

# Progetto Finale M1

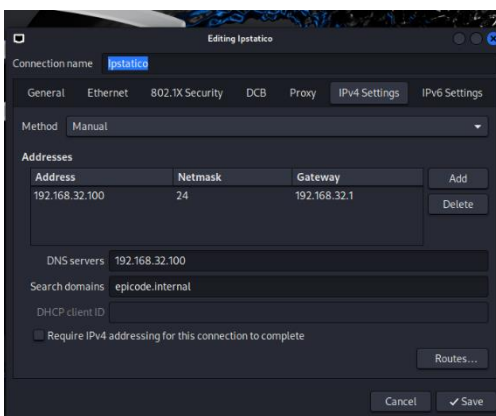
Requisiti e servizi attivi:

- Kali Linux : IP 192.168.32.100
- Windows: IP 192.168.32.101
- HTTPS Server :ATTIVO
- Servizio DNS per risoluzione nomi di domini: ATTIVO

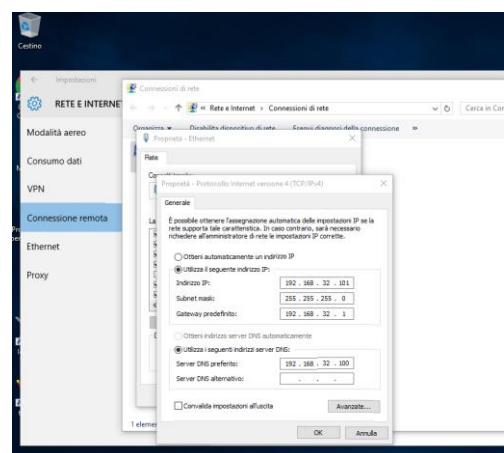
Procedimento con relativi screen.

Con il progetto finale andremmo a lavorare su due macchine virtuali installate in precedenza su Virtual Box, Kali Linux che sarà il nostro Server ed in fine Windows 10 che sarà il Client.

Per prima cosa ho avviato le due macchine e sono andato subito ad impostare i relativi indirizzi IP prima su Kali Linux e poi su Windows



