REPORT W4D4

EPICODE

Ese
Traccia

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

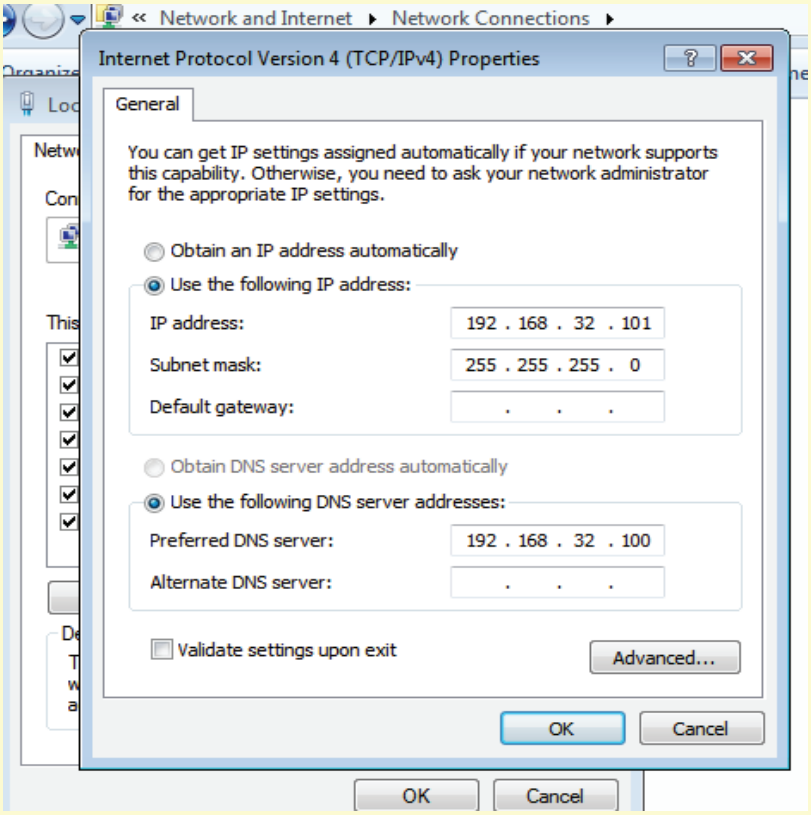
Traccia:

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.

