

Sprawozdanie

Lista nr 1

Józef Piechaczek

1. Wstęp

Celem sprawozdania jest przetestowanie działania programów Ping, Traceroute oraz WireShark. Za ich pomocą należy sprawdzić m. in. ilość węzłów na trasie do i od wybranego, odległego serwera, wpływ wielkości pakietów na czas propagacji, wpływ fragmentacji pakietów na ilość węzłów i czas propagacji, rozmiar największego niefragmentowanego pakietu, „średnicę” Internetu oraz odnaleźć trasę przebiegającej przez sieci wirtualne.

Ping to program, który wysyła zapytanie ICMP ECHO_REQUEST do odpowiedniego hosta, oczekując w odpowiedzi pakietu ICMP ECHO_RESPONSE. Pozwala on na testowanie połączeń sieciowych, zmierzenie liczby zagubionych pakietów, zmierzenie czasu propagacji oraz ilości węzłów pomiędzy hostami. Często jednak zachodzi przypadek, iż host nie odpowiada na zapytanie, co nie zawsze jest spowodowane brakiem połączenia - może oznaczać blokowanie odpowiedzi na zapytanie przez hosta.

Traceroute to program przedstawiający listę routerów pośredniczących w przesyłaniu informacji pomiędzy dwoma urządzeniami. Program wysyła zapytania ICMP do poszczególnego adresu, początkowo z TTL ustawionym na 1. Każdy router znajdujący się na trasie zmniejsza TTL o 1, a gdy wartość wynosi zero odsyła pakiet ICMP o kodzie 11 (Time to Live Exceeded). Pozwala to na uzyskanie adresów kolejnych routerów pośredniczących w komunikacji. Gdy zapytanie trafia do nadawcy odsyła on komunikat ICMP ECHO_RESPONSE co kończy pracę programu. Nie zawsze jednak udaje się uzyskać całą trasę, wtedy program kończy pracę po wykonaniu odpowiedniej liczby skoków, którą można ustawić odpowiednim parametrem (domyślnie: 30).

WireShark to graficzny analizator sieciowy. Pozwala na przechwytywanie i nagrywanie pakietów danych transmitowanych w sieci.

2. Wyszukiwanie serwerów

W celu znalezienia odległych serwerów odpowiadających na pakiety wysłane za pośrednictwem programu ping, posłużyłem się prostym skrypcem wysyłającym jeden pakiet do kolejnych adresów, począwszy od podanego w argumencie oraz zapisujący adresy serwerów, które odpowiedziały na zapytanie do pliku tekstowego.

Do znalezienia początkowych adresów posłużyłem się internetową bazą przypisującą krajom odpowiedni zakres adresów.

3. Sprawdzenie trasy do i od wybranego odległego serwera

W celu sprawdzenia trasy “do” odległego serwera posłużyłem się programem ping z opcją “-t” pozwalającą ustawić “Time to Live” pakietu.

Przykład:

```
dzazef@dzazef-laptop:~$ ping 203.20.244.140 -t 20
PING 203.20.244.140 (203.20.244.140) 56(84) bytes of data.
From 203.100.4.242 icmp_seq=1 Time to live exceeded
^C
--- 203.20.244.140 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss,

dzazef@dzazef-laptop:~$ ping 203.20.244.140 -t 21
PING 203.20.244.140 (203.20.244.140) 56(84) bytes of data.
64 bytes from 203.20.244.140: icmp_seq=1 ttl=48 time=449 ms
^C
--- 203.20.244.140 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 449.272/449.272/449.272/0.000 ms
dzazef@dzazef-laptop:~$ █
```

Rysunek 1

Na powyższej grafice widać, że pakiet z TTL ustawionym na 20 nie dotarł, natomiast dla pakietu z TTL ustawionym na 21 odpowiedź została uzyskana. Oznacza to, że na trasie do znajduje się 21 węzłów.

