

SIMULAZIONE RETE COMPLESSA

Gentile Professore,

In riferimento all'esercizio svolto, desidero presentarvi il report dettagliato sulla simulazione di un'architettura client-server con particolare attenzione alla comunicazione attraverso protocolli HTTPS e HTTP. Di seguito riporto i dettagli dell'esercizio e le spiegazioni relative ai risultati ottenuti.

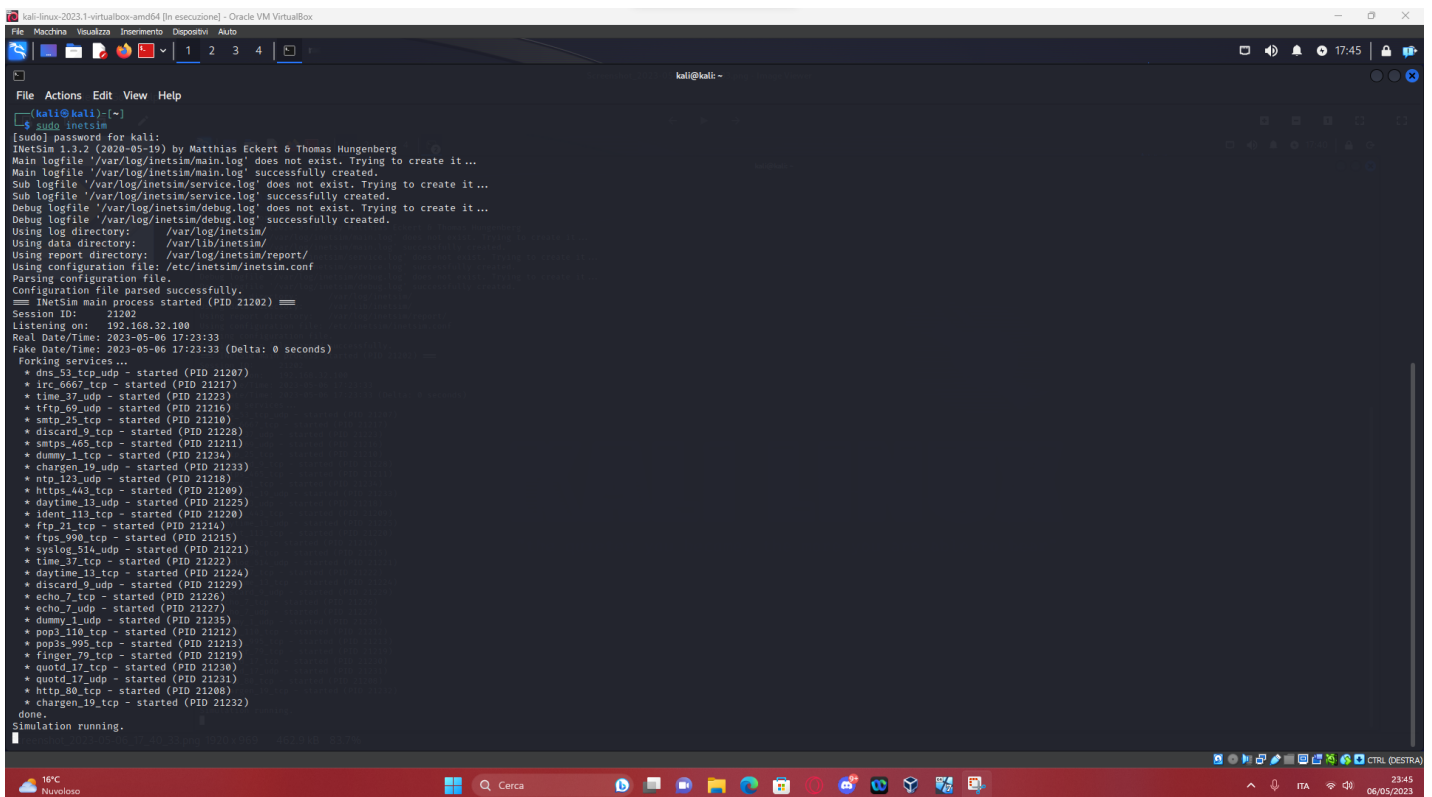
Requisiti e servizi utilizzati:

