



CYBER GGEZ

BUILD WEEK 1

RETE PER LA
COMPAGNIA TETHA

SPECIFICHE

Siamo stati ingaggiati dalla compagnia Theta per sviluppare un preventivo di spesa e un progetto di rete per la loro infrastruttura IT

REQUISITI

Struttura dell'edificio: 6 piani

Dispositivi previsti: 20 computer per piano

Componenti aggiuntivi: 1 Web server, 1 Firewall perimetrale, 1 NAS, 3 IDS/IPS

WORKFLOW

01

- Assegnazione de ruoli
- Divisione del lavoro
- Definizione delle modalità

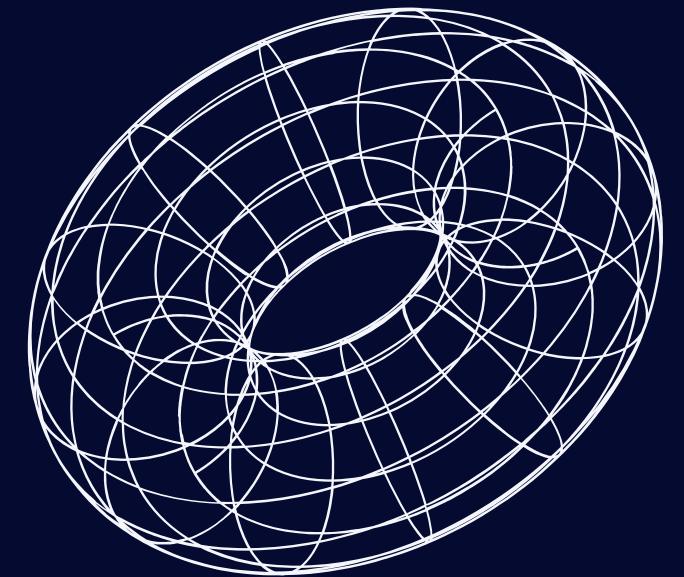
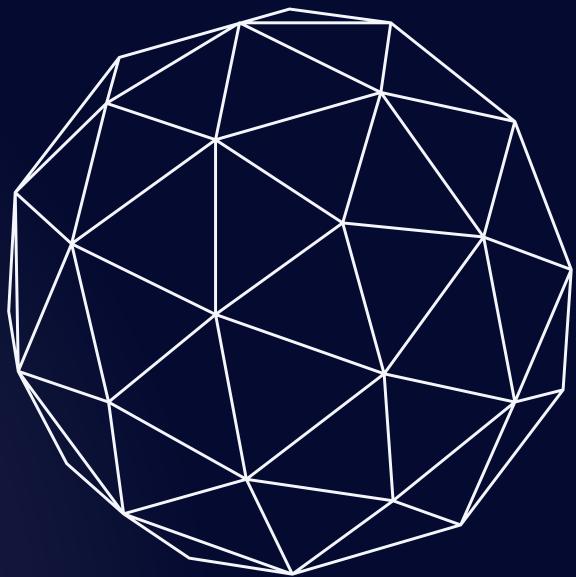
02

- Sviluppo delle attività
- Confronto e feedback
- Risoluzione problemi
- Test e verifiche

03

- Produzione di report
- Organizzazione dei dati
- Impaginazione finale

CONTENUTO



Progettazione della rete

- Switch e VLAN
- Configurazione routing
- Firewall
- Servizi server
- Sistema di monitoraggio (ids/ips)

