1. Traceroute

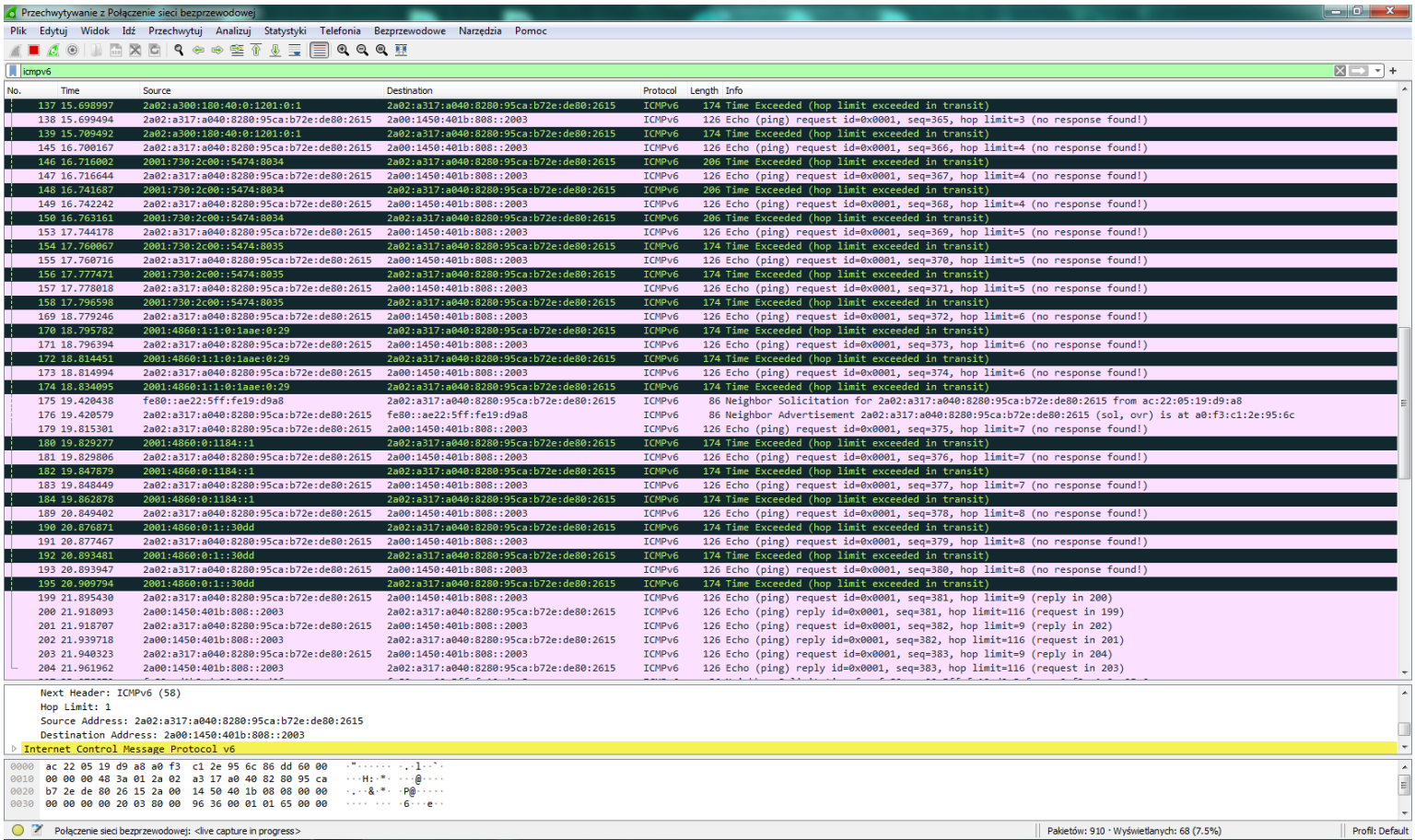
Używając komendy tracert -d (gdzie -d oznacza, że ma nie rozpoznawać adresów jako nazw hostów), można zobaczyć jaką drogę pokonuje pakiet i ile routerów spotyka.

```
Sposób użycia: tracert [-d] [-h maks_przes] [-j lista_hostów] [-w limit_czasu]
               [-R] [-S adres_źródłowy] [-4] [-6] nazwa_celu

Opcje:
    -d                  Nie rozpoznawaj adresów jako nazw hostów.
    -h maks_przes       Maksymalna liczba przeskoków w poszukiwaniu celu.
    -j lista_hostów     Swobodna trasa źródłowa według listy lista_hostów
                        (tylko IPv4).
    -w limit_czasu      Limit czasu oczekiwania na odpowiedź w milisekundach.
    -R                  Śledź ścieżkę błądzenia (tylko IPv6).
    -S adres_źródłowy   Adres źródłowy do użycia (tylko IPv6).
    -4                  Wymuś używanie IPv4.
    -6                  Wymuś używanie IPv6.
```

Poniżej inne dodatkowe opcje tej komendy:
W tym przypadku pakiet dociera do witryny google.pl w 9 przeskokach, jednak liczba ta może być różna przy każdym wywołaniu komendy oraz może mijać inne routery.
Jeśli wśród wyników pojawia się gwiazdka (*), oznacza to, że nie było odpowiedzi na wysłany pakiet, a powodem mógł być przykładowo problem z siecią.

```
Śledzenie trasy do www.google.pl [2a00:1450:401b:808::2003]
z maksymalną liczbą 30 przeskoków:

  1     3 ms     3 ms     3 ms  2a02:a317:a040:8280:ae22:5ff:fe19:d9a8
  2   267 ms    32 ms    13 ms  2a02:a304:0:177::1
  3    17 ms     9 ms    10 ms  2a02:a300:180:40:0:1201:0:1
  4    15 ms    25 ms    20 ms  2001:730:2c00::5474:8034
  5    15 ms    16 ms    18 ms  2001:730:2c00::5474:8035
  6    16 ms    18 ms    19 ms  2001:4860:1:1:0:1aae:0:29
  7    14 ms    18 ms    14 ms  2001:4860:0:1184::1
  8    27 ms    16 ms    15 ms  2001:4860:0:1::30dd
  9    22 ms    21 ms    21 ms  2a00:1450:401b:808::2003

Śledzenie zakończone.
```

W Wiresharku zawężając poszukiwania przy pomocy icmp lub w tym przypadku icmpv6, można zobaczyć jak wygląda badanie trasy pakietów.



