



Progetto di Rete **Compagnia Theta**

da Spegni&Riaccendi S.p.A.

Chi siamo?

Team Leader:

Amin El Kassimi

Team di Configurazione Rete:

Leonardo Takeshi Chiaverini,
Josh Van Edward Abanico

Team di Sviluppo & Testing:

Bartolomeo Tarantino,
Nicolò Calì

Team di Ricerca & Preventivo:

Sergio Falcone



Table of contents

01

Configurazione Rete

02

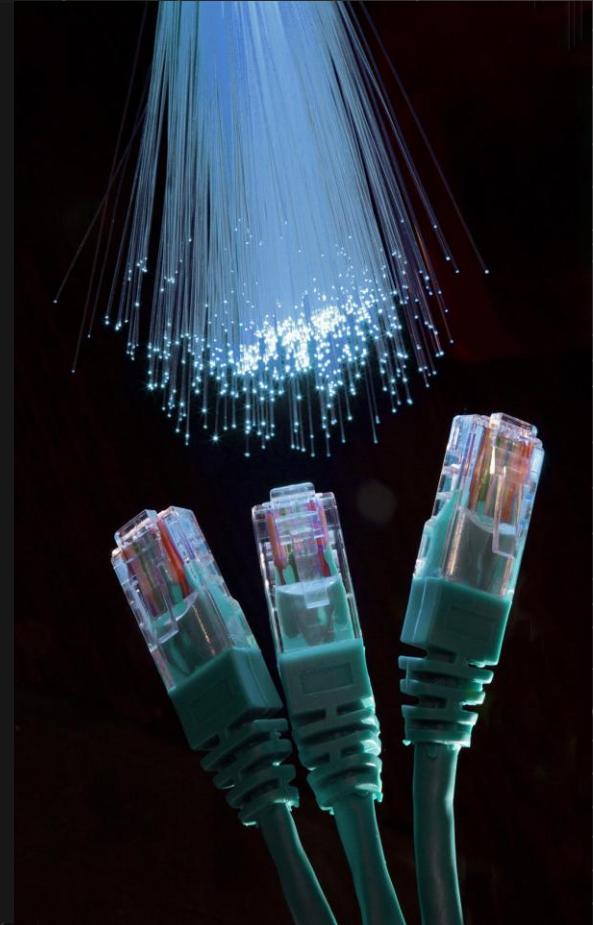
Certificazione Rete

03

Ricerca e Preventivo

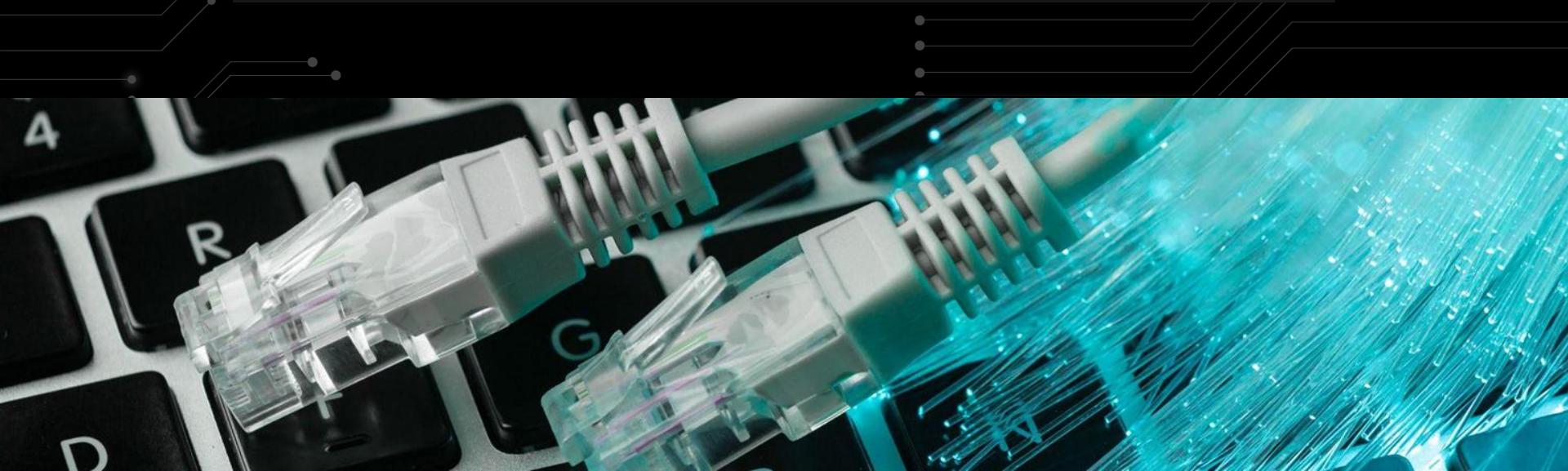
04

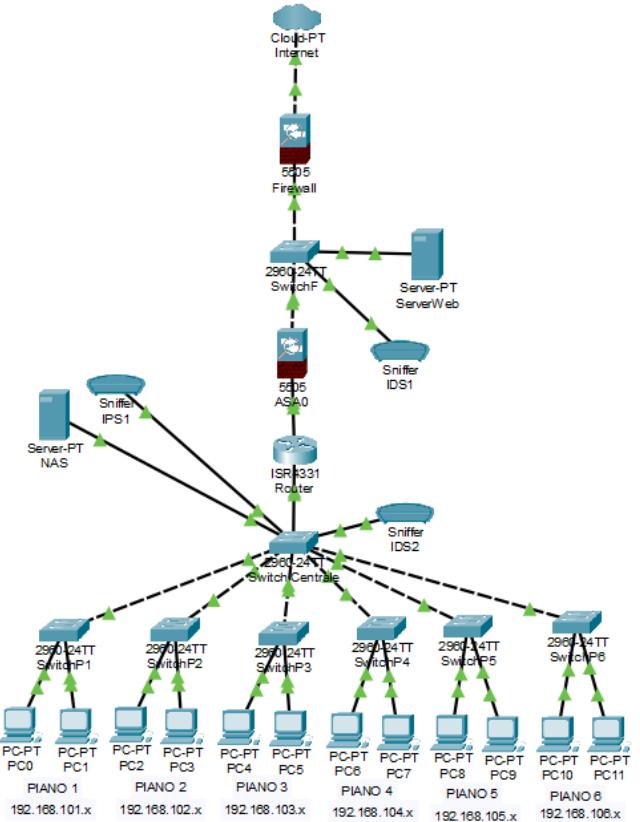
Conclusione



01

Configurazione Rete





TOPOLOGIA DELLA RETE

L'infrastruttura di rete è progettata secondo un'architettura gerarchica e segmentata, basata sulla separazione tra **LAN**, **DMZ** e **WAN**.

Il **firewall perimetrale** protegge l'accesso verso Internet, la DMZ ospita i servizi esposti, mentre la LAN è dedicata alla rete interna degli utenti.

Questa struttura consente una gestione ordinata del traffico e un miglior **controllo** della sicurezza.

I sistemi **IDS** e **IPS** sono integrati nell'infrastruttura e posizionati in punti strategici della rete consentendo il monitoraggio del traffico tra rete interna, DMZ e perimetro.

Il sistema **NAS** è collocato in una VLAN dedicata ai servizi di storage, separata sia dalle VLAN utente sia dalla VLAN di management.

SEGMENTAZIONE DELLA RETE

La rete interna è segmentata logicamente tramite **VLAN**, una per ciascun piano dell'edificio.

La segmentazione permette di **isolare** i domini di broadcast e **limitare** la propagazione di eventuali problemi o **attacchi** interni.

È inoltre prevista una VLAN di management, separata dalle VLAN utente, dedicata al monitoraggio e al controllo dell'infrastruttura di rete.

Nome VLAN	ID VLAN
Management	50
Piano 1	101
Piano 2	102
Piano 3	103
Piano 4	104
Piano 5	105
Piano 6	106
NAS	90

SUBNETTING IP

Per ogni VLAN è previsto un indirizzamento **IP dedicato**.

