

**La Compagnia
dell'Anello di Rete SrL**

Theta SrL

By: Valerio Benedetti, Marco Maniaci,
Sonia Laterza, Matteo Piccinini, Krishi
Chuttur, Andrea Morici, Lorenzo Croci



Indice

- Introduzione
- Specifiche del progetto
- Architettura di rete
- Configurazione e subnet
- Preventivo
- Test di rete
- Conclusione





La Compagnia dell'Anello di Rete

La Compagnia dell'Anello di Rete nasce nel 2024, quando, dopo il caos digitale prodotto dalla pandemia, 7 Cybersecurity Specialists decidono di liberarsi dalle catene del lavoro dipendente per collaborare alla stessa missione: monitorare ogni angolo della rete, affinché i tuoi dati viaggino protetti fino alla fine della vostra epica avventura aziendale.

Specifiche progettuali

Siamo stati ingaggiati dalla compagnia Theta per sviluppare un preventivo di spesa e un progetto di rete per la loro infrastruttura IT. Ecco i requisiti e i componenti necessari:

- Struttura dell'edificio: 6 piani
- Dispositivi previsti: 20 computer per piano, per un totale di 120 computer
- Componenti aggiuntivi:
 - 1 Web server (rappresentato dalla macchina DVWA di Metasploitable)
 - 1 Firewall perimetrale
 - 1 NAS (Network Attached Storage)
 - 3 IDS/IPS (Intrusion Detection System / Intrusion Prevention System)



