

# REPORT DANIELE NIEDDU – ESERCIZIO 27/09/2024

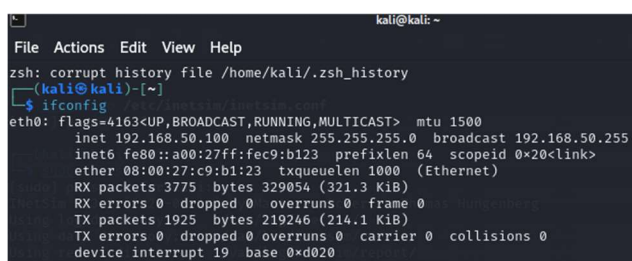
## FINE MODULO

### Svolgimento:

Per svolgere questo esercizio è stato necessario avere installato 2 macchine virtuali, una con KALI LINUX e una con WINDOWS 10. Entrambe con le impostazioni di rete su Internal (Ovvero una rete interna loro). Per far sì che fossero raggiungibili tra di loro è stato necessario impostare gli indirizzi IP in modo statico togliendo l'impostazione DHCP, questo per far sì che le due macchine fossero sulla stessa rete.

Per fare questo su Kali è servito andare nella shell e digitare i seguenti comandi:

```
auto eth0
iface eth0 inet static
address 192.168.50.100
netmask 255.255.255.0
gateway 192.168.50.1
```



```
kali@kali: ~
File Actions Edit View Help
zsh: corrupt history file /home/kali/.zsh_history
(kali@kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.50.100 netmask 255.255.255.0 broadcast 192.168.50.255
    inet6 fe80::a00:27ff:fec9:b123 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:c9:b1:23 txqueuelen 1000 (Ethernet)
    RX packets 3775 bytes 329054 (321.3 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1925 bytes 219246 (214.1 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 19 base 0xd020
```

Nella immagine sopra vediamo la configurazione andata a buon fine.

Per windows invece è bastato andare nella configurazione “grafica” (Vedi foto sotto)

#### Modifica impostazioni IP

Manuale

#### IPv4

Attivato

Indirizzo IP

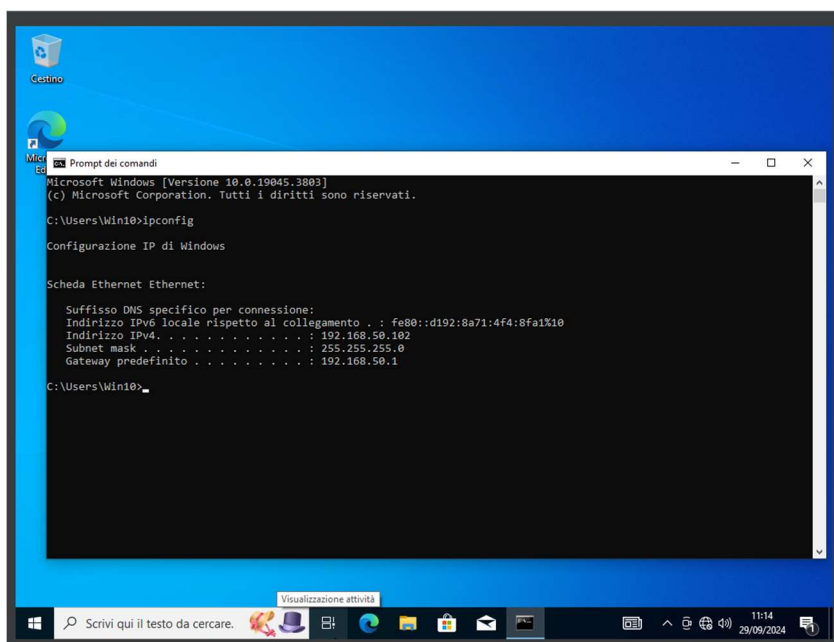
192.168.50.102

Lunghezza prefisso subnet

24

Gateway

192.168.50.1



Fatto questo, per far sì che le due macchine fossero entrambe raggiungibili tra di loro, abbiamo dovuto impostare le regole firewall di Windows 10 che bloccava la comunicazione.

Una volta settata la regola, con i permessi di poter “far passare” i pacchetti provenienti da Linux, il ping è andato a buon fine.

