

Progetto fine Modulo 1

Rebecca Caldarella

Requisiti e servizi:

- Kali Linux IP 192.168.32.100
- Windows 7 IP 192.168.32.101
- HTTPS server: attivo
- Servizio DNS per risoluzione nomi di dominio: attivo

Traccia:

Simulare, in ambiente di laboratorio virtuale, un'architettura client server in cui un client con indirizzo 192.168.32.101 (Windows 7) richiede tramite Web browser una risorsa all'hostname `epicode.internal` che risponde all'indirizzo 192.168.32.100 (Kali).

Si intercetti poi la comunicazione con Wireshark, evidenziando i MAC address di sorgente e destinazione ed il contenuto della richiesta HTTPS.

Ripetere l'esercizio, sostituendo il server HTTPS con un server HTTP. Si intercetti nuovamente il traffico, evidenziando le eventuali differenze tra il traffico appena catturato in HTTP ed il traffico precedente in HTTPS.

Spiegare, motivandole, le principali differenze se presenti.

Svolgimento

I. Configurazione IP Kali Linux e Windows 7

L'esercitazione prevede l'utilizzo delle macchine virtuali Kali Linux e Windows 7. Per fare in modo che le macchine comunichino tra loro il primo passo è configurare gli indirizzi IP dei due ambienti:

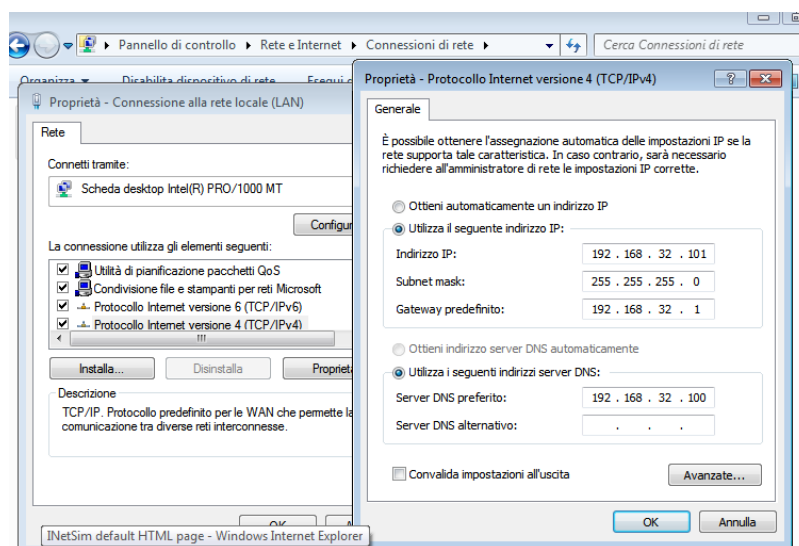
Kali Linux con IP 192.168.32.100

```
File Actions Edit View Help
GNU nano 7.2
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback
iface eth0 inet static
address 192.168.32.100/24
gateway 192.168.50.1
```

E Windows 7 con IP 192.168.32.101



Per quanto riguarda il Windows 7 è stato necessario anche configurare il servizio DNS, in quanto sarà questa la macchina che andrà a richiedere al browser l'hostname epicode.internal. Da traccia, infatti, l'IP del server DNS deve corrispondere a quello della macchina Kali Linux, pertanto è stato inserito l'IP 192.168.32.100 nella voce "server DNS preferito".

Per verificare che la configurazione sia avvenuta con successo basta aprire il prompt dei comandi e lanciare il comando "ipconfig all":

```

C:\Windows\system32\cmd.exe
Routing IP abilitato. . . . . : No
Proxy WINS abilitato . . . . . : No

Scheda Ethernet Connessione alla rete locale (LAN):
Suffisso DNS specifico per connessione:
Descrizione . . . . . : Scheda desktop Intel(R) PRO/1000 MT
Indirizzo fisico . . . . . : 08-00-27-62-50-67
DHCP abilitato . . . . . : No
Configurazione automatica abilitata . . : Sì
Indirizzo IPv6 locale rispetto al collegamento . : fe80::d04f:60f9:23e6:48ca%11(Preferenziale)
Indirizzo IPv4 . . . . . : 192.168.32.101(Preferenziale)
Subnet mask . . . . . : 255.255.255.0
Gateway predefinito . . . . . : 192.168.32.1
IAID DHCPv6 . . . . . : 235405351
DUID Client DHCPv6 . . . . . : 00-01-00-01-2C-F0-3A-CD-08-00-27-62-50-67

Server DNS . . . . . : 192.168.32.100
NetBIOS su TCP/IP . . . . . : Attivato

Scheda Tunnel isatap.{D780D927-7E16-4AD5-B384-1F63D0D4110A}:
Stato supporto. . . . . : Supporto disconnesso
Suffisso DNS specifico per connessione:

```

da qui infatti è possibile vedere che l'indirizzo IPv4 e il Server DNS sono quelli impostati manualmente tramite Pannello di Controllo.

Una volta configurate entrambi gli ambienti virtuali, possiamo assicurarci che comunichino correttamente tramite il comando "Ping":

```

— 192.168.32.101 ping statistics —
133 packets transmitted, 133 received, 0% packet loss, time 133977ms
rtt min/avg/max/mdev = 0.471/4.120/87.683/12.618 ms

```

```

Microsoft Windows [Versione 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. Tutti i diritti riservati.

C:\Users\wboxuser>ping 192.168.32.100

Esecuzione di Ping 192.168.32.100 con 32 byte di dati:
Risposta da 192.168.32.100: byte=32 durata=1ms TTL=64
Risposta da 192.168.32.100: byte=32 durata=1ms TTL=64
Risposta da 192.168.32.100: byte=32 durata=1ms TTL=64
Risposta da 192.168.32.100: byte=32 durata=1ms TTL=64

Statistiche Ping per 192.168.32.100:
Pacchetti trasmessi = 4, Ricevuti = 4,
Perdi = 0 (0% persi),
Tempo approssimativo percorsi andata/ritorno in millisecondi:
Minimo = 1ms, Massimo = 1ms, Medio = 1ms

C:\Users\wboxuser>

```

II. Configurazione INetSim

INetSim è un tool open source che ci permette di simulare i servizi internet standard, per cui seguendo la traccia lo utilizzeremo per configurare i servizi DNS, HTTP e HTTPS.

Per poter effettuare le modifiche che ci interessano bisogna lanciare dal terminale di Kali Linux il comando “sudo nano /etc/inetsim/” e recarci all’interno di “inetsim.conf”. Il comando “sudo” ci fornisce i permessi di super amministratore e il comando “nano” ci permette di aprire il file in modo da poterlo editare.

Da qui possiamo mantenere attivi i servizi che ci interessano, come da figura:

```
# INetSim configuration file
#
#####

#####
# Main configuration
#####

#####
# start_service
#
# The services to start
#
# Syntax: start_service <service name>
#
# Default: none
#
# Available service names are:
# 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
#
start_service dns
start_service http
start_service https
#start_service smtp
#start_service smtps
#start_service pop3
#start_service pop3s
#start_service ftp
#start_service ftps
#start_service tftp
#start_service irc
#start_service ntp
#start_service finger
#start_service ident
#start_service syslog
#start_service time_tcp
#start_service time_udp
#start_service daytime_tcp
#start_service daytime_udp
#start_service echo_tcp
#start_service echo_udp
#start_service discard_tcp
#start_service discard_udp
#start_service quotd_tcp
#start_service quotd_udp
#start_service chargen_tcp
#start_service chargen_udp
#start_service dummy_tcp
#start_service dummy_udp
```

Una volta fatta questa operazione possiamo impostare il server bind address 0.0.0.0 (valido quindi per qualunque interfaccia)

```
#####
# service_bind_address
#
# IP address to bind services to
#
# Syntax: service_bind_address <IP address>
#
# Default: 127.0.0.1
#
service_bind_address 0.0.0.0
```

per poi configurare il DNS di default con l'IP di Kali Linux e il DNS statico che associ all'hostname epicode.internal l'IP 192.168.32.100:

```
#####
# dns_default_ip
#
# Default IP address to return with DNS replies
#
# Syntax: dns_default_ip <IP address>
#
# Default: 127.0.0.1
#
dns_default_ip 192.168.32.100

#####
# dns_default_hostname
#
# Default hostname to return with DNS replies
#
# Syntax: dns_default_hostname <hostname>
#
# Default: www
#
#dns_default_hostname somehost

#####
# dns_default_domainname
#
# Default domain name to return with DNS replies
#
# Syntax: dns_default_domainname <domain name>
#
# Default: inetsim.org
#
#dns_default_domainname some.domain

#####
# dns_static
#
# Static mappings for DNS
#
# Syntax: dns_static <fqdn hostname> <IP address>
#
# Default: none
#
#dns_static www.foo.com 10.10.10.10
#dns_static ns1.foo.com 10.70.50.30
#dns_static ftp.bar.net 10.10.20.30
dns_static epicode.internal 192.168.32.100
#####
```