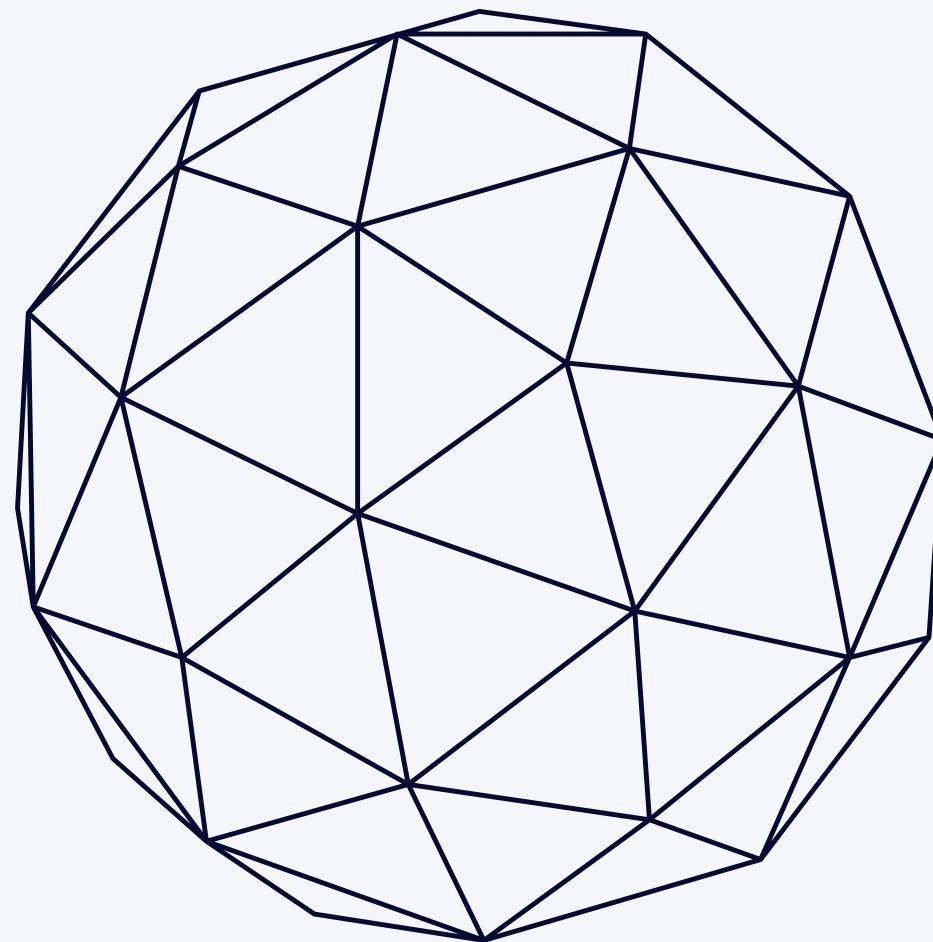
Programmi Python

- Verifica dei verbi HTTP
- Scansione delle porte
- Socket di rete

Considerazioni finali

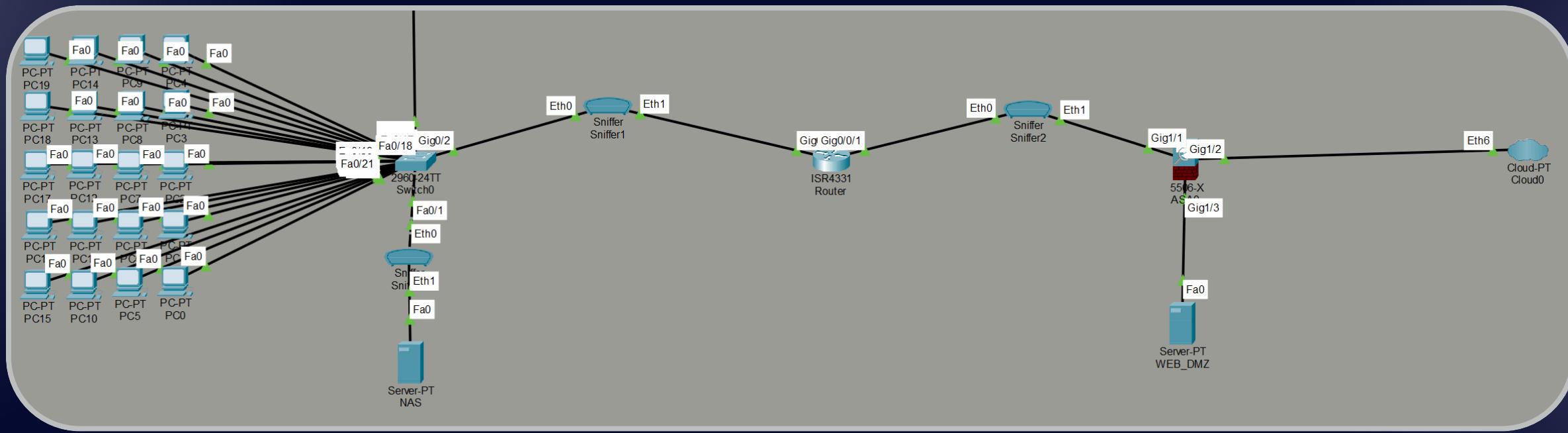
- Risultati dei test
- Raccomandazioni

PROGETTAZIONE DELLA
RETE

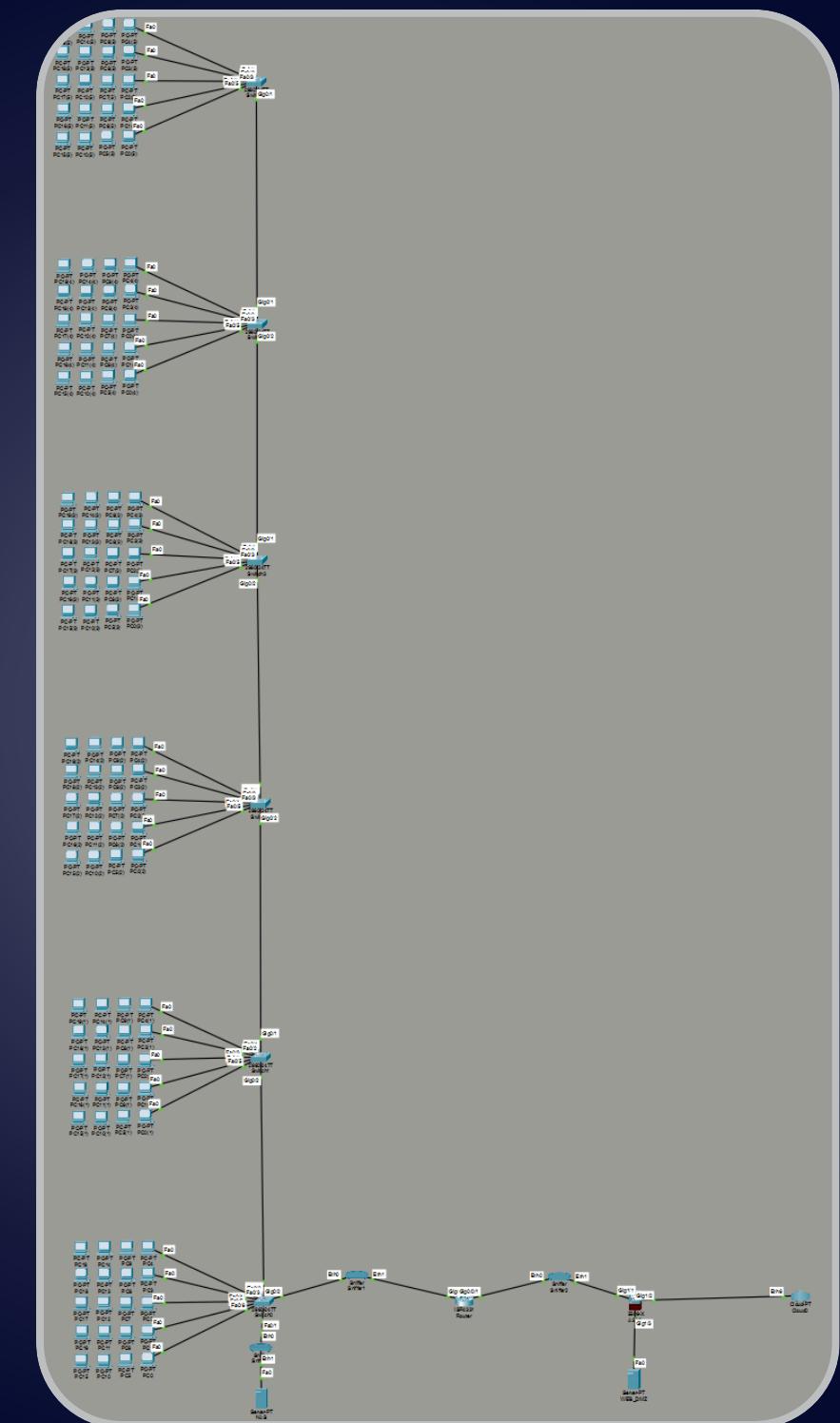


STRUTTURA

Piano terra



Rete completa



SWITCH & VLAN

Creazione VLAN

Abbiamo utilizzato 6 switch (uno per piano) e implementato le VLAN:
VLAN 10 = Piano terra
VLAN 20 = Primo piano
VLAN 30 = Secondo piano ...
Abbiamo usato le VLAN per poter dividere ogni piano e far sì che un problema al piano terra non potesse bloccare altri piani.

Trunk

Sono state configurate le porte gigabit tra i vari Switch e tra Switch e Router in mod TRUNK. Questo perché una porta normale (access) può portare una sola VLAN, mentre il TRUNK usa il protocollo 802.1Q per taggare i pacchetti, permettendo a un solo cavo fisico di trasportare il traffico di tutte le VLAN contemporaneamente

Access

Le porte FastEthernet collegate ai PC sono state settate su ACCESS sulla VLAN specifica del proprio piano

VLAN No	
1	default
10	PIANO_0
20	PIANO_1
30	PIANO_2
40	PIANO_3
50	PIANO_4
60	PIANO_5
1002	fddi-default
1003	token-ring-default
1004	fddinet-default
1005	trnet-default

