PROGETTAZIONE RETE SICURA COMPAGNIA THETA

Relazione Tecnica

Il progetto di rete per la compagnia Theta prevede la progettazione di un'infrastruttura IT scalabile e sicura per un edificio di 6 piani, con 120 computer totali (20 per piano), integrando componenti critici come un web server DVWA su Metasploitable, un firewall perimetrale, un NAS e 3 IDS/IPS per la protezione avanzata.

Requisiti e Specifiche

L'edificio richiede una struttura gerarchica: switch dedicati per piano collegano i 20 PC ciascuno, interconnessi a uno switch principale anch'esso collegato a un router centrale per il traffico interno. Il NAS si posiziona allo switch del piano terra per storage condiviso accessibile da tutti i dispositivi, mentre i 3 IDS/IPS monitorano il perimetro interno ed esterno rilevando e prevenendo intrusioni in tempo reale. Il web server DVWA, simulato su Metasploitable, opera come DMZ tra firewall e Internet, con opzione per un secondo firewall per maggiore segmentazione.

Componenti Infrastrutturali

Componente	Quantità	Posizione	Funzione Principale
Switch	6 + 1	Uno per piano + switch principale	Connessione PC locali
Router	1	Centrale	Routing inter-piani
Firewall	1	Perimetrale/DMZ	Blocco minacce esterne
NAS	1	Piano terra	Archiviazione dati
IDS/IPS	3	Perimetro interno ed esterno	Detection/ prevenzione
Web Server DVWA	1	DMZ	Servizi web esposti

Architettura di Rete

La rete interna centralizza il router per unificare gli switch, con il firewall a isolare l'accesso Internet e il NAS ottimizzato per bassa latenza. Esternamente, la DMZ protegge il web server esposto, garantendo comunicazioni sicure senza esposizione diretta della LAN. Questa topologia riduce i punti di failure e migliora la resilienza, con IDS/IPS distribuiti per analisi profonda del traffico.

Test di Verifica

I test usano script Python: uno verifica verbi HTTP (GET, POST, PUT, DELETE) sul web server, documentando risposte su path come phpMyAdmin di Metasploitable (porta 80), richiedendo prima ping da Kali. Un secondo scanner porte accetta IP e range, elencando stati aperti/chiusi senza tool esterni, essenziale per validare esposizioni. Il report finale include risultati dettagliati, raccomandazioni (es. chiusura porte non necessarie) e conferme di connettività.

Di seguito il codice python per la scansione delle porte:

```python
import socket

target = input("IP da scansionare: ")
portrange = input("Range di porte (es. 20-100): ")

lowport = int(portrange.split('-')[0]) # ['20', '100'] -> 20
highport = int(portrange.split('-')[1]) # ['20', '100'] -> 20

print(f"Scannerizzando l'IP {target} da porta {lowport} a porta {highport}")

for port in range(lowport, highport + 1):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    status = s.connect_ex((target, port))
    if status == 0:
        print(f"Porta {port} : aperta")
    else:
        print(f"Porta {port} : chiusa")
    s.close()
```

Di seguito il codice python per catturare il socket di rete:

```python
import socket
import sys #Permette la comunicazione tra programma e sistema operativo

# --- CONFIGURAZIONE SERVER ---
# '0.0.0.0' ascolta su tutte le interfacce disponibili (necessario se ti connetti da un altro PC)
# Se vuoi ascoltare solo su un IP specifico, usa ad esempio: SRV_ADDR = "192.168.1.34"
SRV_ADDR = input("Inserisci l'IP a cui collegarti: ")
SRV_PORT = int(input("Inserisci la porta: "))
BUFFER_SIZE = 1024 # Dimensione standard per la ricezione dei dati (BIT)

# Creazione del socket (TCP/IPv4)
try:
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print("Socket creato con successo.")
except socket.error as err:
    print(f"Errore nella creazione del socket: {err}")
    sys.exit()

# 1. Assegnazione di IP e porta (Bind)
try:
    s.bind((SRV_ADDR, SRV_PORT))
    print(f"Socket legato a {SRV_ADDR}:{SRV_PORT}")
except socket.error as err:
    print(f"Bind fallito. Errore: {err}")
    sys.exit()

# 2. Metti il socket in ascolto (Listen)
# 5 Ã¨ il numero di connessioni client che possono essere in coda.
s.listen(1)
print("Socket in ascolto...")

# --- LOOP PRINCIPALE (Accetta Connessioni Multiple) ---
# Questo loop permette al server di accettare nuovi client dopo che i precedenti si sono disconnessi.
while True:
    connection = None #default
    try:
        # 3. Accettazione della connessione (Accept)
        # Il programma si blocca qui in attesa di un client
        connection, address = s.accept()
        print(f"\n[!!!] Connessione accettata da: {address[0]}:{address[1]}")
```

```
41
42          # --- GESTIONE DEL CLIENT SINGOLO (LOOP INTERNO) ---
43          while True:
44              # 4. Ricezione dei dati
45              data = connection.recv(BUFFER_SIZE)
46
47              # Controllo fondamentale: se recv() restituisce un oggetto vuoto (b''),
48              # significa che il client ha chiuso la connessione.
49              if not data:
50                  print(f"[---] Client {address[0]} disconnesso.")
51                  break # Esci dal loop interno
52
53              # Decodifica e stampa del messaggio
54              try:
55                  message = data.decode('utf-8').strip()
56                  print(f"   << Ricevuto: {message}")
57              except UnicodeDecodeError:
58                  print("   << Dati ricevuti non decodificabili in UTF-8.")
59
60              # 5. Risposta al client
61              # Invia una conferma di ricezione.
62              response = b"SERVER: Messaggio ricevuto.\n"
63              connection.sendall(response)
64              print(f"   >> Inviato: {response.decode().strip()}")
65
66      except KeyboardInterrupt:
67          # Gestisce la chiusura con Ctrl+C
68          print("\n\n[-] Server terminato da utente (Ctrl+C).")
69          break
70
71      except Exception as e:
72          print(f"\n[!!!] Errore critico durante la comunicazione: {e}")
73
74      finally:
75          # 6. Chiusura della connessione (Cruciale)
76          if connection:
77              connection.close()
78
79  # Chiusura finale del socket del server
80  s.close()
81  print("Socket principale chiuso. Programma terminato.")
```

