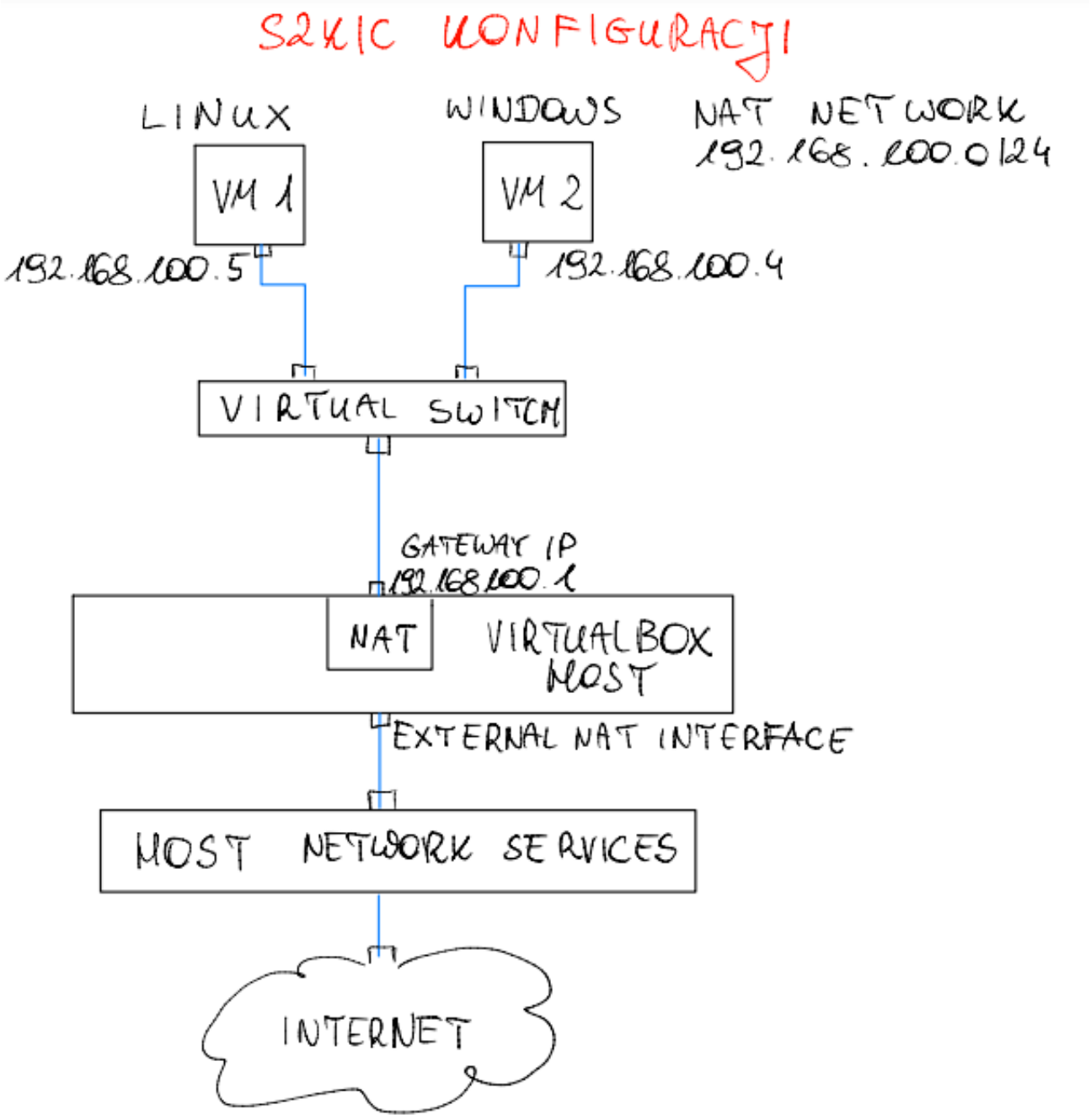
Informatyka śledcza Laboratorium nr 7

Raport – Nikodem Jakubowski

Zadanie 1 – Przygotowanie środowiska testowego.

Przedstawię tutaj szkic środowiska, które ostatecznie wykorzystałem w laboratorium. Początkowo wszystko miałem zrobione w domyślnej instancji NAT i zrezygnowałem z tego, bo urządzenia się nie widziały. Ostateczny szkic.



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

Użycie ifconfig na maszynie z linuxem.

```
┌──(user㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fe28:cbbb  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:28:cb:bb  txqueuelen 1000  (Ethernet)
        RX packets 1  bytes 590 (590.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 20  bytes 2910 (2.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Sprawdzenie tablicy routingu.

```
┌──(user㉿kali)-[~]
└─$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.2.2        0.0.0.0         UG    100    0        0 eth0
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 eth0
```

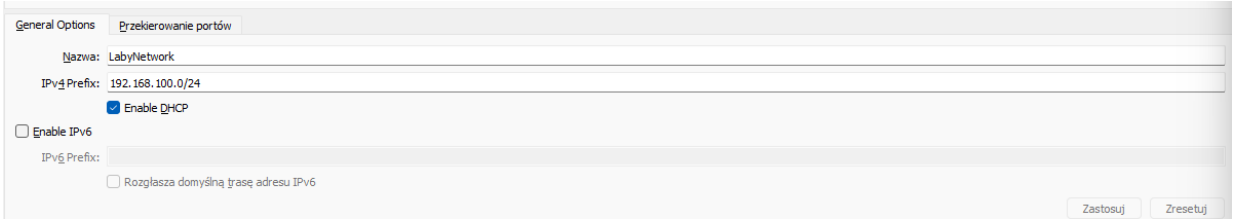
VirtualBox maszynie z Windowsem przydziela taki sam adres, obie maszyny korzystają z tego samego gateway. Jak przeczytałem w dokumentacji – to normalne, VirtualBox podobno ma kilka osobnych adapterów dla każdej maszyny…