ROUTING

Configurazione

È stato usato un Router firewall posizionato al piano terra e collegato allo switch, per far parlare i vari piani con il NAS al piano terra, in questo caso il Router fa da ponte, ed è stato utilizzato anche per assegnare i 120 IP differenti per i vari PC. Configurazione eseguita: Abbiamo utilizzato un Router , configurandolo in modalità “Router-on-a-stick” e come server DHCP

```
Router>enable
Router#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface GigabitEthernet0/0/0.10
Router(config-subif)# encapsulation dot1Q 10
Router(config-subif)# ip address 192.168.10.1 255.255.255.0
Router(config-subif)# exit
Router(config)#interface GigabitEthernet0/0/0.20
Router(config-subif)# encapsulation dot1Q 20
Router(config-subif)# ip address 192.168.20.1 255.255.255.0
Router(config-subif)# exit
Router(config)#interface GigabitEthernet0/0/0.30
Router(config-subif)# encapsulation dot1Q 30
Router(config-subif)# ip address 192.168.30.1 255.255.255.0
Router(config-subif)# exit
Router(config)#interface GigabitEthernet0/0/0.40
Router(config-subif)# encapsulation dot1Q 40
Router(config-subif)# ip address 192.168.40.1 255.255.255.0
Router(config-subif)# exit
Router(config)#interface GigabitEthernet0/0/0.50
Router(config-subif)# encapsulation dot1Q 50
Router(config-subif)# ip address 192.168.50.1 255.255.255.0
Router(config-subif)# exit
Router(config)#interface GigabitEthernet0/0/0.60
Router(config-subif)# encapsulation dot1Q 60
Router(config-subif)# ip address 192.168.60.1 255.255.255.0
Router(config-subif)# exit
Router(config)#
Router(config)#

```

```
Router(config)*
Router(config)#ip dhcp pool P0
Router(dhcp-config)# network 192.168.10.0 255.255.255.0
Router(dhcp-config)# default-router 192.168.10.1
Router(dhcp-config)# dns-server 8.8.8.8
Router(dhcp-config)#ip dhcp pool P1
Router(dhcp-config)# network 192.168.20.0 255.255.255.0
Router(dhcp-config)# default-router 192.168.20.1
Router(dhcp-config)# dns-server 8.8.8.8
Router(dhcp-config)#ip dhcp pool P2
Router(dhcp-config)# network 192.168.30.0 255.255.255.0
Router(dhcp-config)# default-router 192.168.30.1
Router(dhcp-config)# dns-server 8.8.8.8
Router(dhcp-config)#ip dhcp pool P3
Router(dhcp-config)# network 192.168.40.0 255.255.255.0
Router(dhcp-config)# default-router 192.168.40.1
Router(dhcp-config)# dns-server 8.8.8.8
Router(dhcp-config)#ip dhcp pool P4
Router(dhcp-config)# network 192.168.50.0 255.255.255.0
Router(dhcp-config)# default-router 192.168.50.1
Router(dhcp-config)# dns-server 8.8.8.8
Router(dhcp-config)#ip dhcp pool P5
Router(dhcp-config)# network 192.168.60.0 255.255.255.0
Router(dhcp-config)# default-router 192.168.60.1
Router(dhcp-config)# dns-server 8.8.8.8
Router(dhcp-config)#exit
Router(config)#interface GigabitEthernet0/0/1
Router(config-if)# ip address 10.0.0.1 255.255.255.252
Router(config-if)# no shutdown
Router(config)#

```

```
Router(config-if)# exit
%LINK-3-UPDOWN: Interface GigabitEthernet0/0/1, changed state to down
Router(config)#
Router(config)#
Router(config)#interface GigabitEthernet0/0/1
Router(config-if)#exit
Router(config-if)#
Router(config-if)#
%LINK-3-UPDOWN: Interface GigabitEthernet0/0/0, changed state to down
%LINK-3-UPDOWN: Interface GigabitEthernet0/0/0.10, changed state to down
%LINK-3-UPDOWN: Interface GigabitEthernet0/0/0.20, changed state to down
%LINK-3-UPDOWN: Interface GigabitEthernet0/0/0.30, changed state to down
%LINK-3-UPDOWN: Interface GigabitEthernet0/0/0.40, changed state to down
%LINK-3-UPDOWN: Interface GigabitEthernet0/0/0.50, changed state to down
%LINK-3-UPDOWN: Interface GigabitEthernet0/0/0.60, changed state to down
Router(config-if)#ip route 0.0.0.0 0.0.0.0 10.0.0.2
Router(config)#
Router(config)#
Router(config)#
Router#
$SYS-5-CONFIG_I: Configured from console by console

```

ROUTING

On a stick

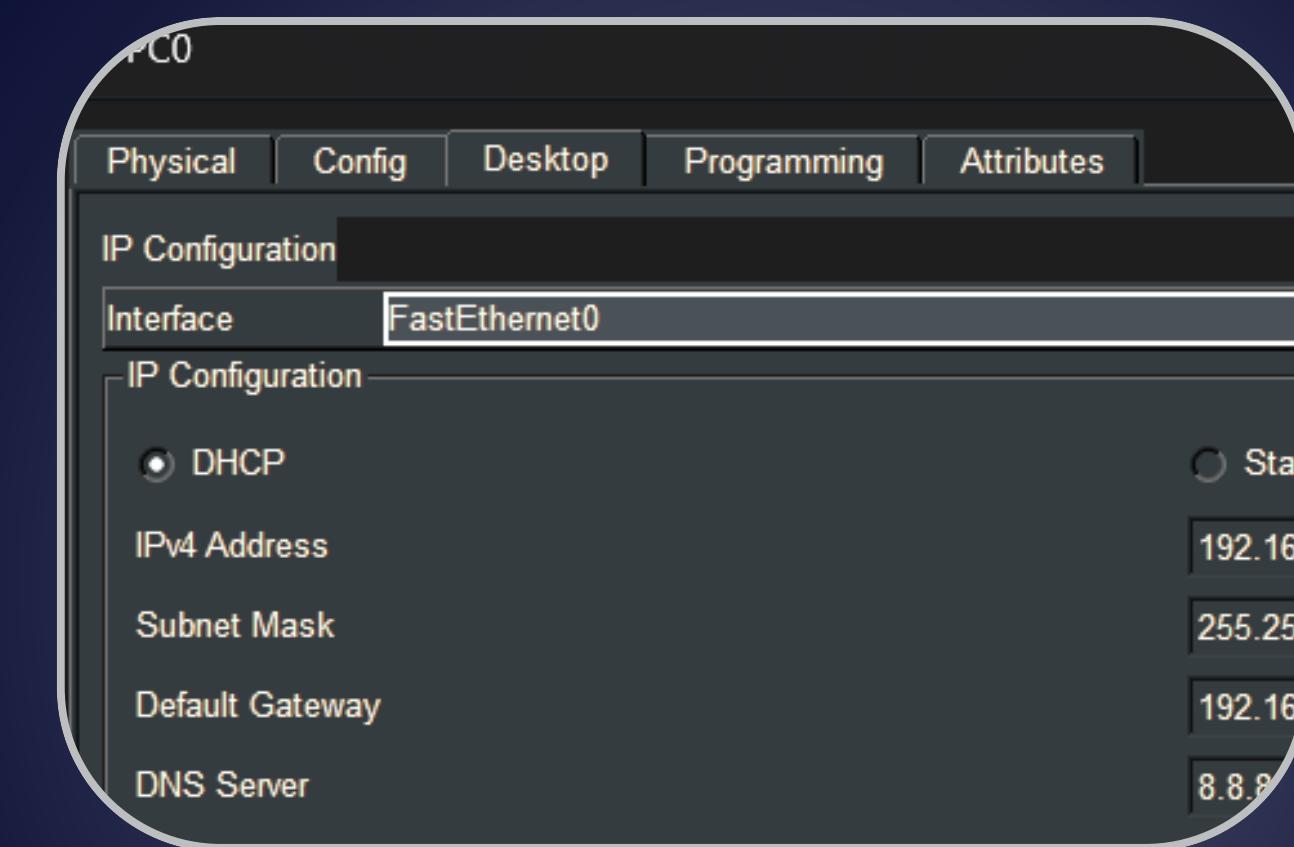
Invece di usare 6 cavi fisici per collegare ogni switch al router, abbiamo acceso una singola porta fisica collegando solo lo Switch al piano terra con il Router e creato 6 interfacce virtuali una per ogni VLAN. Comando Chiave: "encapsulation dot1Q (10,20,30....)" Questo comando dice al router: "Tutto il traffico taggato con ID (10,20,30....), deve essere gestito da questa interfaccia virtuale, che ha l'IP 192.168.(10,20,30...).1". Questo funge da Default Gateway per i PC.