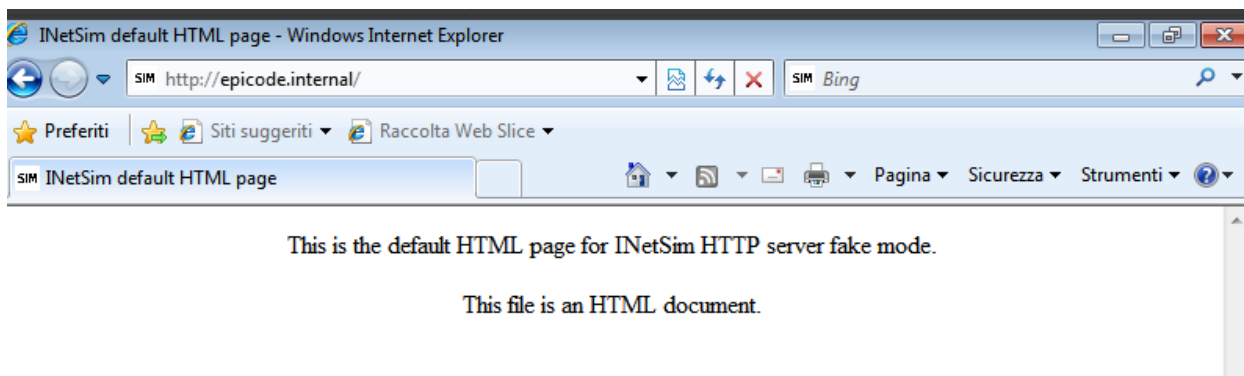
Dopodiché possiamo avviare una simulazione per assicurarci che la configurazione sia andata a buon fine:

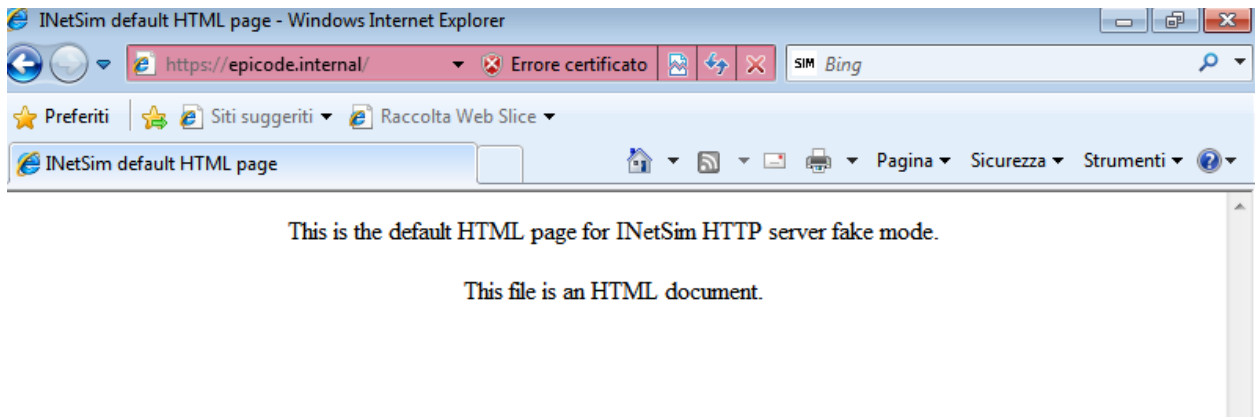
```
(kali@kali)-[/etc/inetsim]
$ sudo inetsim
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Using log directory: /var/log/inetsim/
Using data directory: /var/lib/inetsim/
Using report directory: /var/log/inetsim/report/
Using configuration file: /etc/inetsim/inetsim.conf
Parsing configuration file.
Configuration file parsed successfully.
== INetSim main process started (PID 18960) ==
Session ID: 18960
Listening on: 0.0.0.0
Real Date/Time: 2023-11-24 09:20:25
Fake Date/Time: 2023-11-24 09:20:25 (Delta: 0 seconds)
Forking services ...
* dns_53_tcp_udp - started (PID 18962)
* http_80_tcp - started (PID 18963)
print() on closed filehandle MLOG at /usr/share/perl5/Net/DNS/Nameserver.pm line 399.
print() on closed filehandle MLOG at /usr/share/perl5/Net/DNS/Nameserver.pm line 399.
* https_443_tcp - started (PID 18964)
done.
Simulation running.
```

