Zadanie 1 – Przygotowanie środowiska testowego 2

Zadanie 2 – Pozyskiwanie informacji z sieci przy użyciu skanera Nmap 3

Fedora 3

Linux Mint 3

Kali Linux 4

Windows 10 4

Nmap 5

Ogólne skanowanie 5

Skanowanie wersji 6

Windows 6

Linux Mint 8

Kali Linux 8

Host - Fedora Linux 8

Spostrzeżenie 9

Zadanie 3 – Analiza ruchu sieciowego przy wykorzystaniu narzędzia TCPdump 10

Ping Urządzeń 10

Ping DG 11

Strona internetowa 12

Filtr portu 80 i 443 12

Zadanie 4 – Analiza ruchu sieciowego przy wykorzystaniu programu Wireshark 13

Stealth Scan 13

Skan szybki 14

Zadanie 5 – Analiza pliku zawierającego dane pakietów z zainfekowanego komputera 16

a. Podaj adres IP komputera, który został poddany analizie. 16

b. Podaj adres gatewaya tego komputera. 17

c. Czy przedstawione zdarzenie działo się w ramach wirtualnych maszyn? Na jakiej podstawie zostały wyciągnięte wnioski? 17

d. Czy w trakcie działania zainfekowanego komputera jesteśmy w stanie określić, czy stacja była skanowana w sieci w poszukiwaniu otwartych portów? 17

e. Jeśli tak, to przez kogo (IP sprawcy i jaką metodą), jeśli nie, to jakich informacji brakuje w badanym pliku? 17

f. W takcie działania zainfekowanego komputera został rozgłoszony ARP z adresem MAC (00:0c:29:ec:8a:14). Do kogo należy? 18

g. Analizowane logi zawierają informacje o pliku wykonywalny exe. Sprawdź, kiedy został pobrany, z którego adresu i jak nazywa się plik? 18

h. Przy użyciu opcji z Wireshark „Extract Object” wyciągnij odnaleziony plik, zapisz go w nowym folderze i przy pomocy narzędzia md5sum sprawdź jego sumę kontrolną. 18

i. Pozyskaną sumę kontrolną wklej na stronie https://www.virustotal.com w zakładce search. Przedstaw i opisz wynik analizy. 19

j. Który z portów był wykorzystywany do przesyłania danych pochodzących z ataku? 20

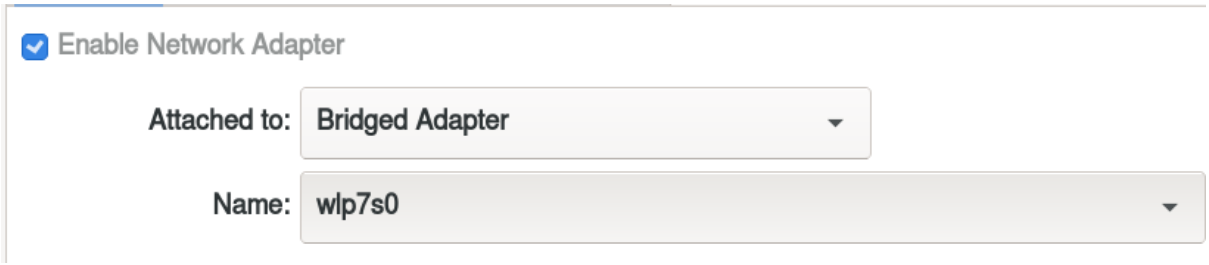
k. Podaj nazwę komputera, który został zaatakowany. 20

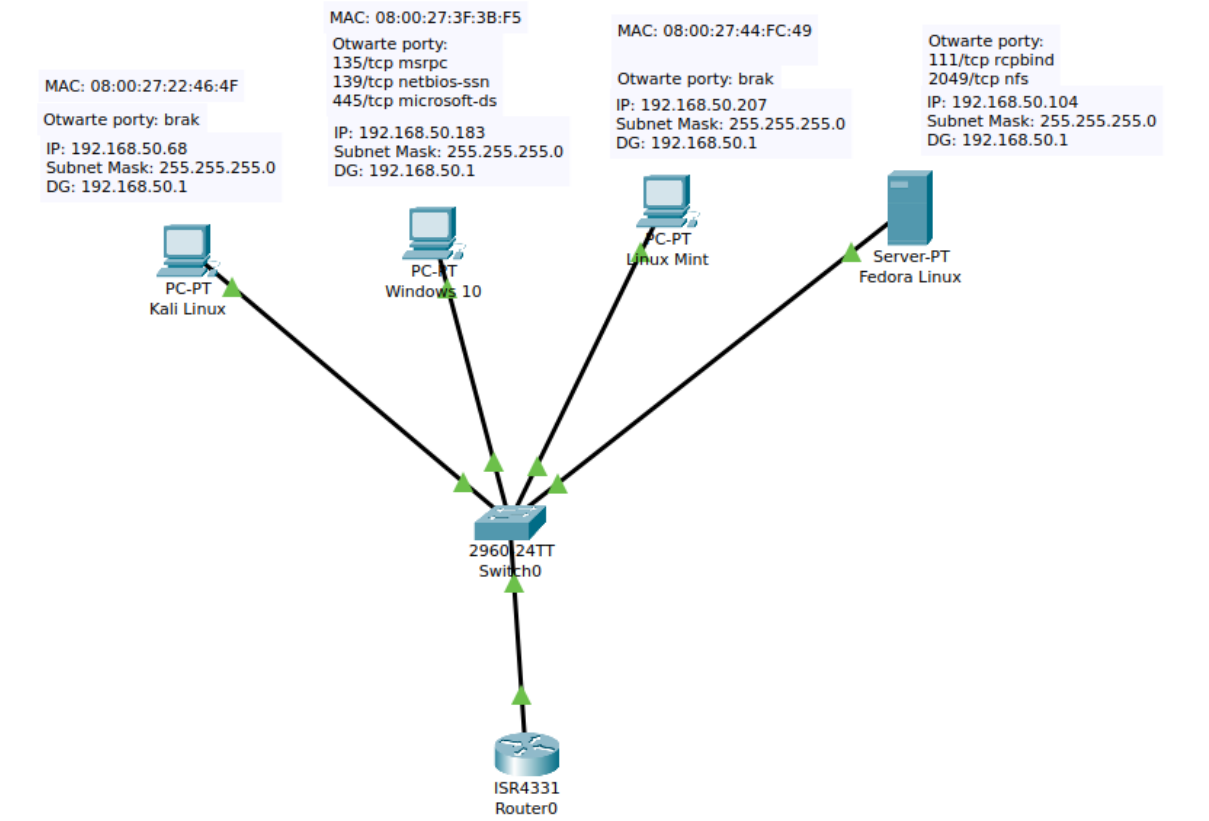
Zadanie 6 – NetworkMiner jako alternatywny program do analizy ruchu sieciowego 21

Zadanie 1 – Przygotowanie środowiska testowego

Stworzyłem 3 maszyny wirtualne - 2 z linuxem (mint + kali) i 1 z windowsem

Wszystkie dodałem do jednej sieci - poprzez kartę sieciową hosta





Logiczna topologia sieci
Mamy dostępne 4 urządzenia pracujące w trybie klienta (mimo iż fizycznie fedora jest hostem dla reszty maszyn wirtualnych - dlatego zaznaczyłem komputer z tym systemem jako *serwer*) w jednej sieci o adresie **192.168.50.0/24**

Urządzenia potrafią się bez problemu komunikować między sobą jak i z siecią zewnętrzną.

Zadanie 2 – Pozyskiwanie informacji z sieci przy użyciu skanera Nmap

Fedora

```
[arek@fedora ~] $ ifconfig wlp7s0 && route -n
wlp7s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.50.104  netmask 255.255.255.0  broadcast 192.168.50.255
        inet6 fe80::4f9a:26d4:f70c:c1d2  prefixlen 64  scopeid 0x20<link>
        ether 48:f1:7f:f0:f9:33  txqueuelen 1000  (Ethernet)
        RX packets 670811  bytes 917342200 (874.8 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 102853  bytes 15469494 (14.7 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.50.1    0.0.0.0         UG    600    0        0 wlp7s0
192.168.50.0    0.0.0.0         255.255.255.0   U     600    0        0 wlp7s0
```

Linux Mint

```
mirek@mirekPC:~$ ifconfig enp0s3 && route -n
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.50.207  netmask 255.255.255.0  broadcast 192.168.50.255
        inet6 fe80::e80b:724e:27ce:1307  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:44:fc:49  txqueuelen 1000  (Ethernet)
        RX packets 537  bytes 429646 (429.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 357  bytes 91255 (91.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.50.1    0.0.0.0         UG    100    0        0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
192.168.50.0    0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
```

Kali Linux

```
┌──(kali㉿kali)-[~]
└─$ ifconfig eth0 && route -n
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.50.68  netmask 255.255.255.0  broadcast 192.168.50.255
        inet6 fe80::9e06:4968:3334:56e6  prefixlen 64  scopeid 0×20<link>
        ether 08:00:27:22:46:4f  txqueuelen 1000  (Ethernet)
        RX packets 87  bytes 13233 (12.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 24  bytes 3940 (3.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.50.1    0.0.0.0         UG    100    0        0 eth0
192.168.50.0    0.0.0.0         255.255.255.0   U     100    0        0 eth0
```

Windows 10

```
C:\Users\winda>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::9fe:673f:4e72:1bfd%13
   IPv4 Address. . . . . . . . . . . : 192.168.50.183
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.50.1
```

```
C:\Users\winda>route print
===========================================================================
Interface List
 13...08 00 27 3f 3b f5 ......Intel(R) PRO/1000 MT Desktop Adapter
  1...........................Software Loopback Interface 1
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0     192.168.50.1   192.168.50.183     25
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    331
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    331
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    331
    192.168.50.0    255.255.255.0         On-link    192.168.50.183    281
   192.168.50.183  255.255.255.255         On-link    192.168.50.183    281
   192.168.50.255  255.255.255.255         On-link    192.168.50.183    281
        224.0.0.0        240.0.0.0         On-link         127.0.0.1    331
        224.0.0.0        240.0.0.0         On-link    192.168.50.183    281
  255.255.255.255  255.255.255.255         On-link         127.0.0.1    331
  255.255.255.255  255.255.255.255         On-link    192.168.50.183    281
===========================================================================
```

Nmap

Ogólne skanowanie

Do przeprowadzenia skanowania użyję maszyny hosta (Fedora Linux)

```
[arek@fedora ~] $ sudo nmap -sN 192.168.50.0/24
```

