Manuale Tecnico – Progetto di Rete CyberFort per Compagnia Theta

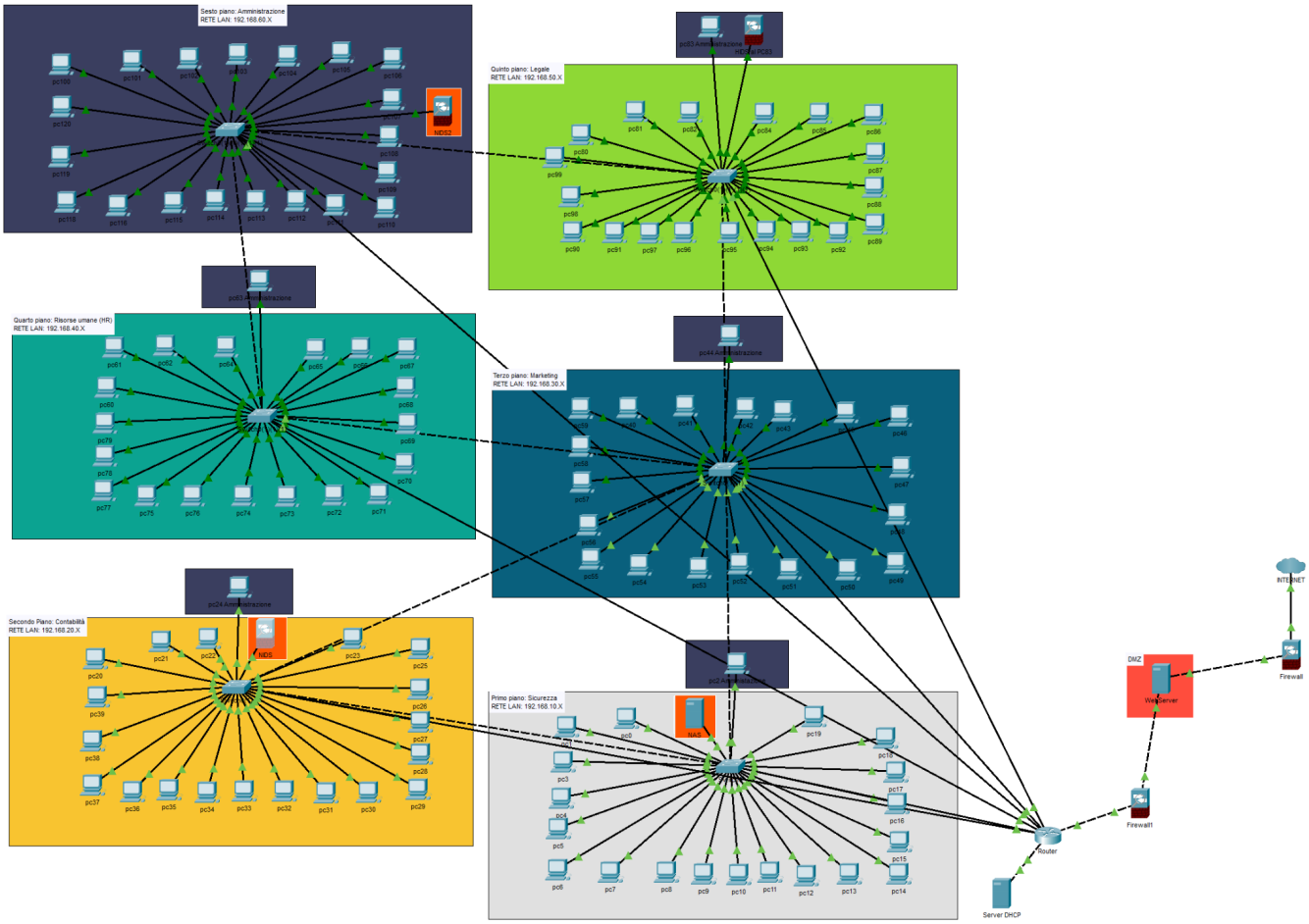
Progetto a cura di Iris Canole, Luca Pani, Daniele Castello, Alessandro Salerno, Rosario Papa e Federico Giannini

Questa parte del progetto è pensata e sviluppata come un manuale di istruzione volta al team tecnico dell'azienda Theta.

