

Progetto Build Week S4

17/12/2024

CONSEGNA:

Progetto di Rete per la Compagnia Theta

1. Specifiche del Progetto

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)

2.Rete Interna Aziendale

- Switch per ogni piano: Collegare i 20 computer di ciascun piano a uno switch dedicato.
- Router: Collegare tutti gli switch dei vari piani a un router centrale.
- Firewall: Posizionare il firewall perimetrale tra il router interno e la connessione a Internet.
- NAS: Collegare il NAS allo switch al piano terra (vicino al router) per i computer aziendali.
- IDS/IPS: Implementare 3 IDS/IPS nel perimetro interno per monitorare il traffico di rete e prevenire intrusioni. Rete Esterna (Internet)
- Connessione a Internet: Collegare il firewall perimetrale a Internet.
- Web Server: Posizionare il web server (DVWA di Metasploitable) nella zona demilitarizzata (DMZ) tra il firewall e la connessione a Internet, garantendo così un accesso sicuro dall'esterno
- Se avete bisogno di un altro firewall potete comprarlo e montarlo.

3. Testing della Rete

Per concludere il progetto, effettueremo una serie di test sulla rete implementata.

I test includeranno:

3.1 Esercizio Traccia e requisiti Verifica dei Verbi HTTP: Scriveremo un programma in Python per inviare richieste HTTP (GET, POST, PUT, DELETE) al web server e verificare le risposte.

3.2. Scansione delle Porte: Utilizzeremo un programma in Python per eseguire una scansione delle porte sui dispositivi di rete, verificando la sicurezza e l'accessibilità delle varie porte di comunicazione.

4. Report Finale

Alla conclusione dei test, redigeremo un report dettagliato che includerà:

- **Risultati dei Test HTTP:**

Documentazione delle risposte ricevute dal web server per ogni verbo HTTP testato.

- **Risultati della Scansione delle Porte :**

Elenco delle porte aperte e chiuse sui vari dispositivi, , con relative raccomandazioni di sicurezza. Questo approccio garantirà che l'infrastruttura di rete della compagnia Theta sia ben progettata, sicura e pronta per operare modo efficiente.

TAPPE INTERMEDIATE

1. Struttura della rete

Andremo ad esporre la configurazione della rete lato **hardware**, utilizzeremo **Cisco Packet Tracer** per realizzare la rete e valuteremo i vantaggi della configurazione.

2. Configurazione della rete

Utilizzeremo **VirtualBox** per simulare l'ambiente di rete utile a svolgere i test richiesti (nel nostro caso tra un client e il Web Server).

3. Svolgimento dei test richiesti

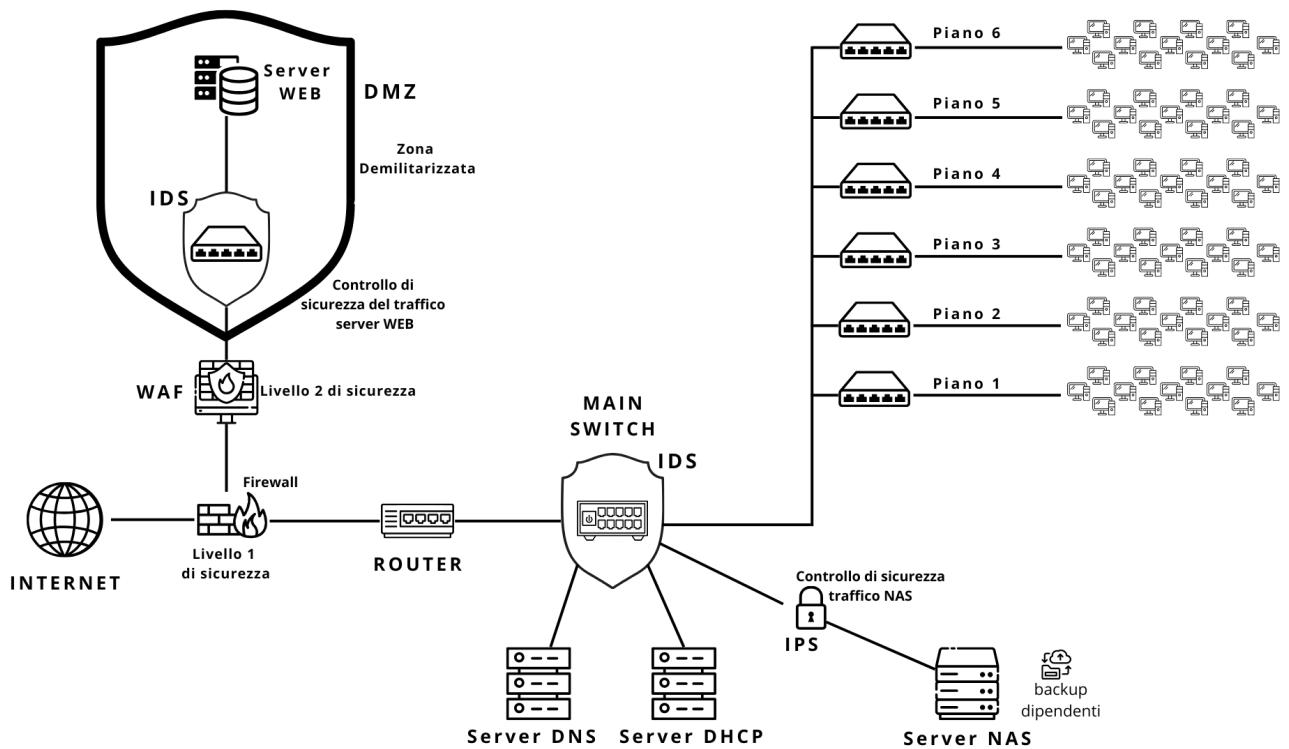
Dimostreremo i metodi che abbiamo utilizzato, spiegandone il funzionamento e dimostrandone l'efficacia

4. Conclusioni, preventivo e raccomandazioni

Concluderemo la spiegazione fornendo vari preventivi, che garantiranno prestazioni e sicurezza proporzionali alla spesa. Forniremo inoltre raccomandazioni inerenti le configurazioni di rete e l'ottimizzazione della stessa.

PROGETTO RETE THETA :

1. Struttura della rete

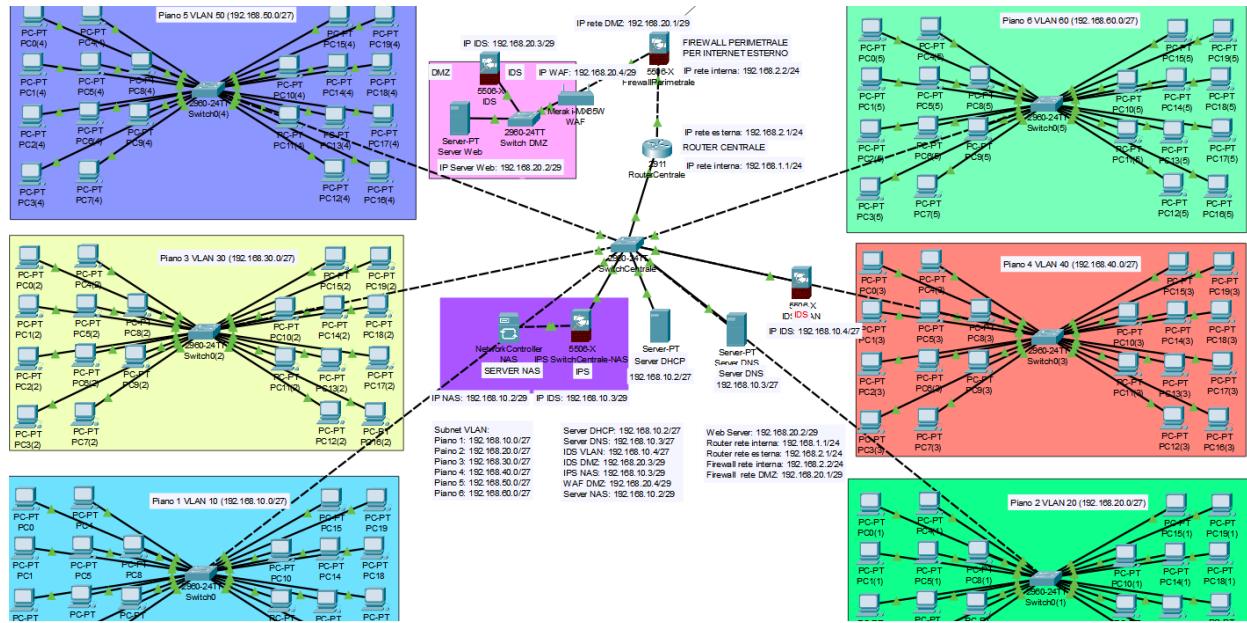


Per creare la nostra rete ci serviremo per un supporto visivo fornитoci da **Canva** per la realizzazione stilizzata del progetto e di **Cisco Packet Tracer con i relativi indirizzi IP locali** per avere una visualizzazione più tecnica dell'infrastruttura.

Per realizzare l'infrastruttura di rete della compagnia Theta, abbiamo progettato una soluzione **scalabile, sicura ed efficiente.**

La rete è stata segmentata per ottimizzare la gestione dei nostri dispositivi e garantire un alto livello di sicurezza.

1.1 Segmentazione della Rete



L'infrastruttura è stata suddivisa in sei piani, ciascuno dei quali rappresenta una **VLAN** (Virtual LAN). Questa suddivisione garantisce:

- **Separazione del traffico:** ogni VLAN isola i 20 computer presenti sul piano, garantendo una maggiore sicurezza e riducendo il traffico broadcast, migliorando marginalmente le prestazioni delle VLAN stesse.
- **Gestione semplificata:** l'uso di VLAN ci dà la possibilità di intervenire sulla rete andando a creare nuove reti in caso di una modifica futura, senza dover intervenire direttamente sul cablaggio.
- **Isolamento di attacchi:** maggior facilità nell'isolare la rete oggetto di attacchi in modo da prevenire la propagazione tempestivamente verso le altre reti.