Architettura di rete

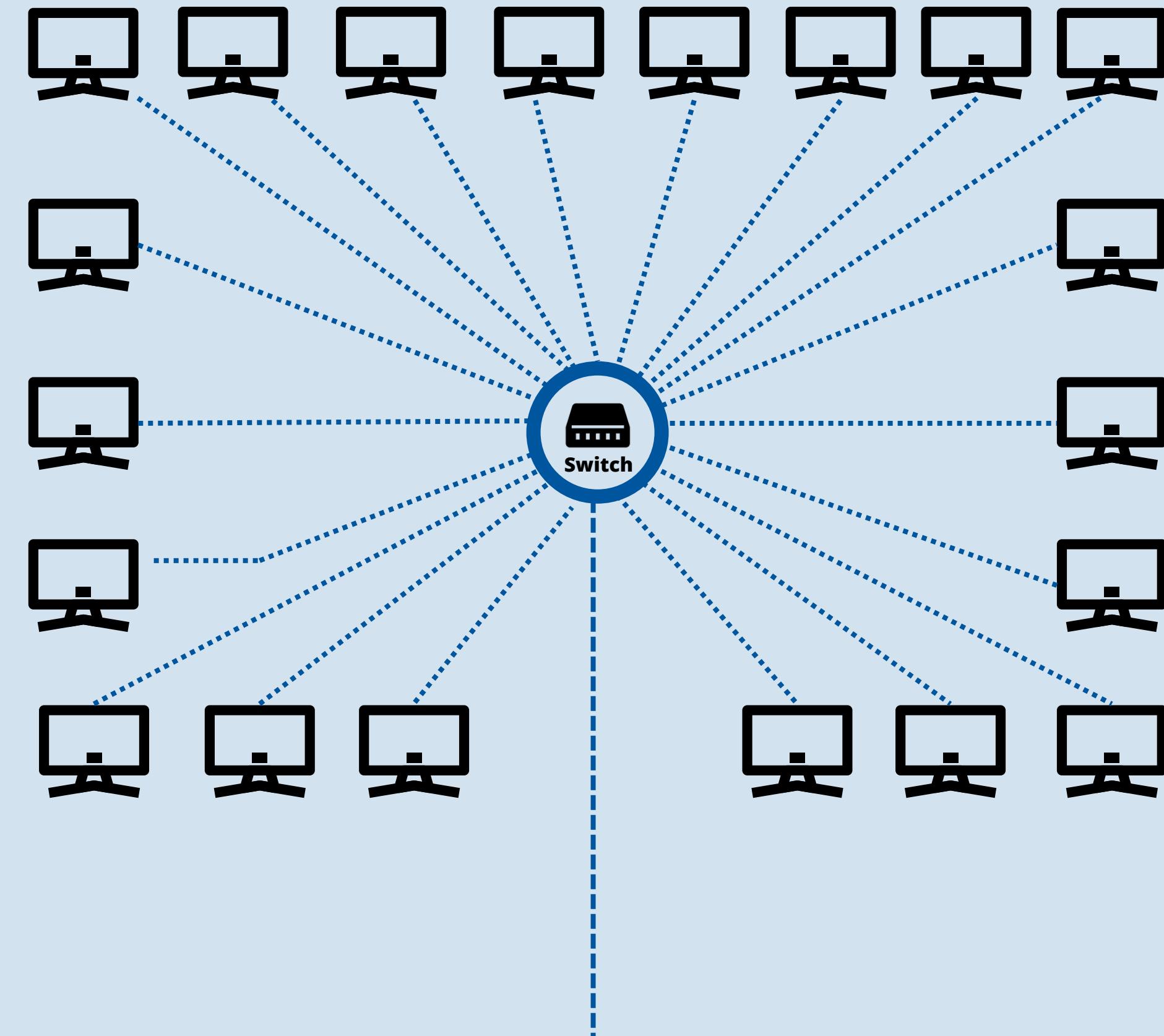


Ogni piano dispone di 20 personal computer collegati a uno switch, per un totale di 120 dispositivi. Gli switch sono collegati a un router gateway, posizionato al 3° piano per minimizzare la lunghezza dei cavi e ridurre la dispersione. Al 3° piano si trovano anche il dispositivo NAS, il web server, il firewall hardware e i 3 sistemi IDS/IPS, in modo che siano vicini tra loro e centralizzati rispetto agli altri piani.