Można zauważyć, że na każdy spotkany router przypadają trzy echo pingi oraz zwracane jest Time Exceeded, po czym wzrasta hop limit. Kiedy pakiet dotrze do celu, zwracana jest odpowiedź (echo reply).

2. Analiza ruchu DNS:

Aby wyczyścić cache DNS, wpisujemy w terminalu ipconfig /flushdns

```
Konfiguracja IP systemu Windows

Pomyślnie opróżniono pamięć podręczną programu rozpoznawania nazw DNS.
```

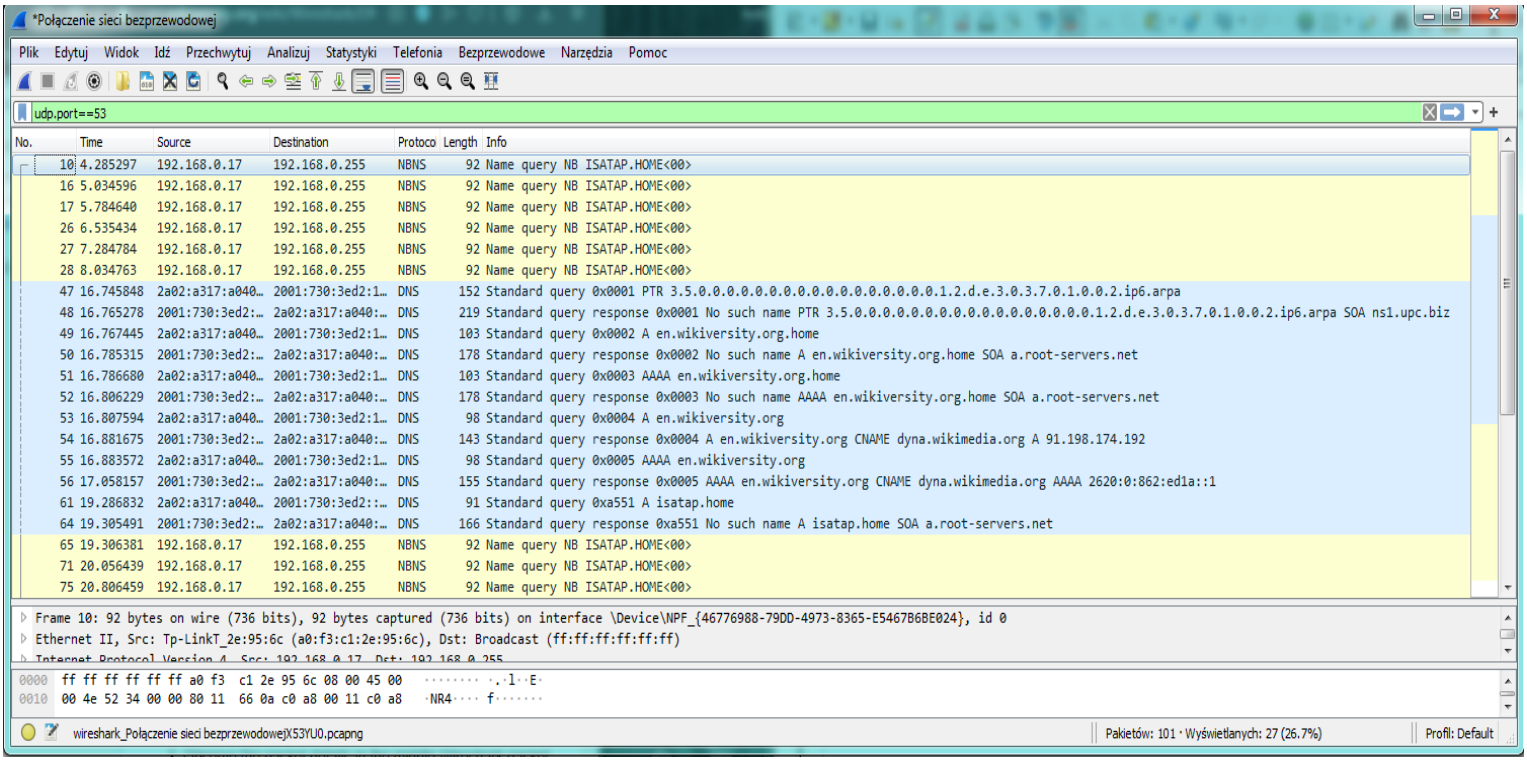
następnie można sprawdzić czy czyszczenie było skuteczne komendą ipconfig /displaydns

```
Konfiguracja IP systemu Windows

    isatap.home
    ----------------------------------------
    Nazwa nie istnieje.
```

