

Laboratorium

Technologii Sieciowych

Temat:

Wprowadzenie do wybranych aspektów programów:
„Ping” i „Wireshark”

Autor: Bartosz Banasik
Informatyka, semestr: 4
Prowadzący: dr Przemysław Kubiak

1. Wstęp

1.1 Cel pracy.

Celem pracy jest przybliżenie możliwości jakie dają nam programy: „Ping” i „Wireshark” oraz wykorzystanie ich do następujących zadań:

1. Określenie długości ścieżki od komputera do wybranego przez nas hosta.
2. Na podstawie odpowiedzi programu „Ping” ustalić długość ścieżki od hosta do naszego komputera.
3. Punkty 1) i 2) wykonać przy ustawionym bicie niefragmentacji i przy różnych rozmiarach pakietów.
4. Określić jak wyglądają opóźnienia w punktach 1) 2) i 3).
5. Używając programu: „Wireshark” określić algorytm dla klucza asymetrycznego którym posługuje się serwer TLS.

2. Opis programu: „Ping”

2.1 Ciekawostka

Nazwa programu pochodzi od onomatopei dźwięku wydawanego przez sonar, ponieważ zasada działania jest analogiczna do działania sonaru. „Ping” został napisany przez Mike’a Muuss’a w Grudniu 1938 r.

2.2 Opis

„Ping” jest komputerowym narzędziem administracyjnym używanym do weryfikacji dostępności hosta w sieci komputerowej. Mierzy on czas w jakim pakiet pokonał trasę od nas do hosta i z powrotem. Można zauważyć analogię między działaniem sonaru, który wysyła sygnał i na podstawie echa może „coś” powiedzieć na temat napotkanych obiektów.

3. Rozwiązanie zadań 1-4

3.1 Zadanie 1

Badanie długości ścieżki od nas do wybranego hosta przy użyciu programu „Ping”.

W tym zagadnieniu potrzebna będzie nam wiedza na temat Time-to-live (ttl). Jest to tak zwany czas życia pakietu. Wysyłając pakiet możemy mu ograniczyć ttl, a na każdym kolejnym routerze wartość jego będzie zmniejszana do czasu aż osiągnie cel lub jego wartość wyniesie 0. Posłuży nam do tego odpowiednia funkcja programu ustawiająca nam ttl. Odpowiednio zwiększając lub zmniejszając jego wartość będziemy w stanie określić ile hopów musimy wykonać aby dotrzeć do hosta.

Input: ping -t 3 www.google.pl (linux)

Input: ping -i 3 www.google.pl (windows)

Output:

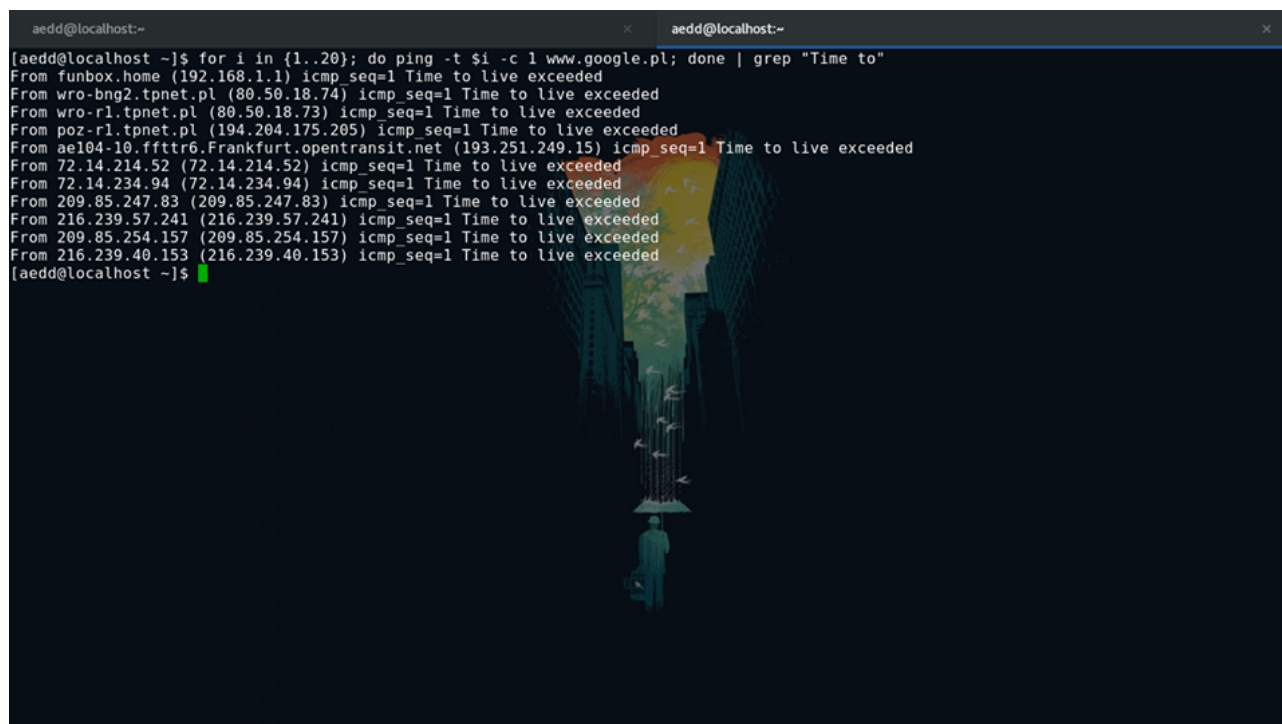
```
C:\Users\Bartosz>ping -i 3 www.google.pl

Pinging www.google.pl [46.134.210.20] with 32 bytes of data:
Reply from 80.50.18.73: TTL expired in transit.
Reply from 80.50.18.73: TTL expired in transit.
Reply from 80.50.18.73: TTL expired in transit.

Ping statistics for 46.134.210.20:
    Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),
```

