

# 1 INTRODUZIONE

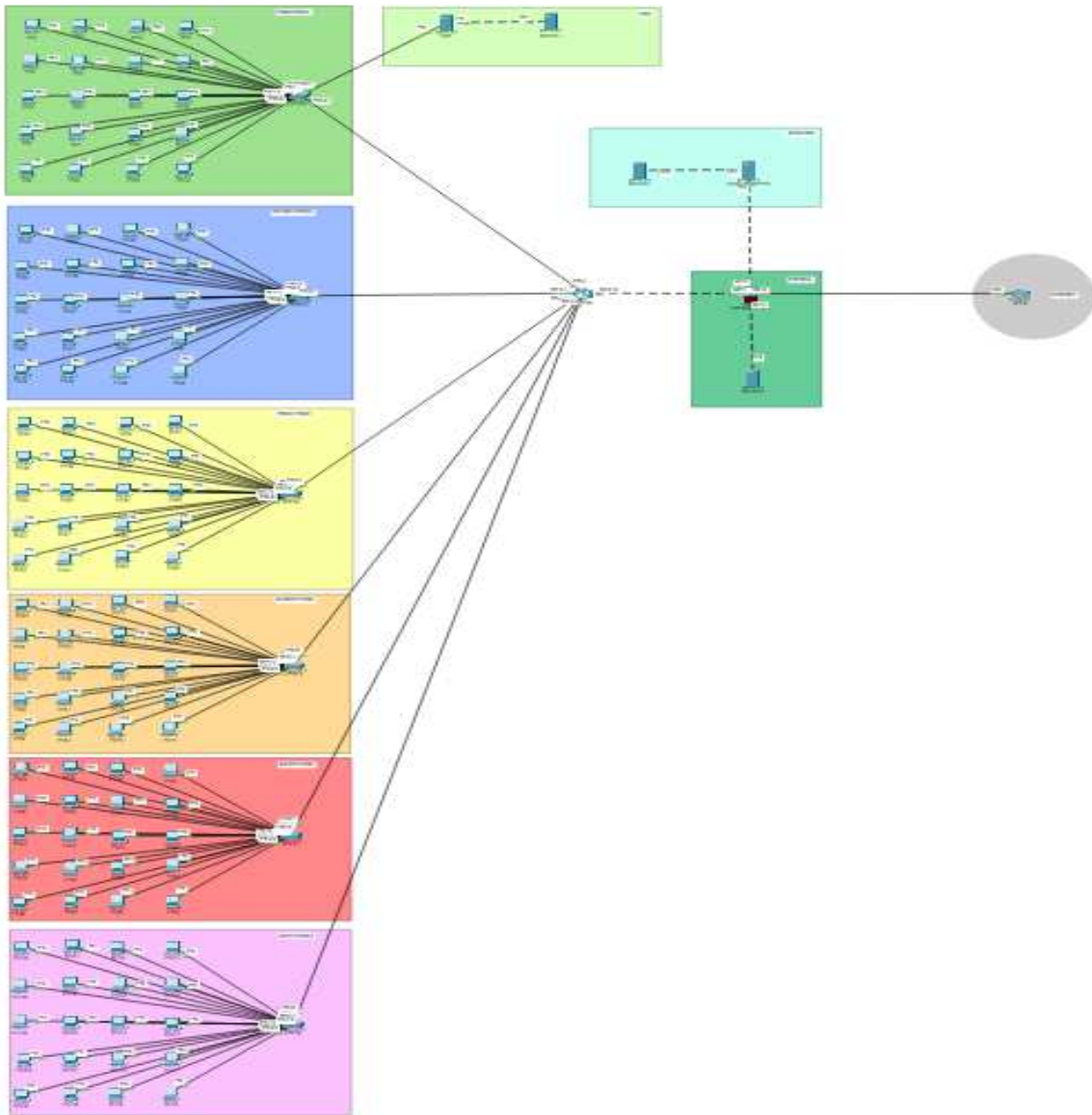
La compagnia Theta ci ha ingaggiati per la progettazione di una rete sicura per la sua azienda e lo sviluppo di un preventivo di spesa.