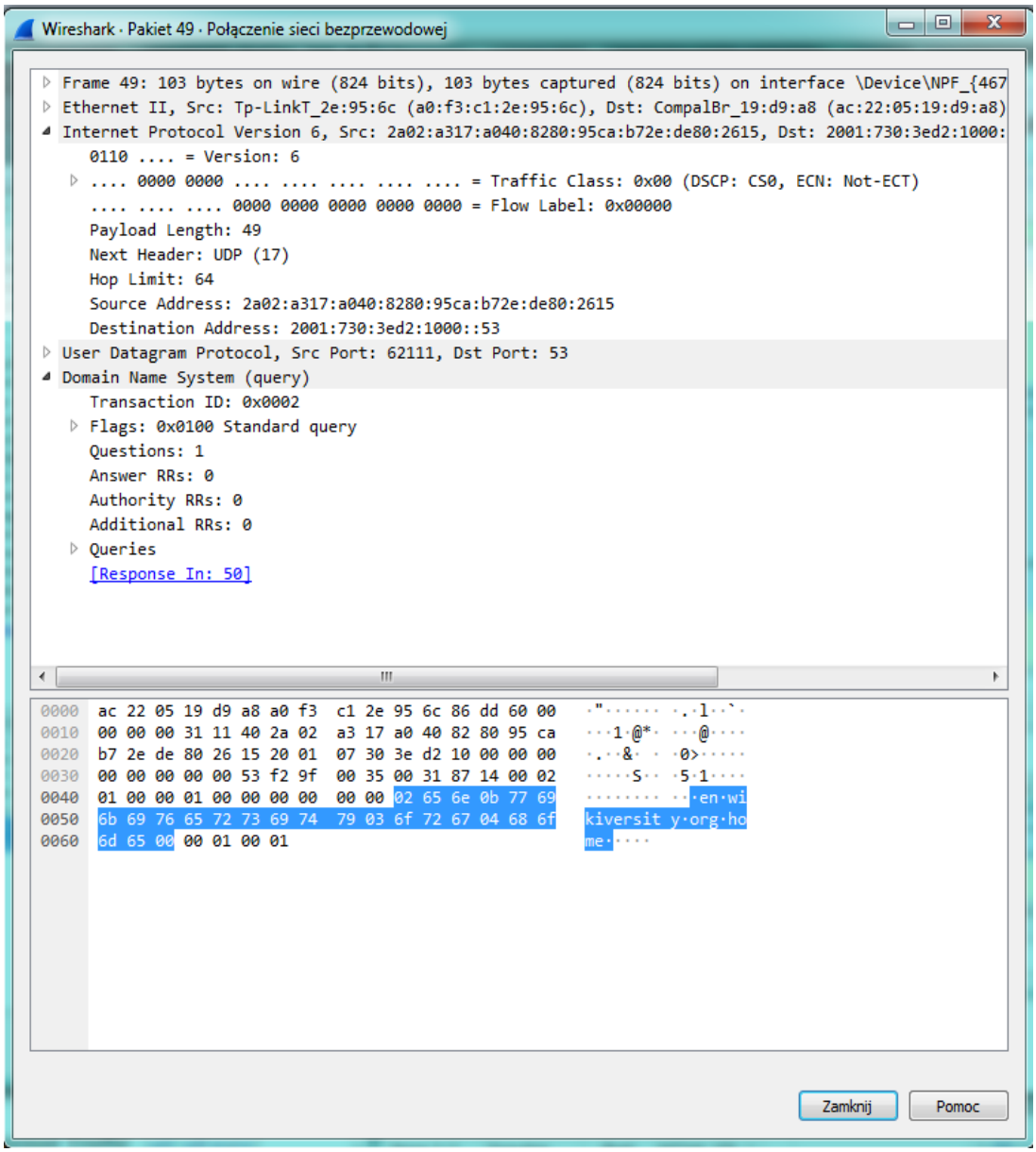
I wpisujemy np. nslookup en.wikiversity.org

```
Serwer:  UnKnown
Address:  2001:730:3ed2:1000::53

Nieautorytatywna odpowiedź:
Nazwa:    dyna.wikimedia.org
Addresses:  2620:0:862:ed1a::1
          91.198.174.192
Aliases:  en.wikiversity.org
```

W Wiresharku filtrujemy według portu 53 komendą udp.port == 53



Sprawdzając pakiety, można zauważyć, że Source Address odpowiada naszemu adresowi IP, natomiast Destination Address odpowiada adresowi IP serwera DNS.
Ponadto można sprawdzić, że adres DNS nie jest szyfrowany, tj. jest zawarty w bajtach 74-98, które przyporządkowane są nazwie.



Analiza ruchu HTTP:

W przeglądarce wpisujemy wybrany adres strony internetowej korzystającej z HTTP.

ip.addr == 69.94.77.202

No.	Time	Source	Destination	Protocol	Length	Info
2035	15.706193	192.168.0.17	69.94.77.202	TCP	66	65296 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
2036	15.752351	69.94.77.202	192.168.0.17	TCP	62	443 → 65296 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1420 WS=128
2037	15.752461	192.168.0.17	69.94.77.202	TCP	54	65296 → 443 [ACK] Seq=1 Ack=1 Win=17040 Len=0
2038	15.752857	192.168.0.17	69.94.77.202	TLSv1.2	236	Client Hello
2040	15.807130	69.94.77.202	192.168.0.17	TCP	56	443 → 65296 [ACK] Seq=1 Ack=183 Win=30336 Len=0
2041	15.809612	69.94.77.202	192.168.0.17	TLSv1.2	1474	Server Hello
2042	15.810717	69.94.77.202	192.168.0.17	TCP	1474	443 → 65296 [ACK] Seq=1421 Ack=183 Win=30336 Len=1420 [TCP segment of a reassembled PDU]
2043	15.810717	69.94.77.202	192.168.0.17	TLSv1.2	481	Certificate, Server Key Exchange, Server Hello Done
2044	15.810771	192.168.0.17	69.94.77.202	TCP	54	65296 → 443 [ACK] Seq=183 Ack=3268 Win=17040 Len=0
2045	15.816351	192.168.0.17	69.94.77.202	TLSv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
2046	15.868318	69.94.77.202	192.168.0.17	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message

Trzy pierwsze pakiety zarejestrowane przez Wiresharka to TCP three-way handshake. SYN wysłany do celu (69.94.77.202), serwer odsyła segmenty SYN oraz ACK, a następnie odpowiedź potwierdzenia odebrania segmentu wysłana przez nadawcę.