```
C:\Users\user>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : home
   Link-local IPv6 Address . . . . . : fe80::78e0:a1fd:d163:84b2%7
   IPv4 Address. . . . . . . . . . . : 10.0.2.15
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.0.2.2
```

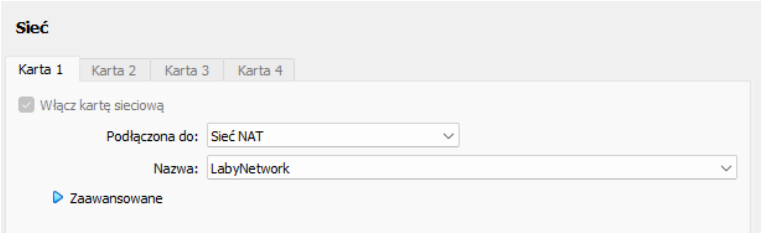
W tej sytuacji maszyny na pewno nie będą się widziały, zmiana planów. Poszukałem w dokumentacji i jest opcja stworzenia NAT Network.

Aby maszyny się widziały, stworzę osobną sieć NAT w VirtualBox.

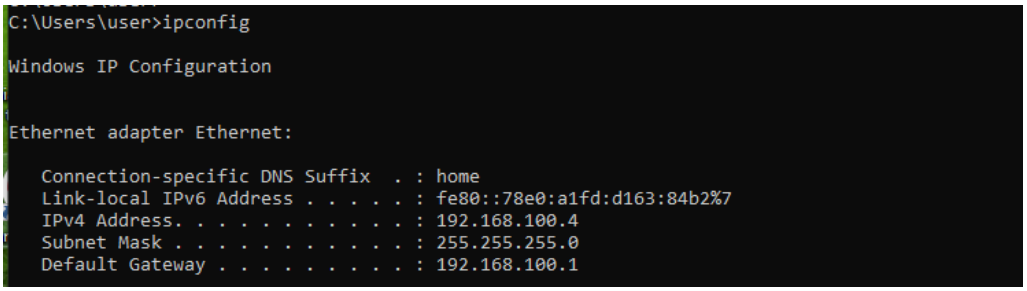
Konfiguracja sieci, oczywiście wybieramy adres z puli prywatnej. Poniżej nowa sieć NAT stworzona przeze mnie. Nazwa to LabyNetwork, a adres 192.168.100.0/24.



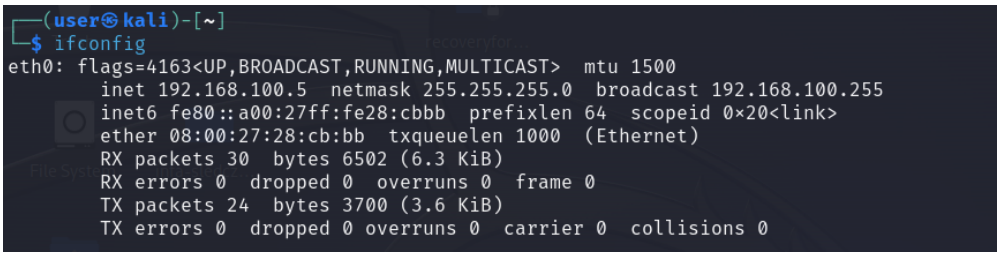
Następnie na obu maszynach w ustawieniach karty zmieniamy na sieć NAT, konkretnie - LabyNetwork.



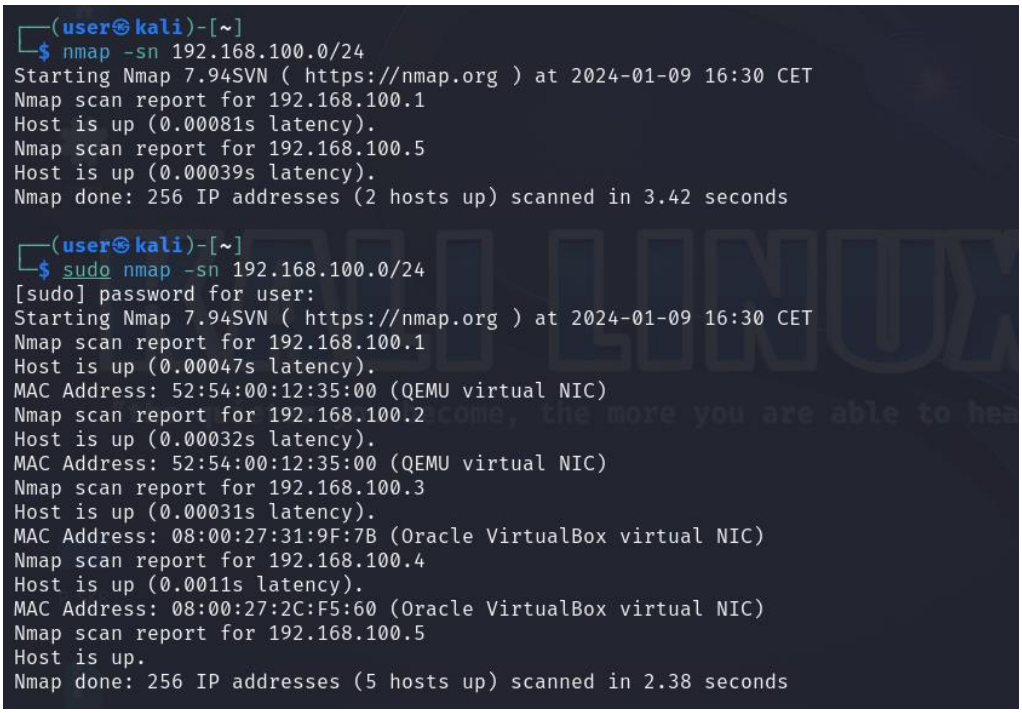
Nowy adres maszyny z Windows.



Po reboot maszyna wirtualna dostała adres.



Skanuję przy pomocy nmap. Dla porównania, przy pomocy sudo w ogóle coś widać!



Zadanie 3 – Analiza ruchu sieciowego przy wykorzystaniu narzędzia TCPdump.

Przesłałem ping od maszyny z Windows do maszyny z Linuxem i przechwyciłem garść informacji.

```
┌──(user㉿kali)-[~]
└─$ sudo tcpdump -i eth0 -v
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
16:33:26.310637 IP (tos 0×0, ttl 128, id 45538, offset 0, flags [none], proto ICMP (1), lengt
h 60)
    192.168.100.4 > 192.168.100.5: ICMP echo request, id 1, seq 28, length 40
16:33:26.310689 IP (tos 0×0, ttl 64, id 41050, offset 0, flags [none], proto ICMP (1), length
 60)
    192.168.100.5 > 192.168.100.4: ICMP echo reply, id 1, seq 28, length 40
16:33:26.344353 IP (tos 0×0, ttl 64, id 42809, offset 0, flags [DF], proto UDP (17), length 7
2)
    192.168.100.5.59125 > funbox.home.domain: 4357+ PTR? 5.100.168.192.in-addr.arpa. (44)
16:33:26.348731 IP (tos 0×0, ttl 255, id 2456, offset 0, flags [none], proto UDP (17), length
 72)
    funbox.home.domain > 192.168.100.5.59125: 4357 NXDomain* 0/0/0 (44)
16:33:26.348811 IP (tos 0×0, ttl 64, id 2068, offset 0, flags [DF], proto UDP (17), length 72
)
    192.168.100.5.60186 > funbox.home.domain: 57879+ PTR? 4.100.168.192.in-addr.arpa. (44)
16:33:26.353255 IP (tos 0×0, ttl 255, id 2457, offset 0, flags [none], proto UDP (17), length
 72)
    funbox.home.domain > 192.168.100.5.60186: 57879 NXDomain* 0/0/0 (44)
16:33:26.438254 IP (tos 0×0, ttl 64, id 21566, offset 0, flags [DF], proto UDP (17), length 7
0)
    192.168.100.5.33567 > funbox.home.domain: 38644+ PTR? 1.1.168.192.in-addr.arpa. (42)
16:33:26.441688 IP (tos 0×0, ttl 255, id 2458, offset 0, flags [none], proto UDP (17), length
 95)
    funbox.home.domain > 192.168.100.5.33567: 38644* 1/0/0 1.1.168.192.in-addr.arpa. PTR funb
ox.home. (67)
16:33:27.355426 IP (tos 0×0, ttl 128, id 45539, offset 0, flags [none], proto ICMP (1), lengt
h 60)
```

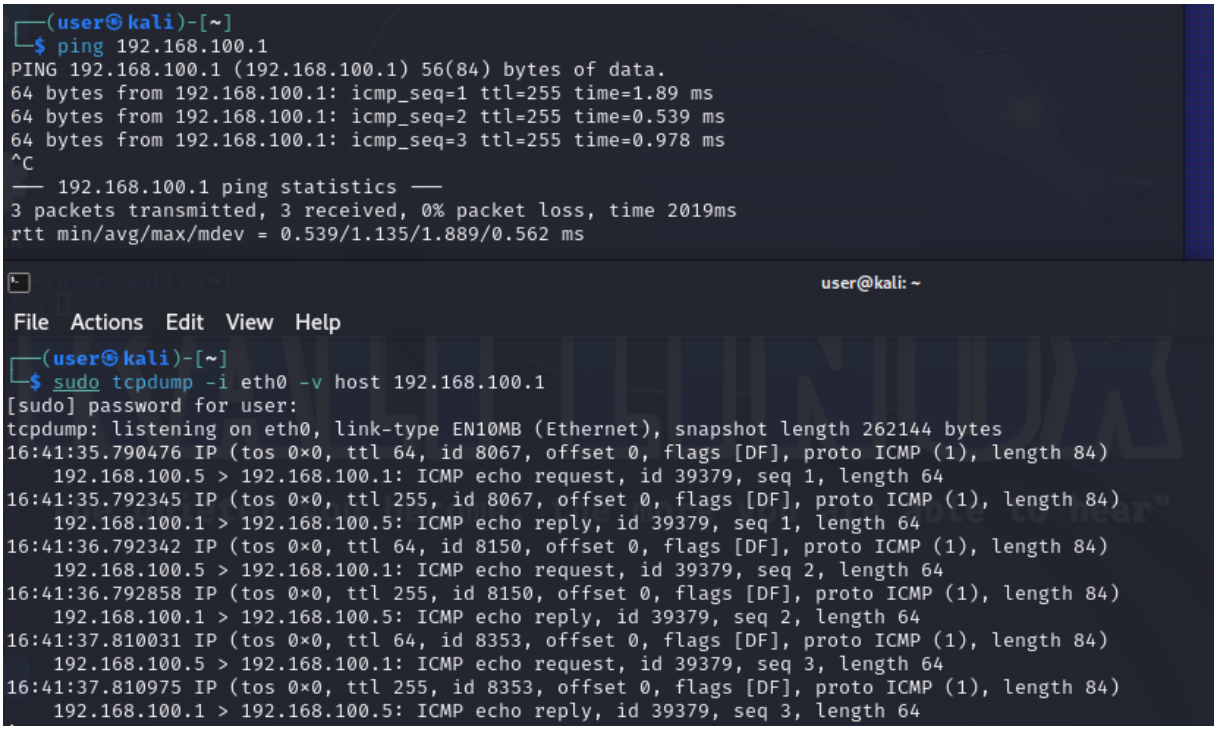
```
61 packets captured
61 packets received by filter
0 packets dropped by kernel
```

Udało się przechwycić oczywiście zapytanie ARP, które idzie przez broadcast, protokół ICMP odpowiedzialny za ping, adres wysyłającego oraz router, z którym komunikuje się urządzenie (funbox).

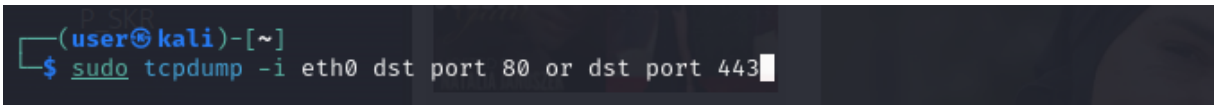
Pinguję bramę domyślną, wszystko jest w porządku.