Indice

Struttura della rete interna e dispositivi (Cisco)

In risposta alle vostre specifiche, abbiamo realizzato un'infrastruttura di rete strutturata su sei piani, focalizzata sulla sicurezza, sull'efficienza del traffico e sulla gestione semplificata.



Architettura di Rete e Subnetting

L'infrastruttura prevede la disposizione di **20 PC e uno switch gestito per ogni piano**. Il router e il NAS, utilizzato come database centrale, sono collocati al primo piano.

Abbiamo eseguito il **subnetting** per segmentare logicamente la rete, ottenendo i seguenti benefici:

- Riduzione del traffico di broadcast non necessario tra i diversi dipartimenti.
- Aumento della sicurezza, isolando i segmenti di rete.
- Semplificazione della gestione e della risoluzione dei problemi di rete.

Le sottoreti assegnate ai rispettivi piani sono:

Piano	Dipartimento	Indirizzo di rete
1°	Sicurezza	192.168.10.0/24
2°	Contabilità	192.168.20.0/24
3°	Marketing	192.168.30.0/24

4°	Risorse Umane	192.168.40.0/24
5°	Legale	192.168.50.0/24
6°	Amministrazione	192.168.60.0/24

Gli indirizzi IP vengono assegnati dinamicamente ai PC di ciascun piano tramite un Server DHCP dedicato.

Configurazione di Rete a Livello di Piano

Su ogni piano, attraverso lo switch, sono state implementate delle VLAN per:

- Limitare ulteriormente il flusso di dati non necessari all'interno del piano.
- Fornire un ulteriore livello di sicurezza tramite la segmentazione logica del traffico.

Inoltre, grazie all'uso delle VLAN, è stato configurato un PC con appartenenza alla VLAN Amministrazione su ogni piano per facilitare la comunicazione e la condivisione di risorse in tutto lo stabile.

Connettività e Accesso al Database (NAS)

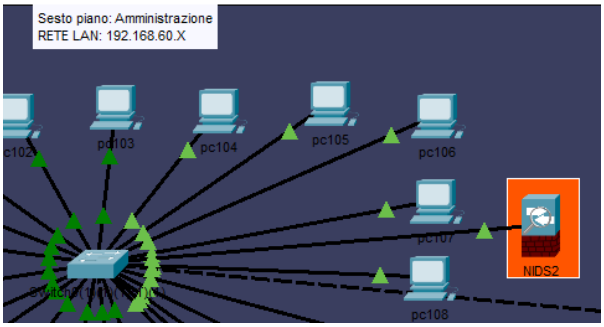
Gli switch di ogni piano sono collegati in modalità *trunk* tra di loro switch per consentire l'estensione della VLAN Amministrazione attraverso tutti i piani, rendendo possibile la comunicazione tra i PC Amministrazione indipendentemente dalla loro ubicazione fisica.
Inoltre, ogni switch è collegato al router centrale.

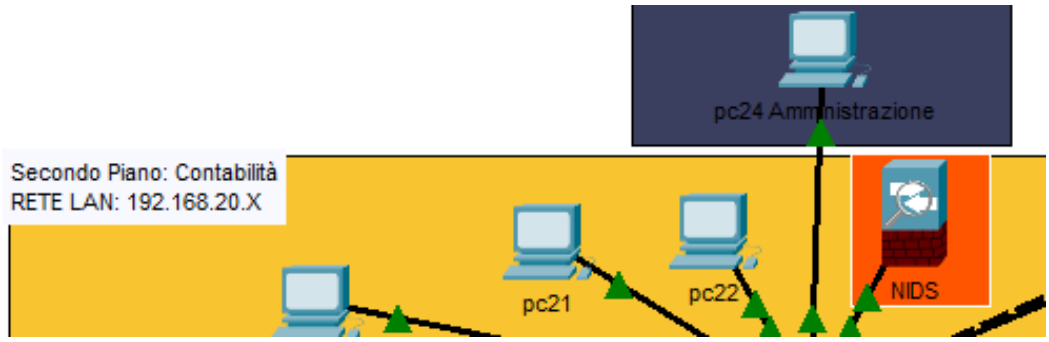
Questa connessione è fondamentale perché il router gestisce l'instradamento del traffico tra le diverse sottoreti, permettendo così a tutti gli host l'accesso a Internet e l'accesso al NAS, che si trova su una sottorete separata.