Da questa immagine possiamo vedere che i servizi dns, http e https si avviano correttamente.

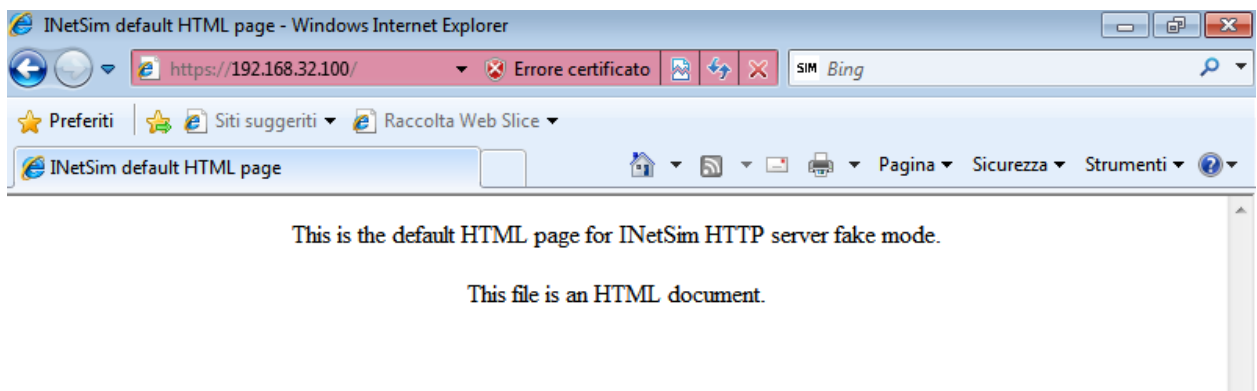
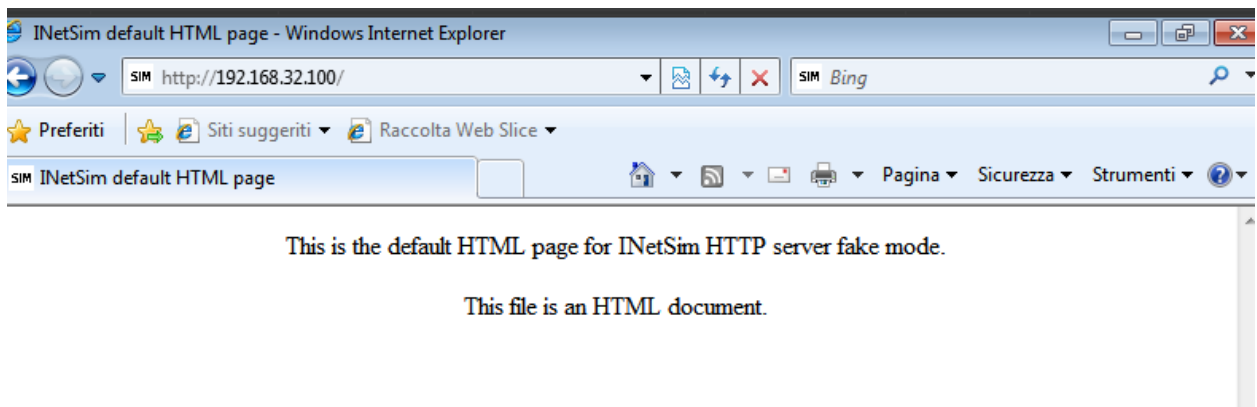
III. Richiesta HTTP e HTTPS

Adesso che tutto è stato configurato, è necessario verificare la buona riuscita della configurazione recandosi su Windows 7 e lanciando le richieste HTTP e HTTPS tramite hostname epicode.internal:





Verificare poi che le richieste vengano aperte correttamente anche con IP 192.168.32.100:



IV. Wireshark

Wireshark è un network sniffer e packet analyzer che ci permette di analizzare qualsiasi pacchetto, flusso di traffico o connessione che passa dalla scheda di rete del computer.