CONFIGURAZIONE IP KALI

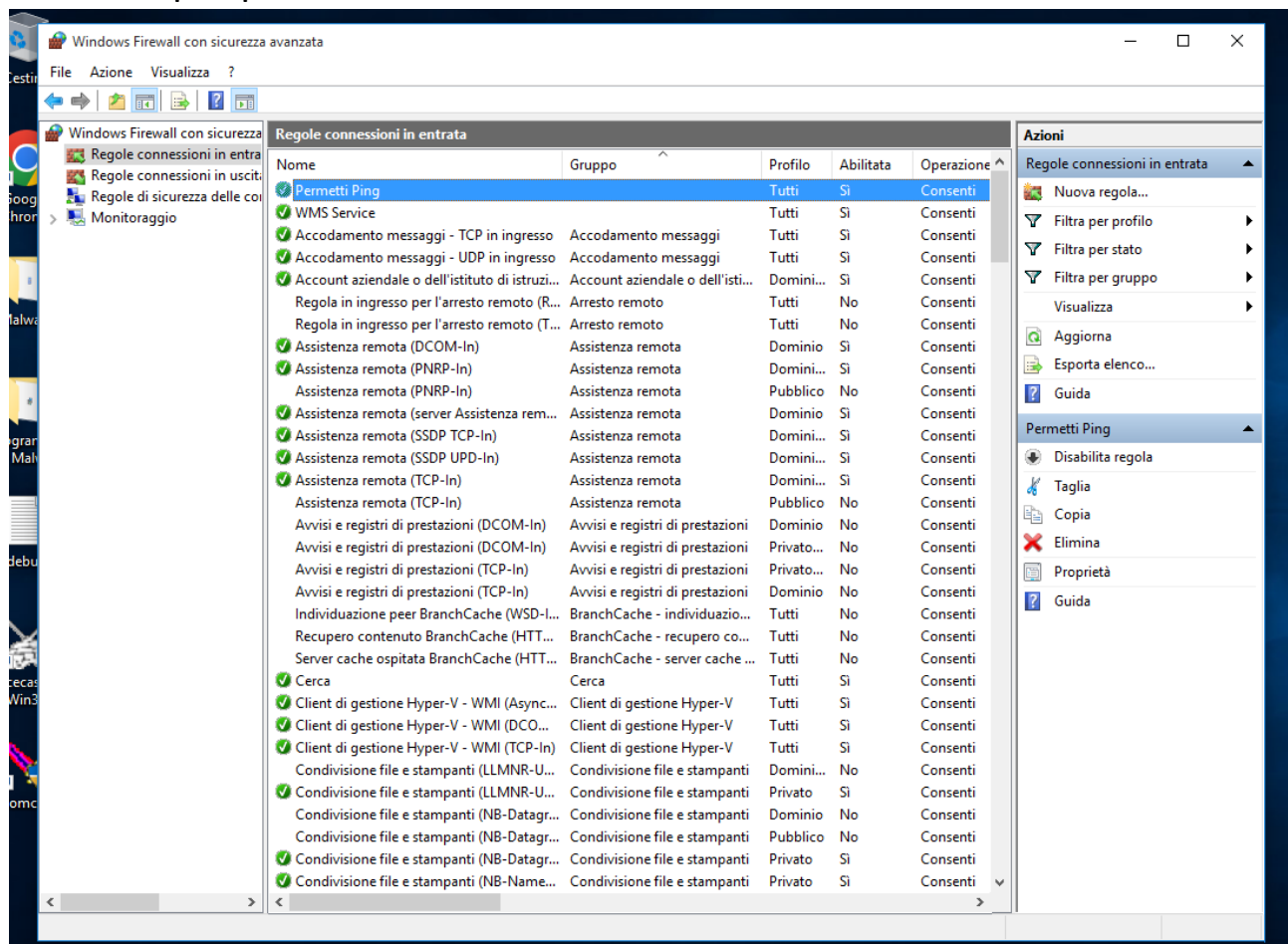


CONFIGURAZIONE IP E DNS WINDOWS

Nelle foto qui di sopra possiamo osservare la configurazione degli IP e DNS sulle due macchine.

Sopra sono possiamo osservare la configurazione degli IP e D

Dopo la configurazione degli IP ho cambiato la policy del firewall di windows per permettere la comunicazione delle due macchine tra di loro



Quindi ho dato al Firewall una nuova regola autorizzandolo a far passare gli IP dati da me in fase di configurazione .

Sopra sono possiamo osservare la configurazione degli IP e D

```
File Actions Edit View Help
GNU nano 8.1
file, getting its servers from this file instead (see below), then
uncomment this.
no-resolv

If you don't want dnsmasq to poll /etc/resolv.conf or other resolv
files for changes and re-read them then uncomment this.
no-poll

Add other name servers here, with domain specs if they are for
non-public domains.
server=/localnet/192.168.0.1

Example of routing PTR queries to nameservers: this will send all
address→name queries for 192.168.3/24 to nameserver 10.1.2.3
server=/3.168.192.in-addr.arpa/10.1.2.3

Add local-only domains here, queries in these domains are answered
from /etc/hosts or DHCP only.
local=/localnet/

Add domains which you want to force to an IP address here.
The example below send any host in double-click.net to a local
web-server.
address=/double-click.net/127.0.0.1

--address (and --server) work with IPv6 addresses too
address=/www.thekelleys.org.uk/fe80::20d:60ff:fe30:f83
address=/epicode.internal/192.168.32.100
Add the IPs of all queries to yahoo.com, google.com, and their
subdomains to the vpn and search ipsets:
ipset=/yahoo.com/google.com/vpn,search

Add the IPs of all queries to yahoo.com, google.com, and their
subdomains to netfilters sets, which is equivalent to
'nft add element ip test vpn { ... }; nft add element ip test search { ... }'
nftset=/yahoo.com/google.com/ip#test#vpn,ip#test#search

Use netfilters sets for both IPv4 and IPv6:
This adds all addresses in *.yahoo.com to vpn4 and vpn6 for IPv4 and IPv6 addresses.
nftset=/yahoo.com/4#ip#test#vpn4
nftset=/yahoo.com/6#ip#test#vpn6

You can control how dnsmasq talks to a server: this forces
queries to 10.1.2.3 to be routed via eth1
server=10.1.2.3@eth1

and this sets the source (ie local) address used to talk to
10.1.2.3 to 192.168.1.1 port 55 (there must be an interface with that
Help Write Out Where Is Cut Execute
```

Nella foto qui di fianco, sono ritornato nuovamente sulla macchina Kali Linux, ma questa volta per andare a configurare Inetsim per poter attivare il DNS ( epicode.internal) con indirizzo IP:192.168.32.100

```
Prompt dei comandi
C:\Microsoft Windows [Versione 10.0.10240]
(c) 2015 Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\User>ping 192.168.32.100

Esecuzione di Ping 192.168.32.100 con 32 byte di dati:
Risposta da 192.168.32.100: byte=32 durata=2ms TTL=64
Risposta da 192.168.32.100: byte=32 durata=2ms TTL=64
Risposta da 192.168.32.100: byte=32 durata<1ms TTL=64
Risposta da 192.168.32.100: byte=32 durata=1ms TTL=64

Statistiche Ping per 192.168.32.100:
    Pacchetti: Trasmessi = 4, Ricevuti = 4,
    Persi = 0 (0% persi),
    Tempo approssimativo percorsi andata/ritorno in millisecondi:
        Minimo = 0ms, Massimo = 2ms, Medio = 1ms

C:\Users\User>
```

