

Esercitazione 7 - W3D4

Regole di firewall e cattura pacchetti con Wireshark

Fabio Benevento - 11/11/2023

Scopo

L'esercitazione mira a consolidare le conoscenze acquisite.

In particolare si compone di due esercizi:

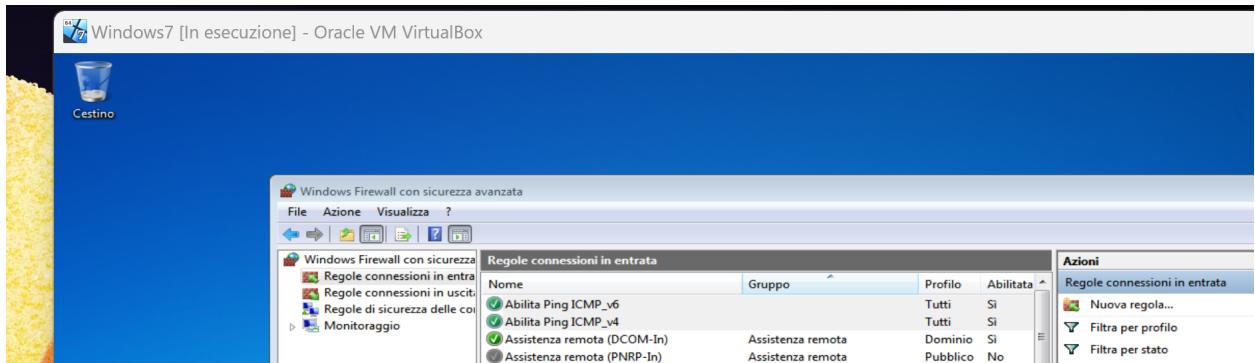
- I) la configurazione di una policy sul firewall windows;
- II) una packet capture con Wireshark.

Implementazione

Esercizio 1

Configurare policy per permettere il ping da macchine Linux a Macchina Windows 7 nel nostro laboratorio (Windows firewall)

Per impostare le regole di firewall per permettere il ping su Windows 7 sono acceduto alla configurazione avanzata di Windows Firewall e ho impostato due regole di tipo ALLOW per quanto riguarda i protocolli ICMP_v4 e ICMP_v6 su cui si basa il ping come illustrato in figura



Abilitazione ping tramite regole di Windows Firewall in Windows 7

Ho quindi verificato il corretto funzionamento del comando ping tra la macchina Kali sulla stessa rete (192.168.50.100) e la macchina Windows 7 (192.168.50.102)

```
(kali㉿kali)-[~]
$ ping 192.168.50.102 -c 5
PING 192.168.50.102 (192.168.50.102) 56(84) bytes of data.
64 bytes from 192.168.50.102: icmp_seq=1 ttl=128 time=6.90 ms
64 bytes from 192.168.50.102: icmp_seq=2 ttl=128 time=1.53 ms
64 bytes from 192.168.50.102: icmp_seq=3 ttl=128 time=2.17 ms
64 bytes from 192.168.50.102: icmp_seq=4 ttl=128 time=4.70 ms
64 bytes from 192.168.50.102: icmp_seq=5 ttl=128 time=4.44 ms

--- 192.168.50.102 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 1.526/3.947/6.895/1.924 ms
```

Test ping da Kali a Windows 7

Esercizio 2

Utilizzo dell'utility InetSim per l'emulazione di servizi Internet

Per utilizzare InetSim ho provveduto alla sua configurazione tramite il file presente sotto il percorso /etc/inetsim/inetsim.conf.