```
┌──(user㉿kali)-[~]
└─$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.100.1   0.0.0.0         UG    100    0        0 eth0
192.168.100.0   0.0.0.0         255.255.255.0   U     100    0        0 eth0

┌──(user㉿kali)-[~]
└─$ ping 192.168.100.1
PING 192.168.100.1 (192.168.100.1) 56(84) bytes of data.
64 bytes from 192.168.100.1: icmp_seq=1 ttl=255 time=0.553 ms
64 bytes from 192.168.100.1: icmp_seq=2 ttl=255 time=1.58 ms
64 bytes from 192.168.100.1: icmp_seq=3 ttl=255 time=0.664 ms
64 bytes from 192.168.100.1: icmp_seq=4 ttl=255 time=2.04 ms
64 bytes from 192.168.100.1: icmp_seq=5 ttl=255 time=0.882 ms
64 bytes from 192.168.100.1: icmp_seq=6 ttl=255 time=0.712 ms
64 bytes from 192.168.100.1: icmp_seq=7 ttl=255 time=4.01 ms
64 bytes from 192.168.100.1: icmp_seq=8 ttl=255 time=1.20 ms
^C
── 192.168.100.1 ping statistics ──
8 packets transmitted, 8 received, 0% packet loss, time 7100ms
rtt min/avg/max/mdev = 0.553/1.453/4.007/1.075 ms
```

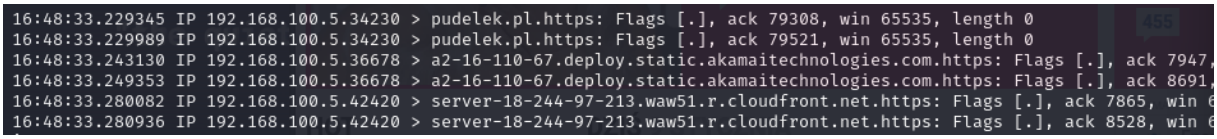
Otworzyłem sobie dwa terminale naraz, żeby przefiltrować ping.



Użyłem takiej komendy, żeby przechwycić wyświetlenie strony pudelek.pl. Przeskakuje na raz dużo linijek, więc trzeba było szybko zatrzymać.



Efekt.



Zadanie 4 – Analiza ruchu sieciowego przy wykorzystaniu programu Wireshark.

Używam nmap -Ss.

```
┌──(user㉿kali)-[~]
└─$ sudo nmap -sS 192.168.100.0/24
[sudo] password for user:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-09 17:03 CET
Nmap scan report for 192.168.100.1
Host is up (0.0018s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE
53/tcp open  domain
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)

Nmap scan report for 192.168.100.2
Host is up (0.0029s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT    STATE SERVICE
135/tcp open  msrpc
445/tcp open  microsoft-ds
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)

Nmap scan report for 192.168.100.3
Host is up (0.00028s latency).
All 1000 scanned ports on 192.168.100.3 are in ignored states.
Not shown: 1000 filtered tcp ports (proto-unreach)
MAC Address: 08:00:27:31:9F:7B (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.100.4
Host is up (0.0099s latency).
All 1000 scanned ports on 192.168.100.4 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)
MAC Address: 08:00:27:2C:F5:60 (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.100.5
Host is up (0.0000070s latency).
All 1000 scanned ports on 192.168.100.5 are in ignored states.
Not shown: 1000 closed tcp ports (reset)