Successivamente dopo aver configurato le due macchine, attivando le policy del firewall windows e impostando il DNS epicode.internal, sono ritornato sulla macchina Windows come si evince da foto, ho aperto il prompt dei

comandi per lanciare il comando PING da Windows a Kali Linux .

Sopra sono possiamo osservare la configurazione degli IP e D

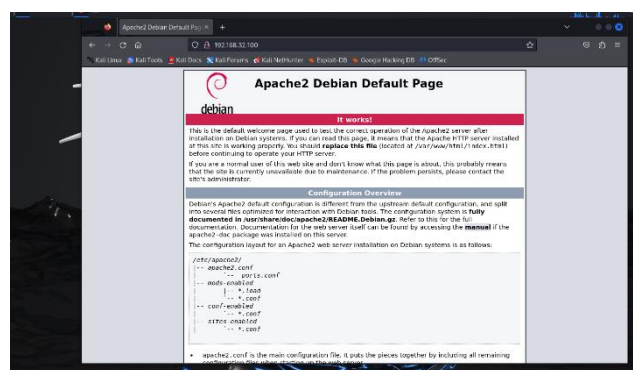
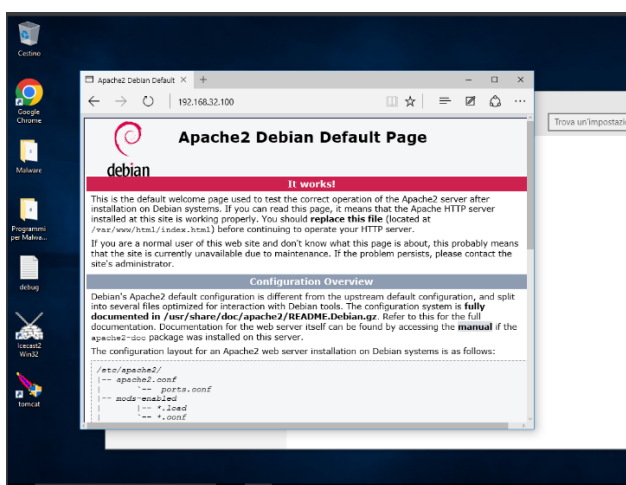
```

(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 122047) ==
Session ID: 122047
Listening on: 127.0.0.1
Real Date/Time: 2024-11-29 15:02:33
Fake Date/Time: 2024-11-29 15:02:33 (Delta: 0 seconds)
Forking services ...
  * https_443_tcp - started (PID 122049)
done.
Simulation running.

```

Dopo aver constatato che le due macchine tra di loro riescono a comunicare, ho aperto Kali Linux e lanciato tramite il terminale ho lanciato il comando inetsim come si può vedere nella foto qui di sopra. Tale comando viene comunemente usato come simulatore per servizi internet, ma viene anche usato in ambienti sicuri come Laboratori virtuali dove poter testare malware o file sospetti in modo da non recare danno.

Tornando al nostro esercizio questo comando una volta avviato ci darà la possibilità di andarci a collegare direttamente al nostro DNS precedentemente configurato sotto l'URL di epicode.internal.



Queste sono le due macchine Kali Linux sulla nostra dx dove abbiamo inserito IP 192.168.32.100 nella barra degli indirizzi per poi portarci sul