Aby sprawdzić trasę “od” posłużę się opcją ping bez dodatkowych parametrów.

Przykład:

```
dzazef@dzazef-laptop:~$ ping 203.20.244.140
PING 203.20.244.140 (203.20.244.140) 56(84) bytes of data.
64 bytes from 203.20.244.140: icmp_seq=1 ttl=48 time=349 ms
^C
--- 203.20.244.140 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 349.455/349.455/349.455/0.000 ms
dzazef@dzazef-laptop:~$ █
```

Rysunek 2

W tym przypadku po otrzymaniu pakietu ma on TTL ustawiony na 48, co pozwala przypuszczać, że TTL ustawiony przez serwer, który wysłał odpowiedź wynosił początkowo 64. Wiedząc, że każdy węzeł zmniejsza wartość TTL o 1, możemy obliczyć, iż ilość węzłów w drodze powrotnej wynosi 16, czyli różnie się od ilości węzłów na trasie “do”.

Powyższy test wykonałem dla serwerów znajdujących się w różnych odległościach geograficznych. Wyniki przedstawiam na poniższej tabeli:

Adres	Odległość (km)	Liczba węzłów (do)	Liczba węzłów (od)
203.20.244.140	15 000	21	16
190.185.198.67	12 500	16	15
42.61.129.133	9 300	20	14
65.255.68.226	8 300	20	11
178.236.7.220	1 600	16	15
212.77.98.9	400	12	7
156.17.7.22	100	16	13

Tabela 1

Patrząc na powyższe wyniki można zauważyć, iż niekoniecznie mniejsza odległość geograficzna oznacza mniejszą liczbę węzłów, choć w wielu wypadkach jest to prawdą. Można również zauważyć, że w powyższych przypadkach trasa “od” różni się średnio o około 25% w porównaniu do trasy “do”.

Trasę “do” można również badać za pomocą programu Traceroute:

```
dzazef@dzazef-laptop:~$ traceroute 156.17.7.22
traceroute to 156.17.7.22 (156.17.7.22), 30 hops max, 60 byte packets
 1 _gateway (192.168.43.1)  6.060 ms  6.104 ms  7.199 ms
 2 * * *
 3 * * *
 4 212.2.102.1 (212.2.102.1)  52.769 ms  52.779 ms  53.851 ms
 5 PoznR018MM01.pozn.bdi.inetia.pl (213.17.205.81)  61.473 ms  61.912 ms  62.758 ms
 6 83.238.248.35 (83.238.248.35)  66.028 ms  57.017 ms  57.935 ms
 7 * * *
 8 156.17.250.215 (156.17.250.215)  74.173 ms  66.599 ms  67.529 ms
 9 rolnik2-karkonosz.wask.wroc.pl (156.17.254.112)  71.021 ms  68.747 ms  69.191 ms
10 wazniak-rolnik.wask.wroc.pl (156.17.254.140)  86.058 ms  64.298 ms  64.268 ms
11 z-wask2-do-pwr2.pwrnet.pwr.wroc.pl (156.17.18.244)  61.443 ms  57.062 ms  64.039 ms
12 156.17.33.1 (156.17.33.1)  61.388 ms  66.901 ms  69.124 ms
13 informatyka.im.pwr.wroc.pl (156.17.7.22)  69.109 ms  67.711 ms  67.725 ms
dzazef@dzazef-laptop:~$
```

Rysunek 3

Na podstawie powyższej grafiki można zauważyć, że liczba węzłów na trasie “do” wyniosła 13, co pokrywa się z wynikiem uzyskanym za pomocą programu ping.