Nmap done: 256 IP addresses (5 hosts up) scanned in 16.01 seconds
```

Po pierwsze widzimy, że host 192.168.100.5 (Linux) wysyła zapytania ARP, żeby rozeznać się w sieci.

No.	Time	Source	Destination	Protocol	Lengt	Info
1	0.000000	PCSSystemtec_28:cb:…	Broadcast	ARP	60	Who has 192.168.100.1? Tell 192.168.100.5
2	0.000000	PCSSystemtec_28:cb:…	Broadcast	ARP	60	Who has 192.168.100.2? Tell 192.168.100.5
3	0.016576	PCSSystemtec_28:cb:…	Broadcast	ARP	60	Who has 192.168.100.3? Tell 192.168.100.5
4	0.016576	PCSSystemtec_28:cb:…	Broadcast	ARP	60	Who has 192.168.100.4? Tell 192.168.100.5
5	0.016576	PCSSystemtec_28:cb:…	Broadcast	ARP	60	Who has 192.168.100.6? Tell 192.168.100.5
6	0.016576	PCSSystemtec_28:cb:…	Broadcast	ARP	60	Who has 192.168.100.7? Tell 192.168.100.5
7	0.016576	PCSSystemtec_28:cb:…	Broadcast	ARP	60	Who has 192.168.100.8? Tell 192.168.100.5
8	0.016576	PCSSystemtec_28:cb:…	Broadcast	ARP	60	Who has 192.168.100.9? Tell 192.168.100.5
9	0.016576	PCSSystemtec_28:cb:…	Broadcast	ARP	60	Who has 192.168.100.10? Tell 192.168.100.5
10	0.016576	PCSSystemtec_28:cb:…	Broadcast	ARP	60	Who has 192.168.100.11? Tell 192.168.100.5
11	0.016606	PCSSystemtec_2c:f5:…	PCSSystemtec_28:cb:…	ARP	42	192.168.100.4 is at 08:00:27:2c:f5:60
12	0.048847	PCSSystemtec_28:cb:…	Broadcast	ARP	60	Who has 192.168.100.14? Tell 192.168.100.5
13	0.048847	PCSSystemtec_28:cb:…	Broadcast	ARP	60	Who has 192.168.100.15? Tell 192.168.100.5
14	0.048847	PCSSystemtec_28:cb:…	Broadcast	ARP	60	Who has 192.168.100.16? Tell 192.168.100.5
15	0.048847	PCSSystemtec_28:cb:…	Broadcast	ARP	60	Who has 192.168.100.17? Tell 192.168.100.5
16	0.048847	PCSSystemtec_28:cb:…	Broadcast	ARP	60	Who has 192.168.100.18? Tell 192.168.100.5
17	0.048847	PCSSystemtec_28:cb:…	Broadcast	ARP	60	Who has 192.168.100.19? Tell 192.168.100.5
18	0.048847	PCSSystemtec_28:cb:…	Broadcast	ARP	60	Who has 192.168.100.20? Tell 192.168.100.5
19	0.048847	PCSSystemtec_28:cb:…	Broadcast	ARP	60	Who has 192.168.100.21? Tell 192.168.100.5
20	0.104023	PCSSystemtec_28:cb:…	Broadcast	ARP	60	Who has 192.168.100.24? Tell 192.168.100.5
21	0.104419	PCSSystemtec_28:cb:…	Broadcast	ARP	60	Who has 192.168.100.25? Tell 192.168.100.5
22	0.104419	PCSSystemtec_28:cb:…	Broadcast	ARP	60	Who has 192.168.100.26? Tell 192.168.100.5
23	0.104927	PCSSystemtec_28:cb:…	Broadcast	ARP	60	Who has 192.168.100.27? Tell 192.168.100.5
24	0.104927	PCSSystemtec_28:cb:…	Broadcast	ARP	60	Who has 192.168.100.28? Tell 192.168.100.5
25	0.104927	PCSSystemtec_28:cb:…	Broadcast	ARP	60	Who has 192.168.100.29? Tell 192.168.100.5
26	0.157866	PCSSystemtec_28:cb:…	Broadcast	ARP	60	Who has 192.168.100.32? Tell 192.168.100.5

Następnie, co typowe dla wyszukiwania sS (stealth scan), host Linuxowy wysyła tylko inicjalizującą część „3-way handshake”, czyli SYN. Nasza maszyna odpowiada SYN-ACK (czego się domyślamy), a host Linuxowy w ostatniej chwili się rozmyśla i nie odpowiada ACK (nie nawiązuje pełnego połączenia).

Użycie nmap z fragmentacją.

```
┌──(user㉿kali)-[~]
└─$ sudo nmap 192.168.100.0/24 -data-length 32 -f -T5
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-09 17:20 CET
Nmap scan report for 192.168.100.1
Host is up (0.0031s latency).
All 1000 scanned ports on 192.168.100.1 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)

Nmap scan report for 192.168.100.2
Host is up (0.0028s latency).
All 1000 scanned ports on 192.168.100.2 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)

Nmap scan report for 192.168.100.3
Host is up (0.00069s latency).
All 1000 scanned ports on 192.168.100.3 are in ignored states.
Not shown: 1000 filtered tcp ports (proto-unreach)
MAC Address: 08:00:27:31:9F:7B (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.100.4
Host is up (0.0018s latency).
All 1000 scanned ports on 192.168.100.4 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)
MAC Address: 08:00:27:2C:F5:60 (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.100.5
Host is up (0.000062s latency).
All 1000 scanned ports on 192.168.100.5 are in ignored states.
Not shown: 1000 closed tcp ports (reset)