Nello specifico ho abilito i soli servizi http e https, commentando gli altri per disabilitarli

```

#####
# start_service [application] Data
#   68 34734 -> 443 [ACK] Seq=1155 Ack=1677 Win=65536 Len=0
# The services to start
#   68 34734 -> 443 [ACK] Seq=1155 Ack=1932 Win=65408 Len=0
# Syntax: start_service <service name>
#   68 34734 -> 443 [ACK] Seq=1155 Ack=2105 Win=65280 Len=0
# Default: none
#   348 Application Data
# Available service names are:] Seq=1155 Ack=2385 Win=65024 Len=0
# dns, http, smtp, pop3, tftp, ftp, ntp, time_tcp,
# time_udp, daytime_tcp, daytime_udp, echo_tcp, echo_udp, discard_tcp, discard_udp, quotd_tcp,
# quotd_udp, chargen_tcp, chargen_udp, finger,
# ident, syslog, dummy_tcp, dummy_udp, smtps, pop3s,
# ftps, irc, https
#   76 80 -> 40608 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495
#   76 80 -> 40608 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495
#start_service dns
start_service http
start_service https
#start_service smtp
#start_service smtps Len=431
#start_service pop3
#start_service pop3s
#start_service ftp
#start_service fann

```

e ho effettuato il bind per l'ascolto con l'indirizzo di loopback 127.0.0.1 (di default sarebbe comunque questo)

```

#####
# service_bind_address [application] Data
#   68 34734 -> 443 [ACK] Seq=1155 Ack=2105 Win=0 Len=0
# IP address to bind services to
#   68 34734 -> 443 [ACK] Seq=1155 Ack=2385 Win=0 Len=0
# Syntax: service_bind_address <IP address>
#   68 34734 -> 443 [FIN, ACK] Seq=1179 Ack=2105 Win=0 Len=0
# Default: 127.0.0.1
#   76 80 -> 40608 [SYN] Seq=0 Win=65495 Len=0 MSS=65495
#   76 80 -> 40608 [SYN] Seq=0 Win=65495 Len=0 MSS=65495
service_bind_address 127.0.0.1

```

Ho avviato quindi la simulazione con il comando `sudo inetsim`. I due servizi http e https risultano quindi attivi ed in ascolto sulla porte 80 e 443 rispettivamente come mostrato in figura.

```

(kali㉿kali)-[~] ~]$ sudo inetsim
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Using log directory: /var/log/inetsim/
Using data directory: /var/lib/inetsim/ SACK
Using report directory: /var/log/inetsim/report/ Ack=1 Win=65483 Len=0 MSS=65495
Using configuration file: /etc/inetsim/inetsim.conf
Parsing configuration file.
Configuration file parsed successfully.
== INetSim main process started (PID 27145) ==
Session ID: 27145
Listening on: 127.0.0.1
Real Date/Time: 2023-11-10 14:30:47
Fake Date/Time: 2023-11-10 14:30:47 (Delta: 0 seconds)
Forking services ...
* http_80_tcp - started (PID 27155)
* https_443_tcp - started (PID 27156)
done.
Simulation running.

```

Cattura di pacchetto con Wireshark

Il server HTTP/HTTPS di InetSim mette a disposizione una serie di risorse di esempio fruibili tramite il server stesso, come mostrato nella immagine seguente relativamente a HTTPS (quelle HTTP sono analoghe).

