Michał Budnik – sprawozdanie

# 1. Raport

## 1.1 Ping

### 1.1.1 Opis programu

Głównym zadaniem polecenia ping jest sprawdzenie czasu (w millisekundach) jaki upływa pomiędzy wysłaniem pakietu na maszynie wysyłającej, a odebraniem go na serwerze docelowym. Bez używania dodatkowych parametróew polecenie wyświetla dodatkowo parametry takie jak TTL czy też wielkość pakietu. Działanie programu opiera się na wysłaniu ECHO\_REQUEST do maszyny docelowej, która, jeżeli jest tak skonfigurowana, odpowie nam na nasze zapytanie. Program posiada również sporo dodatkowych opcji, dzięki którym jego funkcjonalność jest rozszerzona. Z kilku najważniejszych opcji:

-c (count) pozwala ustawić ile razy pakiet ma zostać wysłany (domyślnie program ponawia wysyłanie pakietu dopóki nie zostanie to przerwane),

-s (size) definiuje wysyłaną wielkość pakietu (w bajtach),

-t (time to live) definiujemy startową wielkość zmiennej TTL. Większość serwerów przez które nasz pakiet się przemieści dekrementuje tę wartość o 1,

-M ustawia, czy pakiety powinny być pofragmentowane (do|dont|want).

### 1.1.2 Opis funkcjonalności

Do zliczania ilości serwerów pośredniczących w przesłaniu naszego pakietu należy użyć właściwości wartości TTL. Większość serwerów które dalej przesyłają dany pakiet dekrementują wartość TTL o 1. Ustawiając więc wartość ‘-t 1’ w poleceniu ping, i stopniowym zwiększaniu wielkości TTL dowiemy się dla jakiej minimalnej wartości nasz pakiet będzie w stanie dotrzeć do serwera docelowego, a co za tym idzie, ile serwerów musi pośredniczyć w doręczeniu naszego pakietu. Jest to jednak nie w pełni precyzyjna metoda, gdyż niektóre serwery są w stanie wyłączyć dekrementację wartości TTL, przez co nie są ‘wykrywalne’.

Inną opcją jest również regulowanie rozmiaru pakietu. Udało mi się zauważyć, że całkowita największa wielkość niepofragmentowanego pakietu, który był akceptowany przez dany serwer to 1,5kiB. Niektóre serwery akceptują pofragmentowane pakiety o wielkości do 65kiB, jednakże nie udało mi się znaleźć wielu takich serwerów. W tabelce poniżej przedstawiam wyniki zastosowania różnych kombinacji opcji dla serwera geograficznie blisko, jak również geograficznie daleko od mojej maszyny.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Serwer geograficznie blisko: | | | | | Serwer geograficznie daleko: 202.46.32.22  (Chiny, Shenzhen) | | | | | |
| 64kiB danych | | 1460kiB danych | | | 64kiB danych | | | 1460kiB danych | | |
| Frag. | Niefrag. | | Frag. | Niefrag. | | Frag. | Niefrag. | | Frag. | Niefrag. |
| 28ms | 25ms | | 25ms | 25ms | | 478ms | 507ms | | 530ms | 601ms |

Text 1: Przykładowe użycie programu PING

PING jsos.pwr.edu.pl (156.17.28.249) 1432(1460) bytes of data.

