



To zdjęcie, autor: Nieznany autor, licencja: CC BY-SA-NC

Lista 1

TECHNOLOGIE SIECIOWE

Dominika Szydło | 250109 | wtorek 9:15

1. Programy i przykłady wywołań

PING

Ping jest programem służącym do badania połączeń sieciowych. Ping wykorzystuje protokół ICMP (Internet Control Message Protocol) - wysyła do podanego hosta pakiety ICMP Echo request i czeka na odpowiedź w postaci ICMP Echo Reply. Następnie wypisuje statystyki przeprowadzonej procedury - liczba wysłanych pakietów, liczba otrzymanych pakietów, minimum, maksimum, średnią i standardowe odchylenie rtt (czasu, jaki zajmuje wysłanie sygnału od nadawcy do odbiorcy i z powrotem). W pingu możemy korzystać z różnych flag, m. in.:

- -c count - wysyła count echo requestów
- -i interval - czeka interval sekund między wysyłaniem kolejnych echo requestów, defaultowo ustawiony na sekundę,
- -s packetsize - ustala liczbę bajtów do wysłania, defaultowo ustawione na 56 bajtów, co daje nam 64 bajtów ICMP po dołączeniu ośmiobajtowego nagłówka
- -t ttl - ustawia czas życia pakietów na ttl.

Flagi różnią się nazwami między systemami. Powyższych używa się na Linuxie, ja natomiast przy wykonywaniu zadań korzystałam z Windowsa (niestety VirtualBox ogranicza tracerouta).

Przykładowe wywołanie:

```
PING google.com (172.217.23.206) 56(84) bytes of data.
 64 bytes from prg03s05-in-f206.1e100.net (172.217.23.206) :
icmp_seq=1 ttl=57 time=50.8 ms
 64 bytes from prg03s05-in-f206.1e100.net (172.217.23.206) :
icmp_seq=2 ttl=57 time=49.8 ms
 64 bytes from prg03s05-in-f206.1e100.net (172.217.23.206) :
icmp_seq=3 ttl=57 time=52.8 ms
 64 bytes from prg03s05-in-f206.1e100.net (172.217.23.206) :
icmp_seq=4 ttl=57 time=50.0 ms
--- google.com ping statistics ---
 4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 49.865/50.904/52.856/1.221 ms
```

TRACEROUTE

Traceroute, jak wskazuje nazwa, bada trasę pakietów od wysyłającego na docelowy serwer za pomocą protokołów ICMP i UDP (User Datagram Protocol). Robi to w następujący sposób - wysyła po kolei pakiety o ttl = 1, 2, 3,... i dla każdego zwraca adres IP routera, dla którego ttl osiągnęło wartość 0, aż dojdzie do routera docelowego. Ttl dekrementuje się w trakcie trasy, ponieważ każdy router na trasie zmniejsza ttl o jeden, a gdy dojdzie ono do 1 router, który go otrzyma zmniejsza tę wartość do zera i odrzuca pakiet, a nadawca otrzymuje komunikat o błędzie.

Przykład użycia:

```
Tracing route to google.com [216.58.201.110]
```

1	2 ms	3 ms	9 ms	funbox.home [192.168.1.1]
2	25 ms	28 ms	27 ms	wro-bng3.neo.tpnet.pl [83.1.5.4]
3	26 ms	26 ms	28 ms	wro-r2.tpnet.pl [80.50.122.77]
4	37 ms	32 ms	31 ms	poz-r1.tpnet.pl [194.204.175.205]
5	54 ms	55 ms	52 ms	ae104-10.ffttr6.-.opentransit.net 93.251.249.15]
6	62 ms	66 ms	65 ms	72.14.214.52
7	68 ms	72 ms	68 ms	108.170.252.1
8	58 ms	62 ms	56 ms	108.170.252.19
9	74 ms	48 ms	50 ms	108.170.236.68
10	51 ms	50 ms	56 ms	108.170.245.33
11	49 ms	58 ms	57 ms	108.170.238.235
12	69 ms	52 ms	50 ms	prg03s02-in-f110.1e100.net 16.58.201.110]

WIRESHARK

- możliwość analizy różnorodnych protokołów sieciowych,
- wieloplatformowość,
- wsparcie dla dekrptowania protokołów takich jak SNMPv3, SSL/TLS, EP, WPA/WPA2,
- analiza VoIP ("telefonii internetowej").