Per esercitazione ho provato a richiedere tramite browser la risorsa sample.txt utilizzando i protocolli http e https e successivamente la risorsa sample.html (risorsa di default) sempre in entrambi i casi, catturando il transito dei pacchetti mediante Wireshark.

```
#####
# https_fakefile 574 127.0.0.1          127.0.0.1    TCP      68 34
# 5 0.001944357 127.0.0.1          127.0.0.1    TLSv1.3   689 0
# Fake files returned in fake mode based on the file extension      127.0.0.1    TCP      68 44
# in the HTTPS request. 127.0.0.1          127.0.0.1    TLSv1.3   1489 Se
# The fake files must be placed in <data-dir>/http/fakefiles      127.0.0.1    TCP      68 34
# 0.0.04350384 127.0.0.1          127.0.0.1    TLSv1.3   148 Ch
# Syntax: https_fakefile <extension> <filename> <mime-type>      127.0.0.1    TLSv1.3   521 Ad
# 10.0.051447367 127.0.0.1          127.0.0.1    TLSv1.3   323 Ap
# Default: none 3761 127.0.0.1          127.0.0.1    TCP      68 34
# 12.0.099389213 127.0.0.1          127.0.0.1    TLSv1.3   323 Ap
https_fakefile txt sample.txt      text/plain 127.0.0.1    TCP      68 34
https_fakefile htm sample.html     text/html   127.0.0.1    TLSv1.3   241 Ap
https_fakefile html sample.html    text/html   127.0.0.1    TCP      68 34
https_fakefile php sample.html     text/html   127.0.0.1    TLSv1.3   348 Ap
https_fakefile gif sample.gif     image/gif   127.0.0.1    TLSv1.3   68 34
https_fakefile jpg sample.jpg     image/jpeg  127.0.0.1    TCP      68 34
https_fakefile jpeg sample.jpg    image/jpeg  127.0.0.1    TLSv1.3   92 Ap
https_fakefile png sample.png    image/png   127.0.0.1    TCP      68 34
https_fakefile bmp sample.bmp    image/x-ms-bmp 127.0.0.1    TLSv1.3   92 Ap
https_fakefile ico favicon.ico   image/x-icon 127.0.0.1    TCP      56 34
https_fakefile exe sample_gui.exe x-msdos-program 127.0.0.1    TCP      76 46
https_fakefile com sample_gui.exe x-msdos-program 127.0.0.1    TCP      76 86
Frame 25: 499 bytes on wire (3992 bits), 499 bytes captured (3992 bits)
#####
# https_default_fakefile
# 127.0.0.1 127.0.0.1 Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
# Transmission Control Protocol Version 4, Src Port: 40000, Dst Port: 80, Seq: 1, Ack: 1
# The default fake file returned in fake mode if the file extension
# in the HTTPS request does not match any of the extensions
# defined above.
# [Stream index: 1]
# The default fake file must be placed in <data-dir>/http/fakefiles
# [TCP Segment Len: 431]
# Syntax: https_default_fakefile <filename> <mime-type>
# Sequence Number (raw): 391379247
# Default: none
# [Next Sequence Number: 432 (relative sequence number)]
# Acknowledgment Number (raw): 2383623590
# Acknowledgment number (raw): 2383623590
https_default_fakefile sample.html text/html
```

Test 1: richiesta HTTP della risorsa sample.txt

L'immagine seguente mostra la cattura dei pacchetti mediante Wireshark in seguito alla richiesta della risorsa `sample.txt` su protocollo HTTP

1	0.0000000000 127.0.0.1	127.0.0.1	TCP	76 37890 - 80 [SYN] Seq=0 Win=65495 MSS=65495 SACK_PERM Tsvl=3012438604 Tscr=0 WS=128
2	0.000158504 127.0.0.1	127.0.0.1	TCP	76 80 - 37890 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM Tsvl=3012438604 Tscr=3012438604 WS=128
3	0.000176441 127.0.0.1	127.0.0.1	TCP	68 37890 - 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0 Tsvl=3012438604 Tscr=3012438604
4	0.138329539 127.0.0.1	127.0.0.1	HTTP	509 GET /sample.txt HTTP/1.1
5	0.138352121 127.0.0.1	127.0.0.1	TCP	68 80 - 37890 [ACK] Seq=1 Ack=442 Win=65152 Len=0 Tsvl=3012438742 Tscr=3012438742
6	0.249401167 127.0.0.1	127.0.0.1	TCP	218 80 - 37890 [PSH, ACK] Seq=1 Ack=442 Win=65536 Len=150 Tsvl=3012438853 Tscr=3012438742 [TCP segment of a reassembly]
7	0.249428790 127.0.0.1	127.0.0.1	TCP	68 37890 - 80 [ACK] Seq=442 Ack=151 Win=65408 Len=0 Tsvl=3012438853 Tscr=3012438853
8	0.250200066 127.0.0.1	127.0.0.1	HTTP	165 HTTP/1.1 200 OK (text/plain)
9	0.250212293 127.0.0.1	127.0.0.1	TCP	68 37890 - 80 [ACK] Seq=442 Ack=248 Win=65408 Len=0 Tsvl=3012438854 Tscr=3012438854
10	0.250374741 127.0.0.1	127.0.0.1	TCP	68 37890 - 80 [FIN, ACK] Seq=442 Ack=248 Win=65536 Len=0 Tsvl=3012438854 Tscr=3012438854
11	0.253775342 127.0.0.1	127.0.0.1	TCP	68 80 - 37890 [FIN, ACK] Seq=248 Ack=443 Win=65536 Len=0 Tsvl=3012438858 Tscr=3012438858
12	0.253801335 127.0.0.1	127.0.0.1	TCP	68 37890 - 80 [ACK] Seq=443 Ack=249 Win=65536 Len=0 Tsvl=3012438858 Tscr=3012438858