Sicurezza (Intrusion Detection System e Firewall)

Per garantire un controllo rigoroso del traffico e degli accessi, abbiamo implementato diverse misure di sicurezza:

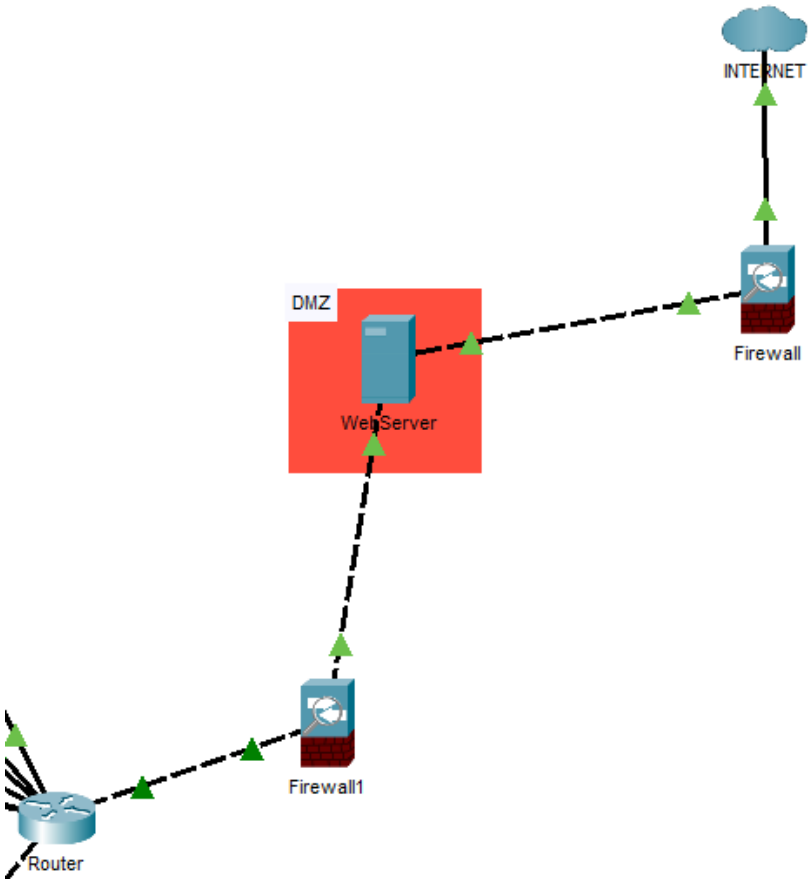
- **Network Intrusion Detection System (NIDS)**: È stato installato un NIDS sui piani Contabilità e Amministrazione. Essendo queste reti particolarmente sensibili, il NIDS monitora il traffico dati in entrata e uscita per rilevare potenziali minacce a livello di segmento.





- **Host Intrusion Detection System (HIDS)**: Un HIDS è stato installato specificamente sul PC dell'amministrazione situato al quinto piano (Legale). Questa scelta è motivata dal passaggio di dati e documenti di elevata importanza su tale host, per il quale si è preferito un controllo "al microscopio" sull'attività del singolo sistema.

- **Firewall**: Sono stati installati due firewall per una difesa in profondità (DMZ - Demilitarized Zone):
  1. Il primo firewall è posizionato tra il router (rete interna) e il web server.
  2. Il secondo firewall è posizionato tra il web server e Internet (rete esterna).



Regole Firewall con PfSense

Nella tabella seguente vengono illustrate le diverse regole per il Firewall, create con PfSense:

VLAN	Source	Destination	Protocolo	Ports	Action
Sicurezza	192.168.10.1	any	any	any	allow
Sicurezza	192.168.10.1	any	TCP/UDP	80,443	allow
Sicurezza	any	192.168.10.1	any	any	deny
Contabilità	192.168.20.1	any	TCP/UDP	53	allow
Contabilità	192.168.20.1	any	TCP	443	allow
Contabilità	192.168.20.1	192.168.60.1	TCP	443	allow
Contabilità	192.168.20.1	192.168.60.1	any	any	deny
Marketing	192.168.30.1	any	TCP/UDP	53,80,443	allow
Marketing	192.168.30.1	192.168.0.0/16	any	any	deny
HR	192.168.40.1	any	TCP/UDP	53,443	allow
HR	192.168.40.1	192.168.60.1	TCP	443	allow
HR	192.168.40.1	192.168.0.0/16	any	any	deny
Legale	192.168.50.1	any	TCP/UDP	53,443	allow
Legale	192.168.50.1	192.168.60.1	TCP	443	allow
Legale	192.168.50.1	192.168.10.1	TCP	443	deny
Legale	192.168.50.1	192.168.0.0/16	any	any	allow
Amministrazione	192.168.60.1	192.168.0.0/16	any	any	allow
Amministrazione	192.168.60.1	any	TCP/UDP	53,80,443	allow

Le regole configurate su PfSense hanno l'obiettivo di garantire **isolamento logico** tra i reparti e, al tempo stesso, consentire solo le comunicazioni necessarie al corretto funzionamento dei servizi aziendali.

Ogni VLAN rappresenta un piano dell'edificio e dispone di policy di sicurezza dedicate, definite in base al livello di sensibilità dei dati trattati.

1. VLAN Sicurezza (`192.168.10.0/24`)

È il livello più protetto dell'infrastruttura: ospita il NAS e i servizi di monitoraggio della rete.

- È consentito l'accesso verso Internet solo sulle porte `80` e `443` (HTTP/HTTPS).
- Sono bloccate tutte le altre connessioni in uscita per evitare scambi non autorizzati.
- È permesso l'accesso interno verso il NAS e verso le altre VLAN per la gestione centralizzata dei backup.

2. VLAN Contabilità (`192.168.20.0/24`)

Tratta dati finanziari e sensibili, pertanto ha un set di regole restrittivo.

- Sono consentite le connessioni DNS (porta `53`) e HTTPS (`443`) necessarie per la contabilità online.
- L'accesso alla VLAN Amministrazione è abilitato per consentire scambi di documenti e dati

contabili.

- Tutto il traffico non specificato viene negato per default (“deny any”).

3. VLAN Marketing (`192.168.30.0/24`)

Dipartimento con maggior traffico multimediale.

- Ha accesso a Internet sulle porte `53`, `80` e `443` per campagne pubblicitarie e social media.
- Non può comunicare direttamente con altre VLAN, se non tramite il router centrale.
- Questa separazione limita l’esposizione a possibili minacce provenienti dall’esterno.

4. VLAN Risorse Umane (`192.168.40.0/24`)

Gestisce dati personali dei dipendenti, quindi necessita di isolamento e cifratura del traffico.

- Accesso limitato ai soli servizi interni e alla VLAN Amministrazione per scambi di dati.
- È consentito il traffico HTTPS (`443`) per applicazioni HR cloud.
- Bloccato il traffico non sicuro e tutto ciò che non passa dal firewall.

5. VLAN Legale (`192.168.50.0/24`)

Contiene documenti riservati e comunicazioni confidenziali.

- Permette solo il traffico TCP su porte `53` e `443`.
- Nessuna connessione diretta verso VLAN di livello inferiore.
- Comunicazioni esterne solo tramite proxy o VPN aziendale.

6. VLAN Amministrazione (`192.168.60.0/24`)

Rappresenta la rete direzionale, con pieno controllo e monitoraggio.

- Può comunicare con tutte le altre VLAN per attività di supervisione e gestione.
- L’accesso a Internet è consentito solo sulle porte `53`, `80` e `443`.
- Tutto il traffico in entrata è filtrato e registrato dal sistema IDS/IPS integrato in PfSense.

Conclusione tecnica

Questa configurazione a **livelli di sicurezza differenziati** consente di:

- Prevenire la propagazione di attacchi tra reparti.
- Garantire la riservatezza dei dati sensibili.
- Aumentare l’efficienza del monitoraggio tramite IDS/IPS e log centralizzati.
- Offrire un’architettura flessibile e facilmente scalabile per future integrazioni.

Scan di porte con Python

Per testare il funzionamento della rete e per poterla controllare al meglio abbiamo scritto in python 3 programmi che fossero in grado di fare scan delle porte, verifica dei metodi `HTTP` disponibili su un determinato path e sniffer dei pacchetti di rete.