4. Maksymalny rozmiar niefragmentowanego pakietu

W celu sprawdzenia maksymalnego rozmiaru niefragmentowanego pakietu posłużyłem się programem ping z opcją “-M do”, która nie pozwala na fragmentowanie oraz opcją “-s” pozwalającą ustawić rozmiar wysyłanego pakietu.

```
dzazef@dzazef-laptop:~$ ping 42.61.129.133 -s 1472 -M do
PING 42.61.129.133 (42.61.129.133) 1472(1500) bytes of data.
1480 bytes from 42.61.129.133: icmp_seq=1 ttl=50 time=294 ms
^C
--- 42.61.129.133 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 293.577/293.577/293.577/0.000 ms
dzazef@dzazef-laptop:~$ ping 42.61.129.133 -s 1473 -M do
PING 42.61.129.133 (42.61.129.133) 1473(1501) bytes of data.
ping: local error: Message too long, mtu=1500
^C
--- 42.61.129.133 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
```

Rysunek 4

Z grafiki wynika, że pakiet o rozmiarze 1472 bajtów udało się przesłać, jednak pakiet o rozmiarze 1473 bajtów nie został wysłany, co pokazuje, że w danym przypadku maksymalny rozmiar pakietu wynosi 1472 bajtów. Do rozmiaru pakietu należy jeszcze doliczyć rozmiar nagłówka wynoszący 28 bajtów co łącznie daje MTU równe 1500 bajtów.

Powyższy test można potwierdzić przy pomocy programu WireShark obserwując czy wysłane pakiety są fragmentowane:

```

ICMP      35 Echo (ping) request id=0x17f4, seq=2/512, ttl=64 (reply in
IPv4     1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=8105) [Reass
ICMP      60 Echo (ping) reply id=0x17f4, seq=2/512, ttl=50 (request i
IPv4     1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=de91) [Reass
ICMP      35 Echo (ping) request id=0x17f4, seq=3/768, ttl=64 (reply in
IPv4     1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=8106) [Reass
ICMP      60 Echo (ping) reply id=0x17f4, seq=3/768, ttl=50 (request i
ICMP     1514 Echo (ping) request id=0x17f5, seq=1/256, ttl=64 (reply in
ICMP     1514 Echo (ping) reply id=0x17f5, seq=1/256, ttl=50 (request i
ICMP     1514 Echo (ping) request id=0x17f5, seq=2/512, ttl=64 (reply in
ICMP     1514 Echo (ping) reply id=0x17f5, seq=2/512, ttl=50 (request i

```

Rysunek 5

5. Wpływ rozmiaru pakietu na ilość węzłów

W celu sprawdzenia wpływu rozmiaru pakietu na ilość węzłów posłużyłem się programem ping z opcją “-s” pozwalającą ustawić rozmiar wysyłanego pakietu. Dla każdego adresu wykonałem dodatkowo dwie próby, jedną dla pakietu o rozmiarze 1472B (niefragmentowany) oraz drugą dla 10000B (fragmentowany).

Adres	Liczba węzłów (od)	Liczba węzłów (od, 1472B)	Liczba węzłów (od, 10000B)
203.20.244.140	16	16	16
190.185.198.67	15	15	-
42.61.129.133	14	14	14
65.255.68.226	11	11	11
178.236.7.220	15	15	-
212.77.98.9	7	7	7
156.17.7.22	12	12	12

Tabela 2

Jak można zauważyć, rozmiar pakietu nie wpłynął na ilość węzłów, jednak niektóre serwery nie zwróciły odpowiedzi dla pakietów fragmentowanych.

6. Wpływ fragmentacji i rozmiaru pakietu na czas propagacji

W celu sprawdzenia wpływu fragmentacji pakietów posłużyłem się programem ping, z opcją “-s” dla której ustalałem różne rozmiary pakietów z zakresu 56-55000, oraz opcją “-c 100” w celu wysłania zapytania stukrotnie. Następnie sprawdzałem średni czas propagacji.