A questo punto l'esercizio chiedeva di simulare un'architettura client server in cui un client Windows10 richiede tramite web browser una risorsa all'hostname internal.epicode che risponde all'indirizzo 192.168.50.100 ovvero Kali.

Come prima cosa, da Kali, ho configurato INETSIM (Simulatore di servizi internet) attivando il Server DNS (Un server DNS è un servizio che traduce i nomi di dominio leggibili dagli esseri umani (come *www.google.com*) in indirizzi IP (come *142.250.74.78*), che sono utilizzati dai computer per identificarsi e comunicare tra loro su una rete, come Internet.) e HTTPS (Un server HTTPS è un tipo di server web che utilizza il protocollo HTTPS (HyperText Transfer Protocol Secure) per trasmettere dati in modo sicuro tra il server e il browser dell'utente. A differenza del protocollo HTTP standard, HTTPS crittografa le informazioni scambiate, garantendo che i dati siano protetti da intercettazioni o manipolazioni da parte di terzi.).

```
# 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
```

```
#####
# service_bind_address
#
# IP address to bind services to
#
# Syntax: service_bind_address <IP address>
#
# Default: 127.0.0.1
#
service_bind_address 192.168.50.100
```

Il parametro `service_bind_address` su Inetsim serve per specificare l'indirizzo IP locale (In questo caso è sempre l'indirizzo IP di KALI) a cui i servizi simulati dal programma stesso devono essere associati (bind).

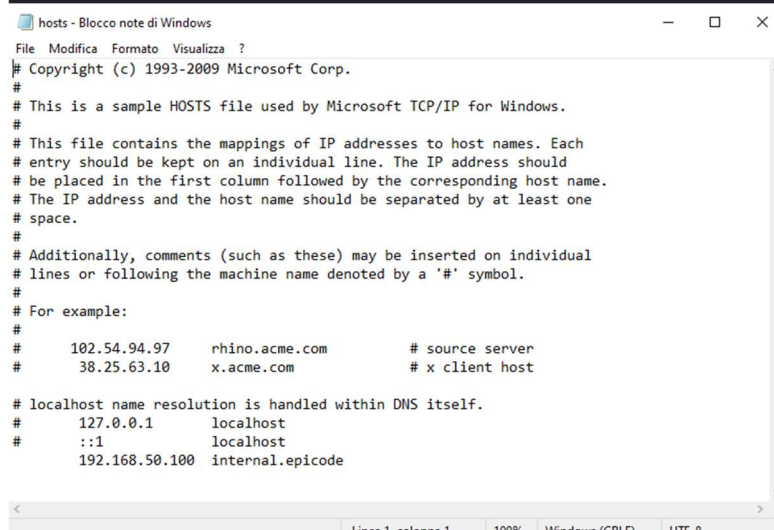
```
#####
# dns_static
#
# Static mappings for DNS
#
# Syntax: dns_static <fqdn hostname> <IP address>
#
# Default: none
#
#dns_static www.foo.com 10.10.10.10
#dns_static ns1.foo.com 10.70.50.30
#dns_static ftp.bar.net 10.10.20.30
dns_static internal.epicode 192.168.50.100
```

Ho inoltre impostato il DNS STATICO inserendo HOSTNAME (internal.epicode) e indirizzo IP che corrisponde sempre a KALI.

