

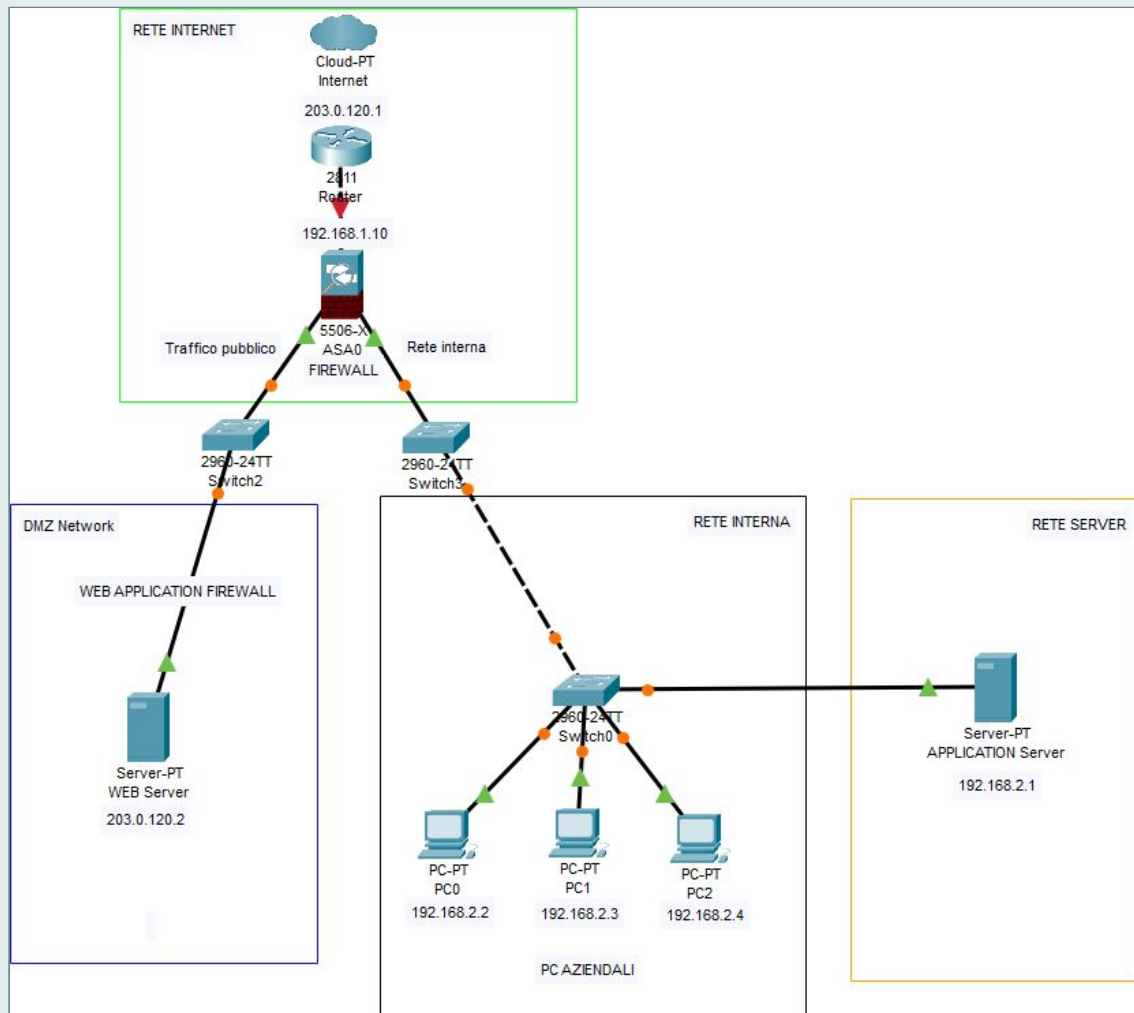


# DESIGN DI RETE

Compagnia Theta



1. **Rafforzare sicurezza Web** (tecniche codifica sicure, protocolli SSL/TSL, scansione vulnerabilità applicazioni Web e test di penetrazione)
2. **Security Policies** (regole accesso ed architettura rete)
3. **Test sicurezza delle applicazioni** (identificare e mitigare difetti di codice)
4. **Gestione delle vulnerabilità** (identificazione, definizione, correzione e segnalazione vulnerabilità)
5. **Network Penetration Testing** (misurare e valutare la sicurezza dell'infrastruttura)
6. **Prevenzione della perdita di dati** (rileva e previene potenziali violazioni dati)
7. **Software antivirus** (previene, scansiona, rileva ed elimina virus)
8. **Misure di sicurezza IDS/IPS** (rilevamento e prevenzione delle intrusioni)
9. **Soluzione SIEM** (rilevamento e gestione incidenti)
10. **Autenticazione a più fattori (MFA)**