Dokonaj skanowania całej podsieci z flagą *-sN* pozwalającą na skanowanie typu PING wraz z szukaniem najpopularniejszych otwartych portów

```
Nmap scan report for kali (192.168.50.68)
Host is up (0.00029s latency).
All 1000 scanned ports on kali (192.168.50.68) are in ignored states.
Not shown: 1000 closed tcp ports (reset)
MAC Address: 08:00:27:22:46:4F (Oracle VirtualBox virtual NIC)
```

```
Nmap scan report for Windows (192.168.50.183)
Host is up (0.00064s latency).
All 1000 scanned ports on Windows (192.168.50.183) are in ignored states.
Not shown: 1000 closed tcp ports (reset)
MAC Address: 08:00:27:3F:3B:F5 (Oracle VirtualBox virtual NIC)
```

```
Nmap scan report for mirekPC (192.168.50.207)
Host is up (0.00031s latency).
All 1000 scanned ports on mirekPC (192.168.50.207) are in ignored states.
Not shown: 1000 open|filtered tcp ports (no-response)
MAC Address: 08:00:27:44:FC:49 (Oracle VirtualBox virtual NIC)
```

```
Nmap scan report for fedora (192.168.50.104)
Host is up.
All 1000 scanned ports on fedora (192.168.50.104) are in ignored states.
Not shown: 1000 open|filtered tcp ports (no-response)
```

Skanowanie przeszło pomyślnie
Udało się znaleźć wszystkie urządzenia (**nmap** nawet przeskanował komputer, z którego nastąpił ten proces - **fedora**), są w stanie aktywnym. Co ciekawe, urządzenia linuxowe figurują jako ich nazwa użytkownika - a system Windows jako po prostu *Windows* (zamiast nazwy użytkownika - *winda*)

Wszystkie urządzenia mają zamknięte porty tcp

Udało się również znaleźć adresy mac urządzeń (najprawdopodobniej za pomocą protokołu ARP) a na ich podstawie jesteśmy w stanie określić dostawcę karty sieciowej - w tym przypadku programu VirtualBox. Dzięki takiemu skanowaniu można pochopnie stwierdzić, czy w sieci znajdują się systemy zwirtualizowane.

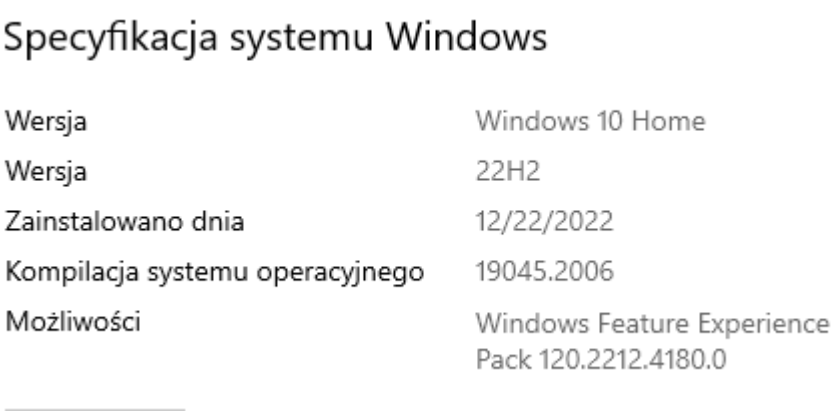
Skanowanie wersji

Windows

```
[arek@fedora ~] $ sudo nmap -O 192.168.50.183
[sudo] password for arek:
Starting Nmap 7.93 ( https://nmap.org ) at 2022-12-23 11:26 CET
Nmap scan report for Windows (192.168.50.183)
Host is up (0.00052s latency).
Not shown: 997 closed tcp ports (reset)
PORT    STATE SERVICE
135/tcp open  msrpc
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds
MAC Address: 08:00:27:3F:3B:F5 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Microsoft Windows 10
OS CPE: cpe:/o:microsoft:windows_10
OS details: Microsoft Windows 10 1709 - 1909
Network Distance: 1 hop
```

Można się stąd dowiedzieć paru rzeczy:

1) Wersja systemu to Windows 10. Co ciekawe, nmap potrafi określić przedział build'u -

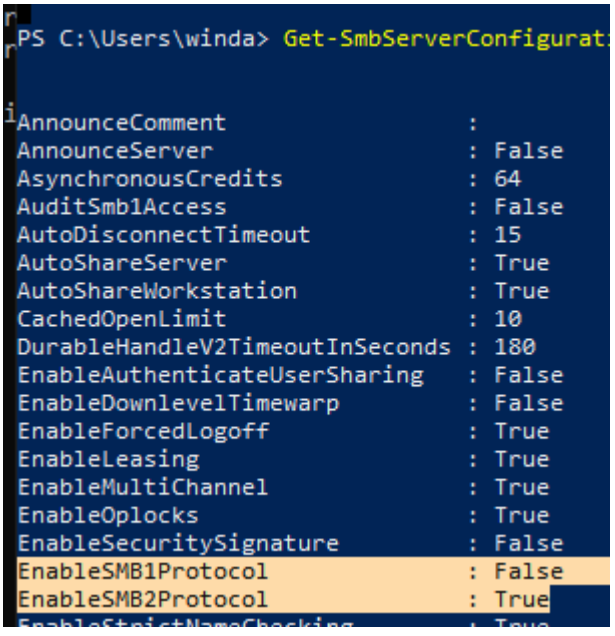


pokazuje przedział 1709 - 1909, jednakże nie jest to prawdą - w rzeczywistości jest wersja **22H2**

2) Poprzednie skanowanie wskazywało na zamknięte porty, a jednak 3 porty tcp są otwarte
   - 135/tcp open msrpc - nie wskazuje na szczególne zagrożenie, odpowiada za transferowanie zasobów za pomocą protokołu SMB, źródło
   - 139/tcp open netbios-ssn - podobnie jak port 135
   - 445/tcp open microsoft-ds - również podobne

   jest to charakterystyczna grupa portów ogólnie odpowiadająca za usługę NetBIOS Microsoft'u

Owe otwarte porty wystawione w sieci lokalnej nie są zagrożeniem - jedynie jeśli system Windows używa starszej wersji Samby - 1, w naszym przypadku jest używana wersja 2




```
Host script results:
|_nbstat: NetBIOS name: WINDOWS, NetBIOS user: <unknown>, NetBIOS MAC: 0800273f3bf5 (Oracle VirtualBox virtual NIC)
| smb2-time:
|   date: 2022-12-23T11:42:48
|_  start_date: N/A
|_clock-skew: -4m19s
| smb2-security-mode:
|   311:
|_    Message signing enabled but not required

TRACEROUTE
HOP RTT     ADDRESS
1   0.23 ms Windows (192.168.50.183)
```

Używając flagi *-A* skanera **nmap**, jesteśmy w stanie dokładnie sprawdzić poziom bezpieczeństwa protokołu **SMB**
Jest dostępny również adres MAC serwera NetBIOS (również widać, że zwirtualizowany)

Linux Mint

```
[arek@fedora ~] $ sudo nmap -O 192.168.50.207
Starting Nmap 7.93 ( https://nmap.org ) at 2022-12-23 11:49 CET
Nmap scan report for mirekPC (192.168.50.207)
Host is up (0.00051s latency).
All 1000 scanned ports on mirekPC (192.168.50.207) are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)
MAC Address: 08:00:27:44:FC:49 (Oracle VirtualBox virtual NIC)
Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop
```

Niestety nie udało się znaleźć szczegółów
Nie wiadomo jaki to system, a tym bardziej jakiej wersji
Potwierdza się wcześniejsze pomyślne skanowanie portów - wszystkie typu tcp są zamknięte

Kali Linux

```
[arek@fedora ~] $ sudo nmap -O 192.168.50.68
Starting Nmap 7.93 ( https://nmap.org ) at 2022-12-23 11:53 CET
Nmap scan report for kali (192.168.50.68)
Host is up (0.00024s latency).
All 1000 scanned ports on kali (192.168.50.68) are in ignored states.
Not shown: 1000 closed tcp ports (reset)
MAC Address: 08:00:27:22:46:4F (Oracle VirtualBox virtual NIC)
Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop
```

Taka sama sytuacja jak w przypadku maszyny z dystrybucją **Mint**

Host - Fedora Linux

Skanowanie przeprowadzę z maszyny z Kali Linuxem

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -O 192.168.50.104
Starting Nmap 7.92 ( https://nmap.org ) at 2022-12-23 05:54 EST
Nmap scan report for 192.168.50.104
Host is up (0.00042s latency).
Not shown: 845 closed tcp ports (reset), 151 filtered tcp ports (no-response), 3 filtered tcp ports (admin-prohibit
ed)
PORT     STATE SERVICE
2049/tcp open  nfs
```