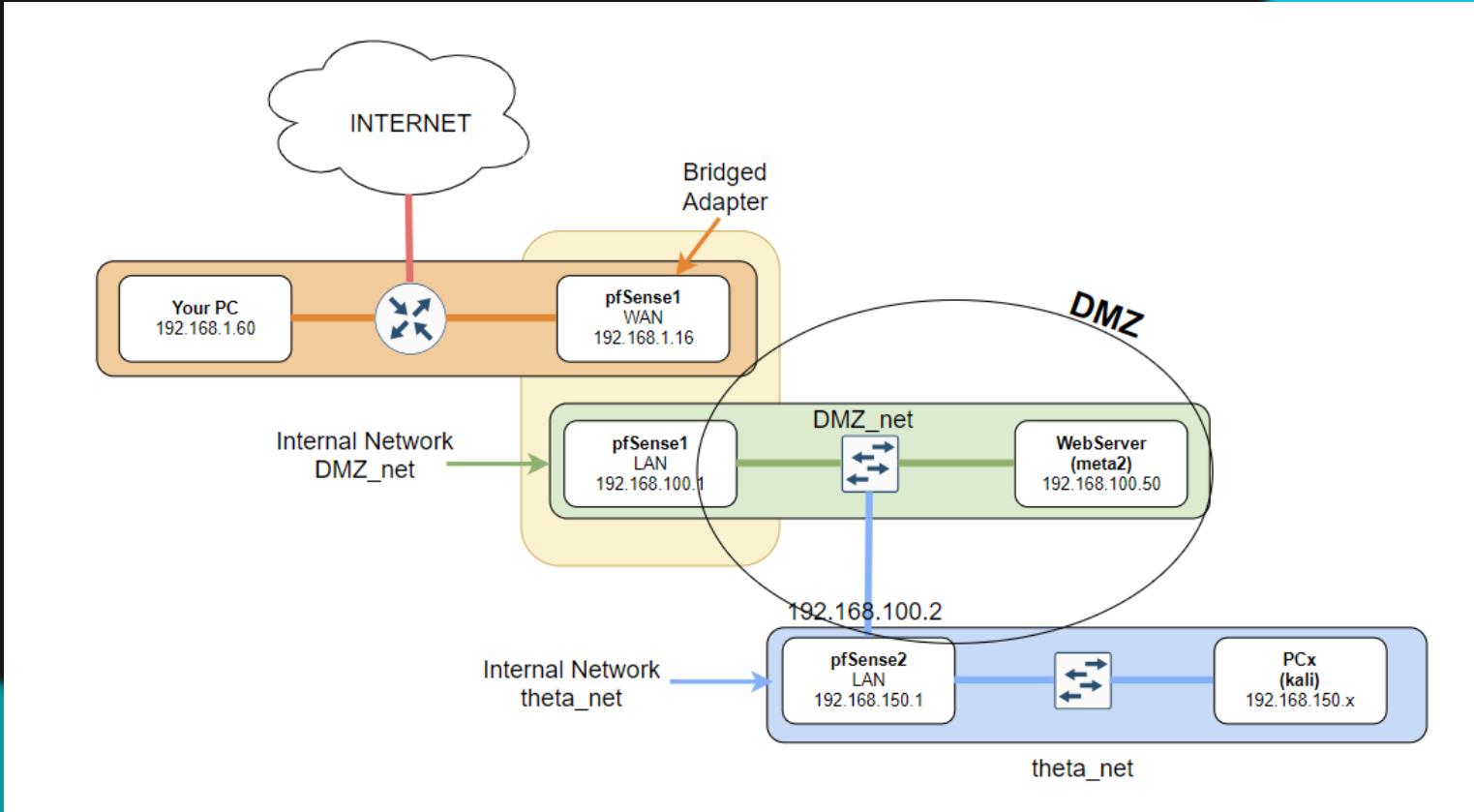
Le VLAN utente utilizzano assegnazione dinamica degli indirizzi tramite **DHCP**, mentre la VLAN di management adotta indirizzi statici per garantire stabilità e raggiungibilità degli apparati di rete.

Questa scelta **semplifica** la **gestione** e il **controllo** dell'infrastruttura.

Nome VLAN	ID VLAN	Subnet (Rete)	Netmask	Gateway (IP Switch Core)	DHCP	Static
Management	50	192.168.50.0/26	255.255.255.192	192.168.50.1	NON consigliato	Vedere tabella dedicata
Piano 1	101	192.168.101.0/26	255.255.255.192	192.168.101.1	192.168.101.20 – 192.168.101.62	NON consigliato
Piano 2	102	192.168.102.0/26	255.255.255.192	192.168.102.1	192.168.102.20 – 192.168.102.62	NON consigliato
Piano 3	103	192.168.103.0/26	255.255.255.192	192.168.103.1	192.168.103.20 – 192.168.103.62	NON consigliato
Piano 4	104	192.168.104.0/26	255.255.255.192	192.168.104.1	192.168.104.20 – 192.168.104.62	NON consigliato
Piano 5	105	192.168.105.0/26	255.255.255.192	192.168.105.1	192.168.105.20 – 192.168.105.62	NON consigliato
Piano 6	106	192.168.106.0/26	255.255.255.192	192.168.106.1	192.168.106.20 – 192.168.106.62	NON consigliato
NAS	90	192.168.90.0/26	255.255.255.192	192.168.90.1	NON consigliato	192.168.90.10