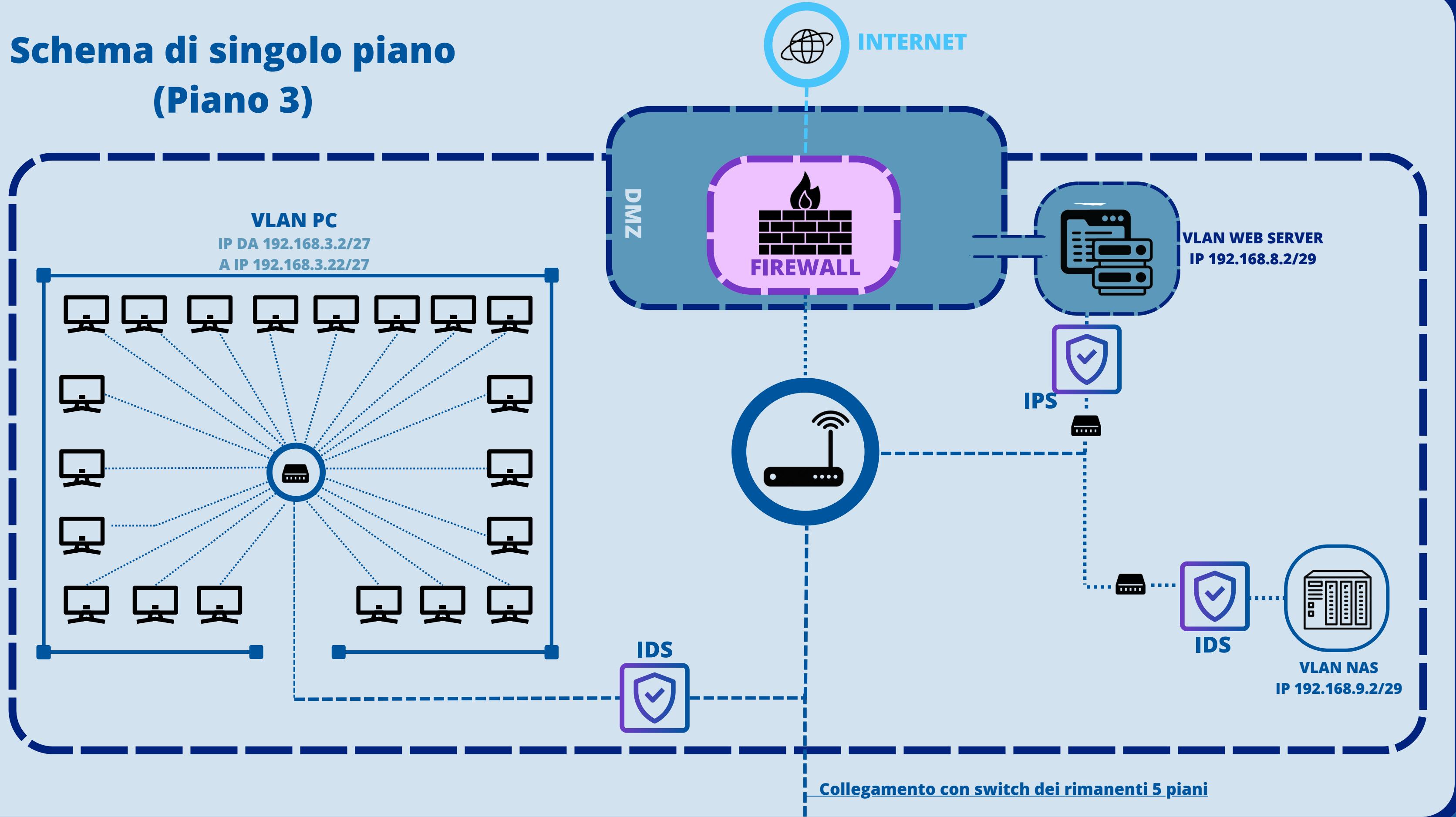
Assegnamento Indirizzi IP

VLAN	Indirizzo di Rete	Indirizzo di Broadcast	Subnet Mask	Primo Host	Gateway
VLAN-Piano-1	192.168.1.0	192.168.1.31	255.255.255.224 (/27)	192.168.1.2	192.168.1.1
VLAN-Piano-2	192.168.2.0	192.168.2.31	255.255.255.224 (/27)	192.168.2.2	192.168.2.1
VLAN-Piano-3	192.168.3.0	192.168.3.31	255.255.255.224 (/27)	192.168.3.2	192.168.3.1
VLAN-Piano-4	192.168.4.0	192.168.4.31	255.255.255.224 (/27)	192.168.4.2	192.168.4.1
VLAN-Piano-5	192.168.5.0	192.168.5.31	255.255.255.224 (/27)	192.168.5.2	192.168.5.1
VLAN-Piano-6	192.168.6.0	192.168.6.31	255.255.255.224 (/27)	192.168.6.2	192.168.6.1
VLAN-WEBSERVER	192.168.8.0	192.168.8.7	255.255.255.248 (/29)	192.168.8.2	192.168.8.1
VLAN-NAS	192.168.9.0	192.168.9.7	255.255.255.248 (/29)	192.168.9.2	192.168.9.2

Schema di singolo piano (Piani 1, 2, 4, 5, 6)



Schema di singolo piano (Piano 3)