Nell'immagine è possibile individuare innanzitutto l'handshake a tre vie iniziale per stabilire la connessione (SYN, SYN-ACK, ACK, righe 1-3), dopodichè la richiesta HTTP GET della risorsa `sample.txt` (riga 10) con la relativa risposta 200 OK che indica il corretto trasferimento della risorsa da parte del server (riga 14).

Le righe successive (righe 16-18) costituiscono lo scambio di pacchetti TCP per la chiusura della connessione instaurata tra client e server (three-way handshake: FIN-ACK, FIN-ACK, ACK). Nello specifico il client (il browser) richiede la chiusura della connessione TCP aperta con l'host destinatario (il server HTTP) inviando un messaggio di FIN-ACK (riga 16) e rimane in attesa della conferma dal ricevente (con un analogo pacchetto FIN-ACK). A questo punto la connessione è ritenuta chiusa per metà: l'host che ha inviato FIN non potrà più inviare dati, mentre l'altro host ha il canale di comunicazione ancora disponibile. Quando anche l'altro host invierà il pacchetto con FIN impostato, la connessione, dopo il relativo FIN-ACK, sarà considerata completamente chiusa (riga 17). La riga 18 è la conferma (ACK) della chiusura da parte del client con la quale termina la richiesta.

Test 2: richiesta HTTPS della risorsa sample.txt

L'immagine seguente mostra la cattura dei pacchetti mediante Wireshark in seguito alla richiesta della risorsa `sample.txt` su protocollo HTTPS

1 0.0000000000 127.0.0.1	127.0.0.1	TCP	76 45514 - 443 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM Tsvl=3009227416 TSecr=0 WS=128
2 0.000038073 127.0.0.1	127.0.0.1	TCP	76 443 - 45514 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM Tsvl=3009227416 TSecr=3009227416 WS=128
3 0.000059530 127.0.0.1	127.0.0.1	TCP	68 45514 - 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0 Tsvl=3009227416 TSecr=3009227416
4 0.002859774 127.0.0.1	127.0.0.1	TLSv1.3	689 Client Hello
5 0.002869681 127.0.0.1	127.0.0.1	TCP	68 443 - 45514 [ACK] Seq=1 Ack=622 Win=64896 Len=0 Tsvl=3009227419 TSecr=3009227419
6 0.069812376 127.0.0.1	127.0.0.1	TLSv1.3	1489 Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data, Application Data
7 0.069845148 127.0.0.1	127.0.0.1	TCP	68 45514 - 443 [ACK] Seq=622 Ack=1422 Win=64384 Len=0 Tsvl=3009227486 TSecr=3009227486
8 0.073425434 127.0.0.1	127.0.0.1	TLSv1.3	148 Change Cipher Spec, Application Data
9 0.073750355 127.0.0.1	127.0.0.1	TLSv1.3	521 Application Data
10 0.074624146 127.0.0.1	127.0.0.1	TLSv1.3	323 Application Data
11 0.074834975 127.0.0.1	127.0.0.1	TCP	68 45514 - 443 [ACK] Seq=1155 Ack=1677 Win=65408 Len=0 Tsvl=3009227490 TSecr=3009227490
12 0.074671442 127.0.0.1	127.0.0.1	TLSv1.3	323 Application Data
13 0.074673697 127.0.0.1	127.0.0.1	TCP	68 45514 - 443 [ACK] Seq=1155 Ack=1932 Win=65280 Len=0 Tsvl=3009227490 TSecr=3009227490
14 0.162135959 127.0.0.1	127.0.0.1	TLSv1.3	241 Application Data
15 0.162173246 127.0.0.1	127.0.0.1	TCP	68 45514 - 443 [ACK] Seq=1155 Ack=2105 Win=65152 Len=0 Tsvl=3009227518 TSecr=3009227518
16 0.162764882 127.0.0.1	127.0.0.1	TLSv1.3	348 Application Data
17 0.162719422 127.0.0.1	127.0.0.1	TCP	68 45514 - 443 [ACK] Seq=1155 Ack=2385 Win=65024 Len=0 Tsvl=3009227519 TSecr=3009227519
18 0.104601957 127.0.0.1	127.0.0.1	TLSv1.3	92 Application Data
19 0.104621935 127.0.0.1	127.0.0.1	TCP	68 45514 - 443 [FIN, ACK] Seq=1179 Ack=2385 Win=65536 Len=0 Tsvl=3009227520 TSecr=3009227519
20 0.110750073 127.0.0.1	127.0.0.1	TLSv1.3	92 Application Data

