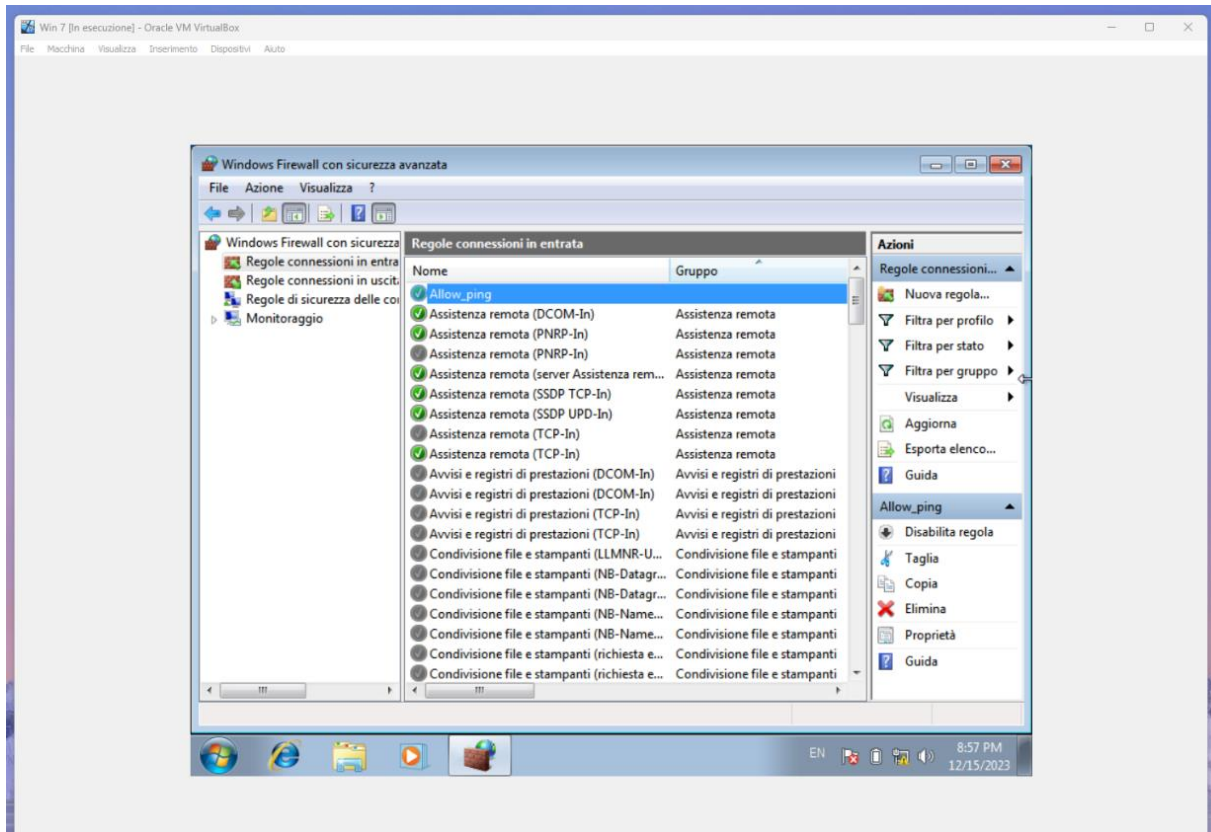
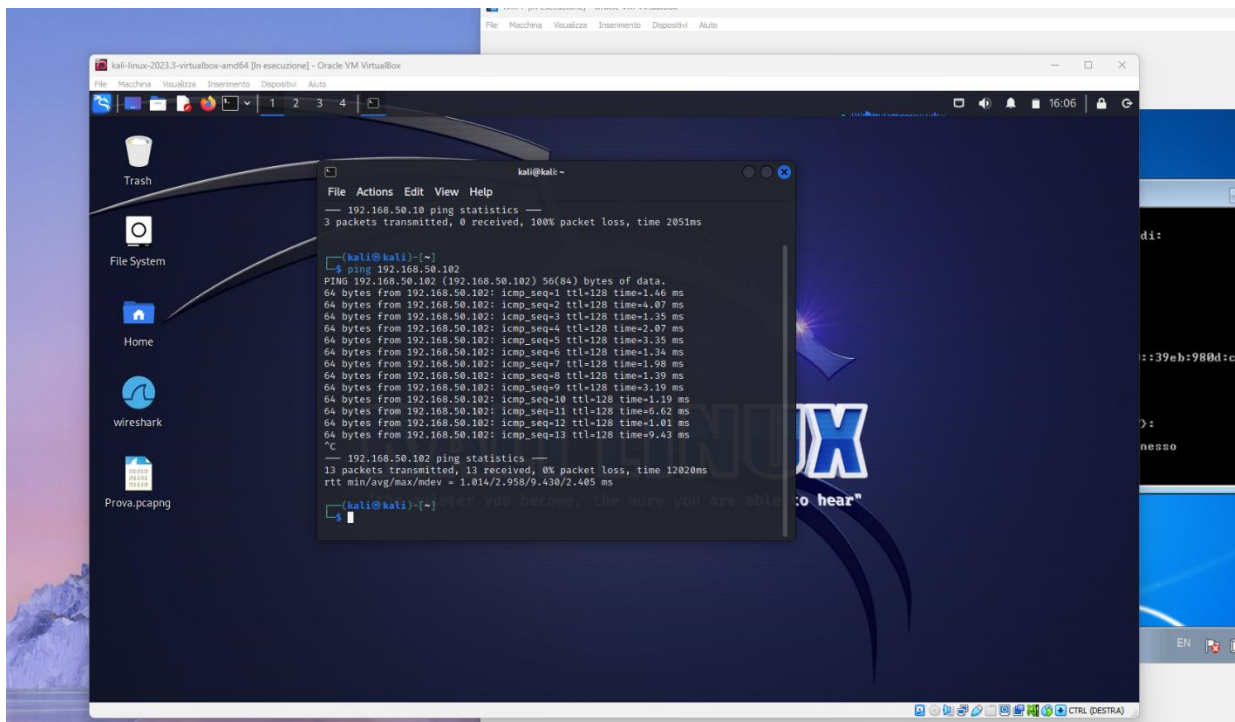