Nmap done: 256 IP addresses (5 hosts up) scanned in 11.75 seconds
```

Efekt.

No.	Time	Source	Destination	Protocol	Length	Info
655	34.209968	192.168.100.5	192.168.100.4	IPv4	60	Fragmented IP protocol (proto=TCP 6, off=40, ID=de13) [Reas
656	34.209968	192.168.100.5	192.168.100.4	VNC	60	
657	34.210429	192.168.100.5	192.168.100.4	IPv4	60	Fragmented IP protocol (proto=TCP 6, off=0, ID=3741) [Reass
658	34.210429	192.168.100.5	192.168.100.4	IPv4	60	Fragmented IP protocol (proto=TCP 6, off=8, ID=3741) [Reass
659	34.210429	192.168.100.5	192.168.100.4	IPv4	60	Fragmented IP protocol (proto=TCP 6, off=16, ID=3741) [Reas
660	34.210429	192.168.100.5	192.168.100.4	IPv4	60	Fragmented IP protocol (proto=TCP 6, off=24, ID=3741) [Reas
661	34.210429	192.168.100.5	192.168.100.4	IPv4	60	Fragmented IP protocol (proto=TCP 6, off=32, ID=3741) [Reas
662	34.210863	192.168.100.5	192.168.100.4	IPv4	60	Fragmented IP protocol (proto=TCP 6, off=40, ID=3741) [Reas
663	34.210863	192.168.100.5	192.168.100.4	TCP	60	62581 → 80 [SYN] Seq=0 Win=1024 Len=32 MSS=1460
664	34.211208	192.168.100.5	192.168.100.4	IPv4	60	Fragmented IP protocol (proto=TCP 6, off=0, ID=44a5) [Reass
665	34.211208	192.168.100.5	192.168.100.4	IPv4	60	Fragmented IP protocol (proto=TCP 6, off=8, ID=44a5) [Reass
666	34.211208	192.168.100.5	192.168.100.4	IPv4	60	Fragmented IP protocol (proto=TCP 6, off=16, ID=44a5) [Reas
667	34.211208	192.168.100.5	192.168.100.4	IPv4	60	Fragmented IP protocol (proto=TCP 6, off=24, ID=44a5) [Reas
668	34.211623	192.168.100.5	192.168.100.4	IPv4	60	Fragmented IP protocol (proto=TCP 6, off=32, ID=44a5) [Reas
669	34.211623	192.168.100.5	192.168.100.4	IPv4	60	Fragmented IP protocol (proto=TCP 6, off=40, ID=44a5) [Reas
670	34.211623	192.168.100.5	192.168.100.4	TCP	60	62581 → 3389 [SYN] Seq=0 Win=1024 Len=32 MSS=1460
671	34.214934	192.168.100.5	192.168.100.4	IPv4	60	Fragmented IP protocol (proto=TCP 6, off=0, ID=f8c3) [Reass

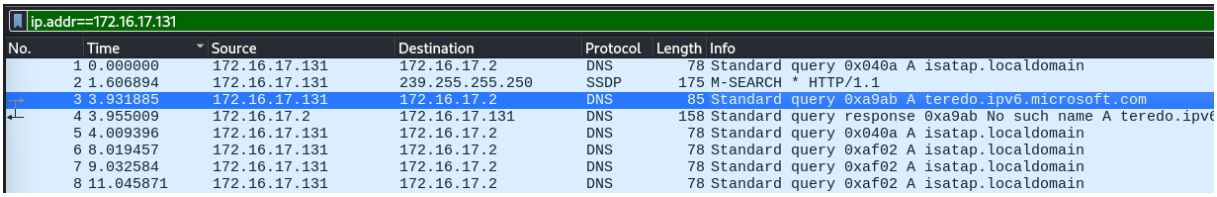
Skanowanie w ten sposób bardzo rzuca się w oczy w wiresharku w momencie przechwytywania. Powoduje to ilość rekordów jak i ich kolor. Niektóre z tych pakietów są zniekształcone albo mają podejrzaną zawartość. Ta metoda mi się wydaje o wiele bardziej widoczna i łatwiejsza do wykrycia „na żywo”, ale gdyby ktoś grzebał w .pcap z takiego urządzenia po dłuższym czasie, to mógłby tego nawet nie zauważyć.

