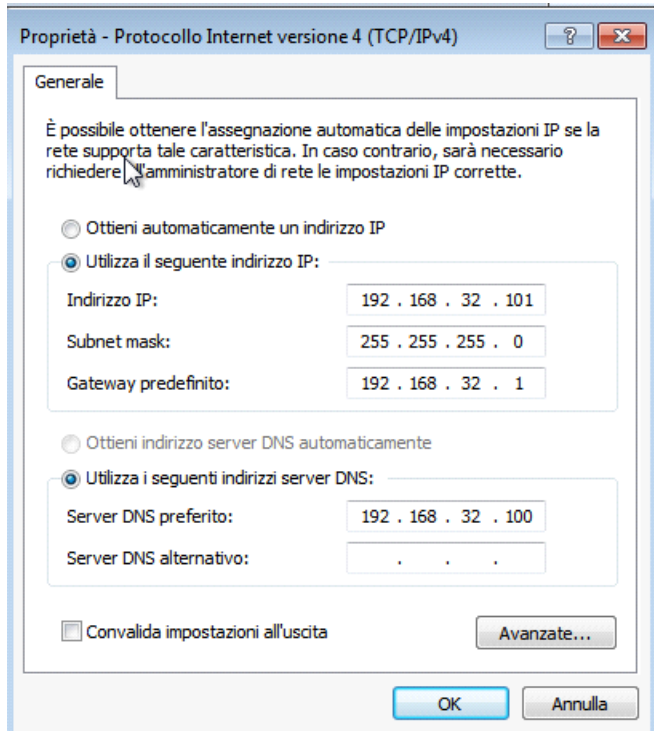


Simulazione servizi DNS, HTTPS, HTTP e utilizzo di Wireshark

Prima di tutto si sono configurati gli IP per l'esercizio su Kali e Windows 7 , inoltre su quest'ultimo si è inserito l'indirizzo del server DNS che verrà successivamente simulato su Kali.

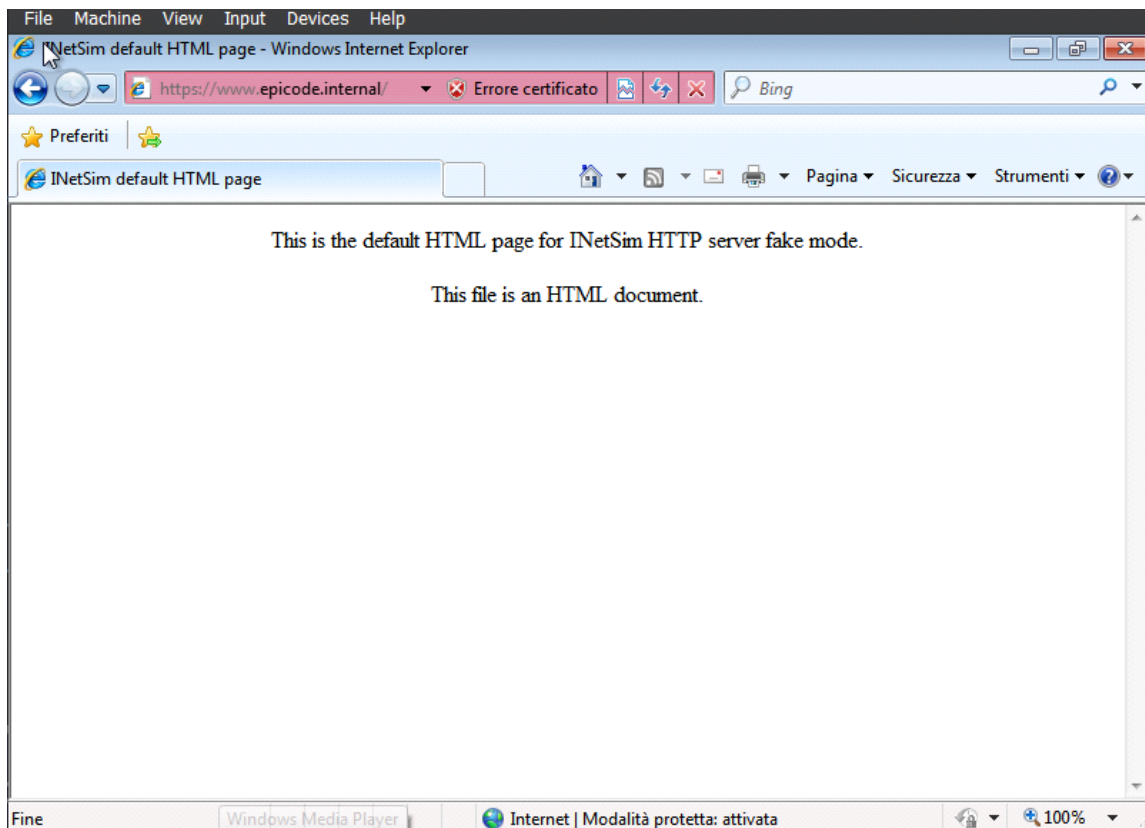


Su Kali si è poi configurato il servizio di DNS con assegnazione dell'indirizzo IP 192.168.32.100 alla risorsa www.epicode.internal