Sopra sono possiamo osservare la configurazione degli IP e D

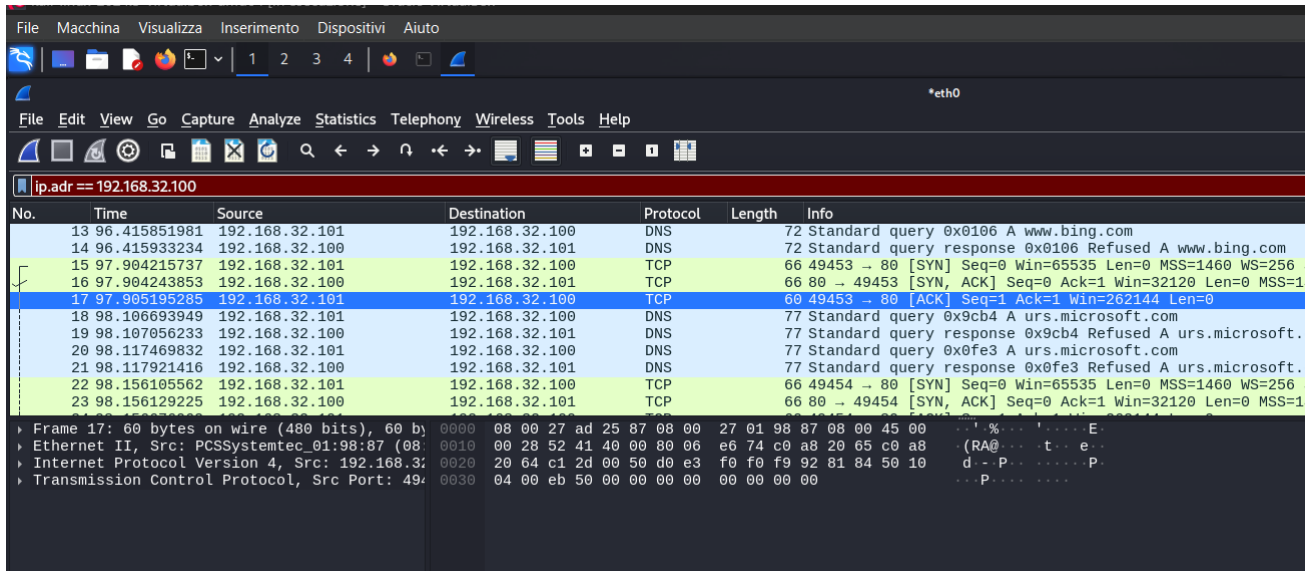
sito internet epicode.internal e mentre nella schermata di sx abbiamo la macchina di windows anche lei collegata all'indirizzo IP 192.168.32.100

```
(kali㉿kali)-[~]
$ ping
ping: usage error: Destination address required

(kali㉿kali)-[~]
$ ping epicode.internal
PING epicode.internal (192.168.32.100) 56(84) bytes of data.
64 bytes from epicode.internal (192.168.32.100): icmp_seq=1 ttl=64 time=0.020 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=2 ttl=64 time=0.022 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=3 ttl=64 time=0.038 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=4 ttl=64 time=0.028 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=5 ttl=64 time=0.026 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=6 ttl=64 time=0.027 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=7 ttl=64 time=0.023 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=8 ttl=64 time=0.023 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=9 ttl=64 time=0.021 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=10 ttl=64 time=0.022 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=11 ttl=64 time=0.023 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=12 ttl=64 time=0.023 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=13 ttl=64 time=0.022 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=14 ttl=64 time=0.024 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=15 ttl=64 time=0.023 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=16 ttl=64 time=0.022 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=17 ttl=64 time=0.021 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=18 ttl=64 time=0.022 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=19 ttl=64 time=0.020 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=20 ttl=64 time=0.047 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=21 ttl=64 time=0.047 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=22 ttl=64 time=0.021 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=23 ttl=64 time=0.018 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=24 ttl=64 time=0.024 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=25 ttl=64 time=0.028 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=26 ttl=64 time=0.023 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=27 ttl=64 time=0.023 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=28 ttl=64 time=0.023 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=29 ttl=64 time=0.043 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=30 ttl=64 time=0.022 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=31 ttl=64 time=0.023 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=32 ttl=64 time=0.025 ms
64 bytes from epicode.internal (192.168.32.100): icmp_seq=33 ttl=64 time=0.039 ms
^C
— epicode.internal ping statistics —
33 packets transmitted, 33 received, 0% packet loss, time 32773ms
rtt min/avg/max/mdev = 0.018/0.025/0.047/0.007 ms
```

Come si nota oltre a provarci a collegare direttamente tramite Browser web abbiamo fatto un 'ulteriore test da terminale sempre con il comando ping ma stavolta anziché di inserire l' IP abbiamo inserito il seguente comando :ping epicode.internal. Ricordo anche che il comando PING può

rivelarsi un' ottimo strumento di diagnostica per capire se è tutto ok nella nostra configurazione.



Sempre con inetsim attivo e provandoci a collegare alla nostra pagina in questione epicode.internal ho eseguito il tool WireShark.

Proprio quest'ultimo serve per andare ad analizzare tutto il traffico di rete e la cattura di pacchetti in transito ed andarli ad analizzare ,infatti wire shark è uno strumento di analisi.

Sopra sono possiamo osservare la configurazione degli IP e D