DHCP

Abbiamo configurato 6 Pool DHCP sul Router. Il DHCP automatizza l'assegnazione degli IP, questo vuol dire che quando un PC si accende, chiede un IP in broadcast, il Router lo intercetta e gli assegna un indirizzo valido per lo specifico piano del PC.

Indirizzi esclusi

Abbiamo escluso i primi 10 indirizzi dal pool (da .1 a .10) Questo è stato fatto per evitare conflitti IP. L'indirizzo IP .1 è del router e il .5 è del NAS. Se il DHCP li assegnasse per sbaglio ad un PC la rete crollerebbe (PC conflict).



ROUTING

Configurazione parte firewall

Sono state inserite delle regole per far sì che il Piano 4 (marketing) e il Piano 5 (segreteria) avessero restrizioni nei propri PC. Piano 4 (marketing) = Alla VLAN di questo piano è stato negato l'accesso al server NAS Piano 5 (segreteria) = Alla VLAN di questo piano è stato negato l'accesso illimitato ad internet, dando l'opzione di poter visitare solo il sito Dilitrust (sito che verrà utilizzato dalle segreteria)

Rules (Drag to Change Order)										
States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
✓ 1/594 KiB	*	*	*	KALI Address	80	*	*		Anti-Lockout Rule	
✗ 0/0 B	IPv4 *	Rete_Piano_4	*	NAS	*	*	none		Blocco Piano 4 al NAS	
✗ ✓ 0/0 B	IPv4 TCP/UDP	Rete_Piano_5	*	8.8.8.8	53 (DNS)	*	none		Pass DNS	
✗ ✓ 0/0 B	IPv4 TCP	Rete_Piano_5	*	Siti_Lavoro	443 (HTTPS)	*	none		Pass segreterie Dilitrust	
✗ ✗ 0/0 B	IPv4 *	Rete_Piano_5	*	*	*	*	none		Blocco totale piano 5	
✗ ✓ 3/687 KiB	IPv4 *	KALI subnets	*	*	*	*	none		Default allow LAN to any rule	
✗ ✓ 0/0 B	IPv6 *	KALI subnets	*	*	*	*	none		Default allow LAN IPv6 to any rule	

Add Add Delete Toggle Copy Save Separate

FIREFWALL

Configurazione

Abbiamo utilizzato un firewall per proteggere la rete interna da internet ed esporre un Web Server pubblico in una zona isolata,in questo modo se il server viene hackerato, la rete interna resta salva.

Security levels

NSIDE (VLAN interna) : Security Level 100 (massima fiducia)

OUTSIDE (Internet): Security Level 0 (Nessuna fiducia)

DMZ (Server Web): Security Level 50 (Fiducia media)

Di default, Questo Firewall permette traffico da livello Alto a Basso (Inside -> Outside), ma blocca tutto da Basso ad Alto. Questo impedisce a Internet di entrare nella rete aziendale.

Routing Statico

Abbiamo aggiunto la rotta:

route inside 192.168.0.0 255.255.0.0 10.0.0.1

Il Firewall non conosce la nostra rete interna (VLAN 10-60). Senza questa rotta, quando il Web Server rispondeva a un ping, il Firewall non sapeva dove mandare la risposta e la buttava via. Ora sa che deve passarla al Router (10.0.0.1)

FIREWALL

Access Control Lists (ACL) e Policy Map

Abbiamo creato una regola (ACL) per permettere il traffico ICMP (Ping) di ritorno. Anche se il livello di sicurezza lo permette, l'ASA blocca i ping per default perché considera l'ICMP un protocollo "non stateful". Abbiamo dovuto forzare l'ispezione ICMP per permettere la diagnostica.

Regole per rete interna = Il firewall è stato configurato per permettere ai pc di navigare (HTTP/HTTPS), risolvere nomi (DNS) e fare ping. non possono fare altro (es. torrent o connessioni strane verso l'esterno).

Regole per la DMZ = il web server deve poter rispondere a internet, ma non deve mai poter iniziare una connessione verso la rete interna. se un hacker prende il server, non può muoversi lateralmente, quindi il firewall è stato settato per fare anche questo.

Regole per l'outside = Il firewall è stato settato anche per far sì che da internet nessuno entri tranne chi vuole vedere il sito web

The screenshot shows the ASA configuration interface under the Firewall / NAT / Port Forward tab. A yellow banner at the top states: "The NAT configuration has been changed. The changes must be applied for them to take effect." Below this, there are tabs for Port Forward, 1:1, Outbound, and NPT. The Port Forward tab is selected. It displays a table of rules:

Interface	Protocol	Source Address	Source Ports	Dest. Address	Dest. Ports	NAT IP	NAT Ports	Description	Actions
WAN	TCP	*	*	WAN address	80 (HTTP)	172.16.1.10	80 (HTTP)	ACCESSO WEB SERVER DMZ	

At the bottom are buttons for Add, Delete, Toggle, Save, and Separator.

The screenshot shows the ASA configuration interface under the Firewall / Rules / METASPOITABLE tab. A yellow banner at the top states: "The 'admin' account password is set to the default value. Change the password in the User Manager." Below this, there are tabs for Floating, WAN, KALI, and METASPOITABLE. The METASPOITABLE tab is selected. It displays a table of rules:

States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
✓	0/0 B	IPv4 *	*	*	*	*	*	none		
✗	0/0 B	IPv4 *	METASPOITABLE subnets	*	KALI subnets	*	*	none	BLOCCO METASPOITABLE VERSO KALI	
✓	0/0 B	IPv4 *	METASPOITABLE subnets	*	*	*	*	none	ACCESSO INTERNET	

At the bottom are buttons for Add, Delete, Toggle, Copy, Save, and Separator.