Kali Linux □ IP 192.168.32.100

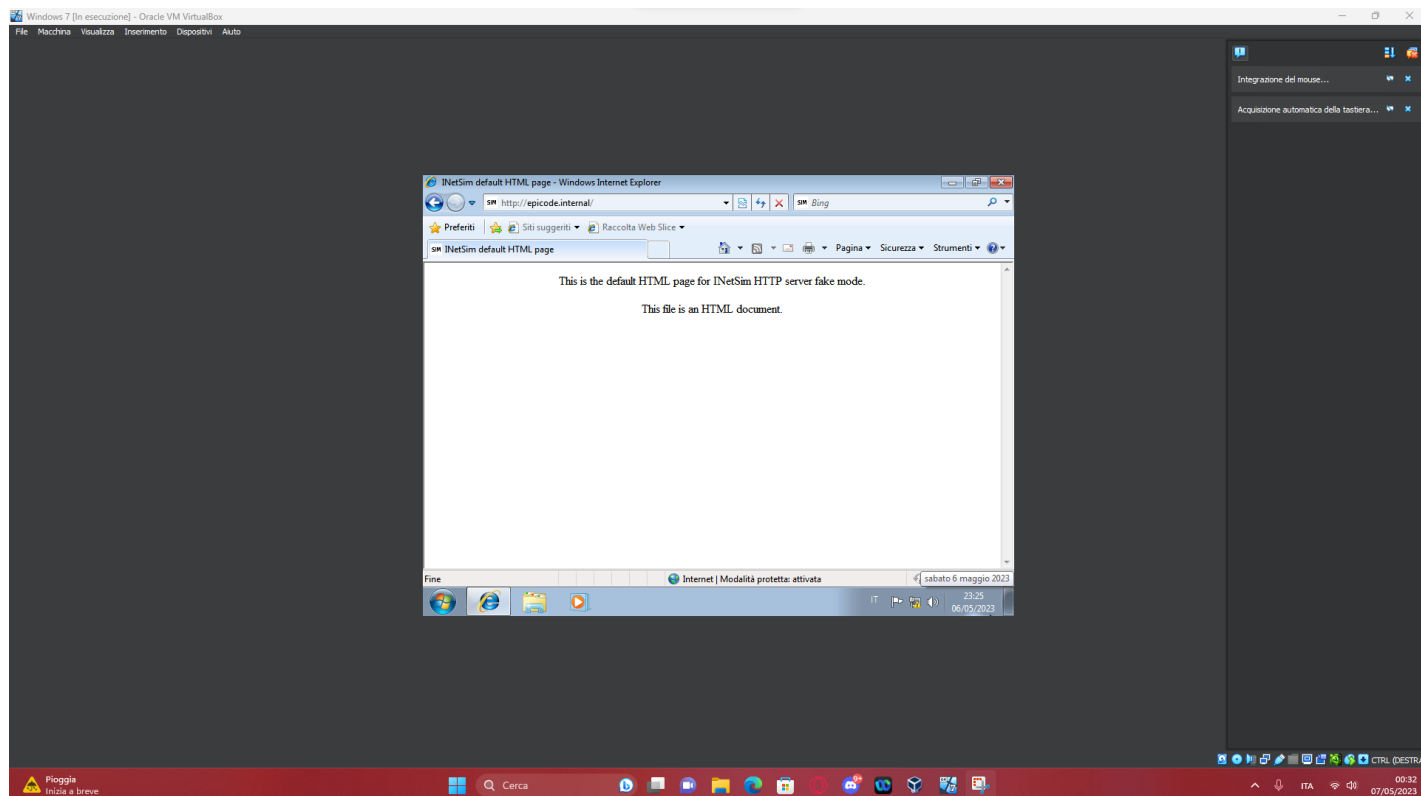
Windows 7 □ IP 192.168.32.101

Server HTTPS: attivo

Servizio DNS per risoluzione nomi di dominio: attivo



```
kali@kali:~$ sudo inetsim
[sudo] password for kali:
InetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Main logfile '/var/log/inetsim/main.log' does not exist. Trying to create it...
Main logfile '/var/log/inetsim/main.log' successfully created.
Sub logfile '/var/log/inetsim/service.log' does not exist. Trying to create it...
Sub logfile '/var/log/inetsim/service.log' successfully created.
Debug logfile '/var/log/inetsim/debug.log' does not exist. Trying to create it...
Debug logfile '/var/log/inetsim/debug.log' successfully created.
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 21202) ==
Session ID: 21202
Listening on: 192.168.32.100
Real Date/Time: 2023-05-06 17:23:33
Fake Date/Time: 2023-05-06 17:23:33 (Delta: 0 seconds)
Forking services ...
* dns_53_tcp_udp - started (PID 21207)
* irc_6667_tcp - started (PID 21215)
* time_37_udp - started (PID 21223)
* tftp_69_udp - started (PID 21216)
* smtp_25_tcp - started (PID 21210)
* discard_9_tcp - started (PID 21228)
* ssmtps_465_tcp - started (PID 21211)
* dummy_1_tcp - started (PID 21234)
* chargen_19_udp - started (PID 21233)
* ntp_123_udp - started (PID 21218)
* https_443_tcp - started (PID 21209)
* daytime_13_udp - started (PID 21225)
* ident_113_tcp - started (PID 21220)
* ftp_21_tcp - started (PID 21214)
* ftps_990_tcp - started (PID 21215)
* syslog_514_udp - started (PID 21221)
* time_37_tcp - started (PID 21222)
* daytime_13_tcp - started (PID 21224)
* discard_9_udp - started (PID 21229)
* echo_7_tcp - started (PID 21226)
* echo_7_udp - started (PID 21227)
* dummy_1_udp - started (PID 21235)
* pop3_110_tcp - started (PID 21212)
* pop3s_995_tcp - started (PID 21213)
* finger_79_tcp - started (PID 21219)
* quotd_17_tcp - started (PID 21230)
* quotd_17_udp - started (PID 21231)
* http_80_tcp - started (PID 21208)
* chargen_19_tcp - started (PID 21232)
done.
Simulation running.
```



Modifiche effettuate e configurazione del laboratorio virtuale:

Prima di procedere con la simulazione dell'architettura client-server e l'intercettazione del traffico di rete, sono state apportate alcune modifiche per garantire il corretto funzionamento e la comunicazione tra i dispositivi coinvolti. Di seguito sono riportate le operazioni effettuate:

Modifica degli indirizzi IP:

Gli indirizzi IP dei dispositivi sono stati adeguatamente configurati per consentire la corretta comunicazione all'interno del laboratorio virtuale. Kali Linux è stato assegnato l'indirizzo IP 192.168.32.100, mentre Windows 7 ha ricevuto l'indirizzo IP 192.168.32.101.

Verifica della comunicazione:

Prima di procedere con l'esercizio, è stata effettuata una verifica della connettività tra Kali Linux e Windows 7 per garantire che fossero sulla stessa rete e in grado di scambiarsi pacchetti di rete correttamente. Questa verifica preliminare è stata fondamentale per assicurare una comunicazione senza problemi tra il client e il server durante la simulazione.

Utilizzo di Inetsim per il laboratorio virtuale:

Per creare un ambiente di laboratorio virtuale controllato, è stato utilizzato Inetsim, uno strumento che permette di emulare diversi servizi di rete. Nello specifico, Kali Linux è stato configurato per agire come un server **DNS**, che a sua volta ha avviato i servizi **HTTP** richiesti. Inetsim ha consentito di simulare in modo preciso i servizi richiesti e di analizzare il traffico di rete generato durante le comunicazioni client-server.

È importante sottolineare che l'utilizzo di **Inetsim** ha consentito di emulare i servizi HTTP richiesti, consentendo così l'analisi dettagliata del traffico di rete generato dai dispositivi coinvolti nel laboratorio virtuale.

Descrizione dell'esercizio:

The screenshot displays the Wireshark network protocol analyzer interface. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. The main window is titled 'HTTPS.pcapng' and shows a list of captured packets. The selected packet is a TCP segment with sequence number 49188, acknowledgment number 443, and length 0. The packet details pane on the right shows the following layers:

- Ethernet II, Src: PcsCompu_94:45:f1 (08:00:27:94:45:f1), Dst: PcsCompu_c7:e1:36 (08:00:27:c7:e1:36)
- Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100
- Transmission Control Protocol, Src Port: 49188, Dst Port: 443, Seq: 0, Len: 0

The packet bytes pane on the right shows the raw data in hexadecimal and ASCII. The status bar at the bottom indicates that 104 packets were captured, with 12 displayed (11.5%) and 1 marked (1.0%).