Dispositivo	Ruolo	IP statico
Router	Gateway VLAN 100	192.168.50.1
Switch principale	Gestione core	192.168.50.2
Switch Piano 1	Gestione access	192.168.50.3
Switch Piano 2	Gestione access	192.168.50.4
Switch Piano 3	Gestione access	192.168.50.5
Switch Piano 4	Gestione access	192.168.50.6
Switch Piano 5	Gestione access	192.168.50.7
Switch Piano 6	Gestione access	192.168.50.8
NAS	Backup / Config / Log	192.168.50.9
IDS2	Monitoraggio traffico interno	192.168.50.61
IPS	Protezione NAS	192.168.50.62

AMBIENTE DI SIMULAZIONE



Dispositivo	ID VLAN	Indirizzo IP	Subnet Mask	Gateway	Interfaccia / Ruolo	Zona (Segmento)	Note
Router	N/A	10.255.0.1	255.255.255.252 (/30)	N/A	Interfaccia verso Firewall interno	Transito Interno	che contiene solo 2 indirizzi utilizzabili (best practice) Router - Firewall (int)
Firewall (Interno)	N/A	10.255.0.2	255.255.255.252 (/30)	10.255.0.1	Interfaccia verso Router	Transito Interno	
Firewall (Interno)	99	172.16.1.1	255.255.255.248 (/29)	N/A	Interfaccia verso DMZ	DMZ	
Switch	99	172.16.1.2	255.255.255.248 (/29)	172.16.1.1	VLAN Management	DMZ	IP per gestire lo switch (Switch Managed)
IDS_DMZ	99	172.16.1.3	255.255.255.248 (/29)	172.16.1.1	IDS	DMZ	
Server Web	99	172.16.1.5	255.255.255.248 (/29)	172.16.1.1	Server Web (meta)	DMZ	Accessibile dal piano "sviluppatori" tramite regola Firewall (FW Int), non tramite VLAN diretta
Firewall (Esterno)	99	172.16.1.6	255.255.255.248 (/29)	N/A	Interfaccia verso DMZ	DMZ	contiene 6 indirizzi per minimizzare i n. di IP
Firewall (Esterno)	N/A	203.0.113.2*	255.255.255.248 (/29)	203.0.113.1 (ISP)	Interfaccia verso ISP	WAN (Pubblica)	*è un esempio standard di IP pubblico. Nella realtà, qui inseriamo l'IP statico fornito dal provider Internet.

Questa tabella definisce la topologia logica della nostra rete perimetrale:
utilizziamo segmenti di rete ristretti (/30 e /29) per **isolare** il traffico di transito dalla DMZ, dove risiedono i servizi pubblici come il Web Server e l'IDS, garantendo che nessun accesso esterno raggiunga direttamente la rete interna senza passare dai firewall.

REGOLE FIREWALL PERIMETRALE

pfSense1	Interfaccia	Protocollo	Action	Source	Destination	Port
	WAN					
		Any	Pass	Any	172.16.1.5	80
	LAN (verso DMZ_net)					
		Any	Block	LAN net	172.16.1.6	443/80/22
		Any	Pass	LAN net	Any	Any

Qui l'obiettivo è **minimizzare l'esposizione pubblica**.

Come mostrato nella prima regola WAN, l'unica porta aperta verso l'esterno è la **80 (HTTP)** indirizzata esclusivamente al Web Server (172.16.1.5); tutto il resto è bloccato di default. Inoltre, sulla l'interfaccia interna, è stata applicata una regola di **Hardening**: l'accesso amministrativo al firewall stesso (SSH/HTTPS verso 172.16.1.6) viene esplicitamente bloccato per prevenire tentativi di manomissione della configurazione, anche se provenienti dalla rete interna.

REGOLE FIREWALL INTERNO

Definisce il **comportamento** di sicurezza (ovvero *chi* può parlare con *chi*).