Le esigenze di Theta sono di sviluppare una rete con 120 computer, suddivisi tra 6 piani (con 20 computer per piano). Ogni piano deve essere collegato ad uno switch dedicato e tutti gli switch ad un router centrale. Al primo piano, inoltre, deve essere presente un NAS che garantisca l'accesso ai dati da parte di tutti gli utenti dell'azienda.

Per quanto riguarda la sicurezza, invece, sono previsti un firewall perimetrale, da posizionare tra il router interno e internet e 3 IDS/IPS per monitorare il traffico interno e prevenire intrusioni.

Infine, un web server (nella DMZ) in grado di fornire l'accesso sicuro sia ad utenti interni che esterni.

## 1.1 Topologia generale della rete



In questa immagine è riprodotta la topologia di rete, elaborata dalla nostra azienda, per la compagnia Theta.

Ci siamo attenuti alle indicazioni fornite, tuttavia, per garantire una maggiore sicurezza, abbiamo effettuato il subnetting della rete, che è stata suddivisa per ogni piano.

**Primo piano (incluso il NAS) --> 192.168.1.1**

**Secondo piano --> 192.168.2.1**

**Terzo piano --> 192.168.3.1**

**Quarto piano --> 192.168.4.1**

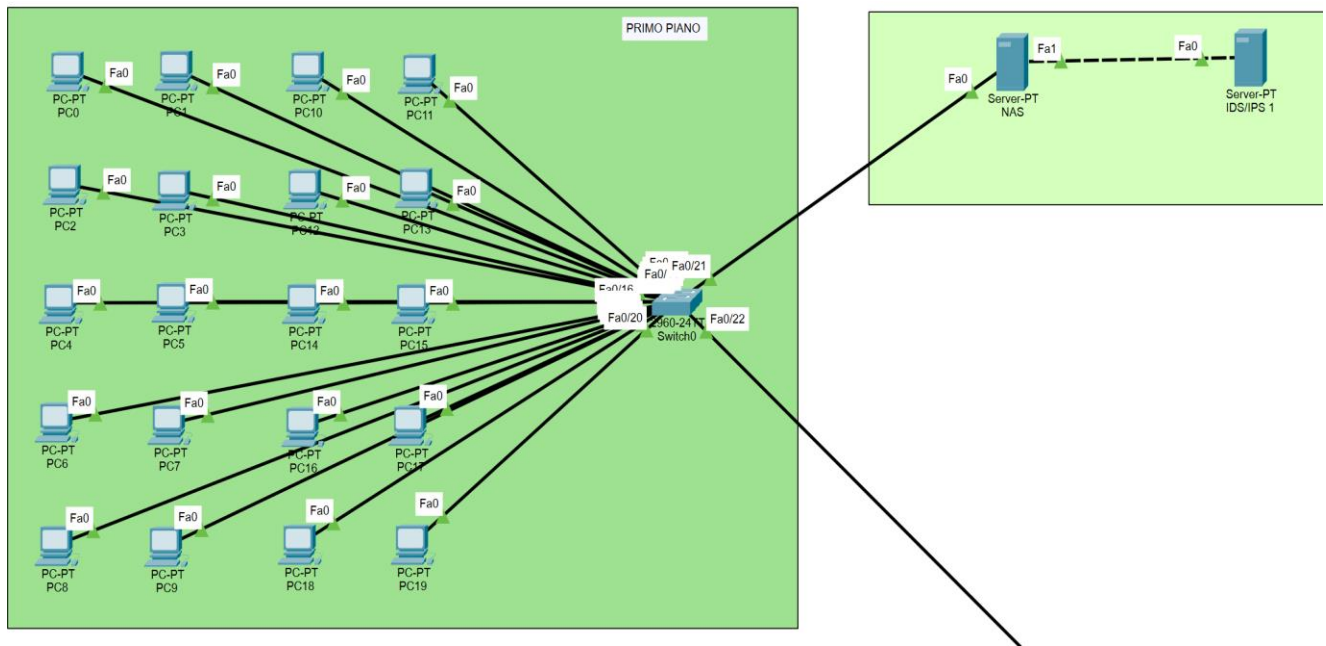
**Quinto piano --> 192.168.5.1**

**Sesto piano --> 192.168.6.1**

Nella nostra configurazione abbiamo impostato degli indirizzi IP statici per tutti gli utenti della rete. Come è possibile notare, ogni switch è collegato fisicamente al router, al quale sono stati aggiunti dei moduli per ampliare le porte disponibili.

Tuttavia, ciò non esclude la possibilità di implementare soluzioni alternative in futuro, come la configurazione di un servizio DHCP, per l'assegnazione automatica degli indirizzi IP.

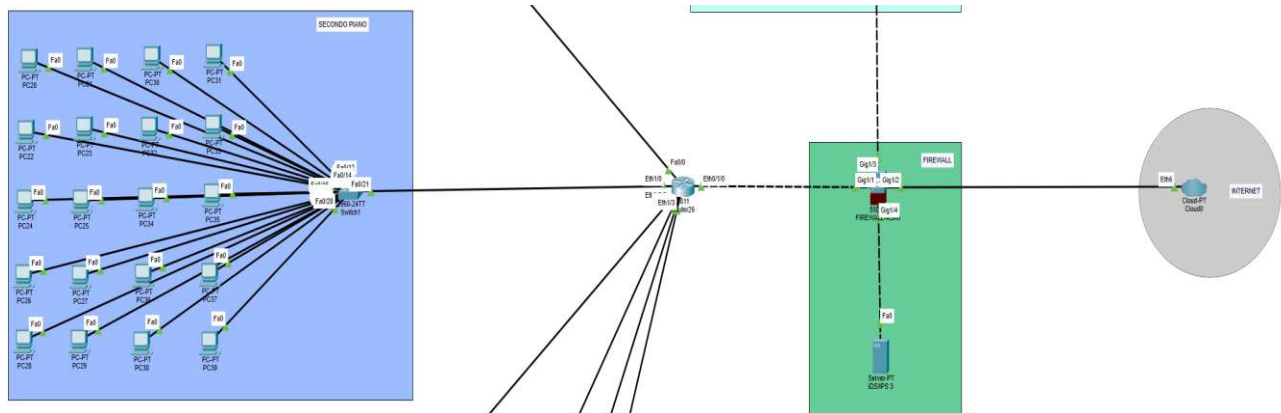
## 1.2 Topologia del primo piano



In questa immagine possiamo vedere l'esempio della topologia prevista per ogni piano. Va notato, però, che, su richiesta della compagnia Theta, al primo piano è anche collegato il NAS con il suo IDS/IPS dedicato, in modo da poter monitorare il traffico verso il NAS e bloccare eventuali richieste di accesso non autorizzate.

I 20 computer ed il NAS utilizzano la stessa subnet.