Input: for i in {1..20}; do ping -t \$i -c 1 www.google.pl; done | grep „Time to”

Output:

A terminal window titled 'aedd@localhost:~' showing the execution of a command: 'for i in {1..20}; do ping -t \$i -c 1 www.google.pl; done | grep "Time to"'. The output consists of 20 lines, each starting with 'From' followed by an IP address and the message 'icmp_seq=1 Time to live exceeded'. The IP addresses are: 192.168.1.1, 80.50.18.74, 80.50.18.73, 194.204.175.205, 193.251.249.15, 72.14.214.52, 72.14.234.94, 209.85.247.83, 216.239.57.241, 209.85.254.157, 216.239.40.153, and then repeats the last three. The terminal background features a dark theme with a faint illustration of a person walking through a city street at night.

```
aedd@localhost:~  
[aedd@localhost ~]$ for i in {1..20}; do ping -t $i -c 1 www.google.pl; done | grep "Time to"  
From funbox.home (192.168.1.1) icmp_seq=1 Time to live exceeded  
From wro-bng2.tpnet.pl (80.50.18.74) icmp_seq=1 Time to live exceeded  
From wro-r1.tpnet.pl (80.50.18.73) icmp_seq=1 Time to live exceeded  
From poz-r1.tpnet.pl (194.204.175.205) icmp_seq=1 Time to live exceeded  
From ae104-10.ffftr6.Frankfurt.opentransit.net (193.251.249.15) icmp_seq=1 Time to live exceeded  
From 72.14.214.52 (72.14.214.52) icmp_seq=1 Time to live exceeded  
From 72.14.234.94 (72.14.234.94) icmp_seq=1 Time to live exceeded  
From 209.85.247.83 (209.85.247.83) icmp_seq=1 Time to live exceeded  
From 216.239.57.241 (216.239.57.241) icmp_seq=1 Time to live exceeded  
From 209.85.254.157 (209.85.254.157) icmp_seq=1 Time to live exceeded  
From 216.239.40.153 (216.239.40.153) icmp_seq=1 Time to live exceeded  
[aedd@localhost ~]$
```

3.2 Analiza wyników

Jak widzimy dla pierwszych danych wejściowych wartość ttl była za niska aby osiągnąć host docelowy. W takim przypadku ping zgłasza nam „TTL expired in transit” lub „Time to live exceeded” w zależności od platformy na jakiej pracujemy. Dla drugich danych wejściowych iteracyjnie sprawdzamy czy dla danej wartości ttl dostaniemy odpowiedź od serwera. Jak widzimy jedenaście razy nie dostaliśmy się do docelowego hosta. Właśnie taka odległość dzieli nas od danego hosta.

3.3 Zadanie 2

Badanie długości ścieżki od hosta do nas. Tutaj również będziemy wykorzystywać wiedzę na temat TTL. Warto zauważyć, że odpowiedź zostanie wysłana z pewną wartością TTL. Jaka? To zależy od systemu na jakim pracuje serwer. Odpowiednio dla Windows 98, Linux wartość ta wynosi 64, a dla nowszych systemów Windows – 128. Warto zauważyć, że jest to potęga dwójki.