Le regole qui configurate rendono operativa la segmentazione logica: sfruttiamo le subnet appena create per applicare controlli granulari, garantendo che l'accesso alla DMZ sia concesso solo a specifici ruoli (Management e Sviluppatori) e bloccato per tutto il resto della LAN, **riducendo** drasticamente i **rischi** di movimenti laterali non autorizzati

pfSense2	Interfaccia	Protocollo	Action	Source	Destination	Port
	WAN					
		Any	Block ✘	Any	Any	Any
	LAN (tra DMZ_net e theta_net)					
		TCP / UDP	Pass ✅	192.168.100.0/26 (Management Net)	172.16.1.0/29 (DMZ net)	Any
		TCP / UDP	Pass ✅	192.168.106.0/26 ("Sviluppatori" Net)	172.16.1.5 (Web Server)	Any
		Any	Block ✘	LAN net (101.x,102.x,...,105.x)	172.16.1.0/29 (DMZ net)	Any
		Any	Pass ✅	LAN net (101.x,102.x,...,106.x)	Any	Any

02

Certificazione Rete



The screenshot shows the Kali Linux desktop environment. In the foreground, a terminal window displays the output of a script named 'bw.py'. The script is configured to scan a target at 192.168.100.50:80 using various HTTP methods. The output shows successful responses for all methods tested (OPTIONS, GET, POST, HEAD, PUT, DELETE, PATCH) with a body length of 4145 bytes. A report file is saved at the end.

```
Target: 192.168.100.50:80 Path: /phpMyAdmin/index.php
Target: 192.168.100.50:80 Path: /phpMyAdmin/index.php
Testing verbs: OPTIONS, GET, POST, HEAD, PUT, DELETE, PATCH
-> OPTIONS /phpMyAdmin/index.php
  200 OK Allow- Location- BodyLen=4145
-> GET /phpMyAdmin/index.php
  200 OK Allow- Location- BodyLen=4145
-> POST /phpMyAdmin/index.php
  200 OK Allow- Location- BodyLen=4145
-> HEAD /phpMyAdmin/index.php
  200 OK Allow- Location- BodyLen=0
-> PUT /phpMyAdmin/index.php
  200 OK Allow- Location- BodyLen=4145
-> DELETE /phpMyAdmin/index.php
  200 OK Allow- Location- BodyLen=4145
-> PATCH /phpMyAdmin/index.php
  200 OK Allow- Location- BodyLen=4145

Sanning Done.
Saved report: /home/kali/Desktop/Python-programming/bw/results_192.168.100.50_80_20251218_042538.txt
Tip: Open the .txt report and compare status codes, Allow/Location/Set-Cookie across verbs.

33 Server: Apache/2.2.8 (Ubuntu) DAV/2
34 Location: http://192.168.100.50/dvwa/
```

HTTP Scanner

Lo script implementa un tool progettato per analizzare la configurazione dei server **HTTP**. Utilizzando la libreria `http.client`, il software interroga un **URL** specifico testando sequenzialmente diversi metodi HTTP (GET, POST, PUT, DELETE, OPTIONS, ecc.) per identificare quelli abilitati.

Per ogni richiesta, lo script analizza la **HTTP Response**, estraendo codici di stato, intestazioni (*headers*) come "Allow" e "Location", e un'anteprima del corpo della risposta. Attraverso un'interfaccia grafica implementata attraverso la libreria `tkinter`, l'utente può configurare il target e generare automaticamente un **report tecnico** (.txt).

Questo processo permette di rilevare vulnerabilità di configurazione, come metodi pericolosi (es. PUT o DELETE) lasciati inavvertitamente esposti, facilitando il controllo delle policy di sicurezza del server web.

Port Scanner

Lo script implementa un tool di **network discovery** e **vulnerability probing** strutturato in due fasi sequenziali. Inizialmente, il software sfrutta il modulo subprocess per eseguire una verifica di raggiungibilità tramite protocollo **ICMP (Ping)**, validando lo stato "up" dell'host target assicurandoci che la macchina che cerchiamo di contattare sia online.

Superata la fase di discovery, lo script esegue una scansione **TCP Connect** iterativa utilizzando la libreria **socket**: attraverso il metodo **connect_ex()**, viene tentato l'**handshake** su un range definito di porte, identificando come "aperte" solo le risorse che restituiscono un valore di ritorno nullo.