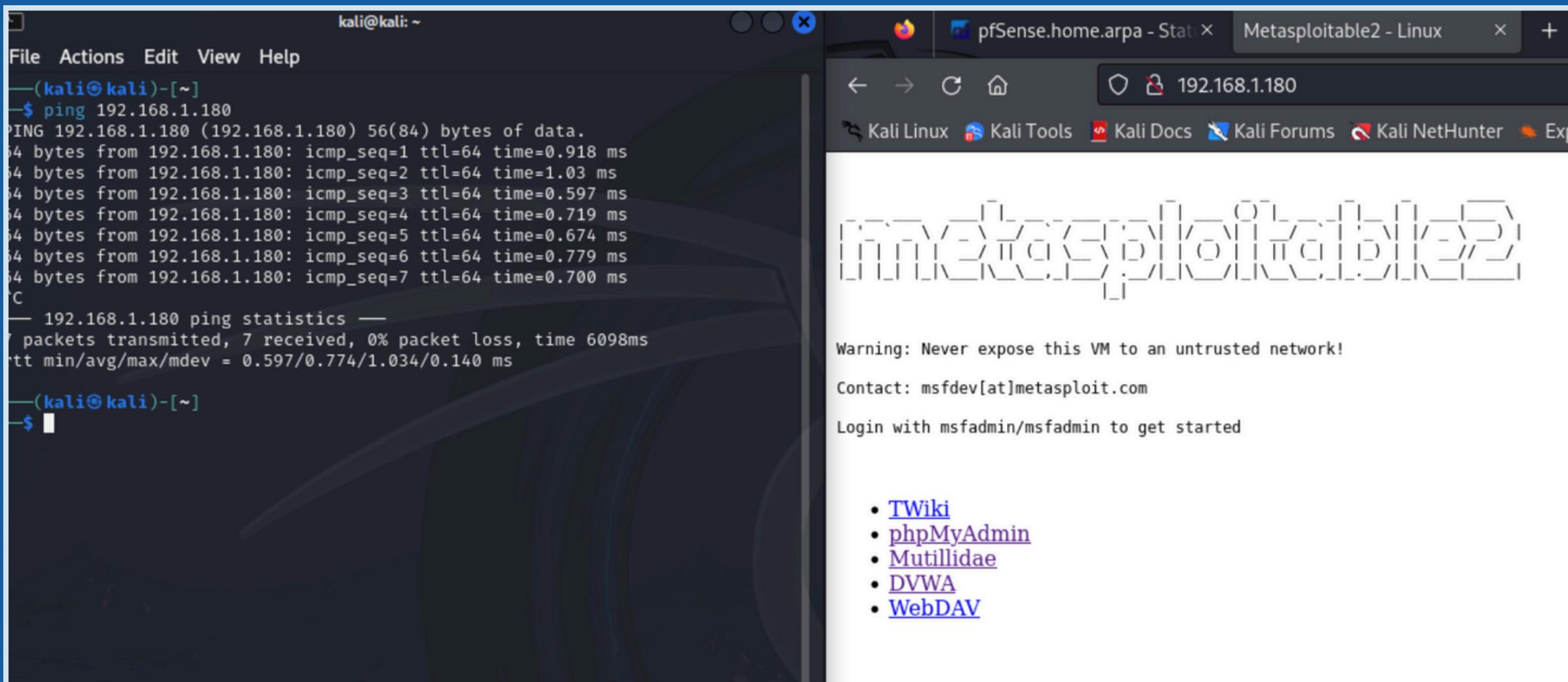


Totale preventivo: 174.490,58€

Test di Rete - Ping

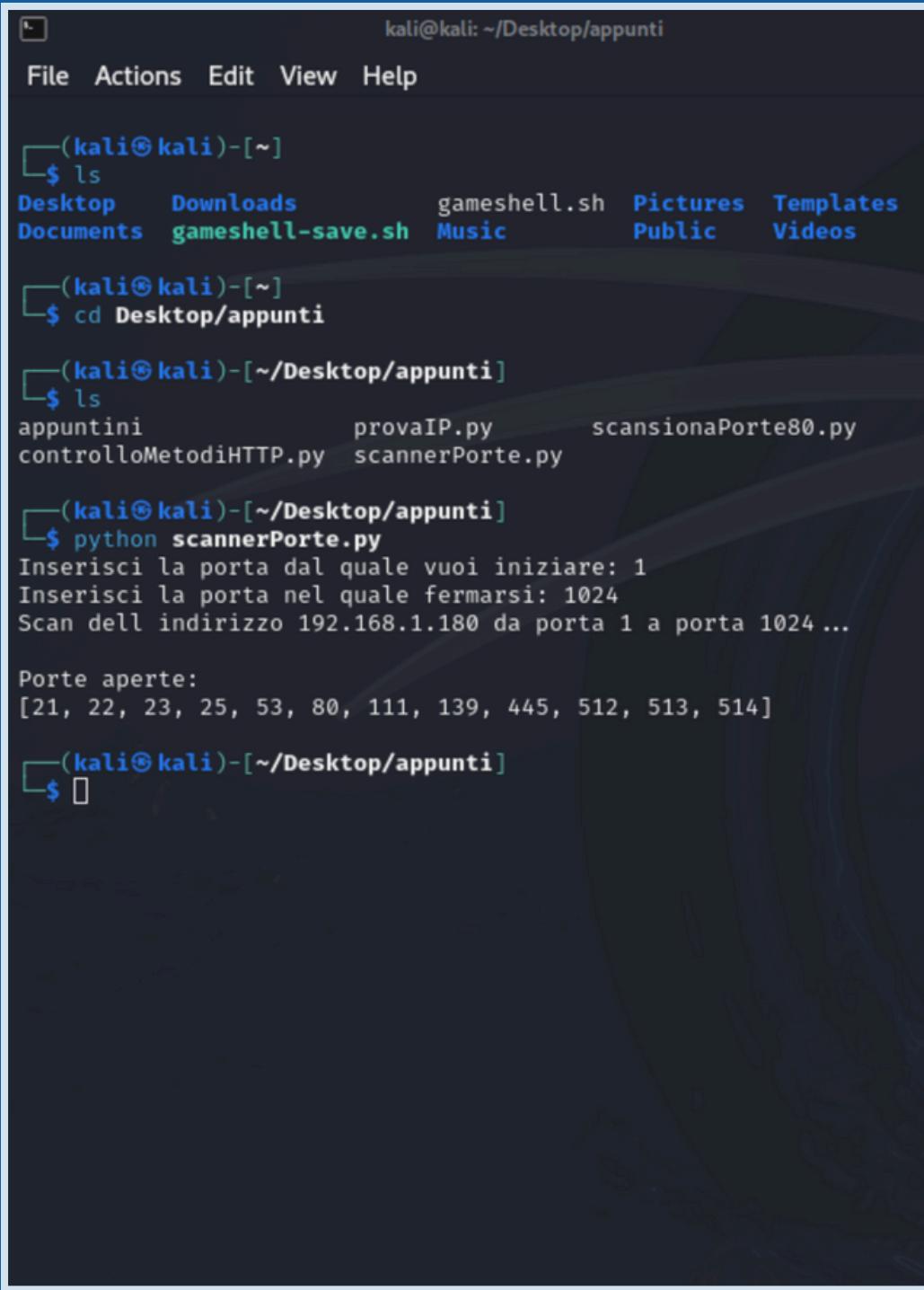
Abbiamo eseguito già, attraverso una simulazione strutturata come la configurazione proposta nelle slide precedenti, dei test preventivi di rete per verificare la vulnerabilità dei protocolli e la corretta comunicazione dei dispositivi.

