Protocollowolf

PRESENTAZIONE PROGETTO TETHA

Protocollowolf

Chi siamo

Quando un disastro digitale paralizza un'azienda, loro sono la telefonata da fare. Applicano un protocollo freddo e infallibile per ripulire il caos in tempo record, operando con una calma quasi soprannaturale. Sono i chirurghi della crisi, efficienti e invisibili.

Richiesta

TETHA ci chiede di sviluppare un progetto di rete per la loro infrastruttura IT. I requisiti e i componenti necessari sono: Struttura dell'edificio: 6 piani, dispositivi previsti: 20 computer per piano, per un totale di 120 computer

Protocollowolf

Infrastruttura di Rete

Infrastruttura di rete

Nella nostra idea di infrastruttura di rete, ogni piano dei complessivi 6 piani sarà composto da:

20 PC

1 SWITCH

Infrastruttura di rete

Si propone una soluzione che comprende una segmentazione di rete per mezzo di VLAN, esclusive per ogni piano, ma comunicanti tra loro mediante dispositivi di quali:

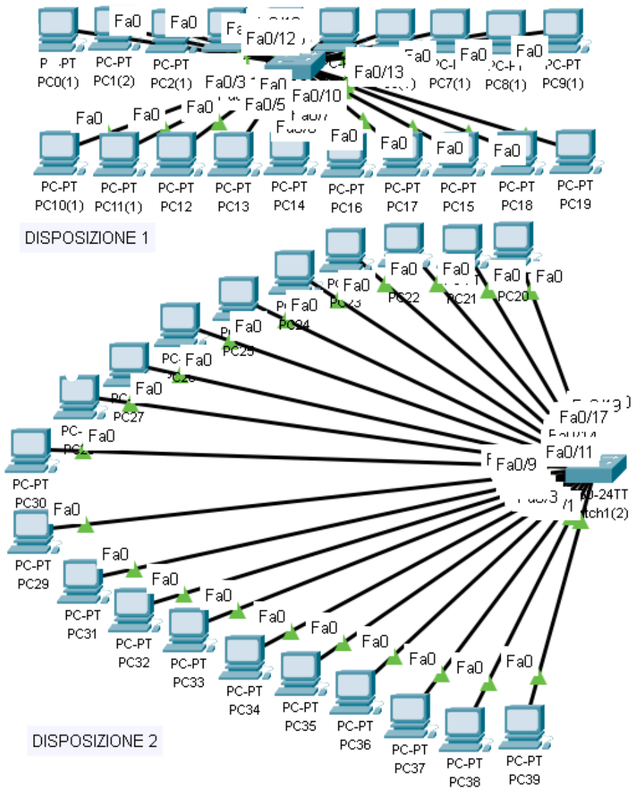
1 SERVER DHCP

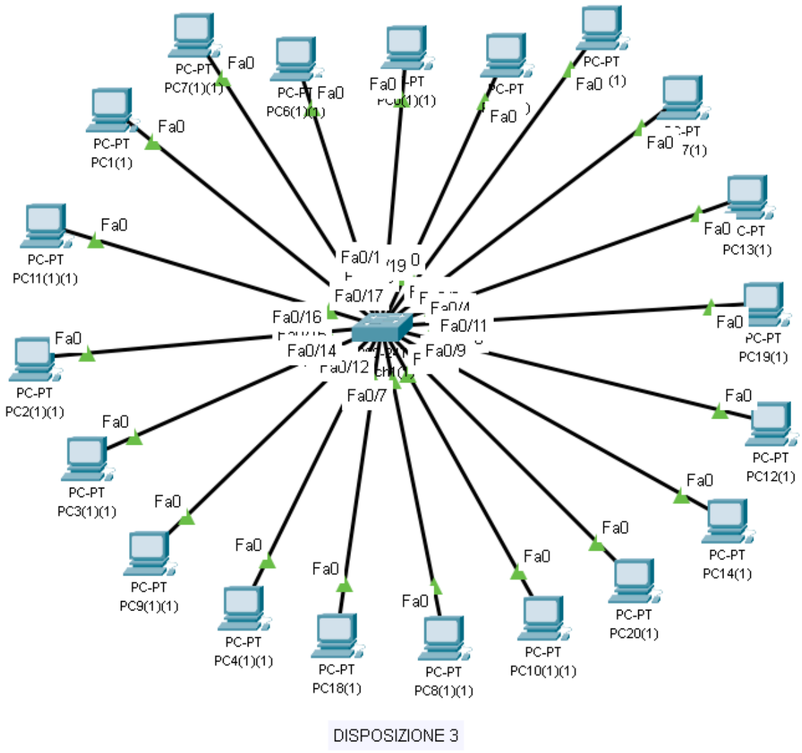
1 NAS

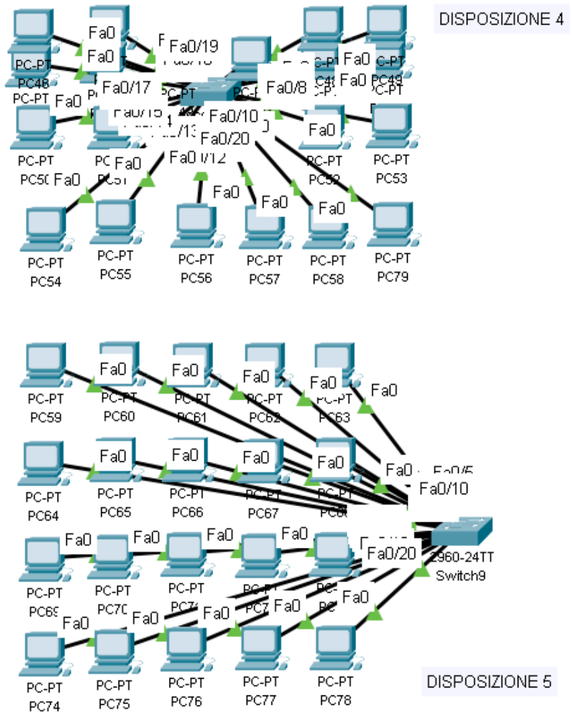
1 SWITCH MULTILAYER

Infrastruttura di rete

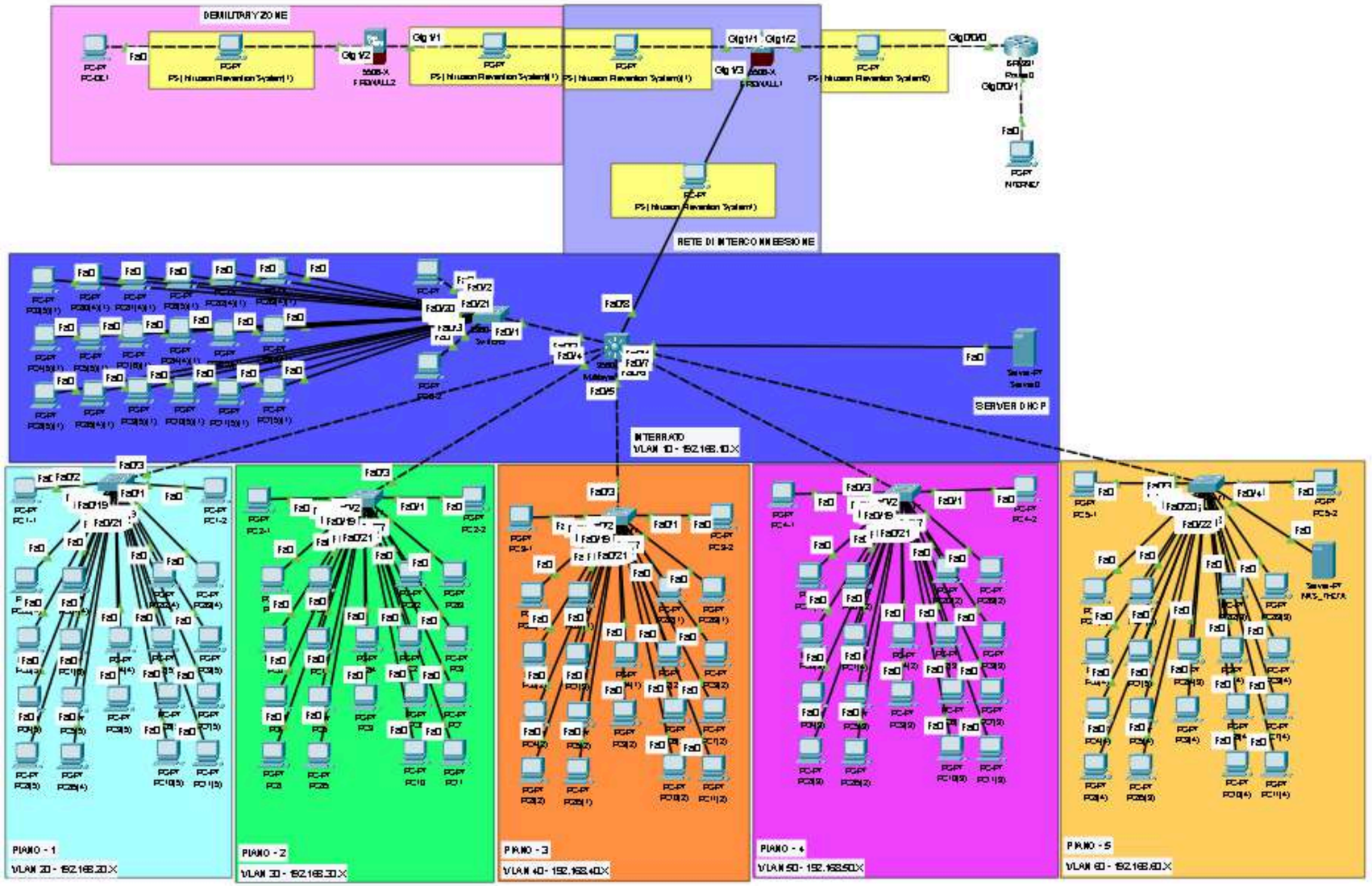
Valutazione della disposizione di rete







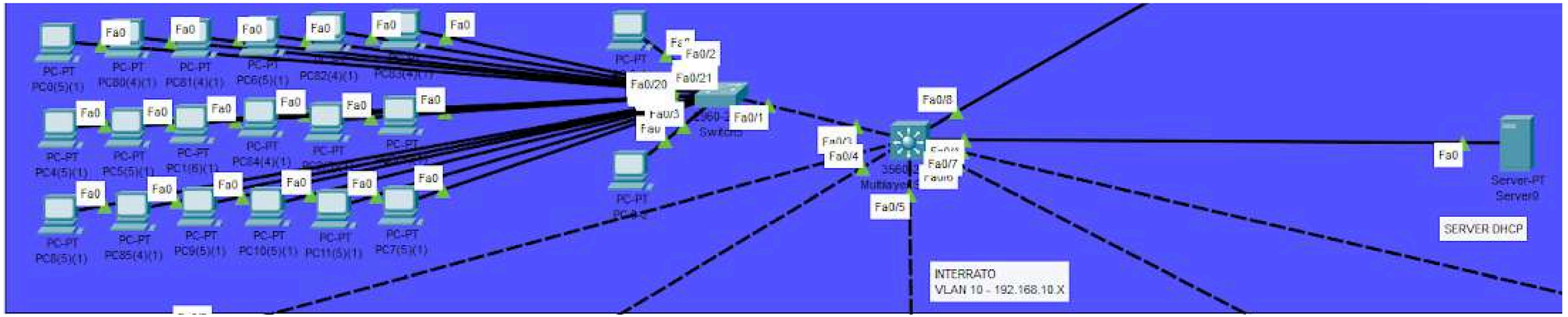
Infrastruttura di rete



Infrastruttura di rete

PIANO INTERRATO

Al centro della rete abbiamo uno **switch multi-layer**, che funge da dorsale, collegando un “**SERVER DHCP**” che assegna automaticamente gli indirizzi IP ai PC, con sei switch di accesso, ognuna dei quali con una zona ed un piano distinto mediante specifiche VLAN (nominate in base al piano di riferimento).

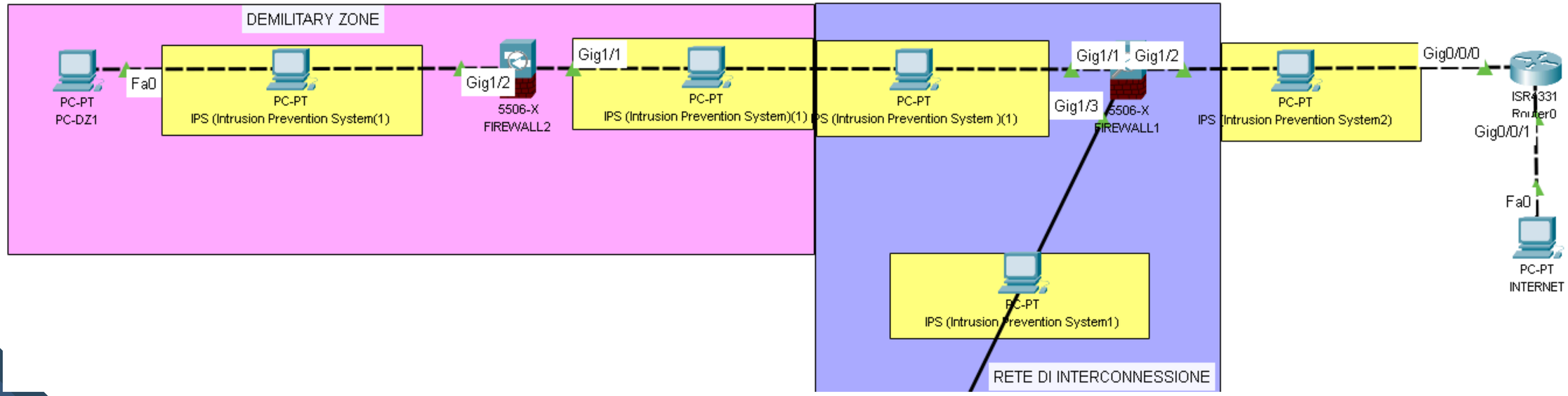


Infrastruttura di rete

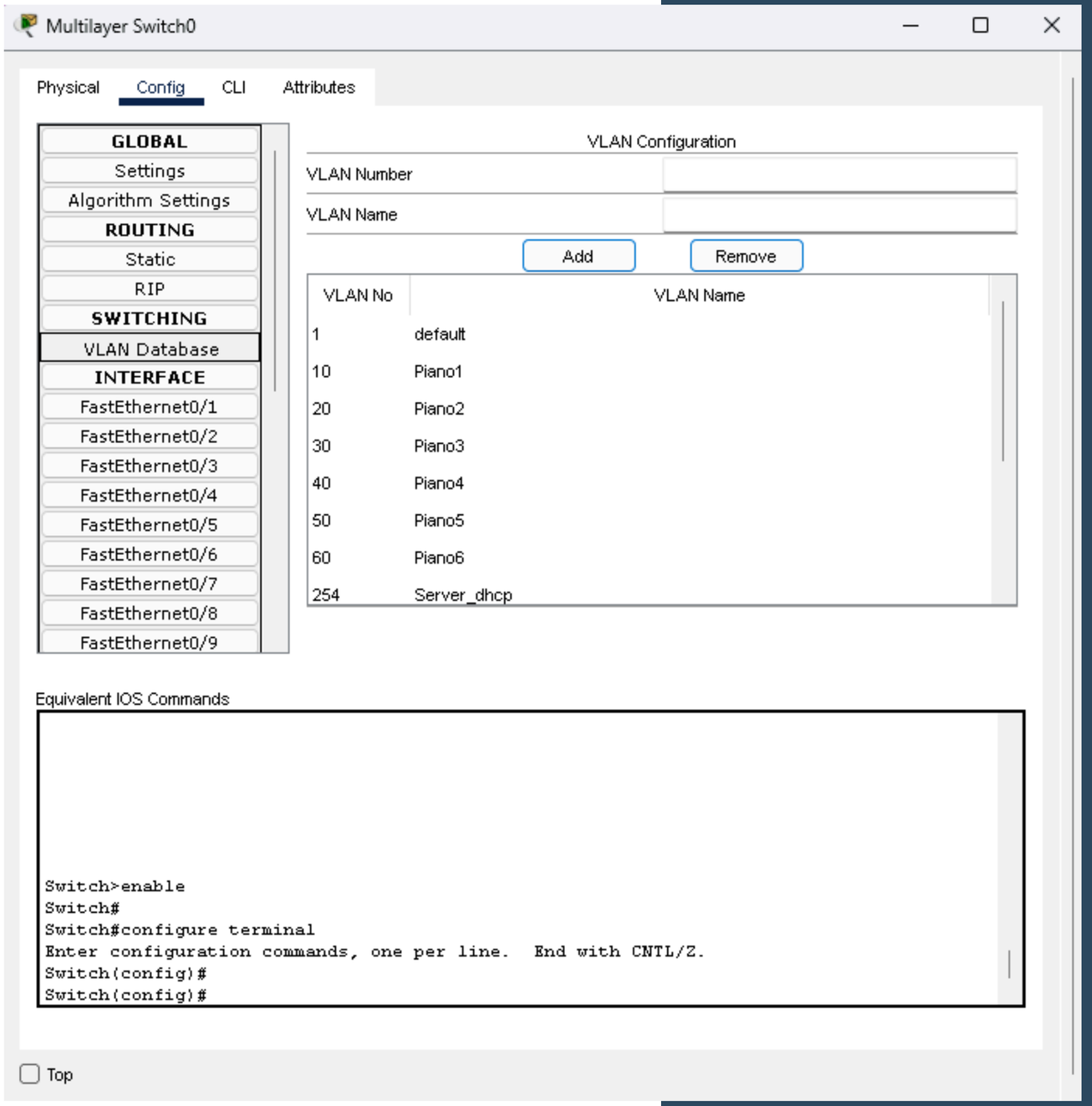
DEMILITARY ZONE

Come si puo vedere in figura oltre alla rete isolata che rappresenta la DMZ (Rete Web Server) abbiamo anche un’altra zona chiamata “RETE DI INTERCONNESSIONE” che contiene:

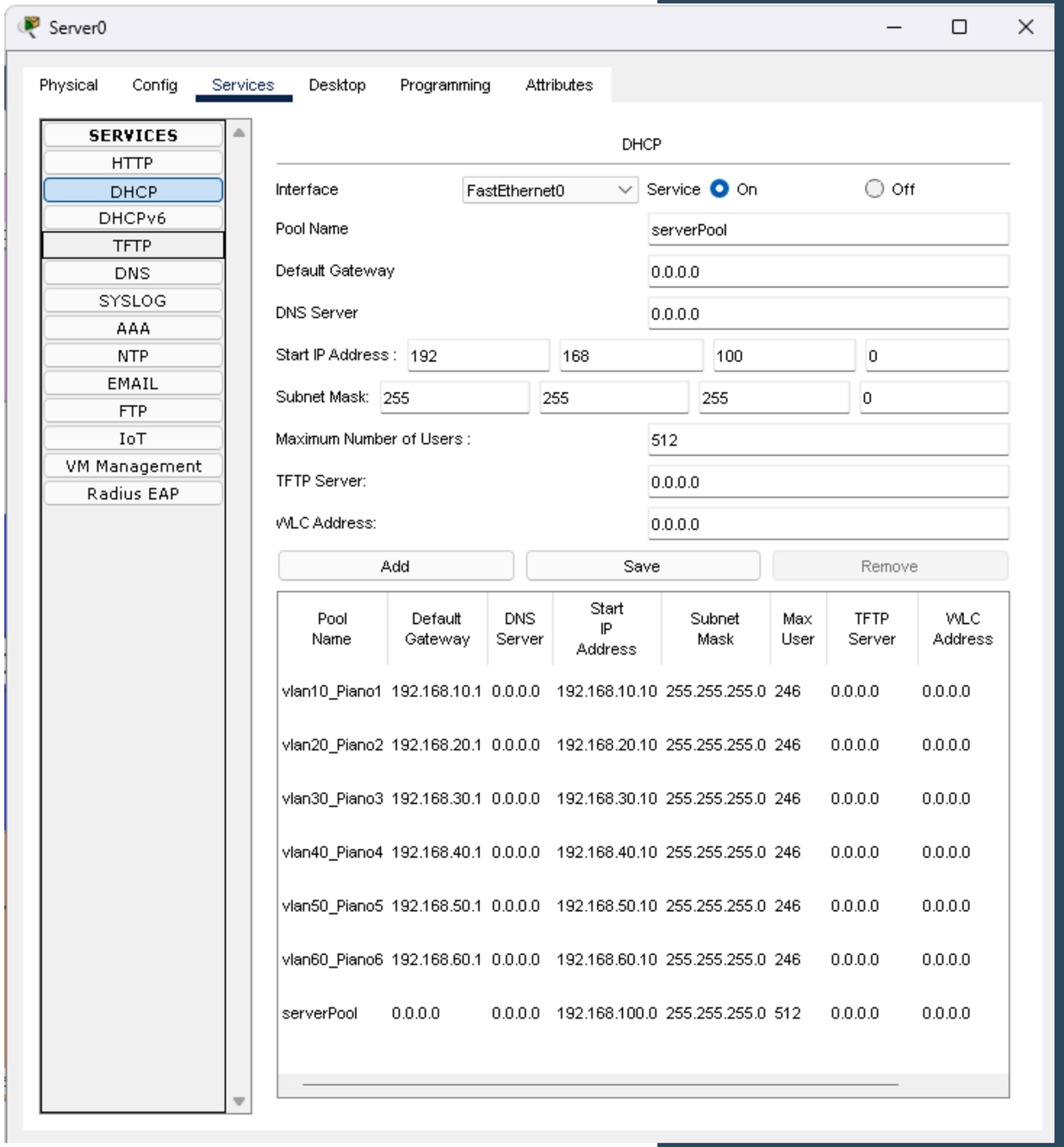
- **5 IPS**
- **2 FIREWALL**



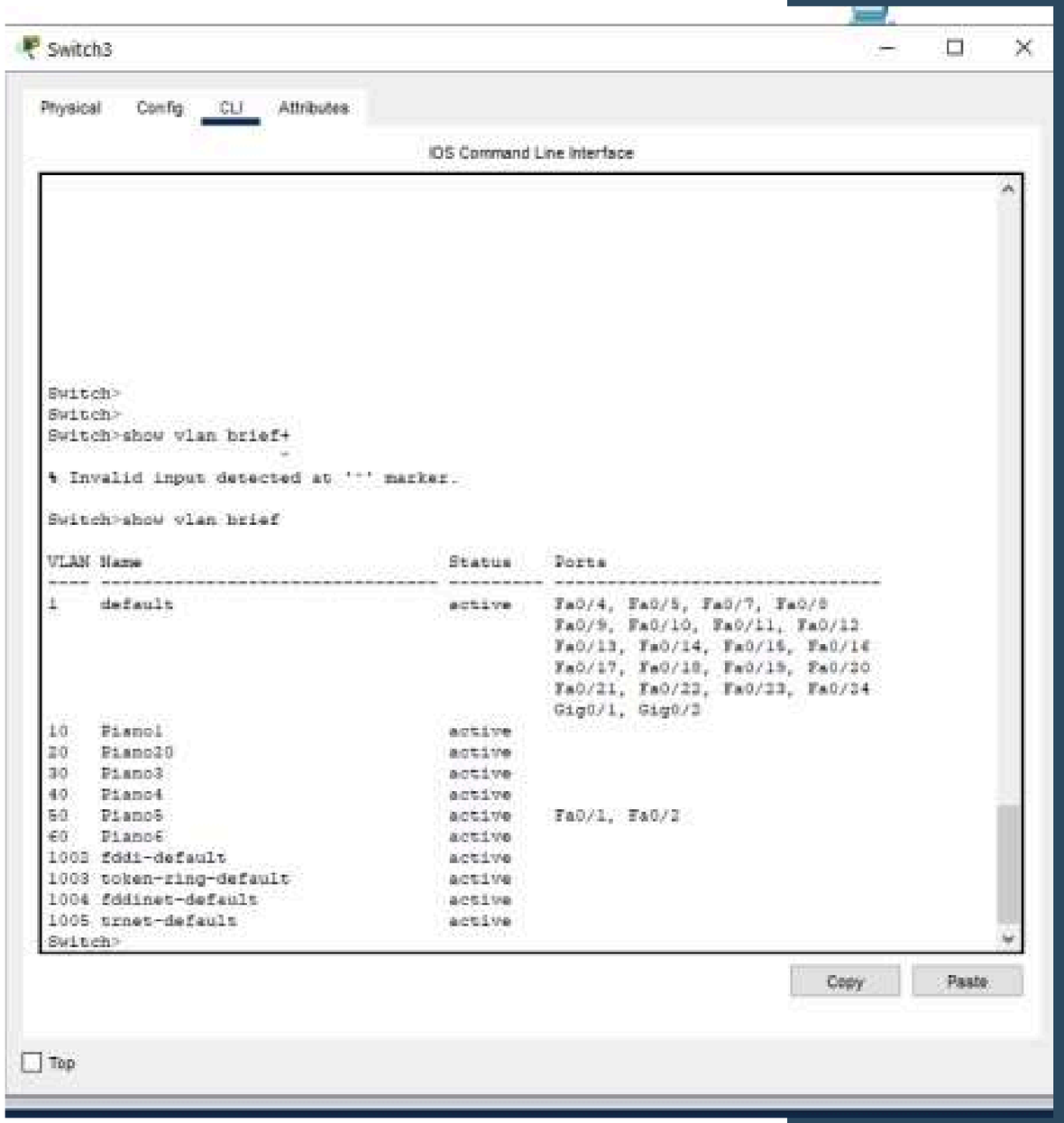
Configurazione Multilayer Switch



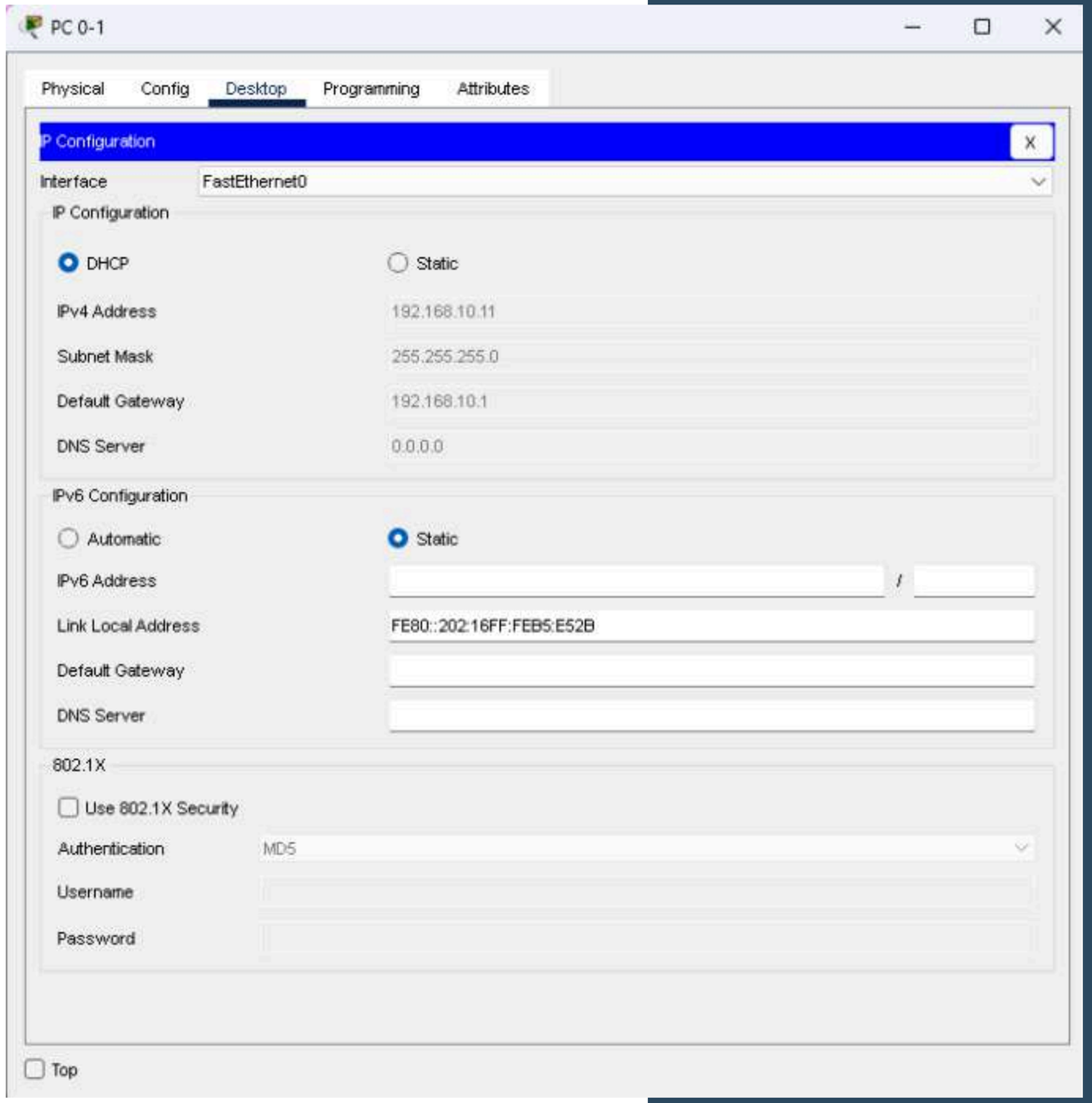
Configurazione POOL DHCP (Server)