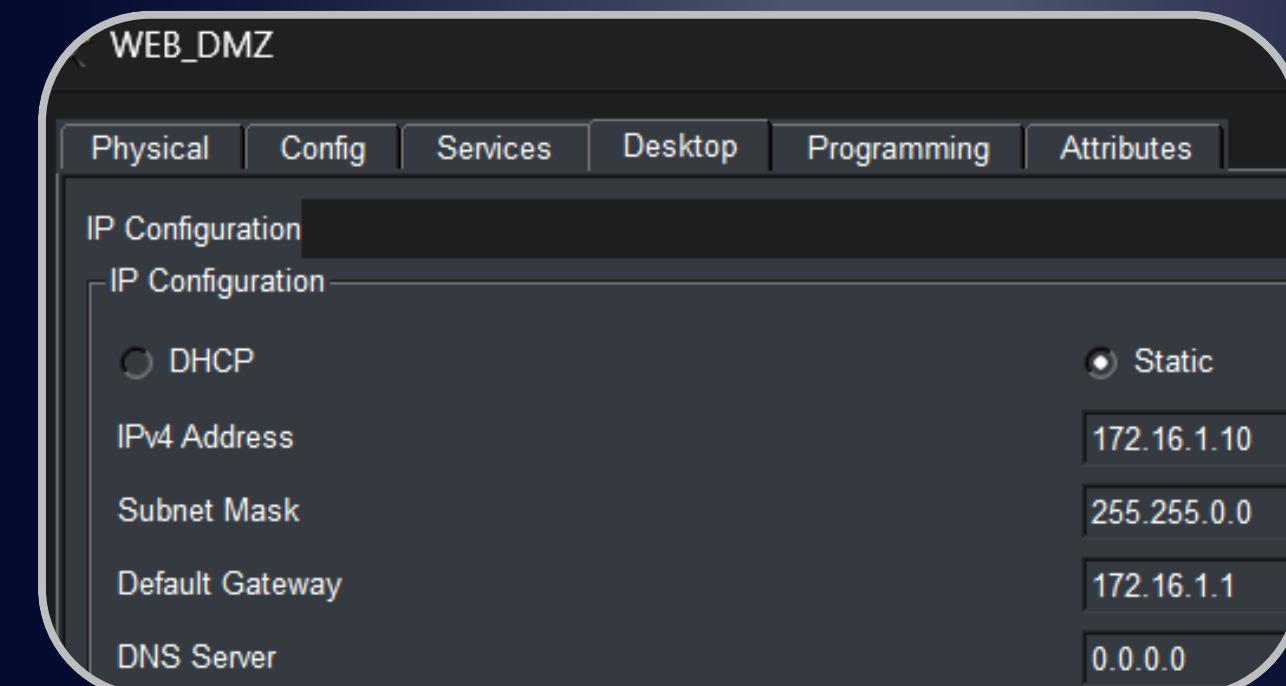
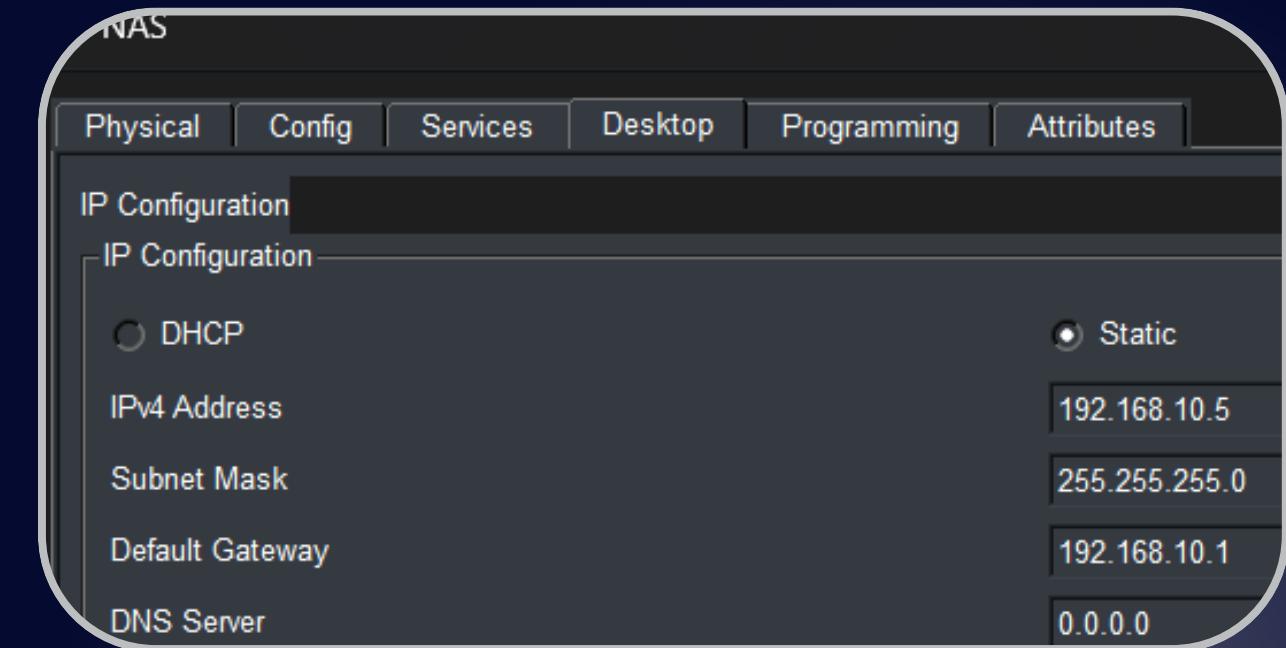
SERVER

NAS

Posizione : Piano terra (Vlan 10), collegato allo switch Piano_Terra.
Configurazione : IP statico 192.168.10.5, Gateway 192.168.10.1
Funzionamento : Grazie al settaggio Routing con le VLAN, tutti i PC di qualsiasi piano possono salvare file sul NAS attraverso il Gateway (dispositivo hardware o software che collega due reti diverse, traducendo i protocolli per permettere la comunicazione)

SERVER WEB

Posizione: Zona DMZ (Demilitarized Zone) del Firewall. Dettagli di Configurazione : Indirizzo IP: Statico 172.16.1.10 Gateway : 172.16.1.1 Funzionalità e Sicurezza : È isolato. Se un hacker compromette questo server, il Firewall (Security Level 50) gli impedisce di saltare nella rete interna (Security Level 100).



IPS / IDS

Sistema di monitoraggio

Sono stati posizionati 3 Sniffer in punti strategici con l'obiettivo di avere visibilità sul traffico che passa nella rete

1 - CORE

Posizione : sul collegamento Trunk tra Router e Switch Piano_Terra

Utilità e Funzionamento : Questa sonda è fondamentale per rilevare le minacce interne. Se un PC del piano 5, infettato da un malware, cercasse di attaccare un pc su un altro piano, il Firewall perimetrale non se ne accorgere

2 - EDGE

Posizione : sul collegamento tra Router e Firewall

Utilità e Funzionamento : Agisce come una seconda linea di difesa dietro al firewall. Rileva attacchi che sono riusciti a bypassare le regole statiche del firewall. Rileva tentativi di Command & Control ovvero PC interni che cercano di comunicare con server hacker esterni.

IPS / IDS

3 – ASSET

Posizione : sul collegamento diretto tra Switch Piano_Terra e Server NAS

Utilità e Funzionamento : Data loss prevention (DLP) e integrità dei file.

Sonda più critica. Poiché il NAS contiene i dati aziendali sensibili, questa sonda applica filtri molto stringenti. Analizza specificamente i protocolli di trasferimento file per rivelare:

- Tentativi di accesso non autorizzato
- Esfiltrazione massiva di dati
- Ransomware (malware che limita l'accesso del dispositivo che infetta chiedendo un riscatto) che tenta di cifrare i file condivisi.