Adres	Odległość (km)	Rozmiar pakietu							
		54B (ms)	1472B (ms)	5000B (ms)	15000B (ms)	25000B (ms)	35000B (ms)	45000B (ms)	55000B (ms)
203.20.244.140	15 000	350	350	353	358	362	376	372	-
wikipedia.com	1000	78	94	110	139	167	189	211	256
212.77.98.9	400	16	17	19	24	27	31	36	40

Tabela 3

Zgodnie z powyższymi wynikami czas propagacji wzrasta wraz z rozmiarem wysyłanego pakietu niezależnie od odległości geograficznej serwera. Niektóre serwery jednak nie odpowiadają na fragmentowane lub zbyt duże pakiety.

Przykład wykonanego testu:

```
55008 bytes from 212.77.98.9: icmp_seq=95 ttl=57 time=40.4 ms
55008 bytes from 212.77.98.9: icmp_seq=96 ttl=57 time=39.5 ms
55008 bytes from 212.77.98.9: icmp_seq=97 ttl=57 time=39.6 ms
55008 bytes from 212.77.98.9: icmp_seq=98 ttl=57 time=39.3 ms
55008 bytes from 212.77.98.9: icmp_seq=99 ttl=57 time=39.8 ms
55008 bytes from 212.77.98.9: icmp_seq=100 ttl=57 time=39.5 ms

--- 212.77.98.9 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 175ms
rtt min/avg/max/mdev = 38.843/39.745/45.882/0.779 ms
dzazef@dzazef-laptop:~$ █
```

Rysunek 6

7. Poszukiwanie najdłuższej ścieżki

W celu znalezienia najdłuższej ścieżki posłużyłem się bazą adresów oraz skryptem wspomnianym wcześniej. Dzięki niemu uzyskałem 13345 adresów, które odpowiedziały na zapytanie ping.

Aby wybrać najbardziej odległy serwer posłużyłem się skryptem, który testował ilość węzłów “do” zgodnie z metodą opisaną w punkcie 3. Wyniki przedstawiłem w poniższej tabeli:

	Liczba węzłów											
	dowolna	>22	>23	>24	>25	>26	>27	>28	>29	>30	>31	>32
Liczba adresów	13345	1876	1446	1189	722	128	20	16	2	2	2	2

Tabela 4

Zgodnie z powyższymi wynikami odległość dla dwóch serwerów była wyższa niż 32 węzły. Po sprawdzeniu programem Traceroute (z opcją “-m 255”, która ustawia maksymalną liczbę węzłów na 255) ścieżki dla powyższych adresów uzyskałem następujące wyniki:

```
238 241.207.185.190.core.ridsa.com.ar (190.185.207.241) 308.231 ms 329.207 ms 328.739 ms
239 winifreda.ridsa.com.ar (190.185.207.242) 328.385 ms 314.807 ms 314.702 ms
240 241.207.185.190.core.ridsa.com.ar (190.185.207.241) 306.607 ms 311.240 ms 307.943 ms
241 winifreda.ridsa.com.ar (190.185.207.242) 312.590 ms 311.840 ms 316.070 ms
242 241.207.185.190.core.ridsa.com.ar (190.185.207.241) 309.385 ms 310.891 ms 315.384 ms
243 winifreda.ridsa.com.ar (190.185.207.242) 310.873 ms 312.838 ms *
244 241.207.185.190.core.ridsa.com.ar (190.185.207.241) 312.460 ms 312.498 ms 306.494 ms
245 winifreda.ridsa.com.ar (190.185.207.242) 317.042 ms 310.783 ms 313.133 ms
246 241.207.185.190.core.ridsa.com.ar (190.185.207.241) 316.231 ms 316.217 ms 316.549 ms
247 winifreda.ridsa.com.ar (190.185.207.242) 313.853 ms 309.796 ms 313.404 ms
248 241.207.185.190.core.ridsa.com.ar (190.185.207.241) 309.236 ms 315.130 ms 315.158 ms
249 * * *
250 241.207.185.190.core.ridsa.com.ar (190.185.207.241) 314.625 ms 318.461 ms 312.973 ms
251 * * *
252 * * *
253 * * *
254 * * *
255 * winifreda.ridsa.com.ar (190.185.207.242) 313.903 ms 316.431 ms
dzazef@dzazef-laptop:~$ █
```

