

Progetto Modulo 1

Ettore Farris - 17/11/2023

1) Consegna del progetto

“Nell’esercizio di oggi metteremo insieme le competenze acquisite finora. Lo studente verrà valutato sulla base della risoluzione al problema seguente.

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.”

2) Svolgimento

2.1) Impostazioni IP su Kali Linux e Windows 7

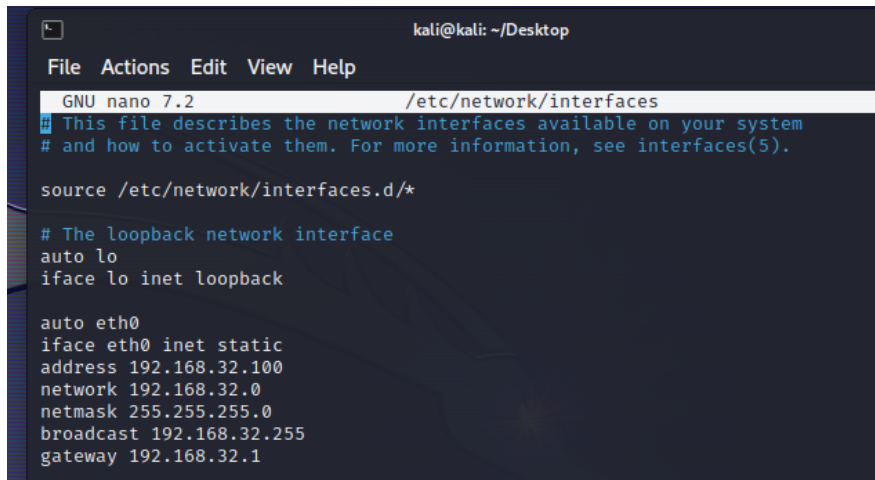
L'ambiente di lavoro è composto da due macchine virtuali, una Kali Linux e una Windows 7 in una rete interna su VirtualBox. Per far sì che le due macchine comunichino, la prima cosa da fare è cambiare le impostazioni IP delle due macchine. In questo progetto, lavoreremo con degli indirizzi IP statici.

- Configurazione IP Kali Linux

Per modificare le impostazioni IP di Kali Linux apriamo un terminale e lanciamo il seguente comando:

- `sudo nano /etc/network/interfaces`

e modificare il file come segue per far sì che venga impostato l'indirizzo IP `192.168.32.100`:



```
kali@kali: ~/Desktop
File Actions Edit View Help
GNU nano 7.2 /etc/network/interfaces
# 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

auto eth0
iface eth0 inet static
address 192.168.32.100
network 192.168.32.0
netmask 255.255.255.0
broadcast 192.168.32.255
gateway 192.168.32.1
```

Affinché le modifiche abbiano effetto, salviamo il file e riavviamo l'interfaccia di rete col comando:

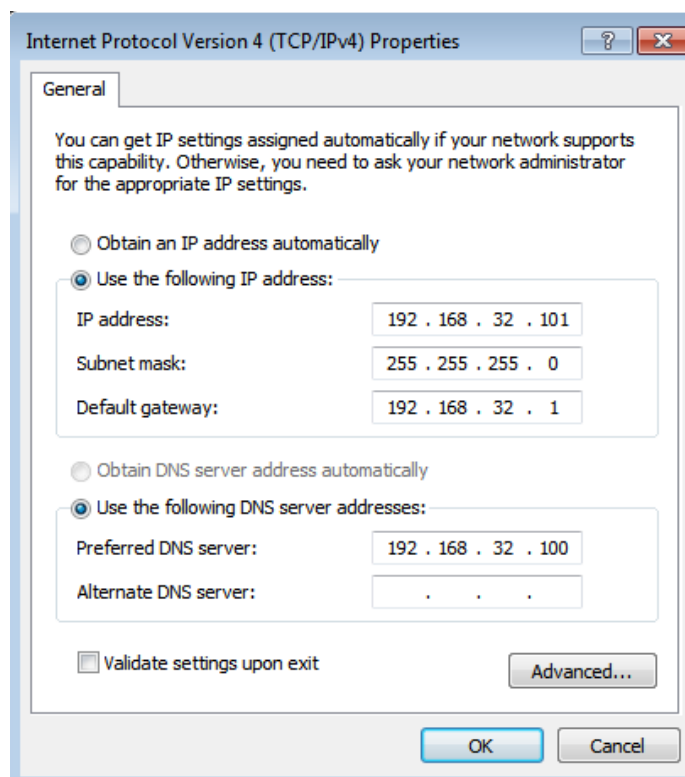
- `sudo systemctl restart networking`

Lanciando il comando *ifconfig* possiamo confermare che le modifiche sono state salvate con successo.

```
(kali@kali)-[~/Desktop]
$ ifconfig
eth0: flags=4163<UP,BROADCAST
    inet 192.168.32.100
```

- Configurazione IP e DNS su Windows 7

Facciamo la stessa cosa su Windows 7 entrando nelle impostazioni di rete dal Pannello di Controllo. Dato che questa macchina dovrà richiedere tramite il browser delle risorse al dominio *epicode.internal*, è necessario impostare anche l'indirizzo DNS che risolverà tale hostname. In questo progetto, l'IP del server DNS sarà quello della macchina di Kali Linux, che mediante il tool *INetSim* ospiterà sia il server HTTP e HTTPS che quello DNS (vedi sezione successiva).



Lanciando il comando *ipconfig /all* nel prompt dei comandi si può notare che le impostazioni IP e DNS sono state salvate con successo.

```
C:\Windows\system32\cmd.exe

C:\Users\user>ipconfig /all

Windows IP Configuration

Host Name . . . . . : WIN-845Q99004PP
Primary Dns Suffix . . . . . :
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No

Ethernet adapter Local Area Connection 2:

   Connection-specific DNS Suffix  :
   Description . . . . . : Intel(R) PRO/1000 MT Network Connection
   2
   Physical Address. . . . . : 08-00-27-97-4D-54
   DHCP Enabled. . . . . : No
   Autoconfiguration Enabled . . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::b0b9:3bd:84d0:e820%15(Preferred)
   IPv4 Address. . . . . : 192.168.32.101(Preferred)
   Subnet Mask . . . . . : 255.255.255.0
   Default Gateway . . . . . : 192.168.32.1
   DHCPv6 IAID . . . . . : 319291431
   DHCPv6 Client DUID. . . . . : 00-01-00-01-28-88-AD-4E-00-0C-29-4A-E2-4
   .
   DNS Servers . . . . . : 192.168.32.100
   NetBIOS over Tcpip. . . . . : Enabled
```

- Ping tra Kali e Windows 7

Dopo aver cambiato le impostazioni IP delle due macchine assicuriamoci che le due VM possano comunicare *pingando* una macchina dall'altra, in questo caso Kali Linux da Windows 7.

```
C:\Users\user>ping 192.168.32.100

Pinging 192.168.32.100 with 32 bytes of data:
Reply from 192.168.32.100: bytes=32 time<1ms TTL=64
Reply from 192.168.32.100: bytes=32 time=2ms TTL=64
Reply from 192.168.32.100: bytes=32 time=2ms TTL=64
Reply from 192.168.32.100: bytes=32 time=1ms TTL=64
```

2.2) Configurazione server DNS, HTTP e HTTPS

Per fare in modo che il browser Windows 7 possa richiedere una pagina web con protocollo HTTP e HTTPS all'hostname *epicode.internal*, ci serviremo del tool *INetSim*, un simulatore di servizi presente su Kali Linux. Questo programma, una volta lanciato, simulerà i server HTTP, HTTPS e DNS che ci serviranno. Per prima

cosa, dobbiamo modificare le impostazioni di INetSim cambiando alcune voci dal file di configurazione lanciando il seguente comando:

- `sudo nano /etc/inetsim/inetsim.conf`

Commentiamo tutto tranne i servizi che ci servono, ovvero HTTP, HTTPS e DNS. A seguire, impostiamo il *service bind address* sull'indirizzo IP per tutte le interfacce (0.0.0.0), impostiamo l'IP DNS di default con l'IP di Kali e un DNS statico che associ l'IP di Kali Linux all'hostname *epicode.internal*:

```
GNU nano 7.2 /etc/inetsim/inetsim.conf *
#
# 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
```

```
#####
# service_bind_address ff02::1:2
#
# IP address to bind services to
#
# Syntax: service_bind_address <IP address>
# : : b0b9:3bd:84d0::ff02::1:2
# Default: 127.0.0.1 ff02::1:2
# : : b0b9:3bd:84d0::ff02::1:2
service_bind_address 0.0.0.0
```

```
#####
# dns_default_ip Destination Proto
# : : b0b9:3bd:84d0::ff02::1:2 DHCP
# Default IP address to return with DNS replies
# : : b0b9:3bd:84d0::ff02::1:2 DHCP
# Syntax: dns_default_ip <IP address>
# : : b0b9:3bd:84d0::ff02::1:2 DHCP
# Default: 127.0.0.1 ff02::1:2
# : : b0b9:3bd:84d0::ff02::1:2 DHCP
dns_default_ip 192.168.32.100
```

```
#####
# 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
#####
# dns_version
```

Salviamo il file e, affinché le modifiche abbiano effetto, *"riavviamo"* il programma con i seguenti comandi:

- `sudo systemctl restart inetsim.service`
- `service inetsim stop`

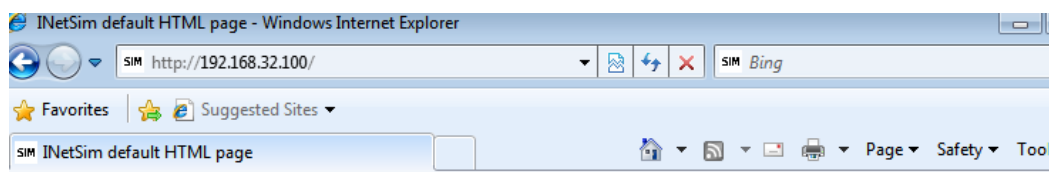
e lanciamolo con `sudo inetsim` per aprire una simulazione e per verificare che i 3 servizi che ci servono vengano avviati correttamente:

```
(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/
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 89790) ==
Session ID: 89790
Listening on: 0.0.0.0
Real Date/Time: 2023-11-17 16:41:28
Fake Date/Time: 2023-11-17 16:41:28 (Delta: 0 seconds)
Forking services ...
* https_443_tcp - started (PID 89794)
* dns_53_tcp_udp - started (PID 89792)
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.
* http_80_tcp - started (PID 89793)
done.
Simulation running.
```

2.2) Richiesta HTTP e HTTPS da Internet Explorer e cattura pacchetti con Wireshark

