0
```

d) Zakończenie sesji następuje na zdj powyżej (flagi F., ., F., .)

```
┌──(kali㉿kali)-[~]
└─$ tcpdump -r result.pcap | grep 'log'
reading from file result.pcap, link-type EN10MB (Ethernet), snapshot length 262144
```

e) Żeby wyszukać w pliku moment logowanie używam komendy `grep 'log'` żeby zostawić tylko linijki zawierające „log"

```
359, win 65535, length 0
21:05:07.532708 IP 10.0.2.15.45794 > compalhub.home.domain: 20898+ A? autologon.microsoftazuread-sso.com. (52
)
```

f) Wśród odfiltrowanych linijek jest sekwencja logowania – połączenie ze stroną autologon.microsoft.zuread-sso.com

```
┌──(kali㉿kali)-[~]
└─$ tcpdump port 443 -r result.pcap
reading from file result.pcap, link-type EN10MB (Ethernet), snapshot length 262144
21:04:52.209795 IP 10.0.2.15.45558 > 40.99.217.130.https: Flags [.], ack 396410312, win 65535, length 0
21:04:52.210184 IP 40.99.217.130.https > 10.0.2.15.45558: Flags [.], ack 1, win 65535, length 0
21:04:52.471889 IP 10.0.2.15.53354 > 40.126.32.133.https: Flags [.], ack 392458519, win 65535, length 0
21:04:52.471928 IP 10.0.2.15.48450 > 40.79.141.153.https: Flags [.], ack 388423702, win 62780, length 0
21:04:52.471937 IP 10.0.2.15.44498 > 13.69.239.72.https: Flags [.], ack 396934767, win 62780, length 0
21:04:52.472362 IP 40.126.32.133.https > 10.0.2.15.53354: Flags [.], ack 1, win 65535, length 0
```

g) Żeby wyświetlić ruch tylko na porcie 443 używam komendy `tcpdump port 443 -r result.cpap`

```
┌──(kali㉿kali)-[~]
└─$ sudo tcpdump port 443 -w result2.pcap
[sudo] hasło użytkownika kali:
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

h) Żeby przechwycić ruch tylko na porcie 443 i zapisać używam komendy `sudo tcpdump port 443 -w result2.pcap`

Zadanie 3

a) 192.168.1.16/22:
   a. Adres: 11000000.10101000.00000001.00010000
   b. Maska: 11111111.11111111.11111100.00000000
   c. Adres urządzenia ma 10 bitów. Dostępnych jest 2^10 – 2 = 1022 adresów

b) Adres Rozgłoszeniowy: 11000000.10101000.00000011.11111111 – 192.168.3.255

c) 192.168.111.0/24:
   a. Adres: 11000000.10101000.01101111.00000000
   b. Maska: 11111111.11111111.11111111.00000000
   c. Sieć pierwsza:
      i. Adres: 11000000.10101000.01101111.00000000
      ii. Maska: 11111111.11111111.11111111.11000000
      iii. Odp: 192.168.111.0/26
   d. Sieć druga:
      i. Adres: 11000000.10101000.01101111.01000000
      ii. Maska: 11111111.11111111.11111111.11000000
      iii. Odp: 192.168.111.64/26

e. Sieć trzecia:
   i. Adres: 11000000.10101000.01101111.10000000
   ii. Maska: 11111111.11111111.11111111.11000000
   iii. Odp: 192.168.111.128/26
f. Sieć czwarta:
   i. Adres: 11000000.10101000.01101111.11000000
   ii. Maska: 11111111.11111111.11111111.11000000
   iii. Odp: 192.168.111.192/26