Scan di porte

Il programma è eseguibile da terminale e viene subito chiesto l'indirizzo IP dell'host che vogliamo analizzare e un range di porte da definire.

Per quanto riguarda la gestione degli errori nella definizione dell'host, viene segnalato sul terminale se l'indirizzo IP non viene definito correttamente o con numeri impossibili per un IPv4 (ad esempio un numero superiore a `255`).

Abbiamo gestito anche gli errori potenziali nella definizione del range delle porte, usando "`.strip()`" per eliminare spazi superflui, segnalando in caso l'assenza di un trattino tra i due numeri e invertendo l'ordine dei numeri dal più piccolo al più grande nel caso in cui venissero scritto in ordine inverso.

L'output finale sarà quindi una lista delle porte selezionate con la specifica `OPEN` o `CLOSED`.

Di seguito lo script del programma:

```python
import socket

host = input("host/IP: ").strip()
pr = input("inserisci range usando '-' (es. 20-80):
").strip()
if host == "":
    print("inserisci un indirizzo IP corretto")
    raise SystemExit(1)

try:
    low, high = map(int, pr.split("-"))
except Exception:
    print("Range non valido")
    raise SystemExit(1)
if high < low:
    low, high = high, low

try:
    ip = socket.gethostbyname(host)
except socket.gaierror:
    print("Inserito un indirizzo IP errato")
    raise SystemExit(1)
except Exception:
    ip = host

for port in range(low, high + 1):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.settimeout(1.0)
    ok = s.connect_ex((ip, port)) == 0
    print(f"{port}: {'OPEN' if ok else 'CLOSED'}")
    s.close()
```

L'output del programma è il seguente:

```
kali@kali:~/Desktop$ python prova-port.py
host/IP: 192.168.99.50
range (es. 20-80): 20-80

20: CLOSED
21: OPEN
22: OPEN
23: OPEN
24: CLOSED
25: OPEN
26: CLOSED
27: CLOSED
28: CLOSED
29: CLOSED
30: CLOSED
31: CLOSED
32: CLOSED
33: CLOSED
34: CLOSED
35: CLOSED
36: CLOSED
37: CLOSED
38: CLOSED
39: CLOSED
40: CLOSED
41: CLOSED
42: CLOSED
43: CLOSED
44: CLOSED
45: CLOSED
46: CLOSED
47: CLOSED
48: CLOSED
49: CLOSED
50: CLOSED
51: CLOSED
52: CLOSED
53: OPEN
54: CLOSED
55: CLOSED
56: CLOSED
57: CLOSED
58: CLOSED
59: CLOSED
60: CLOSED
61: CLOSED
62: CLOSED
63: CLOSED
64: CLOSED
65: CLOSED
66: CLOSED
67: CLOSED
68: CLOSED
69: CLOSED
70: CLOSED
71: CLOSED
72: CLOSED
73: CLOSED
74: CLOSED
75: CLOSED
76: CLOSED
77: CLOSED
78: CLOSED
79: CLOSED
80: OPEN
```

Verifica dei metodi HTTP

Anche questo programma eseguibile da terminale richiede sin da subito di specificare un indirizzo IP della pagina web, una porta da definire (lasciando di default in caso la porta `80`, dovendola testare sulla metasploitable 2) e il percorso della pagina web da poter scegliere dinamicamente.

A questo punto, avendo inserito in un array i metodi disponibili, quest'ultimi verranno riportati in output come lista, specificando se sono abilitati, non abilitati e tra parentesi anche lo status del server.