```
Device type: general purpose
Running: Linux 5.X
OS CPE: cpe:/o:linux:linux_kernel:5
OS details: Linux 5.0 - 5.4
Network Distance: 1 hop
```

Tutaj output wygląda bogaciej

Typowo otwarty jest jeden port - 2049/tcp nfs
Służy on, podobnie jak Samba, do przesyłania zasobów za pomocą sieci
Nie dziwi mnie iż jest on otwarty, ponieważ w sieci mam serwer typu **NAT**

Udało się znaleźć również iż dostawcą karty sieciowej jest Intel

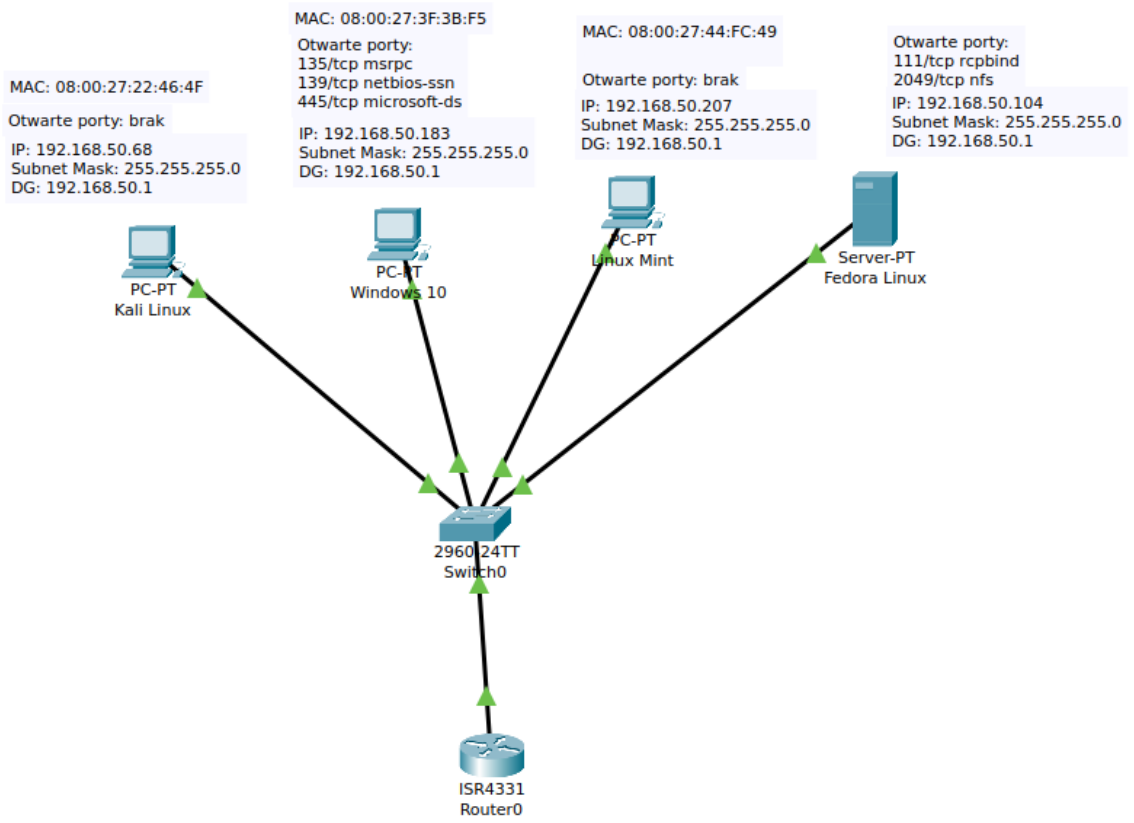
Jeśli chodzi o wersje systemu, to skaner prawidłowo wykrył iż jest oparty na linuxie Podobnie jak w przypadku Windowsa - nie zgadza się wersja. Pokazuje on pewien przedział, jednakże rzeczywista wersja systemu (jądra) jest wyższa

Spostrzeżenie

Skanowanie przeprowadziłem z Fedory na wszystkie pozostałe urządzenia oraz z Kali'ego również na resztę - wyniki były takie same - z jednym małym wyjątkiem

Jeśli przeprowadzimy skan komputera z Fedorą na tym samym komputerze, wynik znacząco się różni

1) jest dodatkowo jeszcze jeden otwarty port - 111/tcp open rpcbind - który odpowiada za komunikację klient <-> serwer, najprawdopodobniej jest to związane z uruchomionym udostępnianiem kart sieciowej programu VirtualBox
2) brak informacji o adresie MAC urządzenia
3) Wersja systemu jest kompletnie inna - Linux 2.6.X - a więc jeszcze bardziej przestarzała i nieaktualna
4) Oczywiście wartość *Network Distance* wynosi 0



Zadanie 3 – Analiza ruchu sieciowego przy wykorzystaniu narzędzia TCPdump

Rozpoczynam łapanie pakietów na komputerze host

```
[arek@fedora ~] $ sudo tcpdump -i wlp7s0 -v
```

Ping Urządzeń

Output komendy:

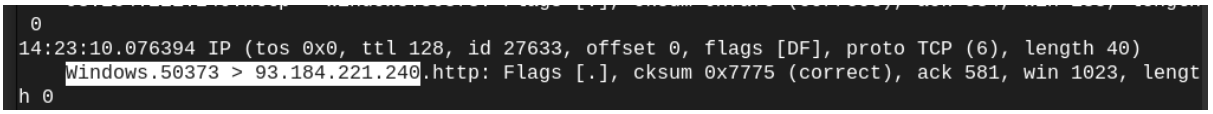
```
14:23:10.319250 IP (tos 0x0, ttl 64, id 51608, offset 0, flags [DF], proto ICMP (1), length 84)
    mirekPC > fedora: ICMP echo request, id 1, seq 15, length 64
14:23:10.319275 IP (tos 0x0, ttl 64, id 63098, offset 0, flags [none], proto ICMP (1), length 84)
    fedora > mirekPC: ICMP echo reply, id 1, seq 15, length 64
```

```
14:23:10.469715 IP (tos 0x0, ttl 64, id 41066, offset 0, flags [DF], proto ICMP (1), length 84)
    kali > fedora: ICMP echo request, id 51438, seq 25, length 64
14:23:10.469771 IP (tos 0x0, ttl 64, id 40160, offset 0, flags [none], proto ICMP (1), length 84)
    fedora > kali: ICMP echo reply, id 51438, seq 25, length 64
```

```
14:23:08.459145 IP (tos 0x0, ttl 128, id 25632, offset 0, flags [none], proto ICMP (1), length 60
    Windows > fedora: ICMP echo request, id 1, seq 3, length 40
14:23:08.459169 IP (tos 0x0, ttl 64, id 15205, offset 0, flags [none], proto ICMP (1), length 60)
    fedora > Windows: ICMP echo reply, id 1, seq 3, length 40
```

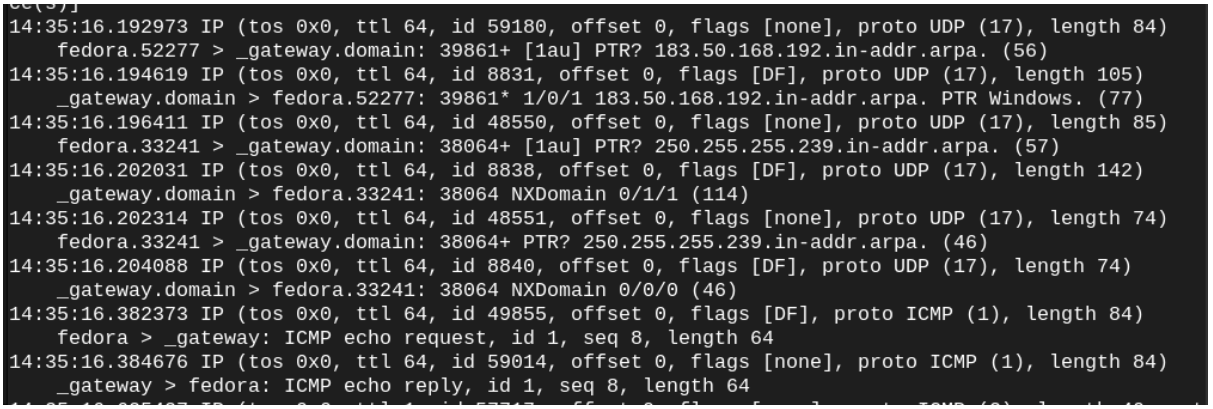
Udało się przechwycić pakiety wszystkich pingujących maszyn - są to pakiety typu ICMP echo request i reply
Można zobaczyć tutaj takie parametry jak: protokół, id, ttl, offset, długość nagłówka

Co ciekawe, możemy zobaczyć iż windows samoistnie łączył się z serwerami, o których nie idzie znaleźć informacji



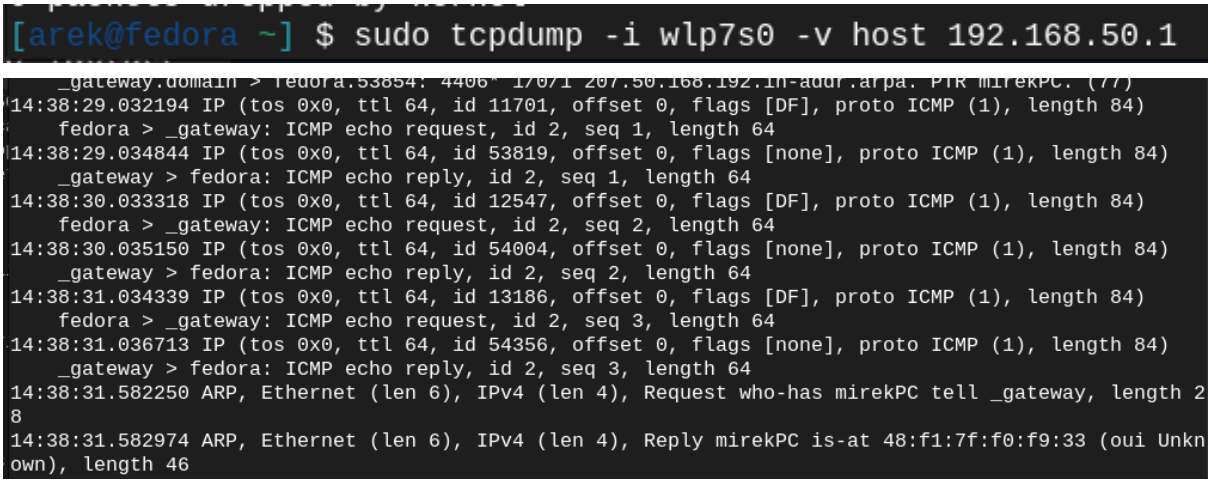
Prawdopodobnie windows (jak zwykle) po prostu pobiera aktualizacje

Ping DG



Struktura output'u wykonanej komendy wygląda bardzo podobnie
Różni się jedynie tym, że brakuje dokładnego adresu ip - a zamiast niego jest nazwa **_gateway.domain**

Przefiltrowane odpowiedzi tylko od bramy domyślnej



Co ciekawe, udało się również znaleźć pakiet protokołu ARP z innej maszyny

Strona internetowa

Przechwytywanie pakietów z połączenia się z przykładową stroną internetową:

```
[arek@fedora ~] $ sudo tcpdump -i wlp7s0 -v dst host www.google.com
dropped privs to tcpdump
tcpdump: listening on wlp7s0, link-type EN10MB (Ethernet), snapshot length 26214
4 bytes
16:10:51.847994 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto UDP (17),
 length 1385)
    fedora.36916 > waw07s06-in-f4.1e100.net.https: UDP, length 1357
16:10:51.848020 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto UDP (17),
 length 299)
    fedora.36916 > waw07s06-in-f4.1e100.net.https: UDP, length 271
16:10:51.883839 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto UDP (17),
```

Filtr portu 80 i 443

```
[arek@fedora ~] $ sudo tcpdump -i wlp7s0 -v dst port 80
```

```
16:12:48.510631 IP (tos 0x0, ttl 64, id 46725, offset 0, flags [DF], proto TCP (6), length 478)
    fedora.39534 > waw02s17-in-f3.1e100.net.http: Flags [P.], cksum 0xb1fd (correct), seq 1:427, ack 1, win
 501, options [nop,nop,TS val 899294136 ecr 3465891451], length 426: HTTP, length: 426
        POST /gts1c3 HTTP/1.1
        Host: ocsp.pki.goog
        User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:108.0) Gecko/20100101 Firefox/108.0
        Accept: */*
        Accept-Language: en-US,en;q=0.5
        Accept-Encoding: gzip, deflate
        Content-Type: application/ocsp-request
        Content-Length: 83
        DNT: 1
        Connection: keep-alive
        Pragma: no-cache
        Cache-Control: no-cache
```

Wysyłając zapytanie do strony internetowej program tcpdump odzyskał cały nagłówek strony

oraz na porcie 443