Il primo test eseguito è stato di comunicazione fra i dispositivi proposti, effettuando un ping fra i dispositivi, che ha avuto successo senza intoppi.



Test di Rete - Scansione porte

Abbiamo poi proceduto anche con la verifica dell'apertura delle principali porte di comunicazione di rete per assicurarci sia che con i nostri metodi di sicurezza potremo occuparci delle corrette vulnerabilità, sia che invece i servizi a voi necessari siano disponibili fin da subito.



The terminal window shows the following session:

```
kali㉿kali:[~] $ ls
Desktop Downloads gameshell.sh Pictures Templates
Documents gameshell-save.sh Music Public Videos

(kali㉿kali:[~] $ cd Desktop/appunti
(kali㉿kali:[~/Desktop/appunti] $ ls
appuntini provaIP.py scansionaPorte80.py
controlloMetodiHTTP.py scannerPorte.py

(kali㉿kali:[~/Desktop/appunti] $ python scannerPorte.py
Inserisci la porta dal quale vuoi iniziare: 1
Inserisci la porta nel quale fermarsi: 1024
Scan dell indirizzo 192.168.1.180 da porta 1 a porta 1024 ...
Porte aperte:
[21, 22, 23, 25, 53, 80, 111, 139, 445, 512, 513, 514]

(kali㉿kali:[~/Desktop/appunti] $ )
```

The code editor window shows the Python script `scannerPorte.py`:

```
1 import socket
2
3 # Indirizzo IP della macchina Metasploitable2
4 ip = "192.168.1.180"
5
6 # Richiede la porta di partenza
7 portaInizio = int(input("Inserisci la porta dal quale vuoi iniziare: "))
8
9 # Richiede la porta di fine
10 portaFine = int(input("Inserisci la porta nel quale fermarsi: "))
11
12 # Crea un range di porte da scansionare
13 rangePorte = range(portaInizio, portaFine + 1)
14
15 # Lista per memorizzare le porte aperte
16 porteAperte = []
17 print(f"Scan dell indirizzo {ip} da porta {portaInizio} a porta {portaFine} ... ")
18
19 for porta in rangePorte:
20     with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
21         # Imposta il timeout per la connessione a 1 secondo
22         sock.settimeout(1)
23         # Prova a connettersi alla porta
24         result = sock.connect_ex((ip, porta))
25         # Se la connessione ha successo, la porta è aperta
26         if result == 0:
27             porteAperte.append(porta)
28             #print(f"Port {porta}: Open")
29         # Se la connessione fallisce, la porta è chiusa
30         else:
31             #print(f"Port {porta}: Closed")
32
33 print("\nPorte aperte:")
34 print(porteAperte)
```

Test di Rete -

Verifica verbi HTTP

Infine, abbiamo proceduto anche con la verifica dei verbi HTTP più utilizzati, in maniera tale da poter controllare il completo supporto e prevenire successivo debugging.

The screenshot shows a terminal window on the left and a code editor window on the right.

Terminal (kali㉿kali ~)

```
$ cd Desktop/appunti  
$ python controlloMetodiHTTP.py  
Controllo il metodo HTTP per http://192.168.1.180/phpMyAdmin...  
GET: 200 Supportato  
POST: 200 Supportato  
PUT: 200 Supportato  
DELETE: 200 Supportato  
HEAD: 200 Supportato  
OPTIONS: 200 Supportato  
PATCH: 200 Supportato  
  
Metodi HTTP supportati:  
['GET', 'POST', 'PUT', 'DELETE', 'HEAD', 'OPTIONS', 'PATCH']
```