Di seguito il codice python per la verifica dei verbi HTTP:

```
1   import http.client
2
3   def test_metodi(host, port, path):
4       methods = ["OPTIONS", "GET", "POST", "PUT", "DELETE", "HEAD", "PATCH"]
5       results = {}
6       conn = None
7
8       try:
9           conn = http.client.HTTPConnection(host, port, timeout=5)
10
11          for method in methods:
12              conn.request(method, path)
13              response = conn.getresponse()
14              response.read()  # svuota il body
15              results[method] = (response.status, response.reason)
16
17      except ConnectionRefusedError:
18          print("Connessione rifiutata dal server")
19
20      except Exception as e:
21          print("Errore:", e)
22
23      finally:
24          if conn:
25              conn.close()
26
27      return results
28
29
30
31  host = input("Inserire host/IP del sistema target: ")
32  port = input("Inserire la porta del sistema target (default 80): ")
33  path = input("inserisci il root: ")
34
35  port = 80 if port == "" else int(port)
36  path = "/" if path == "" else str(path)
37
38  results = test_metodi(host, port, path)
39
40  print("\nRisultati verifica verbi HTTP:")
41  for method, (status, reason) in results.items():
42      print(f"{method}: {status} {reason}")
```

Aspetti Avanzati e Bonus

Per il subnetting bonus, si calcola una rete /24 (es. 192.168.1.0/24) per 120 host, dividendola in subnet per piani (es. /27 per 30 host/piano) ottimizzando IP e broadcast. Un packet sniffer Python cattura socket di rete per analisi forense aggiuntiva. Il preventivo di spesa (non dettagliato nel brief) copre hardware enterprise-level, priorizzando ROI (return on investment) su sicurezza e scalabilità per Theta.

CATALOGO PREZZI – INFRASTRUTTURA IT AZIENDALE

Scenario: Azienda su 6 piani, 120 PC (20 per piano). Il catalogo include Offerta A (Intermedia) e Offerta B (Avanzata).

OFFERTA A – INTERMEDIA (PC i5 / 16GB / SSD 512GB)

Voce	Prezzo Unitario (€)	Totale (€)
PC Intermedio x120 (i5, 16GB RAM, 512gb)	700	84.000
Sistema Operativo	120	14.400
Totale Offerta A		**98.400**

OFFERTA B – AVANZATA (PC i7 / 32GB / SSD 1TB)

Voce	Prezzo Unitario (€)	Totale (€)
PC Avanzato x120 (i7, 32GB RAM, 1TB)	1.100	132.000
Sistema Operativo	120	14.400
Totale Offerta B		**146.400**

NETWORKING

Voce	Quantità	Totale (€)
Router aziendale	1	800
Switch Layer 3 (core)	1	1.500
Switch Layer 2 (6 piani)	6	3.600
Cablaggio strutturato	-	4.000
Totale Networking		**9.900**

SICUREZZA

Voce	Quantità	Totale (€)
Firewall perimetrale NGFW	1	3.000
IDS/IPS	3	6.000
Licenze sicurezza (1 anno)	-	2.000
Totale Sicurezza		**11.000**

SERVER & STORAGE

Voce	Quantità	Totale (€)
Web Server aziendale	1	2.500
NAS RAID 24TB	1	3.000
Software Backup	-	1.000
Totale Server & Storage		**6.500**

INSTALLAZIONE E CONFIGURAZIONE

Servizio	Totale (€)
Installazione hardware	3.000
Configurazione rete VLAN	2.000
Configurazione firewall e IDS/IPS	2.500
Configurazione server e NAS	2.000
Totale Servizi	**9.500**

RIEPILOGO FINALI

Offerta	Totale (€)
Offerta A – Intermedia	135.300
Offerta B – Avanzata	183.300

19/12/25 - Team Samurai

(Vincenzo Di Salvo, Vincenzo Zarola, Mirko Imbrogno, Victor Rosati, Gerald Bejte, Daniele Di Martino)