Input: ping www.google.pl

Output:

```
C:\Users\Bartosz>ping www.google.pl

Pinging www.google.pl [172.217.20.195] with 32 bytes of data:
Reply from 172.217.20.195: bytes=32 time=68ms TTL=54
Reply from 172.217.20.195: bytes=32 time=68ms TTL=54
Reply from 172.217.20.195: bytes=32 time=68ms TTL=54
Reply from 172.217.20.195: bytes=32 time=68ms TTL=54

Ping statistics for 172.217.20.195:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 68ms, Maximum = 68ms, Average = 68ms

C:\Users\Bartosz>_
```

3.4 Analiza wyników

Widzimy że w odpowiedzi na zadany input ping zwraca nam TTL=54. Możemy założyć że domyślny TTL na serwerze był 64. Różnica tych wartości to będzie odległość od hosta do nas. W tym wypadku $64-54=10$.


3.5 Zadanie 3

Ustawianie bitu niefragmentacji uzyskamy dzięki przełącznikowi „-f” dla Windows i „-M do” dla Linux. Rozmiar pakietu możemy kontrolować dzięki przełącznikom -l (Windows) i -s (Linux).

Input: for i in {1..20}; do ping -t \$i -M do -s 1024 -c 1 www.google.pl; done | grep „Time”

Output:

```
aedd@localhost:~$ for i in {1..20}; do ping -t $i -M do -s 1024 -c 1 www.google.pl; done | grep "Time"
From funbox.home (192.168.1.1) icmp_seq=1 Time to live exceeded
From wro-bng2.tpnet.pl (80.50.18.74) icmp_seq=1 Time to live exceeded
From wro-r1.tpnet.pl (80.50.18.73) icmp_seq=1 Time to live exceeded
From 188.47.255.165 (188.47.255.165) icmp_seq=1 Time to live exceeded
aedd@localhost ~]$
```



Input: ping -f -l 1024 www.google.pl

Output:

```
C:\Users\Bartosz>ping -f -l 1024 www.google.pl

Pinging www.google.pl [46.134.210.20] with 1024 bytes of data:
Reply from 46.134.210.20: bytes=1024 time=32ms TTL=60
Reply from 46.134.210.20: bytes=1024 time=31ms TTL=60
Reply from 46.134.210.20: bytes=1024 time=31ms TTL=60
Reply from 46.134.210.20: bytes=1024 time=32ms TTL=60

Ping statistics for 46.134.210.20:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 31ms, Maximum = 32ms, Average = 31ms

C:\Users\Bartosz>
```

Input: ping -f -l 1465 www.google.pl

Output:

```
C:\Users\Bartosz>ping -f -l 1465 www.google.pl

Pinging www.google.pl [46.134.210.20] 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 46.134.210.20:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

3.6 Analiza wyników i porównanie

Dla standardowego rozmiaru (próba pierwsza) odległość do hosta wynosiła jednaście. W odpowiedzi wartość TTL była równa 54. Rozmiar pakietu był równy 32 bajty. Gdy ustawiliśmy bit niefragmentacji oraz zwiększyliśmy rozmiar pakietu ilość hopów do wykonania była równa 5 (zmniejszyła się). W odpowiedzi otrzymaliśmy TTL równy 60 (również się zmniejszył).

Gdy ustawiliśmy bit niefragmentacji oraz daliśmy wystarczająco duży pakiet, widzimy że nie dostaniemy odpowiedzi od hosta. Rozmiar pakietu jest zbyt duży. Jest to związane z MTU (Maximum transmission Unit). Jeśli bit niefragmentacji nie jest ustawiony wtedy dostaniemy odpowiedź od serwera, ponieważ pakiet zostanie poszatutowany.

3.7 Zadanie 4

```
[aedd@localhost ~]$ clear
[aedd@localhost ~]$ ping -M do -s 1464 -c 10 -q www.google.pl
PING www.google.pl (46.134.210.44) 1464(1492) bytes of data.

--- www.google.pl ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9015ms
rtt min/avg/max/mdev = 39.637/46.996/58.864/5.094 ms
[aedd@localhost ~]$ ping -c 10 -q www.google.pl
PING www.google.pl (46.134.210.20) 56(84) bytes of data.