- **Richiesta HTTP e HTTPS**

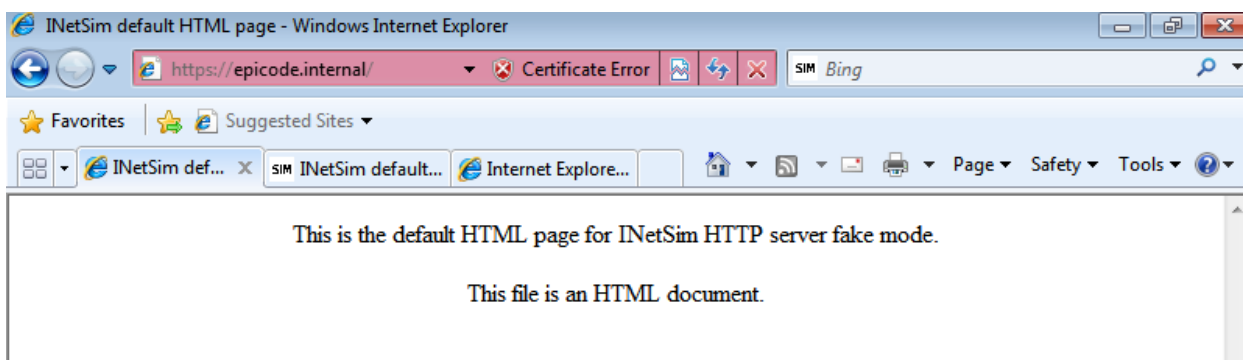
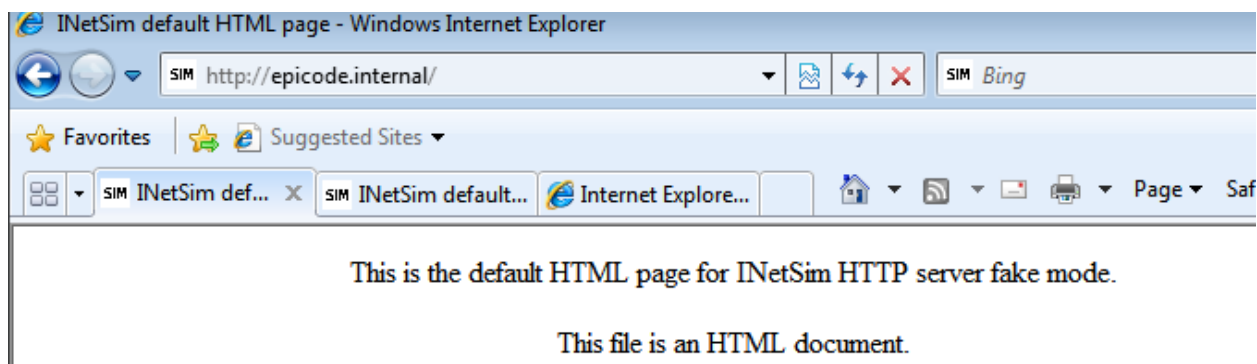
Per verificare il buon funzionamento delle configurazioni che abbiamo impostato sino ad ora, apriamo Internet Explorer da Windows 7 e tentiamo di mandare una richiesta ai server HTTP e HTTPS sia tramite l'indirizzo IP:



This is the default HTML page for INetSim HTTP server fake mode.

This file is an HTML document.

che tramite l'hostname *epicode.internal*:



- **Cattura pacchetti con Wireshark**

Per intercettare il traffico, usiamo Wireshark. Come da consegna, intercetteremo due tipi di traffico: quello HTTP e quello HTTPS relativi all'hostname *"epicode.internal"*. In sostanza, il browser di Windows 7 effettuerà una richiesta GET al server HTTP o HTTPS situato sulla macchina Kali Linux il quale risponderà con successo con una semplice pagina Html, quindi con codice di stato 200 (OK).

- Traffico HTTPS

Su Wireshark, i pacchetti passano per l'interfaccia di rete *eth0*. Selezionandola e contestualmente accedendo a "*https://epicode.internal*" dalla macchina Windows 7 l'intercettazione risultante è la seguente:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.32.101	192.168.32.100	TCP	66	49232 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
2	0.000093102	192.168.32.100	192.168.32.101	TCP	66	443 → 49232 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
3	0.002325160	192.168.32.101	192.168.32.100	TCP	54	49232 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
4	0.005574532	192.168.32.101	192.168.32.100	TLSv1	215	Client Hello
5	0.005603921	192.168.32.100	192.168.32.101	TCP	54	443 → 49232 [ACK] Seq=1 Ack=162 Win=64128 Len=0
6	0.106338367	192.168.32.100	192.168.32.101	TLSv1	1373	Server Hello, Certificate, Server Key Exchange, Server Hello Done
7	0.128832492	192.168.32.101	192.168.32.100	TLSv1	188	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
8	0.130083942	192.168.32.100	192.168.32.101	TLSv1	113	Change Cipher Spec, Encrypted Handshake Message
9	0.173425924	192.168.32.101	192.168.32.100	TLSv1	395	Application Data
10	0.202982154	192.168.32.100	192.168.32.101	TLSv1	235	Application Data
11	0.209085783	192.168.32.100	192.168.32.101	TLSv1	384	Application Data, Encrypted Alert
12	0.211952277	192.168.32.101	192.168.32.100	TCP	54	49232 → 443 [ACK] Seq=637 Ack=1891 Win=65700 Len=0
13	0.211952629	192.168.32.101	192.168.32.100	TCP	54	49232 → 443 [FIN, ACK] Seq=637 Ack=1891 Win=65700 Len=0
14	0.212003255	192.168.32.100	192.168.32.101	TCP	54	443 → 49232 [ACK] Seq=1891 Ack=638 Win=64128 Len=0