Przechwytywanie z Wi-Fi

Plik Edycja Widok Identyfikacja Przechwytywanie Analizy Skanowanie Tabela Bezprzewodowe Narzędzia Pomoc

Zadanie filtry wywołania... <Ctrl+F>

No.	Time	Source	Destination	Protocol	Length	Info
1155	140.266992	13.107.42.12	192.168.1.17	TCP	1506	643 → 59853 [ACK] Seq=1455 Ack=177 Win=525312 Len=0 [TCP segment of a reassembled PDU]
1156	140.266993	13.107.42.12	192.168.1.17	TCP	1506	643 → 59853 [ACK] Seq=2085 Ack=177 Win=525312 Len=0 [TCP segment of a reassembled PDU]
1157	140.266994	13.107.42.12	192.168.1.17	TCP	1506	643 → 59853 [ACK] Seq=1557 Ack=177 Win=525312 Len=0 [TCP segment of a reassembled PDU]
1158	140.266994	13.107.42.12	192.168.1.17	TCP	1506	643 → 59853 [ACK] Seq=5809 Ack=177 Win=525312 Len=0 [TCP segment of a reassembled PDU]
1159	140.266994	13.107.42.12	192.168.1.17	TCP	1506	643 → 59853 [ACK] Seq=7261 Ack=177 Win=525312 Len=0 [TCP segment of a reassembled PDU]
1160	140.266995	13.107.42.12	192.168.1.17	TLSv1..	507	Server Hello, Certificate, Certificate Status, Server Key Exchange, Server Hello Done
1161	140.267006	192.168.1.17	13.107.42.12	TCP	54	59853 → 443 [ACK] Seq=177 Ack=156 Win=6048 Len=0
1162	140.270720	192.168.1.17	13.107.42.12	TLSv1..	121	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
1163	140.308113	13.107.42.12	192.168.1.17	TCP	54	443 → 59853 [ACK] Seq=9166 Ack=335 Win=520586 Len=0
1164	140.308771	13.107.42.12	192.168.1.17	TLSv1..	380	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
1165	140.308852	192.168.1.17	13.107.42.12	TCP	54	59853 → 443 [ACK] Seq=335 Ack=9452 Win=65792 Len=0
1166	140.309791	192.168.1.17	13.107.42.12	TCP	1494	59853 → 443 [ACK] Seq=335 Ack=9452 Win=65792 Len=0 [TCP segment of a reassembled PDU]
1167	140.309793	192.168.1.17	13.107.42.12	TLSv1..	123	Application Data
1168	140.309979	192.168.1.17	13.107.42.12	TLSv1..	305	Application Data
1169	140.344315	13.107.42.12	192.168.1.17	TCP	54	443 → 59853 [ACK] Seq=9492 Ack=1775 Win=525568 Len=0
1170	140.344598	13.107.42.12	192.168.1.17	TCP	54	443 → 59853 [ACK] Seq=9492 Ack=184 Win=525312 Len=0
1171	140.344599	13.107.42.12	192.168.1.17	TCP	54	443 → 59853 [ACK] Seq=9492 Ack=2095 Win=525058 Len=0
1172	140.407829	13.107.42.12	192.168.1.17	TLSv1..	922	Application Data
1173	140.407943	192.168.1.17	13.107.42.12	TCP	54	59853 → 443 [ACK] Seq=2095 Ack=10360 Win=65824 Len=0
1174	140.408373	13.107.42.12	192.168.1.17	TLSv1..	111	Application Data
1175	140.408456	192.168.1.17	13.107.42.12	TCP	54	59853 → 443 [ACK] Seq=2095 Ack=10417 Win=64768 Len=0
1176	140.408464	192.168.1.17	13.107.42.12	TCP	54	59853 → 443 [FIN, ACK] Seq=2095 Ack=10417 Win=64768 Len=0
1177	140.446273	13.107.42.12	192.168.1.17	TCP	54	443 → 59853 [ACK] Seq=10417 Ack=2096 Win=525056 Len=0
1178	140.446937	13.107.42.12	192.168.1.17	TCP	54	443 → 59853 [FIN, ACK] Seq=10417 Ack=2096 Win=525056 Len=0
1179	140.446963	192.168.1.17	13.107.42.12	TCP	54	59853 → 443 [ACK] Seq=2096 Ack=10418 Win=64768 Len=0
1180	140.446957	192.168.1.17	239.255.255.250	UDP	85	5050 → 5050 Len=43
1181	140.655818	192.168.1.17	192.168.1.255	UDP	85	5050 → 5050 Len=43
1182	141.745419	192.168.1.17	64.235.167.188	TCP	55	[TCP Keep-Alive] 59859 → 5228 [ACK] Seq=790 Ack=6425 Win=131528 Len=1
1183	141.773578	64.235.167.188	192.168.1.17	TCP	66	[TCP Keep-Alive] 5228 → 59853 [ACK] Seq=6425 Ack=793 Win=6702 Len=0 SLE=790 SRE=793

```
> Frame 370: 55 bytes on wire (440 bits), 55 bytes captured (440 bits) on interface Device\NPF{2E1A9D0C-BF8F-4B0E-97C7-DAA4A43A60D5}, Id 0
    Ethernet II, Src: IntelCor_20:48:BD (d4:25:b0:20:48:bd), Dst: Sagencom_21:93:c5 (44:e9:d0:21:93:c5)
    Internet Protocol Version 4, Src: 192.168.1.17, Dst: 74.125.133.188
    Transmission Control Protocol, Src Port: 59823, Dst Port: 5228, Seq: 1, Ack: 1, Len: 1
    Data (1 byte)
```