Kali Linux 2023.1 - virtualbox-amd64 [in esecuzione] - Oracle VM VirtualBox

File Macchina Visualizza Impostazioni Dispositivi Auto

1 2 3 4

HTTP.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.port==80

No.	Time	Source	Destination	Protocol	Length	Info
2	7.617834669	192.168.32.101	192.168.32.100	TCP	66	49196 → 80 [SYN] Seq=0 Win=0 Len=0 MSS=1460 WS=4 SACK_PERM
3	7.617858992	192.168.32.100	192.168.32.101	TCP	66	80 → 49196 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
4	7.618236587	192.168.32.101	192.168.32.100	TCP	60	49196 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
5	7.618600193	192.168.32.101	192.168.32.100	HTTP	359	GET / HTTP/1.1
6	7.618606874	192.168.32.100	192.168.32.101	TCP	54	80 → 49196 [ACK] Seq=1 Ack=306 Win=64128 Len=0
7	7.627315494	192.168.32.100	192.168.32.101	TCP	204	80 → 49196 [PSH, ACK] Seq=1 Ack=306 Win=64128 Len=150 [TCP segment of a reassembled PDU]
8	7.628353258	192.168.32.100	192.168.32.101	HTTP	312	HTTP/1.1 200 OK (text/html)
9	7.628761231	192.168.32.101	192.168.32.100	TCP	60	49196 → 80 [ACK] Seq=306 Ack=410 Win=65292 Len=0
10	7.628761313	192.168.32.101	192.168.32.100	TCP	60	49196 → 80 [FIN, ACK] Seq=306 Ack=410 Win=65292 Len=0
11	7.628782954	192.168.32.100	192.168.32.101	TCP	54	80 → 49196 [ACK] Seq=410 Ack=307 Win=64128 Len=0

Frame 2: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface eth0, id 0

Ethernet II, Src: PcsCompu_94:45:f1 (08:00:27:94:45:f1), Dst: PcsCompu_c7:e1:36 (08:00:27:c7:e1:36)

Internet Protocol Version 4, Src: 192.168.32.101, Dst: 192.168.32.100

Transmission Control Protocol, Src Port: 49196, Dst Port: 80, Seq: 0, Len: 0

Ethernet (eth), 14 bytes

Packets: 13 - Displayed: 10 (76.9%) - Dropped: 0 (0.0%)

Profile: Default

16°C Tompesta

Cerca

01:37 07/05/2023

Kali Linux 2023.1 - virtualbox-amd64 [in esecuzione] - Oracle VM VirtualBox

File Macchina Visualizza Impostazioni Dispositivi Auto

1 2 3 4

HTTPS.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

arp

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_94:45:f1	Broadcast	ARP	60	who has 192.168.32.100? Tell 192.168.32.101
2	0.000017771	PcsCompu_c7:e1:36	PcsCompu_94:45:f1	ARP	42	192.168.32.100 is at 08:00:27:c7:e1:36
11	0.257455924	PcsCompu_94:45:f1	Broadcast	ARP	60	who has 192.168.32.1? Tell 192.168.32.101
13	1.052399762	PcsCompu_94:45:f1	Broadcast	ARP	60	who has 192.168.32.1? Tell 192.168.32.101
14	2.052942101	PcsCompu_94:45:f1	Broadcast	ARP	60	who has 192.168.32.1? Tell 192.168.32.101
22	6.047192090	PcsCompu_94:45:f1	Broadcast	ARP	60	who has 192.168.32.1? Tell 192.168.32.101
23	6.556235805	PcsCompu_94:45:f1	Broadcast	ARP	60	who has 192.168.32.1? Tell 192.168.32.101
24	7.556280954	PcsCompu_94:45:f1	Broadcast	ARP	60	who has 192.168.32.1? Tell 192.168.32.101
42	11.780959089	PcsCompu_94:45:f1	Broadcast	ARP	60	who has 192.168.32.1? Tell 192.168.32.101
43	12.558075239	PcsCompu_94:45:f1	Broadcast	ARP	60	who has 192.168.32.1? Tell 192.168.32.101
44	13.558268502	PcsCompu_94:45:f1	Broadcast	ARP	60	who has 192.168.32.1? Tell 192.168.32.101
52	17.488688548	PcsCompu_94:45:f1	Broadcast	ARP	60	who has 192.168.32.1? Tell 192.168.32.101
53	18.061814949	PcsCompu_94:45:f1	Broadcast	ARP	60	who has 192.168.32.1? Tell 192.168.32.101
54	19.061469660	PcsCompu_94:45:f1	Broadcast	ARP	60	who has 192.168.32.1? Tell 192.168.32.101
71	23.224246448	PcsCompu_94:45:f1	Broadcast	ARP	60	who has 192.168.32.1? Tell 192.168.32.101
72	24.064266478	PcsCompu_94:45:f1	Broadcast	ARP	60	who has 192.168.32.1? Tell 192.168.32.101
73	25.064670232	PcsCompu_94:45:f1	Broadcast	ARP	60	who has 192.168.32.1? Tell 192.168.32.101
81	28.933606037	PcsCompu_94:45:f1	Broadcast	ARP	60	who has 192.168.32.1? Tell 192.168.32.101
82	29.567863960	PcsCompu_94:45:f1	Broadcast	ARP	60	who has 192.168.32.1? Tell 192.168.32.101
83	30.567453222	PcsCompu_94:45:f1	Broadcast	ARP	60	who has 192.168.32.1? Tell 192.168.32.101