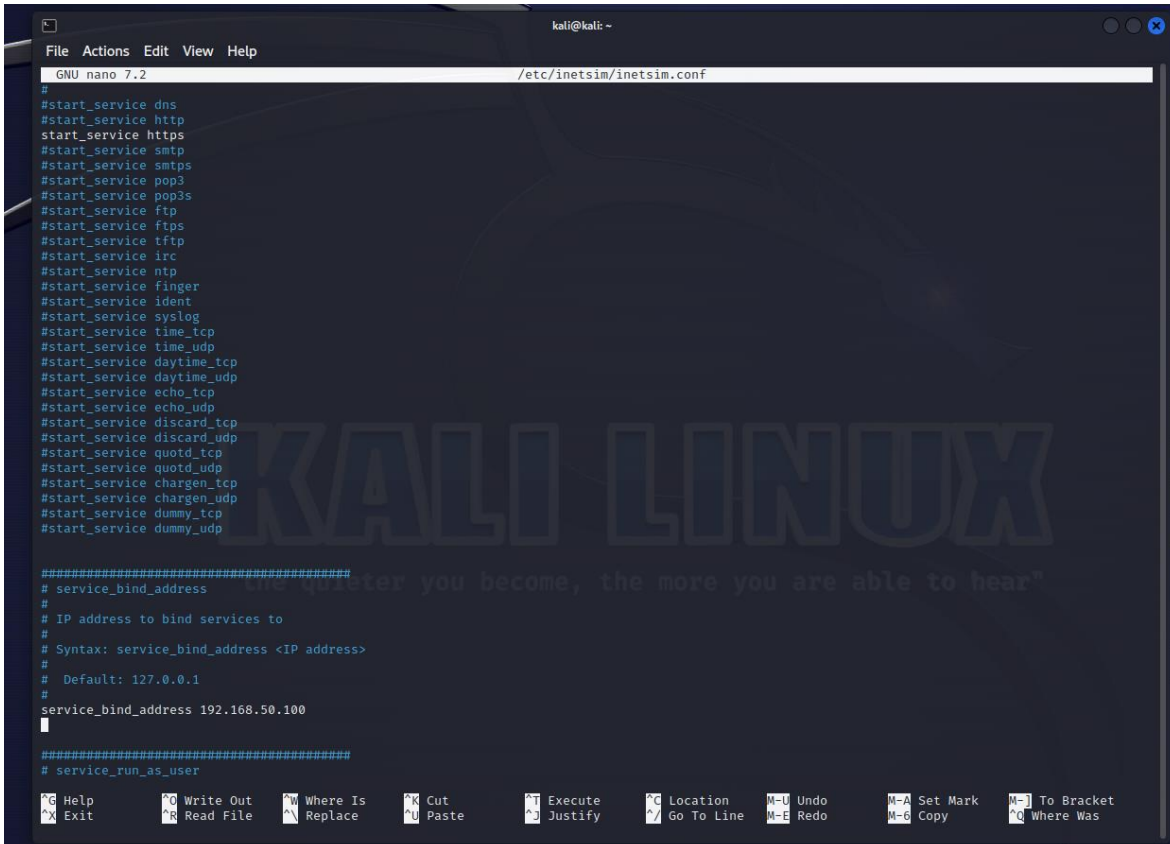
Configurazione di una policy sul firewall windows; Configurare policy per permettere il ping da macchina Linux acWindows 7 nel nostro laboratorio



Ping Kali to Wn7



Configurazione Inetsim Kali. Aggiunta # ai servizi tranne https, in modo da “abilitare” solo quel servizio. Inserisco Ip macchina Kali in “service_bind_address” e lo abilito togliendo il #. Inserisco questo Ip perché questo è l’indirizzo che ha “abilitato” inetsim

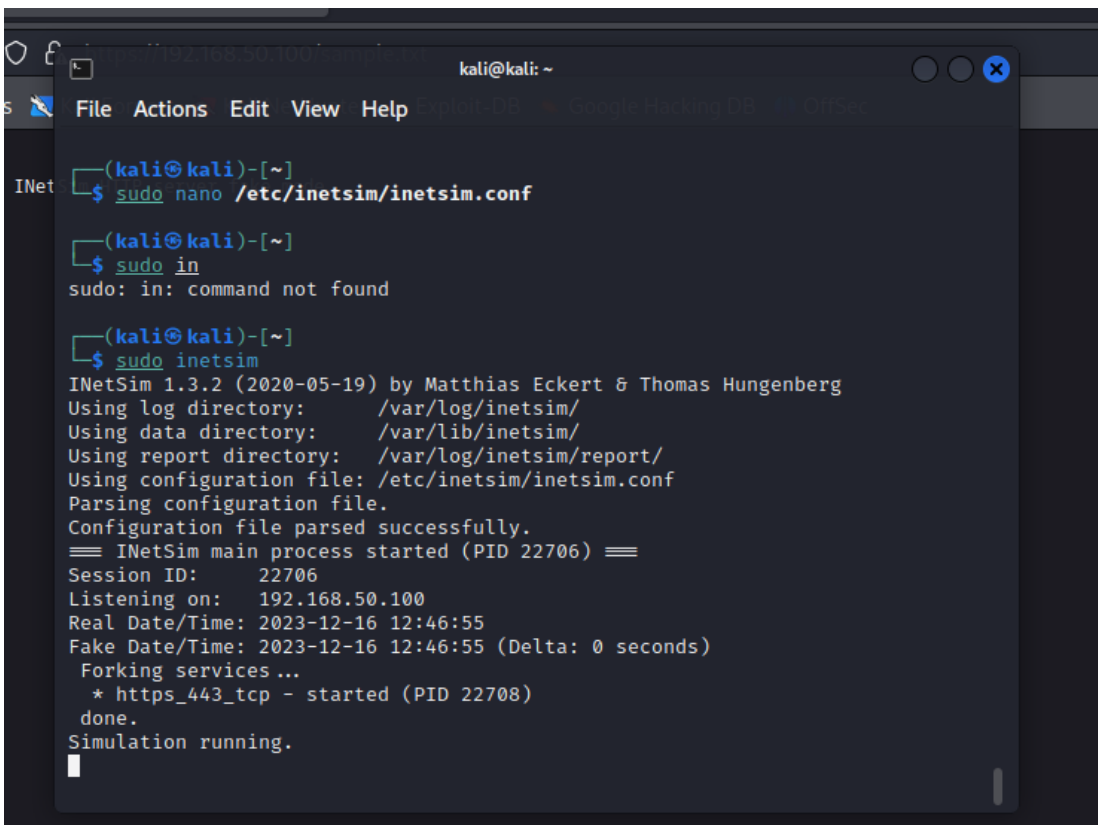


```
GNU nano 7.2 /etc/inetsim/inetsim.conf
#
#start_service dns
#start_service http
start_service https
#start_service smtp
#start_service smtps
#start_service pop3
#start_service pop3s
#start_service ftp
#start_service ftps
#start_service tftp
#start_service irc
#start_service ntp
#start_service finger
#start_service ident
#start_service syslog
#start_service time_tcp
#start_service time_udp
#start_service daytime_tcp
#start_service daytime_udp
#start_service echo_tcp
#start_service echo_udp
#start_service discard_tcp
#start_service discard_udp
#start_service quotd_tcp
#start_service quotd_udp
#start_service chargen_tcp
#start_service chargen_udp
#start_service dummy_tcp
#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 192.168.50.100

#####
# service_run_as_user
```