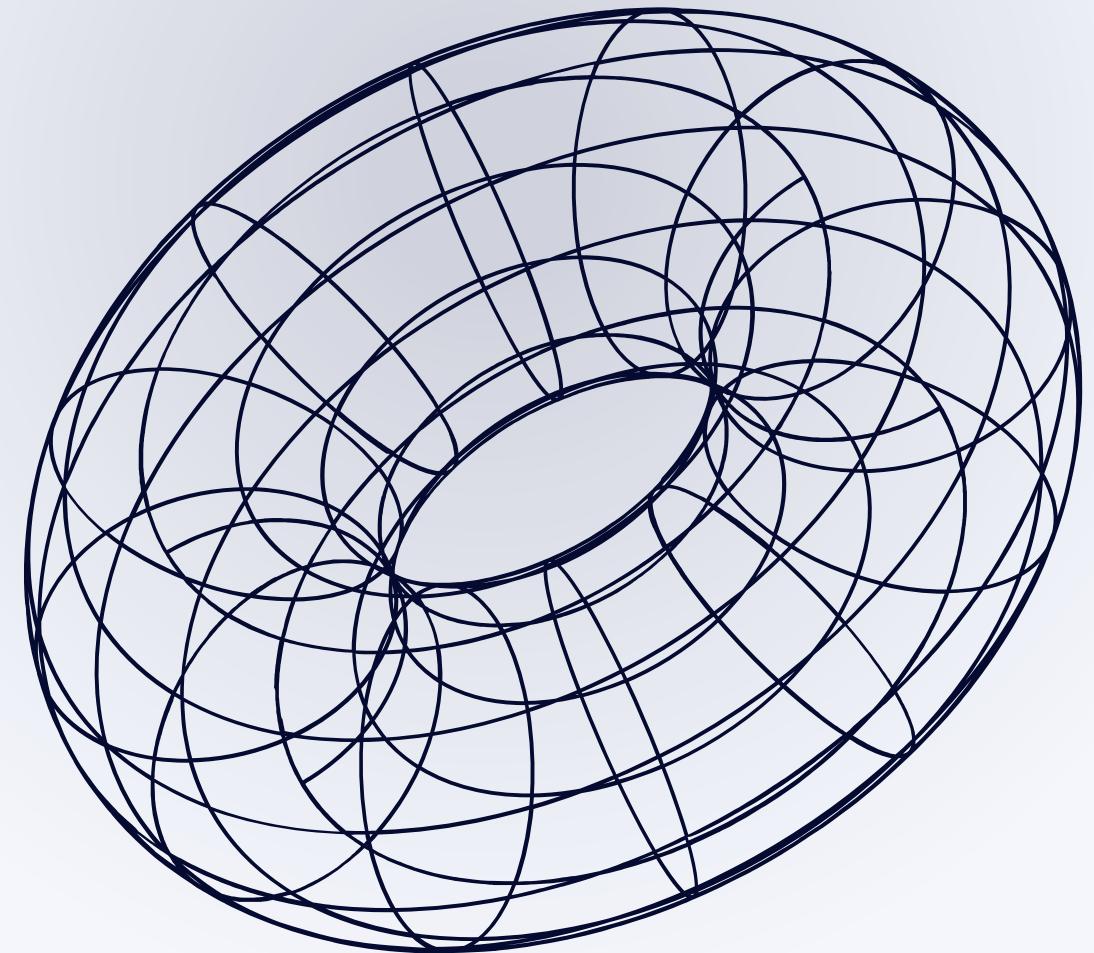
Tutte e tre le sonde sono configurate per l'ispezione profonda dei pacchetti (Deep Packet Inspection) sui protocolli ICMP, TCP, UDP e HTTP.

L'architettura a tre livelli garantisce che nessun pacchetto possa raggiungere i dati sensibili senza essere stato scansionato almeno due volte (una volta al perimetro o nel core, e una volta a ridosso del server).



PROGRAMMI

PYTHON



VERBI HTTP

Obiettivo

Analizzare i metodi HTTP abilitati su un servizio web per valutare la superficie di attacco ed individuare configurazioni non necessarie o potenzialmente rischiose.

Funzionamento

L'utente inserisce host, porta e path; lo script testa diversi metodi HTTP (OPTIONS, GET, POST, PUT, DELETE), analizzando header, status code e parte della risposta.