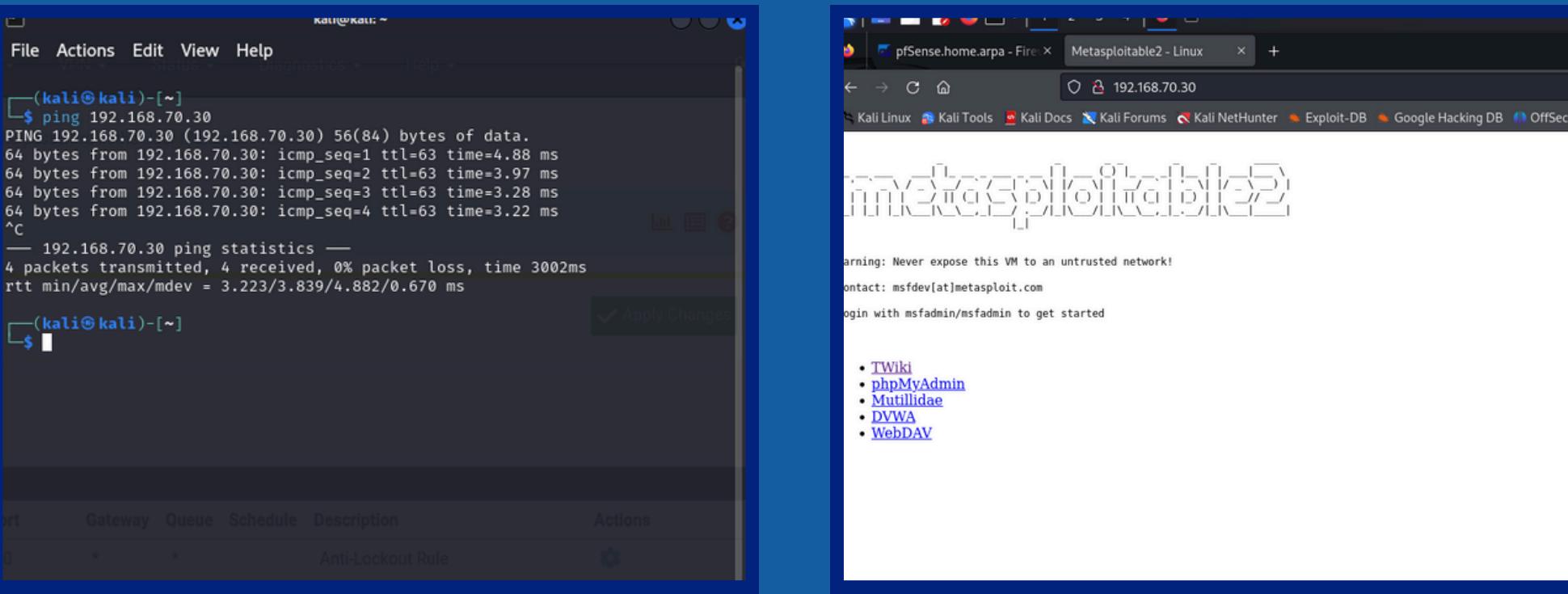
Code Editor (~/Desktop/appunti/controlloMetodiHTTP.py - Mousepad)

```
1 import requests  
2  
3 # URL di phpMyAdmin su Metasploitable2  
4 url = "http://192.168.1.180/phpMyAdmin"  
5 print(f"Controllo il metodo HTTP per {url} ... ")  
6  
7 # Lista per memorizzare i metodi HTTP abilitati  
8 metodi = []  
9  
10 # Definisci i metodi HTTP da testare  
11 metodiHTTP = ['GET', 'POST', 'PUT', 'DELETE', 'HEAD', 'OPTIONS', 'PATCH']  
12  
13 response_get = requests.get(url)  
14  
15 for metodo in metodiHTTP:  
16     try:  
17         # Invia una richiesta per ciascun metodo  
18         if metodo == 'OPTIONS':  
19             # Utilizza il metodo OPTIONS per ottenere i metodi supportati  
20             response = requests.options(url)  
21         else:  
22             response = requests.request(metodo, url)  
23  
24         # Controlla se il metodo è supportato  
25         # Considera metodi supportati se la risposta non è un errore  
26         if response.status_code < 400:  
27             metodi.append(metodo)  
28             print(f"{metodo}: {response_get.status_code} Supportato")  
29         else:  
30             print(f"{metodo}: {response_get.status_code} Non supportato")  
31     except requests.exceptions.RequestException as e:  
32         print(f"{metodo}: Error - {str(e)}")  
33  
34 print("\nMetodi HTTP supportati:")  
35 print(metodi)  
36
```

Protezione di rete - Blocco porta 80

Per concludere, abbiamo inserito una regola firewall come esempio di facile chiusura di una vulnerabilità web: in questo caso useremo la porta 80 come esempio. Questo è atto a confermare la flessibilità della configurazione e la facilità di impostazione di regole.