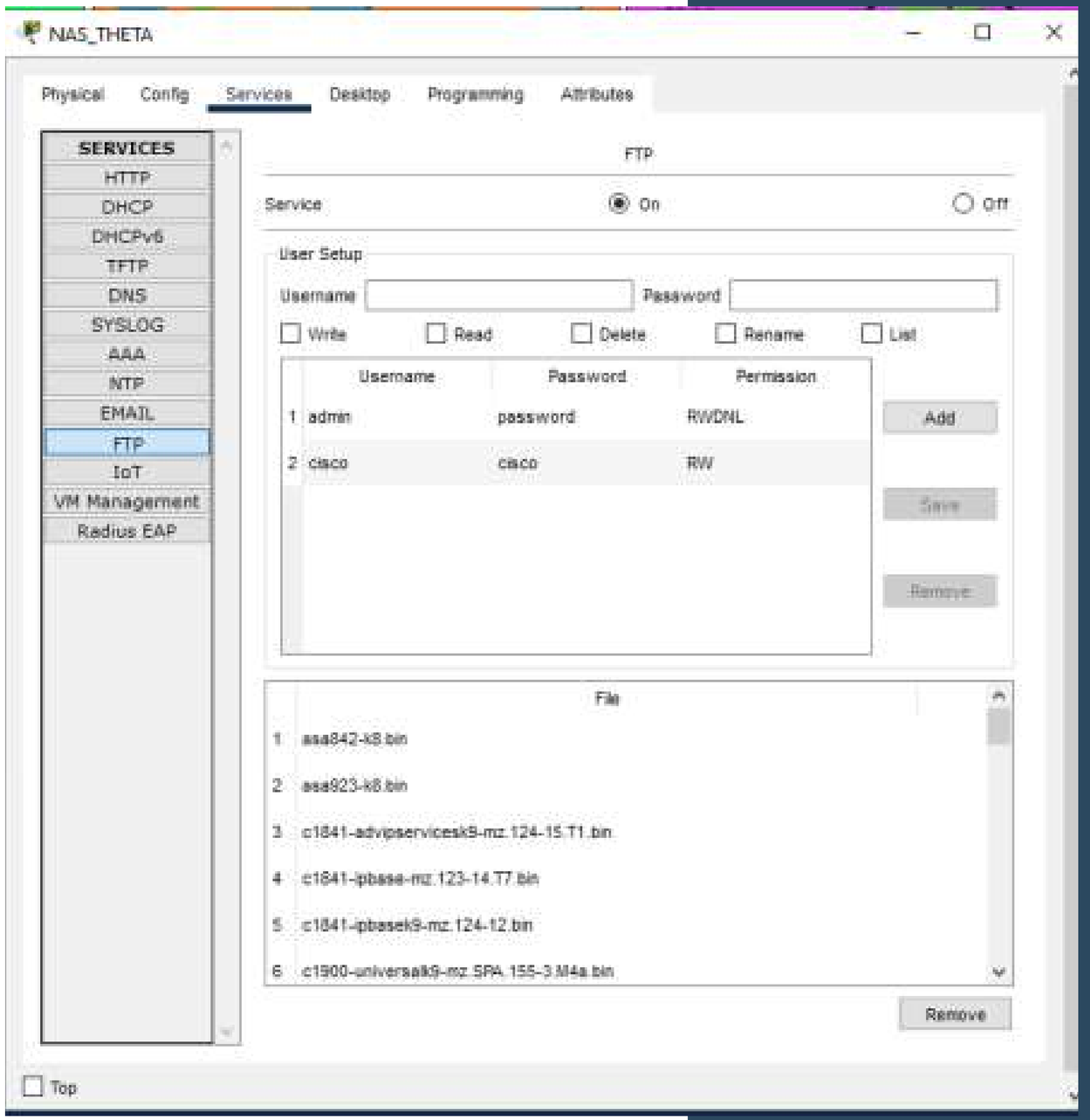
Configurazione Switch 2960



Configurazione PC

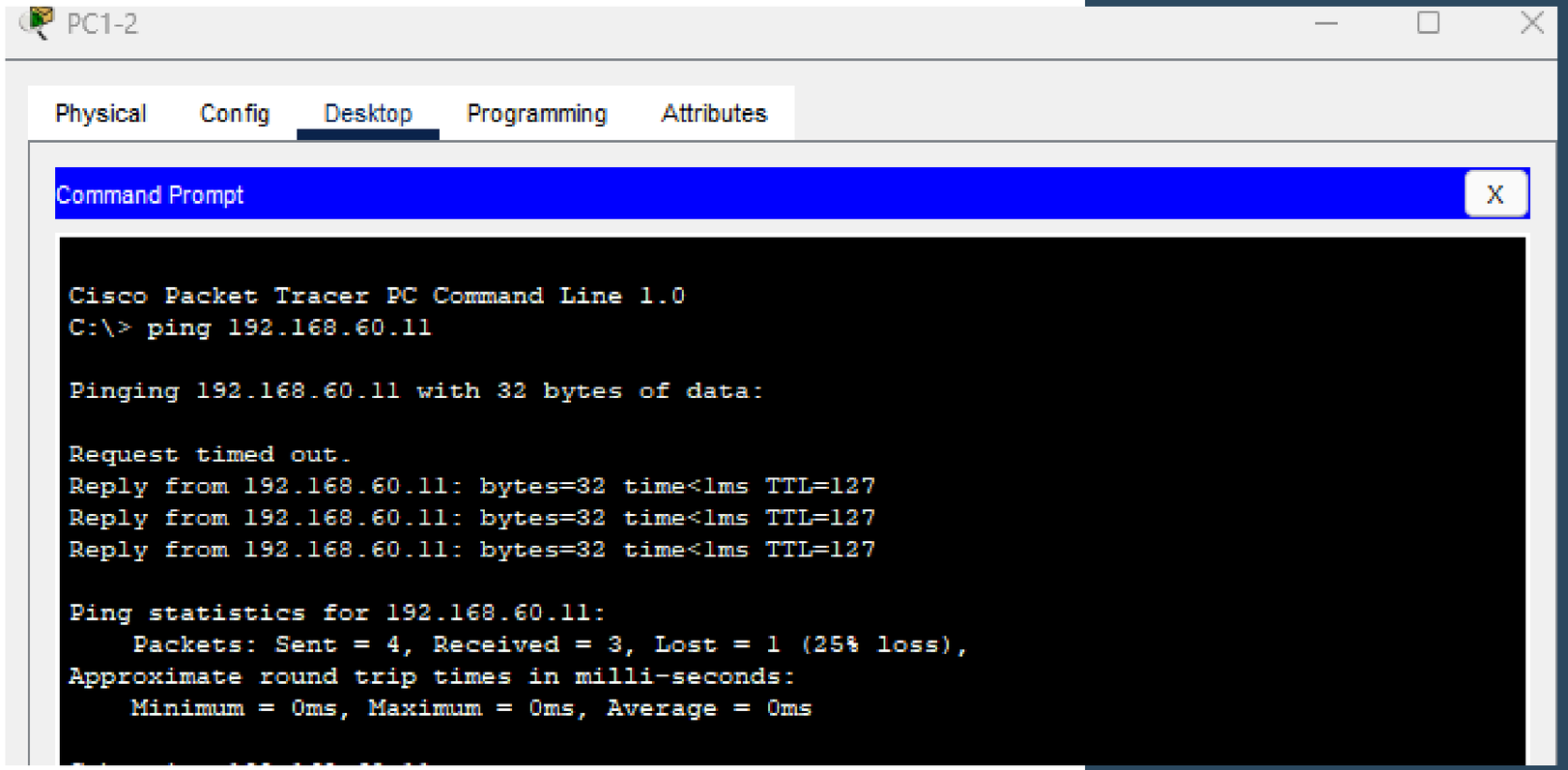


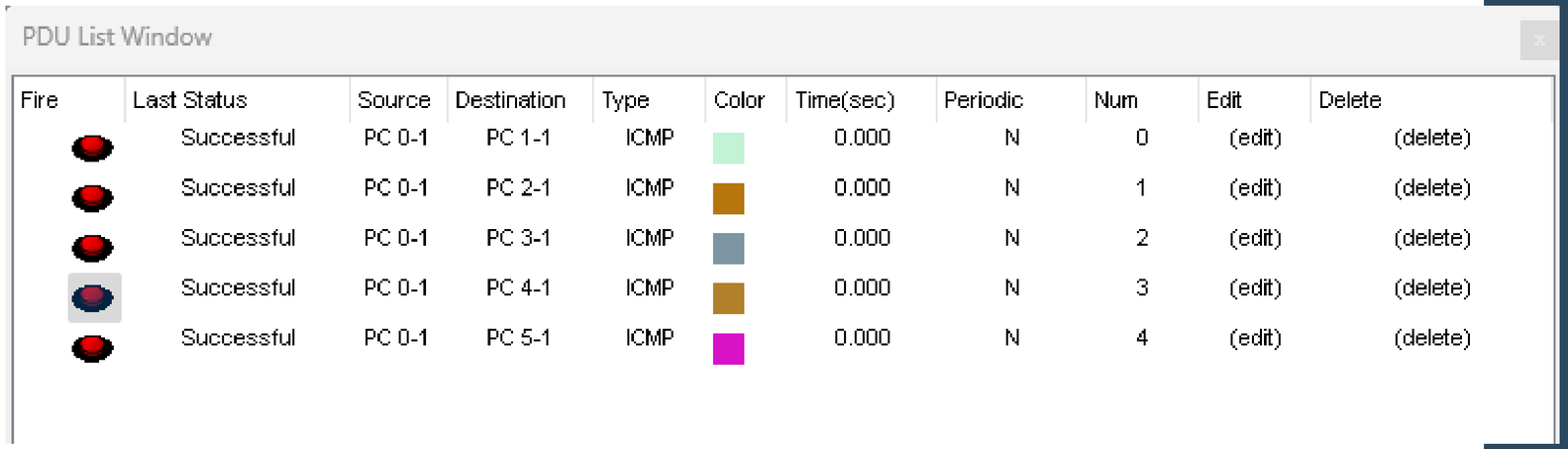
Configurazione Server NAS



Ping Test (PC)