# TEST WEB SERVER



# SCANSIONE SERVIZI ATTIVI

kali@kali: ~/Desktop/prove

File Actions Edit View Help

python3 PORT\_scanner.py

Inserisci l'indirizzo IP: 192.168.32.101

Inserisci il range delle porte (0-65535): 0-65535

Scansione host 192.168.32.101 dalla porta 0 alla porta 65535

```
*** 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 37298 - OPEN ***
*** Port 49916 - OPEN ***
*** Port 53587 - OPEN ***
*** Port 59719 - OPEN ***
```

kali@kali: ~/Desktop/prove

File Actions Edit View Help

GNU nano 7.2

PORT\_scanner.py

```
import socket
import ipaddress

while True:
    try:
        #chiede all'utente l'indirizzo IP
        target = input("Inserisci l'indirizzo IP: ")
        #valida indirizzo IP
        ipaddress.ip_address(target)
        break
    except ValueError:
        print("Indirizzo IP non valido.")

while True:
    try:
        #chiede all'utente il range delle porte
        portrange = input("Inserisci il range delle porte (0-65535): ")
        #estrae low e highport
        lowport, highport = map(int, portrange.split("-"))
        #controlla che il range sia valido
        if not (0 <= lowport <= highport <= 65535):
            raise ValueError
        break
    except ValueError:
        print("Range porte non valido.")

#stampa IP e range porte
print('Scansione host', target, 'dalla porta', lowport, 'alla porta', highport)
#itera ogni porta nel range
for port in range(lowport, highport + 1):
    #crea socket TCP
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    #controlla connessione tra IP e port
    status = s.connect_ex((target, port))
    #se lo stato è 0
    if status == 0:
        #stampa porta aperta
        print('*** Port', port, '- OPEN ***')
    else:
        #print('Port', port, '- CLOSED') #porta chiusa
        #chiude il socket
        s.close()
```

Ctrl Help Write Out Where Is Cut Execute  
Exit Read File Replace Paste Justify

# ENUMERAZIONE METODI HTTP



```
kali@kali: ~/Desktop/prove

File Actions Edit View Help

$ python3 HTTP_req.py
Inserisci l'URL di destinazione (senza http://) o digita 'x' per
uscire: 192.168.32.101
Il metodo HTTP 'GET' è consentito
Il metodo HTTP 'POST' è consentito
Il metodo HTTP 'PUT' è consentito
Il metodo HTTP 'DELETE' è consentito
Il metodo HTTP 'OPTIONS' è consentito
Il metodo HTTP 'HEAD' è consentito
Il metodo HTTP 'TRACE' è consentito
Il metodo HTTP 'PATCH' è consentito
Inserisci l'URL di destinazione (senza http://) o digita 'x' per
uscire: x

(kali@kali)~[~/Desktop/prove]
```

```
kali@kali: ~/Desktop/prove

File Actions Edit View Help

GNU nano 7.2 HTTP_req.py
import requests # Libreria contenente le richieste HTTP
#enumera i metodi HTTP, parametro è indirizzo di destinazione
def enumerare_metodi_http(url_destinazione):
    metodi = ['GET', 'POST', 'PUT', 'DELETE', 'OPTIONS', 'HEAD', 'TRACE', 'PATCH'] #metodi HTTP

    for metodo in metodi:
        try:
            #per ogni metodo controlla il codice di stato
            risposta = requests.request(metodo, url_destinazione)
            #Se il codice di stato è 200
            if risposta.status_code == 200: #richiesta e metodo consentito
                print(f"Il metodo HTTP '{metodo}' è consentito")
            #Se il codice di stato è 405
            elif risposta.status_code == 405: #richiesta e metodo non consentito
                print(f"Il metodo HTTP '{metodo}' non è consentito")
            #Se il codice non è 202 e 405
            else: #stampa il metodo ed il suo codice di stato
                print(f"Il metodo HTTP '{metodo}' - Codice di stato: {risposta.status_code}")
            #Se viene colta un'eccezione durante la richiesta
        except requests.exceptions.RequestException as e:
            print(f"Si è verificato un errore durante l'invio della richiesta con il metodo '{metodo}': {str(e)}")

    while True:
        #Chiede all'utente di inserire l'URL
        destinazione = input("Inserisci l'URL di destinazione (senza http://) o digita 'x' per uscire: ")
        #Possiamo digitare x o X per uscire dal programma
        if destinazione.lower() == 'x':
            #esce dal loop
            break
        #chiama la funzione inserendo http://
        url_destinazione = f"http://{destinazione}"
        enumerare_metodi_http(url_destinazione)

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^J Paste      ^_ Justify    ^_ Go To Line  M-E Redo
```