Sudo inetsim running. il processo è partito, collegamento porta 443. In ascolto su 192.168.50.100 (ip kali)

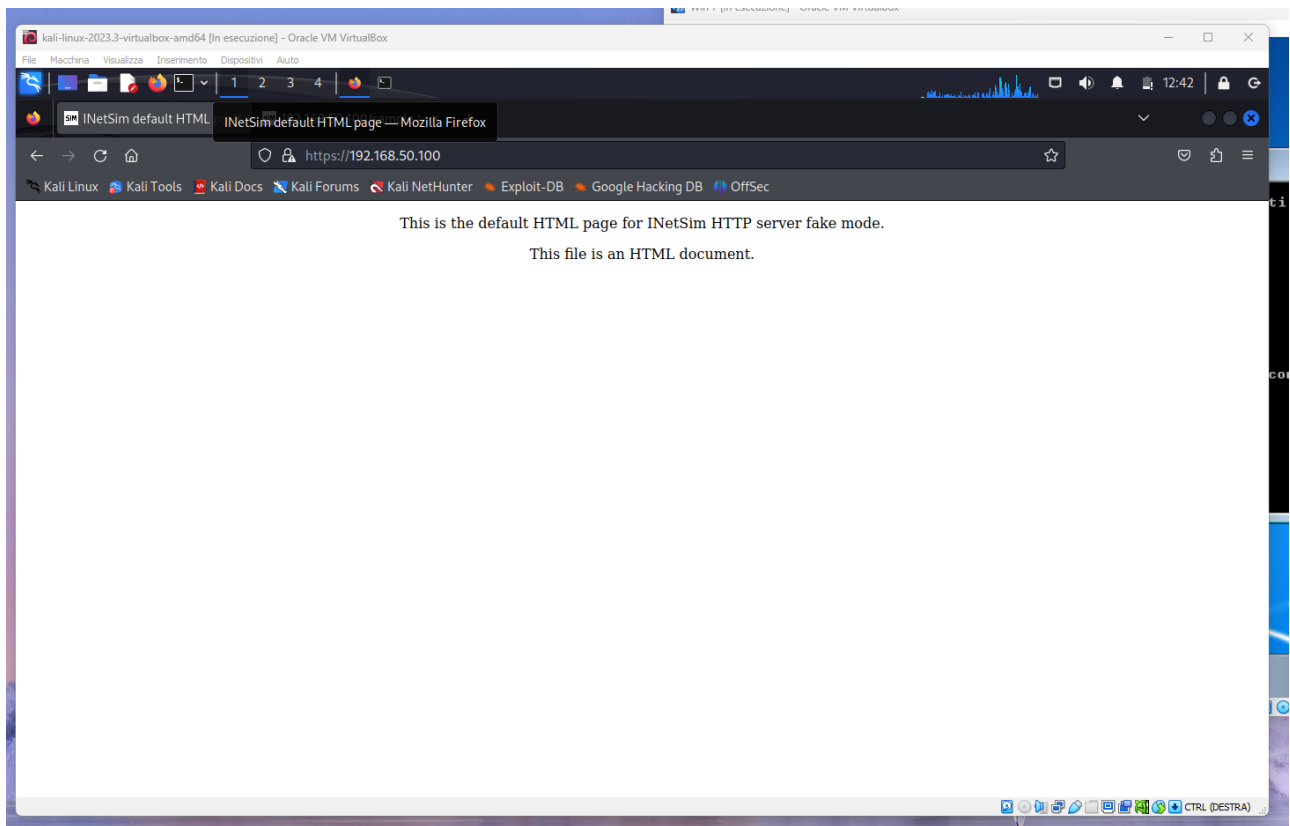


```
(kali@kali)-[~]
$ sudo nano /etc/inetsim/inetsim.conf

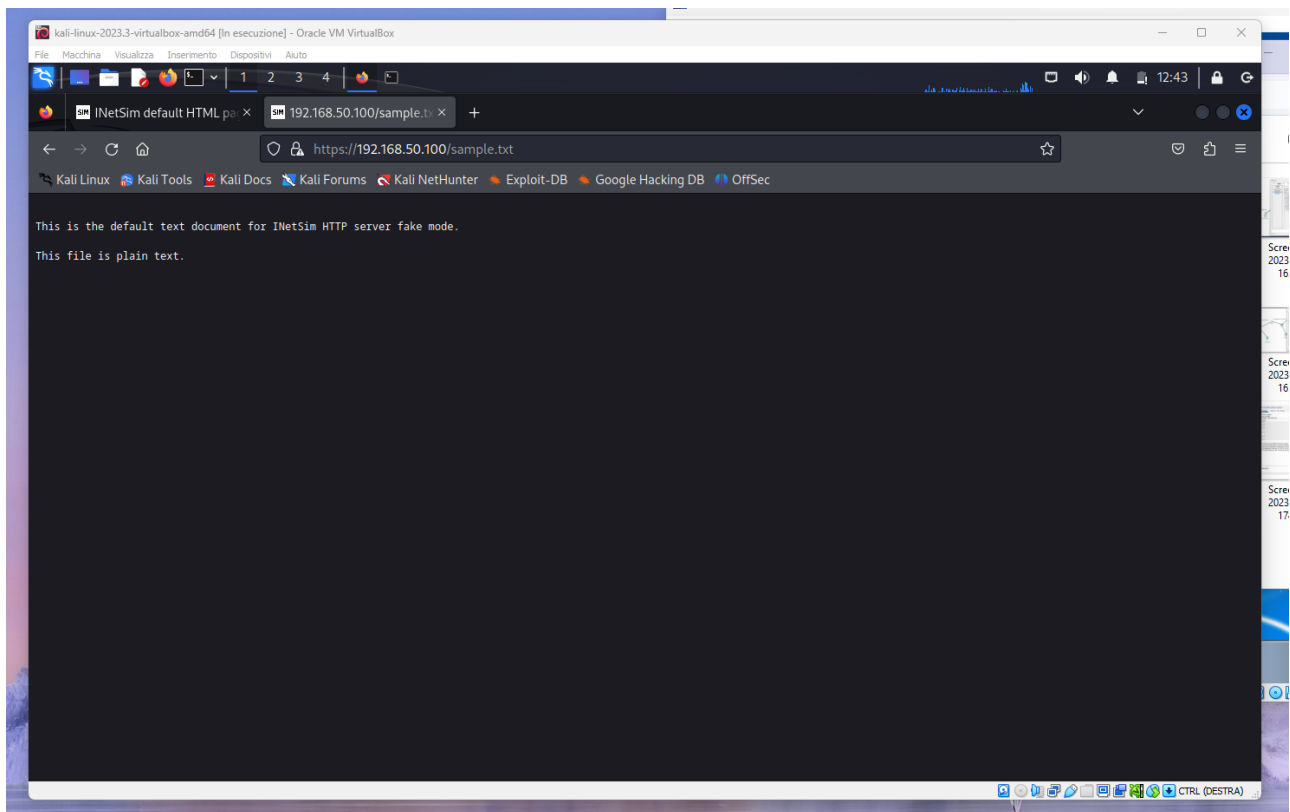
(kali@kali)-[~]
$ sudo in
sudo: in: command not found

(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 22706) ===
Session ID: 22706
Listening on: 192.168.50.100
Real Date/Time: 2023-12-16 12:46:55
Fake Date/Time: 2023-12-16 12:46:55 (Delta: 0 seconds)
Forking services...
* https_443_tcp - started (PID 22708)
done.
Simulation running.
```