Następnie poprzez protokół TLS klient wysyła wiadomość (Client Hello) wraz z wersją protokołu i innymi informacjami, na którą odpowiada serwer odsyłając Server Hello wraz z informacjami o połączniu. Następuje wymiana klucza publicznego serwera i przejście do kolejnej fazy łączenia (Server Hello Done).

Klient wysyła wstępny klucz sesji, zaszyfrowany za pomocą otrzymanego klucza publicznego, po czym klient i serwer generują klucz sesji. Od momentu nadania Change Cipher Spec możliwe jest przesyłanie zaszyfrowanych danych.

3. Cache

Przy pierwszym wejściu na wybraną stronę, można zaobserwować TCP handshake, hosty przesyłają kolejno segmenty SYN, SYN ACK oraz ACK, po czym przesyłane są Client Hello etc. oraz dane.

ip.addr == 128.30.52.100

No.	Time	Source	Destination	Protocol	Length	Info
10	1.682336	192.168.0.17	128.30.52.100	TCP	66	52567 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
13	1.824773	128.30.52.100	192.168.0.17	TCP	66	443 → 52567 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1420 SACK_PERM=1 WS=512
14	1.824904	192.168.0.17	128.30.52.100	TCP	54	52567 → 443 [ACK] Seq=1 Ack=1 Win=17040 Len=0
15	1.840891	192.168.0.17	128.30.52.100	TLSv1.3	571	Client Hello
19	1.991024	128.30.52.100	192.168.0.17	TLSv1.3	1474	Server Hello, Change Cipher Spec, Application Data
20	1.993711	128.30.52.100	192.168.0.17	TCP	1474	443 → 52567 [ACK] Seq=1421 Ack=518 Win=67072 Len=1420 [TCP segment of a reassembled PDU]
21	1.993788	192.168.0.17	128.30.52.100	TCP	54	52567 → 443 [ACK] Seq=518 Ack=2841 Win=17040 Len=0
22	1.994834	128.30.52.100	192.168.0.17	TCP	1310	443 → 52567 [PSH, ACK] Seq=2841 Ack=518 Win=67072 Len=1256 [TCP segment of a reassembled PDU]
23	1.994834	128.30.52.100	192.168.0.17	TLSv1.3	1474	Application Data [TCP segment of a reassembled PDU]
24	1.994834	128.30.52.100	192.168.0.17	TLSv1.3	160	Application Data, Application Data
25	1.994905	192.168.0.17	128.30.52.100	TCP	54	52567 → 443 [ACK] Seq=518 Ack=5623 Win=17040 Len=0
26	2.012042	192.168.0.17	128.30.52.100	TLSv1.3	118	Change Cipher Spec, Application Data
27	2.013568	192.168.0.17	128.30.52.100	TLSv1.3	560	Application Data
29	2.150548	128.30.52.100	192.168.0.17	TLSv1.3	133	Application Data
30	2.151634	128.30.52.100	192.168.0.17	TLSv1.3	133	Application Data
31	2.151810	192.168.0.17	128.30.52.100	TCP	54	52567 → 443 [ACK] Seq=1088 Ack=5781 Win=16880 Len=0
32	2.156772	128.30.52.100	192.168.0.17	TLSv1.3	100	Application Data
33	2.157276	192.168.0.17	128.30.52.100	TLSv1.3	85	Application Data
36	2.184167	128.30.52.100	192.168.0.17	TCP	1474	443 → 52567 [ACK] Seq=5827 Ack=1088 Win=68096 Len=1420 [TCP segment of a reassembled PDU]
37	2.184473	128.30.52.100	192.168.0.17	TCP	1474	443 → 52567 [ACK] Seq=7247 Ack=1088 Win=68096 Len=1420 [TCP segment of a reassembled PDU]
38	2.184473	128.30.52.100	192.168.0.17	TCP	1474	443 → 52567 [ACK] Seq=8667 Ack=1088 Win=68096 Len=1420 [TCP segment of a reassembled PDU]
39	2.184473	128.30.52.100	192.168.0.17	TCP	1474	443 → 52567 [ACK] Seq=10087 Ack=1088 Win=68096 Len=1420 [TCP segment of a reassembled PDU]
40	2.184578	192.168.0.17	128.30.52.100	TCP	54	52567 → 443 [ACK] Seq=1119 Ack=11507 Win=17040 Len=0
41	2.184772	128.30.52.100	192.168.0.17	TCP	1474	443 → 52567 [ACK] Seq=11507 Ack=1088 Win=68096 Len=1420 [TCP segment of a reassembled PDU]
42	2.184772	128.30.52.100	192.168.0.17	TLSv1.3	1418	Application Data
43	2.184772	128.30.52.100	192.168.0.17	TCP	1474	443 → 52567 [ACK] Seq=14291 Ack=1088 Win=68096 Len=1420 [TCP segment of a reassembled PDU]
44	2.185014	192.168.0.17	128.30.52.100	TCP	54	52567 → 443 [ACK] Seq=1119 Ack=15711 Win=17040 Len=0
46	2.290079	128.30.52.100	192.168.0.17	TCP	1474	443 → 52567 [ACK] Seq=15711 Ack=1088 Win=68096 Len=1420 [TCP segment of a reassembled PDU]
47	2.290685	128.30.52.100	192.168.0.17	TCP	1474	443 → 52567 [ACK] Seq=17131 Ack=1088 Win=68096 Len=1420 [TCP segment of a reassembled PDU]
48	2.290685	128.30.52.100	192.168.0.17	TCP	1474	443 → 52567 [ACK] Seq=18551 Ack=1088 Win=68096 Len=1420 [TCP segment of a reassembled PDU]
49	2.290685	128.30.52.100	192.168.0.17	TCP	1474	443 → 52567 [ACK] Seq=19971 Ack=1088 Win=68096 Len=1420 [TCP segment of a reassembled PDU]
50	2.291547	192.168.0.17	128.30.52.100	TCP	54	52567 → 443 [ACK] Seq=1119 Ack=21391 Win=17040 Len=0
51	2.322388	128.30.52.100	192.168.0.17	TLSv1.3	1474	Application Data [TCP segment of a reassembled PDU]
52	2.322909	128.30.52.100	192.168.0.17	TCP	1474	443 → 52567 [ACK] Seq=22811 Ack=1119 Win=68096 Len=1420 [TCP segment of a reassembled PDU]
53	2.322909	128.30.52.100	192.168.0.17	TCP	1474	443 → 52567 [ACK] Seq=24231 Ack=1119 Win=68096 Len=1420 [TCP segment of a reassembled PDU]
54	2.322909	128.30.52.100	192.168.0.17	TCP	1474	443 → 52567 [ACK] Seq=25651 Ack=1119 Win=68096 Len=1420 [TCP segment of a reassembled PDU]
55	2.323018	192.168.0.17	128.30.52.100	TCP	54	52567 → 443 [ACK] Seq=1119 Ack=27071 Win=17040 Len=0
56	2.323192	128.30.52.100	192.168.0.17	TCP	1474	443 → 52567 [ACK] Seq=27071 Ack=1119 Win=68096 Len=1420 [TCP segment of a reassembled PDU]
57	2.323192	128.30.52.100	192.168.0.17	TCP	1474	443 → 52567 [ACK] Seq=28491 Ack=1119 Win=68096 Len=1420 [TCP segment of a reassembled PDU]
58	2.323192	128.30.52.100	192.168.0.17	TCP	1474	443 → 52567 [ACK] Seq=29911 Ack=1119 Win=68096 Len=1420 [TCP segment of a reassembled PDU]
59	2.323421	192.168.0.17	128.30.52.100	TCP	54	52567 → 443 [ACK] Seq=1119 Ack=31331 Win=17040 Len=0