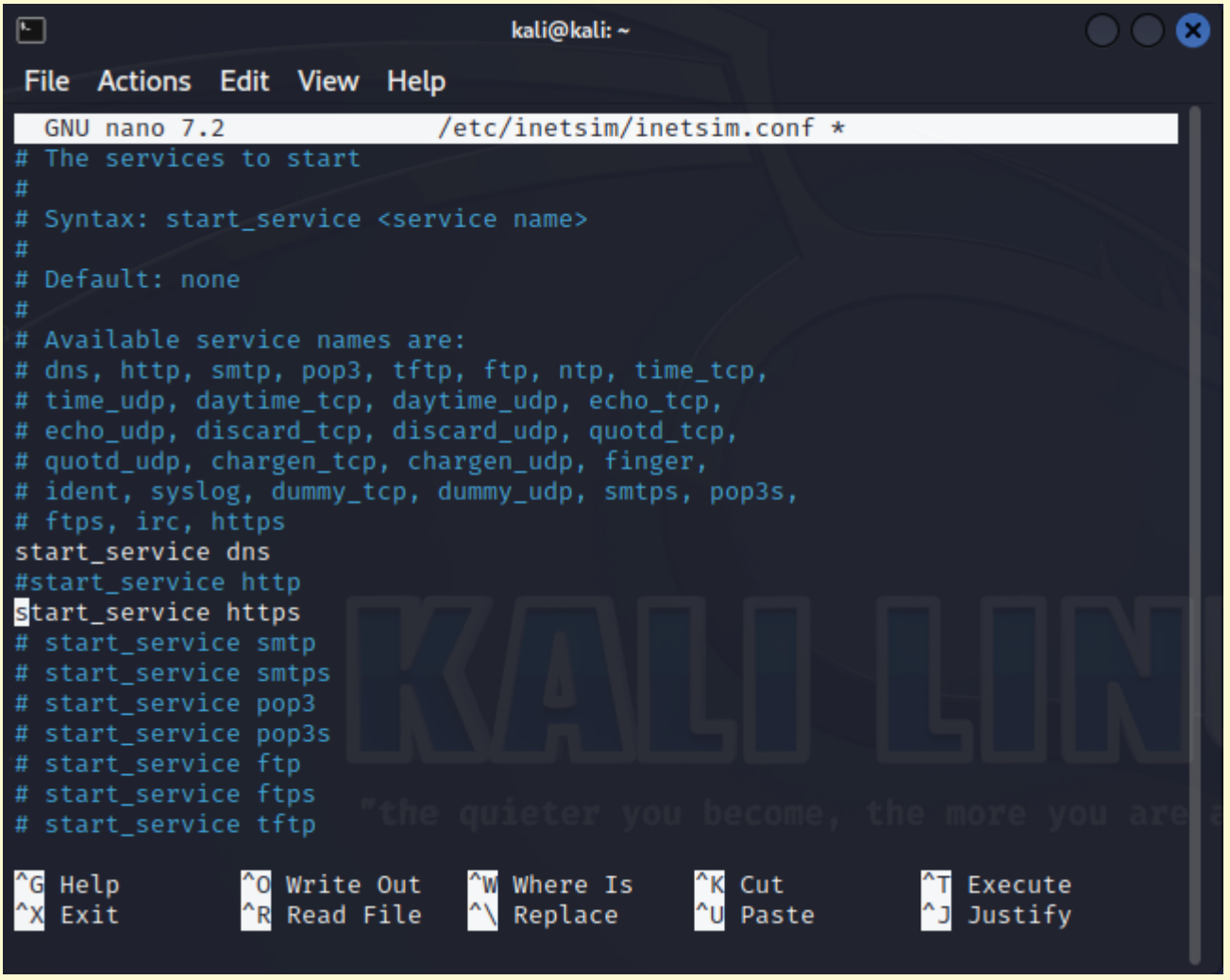
Ripetere l'esercizio, sostituendo il server HTTPS, con un server HTTP. Si intercetti nuovamente il traffico, evidenziando le eventuali differenze tra il traffico appena catturato in HTTP ed il traffico precedente in HTTPS. Spiegare, motivandole, le principali differenze se presenti.