Rysunek 7

```

242 npt-edg-rt3.bnin.net (66.170.44.23) 150.816 ms 150.870 ms 149.904 ms
243 npt-edg-rt1.bnin.net (66.170.44.1) 149.645 ms 149.372 ms 149.677 ms
244 npt-edg-rt3.bnin.net (66.170.44.23) 148.834 ms 149.554 ms 149.525 ms
245 npt-edg-rt1.bnin.net (66.170.44.1) 150.514 ms 149.744 ms 150.114 ms
246 npt-edg-rt3.bnin.net (66.170.44.23) 151.027 ms 150.799 ms 150.262 ms
247 npt-edg-rt1.bnin.net (66.170.44.1) 151.205 ms 150.654 ms 150.607 ms
248 npt-edg-rt3.bnin.net (66.170.44.23) 150.018 ms 150.226 ms 150.007 ms
249 npt-edg-rt1.bnin.net (66.170.44.1) 149.766 ms 150.167 ms 149.790 ms
250 npt-edg-rt3.bnin.net (66.170.44.23) 150.968 ms 150.189 ms 150.854 ms
251 npt-edg-rt1.bnin.net (66.170.44.1) 150.518 ms 151.260 ms 150.860 ms
252 npt-edg-rt3.bnin.net (66.170.44.23) 151.334 ms 151.331 ms 151.645 ms
253 npt-edg-rt1.bnin.net (66.170.44.1) 151.552 ms 150.343 ms 150.598 ms
254 npt-edg-rt3.bnin.net (66.170.44.23) 150.273 ms 150.564 ms 150.506 ms
255 npt-edg-rt1.bnin.net (66.170.44.1) 150.903 ms 150.426 ms 151.096 ms
dzazef@dzazef-laptop:~$ █

```

Rysunek 8

Można dostrzec, że w obu przypadkach wysłany pakiet po trafieniu na odpowiedni serwer “zapętlił się” i był przesyłany pomiędzy dwoma serwerami. Można to również dostrzec w programie ping dla TTL ustawionego na maksymalną wartość (255):

```

dzazef@dzazef-laptop:~$ ping 66.249.234.114 -t 255 -c 1
PING 66.249.234.114 (66.249.234.114) 56(84) bytes of data.
From 66.170.44.1 icmp_seq=1 Time to live exceeded

--- 66.249.234.114 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

dzazef@dzazef-laptop:~$ █

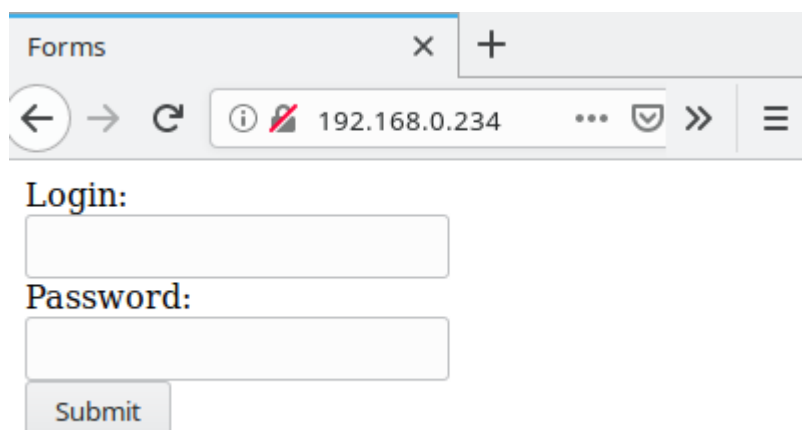
```

Rysunek 9

Jeśli więc pominąć powyższe dwa przypadki, największa znaleziona przeze mnie ścieżka wynosiła 29 węzłów (dla 14 serwerów).