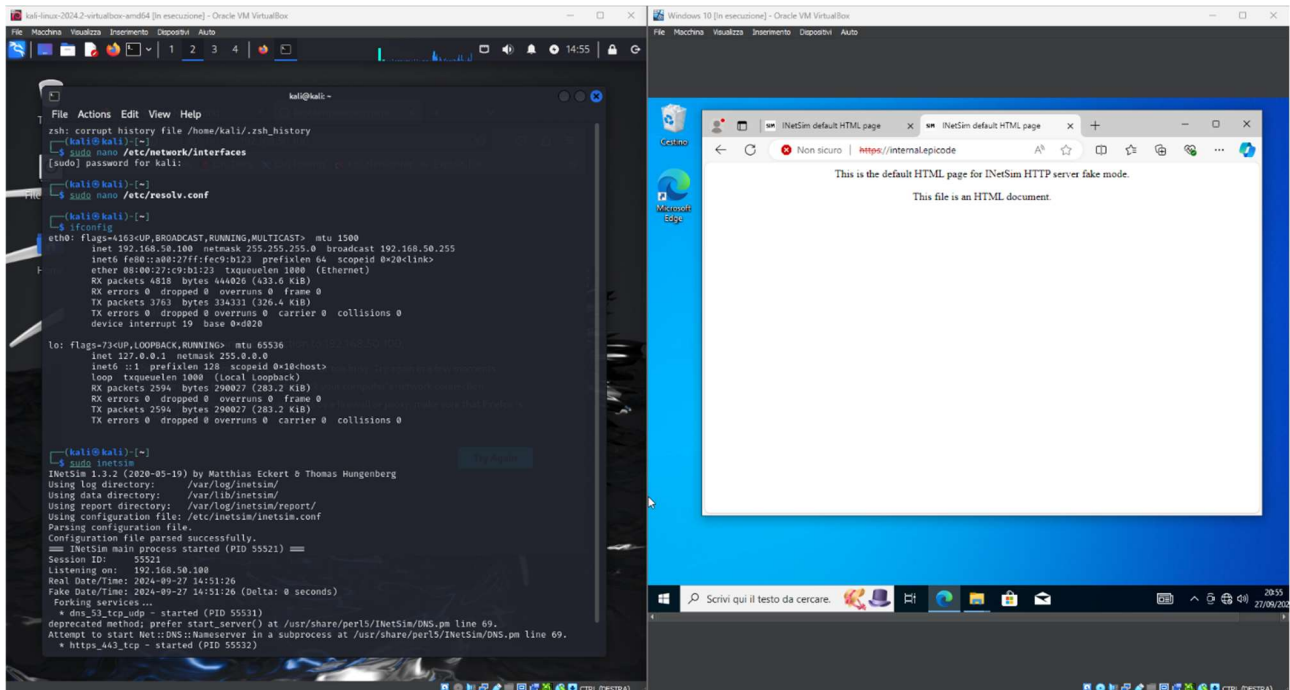
Fatto questo ho raggiunto correttamente, tramite browser web di Windows, il "sito fake" digitando `HTTPS://192.168.50.100`.

Per raggiungere il sito all'indirizzo internal.epicode, son dovuto entrare nel file "hosts\*" di windows e inserire l'indirizzo ip associato a internal.epicode. (Vedi foto)

\*Il file hosts di Windows è utilizzato per associare nomi di dominio a indirizzi IP specifici, bypassando i server DNS configurati sulla rete. In altre parole, il file hosts permette di fare risoluzione locale dei nomi, ovvero di stabilire manualmente come un nome di dominio debba essere tradotto in un indirizzo IP, prima che il sistema interroghi un server DNS.



Nello screenshot sotto vediamo come, una volta avviato Inetsim su KALI, da Web Browser di Windows10 è stato correttamente raggiunto l'indirizzo <https://internal.epicode>



Arrivati a questo punto dell'esercizio, viene richiesto di intercettare la comunicazione con Wireshark, evidenziando i MAC address di sorgente e destinazione ed il contenuto della richiesta HTTPS e in seguito di ripetere l'esercizio, sostituendo il server HTTPS, con un server HTTP intercettandone nuovamente il traffico, evidenziando le differenze tra il traffico in HTTP ed il traffico in HTTPS.

Di seguito vediamo due screenshots delle analisi fatte da wireshark, sia per l'HTTPS (Prima foto) sia per l'HTTP (Seconda foto).