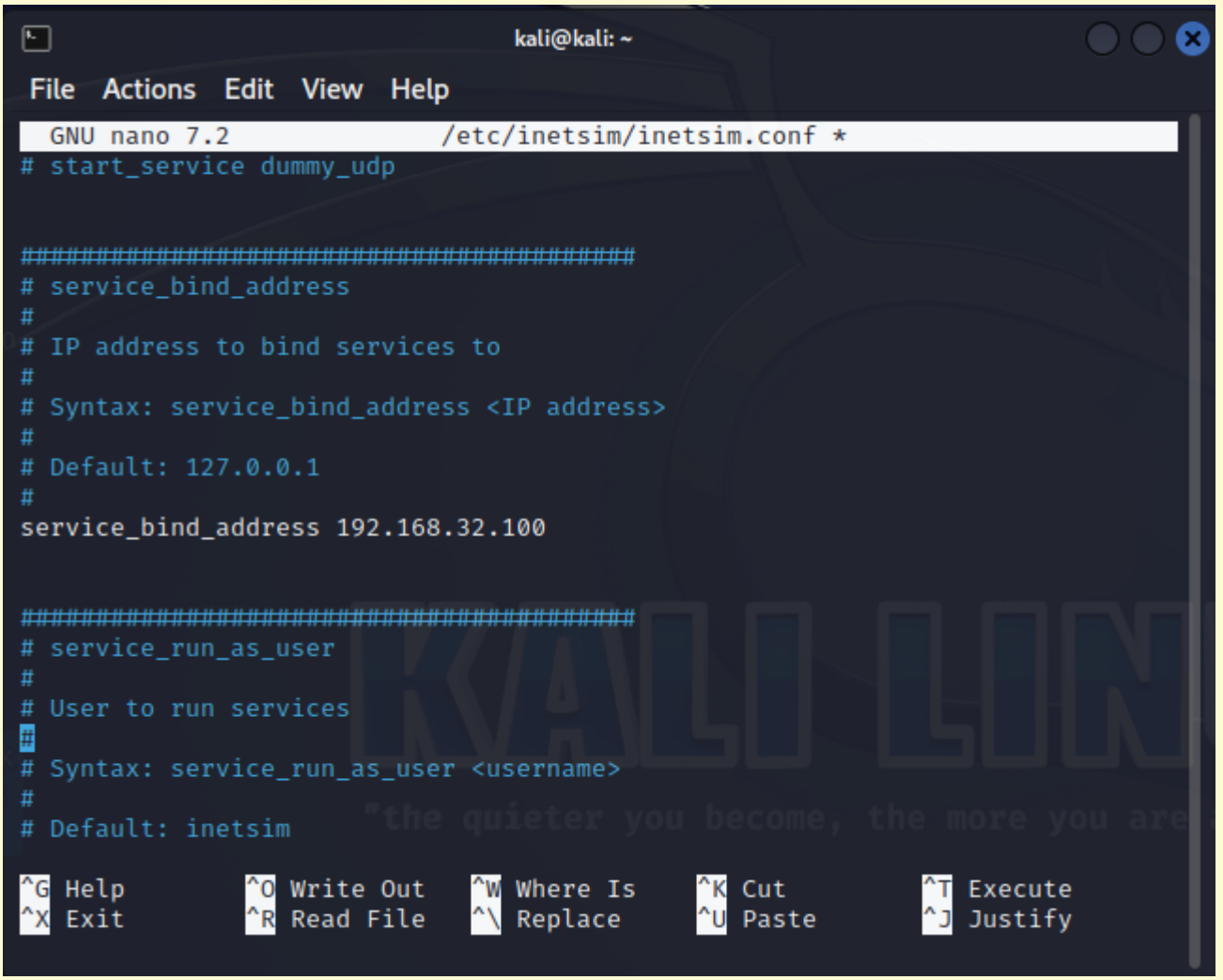
- Iniziamo con l'impostazione degli IP Address di Kali e Windows 7 , impostando il DNS di quest'ultimo come l' IP di Kali .