Di seguito viene fornito il codice:

```
import http.client

host = input("Host/IP: ").strip()
port = int(input("Porta (default 80): ") or 80)
path = input("Path (es. /): ").strip() or "/"

metodi = ["GET", "POST", "PUT", "DELETE", "HEAD", "OPTIONS"]

for m in metodi:
    try:
        c = http.client.HTTPConnection(host, port, timeout=5)
        c.request(m, path)
        r = c.getresponse()
        ok = r.status not in (405, 501)
        print(f"{m}: {'ABILITATO' if ok else 'NON ABILITATO'} (status
{r.status})")
        c.close(
    except Exception as e:
        print(f"{m}: errore ({e})")
```

Viene fornito anche l'output:

```
$ python lista-verb.py
Host/IP: 192.168.99.50
Porta (default 80): 80
Path (es. /): /phpMyAdmin/themes/original/img/logo_right.png
GET: ABILITATO (status 200)
POST: ABILITATO (status 200)
PUT: NON ABILITATO (status 405)
DELETE: NON ABILITATO (status 405)
HEAD: ABILITATO (status 200)
OPTIONS: ABILITATO (status 200)


$ python lista-verb.py
Host/IP: 192.168.99.50
Porta (default 80): 80
Path (es. /): /phpMyAdmin/favicon.ico
GET: ABILITATO (status 200)
POST: ABILITATO (status 200)
PUT: NON ABILITATO (status 405)
DELETE: NON ABILITATO (status 405)
HEAD: ABILITATO (status 200)
OPTIONS: ABILITATO (status 200)


$ python lista-verb.py
Host/IP: 192.168.99.50
Porta (default 80): 80
Path (es. /): /phpMyAdmin
GET: ABILITATO (status 301)
POST: ABILITATO (status 301)
PUT: ABILITATO (status 301)
DELETE: ABILITATO (status 301
HEAD: ABILITATO (status 301)
OPTIONS: ABILITATO (status 301)
```

Sniffer di rete

Anche questo programma è eseguibile da terminale ma specificando il comando `sudo` per poterlo

eseguire da amministratore, in maniera tale da poter attivare la modalità promiscua e poter intercettare e analizzare tutto il traffico di rete.

Viene richiesto quindi di specificare un indirizzo IP e se voler filtrare una porta nello specifico, altrimenti verrà intercettato tutto il traffico di tutte le porte.
Pure in questo caso per la gestione degli errori abbiamo eliminato spazi superflui con `strip()` e la specifica che il valore della porta debba essere un numero intero.

A questo punto apparirà sullo schermo la scritta `Sniffer avviato... Ctrl+C per terminare`.
e aprendo una pagina web sarà possibile vedere la tipologia di pacchetti, se TCP o UDP, gli indirizzi IPv4 che sono in comunicazione e le porte che stanno utilizzando.

Di seguito viene fornito il codice:

```python
#FAR PARTIRE IL COMANDO CON "SUDO"

import socket

host = input("IP da monitorare (premere invio per tutti): ").strip()
port = input("Porta da filtrare (premere invio per tutte): ").strip()
port = int(port) if port else None

s = socket.socket(socket.AF_PACKET, socket.SOCK_RAW,
socket.ntohs(0x0003))
print("Sniffer avviato... Ctrl+C per terminare.\n")

while True:
    pkt, _ = s.recvfrom(65535)
    if len(pkt) < 34:  # header Ethernet (14) + header IP min (20)
        continue

    src_ip = ".".join(str(b) for b in pkt[26:30])
    dst_ip = ".".join(str(b) for b in pkt[30:34])
    proto = pkt[23]

    if host and host not in (src_ip, dst_ip):
        continue

    if proto == 6 and len(pkt) >= 54:  # 6 = TCP
        sport = int.from_bytes(pkt[34:36], 'big')
        dport = int.from_bytes(pkt[36:38], 'big')
        if port and port not in (sport, dport):
            continue
        print(f"TCP {src_ip}:{sport} → {dst_ip}:{dport}")

    elif proto == 17 and len(pkt) >= 42:  # 17 = UDP
        sport = int.from_bytes(pkt[34:36], 'big')
        dport = int.from_bytes(pkt[36:38], 'big')
        if port and port not in (sport, dport):
            continue
        print(f"UDP {src_ip}:{sport} → {dst_ip}:{dport}")
```