No.	Time	Source	Destination	Protocol	Length	Info
7	0.003103494	192.168.50.102	192.168.50.100	DNS	74	Standard query 0xe3a2 HTTPS www.bing.com
8	0.003111021	192.168.50.100	192.168.50.102	ICMP	102	Destination unreachable (Port unreachable)
9	0.025039719	192.168.50.102	192.168.50.100	DNS	74	Standard query 0xc96c A www.bing.com
10	0.063929830	192.168.50.102	192.168.50.100	DNS	93	Standard query 0x2655 A displaycatalog.mp.microsoft.com
11	1.015959782	192.168.50.102	192.168.50.100	TCP	68	59444 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256
12	1.015980509	192.168.50.100	192.168.50.102	TCP	68	443 → 59444 [SYN, ACK] Seq=0 Ack=1 Win=32120 Len=0 MSS=
13	1.016667609	192.168.50.102	192.168.50.100	TCP	62	59444 → 443 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
14	1.020683325	192.168.50.102	192.168.50.100	TCP	1516	59444 → 443 [ACK] Seq=1 Ack=1 Win=2102272 Len=1460 [TCP
15	1.020699456	192.168.50.102	192.168.50.100	TCP	56	443 → 59444 [ACK] Seq=1 Ack=1461 Win=31872 Len=0
16	1.020752487	192.168.50.102	192.168.50.100	TLSv1.3	593	Client Hello (SNI=internal.epicode)
17	1.020758034	192.168.50.100	192.168.50.102	TCP	56	443 → 59444 [ACK] Seq=1 Ack=1998 Win=31872 Len=0
18	1.038648394	192.168.50.102	192.168.50.100	TCP	68	59445 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256
19	1.038667588	192.168.50.100	192.168.50.102	TCP	68	443 → 59445 [SYN, ACK] Seq=0 Ack=1 Win=32120 Len=0 MSS=
20	1.039971284	192.168.50.102	192.168.50.100	TCP	62	59445 → 443 [ACK] Seq=1 Ack=1 Win=262656 Len=0
21	1.045509097	192.168.50.100	192.168.50.102	TLSv1.3	1477	Server Hello, Change Cipher Spec, Application Data, App
22	1.045628787	192.168.50.102	192.168.50.100	TCP	1516	59445 → 443 [ACK] Seq=1 Ack=1 Win=262656 Len=1460 [TCP
23	1.045639093	192.168.50.100	192.168.50.102	TCP	56	443 → 59445 [ACK] Seq=1 Ack=1461 Win=31872 Len=0
24	1.045688507	192.168.50.102	192.168.50.100	TLSv1.3	657	Client Hello (SNI=internal.epicode)
25	1.045693381	192.168.50.102	192.168.50.100	TCP	56	443 → 59445 [ACK] Seq=1 Ack=2062 Win=31872 Len=0
26	1.046582129	192.168.50.102	192.168.50.100	TLSv1.3	86	Change Cipher Spec, Application Data
27	1.047078348	192.168.50.102	192.168.50.100	TCP	62	59444 → 443 [FIN, ACK] Seq=2028 Ack=1422 Win=2100736 Le

Frame 16: 593 bytes on wire (4744 bits), 593 bytes captured  
Linux cooked capture v1  
Packet type: Unicast to us (0)  
Link-layer address type: Ethernet (1)  
Link-layer address length: 6  
Source: PCSSystemtec\_bc:17:14 (08:00:27:bc:17:14)  
Unused: 0000  
Protocol: IPv4 (0x0800)  
Internet Protocol Version 4, Src: 192.168.50.102, Dst: 192.  
Transmission Control Protocol, Src Port: 59444, Dst Port: 4  
[2 Reassembled TCP Segments (1997 bytes): #14(1460), #16(53)  
Transport Layer Security