```
[arek@fedora ~] $ sudo tcpdump -i wlp7s0 -v dst port 443
```

```
16:15:01.400833 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto UDP (17), length 69)
    fedora.34338 > waw07s03-in-f3.1e100.net.https: UDP, length 41
16:15:01.403536 IP (tos 0x0, ttl 64, id 6283, offset 0, flags [DF], proto TCP (6), length 175)
    fedora.44868 > 756573.cloudwaysapps.com.https: Flags [P.], cksum 0xd6dd (correct), seq 3033:3156, ack 51
 280, win 618, options [nop,nop,TS val 2422285190 ecr 148411392], length 123
16:15:01.404133 IP (tos 0x0, ttl 64, id 6284, offset 0, flags [DF], proto TCP (6), length 87)
    fedora.44868 > 756573.cloudwaysapps.com.https: Flags [P.], cksum 0x7ba1 (correct), seq 3156:3191, ack 51
 280, win 618, options [nop,nop,TS val 2422285190 ecr 148411392], length 35
16:15:01.427416 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto UDP (17), length 142)
    fedora.34338 > waw07s03-in-f3.1e100.net.https: UDP, length 114
16:15:01.429221 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto UDP (17), length 64)
    fedora.34338 > waw07s03-in-f3.1e100.net.https: UDP, length 36
16:15:01.459324 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto UDP (17), length 60)
    fedora.34338 > waw07s03-in-f3.1e100.net.https: UDP, length 32
16:15:01.474080 IP (tos 0x0, ttl 64, id 6285, offset 0, flags [DF], proto TCP (6), length 64)
    fedora.44868 > 756573.cloudwaysapps.com.https: Flags [.], cksum 0xcc5b (correct), ack 51771, win 615, op
tions [nop,nop,TS val 2422285260 ecr 148411573,nop,nop,sack 1 {54571:55513}], length 0
16:15:01.474402 IP (tos 0x0, ttl 64, id 6286, offset 0, flags [DF], proto TCP (6), length 52)
    fedora.44868 > 756573.cloudwaysapps.com.https: Flags [.], cksum 0x97cf (correct), ack 55513, win 586, op
tions [nop,nop,TS val 2422285261 ecr 148411574], length 0
```

Można tu odnaleźć m.in. protokół UDP, adres serwera (domena), ttl, offset, wielkość nagłówka

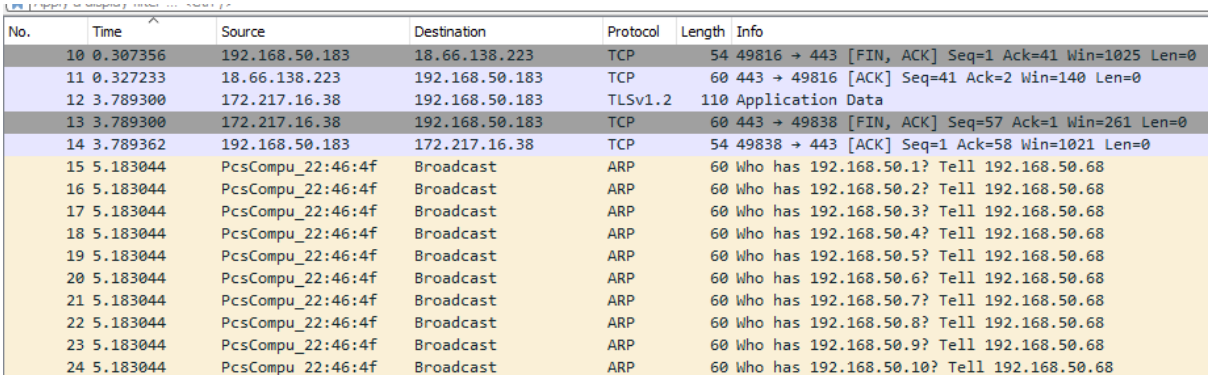
W tym przypadku mamy do czynienia z paroma pakietami typu UDP o małej długości - najprawdopodobniej są to pakiety pochodzące ze strony streamingowej - youtube.com

Zadanie 4 – Analiza ruchu sieciowego przy wykorzystaniu programu Wireshark

Stealth Scan

Rozpoczynam skanowanie całej podsieci za pomocą maszyny z zainstalowanym Kali Linuxem

W międzyczasie rozpocząłem łapanie pakietów w programie **Wireshark**



No.	Time	Source	Destination	Protocol	Length	Info
10	0.307356	192.168.50.183	18.66.138.223	TCP	54	49816 → 443 [FIN, ACK] Seq=1 Ack=41 Win=1025 Len=0
11	0.327233	18.66.138.223	192.168.50.183	TCP	60	443 → 49816 [ACK] Seq=41 Ack=2 Win=140 Len=0
12	3.789300	172.217.16.38	192.168.50.183	TLSv1.2	110	Application Data
13	3.789300	172.217.16.38	192.168.50.183	TCP	60	443 → 49838 [FIN, ACK] Seq=57 Ack=1 Win=261 Len=0
14	3.789362	192.168.50.183	172.217.16.38	TCP	54	49838 → 443 [ACK] Seq=1 Ack=58 Win=1021 Len=0
15	5.183044	PcsCompu_22:46:4f	Broadcast	ARP	60	Who has 192.168.50.1? Tell 192.168.50.68
16	5.183044	PcsCompu_22:46:4f	Broadcast	ARP	60	Who has 192.168.50.2? Tell 192.168.50.68
17	5.183044	PcsCompu_22:46:4f	Broadcast	ARP	60	Who has 192.168.50.3? Tell 192.168.50.68
18	5.183044	PcsCompu_22:46:4f	Broadcast	ARP	60	Who has 192.168.50.4? Tell 192.168.50.68
19	5.183044	PcsCompu_22:46:4f	Broadcast	ARP	60	Who has 192.168.50.5? Tell 192.168.50.68
20	5.183044	PcsCompu_22:46:4f	Broadcast	ARP	60	Who has 192.168.50.6? Tell 192.168.50.68
21	5.183044	PcsCompu_22:46:4f	Broadcast	ARP	60	Who has 192.168.50.7? Tell 192.168.50.68
22	5.183044	PcsCompu_22:46:4f	Broadcast	ARP	60	Who has 192.168.50.8? Tell 192.168.50.68
23	5.183044	PcsCompu_22:46:4f	Broadcast	ARP	60	Who has 192.168.50.9? Tell 192.168.50.68
24	5.183044	PcsCompu_22:46:4f	Broadcast	ARP	60	Who has 192.168.50.10? Tell 192.168.50.68

Skanowanie takie jak najbardziej idzie wykryć, jeśli analizujemy je za pomocą urządzenia z zainstalowanym **Wiresharkiem** w tej samej podsieci
Jest to widoczne po tym iż program **nmap** skanuje każdy host po kolei (iteruje) za pomocą protokołu ARP (sprawdza jakie hosty są dostępne)
Co ciekawe, source o nazwie *PcsCompu_22:46:4f* w łatwy sposób wskazuje na 3 adres MAC urządzenia skanującego - w tym przypadku się zgadza

Następnie rozpoczęła się komunikacja między Kali'm a Windowsem mająca na celu skanowanie różnych portów

No.	Time	Source	Destination	Protocol	Length	Info
610	10.400815	192.168.50.183	192.168.50.68	TCP	54	993 → 58525 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
611	10.404422	192.168.50.68	192.168.50.183	TCP	60	58525 → 8080 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
612	10.404422	192.168.50.68	192.168.50.183	TCP	60	58525 → 139 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
613	10.404479	192.168.50.183	192.168.50.68	TCP	54	8080 → 58525 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
614	10.404708	192.168.50.183	192.168.50.68	TCP	58	139 → 58525 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
615	10.404988	192.168.50.68	192.168.50.183	TCP	60	58525 → 111 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
616	10.404988	192.168.50.68	192.168.50.183	TCP	60	58525 → 445 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
617	10.404988	192.168.50.68	192.168.50.183	TCP	60	58525 → 8443 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
618	10.404988	192.168.50.68	192.168.50.183	TCP	60	58525 → 55056 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
619	10.405062	192.168.50.183	192.168.50.68	TCP	54	111 → 58525 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
620	10.405241	192.168.50.183	192.168.50.68	TCP	58	445 → 58525 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
621	10.405297	192.168.50.183	192.168.50.68	TCP	54	8443 → 58525 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
622	10.405326	192.168.50.183	192.168.50.68	TCP	54	55056 → 58525 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
623	10.405761	192.168.50.68	192.168.50.183	TCP	60	58525 → 139 [RST] Seq=1 Win=0 Len=0
624	10.405761	192.168.50.68	192.168.50.183	TCP	60	58525 → 445 [RST] Seq=1 Win=0 Len=0
625	10.408516	192.168.50.68	192.168.50.183	TCP	60	58525 → 9500 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
626	10.408516	192.168.50.68	192.168.50.183	TCP	60	58525 → 2068 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
627	10.408516	192.168.50.68	192.168.50.183	TCP	60	58525 → 5030 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
628	10.408516	192.168.50.68	192.168.50.183	TCP	60	58525 → 873 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
629	10.408581	192.168.50.183	192.168.50.68	TCP	54	9500 → 58525 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
630	10.408641	192.168.50.183	192.168.50.68	TCP	54	2068 → 58525 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

No.	Time	Source	Destination	Protocol	Length	Info
1600	14.394168	192.168.50.183	192.168.50.68	TCP	54	5822 → 58525 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1601	14.397621	192.168.50.68	192.168.50.183	TCP	60	58525 → 8031 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1602	14.397647	192.168.50.183	192.168.50.68	TCP	54	8031 → 58525 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1603	14.401297	192.168.50.68	192.168.50.183	TCP	60	58525 → 44443 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1604	14.401318	192.168.50.183	192.168.50.68	TCP	54	44443 → 58525 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1605	14.405825	192.168.50.68	192.168.50.183	TCP	60	58525 → 2967 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1606	14.405848	192.168.50.183	192.168.50.68	TCP	54	2967 → 58525 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1607	14.411884	192.168.50.68	192.168.50.183	TCP	60	58525 → 8254 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1608	14.411915	192.168.50.183	192.168.50.68	TCP	54	8254 → 58525 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1609	14.417866	192.168.50.68	192.168.50.183	TCP	60	58525 → 563 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1610	14.417890	192.168.50.183	192.168.50.68	TCP	54	563 → 58525 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1611	14.423923	192.168.50.68	192.168.50.183	TCP	60	58525 → 1443 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1612	14.423945	192.168.50.183	192.168.50.68	TCP	54	1443 → 58525 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1613	14.429917	192.168.50.68	192.168.50.183	TCP	60	58525 → 3766 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1614	14.429942	192.168.50.183	192.168.50.68	TCP	54	3766 → 58525 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1615	14.436293	192.168.50.68	192.168.50.183	TCP	60	58525 → 2121 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1616	14.436314	192.168.50.183	192.168.50.68	TCP	54	2121 → 58525 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1617	14.440501	192.168.50.68	192.168.50.183	TCP	60	58525 → 1164 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1618	14.440518	192.168.50.183	192.168.50.68	TCP	54	1164 → 58525 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1619	14.446974	192.168.50.68	192.168.50.183	TCP	60	58525 → 3369 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1620	14.446992	192.168.50.183	192.168.50.68	TCP	54	3369 → 58525 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

W logach również mogą pojawić się różne TCPStreamy

tcp.stream eq 809

No.	Time	Source	Destination	Protocol	Length	Info
2150	16.521824	192.168.50.68	192.168.50.183	TCP	60	58525 → 9595 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
2151	16.521843	192.168.50.183	192.168.50.68	TCP	54	9595 → 58525 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

A jest ich znacząco więcej, niż jest skanowanych portów

Skan szybki

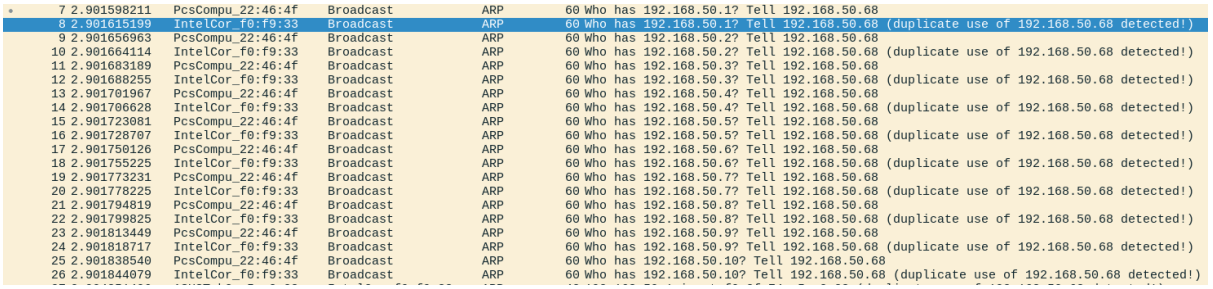
Używam następującej komendy

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap 192.168.50.0/24 -T5 --data-length 32 -f
```