Pagina HTML Inetsim. Https

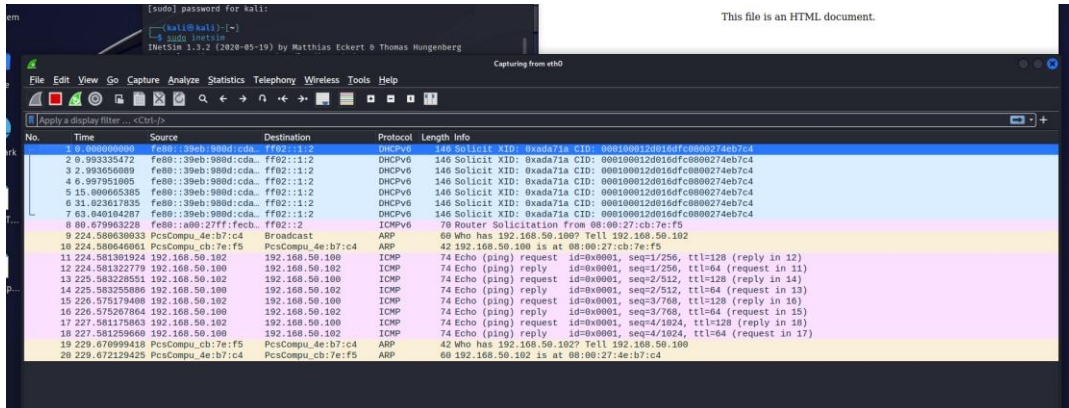


Pagina sample.txt - Https



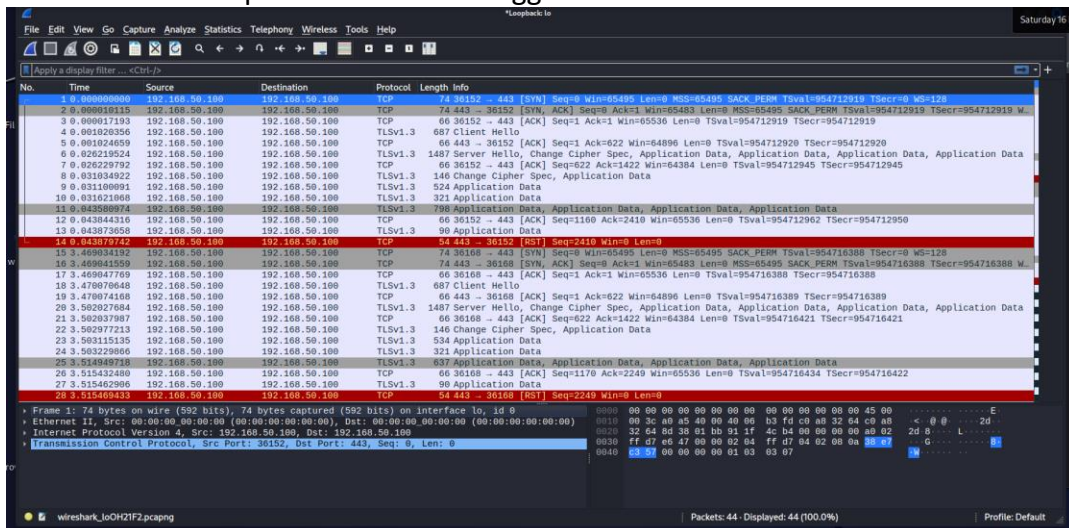
Wireshark attiva su NIC eth0 per vedere pacchetto ARP tra le macchine e il Ping riuscito. La richiesta ARP viene utilizzata per associare un indirizzo MAC ad un indirizzo IP di un host su una LAN.