--- www.google.pl ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9014ms
rtt min/avg/max/mdev = 31.041/32.855/35.218/1.204 ms
[aedd@localhost ~]$ ping -M do -s 1464 -c 50 -q www.google.pl
PING www.google.pl (46.134.210.40) 1464(1492) bytes of data.

--- www.google.pl ping statistics ---
50 packets transmitted, 50 received, 0% packet loss, time 49077ms
rtt min/avg/max/mdev = 34.583/56.746/353.840/45.156 ms
[aedd@localhost ~]$ ping -c 50 -q www.google.pl
PING www.google.pl (46.134.210.59) 56(84) bytes of data.

--- www.google.pl ping statistics ---
50 packets transmitted, 50 received, 0% packet loss, time 49057ms
rtt min/avg/max/mdev = 30.411/31.844/36.188/1.035 ms
[aedd@localhost ~]$
```

Opóźnienie odpowiednio w punktach:

1. 31.844 ms
2. 56.746 ms

Widzimy że przy ustawionym bicie niefragmentacji oraz dużym pakiecie opóźnienie jest odpowiednio większe.

```
C:\Users\Bartosz>ping -f -l 1464 www.google.pl

Pinging www.google.pl [46.134.210.20] with 1464 bytes of data:
Reply from 46.134.210.20: bytes=1464 time=32ms TTL=60
Reply from 46.134.210.20: bytes=1464 time=32ms TTL=60
Reply from 46.134.210.20: bytes=1464 time=32ms TTL=60
Reply from 46.134.210.20: bytes=1464 time=175ms TTL=60

Ping statistics for 46.134.210.20:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
C:\Users\Bartosz>ping www.google.pl

Pinging www.google.pl [46.134.210.20] with 32 bytes of data:
Reply from 46.134.210.20: bytes=32 time=29ms TTL=60
Reply from 46.134.210.20: bytes=32 time=29ms TTL=60
Reply from 46.134.210.20: bytes=32 time=29ms TTL=60
Reply from 46.134.210.20: bytes=32 time=29ms TTL=60

Ping statistics for 46.134.210.20:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 29ms, Maximum = 29ms, Average = 29ms
```


TTL zwracany przez hosta przy dużym pakiecie i ustawionym bicie niefragmentacji jest taki sam jak TTL dla domyślnych wartości.

3.8 Pingowanie serwera alibaba

```
[aedd@localhost ~]$ clear
[aedd@localhost ~]$ for i in {1..30}; do ping -t $i -c 1 www.alibaba.com ;done | grep "Time"
From funbox.home (192.168.1.1) icmp_seq=1 Time to live exceeded
From wro-bng2.tpnet.pl (80.50.18.74) icmp_seq=1 Time to live exceeded
From wro-r1.tpnet.pl (80.50.18.73) icmp_seq=1 Time to live exceeded
From poz-r1.tpnet.pl (194.204.175.209) icmp_seq=1 Time to live exceeded
From bundle-ether16-10.ffftr4.FrankfurtAmMain.opentransit.net (193.251.249.5) icmp_seq=1 Time to live exceeded
From tatateleglobe.GW.opentransit.net (193.251.249.122) icmp_seq=1 Time to live exceeded
From if-ae-7-2.tcore1.FNM-Frankfurt.as6453.net (195.219.50.2) icmp_seq=1 Time to live exceeded
From if-ae-6-2.tcore1.AV2-Amsterdam.as6453.net (195.219.194.237) icmp_seq=1 Time to live exceeded
From if-ae-2-2.tcore2.AV2-Amsterdam.as6453.net (195.219.194.6) icmp_seq=1 Time to live exceeded
From if-ae-8-2.tcore2.L78-London.as6453.net (80.231.131.5) icmp_seq=1 Time to live exceeded
From if-ae-11-2.tcore1.NT0-New-York.as6453.net (63.243.128.37) icmp_seq=1 Time to live exceeded
From if-ae-12-2.tcore1.SQN-San-Jose.as6453.net (63.243.128.29) icmp_seq=1 Time to live exceeded
From 63.243.205.166 (63.243.205.166) icmp_seq=1 Time to live exceeded
From 116.251.64.253 (116.251.64.253) icmp_seq=1 Time to live exceeded
From 198.11.128.65 (198.11.128.65) icmp_seq=1 Time to live exceeded
[aedd@localhost ~]$
```