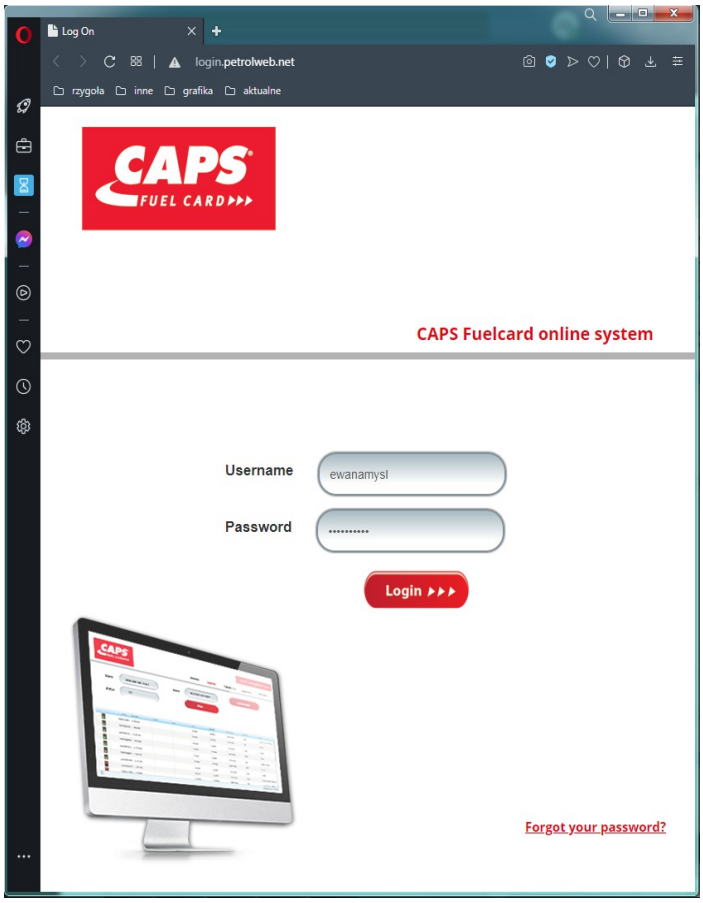
Z kolei po odświeżeniu strony w przeglądarce nie następuje wymiana handshake'ów.