2. Zadania praktyczne

LICZBA WĘZŁÓW NA TRASIE

Aby znaleźć liczbę węzłów na trasie od nas na serwer należy znaleźć pierwszą wartość ttl podaną jako parametr taką, żeby echo doszło na serwer, następnie dekrementować ją dopóki pakietom nie starczy ttlu, a gdy znajdziemy taką wartość należy do niej dodać 1. Żeby znaleźć liczbę węzłów w drodze powrotnej należy od standardowego ttl nadanego przez system (64 dla mniejszych, 255 dla większych) odjąć ten wypisany przez ping.

Wywołanie dla serwera w Nowej Zelandii:

```
C:\Users\dszyd>ping -n 4 -i 15 103.21.194.19
Pinging 103.21.194.19 with 32 bytes of data:
Reply from 114.23.3.231: TTL expired in transit.
Reply from 114.23.3.231: TTL expired in transit.
Reply from 114.23.3.231: TTL expired in transit.
Reply from 114.23.3.231: TTL expired in transit.
Ping statistics for 103.21.194.19:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss)

C:\Users\dszyd>ping -n 4 -i 16 103.21.194.19
Pinging 103.21.194.19 with 32 bytes of data:
Reply from 103.21.194.19: bytes=32 time=484ms TTL=48
Reply from 103.21.194.19: bytes=32 time=359ms TTL=48
Reply from 103.21.194.19: bytes=32 time=617ms TTL=48
Reply from 103.21.194.19: bytes=32 time=533ms TTL=48
Ping statistics for 103.21.194.19:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 359ms, Maximum = 617ms, Average = 498ms
```

Liczba węzłów z powrotem: $64 - 48 = 16$

Liczba węzłów tam: dla ttl = 15 nie starczyło go, by echo doszło do serwera docelowego, natomiast dla ttl = 16 już tak, czyli węzłów od nas na wybrany serwer jest 16.

Wywołanie dla serwera w Radomiu:

```
C:\Users\dszyd>ping -n 4 -i 6 83.18.201.250
Pinging 83.18.201.250 with 32 bytes of data:
Reply from 83.18.201.249: TTL expired in transit.
Reply from 83.18.201.249: TTL expired in transit.
Reply from 83.18.201.249: TTL expired in transit.
Reply from 83.18.201.249: TTL expired in transit.
Ping statistics for 83.18.201.250:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss)
```

```
C:\Users\dszyd>ping -n 4 -i 7 83.18.201.250
Pinging 83.18.201.250 with 32 bytes of data:
Reply from 83.18.201.250: bytes=32 time=60ms TTL=58
Reply from 83.18.201.250: bytes=32 time=62ms TTL=58
Reply from 83.18.201.250: bytes=32 time=58ms TTL=58
Reply from 83.18.201.250: bytes=32 time=60ms TTL=58
Ping statistics for 83.18.201.250:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 58ms, Maximum = 62ms, Average = 60ms
```