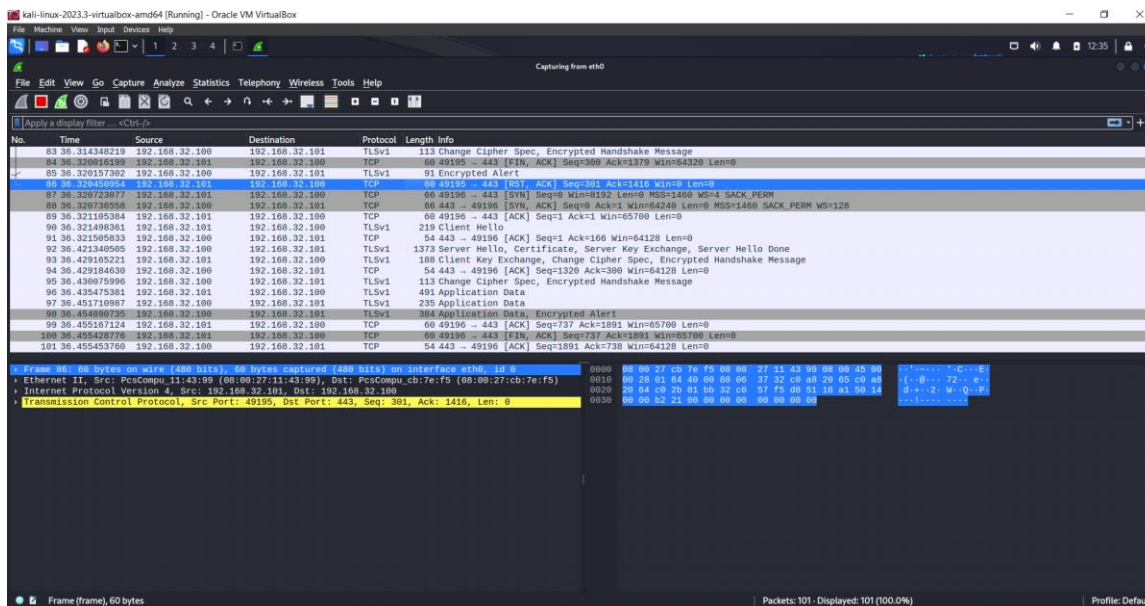
```
kali@kali: ~  
File Actions Edit View Help  
GNU nano 7.2 /etc/inetsim/inetsim.conf *  
#  
# Syntax: dns_default_domainname <domain name>  
#  
# Default: inetsim.org  
#  
#dns_default_domainname epicode.internal  
  
#####  
# dns_static  
#  
# Static mappings for DNS  
#  
# Syntax: dns_static <fqdn hostname> <IP address>  
#  
# Default: none  
#  
dns_static www.epicode.internal 192.168.32.100  
#dns_static ns1.foo.com 10.70.50.30  
#dns_static ftp.bar.net 10.10.20.30  
  
#####  
become, the more you are able to hear"  
  
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute  
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify
```

Poi si è simulato un servizio HTTPS dalla macchina Kali e si è fatta una richiesta dalla macchina Windows 7 per la risorsa epicode.internal

```
kali@kali: ~  
File Actions Edit View Help  
  
(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 40451) ==  
Session ID: 40451  
Listening on: 192.168.32.100  
Real Date/Time: 2023-12-23 12:34:00  
Fake Date/Time: 2023-12-23 12:34:00 (Delta: 0 seconds)  
Forking services ...  
* dns_53_tcp_udp - started (PID 40453)  
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 40454)  
done.  
Simulation running.  
█  
  
"the quieter you become, the more you
```



Si è poi utilizzato il tool Wireshark per la cattura e l'analisi dei pacchetti del traffico HTTPS



Nella figura che segue si può notare sottolineato in rosso l'indirizzo MAC della sorgente ed in verde l'indirizzo MAC del client che ha fatto la richiesta del servizio HTTPS

```

▶ Frame 86: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface eth0, id 0
▶ Ethernet II, Src: PcsCompu_11:43:99 (08:00:27:11:43:99), Dst: PcsCompu_cb:7e:f5 (08:00:27:cb:7e:f5)
▶ Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
▶ Transmission Control Protocol, Src Port: 49195, Dst Port: 443, Seq: 301, Ack: 1416, Len: 0

```

Si è poi effettuato lo stesso procedimento ma con un servizio HTTP, che ha restituito il seguente traffico

The image shows a Wireshark packet capture of an HTTP GET request. The packet list shows a GET request for /favicon.ico. The packet details show the request structure. The packet bytes show the raw data.