8. Przechwytywanie hasła za pomocą programu WireShark.

W celu pokazania przechwytywania przykładowego hasła podanego na niezabezpieczonej witrynie posłużę się prostą stroną, korzystającą z metody POST:



The image shows a web browser window with a single tab titled 'Forms'. The address bar displays the IP address '192.168.0.234'. Below the address bar is a login form with two input fields: 'Login:' and 'Password:'. A 'Submit' button is located at the bottom of the form.

Rysunek 10

Po wysłaniu zapytania i sprawdzeniu nagranych rekordów w programie WireShark można zauważyć m.in. następujący pakiet:

No.	Time	Source	Destination	Protocol	Length	Info
105	6.339651943	192.168.0.234	192.168.0.234	TCP	76	39534 → 80 [SYN, ECN, Seq=39534, Win=0, Len=0]
106	6.339676965	192.168.0.234	192.168.0.234	TCP	76	80 → 39534 [SYN, ACK, Seq=39534, Win=0, Len=0]
107	6.339690788	192.168.0.234	192.168.0.234	TCP	68	39534 → 80 [ACK] Seq=39534, Win=0, Len=0
108	6.339799325	192.168.0.234	192.168.0.234	HTTP	556	POST /php/action.php
109	6.339828290	192.168.0.234	192.168.0.234	TCP	68	80 → 39534 [ACK] Seq=39534, Win=0, Len=0
110	6.340461073	192.168.0.234	192.168.0.234	HTTP	576	HTTP/1.1 404 Not Found
111	6.340488466	192.168.0.234	192.168.0.234	TCP	68	39534 → 80 [ACK] Seq=39534, Win=0, Len=0

▶ Frame 108: 556 bytes on wire (4448 bits), 556 bytes captured (4448 bits) on interface 0
 ▶ Linux cooked capture
 ▶ Internet Protocol Version 4, Src: 192.168.0.234, Dst: 192.168.0.234
 ▶ Transmission Control Protocol, Src Port: 39534, Dst Port: 80, Seq: 1, Ack: 1, Len: 488
 ▶ Hypertext Transfer Protocol
 ▶ HTML Form URL Encoded: application/x-www-form-urlencoded
 ▶ Form item: "login" = "login"
 ▶ Form item: "password" = "tajnehaslo"

Rysunek 11

Wśród przesłanych danych można dostrzec informację o podanym loginie i haśle.

9. Przechwytywanie innych pakietów za pomocą programu Wireshark.

Za pomocą programu Wireshark możemy również przechwycić wymianę kluczy pomiędzy klientem a serwerem:

3878	39.974879601	2a00:86c0:4:4::161	2a02:a317:e23c:a200...	TLSv1.2	3769	Server Hello, Certificate
3879	39.974898939	2a02:a317:e23c:a200...	2a00:86c0:4:4::161	TCP	86	60554 → 443 [ACK] Seq=60554, Win=0, Len=0
3880	39.976042366	2a02:a317:e23c:a200...	2a00:86c0:4:4::161	TLSv1.2	179	Client Key Exchange

▶ [Timestamps]
 [Time since first frame in this TCP stream: 0.118000001 seconds]
 [Time since previous frame in this TCP stream: 0.001143427 seconds]
 TCP payload (93 bytes)
 ▶ Secure Sockets Layer
 ▶ TLSv1.2 Record Layer: Handshake Protocol: Client Key Exchange
 Content Type: Handshake (22)
 Version: TLS 1.2 (0x0303)
 Length: 37
 ▶ Handshake Protocol: Client Key Exchange
 Handshake Type: Client Key Exchange (16)
 Length: 33
 ▶ EC Diffie-Hellman Client Params
 Pubkey Length: 32
 Pubkey: 2ca2629f29e6e68da79aa6a1e155c62c58e1d40a024426e7...
 ▶ TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
 Content Type: Change Cipher Spec (20)
 Version: TLS 1.2 (0x0303)
 Length: 1