1.2 Hardware Necessario

-Switch per piano e uno switch centrale: Ogni piano dispone di uno switch dedicato per collegare i 20 dispositivi richiesti dalla consegna.

Gli switch sono collegati a loro volta ad uno **switch centrale** in modo da fornire una configurazione di maggior flessibilità in relazione al tipo di edificio (che non conosciamo al momento).

Per ragioni tecniche e/o di sicurezza abbiamo implementato lo switch centrale in modo da consentire all'azienda il libero posizionamento di router e DMZ al piano desiderato, magari dotato di un ambiente **fisicamente isolato** nel quale i dipendenti non autorizzati non possono avere accesso.

avremo in tutto un totale di **7 SWITCH**. (escludendo lo switch nella DMZ, come vedremo successivamente)

-**Router centrale**: Per il routing delle comunicazioni tra reti/e interna ed esterna.

-**Firewall perimetrale**: Posizionato tra il router e la connessione a Internet, protegge l'intera rete da attacchi esterni.

-**Web Server (Metasploitable) + switch DMZ**: Inserito nella **DMZ (Demilitarized Zone)**, per garantire un accesso sicuro dall'esterno. Questo posizionamento isola il server dal resto della rete interna, limitando eventuali danni in caso attacchi o traffico illecito ivi diretto.

-**NAS (Network Attached Storage)**: connesso al router centrale, per fornire uno spazio di archiviazione accessibile a tutti i computer e facilitare la trasmissione dati eventuale tra le VLAN senza appesantire il traffico sul dispositivo di routing.

-**Server DHCP**: connesso allo switch centrale, ci servirà per fornire dinamicamente gli indirizzi a tutti i dispositivi della rete Theta.

-**Server DNS**: anch'esso connesso allo switch centrale. Ci sarà fondamentale per la traduzione degli indirizzi IP interni/esterni destinatari delle richieste da parte dei nostri client, inoltre tramite immagazzinamento della cache renderà più snello e veloce il sopradetto traffico.

-**WAF (web application firewall)**: Utilizzeremo un WAF, per isolare la nostra DMZ in quanto abbiamo scelto di localizzarvi solo ed esclusivamente il Web Server. Un dispositivo IPS ad esempio, andrebbe a fare un'azione di blocco più generica e non limitata a HTTP/HTTPS o applicazioni web, (come nel caso del WAF), operando sul livello 3 e andando a rallentare inutilmente il traffico.

-**IDS/IPS (Intrusion Detection/Prevention Systems)**: Collocati strategicamente per monitorare e proteggere i flussi di dati sia dall'interno che dall'esterno:

- **IPS** Tra il SERVER NAS e lo switch Centrale (traffico interno diretto al NAS).
- **IDS** collegato allo switch nella DMZ
- **IDS** All'interno della rete interna, per identificare attività sospette tra le VLAN.

nell'immagine sottostante riprodotta con CISCO vediamo i componenti appena menzionati:

1 switch piano + switch centrale

2 router centrale

3 firewall perimetrale

4 server web

5 server nas

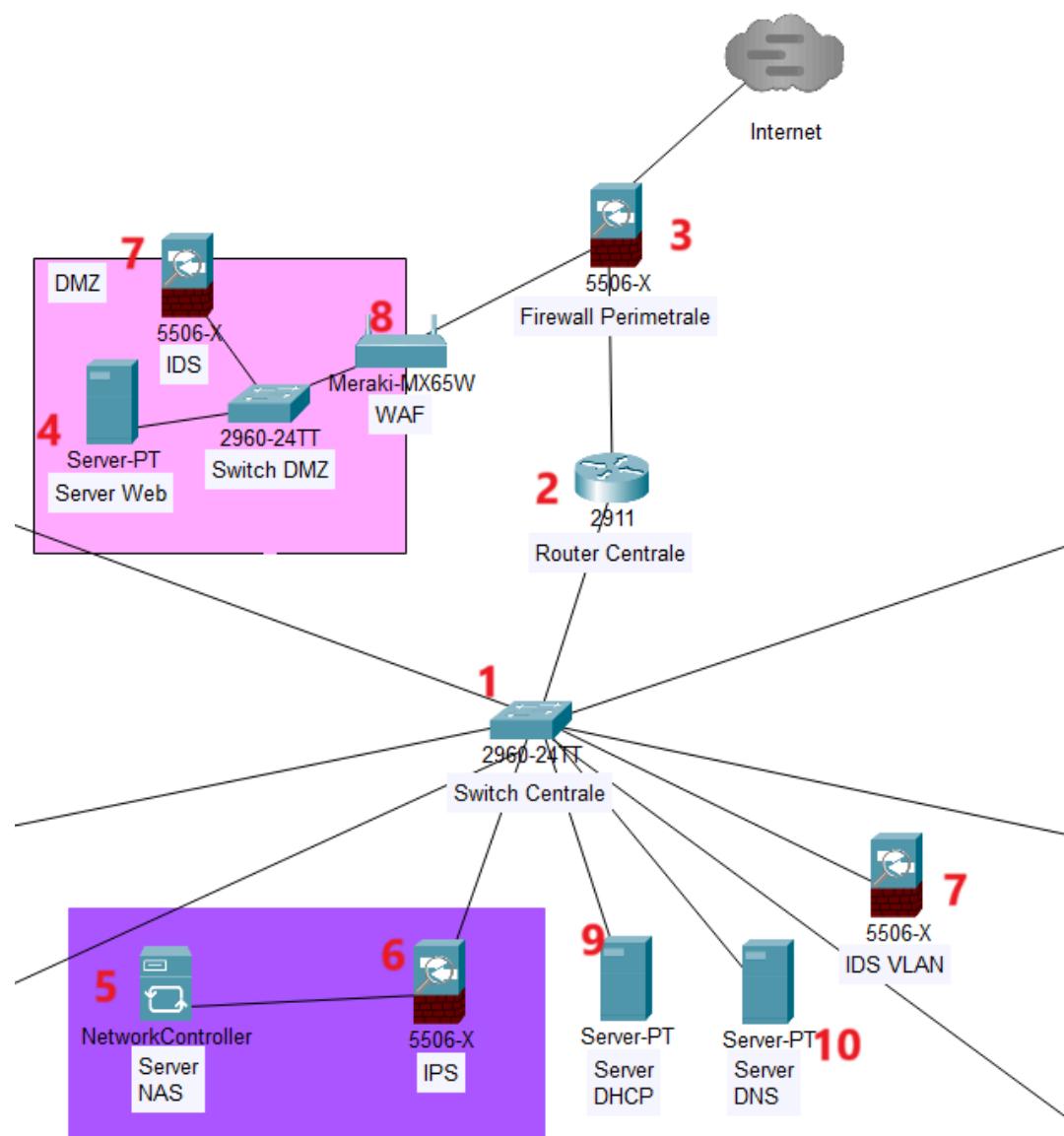
6 IPS

7 IDS

8 WAF

9 Server DHCP

10 DNS Server



1.3 Precisazioni e scelte affrontate:

Analizziamo più da vicino la struttura di rete fornita per motivare le nostre scelte.

Abbiamo già illustrato il motivo del posizionamento di un ulteriore switch centrale, e del collocamento di un IDS che ci permetta di monitorare traffico sospetto tra le VLAN senza però andare a creare rallentamenti inutili.

Rimane cruciale isolare nella nostra **DMZ** il web server, che ora risulta accessibile solo attraverso un doppio controllo da parte di:

- FIREWALL** per il blocco generale del traffico non configurato verso la DMZ
 - WAF** per la configurazione di specifiche regole di blocco che impediscono agli attacchi mirati di raggiungere il server web e i suoi applicativi (HTTP/HTTPS)
- e un ulteriore verifica del traffico Web sospetto avviene tramite **l'IDS** collegato per appunto allo switch della DMZ.

Abbiamo inoltre ritenuto opportuno utilizzare un ulteriore sistema di blocco **IPS** per scongiurare eventuali attacchi interni o comportamenti illeciti verso il server **NAS**, questo per mettere in sicurezza i backup ivi immagazzinati e i dati sensibili in esso contenuti.

Il server NAS può essere infatti collegato allo switch centrale in modo da poter essere ulteriormente isolato anch'esso **sul livello fisico**, sfruttando la flessibilità derivante dalla presenza dello switch centrale.

1.4 Ottimizzazione delle subnet

Per dare maggior completezza al nostro progetto abbiamo identificato una lista di possibili sotto reti per ognuno dei nostri dispositivi come si evince dalla tabella seguente:

Firewall - Rete DMZ	192.168.20.1/29
Firewall - Rete interna	192.168.2.2/24
Web server	192.168.20.2/29
IDS DMZ	192.168.20.3/29
WAF	192.168.20.4/29
RouterCentrale - Rete esterna	192.168.2.1/24
RouterCentrale - Rete interna	192.168.1.1/24
Server NAS	192.168.10.2/29
IPS NAS	192.168.10.3/29
IPS LAN	192.168.1.3/24
IDS SwitchCentrale	192.168.1.2/24
Server DHCP	192.168.1.4/24
Server DNS	192.168.1.5/24
VLAN 10 - Piano 1	192.168.10.0/27
VLAN 20 - Piano 2	192.168.20.0/27
VLAN 30 - Piano 3	192.168.30.0/27
VLAN 40 - Piano 4	192.168.40.0/27
VLAN 50 - Piano 5	192.168.50.0/27
VLAN 60 - Piano 6	192.168.60.0/27

Scelte e consigli per le subnet:

Abbiamo deciso di creare una VLAN per ogni piano in modo tale che ognuno sia isolato rispetto agli altri e la comunicazione avvenga solo tramite uno switch di livello 3 (switch centrale)

Consigliamo una **subnet /27** in modo tale da limitare il numero di **Host** e altri eventuali dispositivi che si possono connettere ad ogni switch.