1440 bytes from 156.17.28.249 (156.17.28.249): icmp\_seq=1 ttl=244 time=33.4 ms

1440 bytes from 156.17.28.249 (156.17.28.249): icmp\_seq=2 ttl=244 time=27.2 ms

1440 bytes from 156.17.28.249 (156.17.28.249): icmp\_seq=3 ttl=244 time=38.6 ms

1440 bytes from 156.17.28.249 (156.17.28.249): icmp\_seq=4 ttl=244 time=22.2 ms

1440 bytes from 156.17.28.249 (156.17.28.249): icmp\_seq=5 ttl=244 time=22.5 ms

1440 bytes from 156.17.28.249 (156.17.28.249): icmp\_seq=6 ttl=244 time=23.7 ms

1440 bytes from 156.17.28.249 (156.17.28.249): icmp\_seq=7 ttl=244 time=25.9 ms

1440 bytes from 156.17.28.249 (156.17.28.249): icmp\_seq=8 ttl=244 time=26.7 ms

1440 bytes from 156.17.28.249 (156.17.28.249): icmp\_seq=9 ttl=244 time=24.7 ms

1440 bytes from 156.17.28.249 (156.17.28.249): icmp\_seq=10 ttl=244 time=27.1 ms

--- jsos.pwr.edu.pl ping statistics ---

10 packets transmitted, 10 received, 0% packet loss, time 9014ms

rtt min/avg/max/mdev = 22.263/27.242/38.672/4.869 ms

PING jsos.pwr.edu.pl (156.17.28.249) 1432(1460) bytes of data.

1440 bytes from 156.17.28.249 (156.17.28.249): icmp\_seq=1 ttl=244 time=33.4 ms

1440 bytes from 156.17.28.249 (156.17.28.249): icmp\_seq=2 ttl=244 time=27.2 ms

1440 bytes from 156.17.28.249 (156.17.28.249): icmp\_seq=3 ttl=244 time=38.6 ms

1440 bytes from 156.17.28.249 (156.17.28.249): icmp\_seq=4 ttl=244 time=22.2 ms

1440 bytes from 156.17.28.249 (156.17.28.249): icmp\_seq=5 ttl=244 time=22.5 ms

1440 bytes from 156.17.28.249 (156.17.28.249): icmp\_seq=6 ttl=244 time=23.7 ms

1440 bytes from 156.17.28.249 (156.17.28.249): icmp\_seq=7 ttl=244 time=25.9 ms

1440 bytes from 156.17.28.249 (156.17.28.249): icmp\_seq=8 ttl=244 time=26.7 ms

1440 bytes from 156.17.28.249 (156.17.28.249): icmp\_seq=9 ttl=244 time=24.7 ms

1440 bytes from 156.17.28.249 (156.17.28.249): icmp\_seq=10 ttl=244 time=27.1 ms

--- jsos.pwr.edu.pl ping statistics ---

10 packets transmitted, 10 received, 0% packet loss, time 9014ms

rtt min/avg/max/mdev = 22.263/27.242/38.672/4.869 ms

## 1.2 Traceroute

### 1.2.1 Opis programu

Program traceroute to narzędzie diagnostyczne które pozwala na sprawdzenie ścieżki oraz pokazania czasu jaki minął od wysłania danego pakietu do jego odebrania na danym serwerze. Służy on do badania tras pakietów.

### 1.2.2 Opis funkcjonalności

Program działa podobnie do ręcznego wyszukiwania ilości węzłów na trasie naszego pakietu dla komendy ping. Początkowo wysyłany jest pakiet z wartością TTL=1, i odbierana wiadomość zwrotna (najprawdopodobniej komunikat ICMP typu „Time exceeded”). Następnie program inkrementuje wartość TTL i powtarza sprawdzanie do którego serwera dotarł pakiet. Jeżeli pakiet dotrze do docelowego hosta to program kończy działanie.

Text 2: Przykładowe użycie programu wireshark

traceroute to 202.46.32.22 (202.46.32.22), 30 hops max, 60 byte packets

1 gateway (192.168.0.1) 7.654 ms 5.678 ms 6.618 ms

2 \* \* \*

3 pl-ktw01a-rc1-ae18-0.aorta.net (84.116.253.129) 41.414 ms 42.512 ms 53.940 ms

4 de-fra04a-rc1-ae30-0.aorta.net (84.116.137.41) 55.329 ms 55.325 ms 59.844 ms

5 de-fra03b-ri1-ae5-0.aorta.net (84.116.133.118) 58.174 ms 59.820 ms 59.824 ms

6 213.46.177.122 (213.46.177.122) 59.820 ms 33.038 ms 37.781 ms

7 TenGE0-5-0-0.br02.hkg15.pccwbtn.net (63.223.15.174) 367.993 ms \* TenGE0-0-0-20.br02.hkg15.pccwbtn.net (63.223.15.94) 365.472 ms