flaga *-f* świadczy o fragmentowaniu pakietów w celu unikania firewall'i/ids'ów

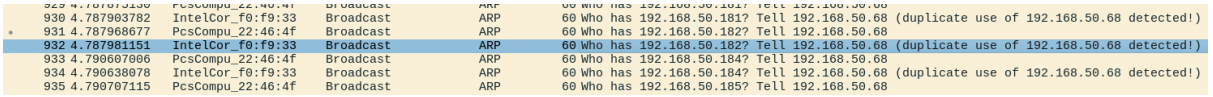
```
Nmap scan report for kali (192.168.50.68)
Host is up (0.000038s latency).
All 1000 scanned ports on kali (192.168.50.68) are in ignored states.
Not shown: 1000 closed tcp ports (reset)

Nmap done: 256 IP addresses (10 hosts up) scanned in 27.73 seconds
```

Skanowanie nie trwało długo - zaledwie 30 sekund



7	2.901598211	PcsCompu_22:46:4f	Broadcast	ARP	60 Who has 192.168.50.1? Tell 192.168.50.68
8	2.901615199	IntelCor_f0:f9:33	Broadcast	ARP	60 Who has 192.168.50.1? Tell 192.168.50.68 (duplicate use of 192.168.50.68 detected!)
9	2.901656963	PcsCompu_22:46:4f	Broadcast	ARP	60 Who has 192.168.50.2? Tell 192.168.50.68
10	2.901664114	IntelCor_f0:f9:33	Broadcast	ARP	60 Who has 192.168.50.2? Tell 192.168.50.68 (duplicate use of 192.168.50.68 detected!)
11	2.901683189	PcsCompu_22:46:4f	Broadcast	ARP	60 Who has 192.168.50.3? Tell 192.168.50.68
12	2.901688255	IntelCor_f0:f9:33	Broadcast	ARP	60 Who has 192.168.50.3? Tell 192.168.50.68 (duplicate use of 192.168.50.68 detected!)
13	2.901701967	PcsCompu_22:46:4f	Broadcast	ARP	60 Who has 192.168.50.4? Tell 192.168.50.68
14	2.901706628	IntelCor_f0:f9:33	Broadcast	ARP	60 Who has 192.168.50.4? Tell 192.168.50.68 (duplicate use of 192.168.50.68 detected!)
15	2.901723081	PcsCompu_22:46:4f	Broadcast	ARP	60 Who has 192.168.50.5? Tell 192.168.50.68
16	2.901728707	IntelCor_f0:f9:33	Broadcast	ARP	60 Who has 192.168.50.5? Tell 192.168.50.68 (duplicate use of 192.168.50.68 detected!)
17	2.901750126	PcsCompu_22:46:4f	Broadcast	ARP	60 Who has 192.168.50.6? Tell 192.168.50.68
18	2.901755225	IntelCor_f0:f9:33	Broadcast	ARP	60 Who has 192.168.50.6? Tell 192.168.50.68 (duplicate use of 192.168.50.68 detected!)
19	2.901773231	PcsCompu_22:46:4f	Broadcast	ARP	60 Who has 192.168.50.7? Tell 192.168.50.68
20	2.901778225	IntelCor_f0:f9:33	Broadcast	ARP	60 Who has 192.168.50.7? Tell 192.168.50.68 (duplicate use of 192.168.50.68 detected!)
21	2.901794819	PcsCompu_22:46:4f	Broadcast	ARP	60 Who has 192.168.50.8? Tell 192.168.50.68
22	2.901799825	IntelCor_f0:f9:33	Broadcast	ARP	60 Who has 192.168.50.8? Tell 192.168.50.68 (duplicate use of 192.168.50.68 detected!)
23	2.901813449	PcsCompu_22:46:4f	Broadcast	ARP	60 Who has 192.168.50.9? Tell 192.168.50.68
24	2.901818717	IntelCor_f0:f9:33	Broadcast	ARP	60 Who has 192.168.50.9? Tell 192.168.50.68 (duplicate use of 192.168.50.68 detected!)
25	2.901838540	PcsCompu_22:46:4f	Broadcast	ARP	60 Who has 192.168.50.10? Tell 192.168.50.68
26	2.901844079	IntelCor_f0:f9:33	Broadcast	ARP	60 Who has 192.168.50.10? Tell 192.168.50.68 (duplicate use of 192.168.50.68 detected!)

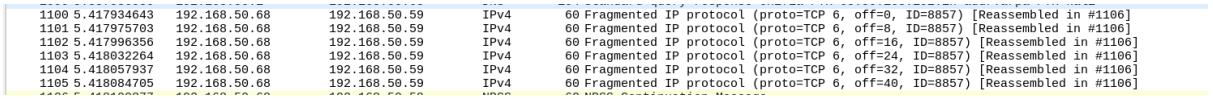
Zasada skanowania z początku działa bardzo podobnie do Stealth skanu (iteracja po adresach ip), z tą różnicą iż pojawiają się zduplikowane pakiety - jeden wysłany z komputera z kali linuxem a drugi z samej karty sieciowej
Warto zaznaczyć, że taka iteracja następowała więcej - a jeśli urządzenie już zostało *znalezione*, to odpowiadający mu adres ip został wykluczony z kolejnej iteracji



930	4.787903782	IntelCor_f0:f9:33	Broadcast	ARP	60 Who has 192.168.50.181? Tell 192.168.50.68 (duplicate use of 192.168.50.68 detected!)
931	4.787968677	PcsCompu_22:46:4f	Broadcast	ARP	60 Who has 192.168.50.182? Tell 192.168.50.68
932	4.787981151	IntelCor_f0:f9:33	Broadcast	ARP	60 Who has 192.168.50.182? Tell 192.168.50.68 (duplicate use of 192.168.50.68 detected!)
933	4.790607006	PcsCompu_22:46:4f	Broadcast	ARP	60 Who has 192.168.50.184? Tell 192.168.50.68
934	4.790638078	IntelCor_f0:f9:33	Broadcast	ARP	60 Who has 192.168.50.184? Tell 192.168.50.68 (duplicate use of 192.168.50.68 detected!)
935	4.790707115	PcsCompu_22:46:4f	Broadcast	ARP	60 Who has 192.168.50.185? Tell 192.168.50.68

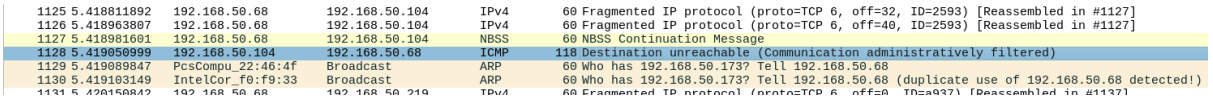
brakuje tutaj adresu 192.168.50.183 - adres komputera z windowsem

Następnie rozpoczęło się przesyłanie zfragmentowanych pakietów, w częściach



1100	5.417934643	192.168.50.68	192.168.50.59	IPv4	60 Fragmented IP protocol (proto=TCP 6, off=0, ID=8857) [Reassembled in #1106]
1101	5.417975703	192.168.50.68	192.168.50.59	IPv4	60 Fragmented IP protocol (proto=TCP 6, off=8, ID=8857) [Reassembled in #1106]
1102	5.417996356	192.168.50.68	192.168.50.59	IPv4	60 Fragmented IP protocol (proto=TCP 6, off=16, ID=8857) [Reassembled in #1106]
1103	5.418032264	192.168.50.68	192.168.50.59	IPv4	60 Fragmented IP protocol (proto=TCP 6, off=24, ID=8857) [Reassembled in #1106]
1104	5.418057937	192.168.50.68	192.168.50.59	IPv4	60 Fragmented IP protocol (proto=TCP 6, off=32, ID=8857) [Reassembled in #1106]
1105	5.418084705	192.168.50.68	192.168.50.59	IPv4	60 Fragmented IP protocol (proto=TCP 6, off=40, ID=8857) [Reassembled in #1106]

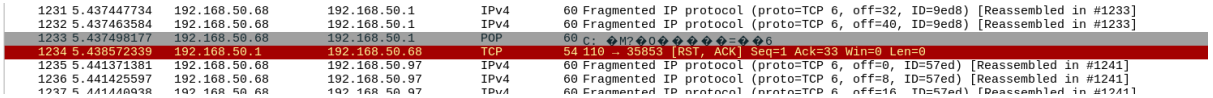
Co ciekawe, komputer z Fedorą nie dopuścił do komunikacji



1125	5.418811892	192.168.50.68	192.168.50.104	IPv4	60 Fragmented IP protocol (proto=TCP 6, off=32, ID=2593) [Reassembled in #1127]
1126	5.418963807	192.168.50.68	192.168.50.104	IPv4	60 Fragmented IP protocol (proto=TCP 6, off=40, ID=2593) [Reassembled in #1127]
1127	5.418981601	192.168.50.68	192.168.50.104	NBSS	60 NBSS Continuation Message
1128	5.419050999	192.168.50.104	192.168.50.68	ICMP	118 Destination unreachable (Communication administratively filtered)
1129	5.419089847	PcsCompu_22:46:4f	Broadcast	ARP	60 Who has 192.168.50.173? Tell 192.168.50.68
1130	5.419103149	IntelCor_f0:f9:33	Broadcast	ARP	60 Who has 192.168.50.173? Tell 192.168.50.68 (duplicate use of 192.168.50.68 detected!)