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

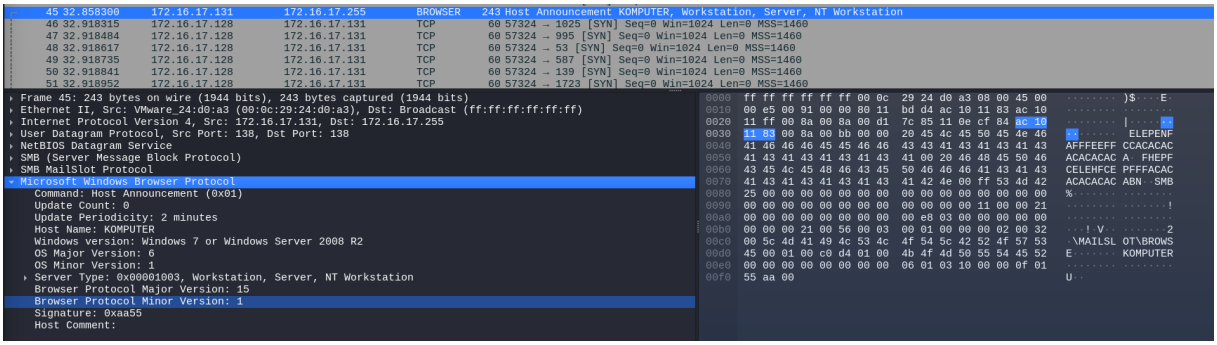
Poniżej dodaję odpowiedzi do odpowiednich podpunktów.

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

Myślę, że jest to adres 172.16.17.131. Pojawia się on bardzo często i wykonuje typowe czynności dla hosta jak zapytania DNS. Równie często pojawia się adres 172.16.17.128, ale on raczej dokonywał jakiegoś skanowania typu sS.

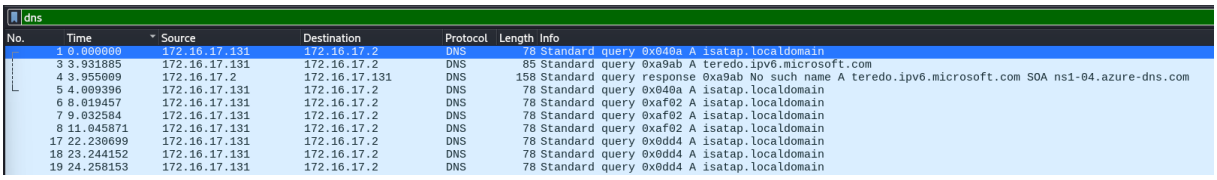


Dodatkowo mamy HostAnnouncement dla przeglądarki z adresu 172.16.17.131.



b. Podaj adres gatewaya tego komputera.

Gateway to 172.16.17.2.

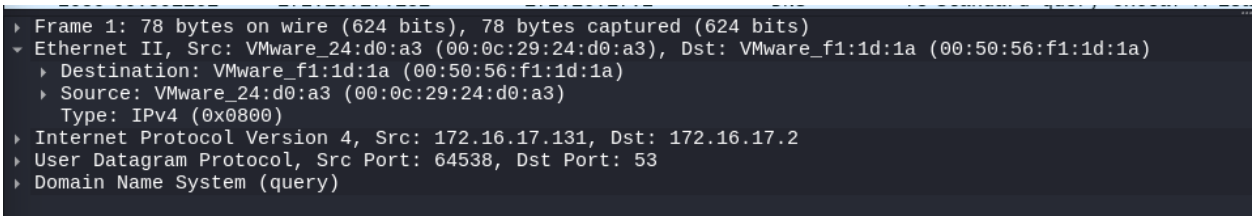


Widać to po komunikacji DNS hosta z bramą.

c. Czy przedstawione zdarzenie działo się w ramach wirtualnych maszyn?

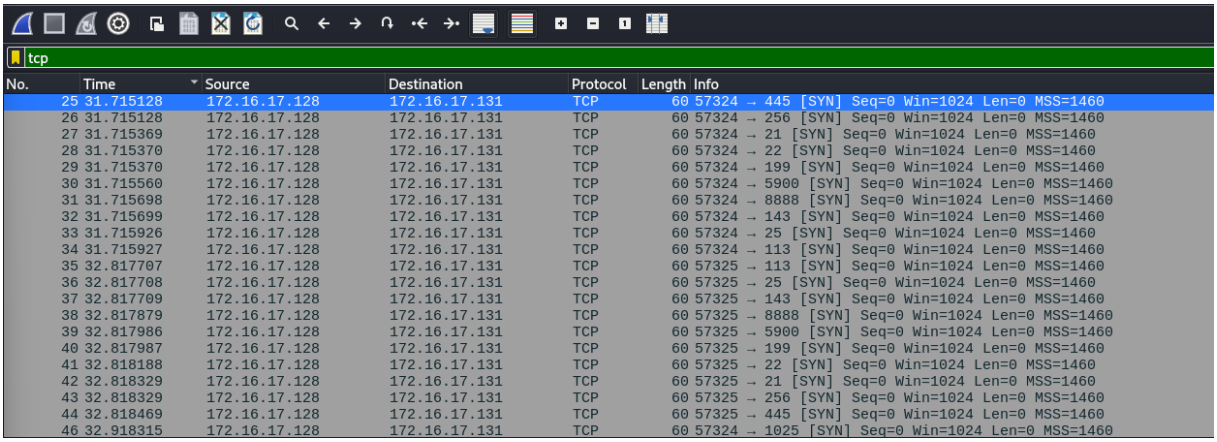
Na jakiej podstawie zostały wyciągnięte wnioski?

Tak zdecydowanie w obrębie maszyn wirtualnych. Można to zauważyć w pakietach ethernetowych, których opis podaję.



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

Tak, zdecydowanie widać, że maszyna 172.16.17.128 wykonywała coś w rodzaju stealth scan i badała, na których portach dostanie odpowiedź. W kolumnie info na samym początku są różne porty docelowe.

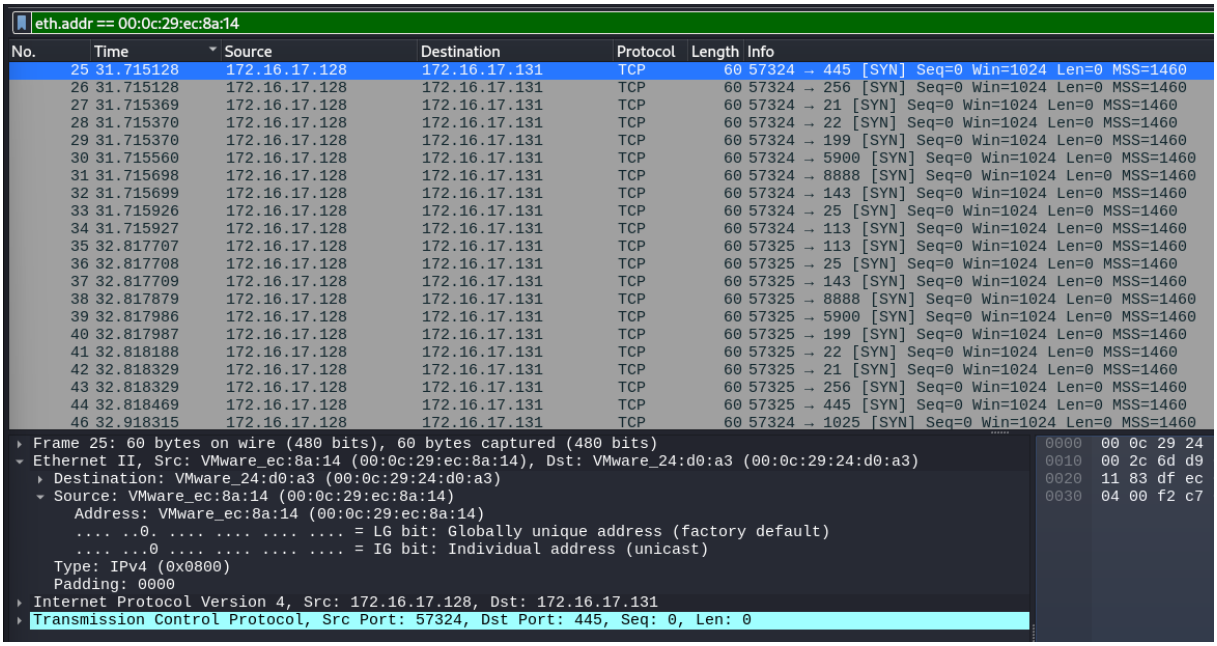


e. Jeśli tak, to przez kogo (IP sprawcy i jaką metodą), jeśli nie, to jakich informacji brakuje w badanym pliku?

Opisałem to już powyżej w rozważaniach. Adres hosta 172.16.17.128 od początku był podejrzany.

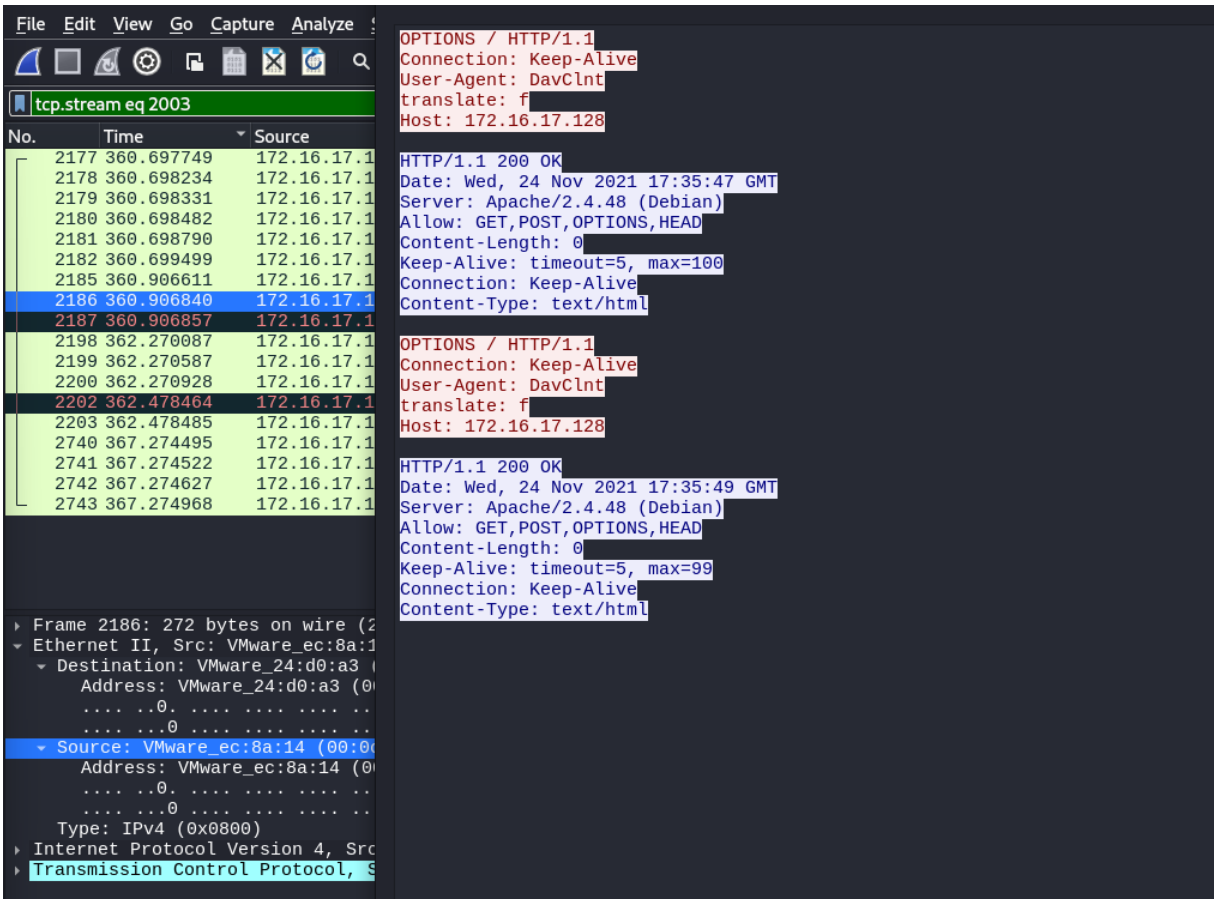
f. W takcie działania zainfekowanego komputera został rozgłoszony ARP z adresem MAC (00:0c:29:ec:8a:14). Do kogo należy?

Należy on do hosta o adresie 172.16.17.128.



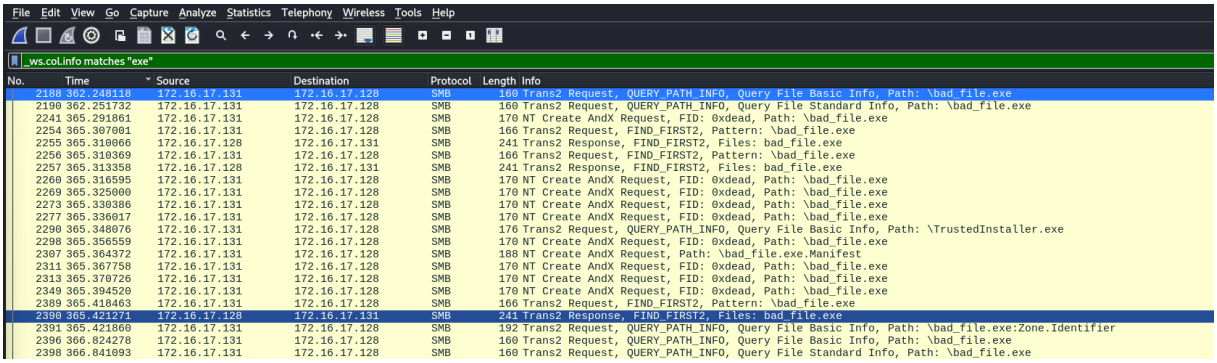
g. Analizowane logi zawierają informacje o pliku wykonywalny exe. Sprawdź, kiedy został pobrany, z którego adresu i jak nazywa się plik?

Najpierw szukałem po http response, a później podążałem strumieniem.



Nie znalazłem jednak tego, czego szukałem. Kombinowałem dalej.

Znalazłem w dokumentacji opcję filtrowania danych w danej kolumnie po nazwie.

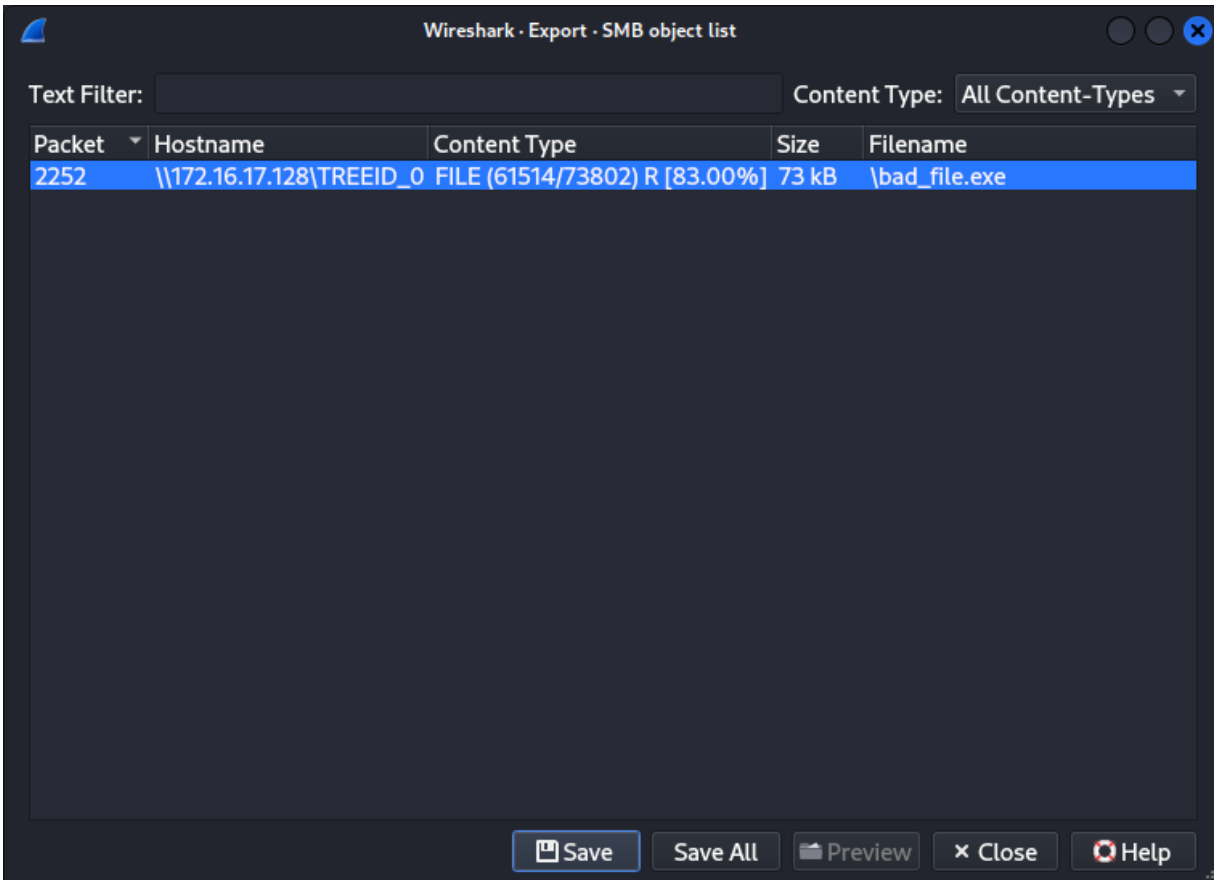


Widzimy plik bad_file.exe.

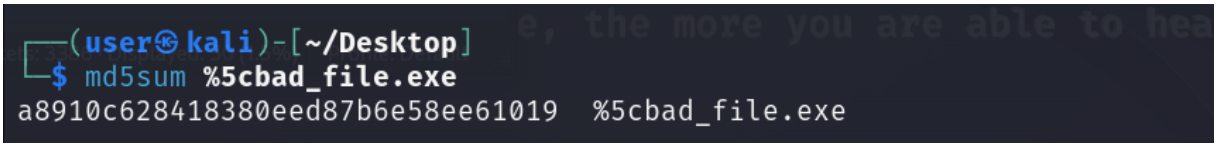
h. Przy użyciu opcji z Wireshark „Extract Object” wyciągnij odnaleziony plik,