## 1.3 Firewall

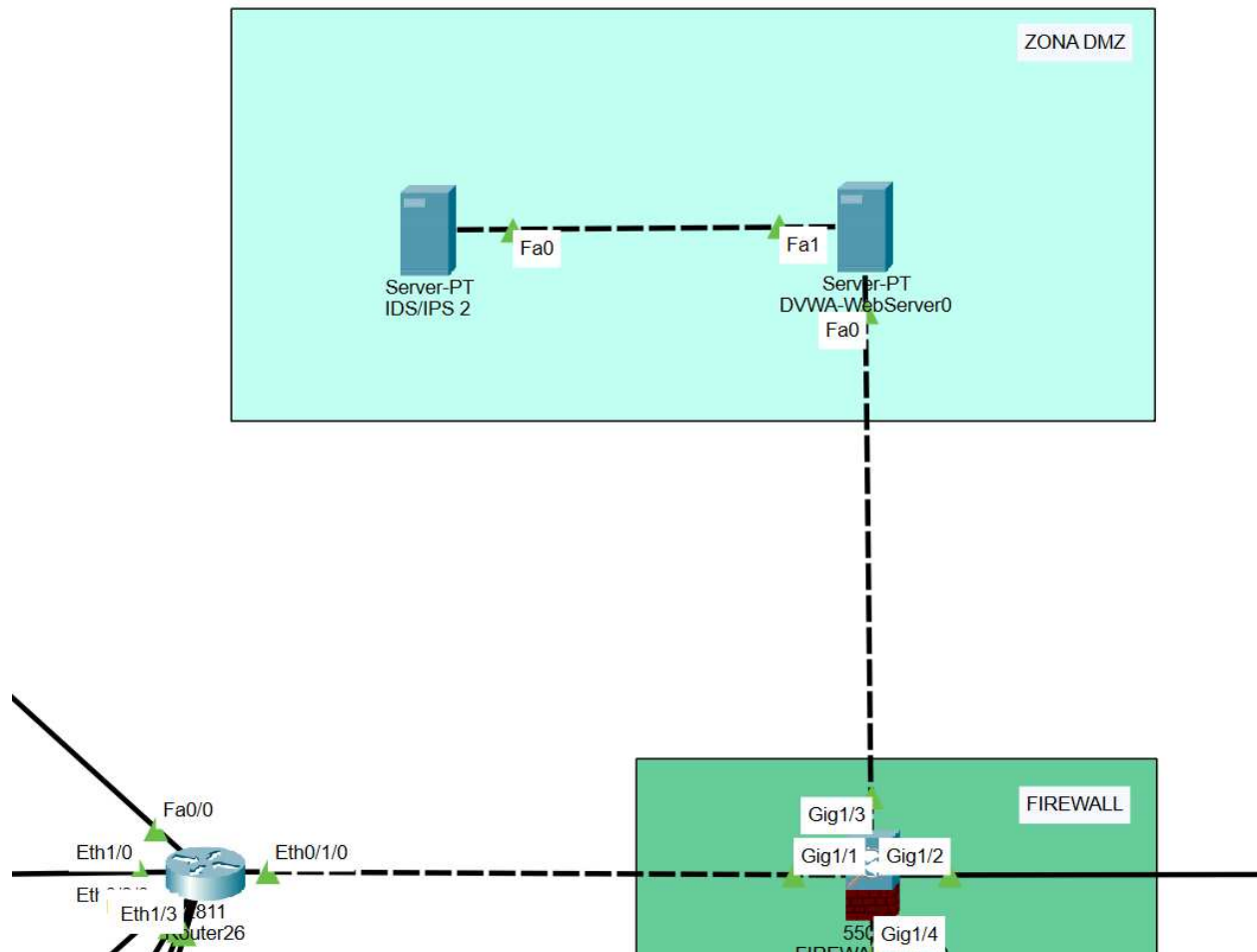


Nell'immagine sopra rappresentata possiamo notare come è stato implementato il firewall, considerando le richieste della compagnia Theta. Va specificato che il firewall utilizzato in questo caso è di tipo fisico e rientra nella configurazione hardware.

Il firewall è stato posizionato tra il router ed internet, per isolare e garantire la sicurezza della rete interna. Inoltre, al firewall è stato affiancato un IDS/IPS, per le ragioni menzionate in precedenza.

Firewall --> 192.168.8.1

## 1.4 DMZ (Zona demilitarizzata)



In questa immagine possiamo vedere la topologia della zona demilitarizzata. È presente il web server con il suo IDS/IPS dedicato.