Liczba węzłów tam: 7

Liczba węzłów z powrotem: $64 - 58 = 6$

WPŁYW ROZMIARU DANYCH NA LICZBĘ WĘZŁÓW I CZAS PROPAGACJI

Wywołanie dla tych samych serwerów, ale wysyłając 6400 bajtów danych zamiast 32.

```
C:\Users\dszyd>ping -n 4 -i 16 -l 6400 103.21.194.19
Pinging 103.21.194.19 with 6400 bytes of data:
Reply from 103.21.194.19: bytes=6400 time=751ms TTL=48
Reply from 103.21.194.19: bytes=6400 time=445ms TTL=48
Reply from 103.21.194.19: bytes=6400 time=453ms TTL=48
Reply from 103.21.194.19: bytes=6400 time=677ms TTL=48
Ping statistics for 103.21.194.19:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss)
Approximate round trip times in milli-seconds:
    Minimum = 445ms, Maximum = 751ms, Average = 581ms
```

Liczba węzłów nie uległa zmianie, natomiast czas propagacji tak – średnia wzrosła o około 83 ms.

```
C:\Users\dszyd>ping -n 4 -i 7 -l 6400 83.18.201.250
Pinging 83.18.201.250 with 6400 bytes of data:
Reply from 83.18.201.250: bytes=6400 time=467ms TTL=58
Reply from 83.18.201.250: bytes=6400 time=150ms TTL=58
Reply from 83.18.201.250: bytes=6400 time=168ms TTL=58
Reply from 83.18.201.250: bytes=6400 time=180ms TTL=58
Ping statistics for 83.18.201.250:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
```

Minimum = 150ms, Maximum = 467ms, Average = 241ms

Liczba węzłów również nie uległa zmianie, ale średni czas propagacji wzrósł o 180 ms. Można stwierdzić zatem, iż wielkość danych nie wpływa na liczbę węzłów, ale wydłuża czas propagacji.

FRAGMENTACJA PAKIETÓW

Opcja -f ustawia flagę, która nie pozwala na fragmentację. Oto fragment mojego ręcznego binary searcha w poszukiwaniu największego niepofragmentowanego pakietu jaki mogę wysłać:

```
C:\Users\dszyd>ping -n 4 -i 7 -f -l 1500 83.18.201.250
Pinging 83.18.201.250 with 1500 bytes of data:
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Ping statistics for 83.18.201.250:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss)
```

```
C:\Users\dszyd>ping -n 4 -i 7 -f -l 1450 83.18.201.250
Pinging 83.18.201.250 with 1450 bytes of data:
Reply from 83.18.201.250: bytes=1450 time=76ms TTL=58
Reply from 83.18.201.250: bytes=1450 time=86ms TTL=58
Reply from 83.18.201.250: bytes=1450 time=80ms TTL=58
Reply from 83.18.201.250: bytes=1450 time=100ms TTL=58
Ping statistics for 83.18.201.250:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 76ms, Maximum = 100ms, Average = 85ms
```

```
C:\Users\dszyd>ping -n 4 -i 7 -f -l 1475 83.18.201.250
Pinging 83.18.201.250 with 1475 bytes of data:
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Ping statistics for 83.18.201.250:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss)
```

```
C:\Users\dszyd>ping -n 4 -i 7 -f -l 1465 83.18.201.250
Pinging 83.18.201.250 with 1465 bytes of data:
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Ping statistics for 83.18.201.250:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

```
C:\Users\dszyd>ping -n 4 -i 7 -f -l 1464 83.18.201.250
Pinging 83.18.201.250 with 1464 bytes of data:
Reply from 83.18.201.250: bytes=1464 time=74ms TTL=58
Reply from 83.18.201.250: bytes=1464 time=74ms TTL=58
Reply from 83.18.201.250: bytes=1464 time=75ms TTL=58
Reply from 83.18.201.250: bytes=1464 time=139ms TTL=58
Ping statistics for 83.18.201.250:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 74ms, Maximum = 139ms, Average = 90ms
```

Są to 1464 bajty. Tak przedstawiają się wyniki wysyłania pakietu tego samego rozmiaru z wyłączoną i włączoną fragmentacją:

```
domka@domka-VirtualBox:~$ ping -c 4 -s 400 -M do 83.18.201.250
PING 83.18.201.250 (83.18.201.250) 400(428) bytes of data.
408 bytes from 83.18.201.250: icmp_seq=1 ttl=57 time=68.4 ms
408 bytes from 83.18.201.250: icmp_seq=2 ttl=57 time=60.9 ms
408 bytes from 83.18.201.250: icmp_seq=3 ttl=57 time=219 ms
408 bytes from 83.18.201.250: icmp_seq=4 ttl=57 time=65.3 ms
--- 83.18.201.250 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 60.921/103.614/219.703/67.078 ms

domka@domka-VirtualBox:~$ ping -c 4 -s 400 -M dont 83.18.201.250
PING 83.18.201.250 (83.18.201.250) 400(428) bytes of data.
408 bytes from 83.18.201.250: icmp_seq=1 ttl=57 time=63.4 ms
408 bytes from 83.18.201.250: icmp_seq=2 ttl=57 time=60.8 ms
408 bytes from 83.18.201.250: icmp_seq=3 ttl=57 time=64.8 ms
408 bytes from 83.18.201.250: icmp_seq=4 ttl=57 time=65.5 ms

--- 83.18.201.250 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 60.899/63.678/65.553/1.795 ms
```

Jak widać fragmentacja nie ma istotnego wpływu na czas propagacji.

„ŚREDNICA INTERNETU”

Najdłuższa trasa jaką udało mi się znaleźć to 22 węzły. Serwer znajduje się w Chinach.

```
C:\Users\dszyd>ping -n 4 202.46.34.76

Pinging 202.46.34.76 with 32 bytes of data:

Reply from 202.46.34.76: bytes=32 time=363ms TTL=42
Reply from 202.46.34.76: bytes=32 time=366ms TTL=42
Reply from 202.46.34.76: bytes=32 time=363ms TTL=42
Reply from 202.46.34.76: bytes=32 time=360ms TTL=42

Ping statistics for 202.46.34.76:

    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 360ms, Maximum = 366ms, Average = 363ms
```

WYKRYWANIE SIECI WIRTUALNYCH

Sieci wirtualne to sieci komputerowe wydzielone logicznie w ramach większej sieci fizycznej. W mojej internetowej przygodzie udało mi się napotkać coś takiego:

```
C:\Users\dszyd>tracert 103.130.4.4

Tracing route to 103.130.4.4 over a maximum of 30 hops
  0  0 ms  0 ms  0 ms  10.0.2.15
  1    2 ms    1 ms    1 ms  funbox.home [192.168.1.1]
  2   62 ms   36 ms   26 ms  wro-bng3.neo.tpnet.pl [83.1.5.4]
  3   25 ms   25 ms   28 ms  wro-r1.tpnet.pl [80.50.18.77]
  4   34 ms   28 ms   54 ms  poz-r1.tpnet.pl [194.204.175.205]
  5   56 ms   54 ms   56 ms  ae104-10.ffttr6.-.opentransit.net
[193.251.249.15]
  6   64 ms   59 ms   60 ms  hundredgige0-4-0-15.auvtr5.-
.opentransit.net [193.251.129.43]
  7   67 ms   68 ms   70 ms  et-13-1-1-0.marcr6.-
.opentransit.net [193.251.131.231]
  8   69 ms   70 ms   69 ms  gigabitethernet5-0-0.marcr4.-
.opentransit.net [193.251.133.210]
  9  103 ms  138 ms  128 ms  telin-2.gw.opentransit.net
[193.251.252.84]
 10  320 ms  320 ms  314 ms  180.240.196.42
 11  315 ms  316 ms  311 ms  180.240.193.202
 12  358 ms  365 ms  358 ms  36.67.254.29
 13  353 ms  354 ms  357 ms  36.92.252.50
 14  278 ms  253 ms  269 ms  203.207.52.17
 15  385 ms  388 ms  391 ms  36.92.252.50
 16  303 ms  317 ms  311 ms  203.207.52.17
 17  415 ms  426 ms  416 ms  36.92.252.50
 18  356 ms  353 ms  351 ms  203.207.52.17
```


19	463 ms	471 ms	473 ms	36.92.252.50
20	422 ms	412 ms	398 ms	203.207.52.17
21	511 ms	519 ms	530 ms	36.92.252.50
22	488 ms	447 ms	444 ms	203.207.52.17
23	581 ms	556 ms	563 ms	36.92.252.50
24	468 ms	471 ms	480 ms	203.207.52.17
25	534 ms	514 ms	531 ms	36.92.252.50
26	443 ms	460 ms	451 ms	203.207.52.17
27	560 ms	551 ms	551 ms	36.92.252.50
28	485 ms	496 ms	486 ms	203.207.52.17
29	583 ms	582 ms	592 ms	36.92.252.50
30	549 ms	555 ms	510 ms	203.207.52.17

Trace complete.

Wydaje mi się, że w 13 kroku wpada w jedną sieć wirtualną, potem w drugą, a potem krąży między nimi.

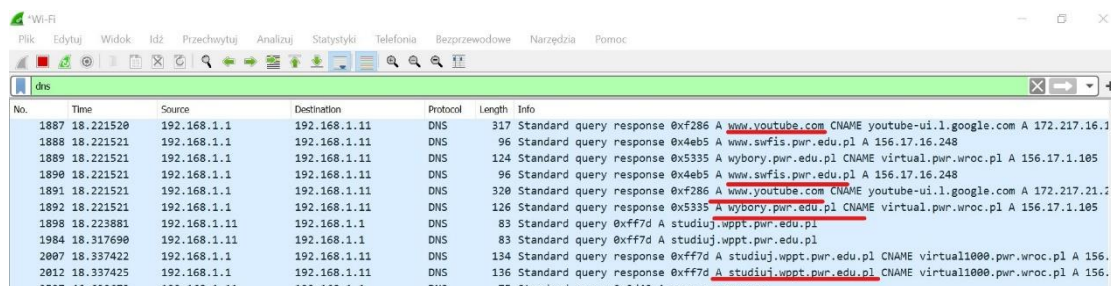
3. Wnioski

Wszystkie testowane przeze mnie programy są użytecznymi narzędziami w diagnozowaniu sieci. Różnią się od siebie informacjami jakie dostarczają, a także konkretnymi zastosowaniami. Szczególnie potężnym narzędziem wydaje mi się być Wireshark, a nawet niebezpiecznym w rękach osoby, która używa go by zdobyć informacje w niezbyt szczytnym celu, jak włamanie się komuś na konto na niezabezpieczonej stronie czy odcięcie kogoś od sieci.

4. Uzupełnienie

WIRESHARK

Wireshark służy do podsłuchiwania i przechwytywania pakietów w sieciach. Uruchomiłam go na swoim komputerze, żeby zobaczyć jakie informacje mogę zdobyć znając jedynie hasło do sieci Wi-Fi, do której mój komputer jest podłączony. Oto czego udało mi się dowiedzieć:



Screenshot 1. Okno Wiresharka po ustawieniu filtra na protokół DNS

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.56.1	192.168.56.255	BROWS...	243	Host Announcement LAPTOP-LV5JDSNQ, Workstation, Server, NT Workstation
2	4.948109	192.168.56.1	192.168.56.255	UDP	86	57621 → 57621 Len=44
3	34.995618	192.168.56.1	192.168.56.255	UDP	86	57621 → 57621 Len=44
4	39.874507	192.168.56.1	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
5	40.875470	192.168.56.1	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
6	41.876893	192.168.56.1	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
7	42.877954	192.168.56.1	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1

SMB Mailslot Protocol

Microsoft Windows Browser Protocol

Command: Host Announcement (0x01)
Update Count: 0
Update Periodicity: 12 minutes
Host Name: LAPTOP-LV5JDSNQ
Windows version:
OS Major Version: 10
OS Minor Version: 0
Server Type: 0x00001003, Workstation, Server, NT Workstation
Browser Protocol Major Version: 15
Browser Protocol Minor Version: 1
Signature: 0xaa55
Host Comment:

Screenshot 2. Okno Wiresharka

```

[M-SEARCH * HTTP/1.1\r\n]
[Severity level: Chat]
[Group: Sequence]
Request Method: M-SEARCH
Request URI: *
Request Version: HTTP/1.1
HOST: 239.255.255.250:1900\r\n
MAN: "ssdp:discover"\r\n
MX: 1\r\n
ST: urn:dial-multiscreen-org:service:dial:1\r\n
USER-AGENT: Google Chrome/81.0.4044.122 Windows\r\n
\r\n
[Full request URI: http://239.255.255.250:1900*]
[HTTP request 1/4]
[Next request in frame: 5]

```

Screenshot 3. Okno Wiresharka

Za pomocą Wiresharka jestem w stanie sprawdzić jak nazywa się mój komputer, jaki system operacyjny posiada, z jakiej przeglądarki internetowej korzystam oraz mam dostęp do przeglądanych przeze mnie stron w czasie rzeczywistym. Może nie jest to bardzo imponujące biorąc pod uwagę, że szukam informacji o własnym komputerze, które poniekąd nie są dla mnie żadną nowością, ale wystarczy sobie wyobrazić, że jestem podłączona do jakiejś sieci publicznej i szukam tych samych informacji, ale o innych urządzeniach do niej podłączonych – wtedy sytuacja zmienia się diametralnie.

„ŚREDNICA INTERNETU”

Poszukałam trasy dłuższej od 22 węzłów. Oto niektóre z wyników:

- Nowa Zelandia