Najprawdopodobniej ma to związek z regułą bezpieczeństwa dostępną w systemie

Są wysyłane pakiety z dziwną zawartością



1231	5.437447734	192.168.50.68	192.168.50.1	IPv4	60 Fragmented IP protocol (proto=TCP 6, off=32, ID=9ed8) [Reassembled in #1233]
1232	5.437463584	192.168.50.68	192.168.50.1	IPv4	60 Fragmented IP protocol (proto=TCP 6, off=40, ID=9ed8) [Reassembled in #1233]
1233	5.437498177	192.168.50.68	192.168.50.1	POP	60 C: ◆M?◆O◆◆◆◆◆=◆◆6
1234	5.438572339	192.168.50.1	192.168.50.68	TCP	54 110 → 35853 [RST, ACK] Seq=1 Ack=33 Win=0 Len=0
1235	5.441371381	192.168.50.68	192.168.50.97	IPv4	60 Fragmented IP protocol (proto=TCP 6, off=0, ID=57ed) [Reassembled in #1241]
1236	5.441425597	192.168.50.68	192.168.50.97	IPv4	60 Fragmented IP protocol (proto=TCP 6, off=8, ID=57ed) [Reassembled in #1241]

Warto już zaznaczyć, że wysłanych pakietów jest znacznie więcej niż w przypadku flagi -sS

Również występuje protokół TCP, w którym występują liczne powtórzenia, co na pewno dało by o sobie znać w logach

2166	5.568540141	192.168.50.1	192.168.50.68	TCP	54 259 → 35853 [RST, ACK] Seq=1 Ack=33 Win=0 Len=0
2167	5.569026270	192.168.50.219	192.168.50.68	TCP	54 554 → 35853 [RST, ACK] Seq=1 Ack=33 Win=0 Len=0
2168	5.569635366	192.168.50.219	192.168.50.68	TCP	54 256 → 35853 [RST, ACK] Seq=1 Ack=33 Win=0 Len=0
2169	5.569854511	192.168.50.219	192.168.50.68	TCP	54 21 → 35853 [RST, ACK] Seq=1 Ack=33 Win=0 Len=0
2170	5.570420964	192.168.50.219	192.168.50.68	TCP	54 587 → 35853 [RST, ACK] Seq=1 Ack=33 Win=0 Len=0
2171	5.570751479	192.168.50.219	192.168.50.68	TCP	54 993 → 35853 [RST, ACK] Seq=1 Ack=33 Win=0 Len=0
2172	5.570751596	192.168.50.219	192.168.50.68	TCP	58 53 → 35853 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
2173	5.570751668	192.168.50.219	192.168.50.68	TCP	54 113 → 35853 [RST, ACK] Seq=1 Ack=33 Win=0 Len=0
2174	5.570751742	192.168.50.219	192.168.50.68	TCP	54 3389 → 35853 [RST, ACK] Seq=1 Ack=33 Win=0 Len=0
2175	5.570751815	192.168.50.219	192.168.50.68	TCP	54 23 → 35853 [RST, ACK] Seq=1 Ack=33 Win=0 Len=0
2176	5.570751889	192.168.50.219	192.168.50.68	TCP	54 135 → 35853 [RST, ACK] Seq=1 Ack=33 Win=0 Len=0
2177	5.570973131	192.168.50.219	192.168.50.68	TCP	58 80 → 35853 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
2178	5.570973269	192.168.50.219	192.168.50.68	TCP	54 199 → 35853 [RST, ACK] Seq=1 Ack=33 Win=0 Len=0
2179	5.570973342	192.168.50.219	192.168.50.68	TCP	54 5900 → 35853 [RST, ACK] Seq=1 Ack=33 Win=0 Len=0
2180	5.570973415	192.168.50.219	192.168.50.68	TCP	54 143 → 35853 [RST, ACK] Seq=1 Ack=33 Win=0 Len=0
2181	5.571076915	192.168.50.68	192.168.50.219	TCP	60 35853 → 53 [RST] Seq=1 Win=0 Len=0
2182	5.571226548	192.168.50.68	192.168.50.219	TCP	60 35853 → 80 [RST] Seq=1 Win=0 Len=0

Przez tą chwilę skanowania, udało się złapać aż prawie 66 tysięcy pakietów!

Statistics

Measurement	Captured
Packets	65979
Time span, s	59.085
Average pps	1116.7
Average packet size, B	67
Bytes	4393243
Average bytes/s	74 k
Average bits/s	594 k

Podsumowując, Stealth Scan okazał się być znacznie mniej widoczny dla osoby analizującej ruch sieciowy niż skanowanie szybkie

Zadanie 5 – Analiza pliku zawierającego dane pakietów z zainfekowanego komputera

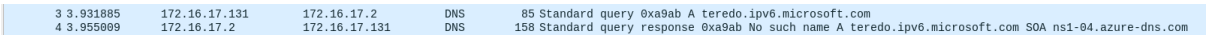
a. Podaj adres IP komputera, który został poddany analizie.

```
▾ Microsoft Windows Browser Protocol
    Command: Host Announcement (0x01)
    Update Count: 0
    Update Periodicity: 2 minutes
    Host Name: KOMPUTER
```

45	32.858300	172.16.17.131	172.16.17.255	BROWSER	243 Host Announcement KOMPUTER, Workstation, Server, NT Workstation

Najprawdopodobniej, poddany analizie, został komputer o nazwie *KOMPUTER*
z adresem ip: ***172.16.17.131***

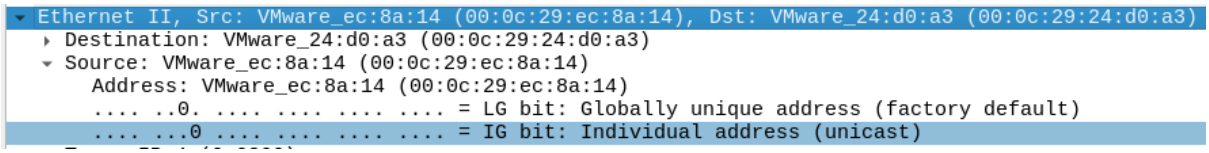
b. Podaj adres gatewaya tego komputera.



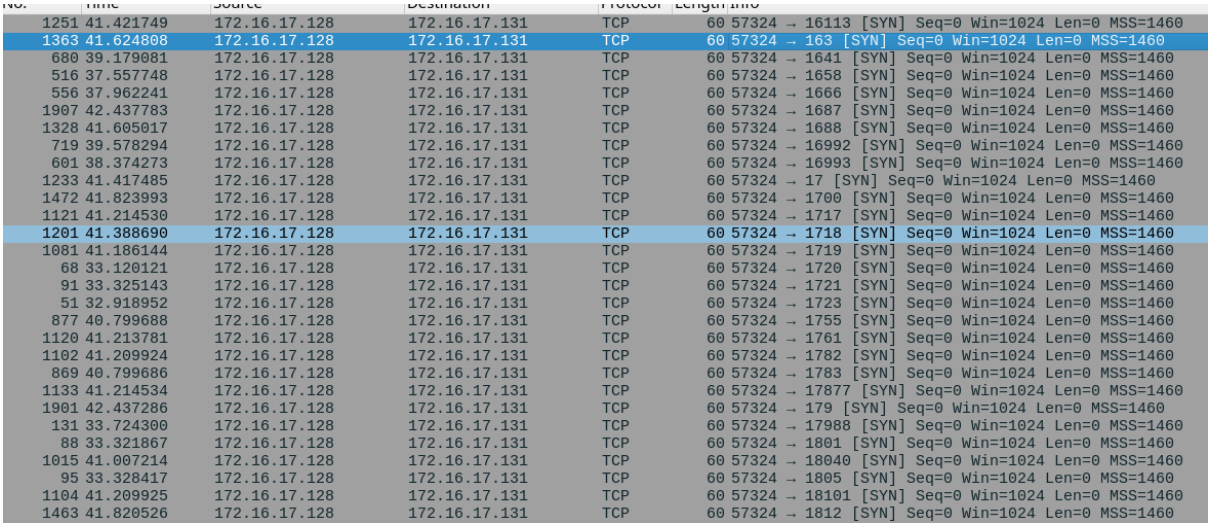
W powyższych pakietach następowało ustalanie obowiązującej domeny, więc najprawdopodobniej interesujący nas adres to ***172.16.17.132***

c. Czy przedstawione zdarzenie działo się w ramach wirtualnych maszyn? Na jakiej podstawie zostały wyciągnięte wnioski?

Tak, ponieważ w ramce warstwy 2, udało mi się znaleźć adres MAC świadczący o dostawcy - w tym przypadku programu VMware



d. Czy w trakcie działania zainfekowanego komputera jesteśmy w stanie określić, czy stacja była skanowana w sieci w poszukiwaniu otwartych portów?