La zona demilitarizzata si trova all'esterno e posizionata dopo il firewall, per fornire maggiore sicurezza alla rete interna della compagnia Theta.

Anche in questo caso il web server si trova su una sottorete dedicata.

Web server --> 192.168.7.1

# 2 Testing Della Rete Theta Network

2.1 La **prima consegna** 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.

Apriamo la macchina **KaliLinux** e creiamo in essa il nostro file

### 'Richieste.py'

Per questo script abbiamo importato la libreria **"http.client"**

Poi andiamo ad impostare il target ip (**192.168.20.20**) e la porta che il nostro web service espone, ovvero la **80**.

Successivamente utilizziamo la struttura di controllo **"try"** per inviare le richieste dei vari verbi http.

```
1 import http.client
2
3 host = "192.168.20.20" #target ip
4 port = 80 #la macchina metasploitable espone un servizio sulla porta 80
5
6
7 try: #richiesta GET
8     connection = http.client.HTTPConnection(host, port) #gestiamo la connessione verso l'host
9     connection.request('GET', '/') #utilizziamo request per inviare una richiesta HTTP specificando il verbo
10    response = connection.getresponse() #otteniamo la risposta dall'host
11    print("GET:", response.status,) #stampa risposta
12    connection.close() #chiudi la connessione
13 except Exception as e:
14     print("Errore")
15
16 try: #richiesta POST
17     connection = http.client.HTTPConnection(host, port)
18     connection.request('POST', '/')
19     response = connection.getresponse()
20     print("POST:", response.status,)
21     connection.close()
22 except Exception as e:
23     print("Errore")
24
25 try: #richiesta PUT
26     connection = http.client.HTTPConnection(host, port)
27     connection.request('PUT', '/')
28     response = connection.getresponse()
29     print("PUT:", response.status,)
30     connection.close()
31 except Exception as e:
32     print("Errore")
33
34 try: #richiesta DELETE
35     connection = http.client.HTTPConnection(host, port)
36     connection.request('DELETE', '/')
37     response = connection.getresponse()[0]
38     print("DELETE:", response.status,)
39     connection.close()
40 except Exception as e:
41     print("Errore")
```



Una volta avviato lo script otteniamo lo **status code** di tutti e quattro



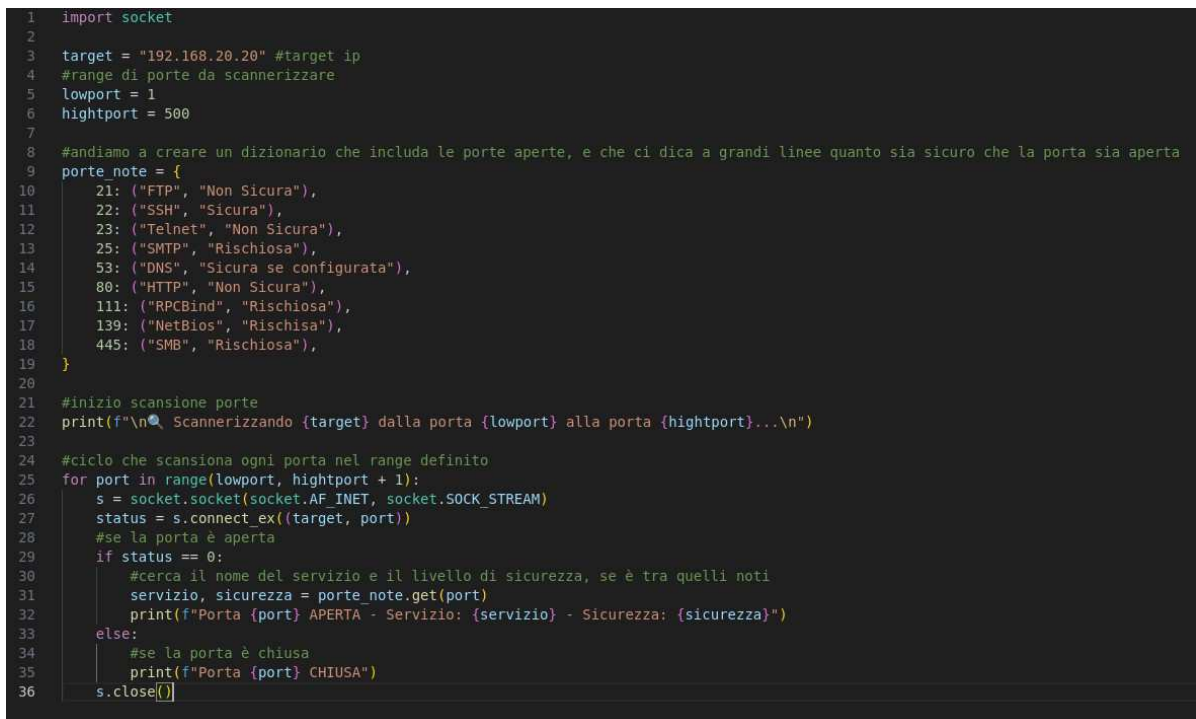
```
kali@kali: ~/Desktop/Buildingweek
File Actions Edit View Help

(kali@kali)-[~/Desktop/Buildingweek]
$ python Richieste.py
GET: 200
POST: 200
PUT: 200
DELETE: 200
```