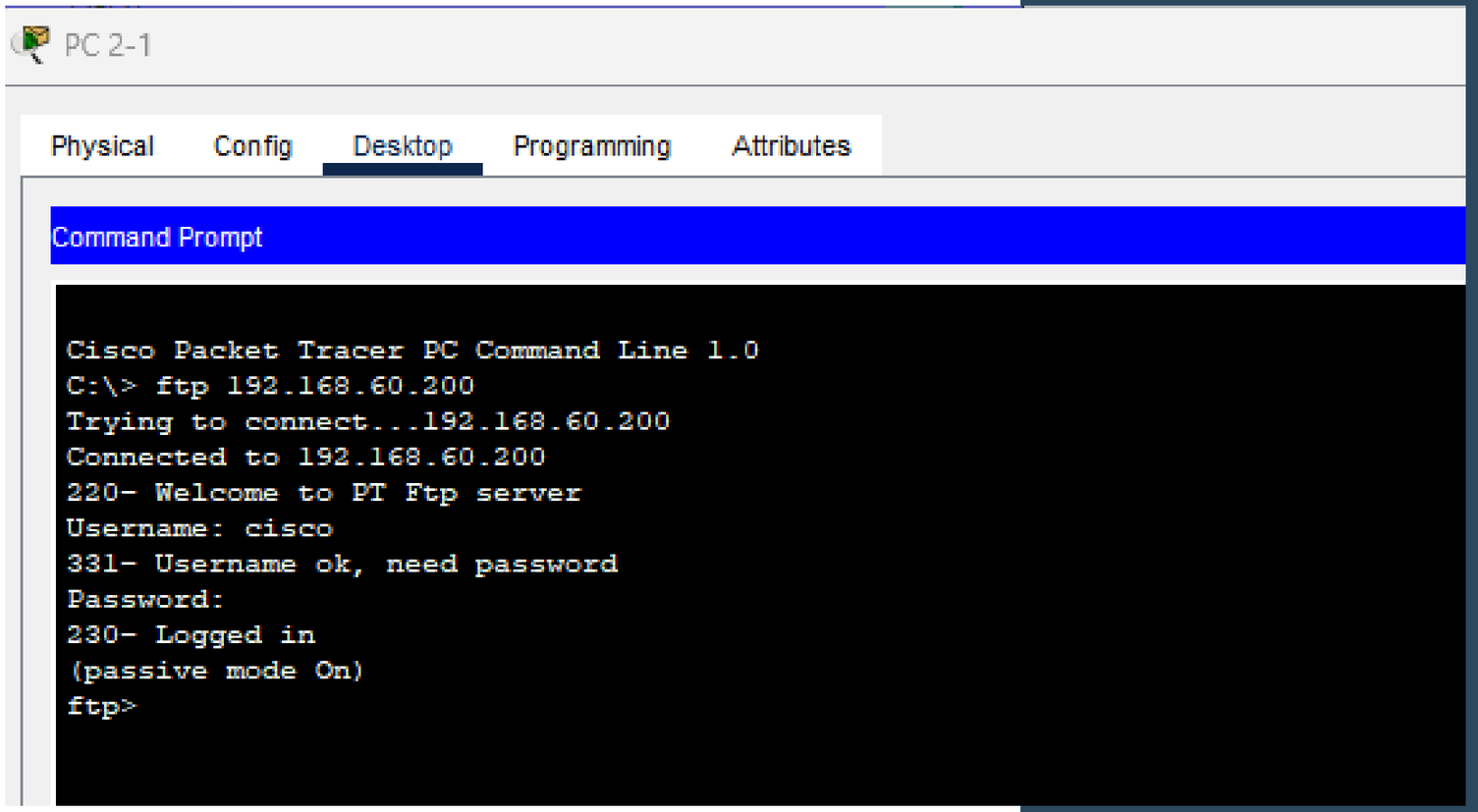
Ping Test tra un PC del Piano 1 e un PC dell'ultimo piano





Ping Test (NAS)

Ping Test per vedere l’effettiva comunicazione tra un PC e un server NAS



Considerazioni sulla rete

Conclusioni

In conclusione, in caso di un attacco esterno, la nostra azienda è protetta in modo da permettere una risposta immediata ed efficace. Questo è reso possibile anche dai cinque indirizzi IP che monitorano costantemente il traffico, agendo come sentinelle sulla rete.

Protocollowolf

Piano di Sicurezza

Piano di Sicurezza

La nostra soluzione difensiva è composta da due **firewall indipendenti** guidati da una serie di regole logiche implementate su misura per le necessità della Theta in modo garantire la sicurezza.

Abbiamo scelto di usare due firewall indipendenti uno dal altro e abbiamo applicato **5 sensori pro attivi** al controllo dei punti sensibili della rete.

Il primo è dedicato al controllo **perimetrale** della rete, il secondo è dedicato a proteggere il **Web Server.**

La suddivisione delle reti, l'utilizzo dei due Firewall con i sensori ci permette nel il Web Server sia compromesso di garantire la sicurezza della rete interna dei lavoratori."

2 FIREWALL

Piano di Sicurezza

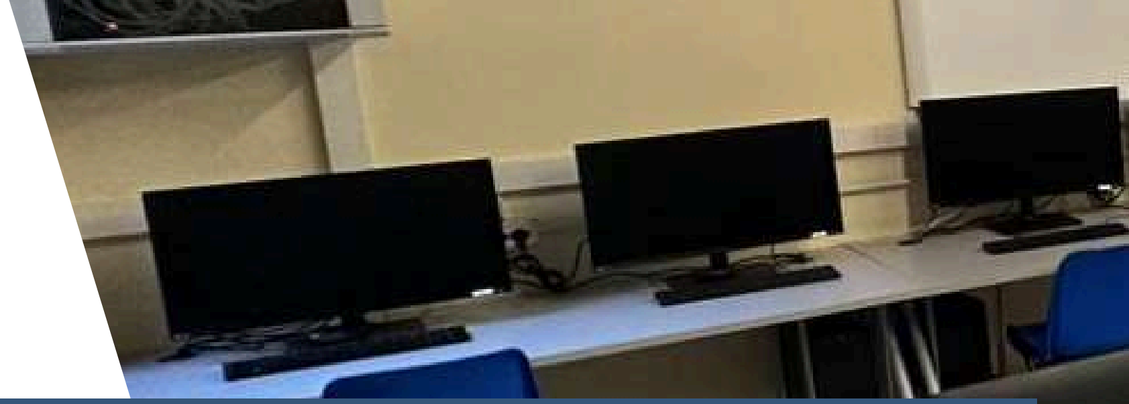
Firewall

1 - **Firewall Perimetrale**: Il primo strato di difesa, gestisce il traffico globale.
2 - **Firewall Dedicato al Web Server**: Un secondo firewall isola il Web Server in una rete dedicata, fornendo una protezione aggiuntiva e specializzata.

Sensori

Abbiamo incluso **5 sensori proattivi IPS** cosi da monitorare il traffico in tempo reale, identificando minacce tramite firme e anomalie bloccando attivamente gli attacchi prima che causino danni.

Impostazione laboratorio



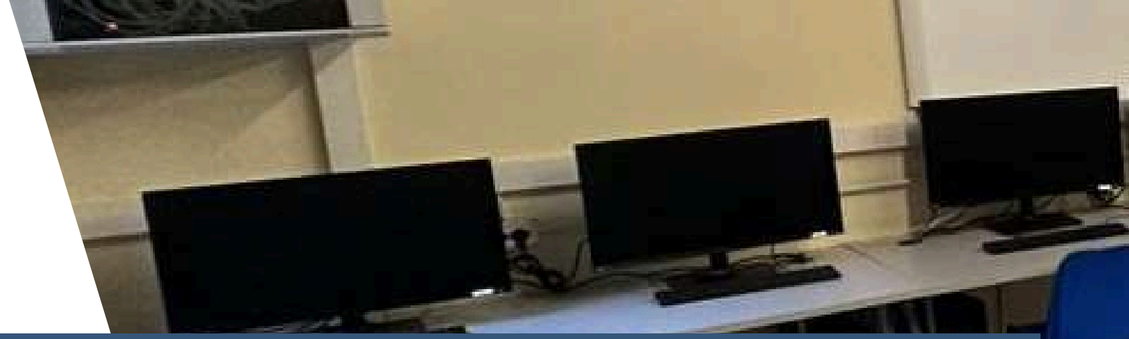
Macchine Virtuali

- 1 macchina virtuale **Kali Linux** rappresentera' un dipendente Theta che lavora da un ufficio situato in uno dei piani della sede
- 2 macchine virtuali **Pfsense**. Ognuna è un firewall dedicato per proteggere la rete aziendale e il webserver
- 1 macchina virtuale **Metasploitable2**, il webserver

Reti Laboratorio

• Rete "**intnet**" rappresenta la rete interna della sede aziendale
• Rete "**DMZ-FW_Link**" rappresenta una rete di interconnessione per creare un ulteriore strato di protezione a difesa del webserver
• Rete "**DMZ-Web_LAN"** rappresenta la rete del web server

Impostazione laboratorio



Configurazione schede di rete e indirizzi delle macchine virtuali:

- kali linux, scheda di rete interna connessa a “intnet”. Per la dimostrazione abbiamo impostato un ip statico con indirizzo 192.168.110.100

- Pfsense1 accessibile da 192.168.110.1 con 3 schede di rete:
- “scheda con bridge” accesso a internet
- rete interna “intnet”
- rete interna “DMZ-FW_Link"

- Pfsense2 situata al indirizzo 192.168.100.9 e’ in una rete di interconnessione che reinderizza solo il traffico legittimo alla rete isolata“DMZ-Web_LAN"
- 1 Metasploitable2 con una scheda di rete interna “DMZ-Web_LAN” con indirizzo statico 192.168.100.