In questo caso la comunicazione è criptata tramite protocollo TSL utilizzato da HTTPS. E' possibile comunque individuare il three-way handshake iniziale per stabilire la connessione (SYN, SYN-ACK, ACK, righe 1-3) e la richiesta di chiusura da parte del client (FIN-ACK, riga 19, l'handshake completo non è visibile in questo caso in quanto la connessione è per l'appunto criptata)

Test 3: richiesta HTTP della risorsa sample.html

L'immagine seguente mostra la cattura dei pacchetti mediante Wireshark in seguito alla richiesta della risorsa sample.html su protocollo HTTP

1 0.0000000000 127.0.0.1	127.0.0.1	TCP	76 37924 - 88 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM Tsvl=3010496022 TSecr=0 WS=128
2 0.000021865 127.0.0.1	127.0.0.1	TCP	76 88 - 37924 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM Tsvl=3010496022 TSecr=3010496022 WS=128
3 0.000038878 127.0.0.1	127.0.0.1	TCP	68 37924 - 88 [ACK] Seq=1 Ack=1 Win=65536 Len=0 Tsvl=3010496022 TSecr=3010496022
4 0.0339349861 127.0.0.1	127.0.0.1	HTTP	518 GET /sample.html HTTP/1.1
5 0.0339349868 127.0.0.1	127.0.0.1	TCP	68 37924 - 88 [ACK] Seq=1 Ack=451 Win=65152 Len=0 Tsvl=3010496061 TSecr=3010496061
6 0.146281399 127.0.0.1	127.0.0.1	TCP	218 88 - 37924 [PSH, ACK] Seq=1 Ack=451 Win=65536 Len=159 Tsvl=3010496168 TSecr=3010496061 [TCP segment of a reassembled PDU]
7 0.146585277 127.0.0.1	127.0.0.1	TCP	68 37924 - 88 [ACK] Seq=451 Ack=151 Win=65408 Len=0 Tsvl=3010496168 TSecr=3010496168
8 0.146756291 127.0.0.1	127.0.0.1	HTTP	326 HTTP/1.1 200 OK (text/html)
9 0.146756443 127.0.0.1	127.0.0.1	TCP	68 37924 - 88 [ACK] Seq=451 Ack=409 Win=65152 Len=0 Tsvl=3010496169 TSecr=3010496169
10 0.147111238 127.0.0.1	127.0.0.1	TCP	68 37924 - 88 [FIN, ACK] Seq=451 Ack=409 Win=65536 Len=0 Tsvl=3010496169 TSecr=3010496169
11 0.148847312 127.0.0.1	127.0.0.1	TCP	68 88 - 37924 [FIN, ACK] Seq=449 Ack=452 Win=65536 Len=0 Tsvl=3010496171 TSecr=3010496169
12 0.148870898 127.0.0.1	127.0.0.1	TCP	68 37924 - 88 [ACK] Seq=452 Ack=410 Win=65536 Len=0 Tsvl=3010496171 TSecr=3010496171
13 0.1499863387 127.0.0.1	127.0.0.1	TCP	76 37924 - 88 [SYN] Seq=0 Win=65495 MSS=65495 SACK_PERM Tsvl=3010496602 TSecr=0 WS=128
14 0.1500007963 127.0.0.1	127.0.0.1	TCP	76 88 - 37924 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM Tsvl=3010496602 TSecr=3010496602 WS=128
15 0.578098133 127.0.0.1	127.0.0.1	TCP	68 37924 - 88 [ACK] Seq=1 Ack=1 Win=65536 Len=0 Tsvl=3010496602 TSecr=3010496602
16 0.580722698 127.0.0.1	127.0.0.1	HTTP	459 GET /favicon.ico HTTP/1.1
17 0.5807308654 127.0.0.1	127.0.0.1	TCP	68 88 - 37928 [ACK] Seq=1 Ack=383 Win=65152 Len=0 Tsvl=3010496603 TSecr=3010496603
18 0.6499859794 127.0.0.1	127.0.0.1	TCP	221 88 - 37928 [PSH, ACK] Seq=1 Ack=383 Win=65536 Len=153 Tsvl=3010496672 TSecr=3010496603 [TCP segment of a reassembled PDU]
19 0.649983844 127.0.0.1	127.0.0.1	TCP	68 37928 - 88 [ACK] Seq=383 Ack=184 Win=65408 Len=0 Tsvl=3010496672 TSecr=3010496672
20 0.650465559 127.0.0.1	127.0.0.1	HTTP	266 HTTP/1.1 200 OK (image/x-icon)
21 0.6509470794 127.0.0.1	127.0.0.1	TCP	68 37928 - 88 [ACK] Seq=383 Ack=352 Win=65280 Len=0 Tsvl=3010496672 TSecr=3010496672
22 0.6509591118 127.0.0.1	127.0.0.1	TCP	68 37928 - 88 [FIN, ACK] Seq=383 Ack=352 Win=65536 Len=0 Tsvl=3010496672 TSecr=3010496672
23 0.6558757572 127.0.0.1	127.0.0.1	TCP	68 88 - 37928 [FIN, ACK] Seq=352 Ack=384 Win=65536 Len=0 Tsvl=3010496678 TSecr=3010496678
24 0.655902792 127.0.0.1	127.0.0.1	TCP	68 37928 - 88 [ACK] Seq=384 Ack=353 Win=65536 Len=0 Tsvl=3010496678 TSecr=3010496678