L'intero processo è ottimizzato mediante la gestione dei **timeout** e del'esecuzione automatizzata , permettendo al tecnico di mappare i servizi esposti e verificare in tempo reale l'efficacia delle policy di filtraggio del Firewall Perimetrale.

```
port_scanner.py > port_scan
40 def port_scan(target, start_port, end_port):
41
42     # --- NUOVO BLOCCO: CONTROLLO HOST ---
43     print(f"\n[*] Verifica stato host {target_ip} in corso...")
44
45     if not check_host_up(target_ip):
46         print(f"[!] Host {target_ip} non raggiungibile (sembra spento o blocca i ping).")
47         print("[!] Scansione annullata.")
48         return
49     else:
50         print(f"[*] Host attivo! Inizio scansione...")
51         #
52
53     print(f"[*] Scansione porte {start_port}-{end_port}")
54
55
56
57
58
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
> $ /usr/bin/python /home/kali/Desktop/Python-programming/bw/port_scanner.py
Inserisci IP o hostname: 192.168.100.50
Porta iniziale: 20
Porta finale: 90
[*] Verifica stato host 192.168.100.50 in corso...
[*] Host attivo! Inizio scansione...
[*] Scansione porte 20-90
[*] Ora di inizio: 2025-12-18 04:42:17.935069
[+] Porta 21 APERTA
[+] Porta 22 APERTA
[+] Porta 23 APERTA
[+] Porta 25 APERTA
[+] Porta 53 APERTA
[+] Porta 80 APERTA
Scansione completata: 2025-12-18 04:42:19.340997
Totale porte aperte trovate: 6
(kali㉿kali)-[~/Desktop/Python-programming/bw]
```

Sniffer Tool

Il programma permette di osservare il comportamento reale della rete.

Questo codice realizza uno sniffer di rete usando la libreria **Scapy** per intercettare pacchetti in tempo reale.

La funzione **packet_handler()** viene chiamata ogni volta che un pacchetto viene catturato.

Viene aggiunto un timestamp per sapere quando ogni pacchetto è stato intercettato.

Il codice analizza prima i pacchetti **ARP**, fondamentali per la risoluzione **IP-MAC** nella rete locale.

Per ARP distingue le operazioni **who-has** e **is-at**, mostrando IP e MAC sorgente e destinazione.

Successivamente intercetta pacchetti **IP/TCP**, tipici delle comunicazioni di rete.

Per ogni pacchetto TCP stampa indirizzi IP, porte sorgente/destinazione e flag TCP.

Mostra anche la dimensione del payload, utile per analizzare il traffico dati.