No.	Time	Source	Destination	Protocol	Length	Info
7	0.004625266	192.168.50.102	192.168.50.100	DNS	74	Standard query 0x71a3 A www.bing.com
8	0.004636859	192.168.50.100	192.168.50.102	ICMP	102	Destination unreachable (Port unreachable)
9	0.024056572	192.168.50.102	192.168.50.100	DNS	74	Standard query 0x0124 A www.bing.com
10	0.024074892	192.168.50.100	192.168.50.102	ICMP	102	Destination unreachable (Port unreachable)
11	0.027166934	192.168.50.102	192.168.50.100	DNS	74	Standard query 0x0124 A www.bing.com
12	0.027178998	192.168.50.100	192.168.50.102	ICMP	102	Destination unreachable (Port unreachable)
13	0.027553085	192.168.50.102	192.168.50.100	DNS	74	Standard query 0x0124 A www.bing.com
14	1.598440673	192.168.50.102	192.168.50.100	TCP	68	59465 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK
15	1.598461102	192.168.50.100	192.168.50.102	TCP	68	80 → 59465 [SYN, ACK] Seq=0 Ack=1 Win=32120 Len=0 MSS=1460 S
16	1.599034615	192.168.50.102	192.168.50.100	TCP	62	59465 → 80 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
17	1.608787825	192.168.50.102	192.168.50.100	HTTP	560	GET / HTTP/1.1
18	1.608851810	192.168.50.100	192.168.50.102	TCP	56	80 → 59465 [ACK] Seq=1 Ack=505 Win=31872 Len=0
19	1.620738512	192.168.50.100	192.168.50.102	TCP	206	80 → 59465 [PSH, ACK] Seq=1 Ack=505 Win=31872 Len=150 [TCP S
20	1.622608529	192.168.50.100	192.168.50.102	HTTP	314	HTTP/1.1 200 OK (text/html)
21	1.623169766	192.168.50.102	192.168.50.100	TCP	62	59465 → 80 [ACK] Seq=505 Ack=410 Win=2101760 Len=0
22	1.636977090	192.168.50.102	192.168.50.100	TCP	68	59466 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK
23	1.636997448	192.168.50.100	192.168.50.102	TCP	68	80 → 59466 [SYN, ACK] Seq=0 Ack=1 Win=32120 Len=0 MSS=1460 S
24	1.637878076	192.168.50.102	192.168.50.100	TCP	62	59466 → 80 [ACK] Seq=1 Ack=1 Win=262656 Len=0
25	1.640605652	192.168.50.102	192.168.50.100	DNS	96	Standard query 0x9ca1 A nav-edge.smartscreen.microsoft.com
26	1.640622318	192.168.50.100	192.168.50.102	ICMP	124	Destination unreachable (Port unreachable)
27	1.641691294	192.168.50.102	192.168.50.100	DNS	96	Standard query 0x54a8 A nav-edge.smartscreen.microsoft.com

Frame 17: 560 bytes on wire (4480 bits), 560 bytes captured on interface	0000	00 00 00 01 00 06 08 00	27 bc 17 14 00 00 08 00	.....!.....
Linux cooked capture v1	0010	45 00 02 20 c4 9d 40 00	80 06 4e 1f c0 a8 32 66	E...@...N...2f
Packet type: Unicast to us (0)	0020	c0 a8 32 64 e8 49 00 50	04 ff d6 1b fc 19 41 f2	..2d I P .....A..
Link-layer address type: Ethernet (1)	0030	50 18 20 14 b0 2b 00 00	47 45 54 20 2f 20 48 54	P...+...GET / HT
Link-layer address length: 6	0040	54 50 2f 31 2e 31 0d 0a	48 6f 73 74 3a 20 69 6e	TP/1.1...Host: in
Source: PCSSystemtec_bc:17:14 (08:00:27:bc:17:14)	0050	74 65 72 6e 61 6c 2e 65	70 69 63 6f 64 65 0d 0a	ternal.e picode..
Unused: 0000	0060	43 6f 6e 6e 65 63 74 69	6f 6e 3a 20 6b 65 65 70	Connecti on: keep
Protocol: IPv4 (0x0800)	0070	2d 61 6c 69 76 65 0d 0a	43 61 63 68 65 2d 43 6f	-alive . Cache-Co
Internet Protocol Version 4, Src: 192.168.50.102, Dst: 192.168.50.80	0080	6e 74 72 6f 6c 3a 20 6d	61 78 2d 61 67 65 3d 30	ntrol: m ax-age=0
Transmission Control Protocol, Src Port: 59465, Dst Port: 80	0090	0d 0a 55 70 67 72 61 64	65 2d 49 6e 73 65 63 75	--Upgrad e-Insecu
Hypertext Transfer Protocol	00a0	72 65 2d 52 65 71 75 65	73 74 73 3a 20 31 0d 0a	re-Reqe sts: 1..
	00b0	55 73 65 72 2d 41 67 65	6e 74 3a 20 4d 6f 7a 69	User-Age nt: Mozi
	00c0	6c 6c 61 2f 35 2e 30 20	28 57 69 6e 64 6f 77 73	lla/5.0 (Windows
	00d0	20 4e 54 20 31 30 2e 30	3b 20 57 69 6e 36 34 3b	NT 10.0 ; Win64;
	00e0	20 78 36 34 29 20 41 70	70 6c 65 57 65 62 4b 69	x64) Ap pleWebKi
	00f0	74 2f 35 33 37 2e 33 36	20 28 4b 48 54 4d 4c 2c	t/537.36 (KHTML,
	0100	20 6c 69 6b 65 20 47 65	63 6b 6f 29 20 43 68 72	like Ge cko) Chr
	0110	6f 6d 65 2f 31 32 38 2e	30 2e 30 2e 30 20 53 61	ome/128.0.0.0 Sa
	0120	66 61 72 69 2f 35 33 37	2e 33 36 20 45 64 67 2f	fari/537.36 Edg/
	0130	31 32 38 2e 30 2e 30 2e	30 0d 0a 41 63 63 65 70	128.0.0.0 .Accep
	0140	74 3a 20 74 65 78 74 2f	68 74 6d 6c 2c 61 70 70	t: text/ html,app
	0150	6c 69 63 61 74 69 6f 6e	2f 78 68 74 6d 6c 2b 78	lication /xhtml+x