Abbiamo simulato quindi una rete funzionante attraverso 3 macchine virtuali, Metasploitable, pfsense e Kali.



Come dimostrato dai due screenshot della nostra simulazione, potete vedere che tutti i pacchetti (packets) di informazione fra Kali e Metasploitable sono condivisi senza nessuna perdita.

Protezione di rete - Blocco porta 80

Concluso che i dispositivi comunicano, abbiamo apposto una regola attraverso pfSense (in questo caso funge da nostro firewall) per bloccare le comunicazioni attraverso la porta 80 (HTTP).

The screenshot shows the configuration for a new firewall rule. The 'Action' dropdown is set to 'Block'. The 'Disabled' checkbox is checked. The 'Interface' is set to 'LAN'. The 'Address Family' is 'IPv4'. The 'Protocol' is 'TCP'. In the 'Source' section, the source is '192.168.60.30'. In the 'Destination' section, the destination is '192.168.70.30' and the destination port range is 'HTTP (80)'.

Nello screenshot qui sopra, i dettagli della regola per il blocco della porta 80. Qui sotto invece, una lista delle regole firewall.

The screenshot shows a list of existing firewall rules. The 'LAN' tab is selected. There are four rules listed:

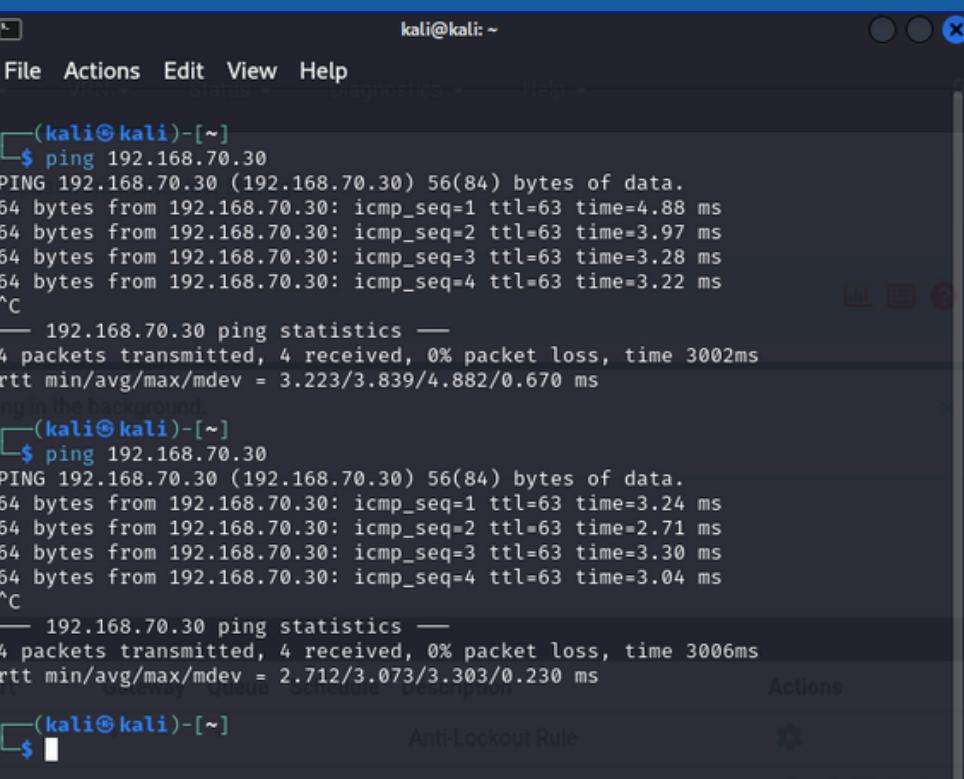
States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
✓ 1/339 KiB	*	*	*	LAN Address	80	*	*	*	Anti-Lockout Rule	
✗ 0/0 B	IPv4 TCP	192.168.60.30	*	192.168.70.30	80 (HTTP)	*	none			
✓ 0/70 KiB	IPv4 *	LAN subnets	*	*	*	*	none		Default allow LAN to any rule	
✓ 0/0 B	IPv6 *	LAN subnets	*	*	*	*	none		Default allow LAN IPv6 to any rule	

At the bottom, there are buttons for 'Add', 'Delete', 'Toggle', 'Copy', 'Save', and 'Separator'.