Trattandosi di protocollo HTTPS, i dati sono criptati. Infatti, dopo la *three-way-handshake* il client invia un pacchetto TLSv1 "*Client Hello*" a cui il server risponde con un altro pacchetto TLSv1 "*Server Hello*". Questo scambio di pacchetti TLS è una negoziazione client/server per garantire la protezione dei dati sensibili durante la comunicazione tra client e server. Il TLS (*Transport Layer Security*) è un protocollo di livello 4 finalizzato a proteggere i dati sensibili durante la comunicazione tra client e server.

Le informazioni trasmesse sono criptate, pertanto non sono visibili in chiaro. Possiamo tuttavia vedere gli indirizzi IP delle due macchine e gli indirizzi MAC sorgente e destinazione per ogni singolo frame. Il primo pacchetto scambiato è quello ACK da Windows 7 a Kali Linux. L'header di livello 2 contenente i MAC source (Windows 7) e destination (Kali Linux) restituisce questi risultati:

```
▶ Frame 1: 66 bytes on wire (528 bits), 66 bytes captured
▶ Ethernet II, Src: PcsCompu_97:4d:54 (08:00:27:97:4d:54)
  ▶ Destination: PcsCompu_78:a0:d3 (08:00:27:78:a0:d3)
  ▶ Source: PcsCompu_97:4d:54 (08:00:27:97:4d:54)
  Type: IPv4 (0x0800)
```


Confrontando gli indirizzi MAC delle due macchine (ottenibili lanciando il comando *ifconfig* su terminale Linux e *ipconfig /all* su prompt dei comandi Windows) possiamo confermare l'attendibilità dei dati intercettati.

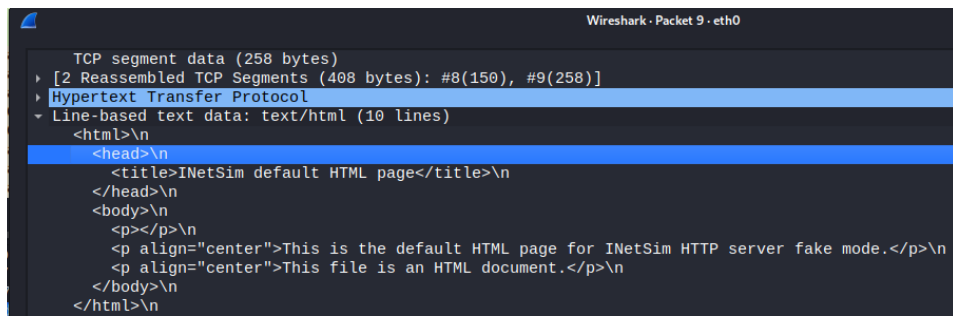
<pre>Physical Address. : 08-00-27-97-4D-54 DHCP Enabled. : No</pre>	Windows 7
<pre>ether 08:00:27:78:a0:d3</pre>	Kali Linux