Da dx a sx: Il numero del pacchetto catturato, il tempo di arrivo del pacchetto in secondi, dal momento dell'avvio della cattura, indirizzo sorgente, ovvero l'IP di chi invia il pacchetto, la destinazione, ovvero l'IP di chi riceve il pacchetto. In questo caso ARP è una richiesta broadcast, verrà quindi inviata come ci suggerisce anche Wireshark all'indirizzo di broadcast della rete. Dimensione del pacchetto e informazioni di carattere generali, come richieste o dettagli.

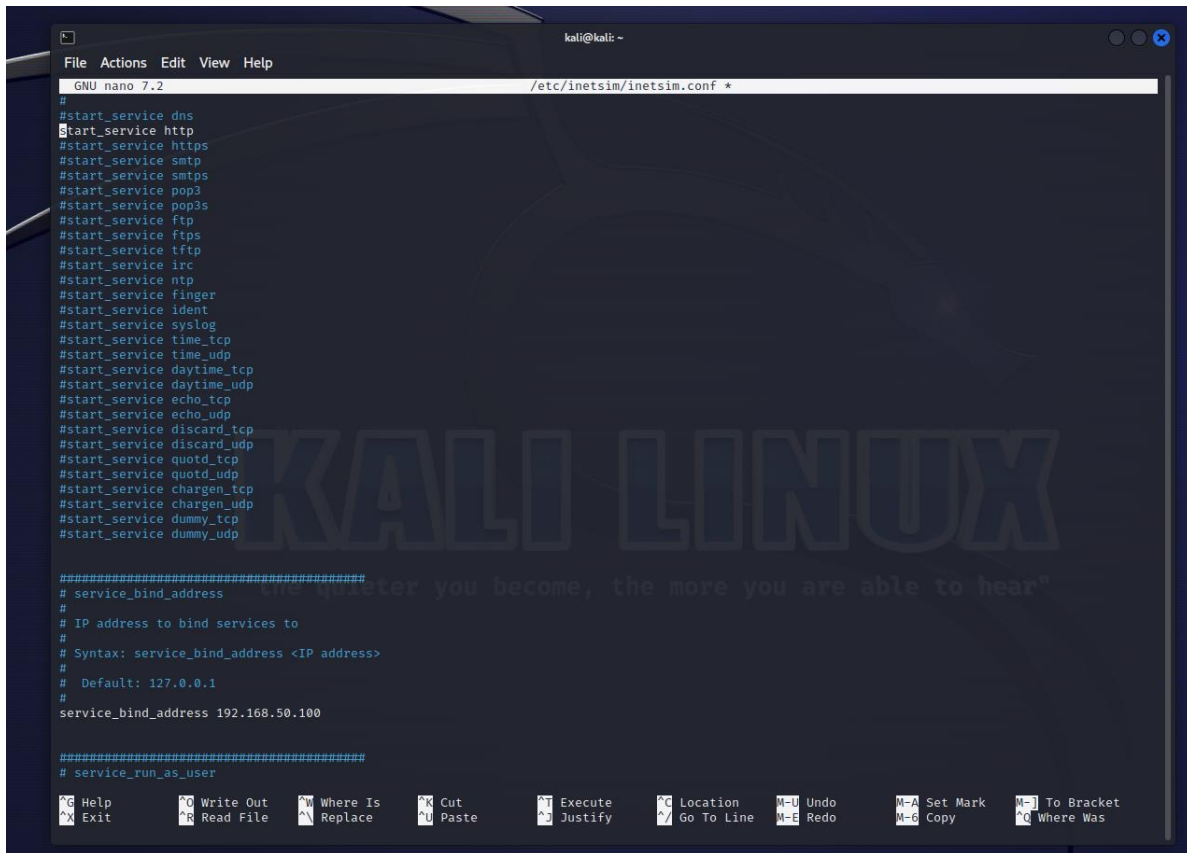


Wireshark attiva su NIC loopback per vedere sessione TCP e sequenza SYN, SYN ACK, ACK. In questo caso da riga 1 a riga 14 è la sessione TCP. TCP stabilisce un canale dedicato tra gli hosts che devono prendere parte ad una comunicazione. Per farlo, si seguono 3 passi uno dopo l'altro, questo processo prende il nome di «**three-way-handshake**», ovvero stretta di mano in tre fasi.

1. Il client (36152) che inizia la connessione invia un pacchetto TCP al server (443) destinatario con il flag SYN abilitato ed un numero di sequenza casuale (SYN è un flag del protocollo TCP presente anche nell'header). In questo caso Seq=0
2. Il server (443) risponde inviando al client (36152) un pacchetto con i flag SYN e ACK abilitati, ed un altro numero di sequenza (Seq) casuale, sempre 0 in questo caso. ACK sarà uguale al precedente Seq + 1. In questo caso ACK=1 visto che la Seq precedente era 0.
3. Il client (36152) completa la sincronizzazione inviando un pacchetto ACK, ed inviando i numeri Seq, Ack, come fatto dal server. ACK = 1. Anche qui la precedente Seq era 0.
4. Il "ClientHello" è un messaggio inviato dal client durante la fase iniziale dell'handshake TLS (Transport Layer Security) quando si stabilisce una connessione sicura con un server.
5. Il server risponde con un messaggio "ServerHello"



Configurazione Inetsim Kali. Aggiunta # ai servizi tranne http, in modo da “abilitare” solo quel servizio. Inserisco Ip macchina Kali in “service_bind_address” e lo abilito togliendo il #. Inserisco questo Ip perché questo è l’indirizzo che ha “abilitato” inetsim.