Viene fornito anche l'output del programma:

```
$ sudo python snifferfinale.py
IP da monitorare (premere invio per tutti):
192.168.50.10
Porta da filtrare (premere invio per tutte): 80

Sniffer avviato... Ctrl+C per terminare.

TCP 192.168.50.10:54172 → 192.168.99.50:80
TCP 192.168.50.10:54172 → 192.168.99.50:80
TCP 192.168.99.50:80 → 192.168.50.10:54172
TCP 192.168.99.50:80 → 192.168.50.10:54172
TCP 192.168.50.10:54172 → 192.168.99.50:80
TCP 192.168.50.10:54172 → 192.168.99.50:80
TCP 192.168.99.50:80 → 192.168.50.10:54172
TCP 192.168.99.50:80 → 192.168.50.10:54172
TCP 192.168.50.10:54172 → 192.168.99.50:80
TCP 192.168.50.10:54172 → 192.168.99.50:80
TCP 192.168.99.50:80 → 192.168.50.10:54172
TCP 192.168.99.50:80 → 192.168.50.10:54172
TCP 192.168.50.10:54172 → 192.168.99.50:80
TCP 192.168.50.10:54172 → 192.168.99.50:80
TCP 192.168.99.50:80 → 192.168.50.10:54172
TCP 192.168.99.50:80 → 192.168.50.10:54172
TCP 192.168.50.10:54172 → 192.168.99.50:80
TCP 192.168.50.10:54172 → 192.168.99.50:80
TCP 192.168.99.50:80 → 192.168.50.10:54172
TCP 192.168.99.50:80 → 192.168.50.10:54172
TCP 192.168.50.10:43062 → 185.85.0.29:80
TCP 192.168.50.10:43062 → 185.85.0.29:80
TCP 185.85.0.29:80 → 192.168.50.10:43062
TCP 185.85.0.29:80 → 192.168.50.10:43062
TCP 192.168.50.10:54172 → 192.168.99.50:80
TCP 192.168.50.10:54172 → 192.168.99.50:80
TCP 192.168.99.50:80 → 192.168.50.10:54172
TCP 192.168.99.50:80 → 192.168.50.10:54172
TCP 192.168.50.10:43062 → 185.85.0.29:80
TCP 192.168.50.10:43062 → 185.85.0.29:80
TCP 185.85.0.29:80 → 192.168.50.10:43062
TCP 185.85.0.29:80 → 192.168.50.10:43062
TCP 192.168.50.10:44296 → 185.85.0.29:80
```

Conlcusioni

Il progetto di rete sviluppato da **CyberFort** per la Compagnia **Theta** rappresenta un'infrastruttura solida, scalabile e in linea con i moderni standard di sicurezza informatica.
La progettazione ha seguito un approccio multilivello, dove ogni componente — hardware, software e logico — contribuisce a un ecosistema stabile, controllato e facilmente gestibile.
A livello **architetturale**, la rete è stata segmentata in **sei VLAN** dedicate ai rispettivi reparti aziendali. Questa scelta consente di:

- ridurre il traffico di broadcast,
- isolare i domini di sicurezza,
- migliorare la gestione delle policy di rete.

Il **router centrale** gestisce l'instradamento tra le sottoreti e connette la LAN al firewall perimetrale basato su **pfSense**, garantendo un controllo puntuale dei flussi in ingresso e in uscita.
Le **regole firewall** applicate assicurano che ogni reparto comunichi solo con i servizi strettamente necessari, bloccando il traffico non autorizzato e proteggendo le aree più sensibili (Contabilità, HR, Legale).
L'uso combinato di **IDS e IPS** permette di individuare e mitigare in tempo reale eventuali anomalie o tentativi di intrusione.
Sul fronte **operativo e di testing**, l'integrazione di script in **Python** ha reso possibile:

- la verifica dell'apertura delle porte e della corretta segmentazione (scanner TCP),
- il controllo dei **metodi HTTP** abilitati sui server web interni ed esterni,
- il monitoraggio del traffico di rete in tempo reale tramite **sniffer** personalizzato.

I risultati ottenuti durante i test — come la corretta rilevazione delle porte aperte, la risposta HTTP 200/301 e la cattura del traffico TCP — hanno confermato l'efficacia della configurazione.
Tutti i sistemi rispondono in modo coerente con le policy di sicurezza definite, dimostrando che l'infrastruttura è **funzionale, sicura e pronta per un utilizzo aziendale reale**.
In sintesi, il progetto combina:

- **progettazione logica e fisica avanzata**,
- **configurazione firewall professionale**,
- **strumenti di analisi e controllo personalizzati**,

realizzando un modello di rete che rappresenta un perfetto equilibrio tra **efficienza, protezione e scalabilità**.