Tutti e quattro ci danno uno status code **“200”**, quindi il server ha ricevuto le richieste e ha risposto con successo.

2.2 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. Procediamo come prima con la creazione del nostro nuovo programma Python, chiamandolo

## PortScanner.py



```
1 import socket
2
3 target = "192.168.20.20" #target ip
4 #range di porte da scannerizzare
5 lowport = 1
6 hightport = 500
7
8 #andiamo a creare un dizionario che includa le porte aperte, e che ci dica a grandi linee quanto sia sicuro che la porta sia aperta
9 porte_note = {
10     21: ("FTP", "Non Sicura"),
11     22: ("SSH", "Sicura"),
12     23: ("Telnet", "Non Sicura"),
13     25: ("SMTP", "Rischiosa"),
14     53: ("DNS", "Sicura se configurata"),
15     80: ("HTTP", "Non Sicura"),
16     111: ("RPCBind", "Rischiosa"),
17     139: ("NetBios", "Rischiosa"),
18     445: ("SMB", "Rischiosa"),
19 }
20
21 #inizio scansione porte
22 print(f"\n🔍 Scannerizzando {target} dalla porta {lowport} alla porta {hightport}...\n")
23
24 #ciclo che scansiona ogni porta nel range definito
25 for port in range(lowport, hightport + 1):
26     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
27     status = s.connect_ex((target, port))
28     #se la porta è aperta
29     if status == 0:
30         #cerca il nome del servizio e il livello di sicurezza, se è tra quelli noti
31         servizio, sicurezza = porte_note.get(port)
32         print(f"Porta {port} APERTA - Servizio: {servizio} - Sicurezza: {sicurezza}")
33     else:
34         #se la porta è chiusa
35         print(f"Porta {port} CHIUSA")
36     s.close()
```

Importiamo la libreria **“socket”** questo perchè i socket di rete sono utilizzati per scambiare dati tra due computer, una sorgente ed un destinatario e nel nostro script creeremo un socket in ascolto (**s**).

Anche qui andiamo a definire il target ip della macchina da scansionare (sempre **192.168.20.20** di **Metasploitable**).

Andiamo poi a scegliere un range di porte da analizzare, per comodità non andremo ad analizzare tutte le **65536**, ma arriveremo fino alla porta **500**.

Andiamo poi a creare un **“ dizionario ”** con tutte le porte aperte note, inserendo delle **“ tuple ”** con il **nome del servizio** e una **valutazione della sicurezza**.