ip.addr == 128.30.52.100

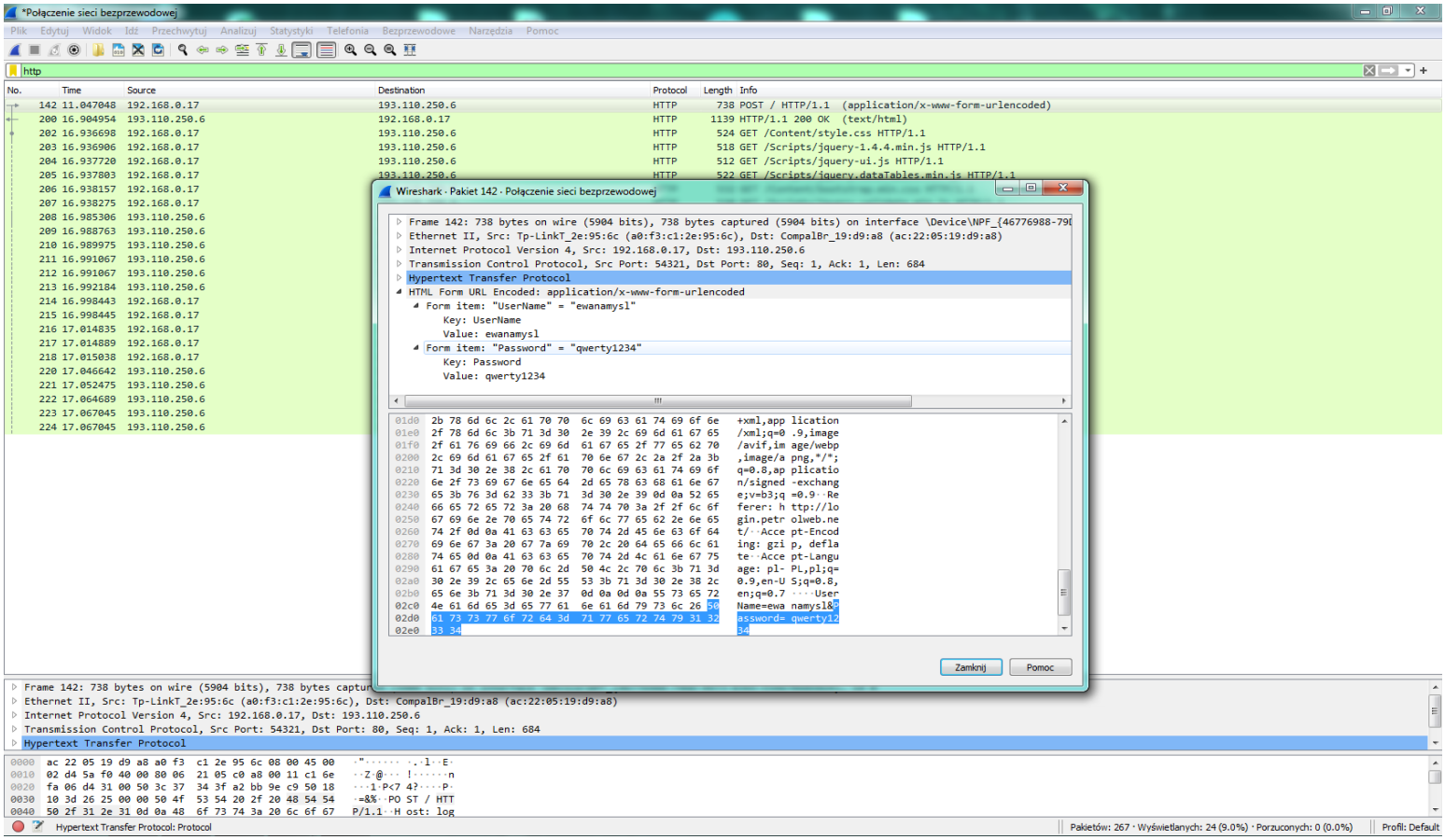
No.	Time	Source	Destination	Protocol	Length	Info
14	1.599984	192.168.0.17	128.30.52.100	TLSv1.2	168	Application Data
17	1.827280	128.30.52.100	192.168.0.17	TLSv1.2	93	Application Data
19	1.871220	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=40 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
20	1.871330	192.168.0.17	128.30.52.100	TCP	54	53121 → 443 [ACK] Seq=115 Ack=1460 Win=65320 Len=0
21	1.872083	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=1460 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
22	1.872083	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=2880 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
23	1.872083	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=4300 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
24	1.872139	192.168.0.17	128.30.52.100	TCP	54	53121 → 443 [ACK] Seq=115 Ack=5720 Win=65320 Len=0
25	1.872527	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=5720 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
26	1.872527	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=7140 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
27	1.872527	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=8560 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
28	1.872573	192.168.0.17	128.30.52.100	TCP	54	53121 → 443 [ACK] Seq=115 Ack=9980 Win=65320 Len=0
29	1.873628	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=9980 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
30	1.873628	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=11400 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
31	1.873700	192.168.0.17	128.30.52.100	TCP	54	53121 → 443 [ACK] Seq=115 Ack=12820 Win=65320 Len=0
34	2.098834	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=12820 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
35	2.099486	128.30.52.100	192.168.0.17	TLSv1.2	1474	Application Data [TCP segment of a reassembled PDU]
36	2.099486	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=15660 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
37	2.099486	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=17080 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
38	2.099570	192.168.0.17	128.30.52.100	TCP	54	53121 → 443 [ACK] Seq=115 Ack=18500 Win=65320 Len=0
39	2.099850	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=18500 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
40	2.099850	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=19920 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
41	2.099850	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=21340 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
42	2.099919	192.168.0.17	128.30.52.100	TCP	54	53121 → 443 [ACK] Seq=115 Ack=22760 Win=65320 Len=0
43	2.100186	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=22760 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
44	2.100186	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=24180 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
45	2.100186	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=25600 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
46	2.100239	192.168.0.17	128.30.52.100	TCP	54	53121 → 443 [ACK] Seq=115 Ack=27020 Win=65320 Len=0
47	2.100532	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=27020 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
48	2.100532	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=28440 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
49	2.100532	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=29860 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
50	2.100777	192.168.0.17	128.30.52.100	TCP	54	53121 → 443 [ACK] Seq=115 Ack=31280 Win=65320 Len=0
51	2.101441	128.30.52.100	192.168.0.17	TLSv1.2	1474	Application Data [TCP segment of a reassembled PDU]
52	2.101441	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=32700 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
53	2.101441	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=34120 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
54	2.101500	192.168.0.17	128.30.52.100	TCP	54	53121 → 443 [ACK] Seq=115 Ack=35540 Win=65320 Len=0
55	2.101818	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=35540 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
56	2.101818	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=36960 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
57	2.101818	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=38380 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]
58	2.101897	192.168.0.17	128.30.52.100	TCP	54	53121 → 443 [ACK] Seq=115 Ack=39800 Win=65320 Len=0
59	2.102916	128.30.52.100	192.168.0.17	TCP	1474	443 → 53121 [ACK] Seq=39800 Ack=115 Win=135 Len=1420 [TCP segment of a reassembled PDU]