In questo caso lo scambio di pacchetti è del tutto analogo con la richiesta della risorsa sample.txt, con la differenza che in questo caso siamo in presenza di una pagina ipertestuale. Viene quindi effettuata una prima richiesta HTTP GET con la pagina richiesta (sample.html, riga 8) e poi delle risorse contestuali presenti nella pagina (immagini, video, etc.). Nel caso in esame viene richiesta tramite HTTP GET la risorsa favicon.ico (riga 16) che rappresenta una piccola immagine che rappresenta il sito e che viene visualizzata nella barra degli indirizzi del browser accanto all'URL del sito web

e nella scheda del browser. Per ciascuna risorsa è possibile individuare l'apertura della connessione (SYN, SYN-ACK, ACK) e la chiusura (FIN, FIN-ACK, ACK). Si tratta quindi di una connessione non persistente o parallelizzata dal browser.

Test 4: richiesta HTTPS della risorsa sample.html

L'immagine seguente mostra la cattura dei pacchetti mediante Wireshark in seguito alla richiesta della risorsa sample.html su protocollo HTTPS.

Analogamente alla richiesta HTTPS per la risorsa HTTP non è possibile visualizzare con Wireshark l'intero scambio di pacchetti in quanto la connessione criptata. E' comunque possibile individuare la richiesta di apertura di connessione per la prima risorsa (SYN, SYN-ACK, ACK, righe 1-3) e la relativa chiusura (FIN-ACK, riga 34), così come la richiesta di apertura di connessione per la seconda risorsa (SYN, SYN-ACK, ACK, righe 6-8) e la corrispondente chiusura (FIN-ACK, riga 46).