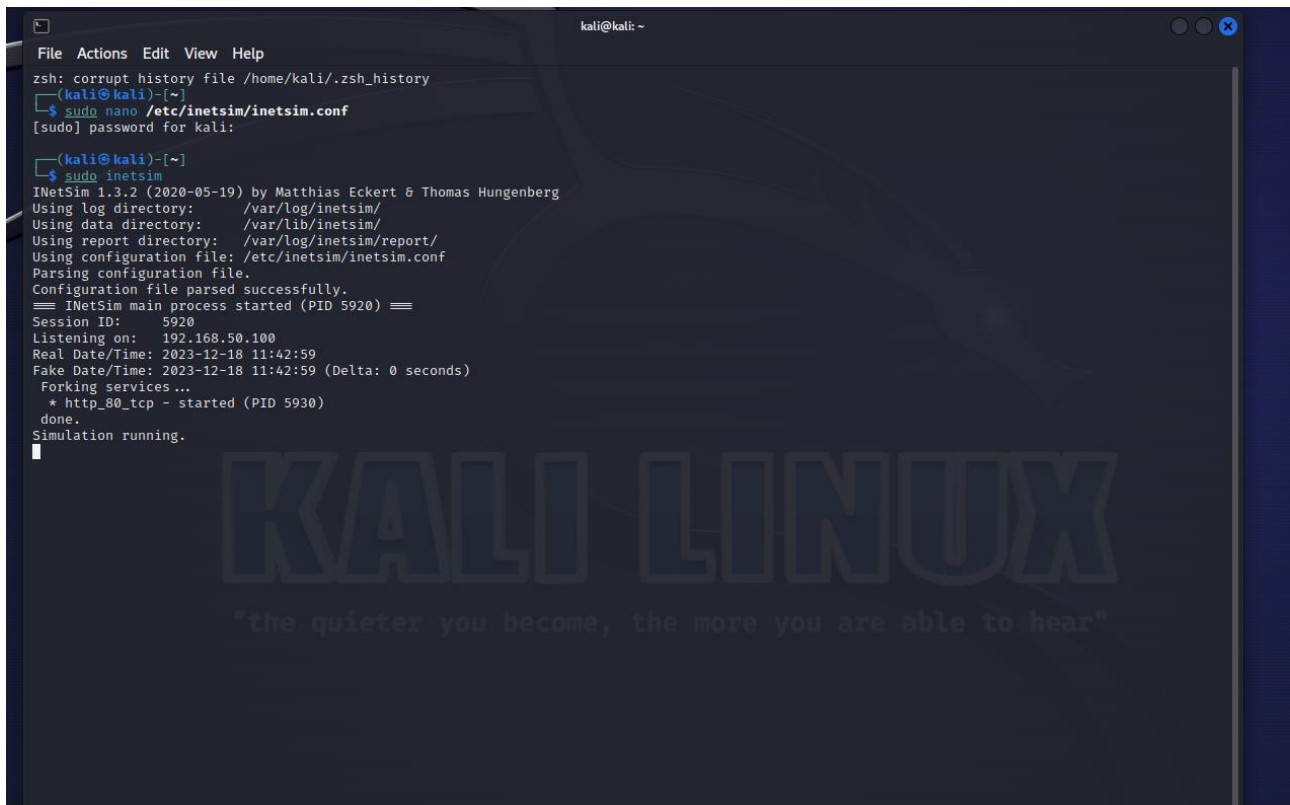


```
GNU nano 7.2 /etc/inetsim/inetsim.conf *
#
#start_service dns
#start_service http
#start_service https
#start_service smtp
#start_service smtps
#start_service pop3
#start_service pop3s
#start_service ftp
#start_service ftps
#start_service tftp
#start_service irc
#start_service ntp
#start_service finger
#start_service ident
#start_service syslog
#start_service time_tcp
#start_service time_udp
#start_service daytime_tcp
#start_service daytime_udp
#start_service echo_tcp
#start_service echo_udp
#start_service discard_tcp
#start_service discard_udp
#start_service quotd_tcp
#start_service quotd_udp
#start_service chargen_tcp
#start_service chargen_udp
#start_service dummy_tcp
#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 192.168.50.100

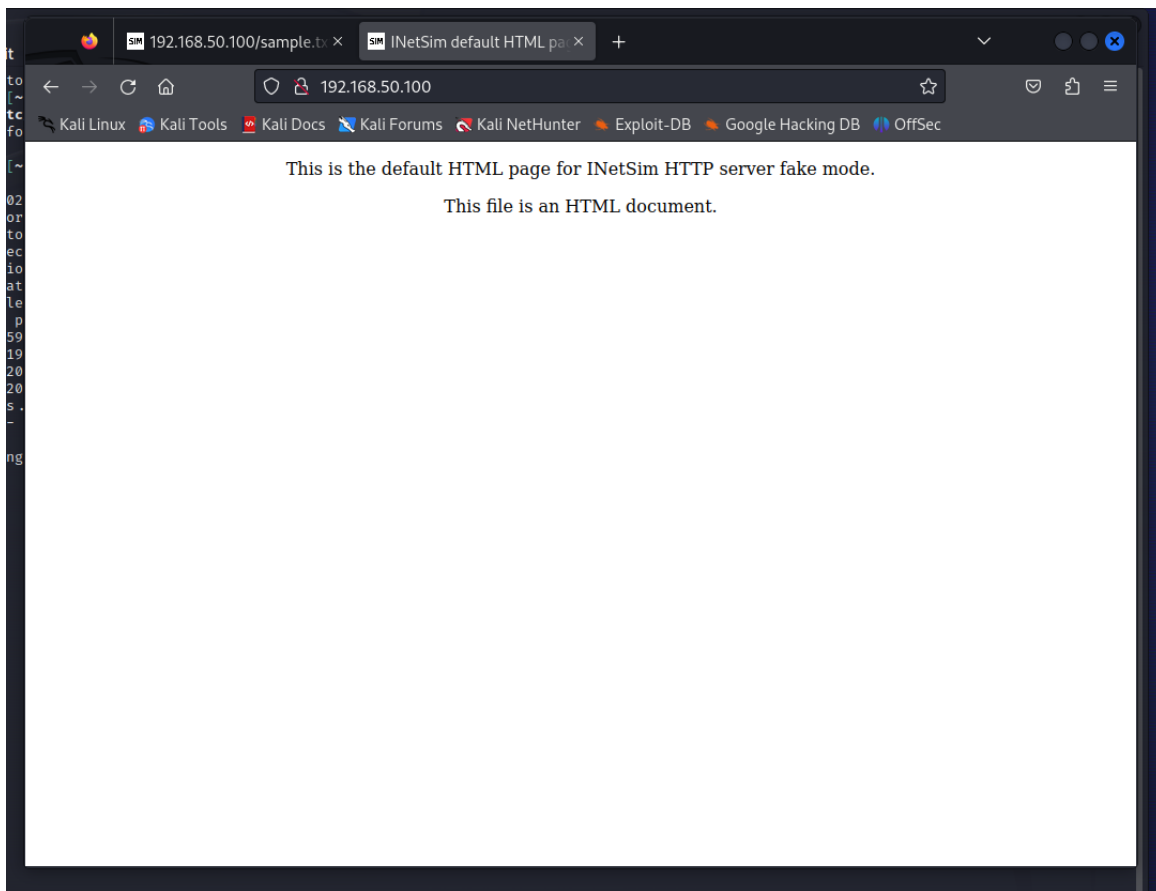
#####
# service_run_as_user
```