Abbiamo utilizzato uno switch con 48 porte ma abbiamo limitato il numero massimo a 30 dispositivi permettendo un ampliamento di 10 dispositivi su ogni piano.

Per motivi di sicurezza invece abbiamo deciso di usare delle **subnet /29** per la rete del **NAS** e della **DMZ**. Essendo dei dispositivi di fondamentale importanza per l'azienda, abbiamo creato delle zone di sicurezza separate limitando a 6 il numero totale di indirizzi per i dispositivi nella rete. In questo modo, il nostro progetto permette di isolare gli apparati strategici sia fisicamente che a livello di connessione di rete.

I server **DNS e DHCP** non sono stati inseriti all'interno di una VLAN per facilitarne il raggiungimento da tutti i piani. Tuttavia rimane comunque possibile modificare questa scelta, includendoli in una delle VLAN create in base alle esigenze di utilizzo.

N.B.

Nella configurazione analizzata è il **router centrale** a permettere il routing tra le nostre reti VLAN e le comunicazioni con l'esterno, ma siccome abbiamo incluso nel preventivo uno switch di 3° livello, possiamo affidare il ruolo di instradamento del traffico interno allo switch centrale velocizzando ulteriormente il traffico tra reti interne.

2. Configurazione Della Rete - Theta Network

Per configurare la rete avremo bisogno di creare un ambiente virtuale nel quale faremo comunicare il nostro client

CLIENT 1 (Macchina virtuale Kali Linux)

con la macchina che andrà a rappresentare il Web Server dell'azienda

WEB SERVER (Macchina virtuale Metasploitable)

Utilizzeremo anche un firewall, utile a instradare il traffico tra le due reti: vedremo infatti successivamente come utilizzare reti differenti ci garantisca uno svolgimento efficiente lato sicurezza. La macchina che opererà come firewall sarà una macchina PFSense

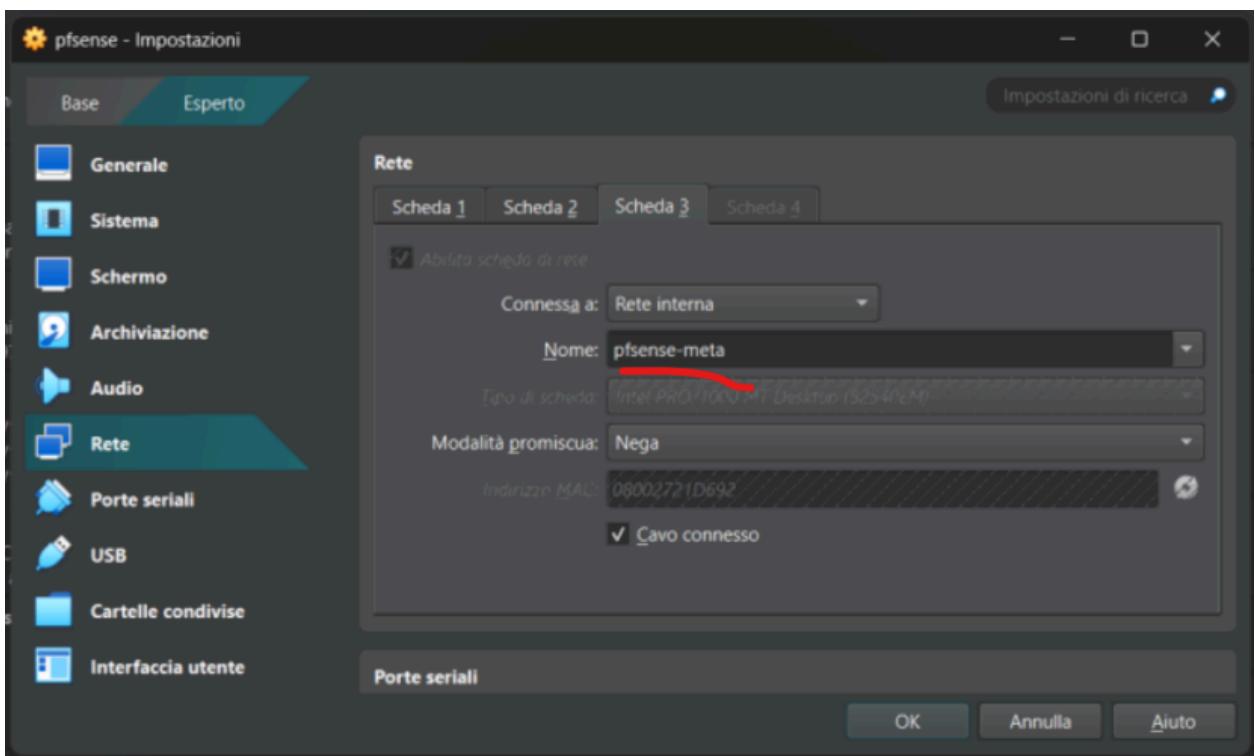
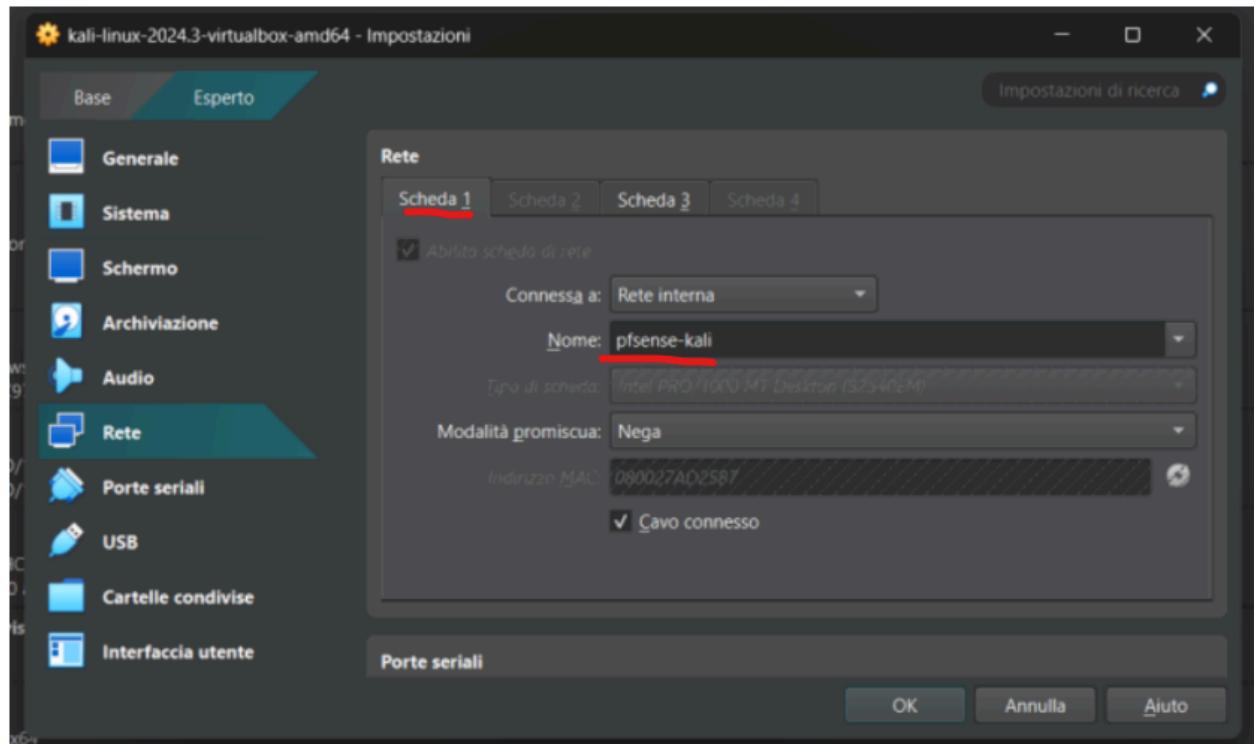
FIREWALL (Macchina virtuale pfSense)

2.1

Cominciamo aprendo VirtualBox e settando le configurazioni per le schede di rete delle nostre macchine. Chiamiamo quindi le nostre due reti interne

- **pfsense-kali (CLIENT-FIREWALL)**
- **pfsense-meta (SERVER-FIREWALL)**

e configuriamo PFSense in modo che sia collegato ad entrambe.



