


Policy & Packet Capture

In questa esercitazione ho inserito una nuova policy all'interno del Firewall Windows della vm Windows 10, l'obiettivo è di consentire il ping in entrata dalla vm KaliLinux.

In Windows firewall ho inserito una nuova regola personalizzata:

 Creazione guidata nuova regola connessioni in entrata

Protocollo e porte

Specificare i protocolli e le porte a cui applicare la regola.

Passaggi:

- Tipo di regola
- Programma
- Protocollo e porte**
- Ambito
- Operazione
- Profilo
- Nome

Selezionare le porte e i protocolli a cui applicare la regola.

Tipo di protocollo: ICMPv4

Numero protocollo: 1

Porta locale: Tutte le porte

Esempio: 80, 443, 5000-5010

Porta remota: Tutte le porte

Esempio: 80, 443, 5000-5010

Impostazioni ICMP (Internet Control Message Protocol): Personalizza...

Dovendo consentire il ping ho applicato il tipo di protocollo su tutti i programmi, su protocollo ICMPv4, su tutte le porte.

Ho poi specificato quale indirizzo ip consentire, inserendo quello della vm KaliLinux con ip:

192.168.50.100

Selezionare gli indirizzi IP remoti a cui applicare la regola.

☐ Qualsiasi indirizzo IP

☒ Questi indirizzi IP:

192.168.50.100

Aggiungi...

Modifica...

Rimuovi

Procedendo ho poi applicato l'action da eseguire:

Selezionare l'azione desiderata per le connessioni che soddisfano le condizioni specificate.

☒ **Consenti la connessione**
Include le connessioni protette con IPsec e quelle non protette.

☐ **Consenti solo connessioni protette**
Include solo le connessioni autenticate mediante IPsec. Le connessioni saranno protette con le impostazioni delle regole e proprietà IPsec nel nodo Regole di sicurezza delle connessioni.

Personalizza...

☐ **Blocca la connessione**

Infine ho salvato la regola su tutti i profili firewall.

Test del ping

```

(kali㉿kali)-[~]
$ ping 192.168.50.102
PING 192.168.50.102 (192.168.50.102) 56(84) bytes of data.
64 bytes from 192.168.50.102: icmp_seq=1 ttl=128 time=0.860 ms
64 bytes from 192.168.50.102: icmp_seq=2 ttl=128 time=0.606 ms
64 bytes from 192.168.50.102: icmp_seq=3 ttl=128 time=0.616 ms
64 bytes from 192.168.50.102: icmp_seq=4 ttl=128 time=0.713 ms
^C
— 192.168.50.102 ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3051ms
rtt min/avg/max/mdev = 0.606/0.698/0.860/0.102 ms

```

Provando a fare un ping dalla 192.168.50.100 (vm kali) verso 50.102 (vm Windows) ho constatato che la regola è funzionante.

La seconda parte dell'esercitazione riguarda l'utilizzo dell'utility Inetsim per l'emulazione di servizi internet. Procedo tramite terminale piazzandomi nella directory /etc/inetsim per modificare il file di configurazione di Inetsim tramite il comando:

nano /etc/inetsim/inetsim.conf

qui ci sono vari protocolli da poter emulare, commento tutti eccetto http e https, scommento la riga dove impostare l'ip e inserisco l'ip della vm kali.

```

#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
#####

```

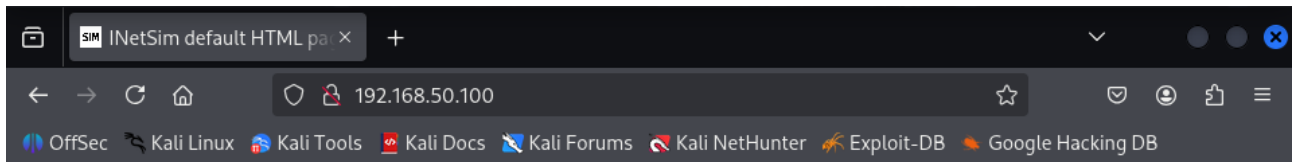
Una volta sistemato il file inetsim.conf ho avviato inetsim.

```

(root㉿kali)-[/etc/inetsim]
# inetsim
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 73998) ==
Session ID: 73998
Listening on: 192.168.50.100
Real Date/Time: 2025-10-12 08:53:23
Fake Date/Time: 2025-10-12 08:53:23 (Delta: 0 seconds)
Forking services ...
* https_443_tcp - started (PID 74001)
* http_80_tcp - started (PID 74000)
done.
Simulation running.