```
C:\Users\Bartosz>ping www.alibaba.com

Pinging www.gds.alibaba.com [198.11.132.23] with 32 bytes of data:
Reply from 198.11.132.23: bytes=32 time=206ms TTL=96
Reply from 198.11.132.23: bytes=32 time=206ms TTL=96
Reply from 198.11.132.23: bytes=32 time=206ms TTL=96
Reply from 198.11.132.23: bytes=32 time=206ms TTL=96

Ping statistics for 198.11.132.23:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 206ms, Maximum = 206ms, Average = 206ms
```

3.9 Wnioski

Odległość do hosta w tym przypadku jest znacznie większa. Opóźnienie również wzrosło aż do 206 ms. Mamy również wysoki TTL = 96. Możemy zakładać że bazowy ttl wnosił 128, a więc trasa powrotu przechodziła przez 32 routery.

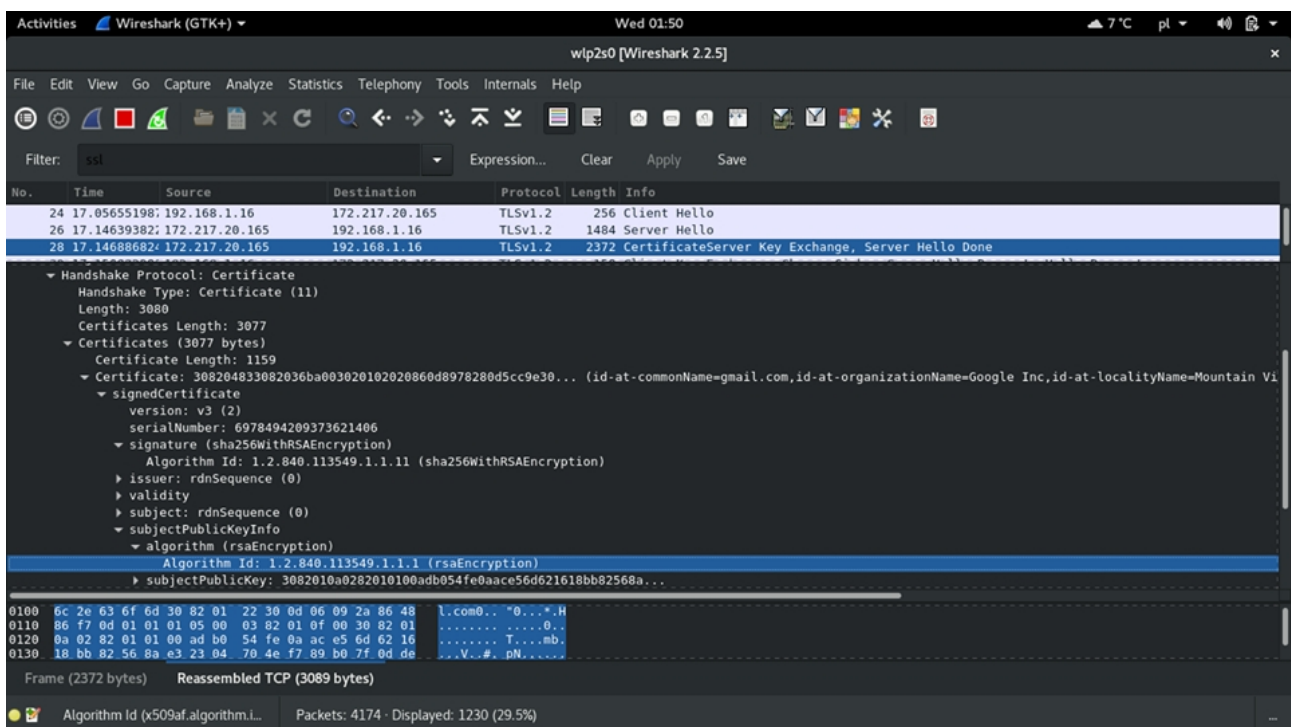
4. Zadanie 5

4.1 Opis Wireshark

Narzędzie umożliwiające przechwytywanie i nagrywanie pakietów danych oraz ich dekodowanie.

4.2 Filtrowanie

Wireshark pozwala na filtrowanie otrzymywanych pakietów. Użyjemy tej opcji aby wyszukać interesujące nasz pakiety. Filtr ssl.



Szukamy pakietu ServerHello i zaraz pod nim mamy certyfikat. Następnie Handshake Protocol. Rozwijamy wszystkie certyfikaty, i sprawdzamy PublicKeyInfo. Jak widać na zamieszczonym zdjęciu algorytm tutaj to RSA.