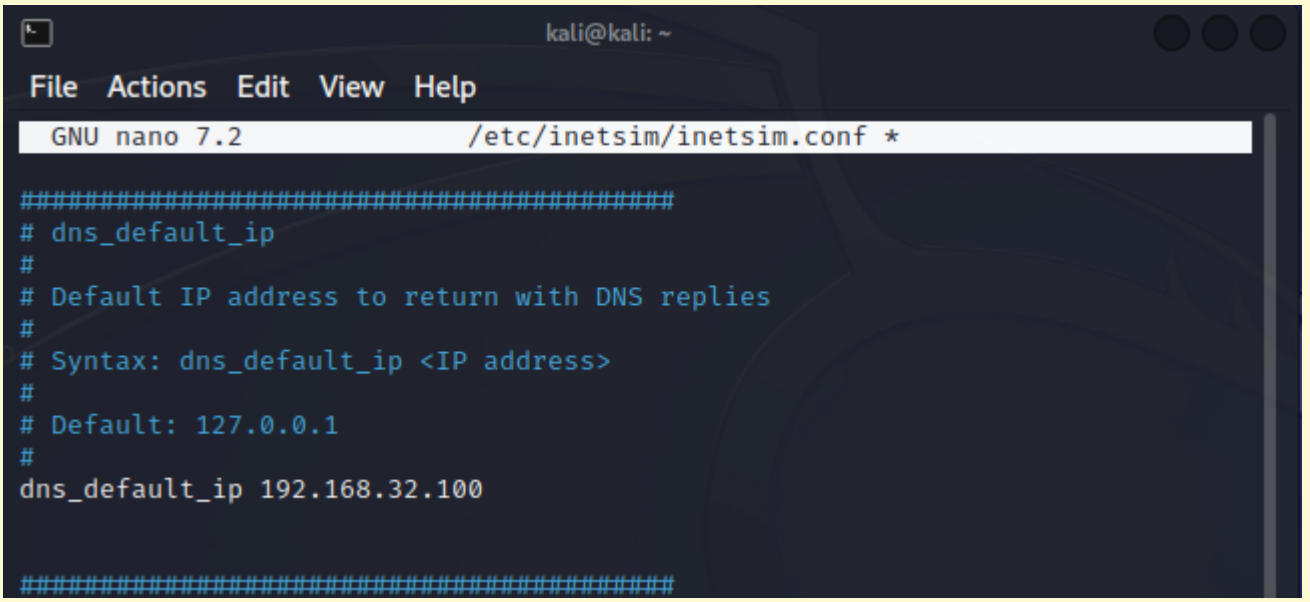


```
  GNU nano 7.2                  /etc/network/interfaces
# This file describes the network interfaces availabl
# and how to activate them. For more information, see

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback


auto eth0
iface eth0 inet static
address 192.168.32.100/24
#gateway 192.168.50.1
```

- Proseguiamo con la configurazione di HTTPS e DNS su Inetsim su Kali Linux come illustrato in figura .



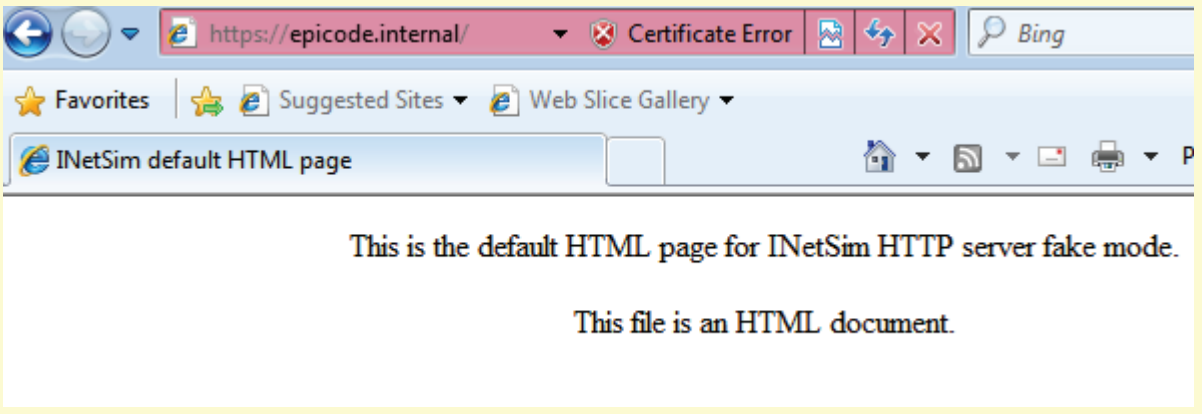

kali@kali: ~
File Actions Edit View Help
GNU nano 7.2 /etc/inetsim/inetsim.conf *
start_service dummy_udp
##
service_bind_address
#
IP address to bind services to
#
Syntax: service_bind_address <IP address>
#
Default: 127.0.0.1
#
service_bind_address 192.168.32.100
##
service_run_as_user
#
User to run services
#
Syntax: service_run_as_user <username>
#
Default: inetsim
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify


kali@kali: ~
File Actions Edit View Help
GNU nano 7.2 /etc/inetsim/inetsim.conf *
##
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
##