8 TenGE0-4-0-8.br02.hkg15.pccwbtn.net (63.223.15.146) 339.943 ms TenGE0-4-0-16.br02.hkg15.pccwbtn.net (63.223.15.150) 361.549 ms 63-216-143-26.static.pccwglobal.net (63.216.143.26) 364.339 ms

9 63-216-143-26.static.pccwglobal.net (63.216.143.26) 379.986 ms 361.737 ms 361.723 ms

10 \* ptr.cnsat.com.cn (202.46.32.170) 372.876 ms \*

11 \* \* \*

traceroute to 202.46.32.22 (202.46.32.22), 30 hops max, 60 byte packets

1 gateway (192.168.0.1) 7.654 ms 5.678 ms 6.618 ms

2 \* \* \*

3 pl-ktw01a-rc1-ae18-0.aorta.net (84.116.253.129) 41.414 ms 42.512 ms 53.940 ms

4 de-fra04a-rc1-ae30-0.aorta.net (84.116.137.41) 55.329 ms 55.325 ms 59.844 ms

5 de-fra03b-ri1-ae5-0.aorta.net (84.116.133.118) 58.174 ms 59.820 ms 59.824 ms

6 213.46.177.122 (213.46.177.122) 59.820 ms 33.038 ms 37.781 ms

7 TenGE0-5-0-0.br02.hkg15.pccwbtn.net (63.223.15.174) 367.993 ms \* TenGE0-0-0-20.br02.hkg15.pccwbtn.net (63.223.15.94) 365.472 ms

8 TenGE0-4-0-8.br02.hkg15.pccwbtn.net (63.223.15.146) 339.943 ms TenGE0-4-0-16.br02.hkg15.pccwbtn.net (63.223.15.150) 361.549 ms 63-216-143-26.static.pccwglobal.net (63.216.143.26) 364.339 ms

9 63-216-143-26.static.pccwglobal.net (63.216.143.26) 379.986 ms 361.737 ms 361.723 ms