Su entrambi possiamo notare la classica sequenza 3 Way Handshake ovvero il modo in cui due macchine si “presentano” e preparano per parlare tra loro su Internet.

(SYN): Il computer A (*Client*) vuole iniziare una conversazione con il computer B (*Server*). Per farlo, invia un messaggio al server. Questo serve a richiedere l’inizio di una connessione. Il computer A include anche un numero di sequenza iniziale, che sarà usato per tracciare i pacchetti dati.

(SYN-ACK): Il computer B riceve il messaggio SYN e risponde con un messaggio che ha entrambi i flag SYN e ACK impostati. Il SYN conferma la ricezione della richiesta di connessione e l’ACK conferma la ricezione del numero di sequenza del client. Il computer B include anche il proprio numero di sequenza.

(ACK): Il computer A riceve il SYN-ACK dal computer B e risponde con un segmento con il flag ACK impostato, confermando la ricezione del numero di sequenza del server. A questo punto, la connessione è stabilita e i dati possono essere trasferiti tra client e server.

Confrontando il traffico HTTPS con quello HTTP, notiamo che la differenza principale è legata al livello di crittografia e visibilità dei dati trasmessi.

Per quanto riguarda il traffico HTTP notiamo che trasmette i dati in chiaro, quindi tutto il traffico può essere visto facilmente, compreso la richiesta GET, inoltre è possibile leggere il contenuto dei pacchetti in maniera facilmente comprensibile.

A differenza dell’HTTP, nell’HTTPS notiamo che viene utilizzato il protocollo TLSv1.3 (Transport Layer Security) per crittografare il traffico, ciò significa che la maggior parte del contenuto non è leggibile. Infatti i pacchetti dati veri e propri sono crittografati e appaiono come dati scritti senza senso.

Negli screenshot sotto la differenza di lettura su HTTP e HTTPS:

```

..2f.P.. ..xYK.m.
P..... <html>
<head>    <tit
le>INetS im defau
lt HTML  page</ti
tle>    < /head>
<body>    <p><
/p>    <p align
="center ">This i
s the de fault HT
ML page  for INet
Sim HTTP  server
fake mod e.</p>
<p al ign="cen
ter">Thi s file i
s an HTM L docume

```

HTTP (Foto sinistra): Riusciamo a leggere tutto il contenuto della pagina web

HTTPS (Foto destra): Non riusciamo a decifrare nulla a causa della crittografia.

```

.....
E A @ P 2f
2d 4 ~K
P      mzE9D,1
x0,      GMC
<      0+
Q A + / 1
D&W      SVG 'B
1&      B " ? :
V      y= j \ F
zY*z} # e z 6
K- Q < ~ j 0"
} N . < NN
U      Q M
~ , i ~ ?
V ?      ZZC
.....
#
h2 ht tp/1.1
.....
ZZ

```

Le foto sotto evidenziano a Sinistra il MAC ADDRESS di Windows (192.168.50.102) e a destra di KALI LINUX (192.168.50.100).