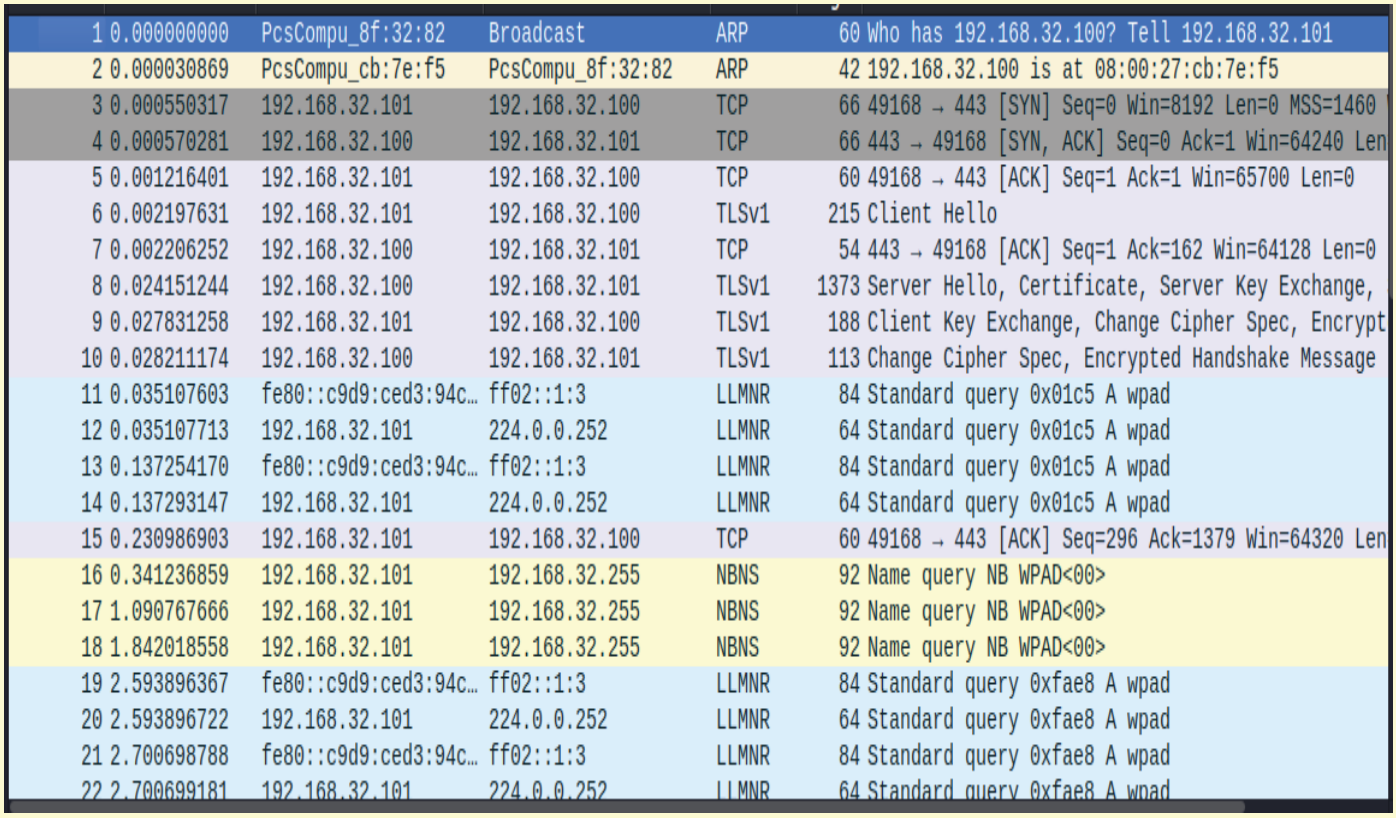
```
kali@kali: ~
File  Actions  Edit  View  Help
 GNU nano 7.2                 /etc/inetsim/inetsim.conf *

##########################################
# dns_default_domainname
#
# Default domain name to return with DNS replies
#
# 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 epicode.internal 192.168.32.100
```