- **Traffico HTTP**

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_97:4d:54	Broadcast	ARP	42	Who has 192.168.32.100? Tell 192.168.32.101
2	0.000035475	PcsCompu_78:a0:d3	PcsCompu_97:4d:54	ARP	42	192.168.32.100 is at 08:00:27:78:a0:d3
3	0.001032852	192.168.32.101	192.168.32.100	TCP	66	49213 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM
4	0.001085258	192.168.32.100	192.168.32.101	TCP	66	80 → 49213 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SAC
5	0.002153158	192.168.32.101	192.168.32.100	TCP	54	49213 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
6	0.003118861	192.168.32.101	192.168.32.100	HTTP	359	GET / HTTP/1.1
7	0.003134333	192.168.32.100	192.168.32.101	TCP	54	80 → 49213 [ACK] Seq=1 Ack=306 Win=64128 Len=0
8	0.021089823	192.168.32.100	192.168.32.101	TCP	294	80 → 49213 [PSH, ACK] Seq=1 Ack=306 Win=64128 Len=150 [TCP seq
9	0.023249366	192.168.32.100	192.168.32.101	HTTP	312	HTTP/1.1 200 OK (text/html)
10	0.023958674	192.168.32.101	192.168.32.100	TCP	54	49213 → 80 [ACK] Seq=306 Ack=410 Win=65292 Len=0
11	0.024377411	192.168.32.101	192.168.32.100	TCP	54	49213 → 80 [FIN, ACK] Seq=306 Ack=410 Win=65292 Len=0
12	0.024389737	192.168.32.100	192.168.32.101	TCP	54	80 → 49213 [ACK] Seq=410 Ack=307 Win=64128 Len=0

Guardando attentamente, possiamo notare:

- uno scambio di pacchetti ARP. Nel primo pacchetto, la macchina Windows (Indirizzo MAC: 08:00:27:92:4d:54) chiede con un messaggio broadcast a tutti i dispositivi della rete a quale indirizzo MAC corrisponde l'indirizzo IP 192.168.32.101. Nel secondo Pacchetto, la VM Kali Linux (Indirizzo MAC: 08:00:27:78:a0:d3) risponde confermando di avere l'indirizzo IP richiesto.
- la *three-way-handshake* tra le due VM;
- la successiva richiesta GET da Windows 7 al server HTTP;
- la risposta con codice di stato 200 che il server trasmette a Windows;
- Cliccando sul pacchetto 7, relativo alla risposta HTTP, possiamo vedere il contenuto in chiaro della pagina Html:

A screenshot of the Wireshark network protocol analyzer interface. The top bar shows 'Wireshark · Packet 9 · eth0'. The left pane shows a tree view with 'TCP segment data (258 bytes)', '[2 Reassembled TCP Segments (408 bytes): #8(150), #9(258)]', 'Hypertext Transfer Protocol', and 'Line-based text data: text/html (10 lines)'. The right pane displays the raw data of the selected packet, which is an HTTP response. The data is shown in a monospaced font with syntax highlighting: <html>\n, <head>\n, <title>INetSim default HTML page</title>\n, </head>\n, <body>\n, <p></p>\n, <p align="center">This is the default HTML page for INetSim HTTP server fake mode.</p>\n, <p align="center">This file is an HTML document.</p>\n, </body>\n, </html>\n.

```
TCP segment data (258 bytes)
  [2 Reassembled TCP Segments (408 bytes): #8(150), #9(258)]
  Hypertext Transfer Protocol
  Line-based text data: text/html (10 lines)
    <html>\n
    <head>\n
      <title>INetSim default HTML page</title>\n
    </head>\n
    <body>\n
      <p></p>\n
      <p align="center">This is the default HTML page for INetSim HTTP server fake mode.</p>\n
      <p align="center">This file is an HTML document.</p>\n
    </body>\n
  </html>\n
```

- Principali differenze tra traffico HTTP e HTTPS

Le principali differenze che si possono notare sono:

- L'HTTP, a differenza dell'HTTPS, dopo la *three-way-handshake* non ha uno scambio di pacchetti TLS per negoziare una connessione criptata;
- Diversamente dall'HTTPS, il traffico HTTP intercettato mostra la "natura" delle richieste e delle risposte, in questo caso rispettivamente HTTP GET e HTTP 200 OK;
- Dato che l'HTTP non consente una connessione criptata, siamo stati in grado di intercettare il contenuto della pagina HTML in chiaro trasmessa dal server in risposta alla richiesta GET.