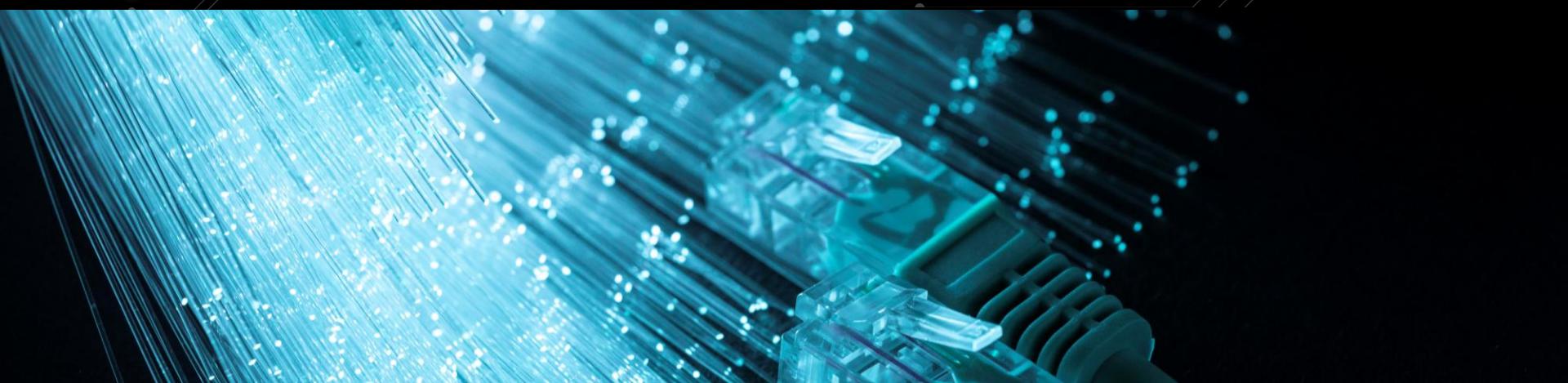
```
sniffer.py > ...
4  def packet_handler(packet):
5      timestamp = datetime.now().strftime("%H:%M:%S")
6
7      # ===== ARP ===== essendo ARP un protocollo di livello 2/3 va già fatto
8      if packet.haslayer(ARP):      #verifica se il pacchetto contiene l'Arp
9          arp = packet[ARP]
10
11         if arp.op == 1:
12             op = "who-has"
13         elif arp.op == 2:
14             op = "is-at"
15         else:
16             op = f"op={arp.op}"
17
18         print(
19             f"{timestamp} ARP {op} "
20             f"{arp.psrc} -> {arp.pdst} "      #{arp.psrc}= IP SORGENTE / {arp.pdst}= IP DESTINAZIONE
21             f"\n{arp.hwsrc}" "# ----> MAC sorgente
22         )
23
24     # ===== TCP =====
25     elif packet.haslayer(IP) and packet.haslayer(TCP):      #verifica che il pacchetto sia un IP con TCP
26         ip = packet[IP]
27         tcp = packet[TCP]
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
04:46:51 TCP 192.168.150.10:34730 -> 13.107.213.43:443 FLAGS=[A] PAYLOAD=1428B
04:46:51 TCP 13.107.213.43:443 -> 192.168.150.10:34730 FLAGS=[A] PAYLOAD=0B
04:46:51 TCP 13.107.213.43:443 -> 192.168.150.10:34730 FLAGS=[PA] PAYLOAD=99B
04:46:51 TCP 192.168.150.10:34730 -> 13.107.213.43:443 FLAGS=[A] PAYLOAD=0B
04:46:51 TCP 192.168.150.10:34730 -> 13.107.213.43:443 FLAGS=[PA] PAYLOAD=846B
04:46:51 TCP 13.107.213.43:443 -> 192.168.150.10:34730 FLAGS=[PA] PAYLOAD=281B
04:46:51 TCP 192.168.150.10:34730 -> 13.107.213.43:443 FLAGS=[A] PAYLOAD=0B
04:46:51 TCP 192.168.150.10:34730 -> 13.107.213.43:443 FLAGS=[PA] PAYLOAD=74B
04:46:51 TCP 13.107.213.43:443 -> 192.168.150.10:34730 FLAGS=[PA] PAYLOAD=62B
04:46:51 TCP 192.168.150.10:34730 -> 13.107.213.43:443 FLAGS=[A] PAYLOAD=0B
04:46:51 TCP 192.168.150.10:34730 -> 13.107.213.43:443 FLAGS=[PA] PAYLOAD=92B
04:46:51 TCP 13.107.213.43:443 -> 192.168.150.10:34730 FLAGS=[PA] PAYLOAD=31B
04:46:51 TCP 192.168.150.10:34730 -> 13.107.213.43:443 FLAGS=[A] PAYLOAD=0B
04:46:51 TCP 192.168.150.10:34730 -> 13.107.213.43:443 FLAGS=[PA] PAYLOAD=31B
04:46:51 TCP 192.168.150.10:34730 -> 13.107.213.43:443 FLAGS=[PA] PAYLOAD=265B
04:46:51 TCP 13.107.213.43:443 -> 192.168.150.10:34730 FLAGS=[A] PAYLOAD=0B
```

03

Ricerca e Preventivo



● Analisi dei Requisiti Tecnici

La selezione dell'hardware (Firewall, Switch, IDS/IPS, Router) è stata determinata dai requisiti di sicurezza e segmentazione emersi in **fase di progettazione**.

Abbiamo cercato dispositivi capaci di sostenere il carico delle regole dei firewall e del funzionamento delle switch senza colli di bottiglia.

● Ricerca di mercato

L'analisi dei bisogni del cliente ha guidato la scelta del fornitore per i componenti hardware **PowerDigit Srl**.

Tra i diversi fornitori contattati, era l'unico in grado di offrire tutto il materiale necessario a prezzo d'ingrosso.

La scelta di un unico fornitore ha permesso di **ridurre tempi e costi** di spedizione.

Questo ha garantito prodotti ad alte prestazioni a costi convenienti per l'azienda e per il cliente.



04 Conclusione

Grazie!

Do you have any questions?

info@spegnietriaccendi.com
+39 333 333 3333
spegnietriaccendi.com