zapisz go w nowym folderze i przy pomocy narzędzia md5sum sprawdź jego

sumę kontrolną.

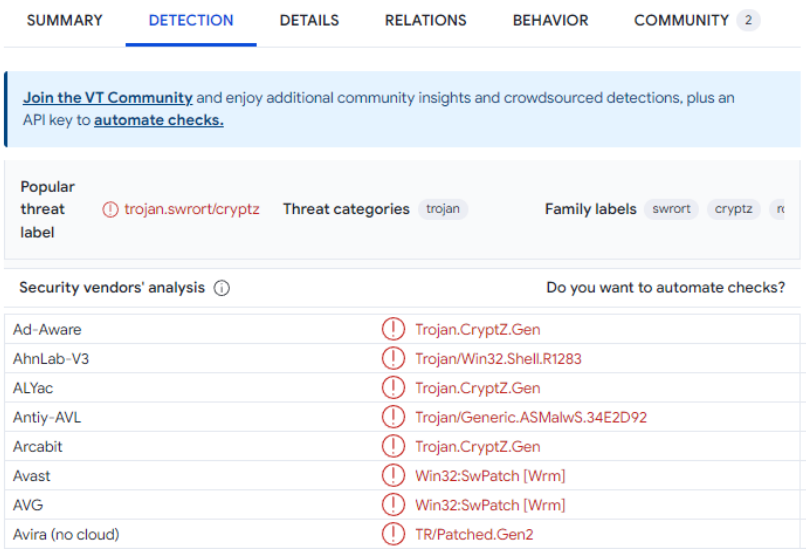


Pozyskuję sumę kontrolną.



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

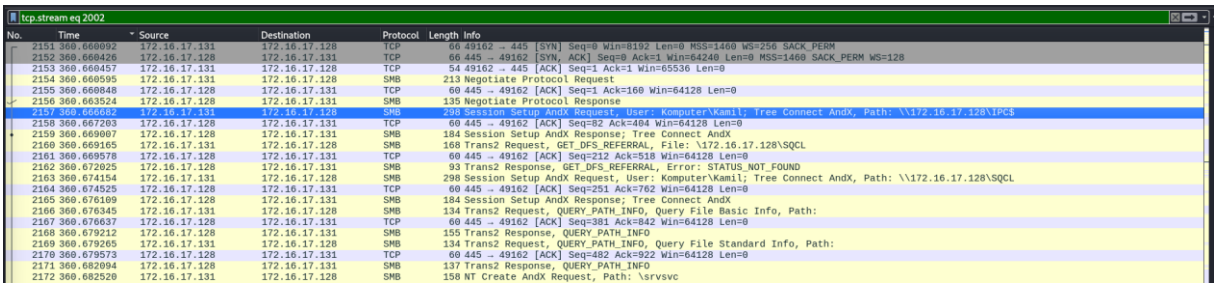
Widzimy, że jest to trojan, dużo sygnatur i zgłoszeń.



j. Który z portów był wykorzystywany do przesyłania danych pochodzących

z ataku? Podaj nazwę komputera, który został zaatakowany.

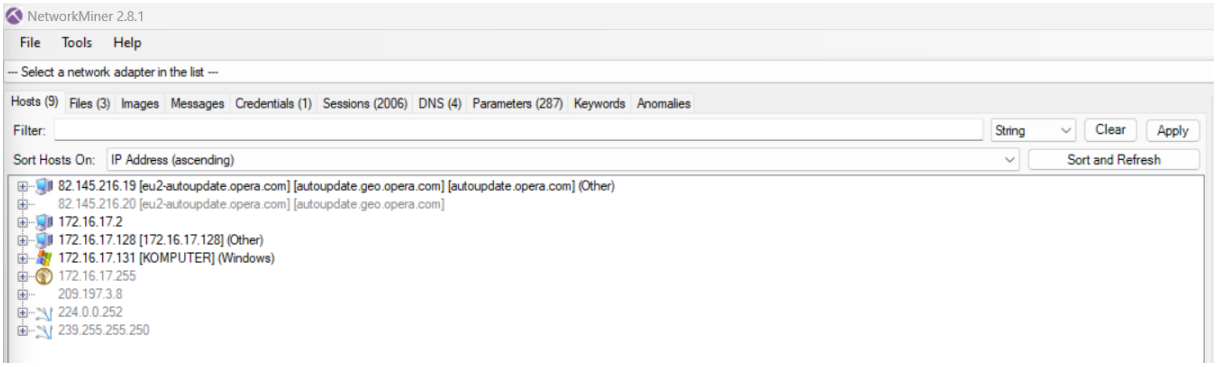
Zaatakowany został komputer 172.16.17.131 o nazwie: Komputer\Kamil.



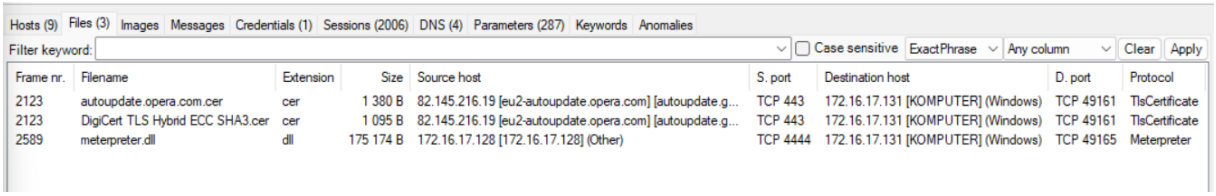
Port, który był wykorzystany do przesyłania danych to 445. Czyli ten odpowiedzialny za SMB. Stary dobry i znany atak.

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

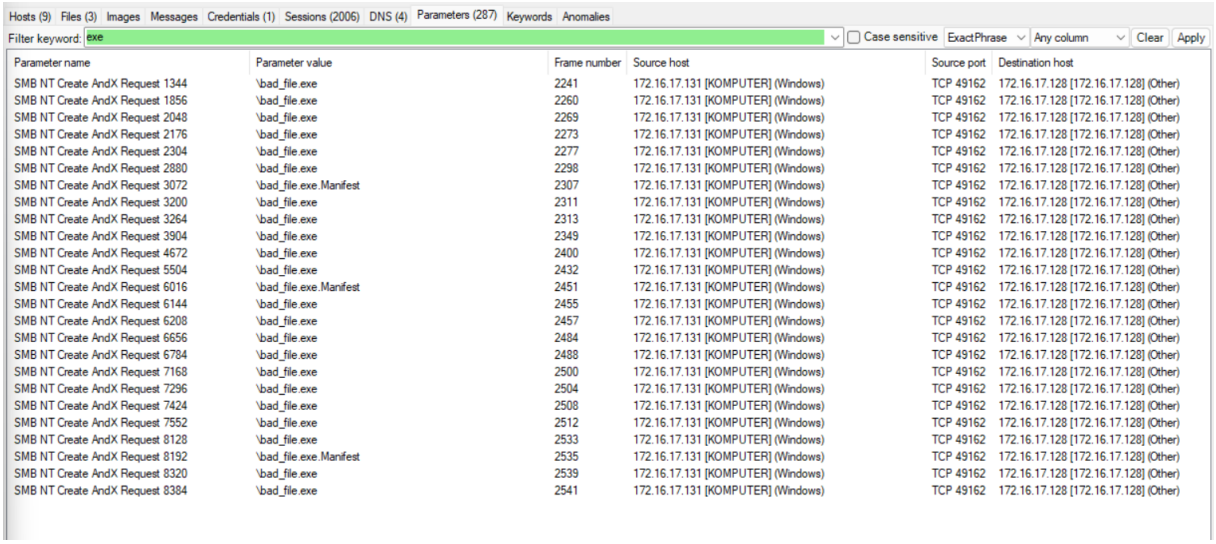
Zainstalowałem program i wgrałem plik .pcap.



Takie pliki są widoczne w files.



Niestety nie ma tutaj interesującego nas pliku. Należy zagłębić się trochę dalej.



Parameter name	Parameter value	Frame number	Source host	Source port	Destination host
SMB NT Create AndX Request 1344	\bad_file.exe	2241	172.16.17.131 [KOMPUTER] (Windows)	TCP 49162	172.16.17.128 [172.16.17.128] (Other)
SMB NT Create AndX Request 1856	\bad_file.exe	2260	172.16.17.131 [KOMPUTER] (Windows)	TCP 49162	172.16.17.128 [172.16.17.128] (Other)
SMB NT Create AndX Request 2048	\bad_file.exe	2269	172.16.17.131 [KOMPUTER] (Windows)	TCP 49162	172.16.17.128 [172.16.17.128] (Other)
SMB NT Create AndX Request 2176	\bad_file.exe	2273	172.16.17.131 [KOMPUTER] (Windows)	TCP 49162	172.16.17.128 [172.16.17.128] (Other)
SMB NT Create AndX Request 2304	\bad_file.exe	2277	172.16.17.131 [KOMPUTER] (Windows)	TCP 49162	172.16.17.128 [172.16.17.128] (Other)
SMB NT Create AndX Request 2880	\bad_file.exe	2298	172.16.17.131 [KOMPUTER] (Windows)	TCP 49162	172.16.17.128 [172.16.17.128] (Other)
SMB NT Create AndX Request 3072	\bad_file.exe.Manifest	2307	172.16.17.131 [KOMPUTER] (Windows)	TCP 49162	172.16.17.128 [172.16.17.128] (Other)
SMB NT Create AndX Request 3200	\bad_file.exe	2311	172.16.17.131 [KOMPUTER] (Windows)	TCP 49162	172.16.17.128 [172.16.17.128] (Other)
SMB NT Create AndX Request 3264	\bad_file.exe	2313	172.16.17.131 [KOMPUTER] (Windows)	TCP 49162	172.16.17.128 [172.16.17.128] (Other)