2.2

Procediamo andando a scegliere le configurazioni di rete per ciascuna delle macchine:

CLIENT:

- IP :**192.168.10.2**

-SUBNET: **255.255.255.224**

-GATEWAY: **192.168.10.1**

WEB SERVER

-IP **192.168.20.2**

-SUBNET **255.255.255.248**

-GATEWAY:**192.168.20.1**

FIREWALL

-WAN: (non rilevante)

-LAN1: **192.168.10.1**

-LAN2:**192.168.20.1**

```
(kali㉿kali)-[~]
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
            inet6 ::1/128 scope host noprefixroute
                valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:00:27:00:c2:32 brd ff:ff:ff:ff:ff:ff
        inet 192.168.10.2/27 brd 192.168.10.31 scope global noprefixroute eth0
            valid_lft forever preferred_lft forever
            inet6 fe80::a00:27ff:fe0a:c232/64 scope link proto kernel_ll
                valid_lft forever preferred_lft forever
```

```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:6d:4b:19
          inet addr:192.168.20.2 Bcast:192.168.20.7 Mask:255.255.255.248
          inet6 addr: fe80::a00:27ff:fe6d:4b19/64 Scope:Link
                  UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
                  RX packets:24 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:70 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:1000
                  RX bytes:2296 (2.2 KB) TX bytes:10792 (10.5 KB)
          Base address:0xd020 Memory:f0200000-f0220000
```

```
*** Welcome to pfSense 2.7.2-RELEASE (amd64) on pfSense ***
```

```
WAN (wan)      -> em0      -> v4/DHCP4: 10.0.2.15/24
LAN (lan)      -> vtne0     -> v4: 192.168.10.1/27
LAN2 (opt1)    -> vtne1     -> v4: 192.168.20.1/29
```

- 0) Logout (SSH only)
- 1) Assign Interfaces
- 2) Set interface(s) IP address
- 3) Reset webConfigurator password
- 4) Reset to factory defaults
- 5) Reboot system
- 6) Halt system
- 7) Ping host
- 8) Shell
- 9) pfTop
- 10) Filter Logs
- 11) Restart webConfigurator
- 12) PHP shell + pfSense tools
- 13) Update from console
- 14) Enable Secure Shell (sshd)
- 15) Restore recent configuration
- 16) Restart PHP-FPM

Possiamo verificare il nostro setup di rete testando le comunicazioni tra le macchine ed inviando un ping in entrambi i versi.

```
[kali㉿kali)-[~]
$ ping 192.168.10.1 (Local Database)
PING 192.168.10.1 (192.168.10.1) 56(84) bytes of data.
64 bytes from 192.168.10.1: icmp_seq=1 ttl=64 time=0.960 ms
64 bytes from 192.168.10.1: icmp_seq=2 ttl=64 time=0.900 ms
64 bytes from 192.168.10.1: icmp_seq=3 ttl=64 time=0.905 ms

msfadmin@metasploitable:~$ ping 192.168.20.1
PING 192.168.20.1 (192.168.20.1) 56(84) bytes of data.
64 bytes from 192.168.20.1: icmp_seq=1 ttl=64 time=2.15 ms
64 bytes from 192.168.20.1: icmp_seq=2 ttl=64 time=0.980 ms
```

ATTENZIONE:

Se andiamo a modificare l'interfaccia WAN e inseriamo manualmente un IP statico, dovremo anche riconfigurare le regole di routing per pfSense, nell'immagine sottostante vediamo le regole pre-impostate di pfSense per la WAN Default

The screenshot shows a network configuration interface with the following details:

- Gateways:**

Name	Default	Interface	Gateway	Monitor IP	Description	Actions
WAN_DHCPC		WAN	10.0.2.2	10.0.2.2	Interface WAN_DHCPC Gateway	
WAN_DHCPC6		WAN	fe80::2%em0	fe80::2%em0	Interface WAN_DHCPC6 Gateway	
- Default gateway:**
 - Default gateway IPv4: Automatic
 - Select a gateway or failover gateway group to use as the default gateway.
 - Default gateway IPv6: Automatic
 - Select a gateway or failover gateway group to use as the default gateway.
- Buttons:** Save, Add, Num Lock On.

3. Testing Della Rete Theta Network

Per prima cosa occupiamoci della consegna che ci richiede la creazione di un programma in Python per inviare richieste

HTTP (GET, POST, PUT, DELETE)

al web server ([metasploitable](#)) e verificare le risposte.

3.1.1

Per prima cosa apriamo la macchina CLIENT e creiamo in essa il nostro file

'invioRichieste.py'

che andremo ad eseguire in modo da poter avere un doppio riscontro:

- **Da TERMINALE di kali.** A tal proposito sfrutteremo anche la libreria **tabulate** per una visualizzazione più User Friendly e '**os**' per ripulire la console all'avvio del programma
- **Da Burp Suite.** Per poter ottenere questo riscontro avremo bisogno anche di configurare indirizzo IP e porta per BurpSuite (default: **127.0.0.1:8080**) in modo da fornirgli la possibilità di intercettare le richieste.

Proxy listeners

Burp Proxy uses listeners to receive incoming HTTP requests from your browser. You will need to configure your browser to use one of the listeners as its proxy server.

Add	Running	Interface	Invisible	Redirect	Certificate	TLS Protocols	Support HTTP/2
<input type="button" value="Edit"/>	<input checked="" type="checkbox"/>	127.0.0.1:8080			Per-host	Default	<input checked="" type="checkbox"/>
<input type="button" value="Remove"/>							

Project setting

Andiamo a creare un codice composto da quattro funzioni che sfruttino i metodi messi a disposizione dalla libreria **request** per ciascuna delle richieste.

```
1 #librerie per le impaginazioni
2 from tabulate import tabulate #libreria per la tabella
3 import os #libreria per pulire la console a inizio programma
4 os.system('clear') #pulizia console
5
6 import requests #libreria per le richieste HTML
7
8 url = "http://192.168.20.2/phpMyAdmin"
9
10 burp = {
11     "http": "http://127.0.0.1:8080",    #porta HTTP di ascolto su burp
12
13 }
14
15 #funzione richiesta GET
16 def getRequest(url):
17     responseGET=requests.get(url, proxies=burp) #invio richiesta GET
18     return responseGET
19
20 #Funzione richiseta POST
21 def postRequest(url):
22     responsePOST=requests.post(url, proxies=burp)
23     return responsePOST
24
25 #Funzione richiesta PUT
26 def putRequest(url):
27     responsePUT=requests.put(url, proxies=burp)
28     return responsePUT
29 #Funzione richiesta DELETE
30 def deleteRequest(url):
31     responseDELETE=requests.delete(url, proxies=burp)
32     return responseDELETE
33
34 #tabella che restituisce gli status codes
35 table=[
36     ["GET request status code", getRequest(url).status_code],
37     ["POST request status code", postRequest(url).status_code],
38     ["PUT request status code", putRequest(url).status_code],
39     ["DELETE request status code", deleteRequest(url).status_code]
40 ]
41 print(tabulate(table, tablefmt="grid"))
42
```

Alla riga 10 si può notare la porzione di codice che ci permetterà di analizzare il traffico tramite Burp Suite.

Utilizzeremo l'url di metasploitable (**WEB SERVER**) nella sezione ‘**phpMyAdmin**’ come destinazione delle nostre richieste.

Nella sezione di codice a **riga 41** diamo al nostro programma il comando per stampare sul terminale le risposte ottenute e precedentemente inserite in ‘**table**’ (**riga 35**)

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. Under the 'HTTP history' tab, there is a table of captured requests:

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes
17	http://192.168.20.2	GET	/phpMyAdmin			301	596	HTML		301 Moved Permanently	
18	http://192.168.20.2	GET	/phpMyAdmin/			200	5041	HTML		phpMyAdmin	
19	http://192.168.20.2	POST	/phpMyAdmin			301	595	HTML		301 Moved Permanently	
20	http://192.168.20.2	GET	/phpMyAdmin/			200	5041	HTML		phpMyAdmin	
21	http://192.168.20.2	PUT	/phpMyAdmin			301	595	HTML		301 Moved Permanently	
22	http://192.168.20.2	PUT	/phpMyAdmin/			200	5041	HTML		phpMyAdmin	
23	http://192.168.20.2	DELETE	/phpMyAdmin			301	595	HTML		301 Moved Permanently	
24	http://192.168.20.2	DELETE	/phpMyAdmin/			200	5041	HTML		phpMyAdmin	

Below the table, a terminal window shows the results of the script execution:

```
kali㉿kali:~/tools
File Actions Edit View Help
+-----+-----+
| GET request status code | 200 |
+-----+-----+
| POST request status code | 200 |
+-----+-----+
| PUT request status code | 200 |
+-----+-----+
| DELETE request status code | 200 |
+-----+-----+
(kali㉿kali)-[~/tools]
$
```

A red checkmark is placed next to the terminal window.

Eccoci le nostre risposte con valore ‘**200**’ segnalandoci che l’esito della richiesta è andato a buon fine.

Da Burp Suite notiamo anche con maggior precisione che ogni richiesta presenta il suo **redirect**.

(**status code:301**) → redirect effettuato di default per consentire una connessione più sicura, la richiesta HTTP viene infatti ri-effettuata modificando il protocollo da HTTP a HTTPS consentendo la crittografia dei dati e la minor esposizione degli stessi, oltre che verificando l’autenticità del nostro server WEB.

3.2.1

Per la **seconda consegna** dovremo creare un programma in Python per eseguire una scansione delle porte sui dispositivi di rete, verificando la sicurezza e l’accessibilità delle varie porte di comunicazione.

3.2.2

Procediamo come prima con la creazione del nostro nuovo programma Python sulla macchina CLIENT, chiameremo il programma

scanPorte.py

```
# Impostazioni target
target = "192.168.20.2"      #target
portrange = "1-500"           #porte
timeout = 1                   #timeout per la connessione

lowport = int(portrange.split('-')[0])    #prendo la porta piu' bassa nella stringa di input
highport = int(portrange.split('-')[1])    #prendo la porta piu' alta nella stringa di input

print(f"Scanning host {target} from port {lowport} to port {highport}\n")

portOpen = []      #lista di porte aperte
portFiltered = []  #lista di porte filtrate
portClosed = []    #lista di porte chiuse
```

il programma sfrutta la libreria ‘**socket**’ per fornirci i metodi necessari ad effettuare lo scan delle porte di rete.

N.B

Abbiamo utilizzato un range di porte 1-500 per evitare che la presentazione/simulazione in live richieda troppo tempo, ovviamente, è sufficiente modificare il valore affidato a ‘portrange’ se si desidera ottenere una scansione di un range di porte differente

In queste prime righe di codice andiamo a impostare il target del nostro programma, ovvero la macchina **Metasploitable che funge da WEB Server**.