Output

Genera un file di log strutturato con tempi di scansione ed errori, utile allo scopo di comprendere il protocollo HTTP e le fasi iniziali di analisi di un servizio web.

```
❶ 11_http.py > ...
 1 import http.client
 2 import socket
 3 from datetime import datetime
 4
 5 # Richiesta dati all'utente
 6
 7 host = input("Inserire IP o hostname: ")
 8 port = input("Inserire la porta (default 80): ")
 9
10 # Se l'utente non inserisce nulla, viene usata la porta HTTP standard
11 if port == "":
12     | port = 80
13 else:
14     | port = int(port)
15
16 # Path del sito da testare
17 path = input("Inserire il path (default /): ")
18
19 # Se non viene inserito nulla, viene usata la root
20 if path == "":
21     | path = "/"
22
23 # Se il path non inizia con / lo aggiungiamo automaticamente
24 if not path.startswith("/"):
25     | path = "/" + path
26
27 # Impostazione file di log
28
29 file_name = f"httpscan_{host}_{port}_{path.replace('/', '_')}.txt"
30
31 # Funzione di logging
32
33 def log(*msg, blank_before=False, blank_after=False):
34     with open(file_name, "a", encoding="utf-8") as f:
35         if blank_before:
36             | f.write("\n")
37
38         message = " ".join(map(str, msg))
39         f.write(f"{message}\n")
40
41         if blank_after:
42             | f.write("\n")
43
44 # Avvio scansione
45
46 start_time = datetime.now()
47
48 with open(file_name, "a", encoding="utf-8") as f:
49     f.write(
50         f"Scansione avviata il {start_time.strftime('%Y-%m-%d')} "
51         f"alle ore {start_time.strftime('%H:%M:%S')}\n"
52     )
53     f.write("=====\n")
54
55 # Metodi HTTP da testare
56
57 methods = ["OPTIONS", "GET", "POST", "PUT", "DELETE"]
58
59 # Ciclo principale di scansione
60
61 for method in methods:
62     print("=====")
63     print(f"Metodo HTTP: {method}")
64     print("=====")
65     print(f"Path: {path}")
66
67     log("=====", blank_before=True)
68     log(f"Metodo HTTP: {method}")
69     log("=====")
70     log(f"Path: {path}")
71
72 try:
73     # Apertura della connessione HTTP verso il target
74     conn = http.client.HTTPConnection(host, port, timeout=5)
75
76     headers = {}
77     body = None
78
79     # POST e PUT richiedono un body di test
80     if method == "POST" or method == "PUT":
81         body = "test-body"
82         headers = {
83             "Content-Type": "text/plain",
84             "Content-Length": str(len(body))
85         }
86
87     # Invio della richiesta HTTP
88     conn.request(method, path, body=body, headers=headers)
89
90     # Ricezione della risposta dal server
91     response = conn.getresponse()
92
93     print("Status:", response.status)
94     print("Reason:", response.reason)
95     log("Status:", response.status)
96     log("Reason:", response.reason)
97
98     # Gestione specifica del metodo OPTIONS
99     if method == "OPTIONS":
100         allow_header = response.getheader("Allow")
101
102         print(
103             f"Metodi consentiti: {allow_header} if {allow_header} else [header Allow non presente]"
104         )
105         log(
106             f"Metodi consentiti: {allow_header} if {allow_header} else [header Allow non presente], blank_after=True"
107         )
108
109     else:
110         # Lettura parziale del body per evitare output troppo lunghi
111         response_body = response.read(200).decode(errors="ignore")
112
113         print("Body:")
114         print(response_body if response_body else "[vuoto]")
115         log("Body:")
116         log(response_body if response_body else "[vuoto]", blank_after=True)
117
118     # Chiusura della connessione
119     conn.close()
120
121     # Timeout di rete
122     except socket.timeout:
123         print("Errore: timeout della connessione")
124         log("Errore: timeout della connessione", blank_after=True)
125
126     # Server raggiungibile ma connessione rifiutata
127     except ConnectionRefusedError:
128         print("Errore: connessione rifiutata")
129         log("Errore: connessione rifiutata", blank_after=True)
130
131     # Qualsiasi altro errore non previsto
132     except Exception as e:
133         print("Errore generico:", e)
134         log("Errore generico:", e, blank_after=True)
135
136     # Fine scansione
137
138     end_time = datetime.now()
139     duration = end_time - start_time
140
141     total_seconds = int(duration.total_seconds())
142     hours = total_seconds // 3600
143     minutes = (total_seconds % 3600) // 60
144     seconds = total_seconds % 60
145
146     with open(file_name, "a", encoding="utf-8") as f:
147         f.write("=====\n")
148         f.write(f"Scansione completata il {end_time.strftime('%Y-%m-%d')}\n")
149         f.write(f"alle ore {end_time.strftime('%H:%M:%S')}\n")
150         f.write("=====\n")
151         f.write(f"Durata totale della scansione: "
152             f"\n{hours:02d}:{minutes:02d}:{seconds:02d}\n"
153         )
154
155     f.write(
156         f"Durata totale della scansione: "
157         f"\n{hours:02d}:{minutes:02d}:{seconds:02d}\n"
158     )
159
160
161     print("\n=====")
162     print("Fine Programma")
163     print("=====")
164
```

```
(kali㉿kali)-[~/Desktop/python]
$ /usr/bin/python /home/kali/Desktop/python/11_http.py
Inserire IP o hostname: 192.168.20.10
Inserire la porta (default 80):
Inserire il path (default /): /twiki/
=====
Metodo HTTP: OPTIONS
=====
Path: /twiki/
Status: 200
Reason: OK
Metodi consentiti: GET,HEAD,POST,OPTIONS,TRACE
=====
Metodo HTTP: GET
=====
Path: /twiki/
Status: 200
Reason: OK
Body:
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<
=====
Metodo HTTP: POST
=====
```

SCAN PORTE

Obiettivo

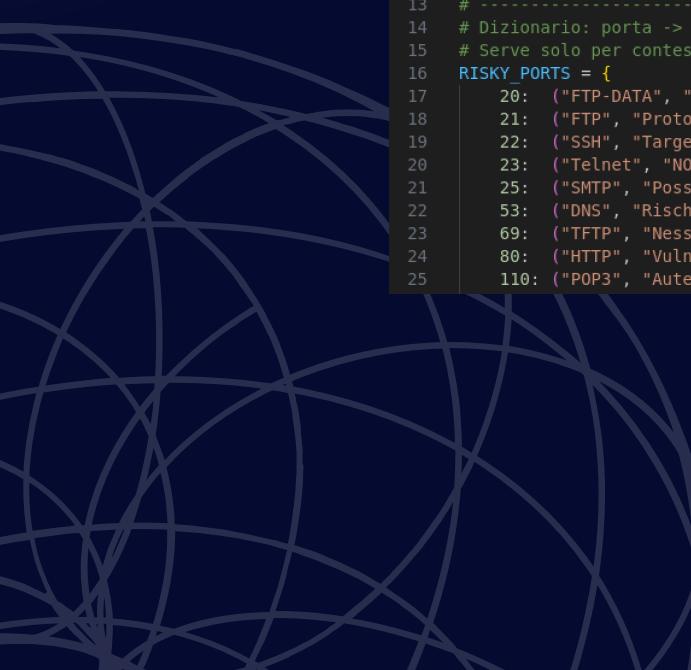
Scansione delle porte TCP per identificare i servizi di rete esposti e analizzare la superficie di attacco di un host.

Funzionamento

L'utente inserisce IP e range di porte; il programma tenta connessioni TCP con timeout breve per determinare se le porte sono aperte o chiuse e individuare servizi sensibili.