Avviamo il nostro script, che ci fornisce la lista delle porte aperte con la loro descrizione

\*Nota: Nello screenshot sottostante, sono presenti solo le porte aperte, questo a causa di una leggera modifica dello script, in modo tale da rendere più fruibile e **“ ordinato ”** il report.

```
(kali@kali)-[~/Desktop/Buildingweek]
$ python PortScanner.py

Scannerizzando 192.168.20.20 dalla porta 1 alla porta 500 ...

Porta 21 APERTA - Servizio: FTP - Sicurezza: Non Sicura
Porta 22 APERTA - Servizio: SSH - Sicurezza: Sicura
Porta 23 APERTA - Servizio: Telnet - Sicurezza: Non Sicura
Porta 25 APERTA - Servizio: SMTP - Sicurezza: Rischiosa
Porta 53 APERTA - Servizio: DNS - Sicurezza: Sicura se configurata
Porta 80 APERTA - Servizio: HTTP - Sicurezza: Non Sicura
Porta 111 APERTA - Servizio: RPCBind - Sicurezza: Rischiosa
Porta 139 APERTA - Servizio: NetBios - Sicurezza: Rischiosa
Porta 445 APERTA - Servizio: SMB - Sicurezza: Rischiosa
```

Lista di porte Aperte (nel nostro range)

**21 = FTP, 22 = SSH, 23 = TELNET, 25 = SMTP, 53 = DNS, 80 = HTTP,**

**111 = RPCBIND, 139 = NETBIOS, 445 = SMB**

Considerazioni sulla sicurezza:

- Porta 21 = Il servizio FTP: Trasmette tutti i dati, in chiaro, rendendolo vulnerabile agli attacchi di intercettazione (sniffing), mancanza di crittografia, vulnerabilità agli attacchi Man-in-the-Middle. Si consiglia di utilizzare **FTPS**

- Ports 22 = Il servizio SSH = É un servizio sicuro, grazie alla sua forte crittografia e all'autenticazione tramite chiavi pubbliche.
- Porte 23 = Il servizio Telnet: Trasmette tutti i dati, vengono trasmessi in chiaro, rendendo Telnet vulnerabile agli attacchi di intercettazione(sniffing), mancanza di crittografia, vulnerabilità agli attacchi Man-in-the-Middle. Si consiglia di disabilitare e utilizzare **SSH**
- Porta 25 = Il servizio SMTP: É un servizio che fa configurato, assicurandosi che tutte le comunicazioni tra client e server SMTP siano crittografate utilizzando **TLS** per proteggere i dati da intercettazioni e attacchi man-in-the-middle
- Porta 53 = Il servizio DNS: E un servizio vulnerabile, soprattutto contro attacchi come lo **spoofing**.
- Porta 80 = Il servizio HTTP: É un servizio standard di comunicazione utilizzato per il trasferimento di dati sul web, principalmente pagine web e risorse correlate. Si consiglia di utilizzare il servizio **HTTPS** (Porta **443**), che utilizza la crittografia **TLS**
- Porta 111 = Il servizio RPCBIND: É un servizio che fa configurato, si consiglia di configurare il firewall per impedire l'accesso esterno.
- Porta 139 = Il servizio NetBios: É un servizio che fa configurato, può esporre informazioni dettagliate sulla rete locale, le sessioni NetBIOS non crittografate possono essere intercettate o dirottate dagli attaccanti, compromettendo i dati trasmessi.
- Porta 445 = Il servizio SMB: É un servizio che fa configurato, può essere pericolo soprattutto se esposto su internet o non protetto in LAN. E una porta di ingresso perfetta per un hacker.