No.	Time	Source	Destination	Protocol	Length	Info
1251	41.421749	172.16.17.128	172.16.17.131	TCP	60	57324 → 16113 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1363	41.624808	172.16.17.128	172.16.17.131	TCP	60	57324 → 163 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
680	39.179081	172.16.17.128	172.16.17.131	TCP	60	57324 → 1641 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
516	37.557748	172.16.17.128	172.16.17.131	TCP	60	57324 → 1658 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
556	37.962241	172.16.17.128	172.16.17.131	TCP	60	57324 → 1666 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1907	42.437783	172.16.17.128	172.16.17.131	TCP	60	57324 → 1687 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1328	41.605017	172.16.17.128	172.16.17.131	TCP	60	57324 → 1688 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
719	39.578294	172.16.17.128	172.16.17.131	TCP	60	57324 → 16992 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
601	38.374273	172.16.17.128	172.16.17.131	TCP	60	57324 → 16993 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1233	41.417485	172.16.17.128	172.16.17.131	TCP	60	57324 → 17 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1472	41.823993	172.16.17.128	172.16.17.131	TCP	60	57324 → 1700 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1121	41.214530	172.16.17.128	172.16.17.131	TCP	60	57324 → 1717 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1201	41.388690	172.16.17.128	172.16.17.131	TCP	60	57324 → 1718 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1081	41.186144	172.16.17.128	172.16.17.131	TCP	60	57324 → 1719 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
68	33.120121	172.16.17.128	172.16.17.131	TCP	60	57324 → 1720 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
91	33.325143	172.16.17.128	172.16.17.131	TCP	60	57324 → 1721 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
51	32.918952	172.16.17.128	172.16.17.131	TCP	60	57324 → 1723 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
877	40.799688	172.16.17.128	172.16.17.131	TCP	60	57324 → 1755 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1120	41.213781	172.16.17.128	172.16.17.131	TCP	60	57324 → 1761 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1102	41.209924	172.16.17.128	172.16.17.131	TCP	60	57324 → 1782 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
869	40.799686	172.16.17.128	172.16.17.131	TCP	60	57324 → 1783 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1133	41.214534	172.16.17.128	172.16.17.131	TCP	60	57324 → 17877 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1901	42.437286	172.16.17.128	172.16.17.131	TCP	60	57324 → 179 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
131	33.724300	172.16.17.128	172.16.17.131	TCP	60	57324 → 17988 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
88	33.321867	172.16.17.128	172.16.17.131	TCP	60	57324 → 1801 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1015	41.007214	172.16.17.128	172.16.17.131	TCP	60	57324 → 18040 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
95	33.328417	172.16.17.128	172.16.17.131	TCP	60	57324 → 1805 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1104	41.209925	172.16.17.128	172.16.17.131	TCP	60	57324 → 18101 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
1463	41.820526	172.16.17.128	172.16.17.131	TCP	60	57324 → 1812 [SYN] Seq=0 Win=1024 Len=0 MSS=1460

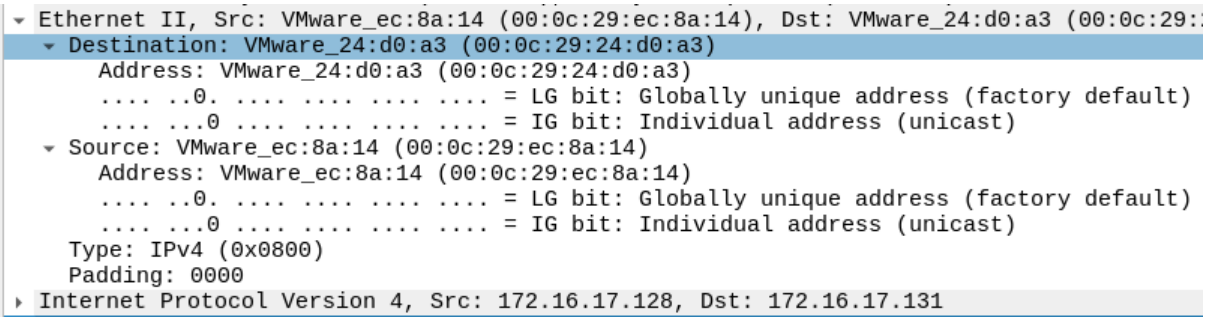
Myślę, że **tak**, ponieważ urządzenie o adresie ip 172.16.17.128 używało różnych portów docelowych (których było naprawdę wiele)

e. Jeśli tak, to przez kogo (IP sprawcy i jaką metodą), jeśli nie, to jakich informacji brakuje w badanym pliku?

IP sprawcy: **172.16.17.128**
Było wysyłanych wiele pakietów typu TCP SYN, mogła to być przykładowo flaga **-sS nmap'a**

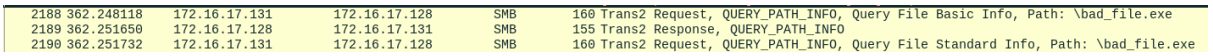
Struktura bardzo podobna do poprzednich zadań

f. W takcie działania zainfekowanego komputera został rozgłoszony ARP z adresem MAC (00:0c:29:ec:8a:14). Do kogo należy?



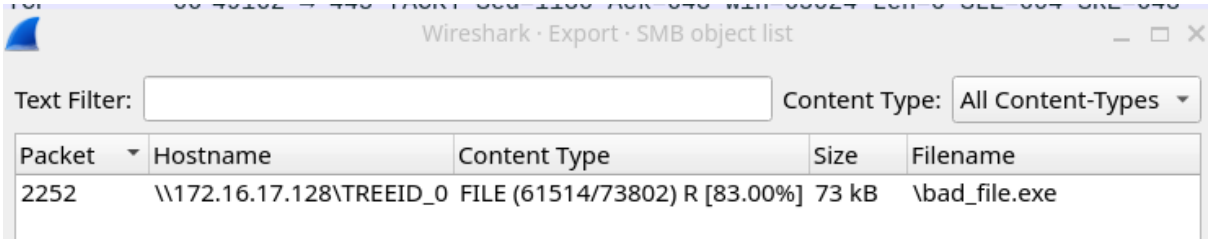
należy do adresu ip **172.16.17.128** - maszyny wirtualnej

g. Analizowane logi zawierają informacje o pliku wykonywalny exe. Sprawdź, kiedy został pobrany, z którego adresu i jak nazywa się plik?



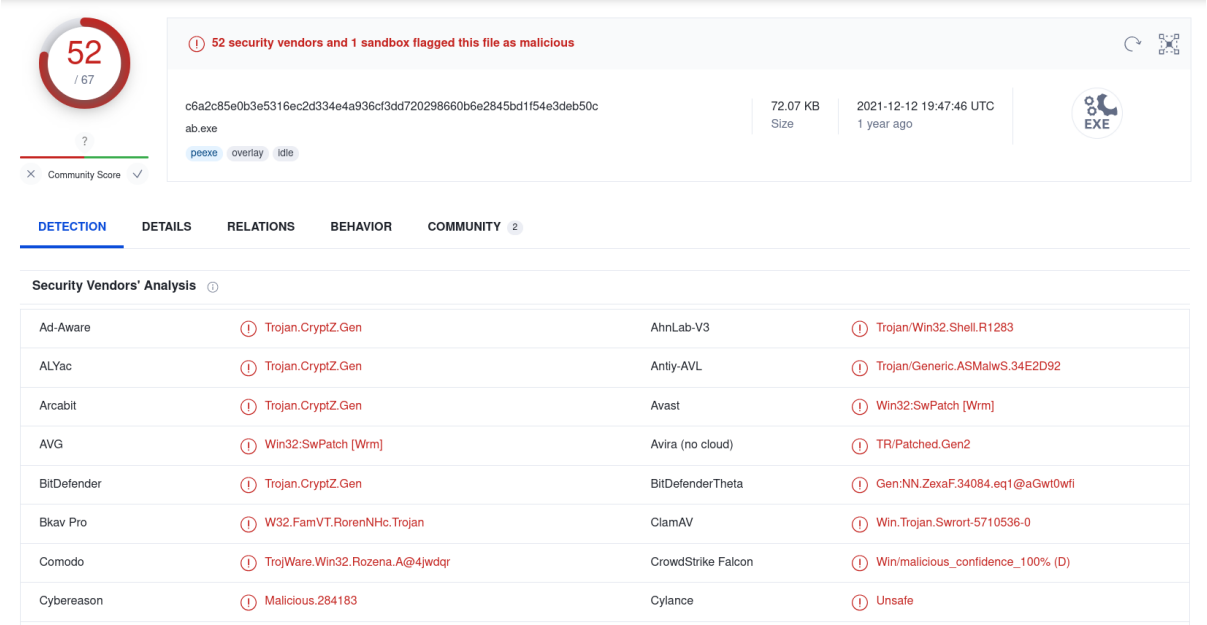
Został pobrany w **szóstej minucie** działania programu wireshark
nazwa: **bad_file.exe**
adres źródłowy: **172.16.17.128**
został przetransferowany za pomocą protokołu SMB

h. Przy użyciu opcji z Wireshark „Extract Object” wyciągnij odnaleziony plik, zapisz go w nowym folderze i przy pomocy narzędzia md5sum sprawdź jego sumę kontrolną.



md5sum: **a8910c628418380eed87b6e58ee61019**

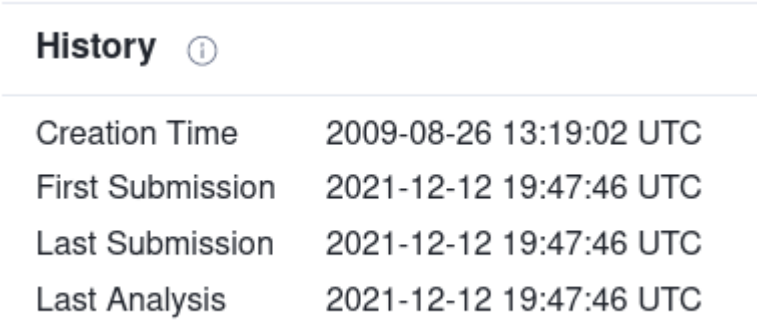
i. Pozyskaną sumę kontrolną wklej na stronie https://www.virustotal.com w zakładce search. Przedstaw i opisz wynik analizy.



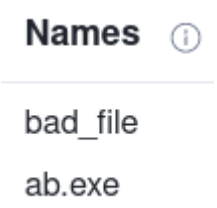
Jak widać, plik jest naprawdę groźny i jest ewidentnie trojanem

File type	Win32 EXE
Magic	PE32 executable for MS Windows (GUI) Intel 80386 32-bit
TrID	Win32 Executable MS Visual C++ (generic) (38.8%) | Microsoft Visual C++ compiled executable (generic) (20.5%) | Win64 Executable (generic) (13%) | Win32 Dynamic Link Library (generic) (8.1%) | Win16 NE executable (generic) (6.2%)
File size	72.07 KB (73802 bytes)

Jest napisany typowo pod system Windows



A został stworzony naprawdę dawno temu - w roku 2009, chociaż pierwsze dodanie do strony virustotal nastąpiło w 2021 roku

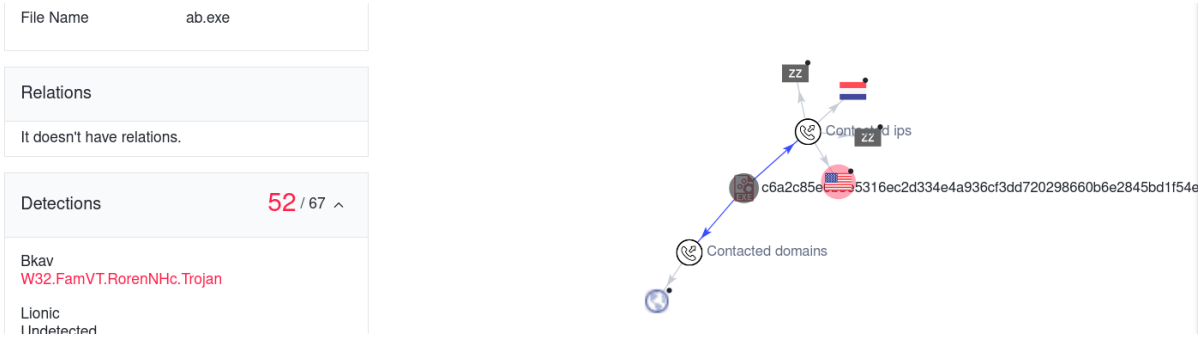


Plik występuje pod dwoja nazwami

File Version Information

Copyright	Copyright 2009 The Apache Software Foundation.
Product	Apache HTTP Server
Description	ApacheBench command line utility
Original Name	ab.exe
Internal Name	ab.exe
File Version	2.2.14
Comments	Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses /LICENSE-2.0 Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Ma związek z serwerem apache



Choć plik nie ma powiązania z konkretną stroną, organizacją

j. Który z portów był wykorzystywany do przesyłania danych pochodzących z ataku?