Output

Risultati mostrati a console e salvati su file di log con dettagli della scansione, utili per analisi interne, verifica delle configurazioni e attività preliminari di sicurezza.



```
❶ 13_port.py > ...
 1 import socket
 2 from datetime import datetime
 3
 4 # -----
 5 # COLORI ANSI (OUTPUT CONSOLE)
 6 #
 7 # Usati per evidenziare in rosso le porte considerate rischiose
 8 RED = "\033[91m"
 9 RESET = "\033[0m"
10
11 # -----
12 # DATABASE PORTE RISCHIOSE
13 #
14 # Dizionario: porta -> (servizio, descrizione rischio)
15 # Serve solo per contestualizzare le porte aperte
16 RISKY_PORTS = {
17     20: ("FTP-DATA", "Trasferimento FTP in chiaro"),
18     21: ("FTP", "Protocollo in chiaro, brute force"),
19     22: ("SSH", "Target comune per brute force"),
20     23: ("Telnet", "NON cifrato, molto insicuro"),
21     25: ("SMTP", "Possibile abuso relay"),
22     53: ("DNS", "Rischio di misconfigurazione"),
23     69: ("FTP", "Nessuna autenticazione"),
24     80: ("HTTP", "Vulnerabilità applicative"),
25     110: ("POP3", "Autenticazione debole"),
 51   # FILE DI LOG
 52   # -----
 53   # Nome del file dove verranno salvati i risultati
 54 filename = "port_scan.txt"
 55
 56   # -----
 57   # INPUT UTENTE
 58   #
 59   # IP o hostname da scansionare
 60 target = input("IP da scansionare: ")
 61
 62   # Range di porte in formato es. 20-100
 63 portrange = input("Range delle porte (es 20-100): ")
 64
 65   # Estrazione porta iniziale e finale dal range
 66 lowport = int(portrange.split("-")[0])
 67 highport = int(portrange.split("-")[1])
 68
 69 print(f"Scansionando l'IP {target} da porta {lowport} a porta {highport}")
 70
 71   # Apertura file in append per non sovrascrivere scansioni precedenti
 72 with open(filename, "a", encoding="utf-8") as file:
 73
 74   # Intestazione della scansione
 75   file.write("Scansione porte\n")
 76   file.write(f"Target: {target}\n")
 77   file.write(f"Porte: {lowport}-{highport}\n")
 78   file.write(f"Ora inizio: {datetime.now()}\n")
 79   file.write("-" * 40 + "\n")
 80
 81   # SCANSIONE PORTE
 82   # -----
 83   for port in range(lowport, highport + 1):
 84
 85       # Creazione socket TCP
 86       s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
 87
 88       # Timeout per evitare blocchi su porte filtrate
 89       s.settimeout(1)
 90
 91       # Tentativo di connessione alla porta
 92       status = s.connect_ex((target, port))
 93
 94       # Se connect_ex restituisce 0, la porta è aperta
 95       if status == 0:
 96
 97           # Se la porta è nel database delle porte rischiose
 98           if port in RISKY_PORTS:
 99               service, risk = RISKY_PORTS[port]
100
101               # Output evidenziato in rosso
102               print(f"\033[91m{RED}Porta {port}: APERTA -> {service} | Rischio: {risk}{RESET}\033[0m")
103
104               # Log dettagliato su file
105               file.write(f"Porta {port}: APERTA -> {service} | Rischio: {risk}\n")
106
107           else:
108               # Porta aperta ma non classificata come rischiosa
109               print(f"Porta {port}: APERTA")
110               file.write(f"Porta {port}: APERTA\n")
111
112       else:
113           # Porta chiusa o filtrata
114           print(f"Porta {port}: CHIUSA")
115           file.write(f"Porta {port}: CHIUSA\n")
116
117   # Chiusura del socket
118   s.close()
119
120   # Footer della scansione
121   file.write("-" * 40 + "\n")
122   file.write(f"Fine scansione: {datetime.now()}\n")
123   file.write("-" * 40 + "\n")
124
125   print(f"\nRisultati salvati in: {filename}")
126
```

```
(kali㉿kali)-[~/Desktop/python]
$ /usr/bin/python /home/kali/Desktop/python/13_port.py
IP da scansionare: 192.168.20.10
Range delle porte (es 20-100): 20-100
Scansionando l'IP 192.168.20.10 da porta 20 a porta 100
Porta 20: CHIUSA
Porta 21: APERTA -> FTP | Rischio: Protocollo in chiaro, brute force
Porta 22: APERTA -> SSH | Rischio: Target comune per brute force
Porta 23: APERTA -> Telnet | Rischio: NON cifrato, molto insicuro
Porta 24: CHIUSA
Porta 25: APERTA -> SMTP | Rischio: Possibile abuso relay
Porta 26: CHIUSA
Porta 27: CHIUSA
Porta 28: CHIUSA
Porta 29: CHIUSA
Porta 30: CHIUSA
Porta 31: CHIUSA
Porta 32: CHIUSA
```

SOCKET

Obiettivo

Monitorare il traffico di rete a livello Ethernet intercettando i frame tramite raw socket, prima dell'elaborazione da parte dello stack di rete.

Funzionamento

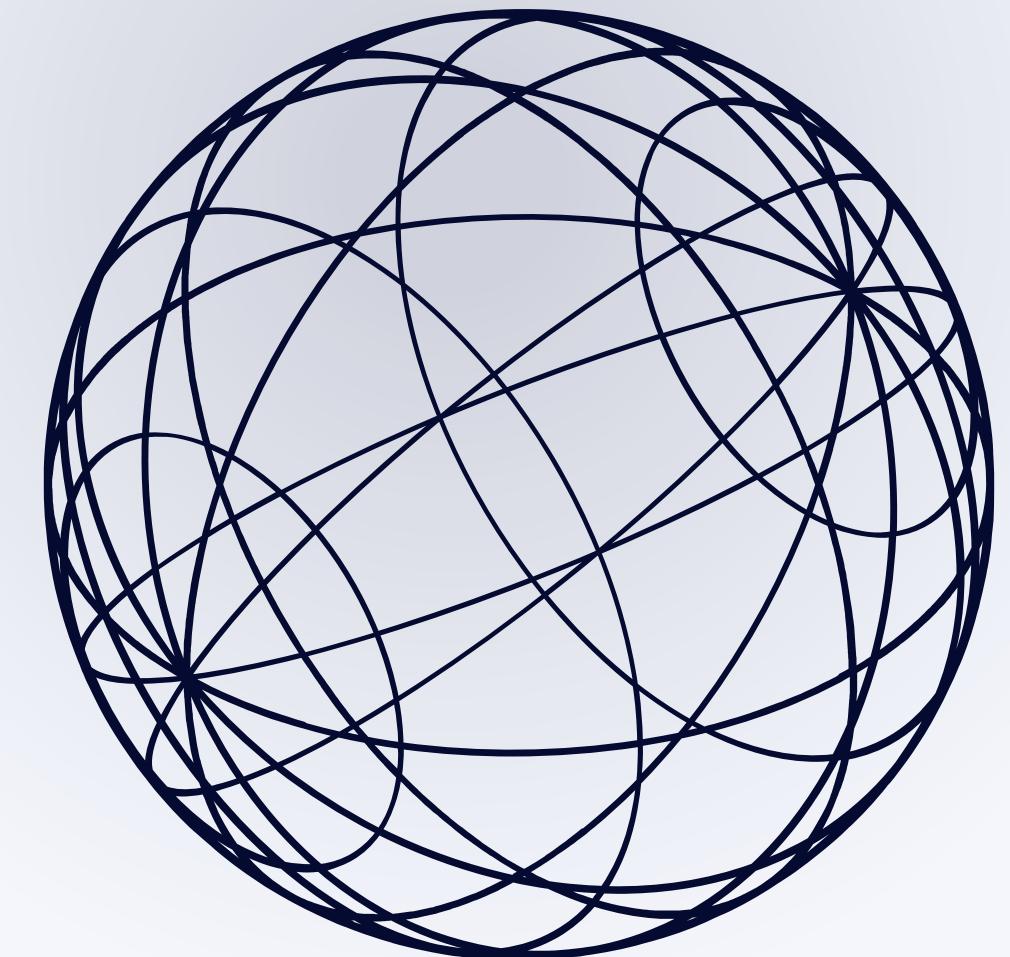
programma cattura i pacchetti in tempo reale, analizza l'header Ethernet ed identifica il protocollo trasportato (IPv4, ARP, IPv6), registrando anche traffico non riconosciuto.

Output

Genera un file di log ordinato con timestamp e numerazione dei frame, utile per comprendere il traffico di rete e supportare attività di monitoraggio, troubleshooting e sicurezza.

CONSIDERAZIONI

FINALI



TEST

CONSIDERAZIONI

Questi programmi permettono di osservare in modo semplice e diretto come una rete e i suoi servizi sono esposti e comunicano tra loro. Forniscono una visione generale del comportamento della rete, aiutando a individuare possibili punti di attenzione e a migliorare la consapevolezza sullo stato complessivo dell'infrastruttura.

RACCOMANDAZIONI

È fondamentale, in ambito di sicurezza informatica, garantire che un web server risponda solo alle richieste strettamente necessarie, al fine di ridurre la superficie di attacco. Lo stesso principio si applica alla gestione delle porte aperte, con particolare attenzione a quelle associate a vulnerabilità note.

In questo contesto, i programmi sviluppati hanno supportato la verifica e il consolidamento della configurazione della rete, contribuendo a garantire un livello di sicurezza complessivamente più elevato.

GRAZIE

CYBER GG EZ