Definiamo degli array vuoti nei quali inseriremo i valori della nostra scansione, e verificheremo se queste siano:

- **OPEN**
- **CLOSED**
- **FILTERED**

```

# Scansione delle porte
for port in range(lowport, highport +1):

    #Apertura del socket per controllo della porta
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    #Timeout per la connessione
    s.settimeout(timeout)

    #tento la connessione includendo le eccezioni
    status = s.connect_ex((target, port))

    if status == 0: #porta aperta
        portOpen.append(port) #metto alla fine della lista
    elif status==111: #porta chiusa
        portClosed.append(port)
    else:
        portFiltered.append(port)

s.close() #chiusura del socket

# Output finale
print("\n\nScanning complete.")

if len(portOpen) > 0:
    print(f"Total open ports: {len(portOpen)}")
    input("Press ENTER to show the list of open ports ... \n")
    for port in portOpen: #stampa le porte aperte
        print(f"*** Port {port} - OPEN ***")
    for port in portFiltered: #stampa le porte filtrate
        print (f"Port {port} - FILTERED")
    #for port in portClosed: #stamoa le porte chiuse
    # print (f"Port {port} - closed")
else:
    print("No open ports found.")

```

In questo caso sfrutteremo un ciclo ‘**for**’ per valutare tutte le porte del range selezionato e controlleremo lo status con un ‘**if/elif/else**’ per verificare lo status della porta.

Nell’*Output finale* diamo al nostro utente un feedback di ‘scanning completato’ e gli forniamo il numero di porte **OPEN** rilevate.

Tramite la pressione del tasto ENTER può successivamente visualizzare le porte aperte rilevate in ordine numerico.

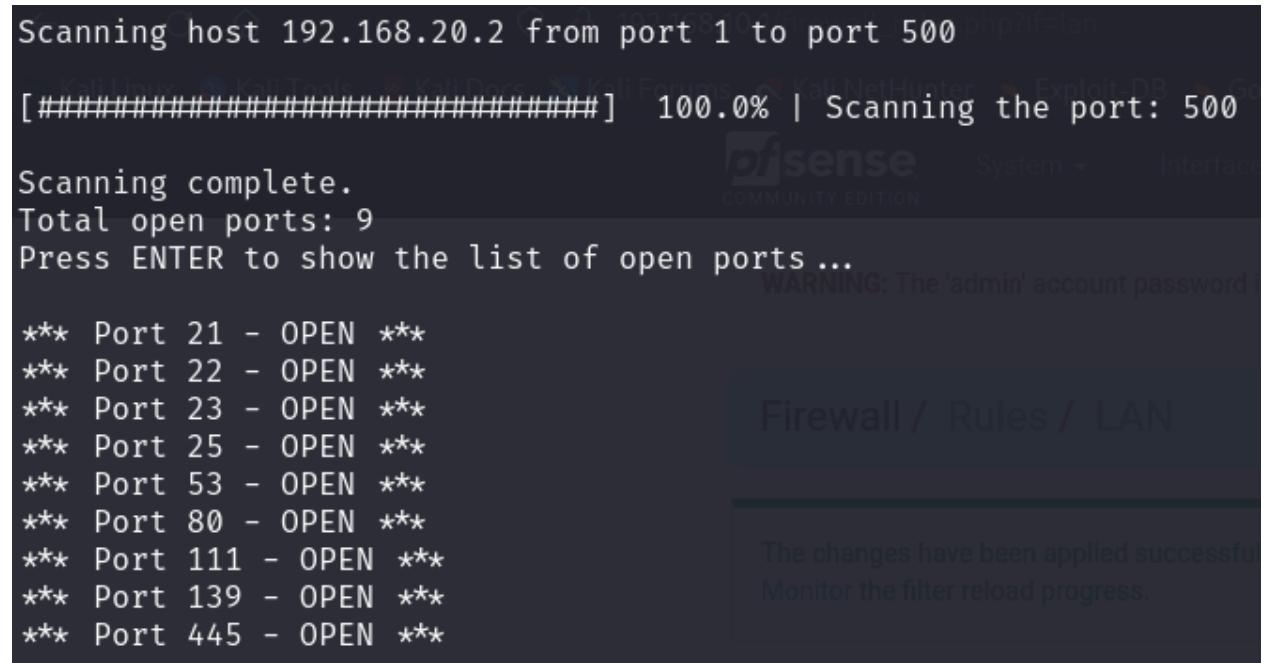
N.B.

Nel codice del nostro programma è stata commentata la sezione in cui vengono stampate le porte chiuse per una resa maggiormente chiara dell'output su terminale.

3.2.3

Andiamo ora a lanciare il nostro programma e verifichiamo l'efficacia tramite alcuni test:

- 1) Come possiamo notare il programma ha scansionato le porte dalla 1 alla 500 mostrandoci **ESCLUSIVAMENTE** le porte OPEN



```
Scanning host 192.168.20.2 from port 1 to port 500
[#####] 100.0% | Scanning the port: 500
Scanning complete.
Total open ports: 9
Press ENTER to show the list of open ports ...
*** Port 21 - OPEN ***
*** Port 22 - OPEN ***
*** Port 23 - OPEN ***
*** Port 25 - OPEN ***
*** Port 53 - OPEN ***
*** Port 80 - OPEN ***
*** Port 111 - OPEN ***
*** Port 139 - OPEN ***
*** Port 445 - OPEN ***
```

- 2) Andiamo a verificare effettivamente se il programma rileva le porte filtrate; per farlo impostiamo delle regole di blocco sulle porte 80 e 443 tramite il nostro **firewall** simulato dalla macchina virtuale PFSense.

Floating	WAN	LAN	LAN2								
Rules (Drag to Change Order)											
	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input checked="" type="checkbox"/>	15/120 Kib	*	*	*	LAN Address	80	*	*		Anti-Lockout Rule	
<input type="checkbox"/>	0/0 B	IPv4 TCP	192.168.10.3	*	192.168.20.2	443 (HTTPS)	*	none			
<input type="checkbox"/>	0/0 B	IPv4 TCP	192.168.10.3	*	192.168.20.2	80 (HTTP)	*	none			
<input type="checkbox"/>	0/54 Kib	IPv4	*	LAN subnets	*	*	*	*	none	Default allow LAN to any rule	
<input type="checkbox"/>	0/0 B	IPv6	*	LAN subnets	*	*	*	*	none	Default allow LAN IPv6 to any rule	

Add Add Delete Toggle Copy Save

Come vediamo ora il traffico viene bloccato dal **firewall** e quindi il nostro programma dovrà dare come risposta '**FILTERED**' sulle porte selezionate

```
Scanning host 192.168.20.2 from port 1 to port 500
[#####] 100.0% | Scanning the port: 50
0

Scanning complete.
Total open ports: 8
Press ENTER to show the list of open ports ...

*** Port 21 - OPEN ***
*** Port 22 - OPEN ***
*** Port 23 - OPEN ***
*** Port 25 - OPEN ***
*** Port 53 - OPEN ***
*** Port 111 - OPEN ***
*** Port 139 - OPEN ***
*** Port 445 - OPEN ***
Port 80 - FILTERED
Port 443 - FILTERED
```

Firewall / Rules / LAN

The changes have been applied successfully. Monitor the filter reload progress.

- 3) Per concludere la nostra fase di testing mostriamo ora come il codice, una volta rimosso il commento ci permetta anche di eventualmente visualizzare le porte CLOSED inerenti al nostro target:

```
Scanning host 192.168.20.2 from port 1 to port 500
[########################################] 100.0% | Scanning the port: 500
Scanning complete.
Total open ports: 8
Press ENTER to show the list of open ports ...

*** Port 21 - OPEN ***
*** Port 22 - OPEN ***
*** Port 23 - OPEN ***
*** Port 25 - OPEN ***
*** Port 53 - OPEN ***
*** Port 111 - OPEN ***
*** Port 139 - OPEN ***
*** Port 445 - OPEN ***
Port 80 - FILTERED
Port 443 - FILTERED
Port 1 - closed
Port 2 - closed
Port 3 - closed
Port 4 - closed
```

3.2.4

A seguire una lista di **TUTTE** le porte aperte rilevate sulla macchina metasploitable (Web Server):

Port 21 - OPEN Port 22 - OPEN Port 23 - OPEN

Port 25 - OPEN Port 53 - OPEN Port 80 - OPEN

Port 111 - OPEN Port 139 - OPEN Port 445 - OPEN

Port 512 - OPEN Port 513 - OPEN Port 514 - OPEN

Port 1099 - OPEN Port 1524 - OPEN Port 2049 - OPEN

Port 2121 - OPEN Port 3306 - OPEN Port 3632 - OPEN

Port 5432 - OPEN Port 5900 - OPEN Port 6000 - OPEN

Port 6667 - OPEN Port 6697 - OPEN Port 8009 - OPEN

Port 8180 - OPEN Port 8787 - OPEN Port 35005 - OPEN

Port 43612 - OPEN Port 51293 - OPEN Port 52773 - OPEN

3.2.5 Raccomandazioni per la sicurezza della rete

Forniamo ora una lista di **suggerimenti e raccomandazioni** in merito alle porte aperte riscontrate:

Porta	Sicurezza	Raccomandazioni
21	Da configurare	Configurare per l'utilizzo di FTPS invece di FTP
22	Sì	
23	No	Disabilitare e utilizzare SSH su porta 22
25	Da configurare	Configurare per l'utilizzo di TLS per proteggere i dati e configurare autenticazione a due fattori
53	Da configurare	Disabilitare
80	No	Utilizzare HTTPS su porta 443
111	Da configurare	Configurare il Firewall per impedire l'accesso dall'esterno (internet) e limitarlo solo agli utenti autorizzati dall'interno della rete
139	No	Disabilitare
445	Da configurare	Configurare con autenticazione forte (Kerberos), con Access Control Lists per stabilire chi possa accedere e con crittografia delle comunicazioni per proteggere i dati
512	No	Disabilitare e utilizzare SSH su porta 22
513	No	Disabilitare e utilizzare SSH su porta 22
514	No	Disabilitare e utilizzare SSH su porta 22
443	Da aprire	Utilizzare la porta 443 per HTTPS al posto della porta 80 (HTTP)
1099		Java Remote Method Invocation, consente l'accesso da remoto a oggetti java.
1524	Da configurare	Configurare il Firewall per limitare l'accesso ai soli IP della rete interna autorizzati, con autenticazione forte e crittografia SSL per protezione dei dati. Considerare l'uso di NFSv4 invece di NFS
2049	Da configurare	Disabilitare se non necessaria dato che usa NFS che è un protocollo non sicuro altrimenti usare NFSv4
2121	Da configurare	Configurare per l'utilizzo di FTPS invece di FTP
3306	Da configurare	La porta intrinsecamente non è sicura quindi va disabilitata altrimenti renderla sicura tramite SSL/TLS
3632	Da configurare	Configurare correttamente dato che può essere usata come vettore d'attacco visto che è un servizio che permette di eseguire codice su macchine remote
5432	Da configurare	Non esporre la porta su internet ed usare crittografia e autenticazione forte
5900	Da configurare	La porta non è sicura dato che viene usata per accedere ad un altro computer da remoto, se necessaria non esporre su internet ed usare crittografia
6000	No	Disabilitare se non necessaria dato che trasmette dati in chiaro ed ha accesso remoto non autenticato
6667	Da configurare	Disabilitare se non necessaria altrimenti renderla sicura usando TLS/SSL oppure limitando l'accesso ad indirizzi IP fidati
6697	Si	Più sicura della porta precedente dato che usa la crittografia ma comunque si consiglia di disabilitarla se non in uso
8009	Da configurare	Non è intrinsecamente insicura ma dipende dalla configurazione
8180	Da configurare	Disabilitare se non necessaria altrimenti usare HTTPS e limitare l'accesso
8787	Da configurare	Disabilitare se non necessaria altrimenti usare HTTPS e limitare l'accesso
35005	Da configurare	Disabilitare se non necessaria altrimenti limitare l'accesso ed usare crittografia
43612	Da configurare	Disabilitare se non necessaria altrimenti limitare l'accesso ed usare crittografia
51293	Da configurare	Disabilitare se non necessaria altrimenti limitare l'accesso ed usare crittografia
52773	Da configurare	Disabilitare se non necessaria altrimenti limitare l'accesso ed usare crittografia

Si raccomanda di chiudere tutte le porte non necessarie, in base all'utilizzo che si intende fare del web server:

- Per il trasferimento di dati sul web si raccomanda di utilizzare la porta 443, per il protocollo HTTPS;
- L'utilizzo di SSL/TLS per la crittografia dei dati trasmessi tra Client-Server offre una maggiore protezione dei dati;
- L'utilizzo di certificati digitali garantisce l'autenticazione del server;
- Il ricorso a meccanismi di hashing assicura l'integrità dei dati.
- Per la gestione del web server da remoto si raccomanda di utilizzare la porta 22, per il protocollo SSH, assicurandosi di implementare alcune misure:
 - Configurare il Firewall per consentire l'accesso solamente agli utenti autorizzati
 - Implementare l'autenticazione a due fattori
 - Utilizzare chiavi SSH per l'autenticazione
 - Configurare l'IDS della DMZ per monitorare le attività sospette/illeggibili

3.3 —BONUS—

Per la realizzazione dell'**esercizio Bonus** andremo a creare un programma capace di catturare il socket di rete su una specifica porta da noi designata.

Tenteremo poi la connessione con il comando **netcat** da parte della macchina **Web Server (metasploitable)** e verificheremo la cattura eseguita

Creiamo quindi il nostro programma cattura socket importando la libreria '**socket**' e chiamiamolo '**catturaSocket.py**'.

```
GNU nano 8.2                               catturaSocket.py

import socket

SERVER_IP = "192.168.10.2" #Inserisci l'indirizzo IP del server
SERVER_PORT = 44444 #Inserisci il numero di porta

#Creo il mio socket
mySocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

#Assegno al socket indirizzo e porta
mySocket.bind((SERVER_IP, SERVER_PORT))

#Ascolto le connessioni in ingresso nel socket creato
mySocket.listen(1) #1 massimo numero di connessioni in coda

print("Server started, waiting for connections ...")

#Accettiamo la connessione in ingresso verso il server
connection, address = mySocket.accept() #Address e' l'indirizzo IP del client in connessione

print("Client connected with address:", address)
print("Waiting for data ...")

while 1:
    data = connection.recv(1024) #1024 grandezza del buffer da ricevere in byte
    if not data: break
    connection.sendall(b"-- Message received -- \n")
    #Stampo i dati decodificandoli in utf-8
    print(data.decode("utf-8"))
connection.close()
|
```

il codice appena visto ci permette di mettere in ascolto il nostro HOST per eventuali connessioni sulla porta designata (**44444**)

```
msfadmin@metasploitable:~$ netcat 192.168.10.2 44444
Catturiamo il socket!
-- Message received --
```

Lanciando il comando **netcat** dal nostro web server andiamo a verificare se il nostro programma ha catturato la connessione:

```
(kali㉿kali)-[~/Documents/Build_Week1]
└─$ python catturaSocket.py
Server started, waiting for connections ...
Client connected with address: ('192.168.20.2', 57219)
Waiting for data ...
Catturiamo il socket!
```

Vediamo infatti il risultato che ci mostra il Client connesso con l'IP statico assegnato al Web Server e la porta su cui ha stabilito la connessione, in questo esempio è la porta numero **57219**

3.4 —SUPER BONUS—

Ultimo bonus Build Week:

Scrivere un programma in Python che utilizzi **scapy** per catturare e analizzare il traffico di rete. Il programma deve:

1. Catturare i pacchetti in tempo reale su un'interfaccia di rete specificata.
2. Filtrare i pacchetti in base al protocollo (ad esempio: TCP, UDP, ICMP).
3. Visualizzare i dettagli dei pacchetti, inclusi indirizzi IP sorgente/destinazione, porte e protocollo utilizzato.
4. Salvare i pacchetti catturati in un file *.pcap* per un'analisi successiva tramite Wireshark o strumenti analoghi.

3.4.1

Per poter adempiere alla consegna dell'esercizio abbiamo scritto un programma '**sniffer.py**' servendoci della libreria '**scapy**' .

Lo **sniffer** cattura i pacchetti in tempo reale su un'interfaccia di rete specificata, li filtra in base al protocollo, li elabora in una funzione di *callback*,

ovvero una funzione passata come argomento a un'altra funzione, che viene eseguita (richiamata) al verificarsi di un evento specifico o al termine di un'operazione.

Infine salva i pacchetti catturati in un file *.pcap* per ulteriori analisi.

```

from scapy.all import *

intr = "eth0"          #select the network adapter
protocolFilter = "icmp"    #select the protocol (tcp, udp icmp)

output_file = "capturedPackets.pcap"  #name of file
captured_packets = []

print(f"sniffing in {protocolFilter} . . .")

#Funzione di callback per elaborare i pacchetti catturati
def packet_callback(packet):
    print("-" * 80)
    print (packet.summary())
    print("-" * 80)

    captured_packets.append(packet)

#sniffing dei pacchetti
sniff(prn=packet_callback, store=0, count=0, filter=(protocolFilter), iface=intr)
#store=0    non salva i pacchetti
#count=0    prende pacchetti infiniti

#Dopo aver terminato lo sniffing, salva i pacchetti in un file .pcap
wrpcap(output_file, captured_packets, append=False)
#append=False sovrascrive il file all'esecuzione
#append=True aggiunge dati senza sovrascrivere

```

Questo è un primo esempio di intercettazione dei pacchetti con protocollo **ICMP**.

(I pacchetti ICMP (Internet Control Message Protocol) sono messaggi usati per diagnosticare reti, segnalare errori o comunicare informazioni, come ping e tracciamento del percorso)

tramite la funzione di **callback** andiamo a spostare i risultati della nostra analisi all'interno dell'array **captured_packets**, che successivamente salviamo in un file *.pcap* come richiesto dall'esercizio.

N.B

!!! IL PROGRAMMA DEVE ESSERE ESEGUITO COME SUPER USER !!!

Nella funzione di sniffing inseriamo i seguenti parametri:

store = 0 count = 0

per avere uno sniffing su un numero illimitato di pacchetti ed evitare il salvataggio automatico degli stessi.

Abbiamo voluto aggiungere alla funzione di salvataggio dei pacchetti su file, il parametro `append = False` per far sì che il file venga ri-sovrascritto ad ogni sniffing. È sufficiente cambiare la voce in `True` se si desidera ottenere un maggior numero di analisi separate, senza sovrascrivere il file `.pcap` ad ogni avvio del programma.

3.4.2

Ora andiamo a lanciare il nostro programma modificando il protocollo per effettuare svariati test e verifichiamo i risultati:

Invieremo in tutti e tre gli esempi successivi delle richieste dalla macchina Web server con protocolli:

-**ICMP** tramite l'invio di un ping

-**UDP** con il l'invio del comando:

```
echo "Test THCP": nc <IP CLIENT> 8080
```

-**TCP** con il l'invio del comando:

```
echo "Test THCP": nc -u <IP CLIENT> 8080
```

Andremo inoltre ad analizzare il traffico tramite Wireshark aprendo i file `.pcap` che il nostro programma ha salvato ad ogni esempio.

- 1) Effettuiamo lo sniffing tramite protocollo ICMP; Come vediamo ci da prima un errore sulla richiesta che contiene il protocollo errato (TCP/UDP). Mentre quando lanciamo le richieste di ping '**sniffiamo**' correttamente i pacchetti trasmessi.