Apriamo Wireshark da Kali Linux, scegliamo la NIC eth0 e nel frattempo accediamo a ["http://epicode.internal"](http://epicode.internal) da Windows 7. Vedremo che la scheda inizierà a popolarsi di pacchetti.

Tra i vari dati che ci mette a disposizione Wireshark, è possibile visualizzare sia per le richieste HTTP che per le HTTPS gli indirizzi IP di Windows 7 e Kali Linux così come i MAC address sorgente e destinazione:

```
▼ Ethernet II, Src: PcsCompu_62:50:67 (08:00:27:62:50:67), Dst: PcsCompu_cb:7e:f5 (08:00:27:cb:7e:f5)
  ► Destination: PcsCompu_cb:7e:f5 (08:00:27:cb:7e:f5)
  ► Source: PcsCompu_62:50:67 (08:00:27:62:50:67)
  Type: IPv4 (0x0800)
  ▼ Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
```

Per avere conferma che anche i MAC address corrispondano a quelli delle macchine virtuali è possibile aprire il terminale da Windows 7 e fare una verifica tramite il comando "ipconfig /all" e poi fare lo stesso da Kali Linux tramite comando "ifconfig":

```
Scheda Ethernet Connessione alla rete locale (LAN):

Suffisso DNS specifico per connessione:
Descrizione . . . . . : Scheda desktop Intel(R) PRO/1000 MT
Indirizzo fisico. . . . . : 08-00-27-62-50-67
DHCP abilitato . . . . . : No
```

```
(kali@kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.32.100 netmask 255.255.255.0 broadcast 192.168.32.255
    inet6 fe80::a00:27ff:feeb:7ef5 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:cb:7e:f5 txqueuelen 1000 (Ethernet)
    RX packets 1067 bytes 93069 (90.8 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 406 bytes 58830 (57.4 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 124 bytes 6240 (6.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 124 bytes 6240 (6.0 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(kali@kali)-[~]
$
```

Richiesta HTTP:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu.62:50:67	Broadcast	ARP	60	who has 192.168.32.100? Tell 192.168.32.101
2	0.000032862	PcsCompu.cb:7e:f5	PcsCompu.62:50:67	ARP	42	192.168.32.100 is at 00:00:27:cb:7e:f5
3	0.001070080	192.168.32.101	192.168.32.100	TCP	60	49179 → 80 [SYN] Seq=0 Win=0 Len=0 MSS=1460 WS=4 SACK_PERM
4	0.001274897	192.168.32.100	192.168.32.101	TCP	60	80 → 49179 [SYN, ACK] Seq=0 Ack=1 Win=0 Len=0 MSS=1460 SACK_PERM WS=128
5	0.001775537	192.168.32.101	192.168.32.100	TCP	60	49179 → 80 [ACK] Seq=1 Ack=1 Win=65760 Len=0
6	0.002244517	192.168.32.100	192.168.32.101	TCP	54	80 → 49179 [ACK] Seq=1 Ack=287 Win=64128 Len=0
7	0.002244517	192.168.32.100	192.168.32.101	TCP	204	80 → 49179 [PSH, ACK] Seq=1 Ack=287 Win=64128 Len=150 [TCP segment of a reassembled PDU]
8	0.002360377	192.168.32.100	192.168.32.101	TCP	332	HTTP/1.1 200 OK (text/html)
9	0.030379894	192.168.32.101	192.168.32.100	HTTP	112	[TCP Retransmission] 80 → 49179 [FIN, 200 OK] Seq=151 Ack=287 Win=64128 Len=250
10	0.045255341	192.168.32.100	192.168.32.101	TCP	60	49179 → 80 [ACK] Seq=287 Ack=410 Win=65292 Len=0 SLE=151 SRE=400
11	0.090277331	192.168.32.101	192.168.32.100	TCP	60	49179 → 80 [FIN, ACK] Seq=287 Ack=410 Win=65292 Len=0
12	0.092182146	192.168.32.101	192.168.32.100	TCP	54	80 → 49179 [ACK] Seq=410 Ack=288 Win=64128 Len=0
13	0.092221012	192.168.32.101	192.168.32.100	TCP	42	who has 192.168.32.100? Tell 192.168.32.100
14	0.226002170	PcsCompu.cb:7e:f5	PcsCompu.62:50:67	ARP	60	192.168.32.101 is at 00:00:27:cb:7e:f5
15	0.239233401	PcsCompu.62:50:67	PcsCompu.cb:7e:f5	ARP	60	192.168.32.101 is at 00:00:27:cb:7e:f5