Frame 2: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface eth0, id 0

Ethernet II, Src: PcsCompu_94:45:f1 (08:00:27:94:45:f1), Dst: PcsCompu_94:45:f1 (08:00:27:94:45:f1)

Destination: PcsCompu_94:45:f1 (08:00:27:94:45:f1)

Source: PcsCompu_c7:e1:36 (08:00:27:c7:e1:36)

Type: ARP (0x0806)

Address Resolution Protocol (reply)

Ethernet (eth), 14 bytes

Packets: 104 - Displayed: 20 (19.2%) - Marked: 1 (1.0%) - Dropped: 0 (0.0%)

Profile: Default

16°C Tompesta

Cerca

01:09 07/05/2023

Nell'ambiente di laboratorio virtuale, è stata simulata un'architettura client-server in cui il client, con indirizzo IP **192.168.32.101**, ha richiesto una risorsa tramite un web browser. La richiesta è stata inviata all'hostname "epicode.internal", corrispondente all'indirizzo IP del server, ossia **192.168.32.100**.

Nella **prima fase** dell'esercizio, è stata effettuata un'intercettazione del traffico con l'ausilio di **Wireshark**, al fine di evidenziare gli **indirizzi MAC** di sorgente e destinazione, nonché il contenuto della richiesta **HTTPS**.

La comunicazione è avvenuta attraverso il protocollo **HTTPS (HyperText Transfer Protocol Secure)**, noto per garantire la sicurezza e la privacy dei dati trasmessi tra client e server. Grazie alla crittografia dei dati, **HTTPS** protegge le informazioni durante la trasmissione.

Nell'analisi del traffico catturato con **Wireshark**, è stato possibile rilevare gli **indirizzi MAC** di sorgente e destinazione coinvolti nella comunicazione. Inoltre, si è notato che il contenuto della richiesta **HTTPS** era crittografato, pertanto non era direttamente leggibile. È importante sottolineare che solo il client e il server sono in grado di interpretare correttamente i dati crittografati.

Nella **seconda fase** dell'esercizio, il server **HTTPS** è stato sostituito con un server **HTTP**, e nuovamente è stato catturato il traffico con Wireshark per evidenziare le differenze rispetto alla comunicazione precedente in **HTTPS**.

La principale differenza tra **HTTP** ed **HTTPS** risiede nella sicurezza e nella crittografia dei dati. Mentre **HTTPS** garantisce la crittografia tramite una connessione **SSL/TLS**, **HTTP** invia i dati in chiaro, **senza crittografia**. Questa differenza comporta un livello di vulnerabilità superiore per il protocollo **HTTP**, che rende più agevole l'intercettazione e la lettura dei dati sensibili da parte di un potenziale attaccante.

Durante l'intercettazione del traffico **HTTP** con **Wireshark**, si è notato che i dati erano trasmessi in chiaro, senza crittografia, e quindi erano direttamente leggibili. La mancanza di crittografia nel protocollo **HTTP** rende i dati trasmessi in chiaro e quindi direttamente leggibili. Questa situazione apre la porta ad attacchi di intercettazione, iniezione di contenuti malevoli e compromissione dell'integrità dei dati.

Un attaccante in grado di intercettare il traffico **HTTP** potrebbe ottenere informazioni sensibili, come password, dati personali o informazioni di pagamento. Inoltre, potrebbe anche manipolare i dati trasmessi durante la comunicazione **HTTP**, ad esempio inserendo codice dannoso o modificando i parametri delle richieste.

L'assenza di crittografia nel protocollo **HTTP** rende più semplice per un attaccante analizzare e manipolare i dati durante la comunicazione. Pertanto, è fondamentale utilizzare **HTTPS** quando si richiede una maggiore sicurezza e privacy dei dati trasmessi su una rete.

Per garantire la sicurezza dei dati durante la comunicazione web, **HTTPS** offre una soluzione crittografata che protegge le informazioni scambiate tra client e server, offrendo una maggiore protezione contro attacchi di intercettazione, iniezione di contenuti malevoli e compromissione dei dati.

In conclusione, l'utilizzo del protocollo **HTTPS** è fortemente consigliato quando si richiede una comunicazione sicura e protetta su una rete. La crittografia offerta da **HTTPS** garantisce la confidenzialità, l'integrità e l'autenticità dei dati trasmessi, fornendo una maggiore protezione rispetto all'utilizzo di **HTTP**.