The screenshot shows two windows. The top window is a terminal window titled 'root@kali: /home/kali/tools' displaying network traffic analysis. It lists several ICMP packets captured on the interface:

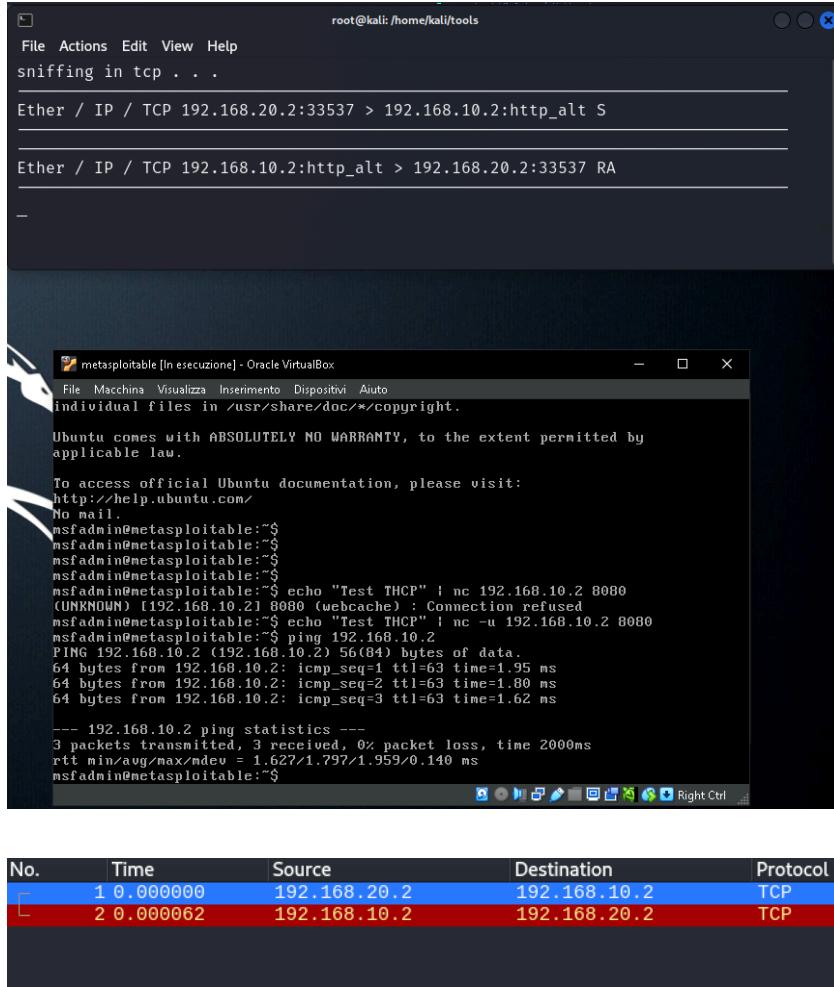
- Ether / IP / ICMP 192.168.10.2 > 192.168.20.2 dest-unreachable / IPerror or / UDPerror / Raw
- Ether / IP / ICMP 192.168.20.2 > 192.168.10.2 echo-request 0 / Raw
- Ether / IP / ICMP 192.168.10.2 > 192.168.20.2 echo-reply 0 / Raw
- Ether / IP / ICMP 192.168.20.2 > 192.168.10.2 echo-request 0 / Raw
- Ether / IP / ICMP 192.168.10.2 > 192.168.20.2 echo-reply 0 / Raw
- Ether / IP / ICMP 192.168.20.2 > 192.168.10.2 echo-request 0 / Raw
- Ether / IP / ICMP 192.168.10.2 > 192.168.20.2 echo-reply 0 / Raw

The bottom window is titled 'metasploitable [In esecuzione] - Oracle VirtualBox' and shows a terminal session on the 'msfadmin' user. It includes commands like 'echo', 'nc', and 'ping' along with their output.

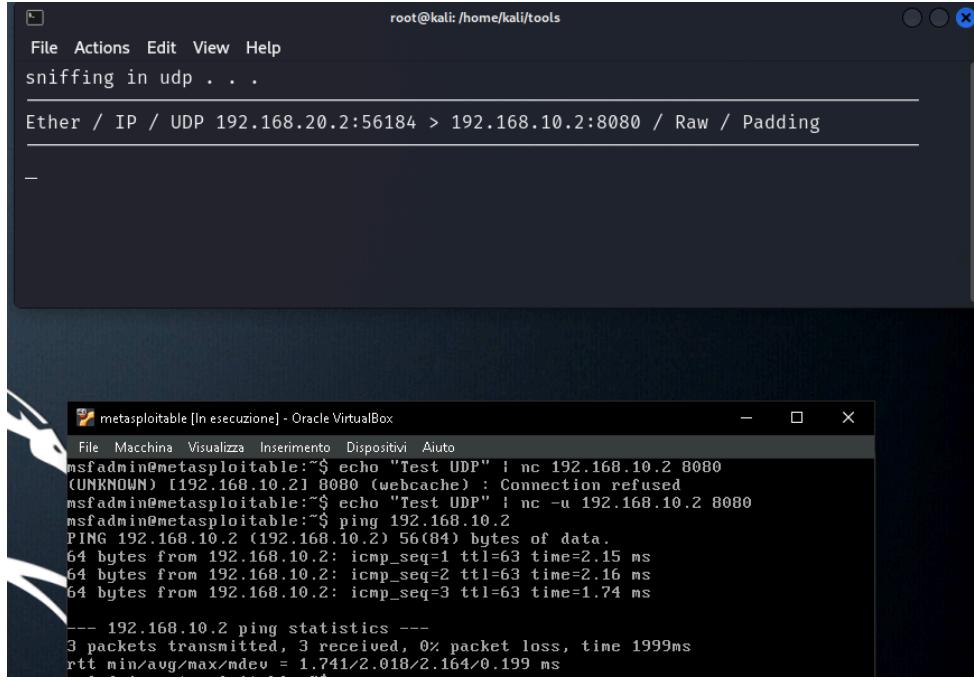
Frame ID	Source IP	Destination IP	Protocol	Description
1	0.0.0.0.0	192.168.10.2	ICMP	80 Destination unreachable (Port unreachable)
2	13.602118	192.168.20.2	ICMP	98 Echo (ping) request id=0xae12, seq=1/256, ttl=63 (reply in 3)
3	13.662193	192.168.10.2	ICMP	98 Echo (ping) reply id=0xae12, seq=1/256, ttl=64 (request in 2)
4	14.661248	192.168.20.2	ICMP	98 Echo (ping) request id=0xae12, seq=2/512, ttl=63 (reply in 5)
5	14.661344	192.168.10.2	ICMP	98 Echo (ping) reply id=0xae12, seq=2/512, ttl=64 (request in 4)
6	15.661943	192.168.20.2	ICMP	98 Echo (ping) request id=0xae12, seq=3/768, ttl=63 (reply in 7)
7	15.662044	192.168.10.2	ICMP	98 Echo (ping) reply id=0xae12, seq=3/768, ttl=64 (request in 6)

2) Lanciamo ora le richieste per effettuare il test con il nostro sniffer settato sul protocollo **TCP** e come notiamo dall'immagine seguente, intercettiamo solo la comunicazione del protocollo desiderato, divisa in due parti **SYN - RST ACK**.

SYN, RST e ACK sono segnali nel protocollo TCP usati per stabilire connessioni: SYN avvia la connessione, ACK conferma i dati, e RST chiude connessioni anomale



3) Lanciamo infine richieste per effettuare il test con il nostro sniffer settato sul protocollo **UDP** e come possiamo vedere in questo caso viene intercettata esclusivamente la connessione del protocollo selezionato.



No.	Time	Source	Destination	Protocol
1	0.000000	192.168.20.2	192.168.10.2	UDP

4. Preventivi e raccomandazioni finali

Procediamo ora con il fornire un preventivo inerente alla struttura e alle raccomandazioni fornite nel punto **1, cioè la struttura della rete:**

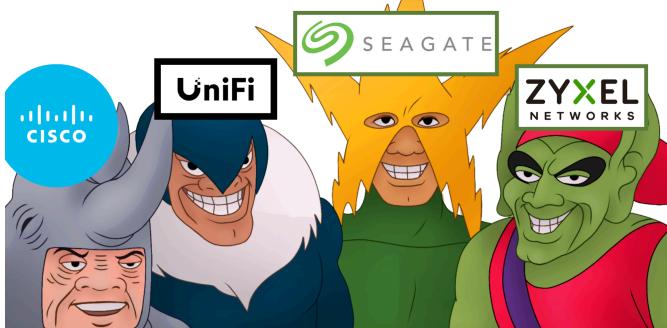
Valuteremo differenti configurazioni per **marca e brand**, ma prima alcune raccomandazioni specifiche sulle attrezzature sotto illustrate:

Oggetto	Quantità	Note
IPS	2	
IDS	1	
Server Firewall	1	
Server NAS	1	Da aggiungere gli HDD
Server Webserver	1	
Server DHCP	1	
Server DNS	1	
Switch 48 porte	7	Una o due porte gigabit
Switch livello 3	1	Tutte gigabit+
Router	1	

Prendiamo nota del fatto che:

- Includiamo nel nostro preventivo per il server NAS anche i dischi HDD (Backup, ecc.). Abbiamo pensato di configurare gli HDD in **RAID 6** (Redundant Array of Independent Disks), che usa una doppia parità e tollera la perdita di 2 dischi. Degli 8 dischi a preventivo, 1 è di scorta e 7 saranno inseriti nel nas di cui saranno usufruibili 5, per un totale di $5 \times 16\text{TB} = 80\text{ TB}$, che corrispondono a circa 670 GB per utente.
- Gli switch selezionati hanno tutti un numero di porte superiore a 24 per garantire flessibilità e scalabilità nell'eventualità di una futura ristrutturazione, riconfigurazione delle VLAN in LAN o eventuali segmentazioni necessarie.
- Opteremo per uno switch **livello 3** a 24+ porte, tutte ad almeno 1Gb Ethernet/SFP+, per garantire al nostro switch centrale la maggior velocità possibile nello smistamento del traffico di rete.
- Trascureremo nei nostri preventivi il costo degli Host (e di relative periferiche), in quanto in base alle esigenze dell'azienda il prezzo di questi ultimi può oscillare di cifre notevolmente significative.