- A questo punto procediamo con la verifica su Windows 7 del domain epicode.internal

```
┌──(kali㉿kali)-[~]
└─$ sudo inetsim
[sudo] password for kali:
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 1396) ===
Session ID:     1396
Listening on:   192.168.32.100
Real Date/Time: 2023-12-29 10:05:28
Fake Date/Time: 2023-12-29 10:05:28 (Delta: 0 seconds)
 Forking services...
  * dns_53_tcp_udp - started (PID 1398)
print() on closed filehandle MLOG at /usr/share/perl5/Net/DNS/Nameserve
ine 399.
print() on closed filehandle MLOG at /usr/share/perl5/Net/DNS/Nameserve
ine 399.
  * https_443_tcp - started (PID 1399)
 done.
Simulation running.
```



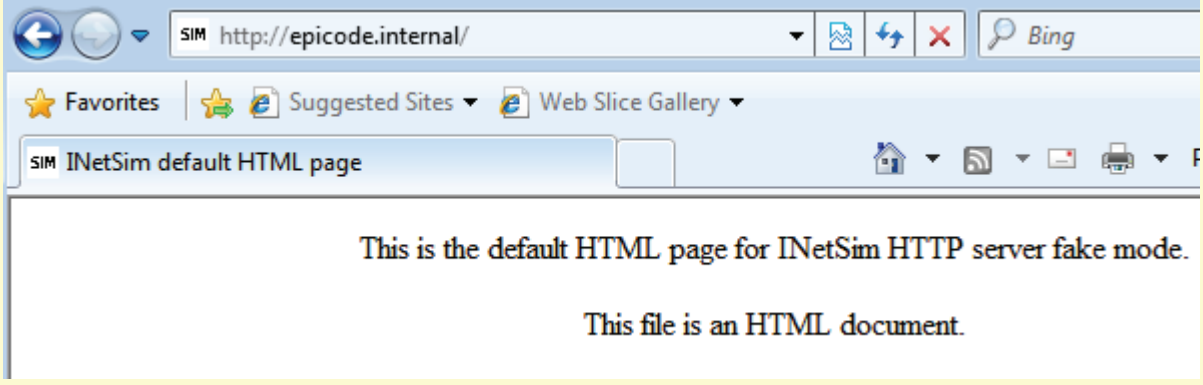
- Effettuiamo quindi la cattura dei pacchetti su Wireshark dei pacchetti inviati su HTTPS



1	0.000000000	PcsCompu_8f:32:82	Broadcast	ARP	60	Who has 192.168.32.100? Tell 192.168.32.101
2	0.000030869	PcsCompu_cb:7e:f5	PcsCompu_8f:32:82	ARP	42	192.168.32.100 is at 08:00:27:cb:7e:f5
3	0.000550317	192.168.32.101	192.168.32.100	TCP	66	49168 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460
4	0.000570281	192.168.32.100	192.168.32.101	TCP	66	443 → 49168 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len
5	0.001216401	192.168.32.101	192.168.32.100	TCP	60	49168 → 443 [ACK] Seq=1 Ack=1 Win=65700 Len=0
6	0.002197631	192.168.32.101	192.168.32.100	TLSv1	215	Client Hello
7	0.002206252	192.168.32.100	192.168.32.101	TCP	54	443 → 49168 [ACK] Seq=1 Ack=162 Win=64128 Len=0
8	0.024151244	192.168.32.100	192.168.32.101	TLSv1	1373	Server Hello, Certificate, Server Key Exchange,
9	0.027831258	192.168.32.101	192.168.32.100	TLSv1	188	Client Key Exchange, Change Cipher Spec, Encrypt
10	0.028211174	192.168.32.100	192.168.32.101	TLSv1	113	Change Cipher Spec, Encrypted Handshake Message
11	0.035107603	fe80::c9d9:ced3:94c…	ff02::1:3	LLMNR	84	Standard query 0x01c5 A wpad
12	0.035107713	192.168.32.101	224.0.0.252	LLMNR	64	Standard query 0x01c5 A wpad
13	0.137254170	fe80::c9d9:ced3:94c…	ff02::1:3	LLMNR	84	Standard query 0x01c5 A wpad
14	0.137293147	192.168.32.101	224.0.0.252	LLMNR	64	Standard query 0x01c5 A wpad
15	0.230986903	192.168.32.101	192.168.32.100	TCP	60	49168 → 443 [ACK] Seq=296 Ack=1379 Win=64320 Len
16	0.341236859	192.168.32.101	192.168.32.255	NBNS	92	Name query NB WPAD<00>
17	1.090767666	192.168.32.101	192.168.32.255	NBNS	92	Name query NB WPAD<00>
18	1.842018558	192.168.32.101	192.168.32.255	NBNS	92	Name query NB WPAD<00>
19	2.593896367	fe80::c9d9:ced3:94c…	ff02::1:3	LLMNR	84	Standard query 0xfae8 A wpad
20	2.593896722	192.168.32.101	224.0.0.252	LLMNR	64	Standard query 0xfae8 A wpad
21	2.700698788	fe80::c9d9:ced3:94c…	ff02::1:3	LLMNR	84	Standard query 0xfae8 A wpad
22	2.700699181	192.168.32.101	224.0.0.252	LLMNR	64	Standard query 0xfae8 A wpad