```

Dopo aver startato la simulazione ho verificato tramite browser che stesse funzionando. Verifica che può essere fatta anche da vm Windows o altra macchina in rete interna.

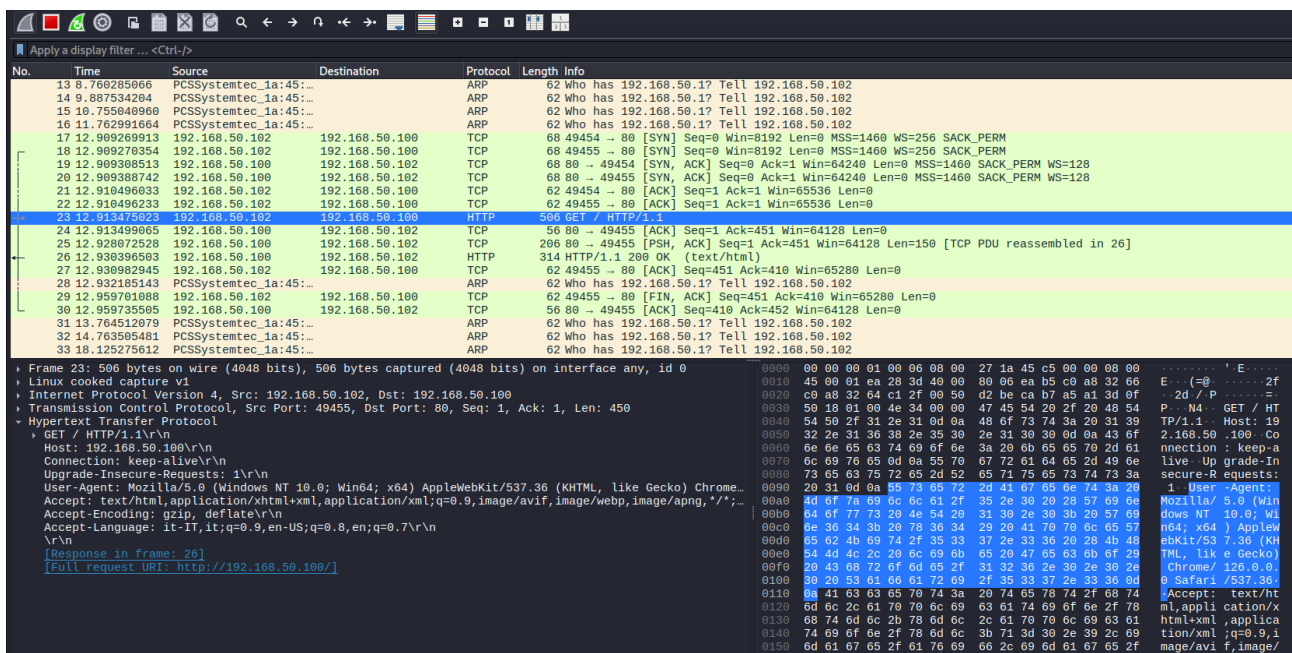


This is the default HTML page for INetSim HTTP server fake mode.

This file is an HTML document.

A questo punto ho iniziato con la parte 3 dell'esercitazione, lo sniffing tramite Wireshark.

Sono prima entrato sulla vm Windows10, aperto un browser e visitato la pagina https simulata da inetsim, ho poi aperto wireshark su Kali e iniziato a sniffare pacchetti su "any".



Questi nell'immagine sono i pacchetti che Wireshark ha sniffato mentre ho visitato la pagina https di Inetsim. Da qui possiamo vedere in dettaglio ogni request e response che le due macchine si inviano.

Conclusione

Fin quando la simulazione di Inetsim è rimasta attiva ho potuto sniffare i pacchetti delle comunicazioni https che arrivavano alla macchina Kali 192.168.50.100, dove era ospitata la simulazione del servizio https.