Zadanie 4

```
┌──(kali㉿kali)-[~]
└─$ wget juniper.net
--2022-11-15 19:48:08--  http://juniper.net/
Translacja juniper.net (juniper.net)... 52.42.68.58
Łączenie się z juniper.net (juniper.net)|52.42.68.58|:80... połączono.
Żądanie HTTP wysłano, oczekiwanie na odpowiedź... 301 Moved Permanently
Lokalizacja: https://www.juniper.net/ [podążanie]
--2022-11-15 19:48:08--  https://www.juniper.net/
Translacja www.juniper.net (www.juniper.net)... 104.81.231.45, 2a02:26f0:d8:3a2::720, 2a02:26f0:d8:38b::720
Łączenie się z www.juniper.net (www.juniper.net)|104.81.231.45|:443... połączono.
Żądanie HTTP wysłano, oczekiwanie na odpowiedź... 200 OK
Długość: nieznana [text/html]
Zapis do: `index.html'

index.html                  [ <=>                                ] 211,95K  --.-KB/s     w 0,05s

2022-11-15 19:48:10 (4,33 MB/s) - zapisano `index.html' [217038]
```

a) Używam komendy `wget jupiter.net`, plik „index.html” zostaje zapisany

```
┌──(kali㉿kali)-[~]
└─$ cat index.html | grep -o 'https://[^"]*' | cut -d "/" -f 3 | sort -u > list.txt

┌──(kali㉿kali)-[~]
└─$ cat list.txt
analytics.twitter.com
api.demandbase.com
apps.juniper.net
ar.wikipedia.org
assets.adobedtm.com
azb.wikipedia.org
bg.wikipedia.org
blogs.juniper.net
```

b) Używam następujących komend, żeby znaleźć wszystkie linki na stronie:
   a. `cat index.html` – wypisz zawartość pliku „index.html
   b. `grep -o 'https://[^"]*'`:
      i. `-o` – znajdź i wypisz tylko część linijki, która zgadza się z wyszukaną regułą
      ii. `[^"]*` – jakikolwiek znak prócz ‘”’ powtórzony ilekolwiek razy
   c. `cut -d „/" -f 3` – podziel po znaku „/” i weź trzeci index (po drugim znaku „/”)
   d. `sort -u` – posortuj usuwając duplikaty
   e. `> list.txt` – zapisz do pliku „list.txt”

```
┌──(kali㉿kali)-[~]
└─$ for url in $(cat list.txt); do host $url; done | grep "has address" | cut -d " " -f 4 | sort -u
100.21.209.79
104.16.122.175
104.16.123.175
104.16.124.175
104.16.125.175
104.16.126.175
104.244.42.129
```

c) Używam powyższego polecenia, żeby znaleźć adresy ip – dla każdej linijki w pliku „list.txt” wołam polecenie host, znajduje tylko te które mają adres, wycinam tak by został sam adres i sortuje usuwając duplikaty

Zadanie 5

```
┌──(kali㉿kali)-[~]
└─$ tree -if / | grep "share" | grep "sshd" > paths.txt

┌──(kali㉿kali)-[~]
└─$ cat paths.txt
/usr/share/desktop-directories/14-06-sshd-service.directory
/usr/share/doc/metasploit-framework/modules/exploit/windows/ssh/freesshd_authbypass.md
/usr/share/man/man1/sshdump.1.gz
/usr/share/man/man5/authorized_keys.5.gz → ../man8/sshd.8.gz
/usr/share/man/man5/sshd_config.5.gz
```

a) Używam komendy powyżej, żeby zapisać wszystkie pełne ścieżki do plików, które zawierają w sobie „share” oraz „sshd”

```
┌──(kali㉿kali)-[~]
└─$ cat paths.txt | grep "sshd[^/]*\." | grep "share[^/]*/" | cut -d " " -f 1 > paths2.txt


┌──(kali㉿kali)-[~]
└─$ cat paths2.txt
/usr/share/desktop-directories/14-06-sshd-service.directory
/usr/share/doc/metasploit-framework/modules/exploit/windows/ssh/freesshd_authbypass.md
/usr/share/man/man1/sshdump.1.gz
/usr/share/man/man5/authorized_keys.5.gz
```

b) Powyższą komendą filtruje dodatkowo ścieżki żeby „share” było w nazwie folderu w ścieżce (po „share” był „/”) a „sshd” było w nazwie pliku (po „sshd” była „.”) oraz usuwam powstałe przy komendzie tree „ -> (inny folder)” komendą cut

```
┌──(kali㉿kali)-[~]
└─$ for p in $(cat paths2.txt); do echo ${#p}; done | sort -n
29
32
33
36
37
40
42
59
83
85
86
89
```

c) Powyższą komendą wypisuje długości ścieżek w kolejności rosnącej