6	0.001782234	192.168.50.102	192.168.50.100	TLSv1	6
7	0.001787867	192.168.50.100	192.168.50.102	TCP	5
8	0.019410833	192.168.50.102	192.168.50.100	TCP	5
9	0.019429942	192.168.50.100	192.168.50.102	TCP	6
10	0.020223321	192.168.50.102	192.168.50.100	TCP	6
11	0.020860925	192.168.50.102	192.168.50.100	TCP	151
12	0.020871723	192.168.50.100	192.168.50.102	TCP	5
13	0.020921990	192.168.50.102	192.168.50.100	TLSv1.3	66
14	0.020928215	192.168.50.100	192.168.50.102	TCP	5
15	0.023772689	192.168.50.100	192.168.50.102	TLSv1.3	147
16	0.024991257	192.168.50.102	192.168.50.100	TLSv1.3	8
17	0.025647173	192.168.50.102	192.168.50.100	TCP	5
18	0.029572617	192.168.50.100	192.168.50.102	TCP	5
19	0.030047426	192.168.50.102	192.168.50.100	TCP	6
20	0.039927174	192.168.50.100	192.168.50.102	TLSv1.3	147
21	0.041022368	192.168.50.102	192.168.50.100	TLSv1.3	8

Frame 6: 632 bytes on wire (5056 bits), 632 bytes captured (5056 bits) on  
 Linux cooked capture v1  
 Packet type: Unicast to us (0)  
 Link-layer address type: Ethernet (1)  
 Link-layer address length: 6  
 Source: PCSSystemtec\_bc:17:14 (08:00:27:bc:17:14)  
 Unused: 0000  
 Protocol: IPv4 (0x0800)  
 Internet Protocol Version 4, Src: 192.168.50.102, Dst: 192.168.50.100  
 Transmission Control Protocol, Src Port: 62117, Dst Port: 443, Seq: 1461,  
 [2 Reassembled TCP Segments (2936 bytes): #4(1460), #6(576)]  
 Transport Layer Security

5	0.001723058	192.168.50.100	192.168.50.102	TCP	5
6	0.001782234	192.168.50.102	192.168.50.100	TLSv1	63
7	0.001787867	192.168.50.100	192.168.50.102	TCP	5
8	0.019410833	192.168.50.102	192.168.50.100	TCP	6
9	0.019429942	192.168.50.100	192.168.50.102	TCP	6
10	0.020223321	192.168.50.102	192.168.50.100	TCP	6
11	0.020860925	192.168.50.102	192.168.50.100	TCP	151
12	0.020871723	192.168.50.100	192.168.50.102	TCP	5
13	0.020921990	192.168.50.102	192.168.50.100	TLSv1.3	66
14	0.020928215	192.168.50.100	192.168.50.102	TCP	5
15	0.023772689	192.168.50.100	192.168.50.102	TLSv1.3	147
16	0.024991257	192.168.50.102	192.168.50.100	TLSv1.3	8
17	0.025647173	192.168.50.102	192.168.50.100	TCP	5
18	0.029572617	192.168.50.100	192.168.50.102	TCP	5
19	0.030047426	192.168.50.102	192.168.50.100	TCP	6
20	0.039927174	192.168.50.100	192.168.50.102	TLSv1.3	147
21	0.041022368	192.168.50.102	192.168.50.100	TLSv1.3	8

Frame 5: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on inter  
 Linux cooked capture v1  
 Packet type: Sent by us (4)  
 Link-layer address type: Ethernet (1)  
 Link-layer address length: 6  
 Source: PCSSystemtec\_c9:b1:23 (08:00:27:c9:b1:23)  
 Unused: 0000  
 Protocol: IPv4 (0x0800)  
 Internet Protocol Version 4, Src: 192.168.50.100, Dst: 192.168.50.102  
 Transmission Control Protocol, Src Port: 443, Dst Port: 62117, Seq: 1, Ack

Riepilogando, la differenza fondamentale tra HTTP e HTTPS sta nella sicurezza e nella visibilità dei dati trasmessi.

Con HTTP, tutto è in chiaro. Questo rende l'analisi molto semplice, perché puoi controllare direttamente cosa sta succedendo tra il computer e il server. Invece l'HTTPS protegge le informazioni usando la crittografia. Quindi, analizzando una connessione HTTPS non si riusciranno a vedere i dati effettivi trasmessi ma si vedranno solo una serie di pacchetti crittografati.

Per entrambi si possono comunque vedere le info tecniche, come gli indirizzi IP del client e del server, le porte usate e i certificati di sicurezza che vengono scambiati all'inizio della connessione ma il contenuto vero e proprio, cioè ciò che viene trasmesso tra il client e il server, nell'HTTPS è nascosto dietro la crittografia e non è accessibile senza la chiave giusta.