NOTA: per una maggior comprensione d’ora in avanti in questa documentazione la macchina virtuale:
- Pfsense sarà chiamata Pfsense Perimetrale
- Pfsense2 sarà chiamata PFsense WebServer"

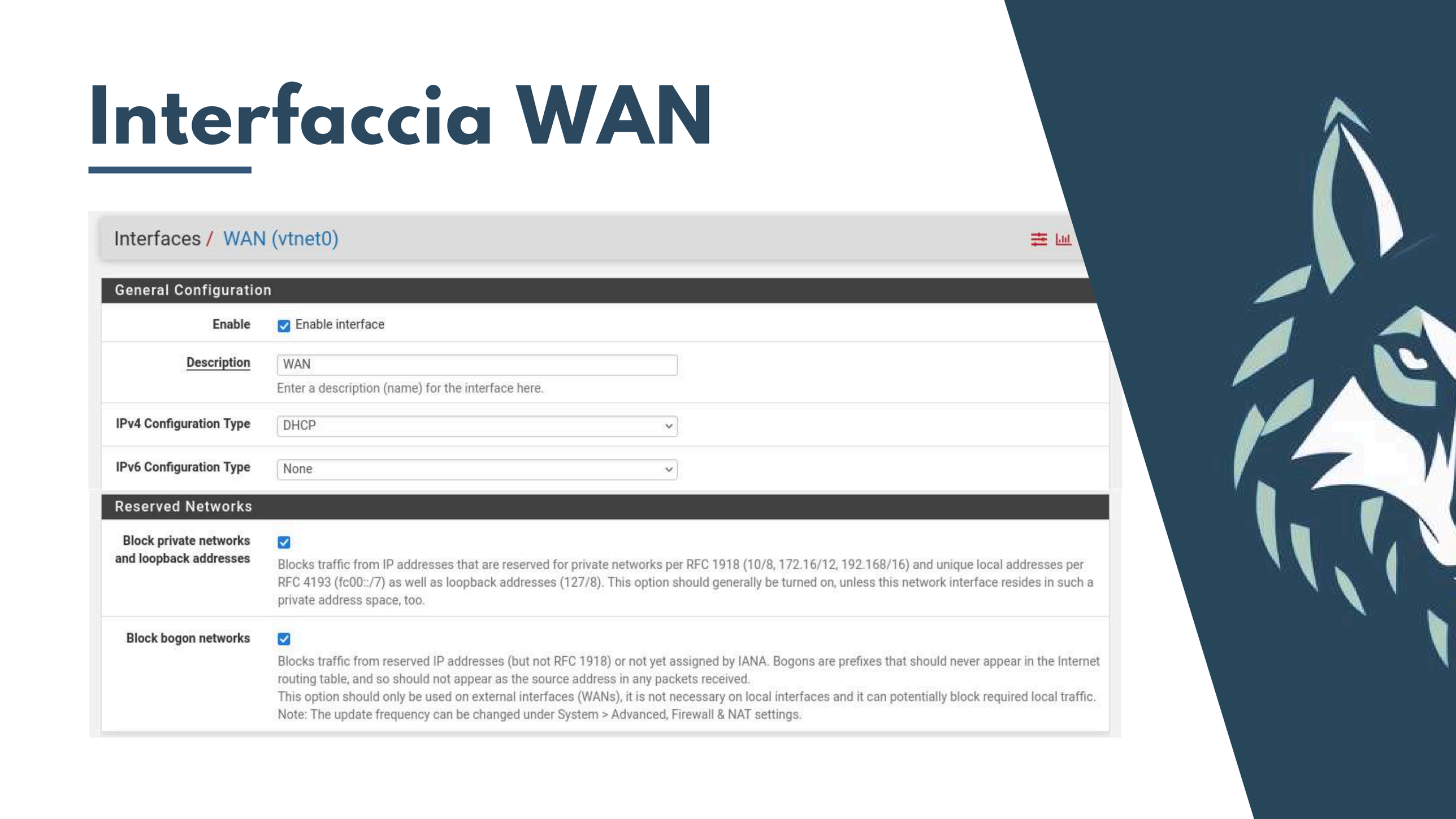
Protocollowolf

Configurazione Pfsense Perimetrale

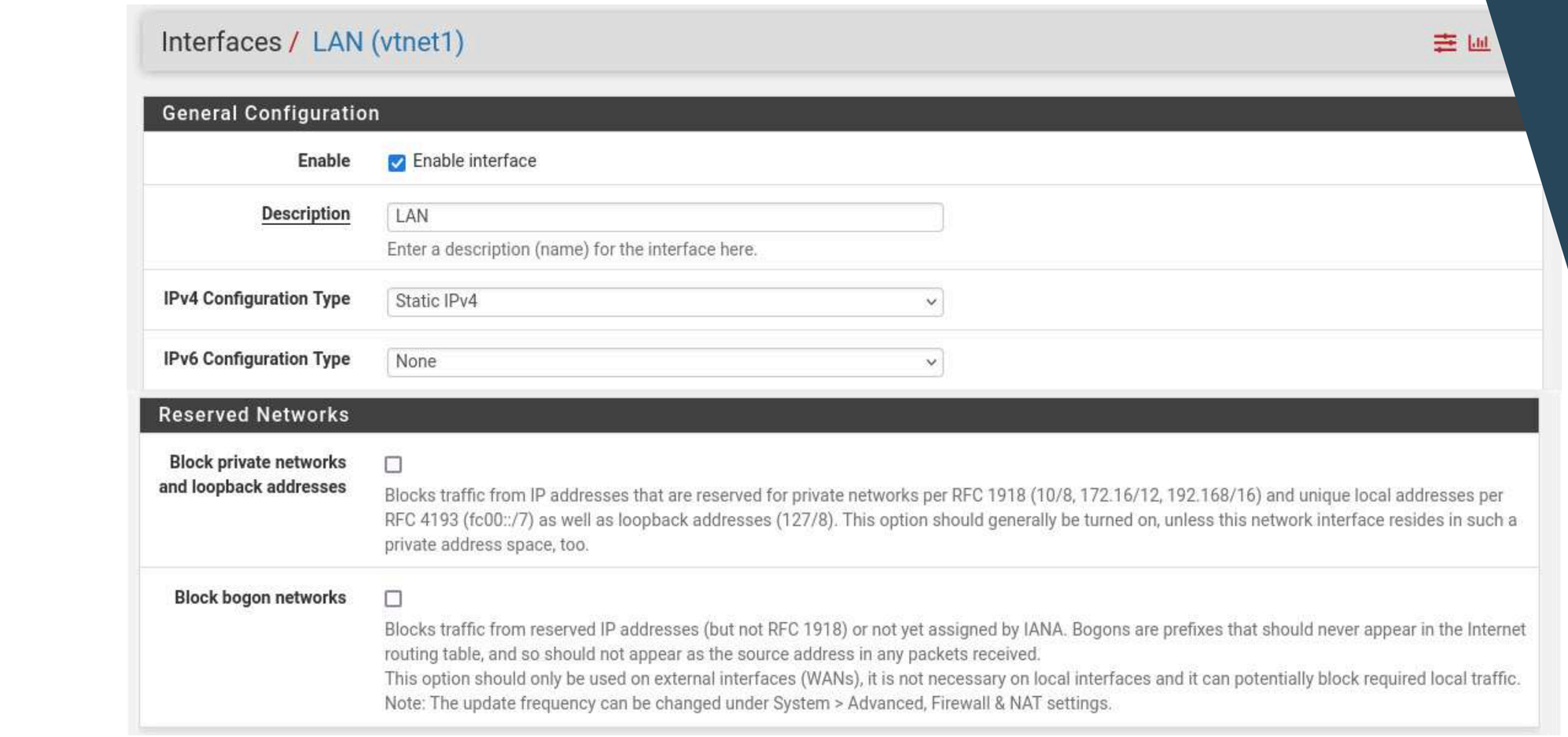
Configurazione Interfacce di Rete

Le tre interfacce che vediamo in foto si occupano di gestire il traffico delle 3 schede di rete sul nostro firewall perimetrale (Pfsense).

Interfaccia WAN



Interfaccia LAN



Interfaccia DMZ

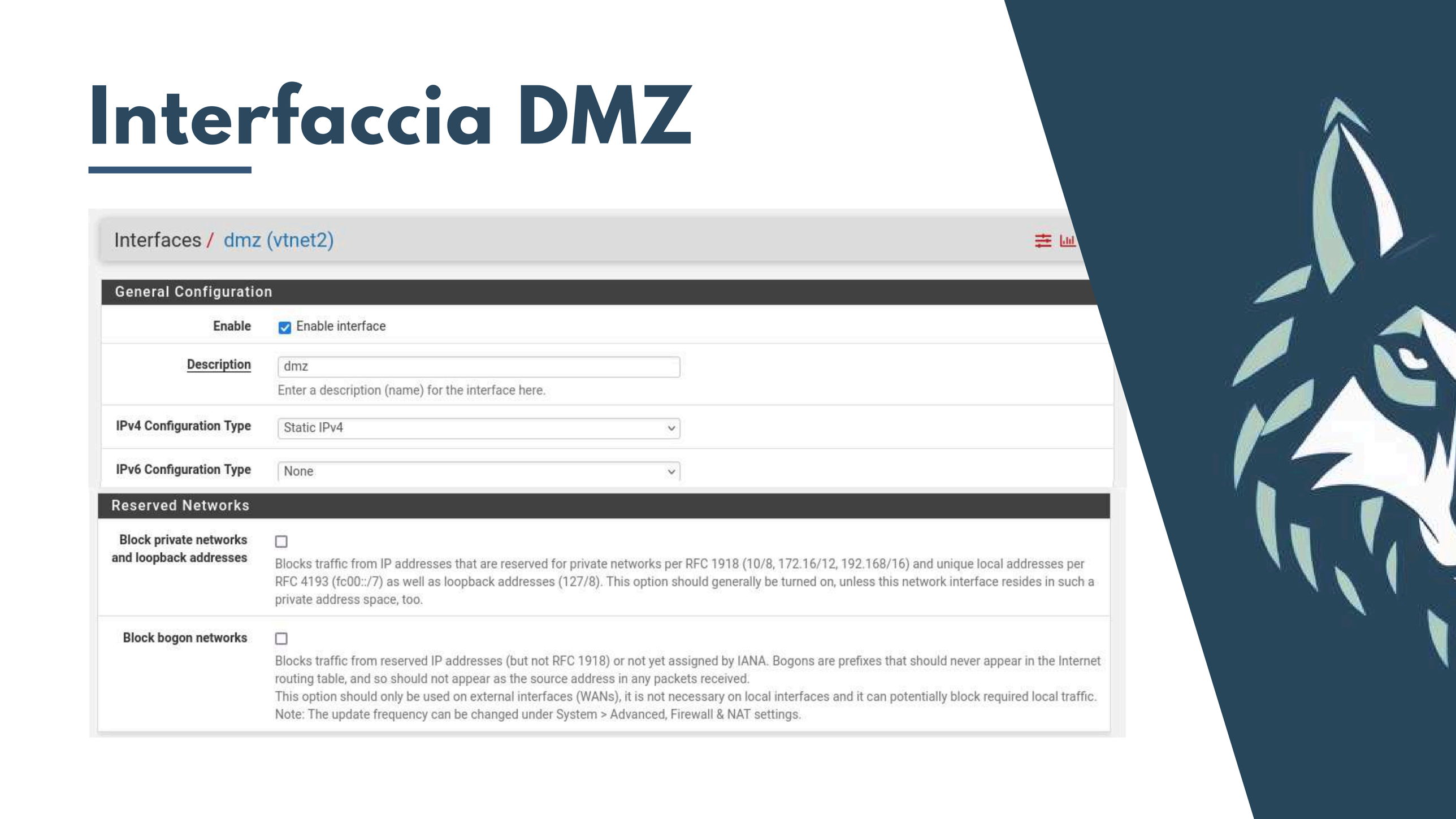
Interfaces / dmz (vtnet2)

General Configuration

Enable	☑ Enable interface
Description	dmz Enter a description (name) for the interface here.
IPv4 Configuration Type	Static IPv4
IPv6 Configuration Type	None

Reserved Networks

Block private networks and loopback addresses	☐ Blocks traffic from IP addresses that are reserved for private networks per RFC 1918 (10/8, 172.16/12, 192.168/16) and unique local addresses per RFC 4193 (fc00::/7) as well as loopback addresses (127/8). This option should generally be turned on, unless this network interface resides in such a private address space, too.
Block bogon networks	☐ Blocks traffic from reserved IP addresses (but not RFC 1918) or not yet assigned by IANA. Bogons are prefixes that should never appear in the Internet routing table, and so should not appear as the source address in any packets received. This option should only be used on external interfaces (WANs), it is not necessary on local interfaces and it can potentially block required local traffic. Note: The update frequency can be changed under System > Advanced, Firewall & NAT settings.