<pre> + Frame 6: 340 bytes on wire (2720 bits), 340 bytes captured (2720 bits) on interface eth0, id 0 + Ethernet II, Src: PcsCompu.62:50:67 (00:00:27:cb:7e:f5), Dst: PcsCompu.cb:7e:f5 (00:00:27:cb:7e:f5) + Internet Protocol Version 4, Src: 192.168.32.100, Dst: 192.168.32.101 + Transmission Control Protocol, Src Port: 49179, Dst Port: 80, Seq: 1, Ack: 1, Len: 286 Source Port: 49179 Destination Port: 80 [Stream index: 0] [Conversation completeness: Complete, WITH_DATA (31)] [TCP Segment Len: 286] Sequence Number: 1 (relative sequence number) Sequence Number (raw): 4184554674 [Next Sequence Number: 287 (relative sequence number)] Acknowledgment Number: 1 (relative ack number) Acknowledgment number (raw): 1923284766 [Data ... = Header Length: 20 bytes (5)] Flags: 0x018 (PSH, ACK) Window: 16425 [Calculated window size: 65760] [Window size scaling factor: 4] Checksum: 0x4085 [unverified] [Checksum Status: Unverified] Urgent Pointer: 0 [Timestamps] [SEQ/ACK analysis] TCP payload (286 bytes) + Hypertext Transfer Protocol </pre>	<pre> 0000 08 00 27 cb 7e f5 08 00 27 62 50 67 00 00 85 00 ..bpc.. 0010 08 00 27 cb 7e f5 08 00 27 62 50 67 00 00 85 00 ..bpc.. 0020 2f 31 2e 31 06 0a 41 63 63 65 70 74 20 4c 61 6e 1.1 AC capt: / 0030 28 0d 0a 41 63 63 65 70 74 20 4c 61 6e 67 75 61 " Accep t-Langua 0040 67 65 3a 20 65 6e 65 63 6d 6a 65 73 63 72 2d 3d get-en-U S User- 0050 41 67 65 6e 74 3a 20 4d 67 7a 69 6c 6c 61 2f 34 Agent: Mozilla/4 0060 26 30 20 20 63 6f 6d 70 61 74 69 62 6c 62 30 20 B [comp atible; 0070 44 53 49 45 20 30 2e 30 20 57 69 6e 64 67 77 MSIE 5.0 ; Window 0080 73 20 4e 54 20 30 2e 31 30 20 57 4f 57 36 34 30 s NT 6.1 ; WdW64; 0090 28 54 72 69 64 65 6e 74 2f 34 2e 30 30 20 30 4e Trident /4.0; SL 00a0 43 43 32 30 20 2e 4e 45 54 20 43 4c 52 20 32 2e C2; .NET CLR 2. 00b0 30 2e 35 39 57 32 37 20 2e 4e 45 54 20 4e 4c 0.50727; .NET CL 00c0 52 20 33 2e 35 2e 33 30 37 32 30 30 20 2e 4e 45 R 3.5.30 729; .NE 00d0 54 20 43 4c 52 20 33 2e 30 2e 33 30 37 32 39 29 T CLR 3. 0.30729) 00e0 8d 0a 41 63 63 65 70 74 20 45 66 63 6f 64 69 6e Accept-Encodin 00f0 67 3a 20 67 7a 69 70 2c 20 64 65 66 6c 61 74 65 g; gzip, deflate 0100 8d 0a 48 6f 73 74 3a 20 65 70 69 63 6f 64 65 2e Host: epacode. 0110 69 6e 74 65 72 6e 61 6c 6d 0a 43 6f 6e 6e 65 63 internal Connec 0120 74 69 6f 6e 3a 20 4b 65 65 70 20 41 6c 69 76 65 tion: Ke ep-Alive 0130 6d 0a 6d 0a </pre>
--	---

Da questa schermata è possibile vedere i vari protocolli. Trattandosi di una richiesta HTTP (traffico non protetto), possiamo notare che non è presente alcun protocollo SSL (Secure Sockets Layer) o TLS (Transport Layer Security) che garantiscano la trasmissione sicura dei dati.