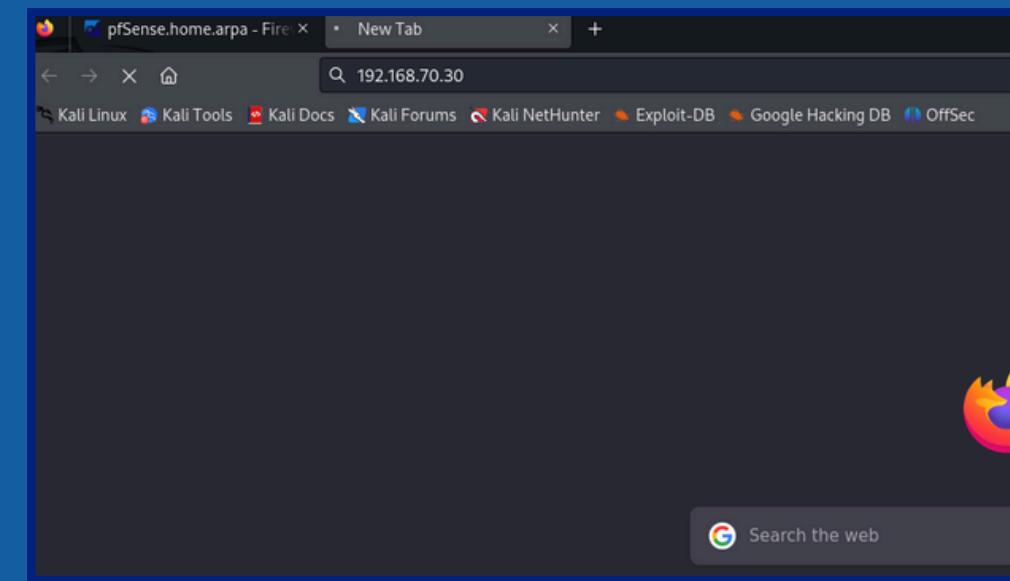
Protezione di rete - Blocco porta 80

A questo punto, ci è solo rimasto da verificare l'effettivo blocco da parte del firewall.

Come ci aspettavamo, la comunicazione è interrotta, eliminando così la vulnerabilità legata alla porta 80.



```
kali@kali: ~
File Actions Edit View Help
(kali㉿kali)-[~]
$ ping 192.168.70.30
PING 192.168.70.30 (192.168.70.30) 56(84) bytes of data.
64 bytes from 192.168.70.30: icmp_seq=1 ttl=63 time=4.88 ms
64 bytes from 192.168.70.30: icmp_seq=2 ttl=63 time=3.97 ms
64 bytes from 192.168.70.30: icmp_seq=3 ttl=63 time=3.28 ms
64 bytes from 192.168.70.30: icmp_seq=4 ttl=63 time=3.22 ms
^C
--- 192.168.70.30 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 3.223/3.839/4.882/0.670 ms
pinging in the background...
(kali㉿kali)-[~]
$ ping 192.168.70.30
PING 192.168.70.30 (192.168.70.30) 56(84) bytes of data.
64 bytes from 192.168.70.30: icmp_seq=1 ttl=63 time=3.24 ms
64 bytes from 192.168.70.30: icmp_seq=2 ttl=63 time=2.71 ms
64 bytes from 192.168.70.30: icmp_seq=3 ttl=63 time=3.30 ms
64 bytes from 192.168.70.30: icmp_seq=4 ttl=63 time=3.04 ms
^C
--- 192.168.70.30 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 2.712/3.073/3.303/0.230 ms
(kali㉿kali)-[~]
```



A differenza dei primi due screenshot di comunicazione, vediamo come i pacchetti di informazione non riescono ad essere scambiati in maniera efficace, e che il sito di Metasploitable non appare più come nel primo screenshot.



Conclusion

Abbiamo progettato una rete aziendale per Theta, distribuendo 120 dispositivi su 6 piani con una struttura sicura e scalabile.

Ogni piano è organizzato tramite VLAN, con centralizzazione del router e dispositivi di sicurezza al 3° piano. Firewall, IDS/IPS, NAS e web server sono stati posizionati strategicamente per garantire la protezione dei dati sensibili e la sicurezza perimetrale.

Il preventivo finale ammonta a 174.490,58€, comprendendo la fornitura di un provider internet per garantire una connessione affidabile. Questo progetto soddisfa tutte le esigenze di rete dell'azienda, con margine per futuri ampliamenti.



**La Compagnia
dell'Anello di Rete
SRL**

THANK YOU!



Valerio Benedetti

CEO & Founder



+39 12345678910



www.lacompagniadellanellodirete.com



Via Finta 123, Roma