```
C:\Users\dszyd>ping -n 4 43.224.122.28
```

```
Pinging 43.224.122.28 with 32 bytes of data:
```

```
Reply from 43.224.122.28: bytes=32 time=427ms TTL=46
```

```
Reply from 43.224.122.28: bytes=32 time=343ms TTL=46
```

```
Reply from 43.224.122.28: bytes=32 time=348ms TTL=46
```

```
Reply from 43.224.122.28: bytes=32 time=422ms TTL=46
```

```
Ping statistics for 43.224.122.28:
```

```
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
```

Approximate round trip times in milli-seconds:

Minimum = 343ms, Maximum = 427ms, Average = 385ms

- **Stany Zjednoczone**

```
C:\Users\dszyd>ping -n 4 66.63.160.2
```

Pinging 66.63.160.2 with 32 bytes of data:

Reply from 66.63.160.2: bytes=32 time=339ms TTL=51

Reply from 66.63.160.2: bytes=32 time=253ms TTL=51

Reply from 66.63.160.2: bytes=32 time=273ms TTL=51

Reply from 66.63.160.2: bytes=32 time=390ms TTL=51

Ping statistics for 66.63.160.2:

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

Minimum = 253ms, Maximum = 390ms, Average = 313ms

- **Tajlandia**

```
C:\Users\dszyd>ping -n 4 1.1.249.189
```

Pinging 1.1.249.189 with 32 bytes of data:

Reply from 1.1.249.189: bytes=32 time=379ms TTL=46

Reply from 1.1.249.189: bytes=32 time=294ms TTL=46

Reply from 1.1.249.189: bytes=32 time=714ms TTL=46

Reply from 1.1.249.189: bytes=32 time=427ms TTL=46

Ping statistics for 1.1.249.189:

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

Minimum = 294ms, Maximum = 714ms, Average = 453ms

- **Singapur**

```
C:\Users\dszyd>ping -n 4 203.121.145.77
```

Pinging 203.121.145.77 with 32 bytes of data:

Reply from 203.121.145.77: bytes=32 time=345ms TTL=114

Reply from 203.121.145.77: bytes=32 time=260ms TTL=114

Reply from 203.121.145.77: bytes=32 time=234ms TTL=114

Reply from 203.121.145.77: bytes=32 time=410ms TTL=114

Ping statistics for 203.121.145.77:

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

Minimum = 234ms, Maximum = 410ms, Average = 312ms

- Japonia

```
C:\Users\dszyd>ping -n 4 101.110.63.150
Pinging 101.110.63.150 with 32 bytes of data:
Reply from 101.110.63.150: bytes=32 time=305ms TTL=44
Reply from 101.110.63.150: bytes=32 time=496ms TTL=44
Reply from 101.110.63.150: bytes=32 time=415ms TTL=44
Reply from 101.110.63.150: bytes=32 time=326ms TTL=44
Ping statistics for 101.110.63.150:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 305ms, Maximum = 496ms, Average = 385ms
```

- Kolejny serwer w Chinach o jeszcze większej liczbie węzłów na trasie, jednak jest ich na tyle dużo, że bez dokładnego śledzenia można stwierdzić, że pakiety wpadają w sieć wirtualną.

```
C:\Users\dszyd>ping -n 4 114.114.114.110
Pinging 114.114.114.110 with 32 bytes of data:
Reply from 114.114.114.110: bytes=32 time=180ms TTL=56
Reply from 114.114.114.110: bytes=32 time=195ms TTL=67
Reply from 114.114.114.110: bytes=32 time=211ms TTL=79
Reply from 114.114.114.110: bytes=32 time=224ms TTL=86
Ping statistics for 114.114.114.110:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 180ms, Maximum = 224ms, Average = 202ms
```