1 0.000000000 127.0.0.1	127.0.0.1	TCP	76 44394 - 443 [SYN] Seq=0 Win=65495 Len=8 MSS=65495 SACK_PERM TStamp=3012043187 TSectr=0 WS=128
2 0.0000001315 127.0.0.1	127.0.0.1	TCP	76 443 - 44394 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=8 MSS=65495 SACK_PERM TStamp=3012043187 TSectr=3012043187 WS=128
3 0.0000038006 127.0.0.1	127.0.0.1	TCP	68 44394 - 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TStamp=3012043187 TSectr=3012043187
4 0.0029536323 127.0.0.1	127.0.0.1	TLSv1	585 Client Hello
5 0.002962445 127.0.0.1	127.0.0.1	TCP	68 443 - 44394 [ACK] Seq=1 Ack=518 Win=65624 Len=8 TStamp=3012043199 TSectr=3012043199
6 0.002962445 127.0.0.1	127.0.0.1	TLSv1	76 44398 - 44398 [SYN] Seq=0 Win=65495 Len=8 MSS=65495 SACK_PERM TStamp=3012043232 TSectr=0 WS=128
7 0.002962445 127.0.0.1	127.0.0.1	TCP	76 44398 - 44398 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=8 MSS=65495 SACK_PERM TStamp=3012043237 TSectr=3012043237 WS=128
8 0.0059881078 127.0.0.1	127.0.0.1	TCP	68 44398 - 44398 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TStamp=3012043237 TSectr=3012043237
9 0.005634420 127.0.0.1	127.0.0.1	TLSv1	585 Client Hello
10 0.005654667 127.0.0.1	127.0.0.1	TCP	68 443 - 44398 [ACK] Seq=1 Ack=518 Win=65624 Len=8 TStamp=3012043282 TSectr=3012043282
11 0.0056623796 127.0.0.1	127.0.0.1	TLSv1.3	1489 Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data, Application Data
12 0.0056623796 127.0.0.1	127.0.0.1	TCP	68 44398 - 443 [ACK] Seq=518 Ack=1422 Win=64384 Len=0 TStamp=3012043753 TSectr=3012043753
13 0.0056623796 127.0.0.1	127.0.0.1	TLSv1.3	148 Change Cipher Spec, Application Data
14 0.0056623796 127.0.0.1	127.0.0.1	TCP	68 443 - 44398 [ACK] Seq=1422 Ack=598 Win=65536 Len=0 TStamp=3012043779 TSectr=3012043779
15 0.0056623796 127.0.0.1	127.0.0.1	TLSv1.3	540 Application Data
16 0.0056623796 127.0.0.1	127.0.0.1	TCP	68 443 - 44398 [ACK] Seq=1422 Ack=1070 Win=65152 Len=0 TStamp=3012043780 TSectr=3012043780
17 0.0056623796 127.0.0.1	127.0.0.1	TLSv1.3	323 Application Data
18 0.0056623796 127.0.0.1	127.0.0.1	TLSv1.3	1489 Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data, Application Data
19 0.0056623796 127.0.0.1	127.0.0.1	TCP	68 44394 - 44384 [ACK] Seq=518 Ack=1422 Win=64384 Len=0 TStamp=3012043785 TSectr=3012043785
20 0.0056623796 127.0.0.1	127.0.0.1	TLSv1.3	148 Change Cipher Spec, Application Data
21 0.0056623796 127.0.0.1	127.0.0.1	TCP	68 443 - 44398 [ACK] Seq=1422 Ack=598 Win=65536 Len=0 TStamp=3012043883 TSectr=3012043883
22 0.0056623796 127.0.0.1	127.0.0.1	TLSv1.3	323 Application Data
23 0.0056623796 127.0.0.1	127.0.0.1	TCP	68 44398 - 44398 [ACK] Seq=1070 Ack=1677 Win=65536 Len=0 TStamp=3012043824 TSectr=3012043824
24 0.0056623796 127.0.0.1	127.0.0.1	TLSv1.3	323 Application Data