Firewall Rules - LAN

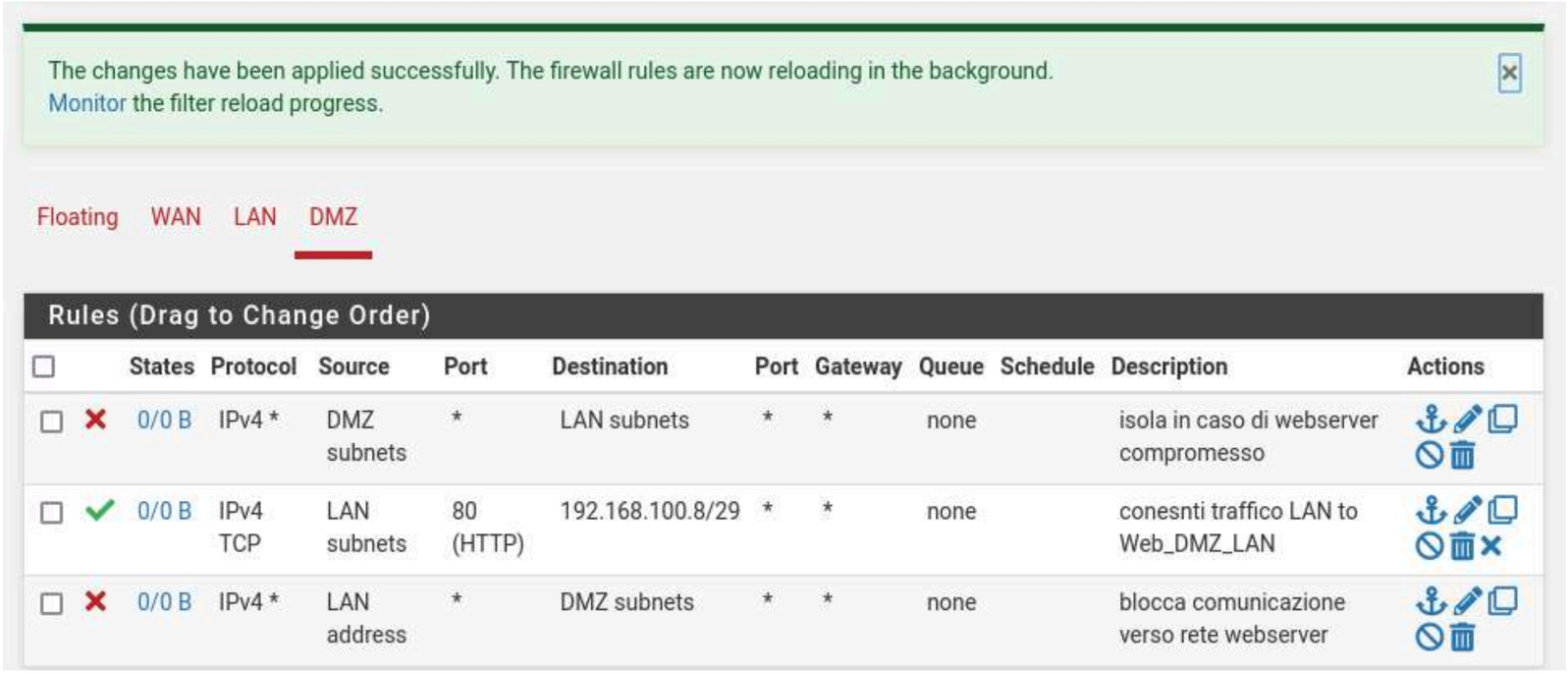
1. **ANTILOCKOUT** – regola predefinita che ci consente di non rimane tagliati fuori per poter usare la GUI del
firewall per le configurazioni
2. Regola che **legittima il traffico** in questo segmento di rete sul protocollo TCP tramite la porta 80 dall'interfaccia LAN verso la metasploitable2
3. Regola che **blocca il resto del traffico** dal interfaccia LAN verso interfaccia DMZ
4. Regola che (spenta per la dimostrazione) **garantisce il traffico verso il NAS**
5. Regola che legittima il resto del traffico dal interfaccia LAN

Rules (Drag to Change Order)

	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description
✓	5/860 KiB	*	*	*	LAN Address	80	*	*		Anti-Lockout Rule
☐ ✓	0/11 KiB	IPv4 TCP	LAN subnets	*	192.168.100.10	80 (HTTP)	*	none		consenti http da lan a dmz
☐ ✗	0/0 B	IPv4 *	LAN subnets	*	DMZ subnets	*	*	none		blocco LAN a DMZ
☐ ✓	0/0 B	IPv4 *	LAN subnets	*	192.168.10.2	*	*	none		accesso al NAT (inserireIP)
☐ ✓	9/29.50 MiB	IPv4 *	LAN subnets	*	*	*	*	none		Default allow LAN to any rule
☐ ✓	0/0 B	IPv6 *	LAN subnets	*	*	*	*	none		Default allow LAN IPv6 to any rule

Firewall Rules - DMZ

1. Regola che **blocca il traffico** dalla DMZ alla LAN isolando il web server in caso fosse compromesso
2. Regola che **legittima il traffico** dalla LAN alla rete di interconnessione Web DMZ LAN
3. Regola che **blocca il traffico** dalla LAN alla DMZ



The changes have been applied successfully. The firewall rules are now reloading in the background.
Monitor the filter reload progress.

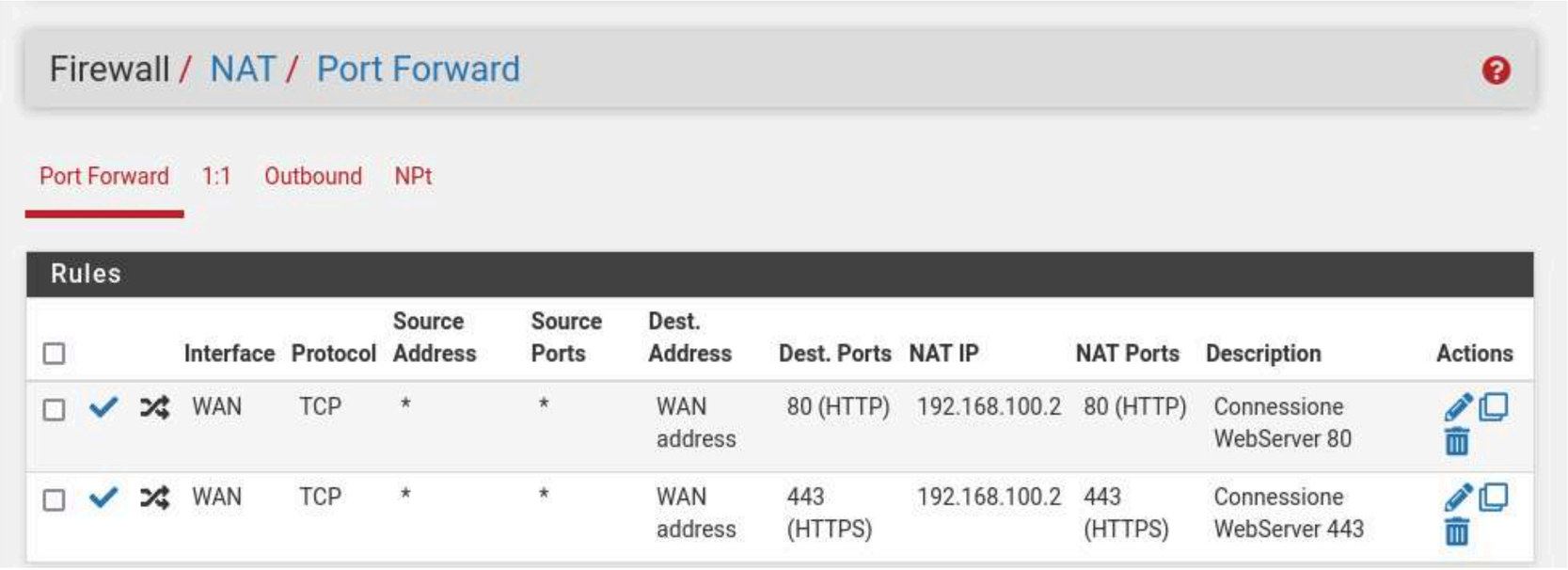
Floating WAN LAN DMZ

Rules (Drag to Change Order)

	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
✗	0/0 B	IPv4 *	DMZ subnets	*	LAN subnets	*	*	none		isola in caso di webserver compromesso	
✓	0/0 B	IPv4 TCP	LAN subnets	80 (HTTP)	192.168.100.8/29	*	*	none		conesnti traffico LAN to Web_DMZ_LAN	
✗	0/0 B	IPv4 *	LAN address	*	DMZ subnets	*	*	none		blocca comunicazione verso rete webserver	