Forniamo ora una configurazione personalizzata per una rapporto qualità VS spesa mediamente consigliato per aziende della dimensione indicata



Oggetto	#	Quantità	Prezzo	Brand	Modello
IPS		1	€ 709,00	Cisco	FirePower 1010 NGFW
IDS		2	€ 709,00	Cisco	FirePower 1010 NGFW
Firewall		1	€ 1.192,00	Cisco	FirePower 2110 NGFW
Dischi NAS		6	€ 300,00	Seagate	IronWolf
Server NAS		1	€ 459,00	Unifi	UNAS PRO
Server Web/DHCP/DNS		3	€ 1.400,00	HPE	ProLiant ML30
SSD 1 TB SATA		4	€ 100,00	Samsung	870 EVO 1TB
Switch 48 porte		7	€ 275,00	Zyxel	GS 1900-48
Switch centrale livello 3		1	€ 1.540,00	Cisco	Catalyst 9300-24U
Cablaggio FTP + RJ45 cat6		10	€ 100,00	Vultec	Cat 6 305m 23AWG
Router		1	€ 450,00	Unifi	Gateway Pro
Consigli d'acquisto extra:					
Extra: UPS		1	€ 370,00	Atlantis-Land	LinePower 1502 PRO 1,5kW
Extra: switch al piano di riserva		1	€ 275,00	Zyxel	GS 1900-48
Extra: switch centrale livello 3		1	€ 1.540,00	Cisco	Catalyst 9300-24U
Costo WAF Azure: 0,45€/ora -> circa 300€ al mese					

€ 15.093,00	Totale
€ 17.278,00	Totale con extra



Oggetto	#	Quantità	Prezzo	Brand	Modello
IPS		1	€ 709,00	Cisco	FirePower 1010 NGFW
IDS		2	€ 709,00	Cisco	FirePower 1010 NGFW
Firewall		1	€ 1.192,00	Cisco	FirePower 2110 NGFW
Server NAS/DHCP/DNS		3	€ 1.400,00	HPE	ProLiant ML30
Dischi HDD		7	€ 300,00	Seagate	IronWolf Pro 16tb
Server Webserver		1	€ 2.500,00	Cisco	IPCE-ACOL-SVR
Switch 30+ porte		7	€ 1.296,00	Cisco	Catalyst C9200-48P
Switch centrale livello 3		1	€ 1.540,00	Cisco	Catalyst 9300-24U
Cablaggio FTP + RJ45 cat6 (300m)		10	€ 70,00	Mysmartshop	300 metri cat6
Router		1	€ 3.550,00	Cisco	2911/K9
Consigli d'acquisto extra:					
Extra: UPS		1	€ 370,00	Atlantis-Land	LinePower 1502 PRO 1,5kW
Extra: switch al piano di riserva		1	€ 1.063,00	Cisco	WS-C2960L-24TS-LL
Extra: switch centrale livello 3		1	€ 1.540,00	Cisco	Catalyst 9300-24U
Costo WAF Azure: 0,45€/ora -> circa 300€ al mese					

€ 26.981,00	Totale
€ 29.954,00	Totale con extra

Punti di Forza:

-
- **Affidabilità e qualità:** Cisco è noto per la robustezza hardware e il software avanzato. Ideale per ambienti aziendali critici come quello della compagnia Theta.
 - **Sicurezza:** Offre soluzioni avanzate come firewall, IDS/IPS e sistemi di controllo accessi ben integrati.
 - **Scalabilità:** Perfetto per un'azienda in crescita, grazie a una vasta gamma di dispositivi.
 - **Supporto e aggiornamenti:** Assistenza tecnica e aggiornamenti costanti per garantire la sicurezza.

Debolezze:

Costo elevato: Cisco è significativamente più costoso rispetto la concorrenza.

Complessità di configurazione: Richiede personale qualificato per l'implementazione e la manutenzione.



Oggetto	#	Quantità	Prezzo	Brand	Modello
IPS		2	€ 345,00	Unifi	Dream Machine Pro
IDS		1	€ 345,00	Unifi	Dream Machine Pro
Server Firewall		1	€ 345,00	Unifi	Dream Machine Pro
Server NAS		1	€ 459,00	Unifi	UNAS-PRO
Dischi HDD		7	€ 300,00	Seagate	IronWolf Pro 16tb
Server Webserver		3	€ 1.400,00	HPE	HPE ProLiant ML30
SSD 1 TB SATA		4	€ 100,00	Samsung	870 EVO 1TB
Switch 48+4 porte		7	€ 600,00	Unifi	USW PRO MAX 48
Switch centrale livello 3		1	€ 819,00	Unifi	Hi-Capacity Aggregation (lv3)
Router		1	€ 345,00	Unifi	Dream Machine Pro
Cablaggio FTP + RJ45 cat6 (300		10	€ 70,00	Mysmartshop	300 metri cat6
Cablaggio SFP		14	€ 45,50	Unifi	10G Long-Range Direct Attach Cable
Adattatori SFP to RJ45		2	€ 45,50	Unifi	10G SFP to RJ45 adapter
Consigli d'acquisto extra:					
Extra: UPS		1	€ 370,00	Atlantis-Land	LinePower 1502 PRO 1,5kW
Extra: switch al piano di riserva		1	€ 600,00	Unifi	USW PRO MAX 48
Extra: switch centrale di riserva		1	€ 819,00	Unifi	Hi-Capacity Aggregation (lv3)
Extra, cavi SFP di scorta		2	€ 50,00	Unifi	10G Long-Range Direct Attach Cable
Costo WAF Azure: 0,45€/ora -> circa 300€ al mese					

€ 15.331,00	Totale
€ 17.220,00	Totale con extra

Punti di Forza:

-
- **Costo contenuto:** Offre buone prestazioni a un prezzo accessibile, rendendolo ideale per aziende con budget limitati.
 - **Gestione centralizzata:** L'ecosistema Unifi offre un'interfaccia utente intuitiva per configurare e gestire i dispositivi.
 - **Semplicità di configurazione:** Adatto anche a team IT con meno esperienza avanzata rispetto a Cisco.

Debolezze:

- **Limitazioni su reti complesse:** Meno adatto per infrastrutture aziendali altamente scalabili o critiche rispetto a Cisco.
- **Sicurezza meno avanzata:** Le funzionalità di sicurezza sono buone, ma meno raffinate rispetto a Cisco.
- **Affidabilità hardware:** I dispositivi potrebbero non essere robusti quanto quelli Cisco per utilizzi intensivi.

Gli extra:

L'aggiunta degli **extra** alle build proposte migliora significativamente la **resilienza**, la **sicurezza** e la **scalabilità** delle infrastrutture.

- L'**UPS** (gruppo di continuità) garantisce la continuità operativa in caso di interruzioni energetiche, proteggendo dispositivi critici.
- Lo **switch centrale di livello 3 di riserva** è essenziale per garantire il funzionamento continuo della rete in caso di guasto dello switch principale, assicurando che il traffico dati continui a fluire senza interruzioni. La sua presenza riduce significativamente i rischi di downtime, mantenendo operativa l'intera infrastruttura e garantendo così la **continuità e affidabilità** dei servizi critici, anche in situazioni di emergenza.
- Lo **switch di riserva** riduce i rischi di downtime, assicurando che la rete continui a funzionare anche in caso di guasto hardware.
- I **cavi SFP** sono componenti delicati che possono rompersi facilmente se sottoposti a tensioni o manovre brusche. Avere **cavi SFP di scorta** è fondamentale per garantire una rapida sostituzione in caso di danni, evitando interruzioni nella rete e assicurando la continuità della connessione.

Il **WAF su Azure** protegge le applicazioni web da attacchi esterni, aumentando la sicurezza senza la necessità di soluzioni hardware aggiuntive. Sebbene gli extra comportino costi

aggiuntivi, distribuiti nel tempo, rappresentano un investimento essenziale per mantenere una rete robusta, sicura e pronta a crescere con le esigenze aziendali.

Conclusione finale:

1. **Configurazione "mista":** Una soluzione ibrida che offre flessibilità, combinando la gestione semplice di Unifi con la sicurezza e le prestazioni di Cisco, ideale per chi ha bisogno di una rete avanzata ma con una gestione centralizzata.
2. **Configurazione Unifi:** Ideale per chi cerca soluzioni economiche e facili da gestire, senza necessità di infrastrutture avanzate. Perfetto per piccole e medie imprese.
3. **Configurazione Cisco:** Adatta a medie e grandi imprese che necessitano di soluzioni robuste e scalabili, con un forte focus sulla sicurezza e sulle prestazioni.

Se il budget non è un problema e si desiderano prestazioni elevate, la configurazione **Cisco** è la migliore opzione.

Al contrario, una **configurazione Unifi** presenta costi di configurazione inferiori e una sinergia tra gli elementi che riduce la complessità operativa, risultando ideale per chi cerca una rete economica e semplice da gestire nel tempo.

Se invece si cerca un compromesso tra facilità di gestione e scalabilità, la **configurazione "mista"** offre il miglior equilibrio: pur avendo una configurazione iniziale più complicata, permette di ottenere componenti ad alte prestazioni per i "core elements" (elementi fondamentali) della rete, distribuendo nel tempo il costo degli interventi di manutenzione più elevati a causa dell'utilizzo di prodotti di brand diversi.