Da qui possiamo notare la richiesta effettuata (GET) inviata dal Windows7 (src 192.168.32.101) e la risposta (200 OK) che il server (src 1902.168.32.100) manda al Windows.

Possiamo notare che la “destination port” che è la 80, porta utilizzata per il traffico HTTP.

Richiesta HTTPS:

No.	Time	Source	Destination	Protocol	Length	Info
48	15.30783608	192.168.32.101	192.168.32.100	TCP	60	49190 → 80 [ACK] Seq=218 Ack=410 Win=5200 Len=0
47	15.30783584	192.168.32.101	192.168.32.100	TCP	60	49190 → 80 [FIN, ACK] Seq=218 Ack=410 Win=5200 Len=0
46	15.307892109	192.168.32.100	192.168.32.101	TCP	54	80 → 49190 [ACK] Seq=410 Ack=218 Win=6420 Len=0
45	15.30789468	192.168.32.100	192.168.32.100	TCP	60	49190 → 443 [FIN, ACK] Seq=410 Ack=410 Win=5200 Len=0
50	15.314548633	192.168.32.100	192.168.32.101	TLSv1	93	Encrypted Alert
51	15.314548633	192.168.32.101	192.168.32.100	TLSv1	93	Encrypted Alert
52	64.42574658	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.100? Tell 192.168.32.101
53	64.426922987	PcCompu.cb:7e:f5	PcCompu.02:50:07	ARP	42	192.168.32.100 is at 90:0e:27:cb:7e:f5
54	64.426922987	192.168.32.100	192.168.32.100	TCP	60	49197 → 443 [FIN] Seq=410 Ack=410 Win=5200 Len=0
55	64.42700044	192.168.32.101	192.168.32.100	TCP	60	49197 → 443 [ACK] Seq=410 Ack=410 Win=5200 Len=0
56	64.42700044	192.168.32.101	192.168.32.100	TCP	60	49197 → 443 [ACK] Seq=410 Ack=410 Win=5200 Len=0
57	64.42700044	192.168.32.101	192.168.32.100	TCP	60	49197 → 443 [ACK] Seq=410 Ack=410 Win=5200 Len=0
58	64.42700044	192.168.32.101	192.168.32.100	TCP	60	49197 → 443 [ACK] Seq=410 Ack=410 Win=5200 Len=0
59	64.42700044	192.168.32.100	192.168.32.101	TLSv1	1360	Server Hello, Certificate, Server Key Exchange, Server Hello Done
60	64.42700044	192.168.32.101	192.168.32.100	TLSv1	1360	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
61	64.42700044	192.168.32.100	192.168.32.101	TCP	54	443 → 49197 [ACK] Seq=310 Ack=218 Win=6420 Len=0
62	64.42700044	192.168.32.100	192.168.32.101	TLSv1	113	Change Cipher Spec, Encrypted Handshake Message
63	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
64	64.42700044	192.168.32.101	192.168.32.100	TCP	60	49197 → 443 [ACK] Seq=310 Ack=310 Win=6420 Len=0
65	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
66	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
67	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
68	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
69	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
70	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
71	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
72	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
73	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
74	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
75	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
76	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
77	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
78	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
79	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
80	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
81	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
82	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
83	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
84	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
85	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
86	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
87	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
88	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
89	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
90	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
91	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
92	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
93	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
94	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
95	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
96	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
97	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
98	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
99	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101
100	64.42700044	PcCompu.02:50:07	Broadcast	ARP	60	Who has 192.168.32.17? Tell 192.168.32.101

Trattandosi qui di una richiesta HTTPS, prima del trasferimento dei dati viene stabilita una connessione sicura e crittografata, com'è possibile notare già dal riquadro nero a destra dell'immagine.

Vediamo inoltre che la destination port è la 443, porta standard per il traffico HTTPS.

A differenza della richiesta HTTP, vediamo che qui è presente il protocollo TLSv1, si può notare infatti che è stata avviata una comunicazione mirata alla protezione dei dati. Inoltre, essendo stata stabilita una connessione sicura, non è visibile qui la richiesta GET né la risposta con codice di stato.