Rysunek 12

1886	20.011245198	2620:1ec:a92::171	2a02:a317:e23c:a200...	TCP	74	443 → 46230 [ACK] Seq=826 Ack=5899 Win=1026 Len=0
1887	20.018653898	2a00:1450:401b:801::	2a02:a317:e23c:a200...	TLSv1.2	3682	Server Hello, Certificate, Server Key Exchange

▶ TLSv1.2 Record Layer: Handshake Protocol: Server Key Exchange
 Content Type: Handshake (22)
 Version: TLS 1.2 (0x0303)
 Length: 116
 ▶ Handshake Protocol: Server Key Exchange
 Handshake Type: Server Key Exchange (12)
 Length: 112
 ▶ EC Diffie-Hellman Server Params
 Curve Type: named_curve (0x03)
 Named Curve: x25519 (0x001d)
 Pubkey Length: 32
 Pubkey: 6ac7c3ed5fc4290b9aaef85de449758893ed85f94013ac1a...
 ▶ Signature Algorithm: ecdsa_secp256r1_sha256 (0x0403)
 Signature Hash Algorithm Hash: SHA256 (4)
 Signature Hash Algorithm Signature: ECDSA (3)
 Signature Length: 72
 Signature: 304602210083cd4e88a0d2a1da35454368b6b8cc3780a7db...

Rysunek 13

Na podstawie powyższych informacji można uzyskać wiele informacji, jak np. zastosowany algorytm hashowania lub klucz publiczny.

10. Źródła

- Podręczniki programów Ping, Traceroute i Wireshark w systemie Linux
- <https://lite.ip2location.com/ip-address-ranges-by-country>
- <https://www.ip2location.com>

11. Kody programów

```
#!/bin/bash
if (( $# != 3 )); then
    echo "Usage: ./ipscanner <start_ip> <output_file> <timeout>"
    exit
fi

a=$(echo $1 | cut -d '.' -f 1)
b=$(echo $1 | cut -d '.' -f 2)
c=$(echo $1 | cut -d '.' -f 3)
d=$(echo $1 | cut -d '.' -f 4)

for (( i=$a; $i <= 255; i++ )) ; do
    for (( j=$b; $j <= 255; j++ )) ; do
        for (( k=$c; $k <= 255; k++ )) ; do
            for (( l=$d; $l <= 255; l++ )) ; do
                ip=$(printf "%s.%s.%s.%s\n" $i $j $k $l)
                ttl=$(ping -c 1 -W $3 $ip | grep ttl | cut -d '=' -f 3 |
cut -d ' ' -f 1)
                echo $ip
                if [ ! -z "$ttl" ]
                then
                    echo "${ttl}  ${ip}" >> $2.txt
                fi
            done
        done
    done
done
```

Skrypt 1.

```
#!/bin/bash

if (( $# != 3 )); then
    echo "Usage: ./ttl <file> <ttl> <show_output>"
    exit
fi

NUMOFLINES=$(wc -l < $1)
COUNTER=0

while read LINE
do
    (( COUNTER++ ))
    IP=$(echo $LINE | cut -d ' ' -f 2)
    PINGOUT=$(ping -W 5 -c 1 -t $2 $IP | sed '/^[[[:space:]]*$/d')
    TTL=$(echo ${PINGOUT} | grep "Time to live exceeded")
    if [ -z "$TTL" ]
    then
        echo "${COUNTER}/${NUMOFLINES}: ${IP} ✕ "
    else
        echo "${COUNTER}/${NUMOFLINES}: ${IP} ✓ "
        echo $IP >> "$2f$1"
    fi
    if [ "$3" = "yes" ]
    then
        echo
        echo "$PINGOUT"
        echo
    fi
done < $1
```

Skrypt 2.