2.3 Per la realizzazione dell'esercizio **Bonus**, andremo a creare un programma capace di catturare i socket di rete. Tenteremo poi la connessione con il comando **netcat** da parte della macchina **Web Server (metasploitable)** e verificheremo la cattura eseguita

Procediamo come prima con la creazione del nostro nuovo programma Python, chiamandolo

## “Sniffer.py”

```
1  import socket
2
3  SRV_ADDR = "192.168.10.10" #IP locale della macchina server
4  SRV_PORT = 44444 #porta su cui il server ascolterà
5
6  #creiamo il socket in ascolto
7  s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8  #collegiamo il socket all'IP e alla porta in ascolto
9  s.bind((SRV_ADDR, SRV_PORT))
10 #mettiamo il server in ascolto
11 s.listen(1)
12 print("Server avviato")
13 #accettiamo la connessione con il client che tenterà di connettersi
14 connection, address = s.accept()
15 print('Connessione avviata da:', address)
16
17 #ciclo per ricevere dati finché non riceve il messaggio 'end\n'
18 while True:
19     data = connection.recv(1024)
20     if not data : break
21     message = data.decode('utf-8')
22     print("Messaggio ricevuto:", message.strip())
23     if message.strip() == "end":
24         print("Messaggio di chiusura ricevuto. Fine connessione.")
25         break
26     connection.sendall(b"Messaggio ricevuto \n")
27 connection.close()
```

Importiamo anche qui la libreria “**socket**” (giustamente) per creare un socket in ascolto.

Andiamo a definire l'indirizzo IP e la porta su cui il server sarà in ascolto (44444). L'IP deve essere quello della macchina su cui gira il server (KaliLinux).

```
msfadmin@metasploitable:~$ netcat 192.168.10.10 44444
ciao
Messaggio ricevuto
end
msfadmin@metasploitable:~$
```

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

```
(kali㉿kali)-[~/Desktop/Buildingweek]
$ python Sniffer.py
Server avviato
Connessione avviata da: ('192.168.20.20', 39255)
Messaggio ricevuto: ciao
Messaggio ricevuto: end
Messaggio di chiusura ricevuto. Fine connessione.
```

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 **35294**.

# Preventivo # 1072



**BrrBrr Patapim**

**Odiamo Cavo**

**Magico S.p.a**

*Via Non ci acquisirete  
mai 666*

*P.IVA: IT99999999666*

CLIENTE:

**Theta Network**

*Via Amiamo l'open source 0/1*

*P.IVA: IT00000000000*

Data: 24/04/2025

**Valido fino a: 24/05/2025**

DESCRIZIONE	QUANTITÀ	PREZZO	SUBTOTALE	IVA
Dell optiflex7000sff (PC-Desktop)	120	737	88.440,00	19.456,80 (22%)
TP-LINK TI sf1024 (Switch)	6	49.18	295.1	64.92 (22%)
MIKROTIK Rb2011uias-rm (Router)	1	106,56	106,56	23,44 (22%)
Fortinet Fortigate 60f (Firewall)	1	983,61	983,61	213,69 (22%)
Synology DiskStation ds920 (NAS)	1	942,62	942,62	207,38 (22%)
Suricata/Snort su hardware dedicato/ Software (IDS/IPS)	3	327,87	983,61	216,39 (22%)
120 Punti rete, dorsali, patch panel, canalone, etichettatura (Cablaggio)	120	122,95	14.754,10	3.245,90 (22%)
Mini Server con Linux - Dell PowerEdge t40 (WebServer)	1	409,84	409,84	90,16 (22%)
Servizio di installazione (hardware) e configurazione (Waln, firewall, ids, DMZ) rete generale (testing)	1	4.918,03	4.918,03	1.081,97 (22%)

SUBTOTALE	111.833,47 €
IVA	24.600,65 €
<b>Totale</b>	<b>136.434,12 €</b>

*Banca BrrBrrPatatim*  
*SWIFT/BIC: [SWIFT/BIC]*  
*Numero del conto corrente: IT00X0123411101000000123456*

*In caso di accettazione del preventivo entro i 30 gg disponibili versare un acconto del 100% (più mance) al seguente iban sopra indicato :)*

  
 MUK  
 19  
  
 /  
 SM