4. Logowanie poprzez protokół HTTP

Korzystając z http://login.petrolweb.net, można sprawdzić jak wygląda logowanie w protokole HTTP.

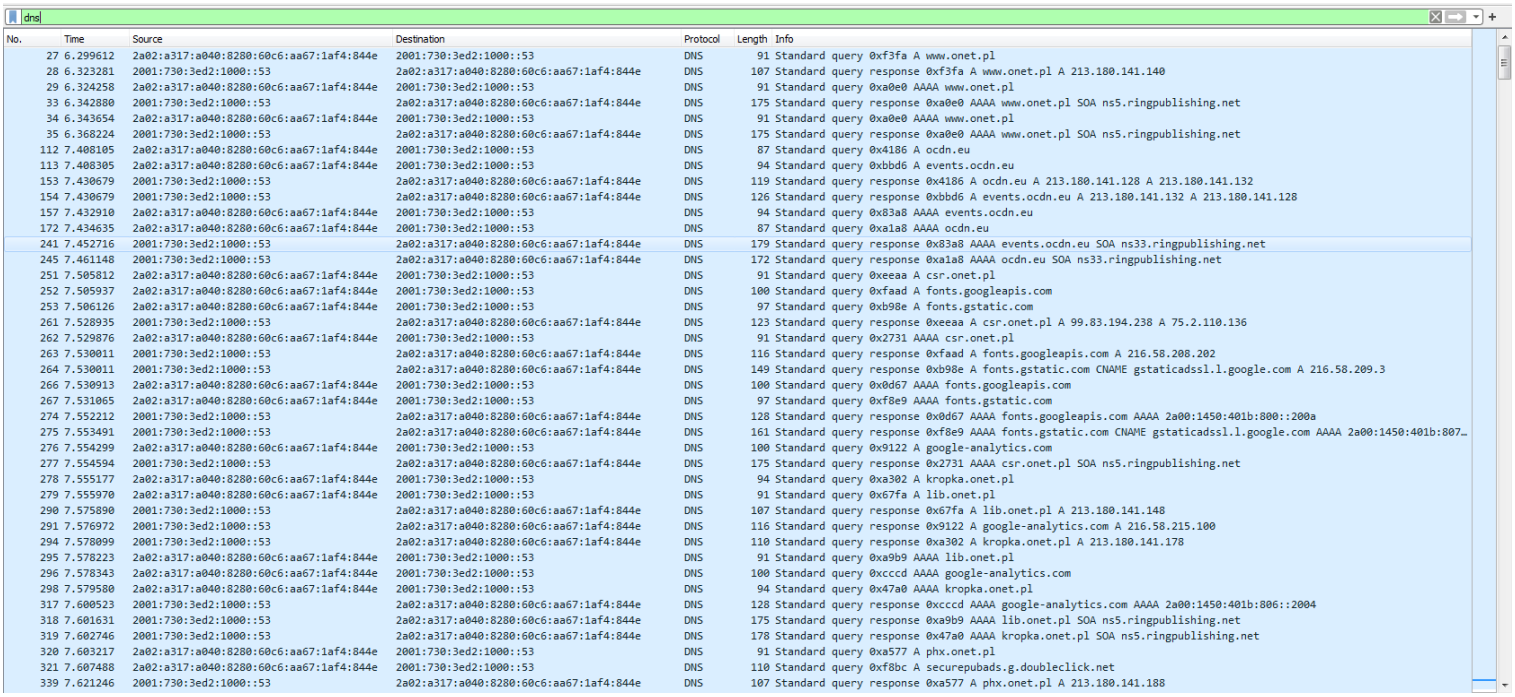


Po wprowadzeniu danych, w Wiresharku można zobaczyć, że podany login i hasło nie są zaszyfrowane i tym samym bardzo łatwo można je zdobyć.



5. Zapytania DNS podczas wywoływania strony

Wchodząc przykładowo na onet.pl, można w Wiresharku zauważyć nie tylko zapytania dla serwera DNS dotyczące samej witryny, ale też podstron serwisu, jak i narzędzi reklamowych (tutaj chociażby Google Analitics).