Sudo inetsim running. il processo è partito, collegamento porta 80. In ascolto su 192.168.50.100 (ip kali)

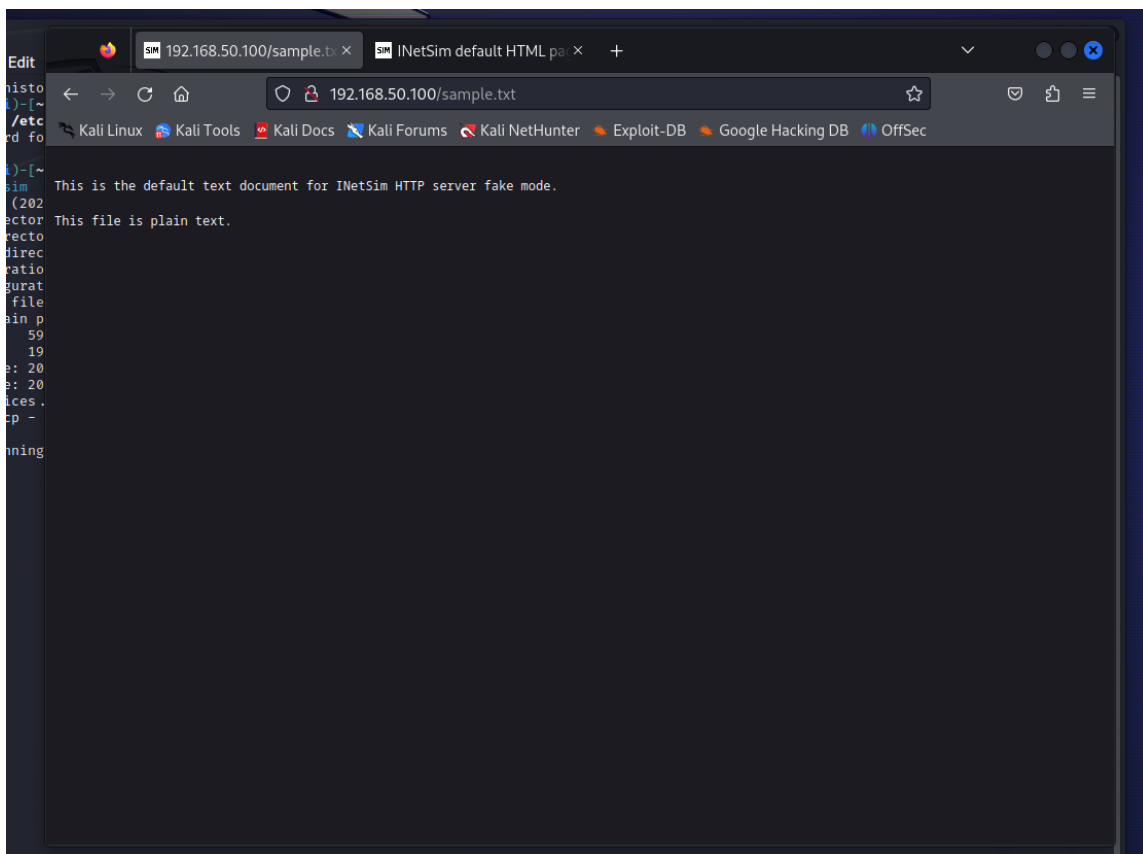


```
kali@kali: ~
File Actions Edit View Help
zsh: corrupt history file /home/kali/.zsh_history
(kali@kali)~$ sudo nano /etc/inetsim/inetsim.conf
[sudo] password for kali:
(kali@kali)~$ sudo inetsim
InetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Using log directory: /var/log/inetsim/
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 5920) ==
Session ID: 5920
Listening on: 192.168.50.100
Real Date/Time: 2023-12-18 11:42:59
Fake Date/Time: 2023-12-18 11:42:59 (Delta: 0 seconds)
Forking services ...
* http_80_tcp - started (PID 5930)
done.
Simulation running.
```