SMB NT Create AndX Request 3904	\bad_file.exe	2349	172.16.17.131 [KOMPUTER] (Windows)	TCP 49162	172.16.17.128 [172.16.17.128] (Other)
SMB NT Create AndX Request 4672	\bad_file.exe	2400	172.16.17.131 [KOMPUTER] (Windows)	TCP 49162	172.16.17.128 [172.16.17.128] (Other)
SMB NT Create AndX Request 5504	\bad_file.exe	2432	172.16.17.131 [KOMPUTER] (Windows)	TCP 49162	172.16.17.128 [172.16.17.128] (Other)
SMB NT Create AndX Request 6016	\bad_file.exe.Manifest	2451	172.16.17.131 [KOMPUTER] (Windows)	TCP 49162	172.16.17.128 [172.16.17.128] (Other)
SMB NT Create AndX Request 6144	\bad_file.exe	2455	172.16.17.131 [KOMPUTER] (Windows)	TCP 49162	172.16.17.128 [172.16.17.128] (Other)
SMB NT Create AndX Request 6208	\bad_file.exe	2457	172.16.17.131 [KOMPUTER] (Windows)	TCP 49162	172.16.17.128 [172.16.17.128] (Other)
SMB NT Create AndX Request 6656	\bad_file.exe	2484	172.16.17.131 [KOMPUTER] (Windows)	TCP 49162	172.16.17.128 [172.16.17.128] (Other)
SMB NT Create AndX Request 6784	\bad_file.exe	2488	172.16.17.131 [KOMPUTER] (Windows)	TCP 49162	172.16.17.128 [172.16.17.128] (Other)
SMB NT Create AndX Request 7168	\bad_file.exe	2500	172.16.17.131 [KOMPUTER] (Windows)	TCP 49162	172.16.17.128 [172.16.17.128] (Other)
SMB NT Create AndX Request 7296	\bad_file.exe	2504	172.16.17.131 [KOMPUTER] (Windows)	TCP 49162	172.16.17.128 [172.16.17.128] (Other)
SMB NT Create AndX Request 7424	\bad_file.exe	2508	172.16.17.131 [KOMPUTER] (Windows)	TCP 49162	172.16.17.128 [172.16.17.128] (Other)
SMB NT Create AndX Request 7552	\bad_file.exe	2512	172.16.17.131 [KOMPUTER] (Windows)	TCP 49162	172.16.17.128 [172.16.17.128] (Other)
SMB NT Create AndX Request 8128	\bad_file.exe	2533	172.16.17.131 [KOMPUTER] (Windows)	TCP 49162	172.16.17.128 [172.16.17.128] (Other)
SMB NT Create AndX Request 8192	\bad_file.exe.Manifest	2535	172.16.17.131 [KOMPUTER] (Windows)	TCP 49162	172.16.17.128 [172.16.17.128] (Other)
SMB NT Create AndX Request 8320	\bad_file.exe	2539	172.16.17.131 [KOMPUTER] (Windows)	TCP 49162	172.16.17.128 [172.16.17.128] (Other)
SMB NT Create AndX Request 8384	\bad_file.exe	2541	172.16.17.131 [KOMPUTER] (Windows)	TCP 49162	172.16.17.128 [172.16.17.128] (Other)

Po odfiltrowaniu wszystko dobrze widać. Znajdujemy sygnatury pliku bad_file.exe. Port docelowy również się zgadza – 445.