10 \* ptr.cnsat.com.cn (202.46.32.170) 372.876 ms \*

11 \* \* \*

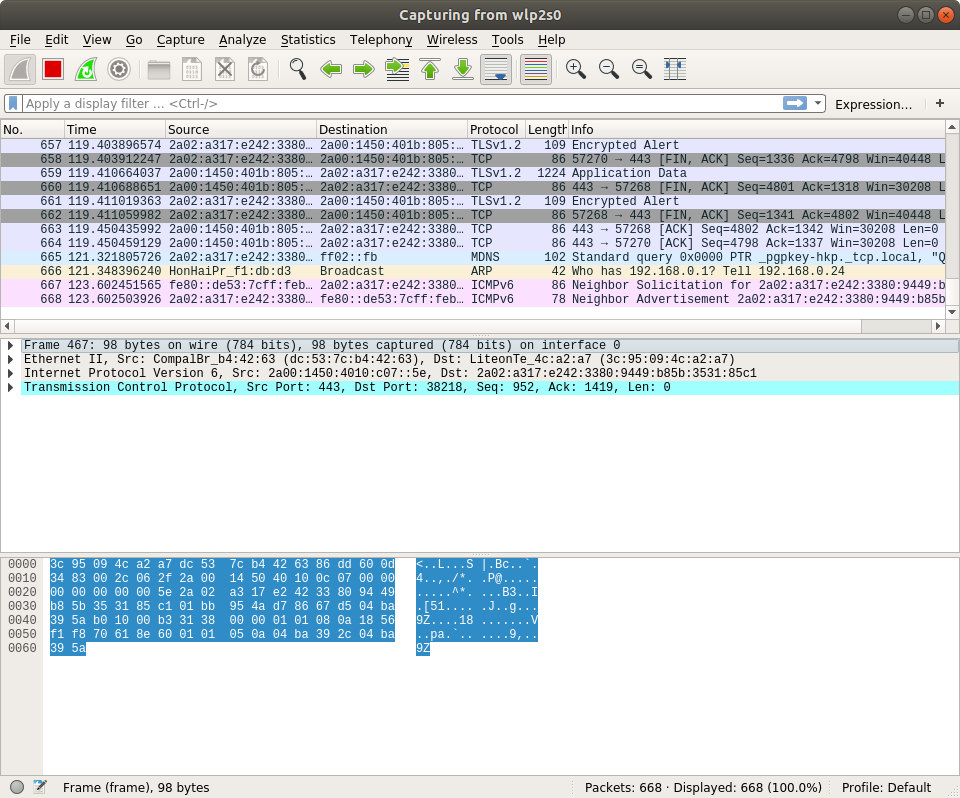
## 1.3 Wireshark

### 1.3.1 Opis programu

Program wireshark jest wolnym oprogramowaniem należącym do rodzaju ‘snifferów’ – programów służących do przechwytywania danych przepływających w danej sieci. Wireshark jest używany głównie przez administratorów sieci, specjalne służby, czy też hakerów do śledzenia pakietów.

### 1.3.2 Opis funkcjonalności

Program wireshark jest niezwykle obszernym narzędziem do nie tylko przechwytywania ruchu sieciowego, lecz również jego analizowania, filtrowania, dekodowania i wiele więcej. Pozwala on wykrywać wszelkiego rodzaju luki bezpieczeństwa danej infrastruktury sieci, lecz również na ataki hackerskie – szczególnie dosyć niewykrywalne ataki typu man in the middle.

  
Illustration 1: Przykładowe wykorzystanie programu wireshark

# 2. Wnioski:

Wyniki sprawdzania liczby skoków metodą ping oraz traceroute przeważnie dają wyniki podobne, jednakże ciężko jest z nich wyciągnąć wnioski bez znajomości dokładnej trasy obranej przez sygnał ping oraz przy timeoutach w programie traceroute. Program wireshark natomiast dostarcza niebywałych ilości informacji na temat pakietów wysyłanych i odbieranych, jednakże dotyczą one ruchu w sieci lokalnej oraz odbieranego/wysyłanego przez kartę sieciową, nie można zatem określić dokładnej trasy pakietu podczas jego drogi do serwera, jednakże z informacji zwrotnych wiadomo jak długą drogę pokonuje z powrotem. Można to sprawdzić za pomocą własności TTL, która domyślnie ustawiana jest na 64. W przypadku programu ping TTL zwrotny jest jedną z informacji zwracanych przez program, Wireshark również wychwytuje tą wartość w przychodzącym pakiecie. Pomiar taki z użyciem Traceroute jest niemożliwy, ponieważ nie jesteśmy w stanie otrzymać listy serwerów, przez które pakiet idzie z odległego serwera.

Co istotne, stosunkowo rzadko zdarza się, żeby droga do serwera była taka sama jak droga z powrotem – przeważnie zdarzają się rozbieżności rzędu co najmniej kilku węzłów. Może to wynikać np. ze skierowanej struktury pośrednich sieci.

W ramach odpowiedzenia na pytanie czy rozmiar pakietu wpływa na trasę pingu przeprowadziłem kilka testów. W przypadku 60000, 6000 oraz 600 bajtów maksymalny TTL został przekroczony, co zapewne oznacza, że pośrednie serwery nie przekazują takich dużych pingów (możliwe, że w celu zapobiegnięcia DoS oraz zwiększenia ilości obsługiwanych klientów na raz). Pakiet 100 bajtów natomiast pokonał nieco dłuższą drogę niż jego mniejszy odpowiednik, co oznacza, że dla małego i szybkiego ruchu obecne są specjalne serwery, a większe obciążenia są przenoszone na wolniejsze trasy. Przy małej ilości skoków, np. przy wywoływaniu strony wp.pl liczba skoków pozostawała taka sama dla 64B i 20kiB pakietów (10 skoków). Warto również zauważyć, że czas podróży pakietu jest proporcjonalny do jego rozmiaru.

Im więcej skoków, tym przeważnie mniejsze rozmiary pakietów są przepuszczane (przepuszczalność trasy = przepuszczalność najmniej przepuszczalnego serwera na niej).

Co bardzo ciekawe, w przypadku strony cnn.com liczba skoków przy mniejszym pakiecie zwiększa się o 1.