No.	Time	Source	Destination	Protocol	Length	Info
15	32.522586142	192.168.32.100	192.168.32.101	TCP	60	80 → 49188 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
16	32.523047217	192.168.32.101	192.168.32.100	TCP	60	49188 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
17	32.523325284	192.168.32.101	192.168.32.100	HTTP	455	GET / HTTP/1.1
18	32.523332625	192.168.32.100	192.168.32.101	TCP	54	80 → 49188 [ACK] Seq=1 Ack=402 Win=64128 Len=0
19	32.546592324	192.168.32.100	192.168.32.101	TCP	204	80 → 49188 [PSH, ACK] Seq=1 Ack=402 Win=64128 Len=150 [TCP segment of a reassembled PDU]
20	32.549582015	192.168.32.100	192.168.32.101	HTTP	312	HTTP/1.1 200 OK (text/html)
21	32.550910722	192.168.32.101	192.168.32.100	TCP	60	49188 → 80 [ACK] Seq=402 Ack=410 Win=65792 Len=0
22	32.550273740	192.168.32.100	192.168.32.100	TCP	60	49188 → 80 [FIN, ACK] Seq=402 Ack=410 Win=65292 Len=0
23	32.550262651	192.168.32.100	192.168.32.101	TCP	54	80 → 49188 [ACK] Seq=410 Ack=403 Win=64128 Len=0
24	32.590442974	192.168.32.101	192.168.32.100	TCP	60	49189 → 80 [SYN] Seq=0 Win=0 MSS=1460 WS=4 SACK_PERM
25	32.590462998	192.168.32.100	192.168.32.101	TCP	60	80 → 49189 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
26	32.591191481	192.168.32.101	192.168.32.100	TCP	60	49189 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
27	32.592059913	192.168.32.101	192.168.32.100	HTTP	331	GET /favicon.ico HTTP/1.1
28	32.592080338	192.168.32.100	192.168.32.101	TCP	54	80 → 49189 [ACK] Seq=1 Ack=278 Win=64128 Len=0
29	32.614416323	192.168.32.100	192.168.32.101	TCP	207	80 → 49189 [PSH, ACK] Seq=1 Ack=278 Win=64128 Len=153 [TCP segment of a reassembled PDU]
30	32.617315330	192.168.32.100	192.168.32.101	HTTP	252	HTTP/1.1 200 OK (image/x-icon)
31	32.617636183	192.168.32.101	192.168.32.100	TCP	60	49189 → 80 [ACK] Seq=278 Ack=353 Win=65340 Len=0
32	32.617840068	192.168.32.101	192.168.32.100	TCP	60	49189 → 80 [FIN, ACK] Seq=278 Ack=353 Win=65340 Len=0
33	32.617851007	192.168.32.100	192.168.32.101	TCP	54	80 → 49189 [ACK] Seq=353 Ack=279 Win=64128 Len=0

Frame 21: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface eth0, id 4
 Ethernet II, Src: PcsCompu_11:43:99 (08:00:27:11:43:99), Dst: PcsCompu_cb:7e:f5 (08:00:27:cb:7e:f5)
 Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
 Transmission Control Protocol, Src Port: 49188, Dst Port: 80, Seq: 402, Ack: 410, Len: 0

0000 00 00 27 cb 7e f5 00 00 27 11 43 99 00 00 45 00
 0010 00 20 01 55 40 00 00 00 37 61 c0 a0 20 05 c0 a0
 0020 20 64 c0 24 00 50 14 12 6e 60 64 0c 43 a0 50 10
 0030 2f c0 c0 50 00 00 00 00 00 00 00

La differenza principale tra i due servizi si può notare nell'utilizzo del protocollo TLS all'interno del servizio HTTPS tramite il quale viene data più sicurezza al traffico e alla lettura dei pacchetti. Mentre utilizzando l'HTTP si può visualizzare il contenuto dei pacchetti non essendo protetti da alcun protocollo.