- A questo punto ripetiamo il tutto ma usando HTTP anzichè HTTPS .

```
# 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
```



• Ripetiamo la cattura dei pacchetti tramite Wireshark e notiamo le differenze .

```
┌──(kali㉿kali)-[~]
└─$ sudo inetsim
[sudo] password for kali:
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 1396) ===
Session ID:     1396
Listening on:   192.168.32.100
Real Date/Time: 2023-12-29 10:05:28
Fake Date/Time: 2023-12-29 10:05:28 (Delta: 0 seconds)
 Forking services...
  * dns_53_tcp_udp - started (PID 1398)
print() on closed filehandle MLOG at /usr/share/perl5/Net/DNS/Nameserve
ine 399.
print() on closed filehandle MLOG at /usr/share/perl5/Net/DNS/Nameserve
ine 399.
  * https_443_tcp - started (PID 1399)
 done.
Simulation running.
```

1	0.000000000	fe80::c9d9:ced3:94c…	ff02::1:2	DHCPv6	157	Solicit XID: 0x9d8853 CID: 000100012cfc3df8080027
2	0.960572437	192.168.32.101	192.168.32.100	TCP	66	49160 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS
3	0.960604921	192.168.32.100	192.168.32.101	TCP	66	80 → 49160 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0
4	0.961067139	192.168.32.101	192.168.32.100	TCP	60	49160 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
5	0.961067322	192.168.32.101	192.168.32.100	HTTP	327	GET /favicon.ico HTTP/1.1
6	0.961099661	192.168.32.100	192.168.32.101	TCP	54	80 → 49160 [ACK] Seq=1 Ack=274 Win=64128 Len=0
7	0.971110844	192.168.32.100	192.168.32.101	TCP	207	80 → 49160 [PSH, ACK] Seq=1 Ack=274 Win=64128 Len
8	0.972331519	192.168.32.100	192.168.32.101	HTTP	252	HTTP/1.1 200 OK (image/x-icon)
9	0.972535749	192.168.32.101	192.168.32.100	TCP	60	49160 → 80 [ACK] Seq=274 Ack=353 Win=65348 Len=0
10	0.972615944	192.168.32.101	192.168.32.100	TCP	60	49160 → 80 [FIN, ACK] Seq=274 Ack=353 Win=65348 L
11	0.972624512	192.168.32.100	192.168.32.101	TCP	54	80 → 49160 [ACK] Seq=353 Ack=275 Win=64128 Len=0

• Come possiamo vedere in HTTPS c'è la presenza del protocollo TLSv1 e dei pacchetti : NBNS e LLMNR ,che aiutano con la cifratura dei dati . Il tutto non è presente nel protocollo HTTP che mostra chiaramente tutti i pacchetti