25 0.0056623796 127.0.0.1	127.0.0.1	TCP	68 44398 - 44398 [ACK] Seq=1070 Ack=1932 Win=65408 Len=0 TStamp=3012043824 TSectr=3012043824
26 0.0056623796 127.0.0.1	127.0.0.1	TCP	68 44394 - 44394 [ACK] Seq=598 Ack=1677 Win=65536 Len=0 TStamp=3012043848 TSectr=3012043848
27 0.0056623796 127.0.0.1	127.0.0.1	TLSv1.3	323 Application Data
28 0.0056623796 127.0.0.1	127.0.0.1	TCP	68 44394 - 44394 [ACK] Seq=598 Ack=1932 Win=65408 Len=0 TStamp=3012043848 TSectr=3012043848
29 0.0056623796 127.0.0.1	127.0.0.1	TLSv1.3	241 Application Data
30 0.0056623796 127.0.0.1	127.0.0.1	TCP	68 44398 - 44398 [ACK] Seq=1070 Ack=2105 Win=65280 Len=0 TStamp=3012044038 TSectr=3012044038
31 0.0056623796 127.0.0.1	127.0.0.1	TLSv1.3	348 Application Data
32 0.0056623796 127.0.0.1	127.0.0.1	TCP	68 44398 - 44398 [ACK] Seq=1070 Ack=2385 Win=65024 Len=0 TStamp=3012044039 TSectr=3012044039
33 0.0056623796 127.0.0.1	127.0.0.1	TLSv1.3	92 Application Data
34 0.0056623796 127.0.0.1	127.0.0.1	TCP	68 44398 - 44398 [FIN, ACK] Seq=1094 Ack=2385 Win=65536 Len=0 TStamp=3012044039 TSectr=3012044039
35 0.0056623796 127.0.0.1	127.0.0.1	TCP	68 [TCP Retransmission] 44398 - 443 [FIN, ACK] Seq=1094 Ack=2385 Win=65536 Len=0 TStamp=3012044052 TSectr=3012044052
36 0.0056623796 127.0.0.1	127.0.0.1	TCP	68 44398 - 44398 [ACK] Seq=2385 Ack=1095 Win=65536 Len=0 TStamp=3012044052 TSectr=3012044052 SLE=1095
37 0.0056623796 127.0.0.1	127.0.0.1	TLSv1.3	92 Application Data
38 0.0056623796 127.0.0.1	127.0.0.1	TCP	56 44398 - 443 [RST] Seq=1095 Win=0 Len=0
39 1.0496743971 127.0.0.1	127.0.0.1	TLSv1.3	473 Application Data
40 1.0496743971 127.0.0.1	127.0.0.1	TCP	68 44398 - 44398 [ACK] Seq=1932 Ack=1093 Win=65536 Len=0 TStamp=3012044724 TSectr=3012044683
41 1.0496743971 127.0.0.1	127.0.0.1	TLSv1.3	244 Application Data
42 1.722678907 127.0.0.1	127.0.0.1	TCP	68 44394 - 44394 [ACK] Seq=1003 Ack=2108 Win=65408 Len=0 TStamp=3012044909 TSectr=3012044909
43 1.7382056540 127.0.0.1	127.0.0.1	TLSv1.3	288 Application Data
44 1.738227757 127.0.0.1	127.0.0.1	TCP	68 44394 - 44394 [ACK] Seq=1003 Ack=2328 Win=65280 Len=0 TStamp=3012044917 TSectr=3012044917
45 1.738598927 127.0.0.1	127.0.0.1	TLSv1.3	92 Application Data
46 1.739528396 127.0.0.1	127.0.0.1	TCP	68 44394 - 44394 [FIN, ACK] Seq=1027 Ack=2328 Win=65536 Len=0 TStamp=3012044917 TSectr=3012044917

Nota: per esser sicuro di vedere l'intero scambio di pacchetti in caso di risorsa html ho eseguito la richiesta tramite browser in modalità privata in maniera da escludere che a seguito di precedenti richieste della risorsa stessa, alcuni elementi venissero presi dalla cache (in alternativa si sarebbe potuto cancellare la cache in suguito ad ogni richiesta tramite browser)