Pagina HTML Inetsim. http

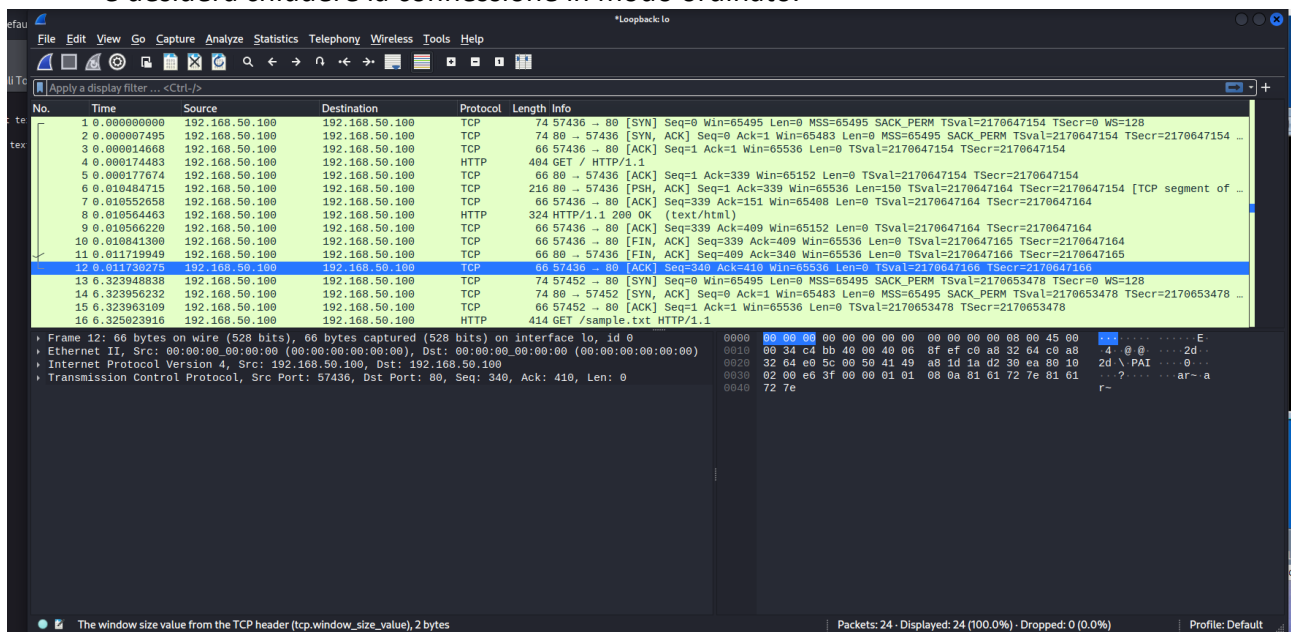


Pagina sample.txt - Http



Wireshark attiva su NIC loopback per vedere sessione TCP e sequenza SYN, SYN ACK, ACK. In questo caso da riga 1 a riga 12 è la sessione TCP su http (porta 80). TCP stabilisce un canale dedicato tra gli hosts che devono prendere parte ad una comunicazione. Per farlo, si seguono 3 passi uno dopo l'altro, questo processo prende il nome di «**three-way-handshake**», ovvero stretta di mano in tre fasi.

1. Il client (57436) che inizia la connessione invia un pacchetto TCP al server (80) destinatario con il flag SYN abilitato ed un numero di sequenza casuale (SYN è un flag del protocollo TCP presente anche nell'header). In questo caso Seq=0
2. Il server (80) risponde inviando al client (57436) un pacchetto con i flag SYN e ACK abilitati, ed un altro numero di sequenza (Seq) casuale, sempre 0 in questo caso. ACK sarà uguale al precedente Seq + 1. In questo caso ACK=1 visto che la Seq precedente era 0.
3. Il client (57436) completa la sincronizzazione inviando un pacchetto ACK, ed inviando i numeri Seq, Ack, come fatto dal server. ACK = 1. Anche qui la precedente Seq era 0.
4. Connessione instaurata. La stringa "Get / HTTP/1.1" è tipicamente associata a un messaggio di risposta HTTP. Questo messaggio indica che il server web ha ricevuto una richiesta (GET) per una risorsa specifica.
5. HTTP/1.1 200 OK: HTTP/1.1: Indica la versione del protocollo HTTP utilizzata. In questo caso, è la versione 1.1. – "200 OK": Questo è il codice di stato HTTP standard. Il "200" indica che la richiesta HTTP è stata completata con successo. Il messaggio "OK" indica che il server ha soddisfatto la richiesta e ha restituito con successo i dati richiesti dal client.
6. PSH (Push): Indica al destinatario di consegnare i dati al livello superiore (applicativo) senza buffering aggiuntivo. È spesso usato quando il mittente desidera inviare immediatamente i dati senza aspettare.
7. FIN (Finish): Il mittente invia un pacchetto FIN per indicare che ha completato l'invio di dati e desidera chiudere la connessione in modo ordinato.



Chiusura Ordinata vs. Chiusura Brusca: Mentre il flag FIN (Finish) viene utilizzato per indicare la chiusura graduale di una connessione, il flag RST (Reset), come nel caso della sessione TCP https, è più deciso e indica una chiusura immediata e senza alcuna forma di negoziazione.

Sintesi differenza:

HTTP (Non Sicuro):

Le sessioni TCP in HTTP sono non sicure perché i dati vengono trasmessi in chiaro attraverso la rete.

La comunicazione non è cifrata, il che significa che se un attaccante intercetta la trasmissione, può leggere e comprendere facilmente i dati scambiati tra il client e il server.

HTTPS (Sicuro):

Le sessioni TCP in HTTPS sono sicure grazie all'aggiunta del layer di sicurezza TLS (Transport Layer Security)

Tutti i dati trasmessi tra il client e il server attraverso una connessione HTTPS sono cifrati. Questo significa che, anche se un attaccante intercetta i dati, non può leggere facilmente le informazioni sensibili.

Handshake TLS:

Quando si utilizza HTTPS, viene eseguito un handshake TLS tra il client e il server durante l'inizio della connessione. Questo processo stabilisce una chiave di sessione segreta che verrà utilizzata per cifrare e decifrare i dati durante la sessione.

Porte Standard:

HTTP utilizza la porta standard 80 per la comunicazione non sicura, mentre HTTPS utilizza la porta standard 443 per la comunicazione sicura.

URL:

Le risorse web servite tramite HTTPS iniziano con "https://" nell'URL, mentre quelle servite tramite HTTP iniziano con "http://".

In sintesi, la principale differenza tra le sessioni TCP in HTTP e HTTPS è la sicurezza. HTTPS aggiunge uno strato di crittografia tramite TLS, rendendo la comunicazione tra client e server più sicura e protetta contro potenziali intercettazioni o manipolazioni dei dati durante il trasferimento.