Były używane różne porty, choć głównie port **445** i **4444** (używany głównie właśnie przez Trojany)

2744	367.326013	172.16.17.128	172.16.17.131	TCP	182 4444 → 49165 [PSH, ACK] Seq=175694 Ack=910 Win=64128 Len=128
2745	367.333902	172.16.17.131	172.16.17.128	TCP	54 49162 → 445 [ACK] Seq=13112 Ack=150522 Win=65280 Len=0
2746	367.380930	172.16.17.131	172.16.17.128	TCP	198 49165 → 4444 [PSH, ACK] Seq=910 Ack=175822 Win=65024 Len=144
2747	367.382113	172.16.17.128	172.16.17.131	TCP	60 4444 → 49165 [ACK] Seq=175822 Ack=1054 Win=64128 Len=0
2748	367.494670	172.16.17.128	172.16.17.131	TCP	1514 4444 → 49165 [ACK] Seq=175822 Ack=1054 Win=64128 Len=1460
2749	367.494670	172.16.17.128	172.16.17.131	TCP	1514 4444 → 49165 [PSH, ACK] Seq=177282 Ack=1054 Win=64128 Len=1460
2750	367.494671	172.16.17.128	172.16.17.131	TCP	1514 4444 → 49165 [ACK] Seq=178742 Ack=1054 Win=64128 Len=1460
2751	367.494671	172.16.17.128	172.16.17.131	TCP	1514 4444 → 49165 [PSH, ACK] Seq=180202 Ack=1054 Win=64128 Len=1460
2752	367.494697	172.16.17.131	172.16.17.128	TCP	54 49165 → 4444 [ACK] Seq=1054 Ack=181662 Win=65536 Len=0
2753	367.495693	172.16.17.128	172.16.17.131	TCP	1514 4444 → 49165 [ACK] Seq=181662 Ack=1054 Win=64128 Len=1460
2754	367.495694	172.16.17.128	172.16.17.131	TCP	1514 4444 → 49165 [PSH, ACK] Seq=183122 Ack=1054 Win=64128 Len=1460
2755	367.495694	172.16.17.128	172.16.17.131	TCP	1514 4444 → 49165 [ACK] Seq=184582 Ack=1054 Win=64128 Len=1460
2756	367.495694	172.16.17.128	172.16.17.131	TCP	1514 4444 → 49165 [PSH, ACK] Seq=186042 Ack=1054 Win=64128 Len=1460
2757	367.495695	172.16.17.128	172.16.17.131	TCP	1514 4444 → 49165 [ACK] Seq=187502 Ack=1054 Win=64128 Len=1460
2758	367.495705	172.16.17.128	172.16.17.131	TCP	1514 4444 → 49165 [PSH, ACK] Seq=188962 Ack=1054 Win=64128 Len=1460
2759	367.495705	172.16.17.128	172.16.17.131	TCP	1514 4444 → 49165 [ACK] Seq=190422 Ack=1054 Win=64128 Len=1460
2760	367.495707	172.16.17.128	172.16.17.131	TCP	1514 4444 → 49165 [PSH, ACK] Seq=191882 Ack=1054 Win=64128 Len=1460
2761	367.495707	172.16.17.128	172.16.17.131	TCP	1514 4444 → 49165 [ACK] Seq=193342 Ack=1054 Win=64128 Len=1460
2762	367.495708	172.16.17.128	172.16.17.131	TCP	1514 4444 → 49165 [PSH, ACK] Seq=194802 Ack=1054 Win=64128 Len=1460
2763	367.495708	172.16.17.128	172.16.17.131	TCP	1514 4444 → 49165 [ACK] Seq=196262 Ack=1054 Win=64128 Len=1460
2764	367.495708	172.16.17.128	172.16.17.131	TCP	1514 4444 → 49165 [PSH, ACK] Seq=197722 Ack=1054 Win=64128 Len=1460
2765	367.495709	172.16.17.128	172.16.17.131	TCP	1514 4444 → 49165 [ACK] Seq=199182 Ack=1054 Win=64128 Len=1460

2462	366.959537	172.16.17.128	172.16.17.131	TCP	1514 445 → 49162 [ACK] Seq=89710 Ack=9292 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
2463	366.959538	172.16.17.128	172.16.17.131	TCP	1514 445 → 49162 [ACK] Seq=91170 Ack=9292 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
2464	366.959539	172.16.17.128	172.16.17.131	TCP	1514 445 → 49162 [ACK] Seq=92630 Ack=9292 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
2465	366.959539	172.16.17.128	172.16.17.131	TCP	1514 445 → 49162 [ACK] Seq=94090 Ack=9292 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
2466	366.959539	172.16.17.128	172.16.17.131	TCP	1514 445 → 49162 [PSH, ACK] Seq=95550 Ack=9292 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
2467	366.959540	172.16.17.128	172.16.17.131	TCP	1514 445 → 49162 [ACK] Seq=97010 Ack=9292 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
2468	366.959540	172.16.17.128	172.16.17.131	TCP	1514 445 → 49162 [ACK] Seq=98470 Ack=9292 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
2469	366.959540	172.16.17.128	172.16.17.131	TCP	1514 445 → 49162 [ACK] Seq=99930 Ack=9292 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
2470	366.959541	172.16.17.128	172.16.17.131	TCP	1514 445 → 49162 [ACK] Seq=101390 Ack=9292 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
2471	366.959542	172.16.17.128	172.16.17.131	TCP	1514 445 → 49162 [PSH, ACK] Seq=102850 Ack=9292 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
2472	366.959562	172.16.17.131	172.16.17.128	TCP	54 49162 → 445 [ACK] Seq=9292 Ack=104310 Win=65536 Len=0
2473	366.959935	172.16.17.128	172.16.17.131	TCP	1514 445 → 49162 [ACK] Seq=104310 Ack=9292 Win=64128 Len=1460 [TCP segment of a reassembled PDU]

k. Podaj nazwę komputera, który został zaatakowany.

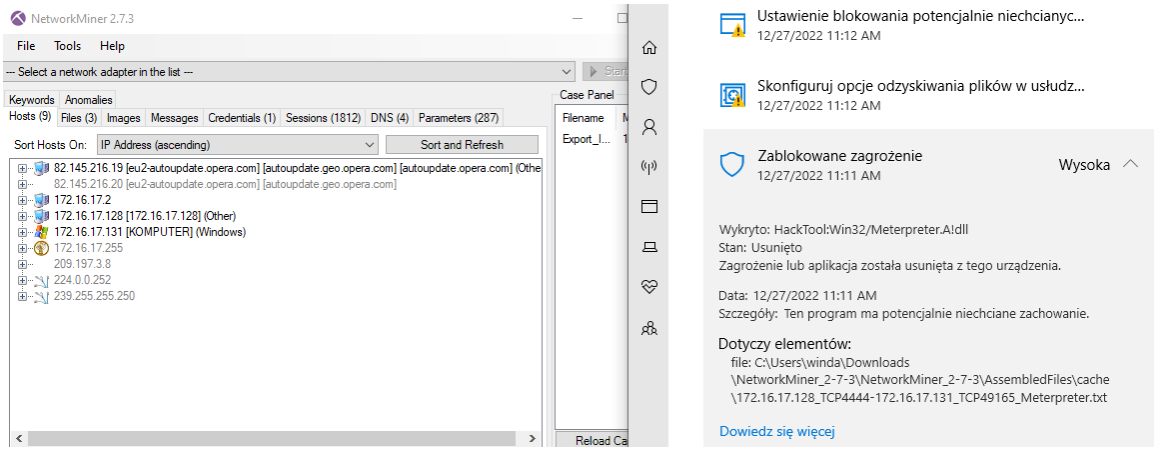
nazwa: *KOMPUTER*

z adresem ip: ***172.16.17.131***

45	32.858300	172.16.17.131	172.16.17.255	BROWSER	243 Host Announcement KOMPUTER, Workstation, Server, NT Workstation

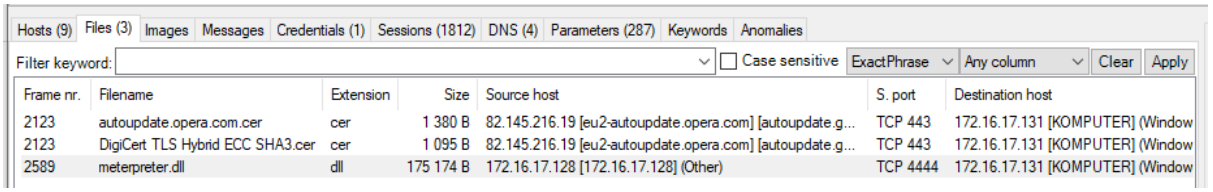
Zadanie 6 – NetworkMiner jako alternatywny program do analizy ruchu sieciowego

Po zainstalowaniu programu i załadowaniu pliku z wiresharka, jesteśmy witani powiadomieniem o zagrożeniu



Najprawdopodobniej program NetworkMiner próbował wydobyć plik bad_file.exe co zostało zakończone zablokowaniem pliku przez Windows Defendera

Niestety, w samej zakładce *files*, nie widać interesującego nas pliku, mimo iż program NetworkMiner wspiera odzyskiwanie plików za pomocą protokołu SMB



Myśle, że ma to związek z tym, że plik był pobierany w częściach a nie całości. Lub ewentualnie z tym, że jest groźny i program domyślnie blokuje złośliwe pliki

Choć ślady pliku dostępne są w zakładce *parameters*