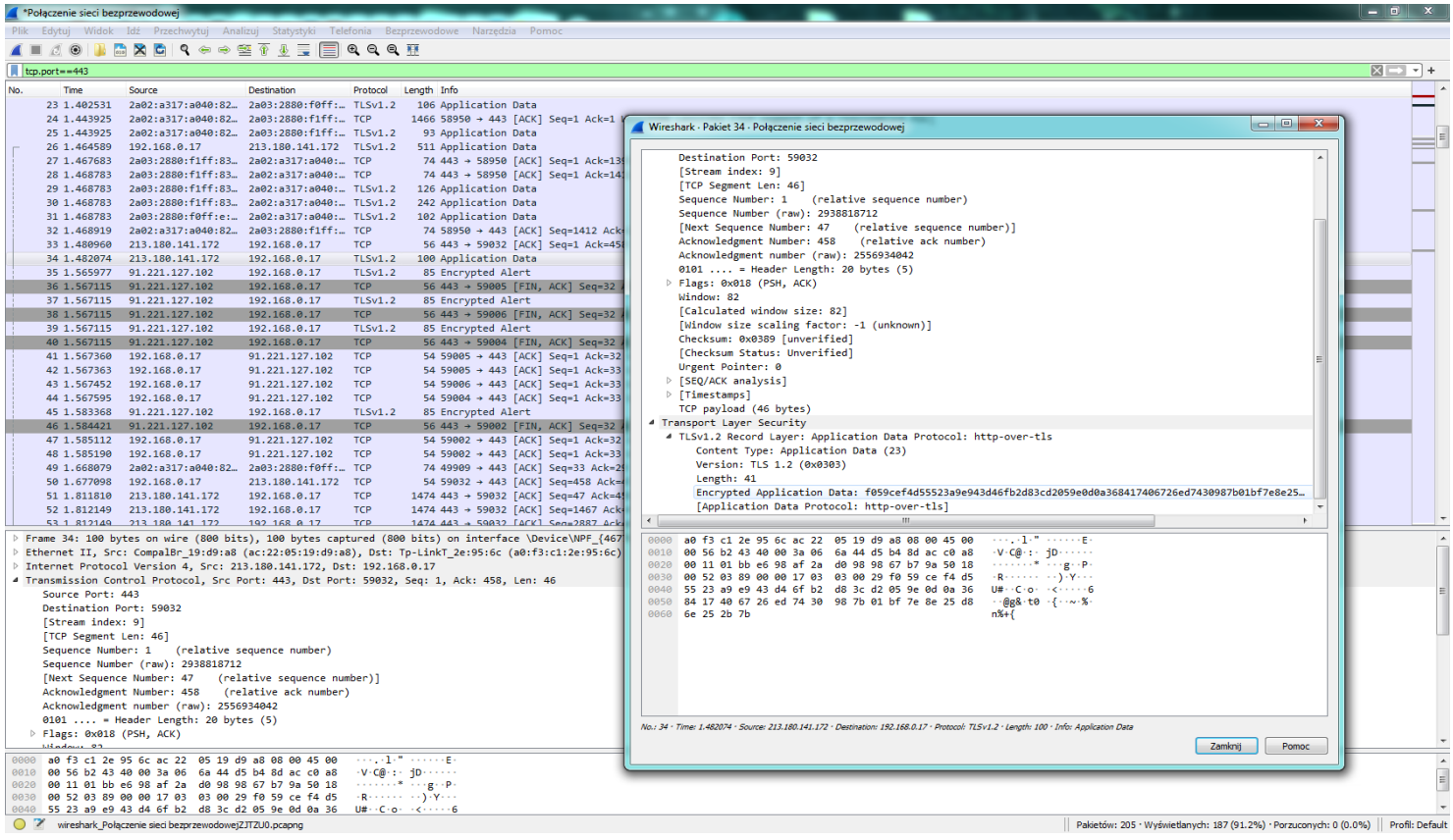
6. Analiza ruchu FTP

ftp

No.	Time	Source	Destination	Protocol	Length	Info
68	9.592472	195.144.107.198	192.168.0.17	FTP	81	Response: 220 Microsoft FTP Service
69	9.592785	192.168.0.17	195.144.107.198	FTP	70	Request: USER anonymous
70	9.637948	195.144.107.198	192.168.0.17	FTP	92	Response: 331 Password required for anonymous.
71	9.638354	192.168.0.17	195.144.107.198	FTP	79	Request: PASS chrome@example.com
72	9.684242	195.144.107.198	192.168.0.17	FTP	79	Response: 530 User cannot log in.
73	9.684530	192.168.0.17	195.144.107.198	FTP	60	Request: QUIT
74	9.734714	195.144.107.198	192.168.0.17	FTP	68	Response: 221 Goodbye.
82	9.825235	195.144.107.198	192.168.0.17	FTP	81	Response: 220 Microsoft FTP Service
83	9.825532	192.168.0.17	195.144.107.198	FTP	65	Request: USER ewan
84	9.872585	195.144.107.198	192.168.0.17	FTP	87	Response: 331 Password required for ewan.
85	9.872927	192.168.0.17	195.144.107.198	FTP	71	Request: PASS qwerty1234
86	9.921914	195.144.107.198	192.168.0.17	FTP	79	Response: 530 User cannot log in.
87	9.922265	192.168.0.17	195.144.107.198	FTP	60	Request: QUIT
88	9.967404	195.144.107.198	192.168.0.17	FTP	68	Response: 221 Goodbye.

Podobnie jak w przypadku logowania na stronie korzystające z protokołu HTTP, login i hasło nie zostały zaszyfrowane.

7. Logowanie poprzez protokół HTTPS



W protokole HTTPS nie jesteśmy w stanie podejrzeć przesyłanych informacji, dane są szyfrowane za pomocą protokołu TLS. Można wyświetlić ilość przesyłanych danych na porcie 443 etc., jednak bez klucza prywatnego nie jesteśmy w stanie stwierdzić, co zawierają.
Klucz publiczny jest udostępniany szyfrującemu dane, jednak klucz prywatny jest dostępny wyłącznie dla wysyłającego informacje i tylko on może je odczytać.