NAT e Port Forwarding

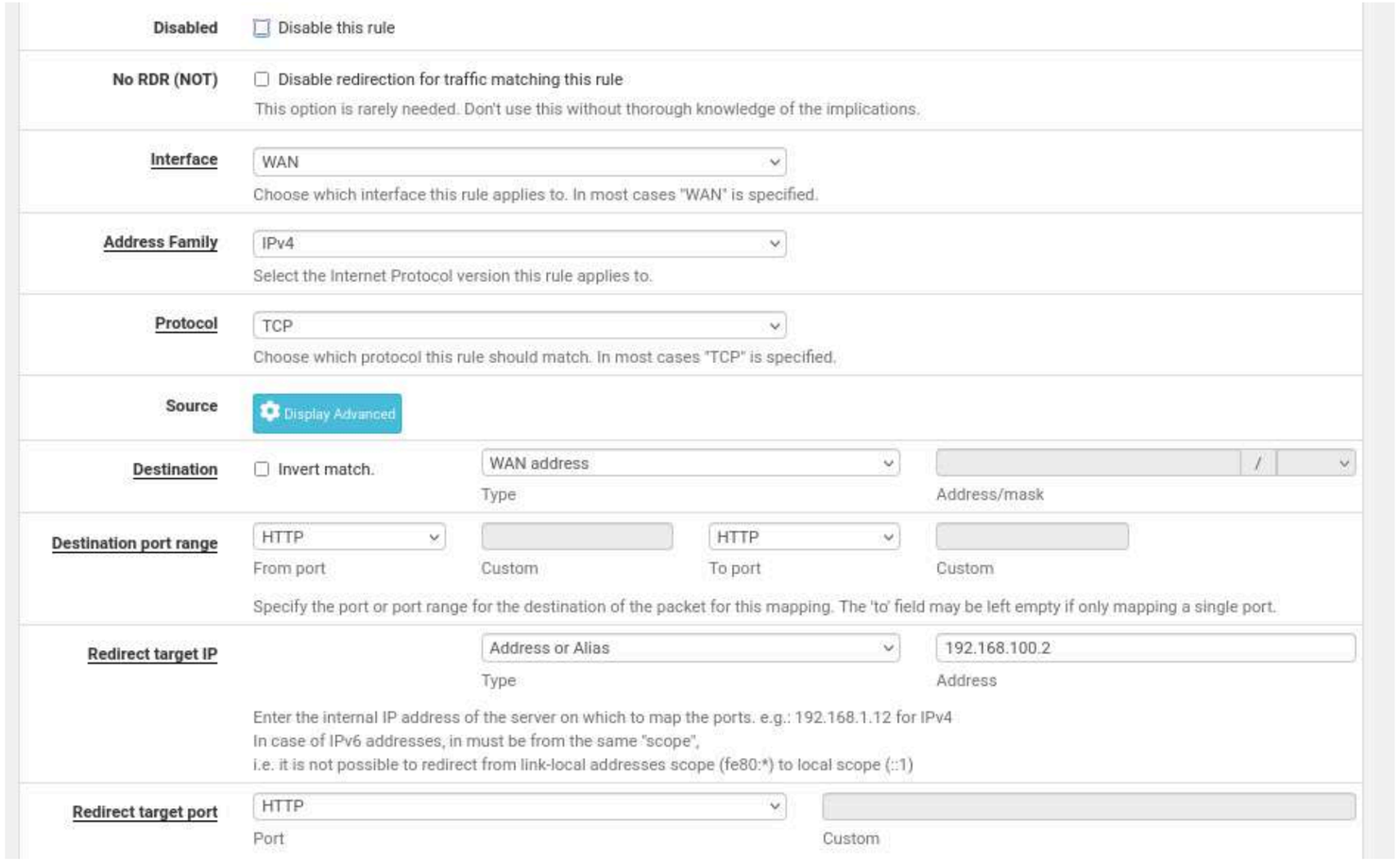
Per riuscire a connettere il dipendendte al webserver solo attraverso internet sfrutteremo il **NAT** che si occupa di tradurre gli indirizzi da pubblico a privato reindirizzando correttamente il traffico attravero la struttura di reti create



	Interface	Protocol	Source Address	Source Ports	Dest. Address	Dest. Ports	NAT IP	NAT Ports	Description	Actions
☐ ✓ ⇄	WAN	TCP	*	*	WAN address	80 (HTTP)	192.168.100.2	80 (HTTP)	Connessione WebServer 80	
☐ ✓ ⇄	WAN	TCP	*	*	WAN address	443 (HTTPS)	192.168.100.2	443 (HTTPS)	Connessione WebServer 443	

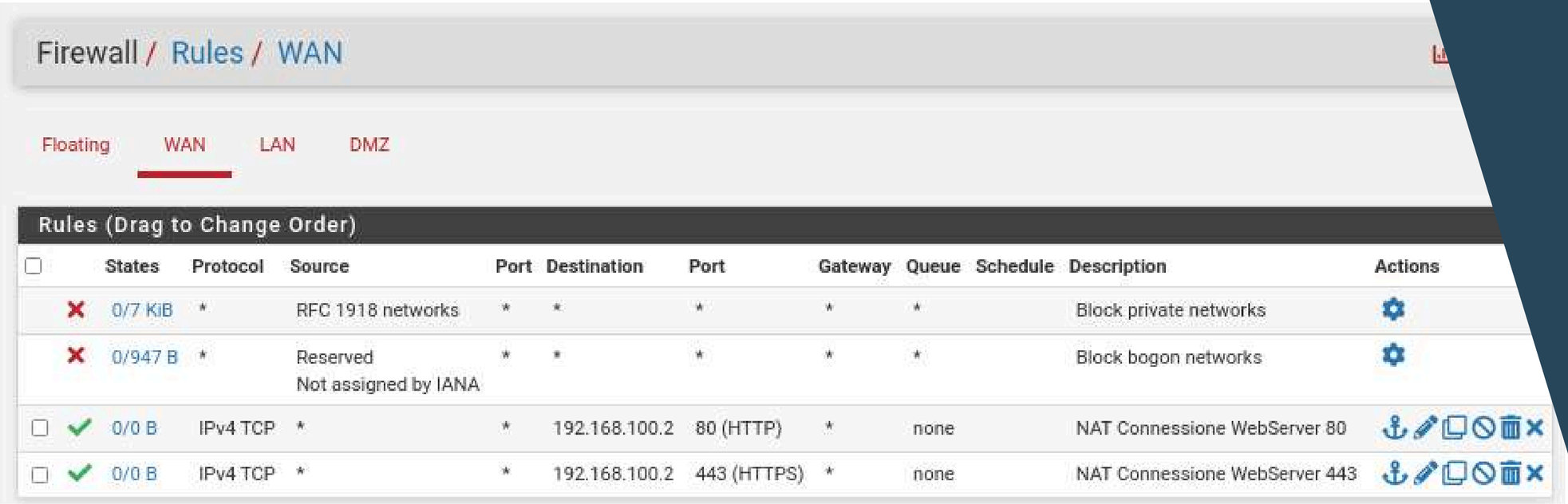
Regola NAT

Dettagli della configurazione delle regole sul NAT, per far passare la comunicazione IPv4 dell'interfaccia WAN tramite il protocollo TCP alle porte di destinazione HTTP"



Firewall Rules - WAN

Una volta configurato il NAT, possiamo osservare nell'interfaccia WAN la comparsa automatica delle **regole 3 e 4** che garantiscono il corretto funzionamento logico del instradamento

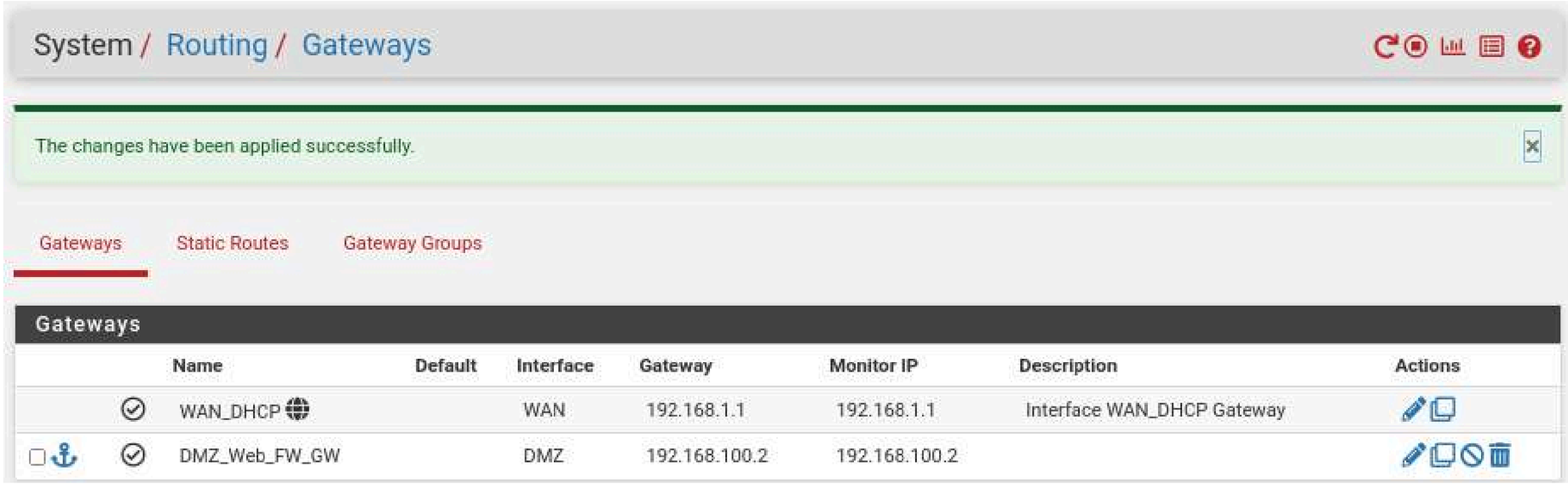


Le regole **1 e 2** sono precauzioni inserite di default per rispettare la sicurezza.

Routing

Per garantire il corretto reindirizzamento del traffico in questa struttura abbiamo bisogno nel **routing** di impostare i gateway

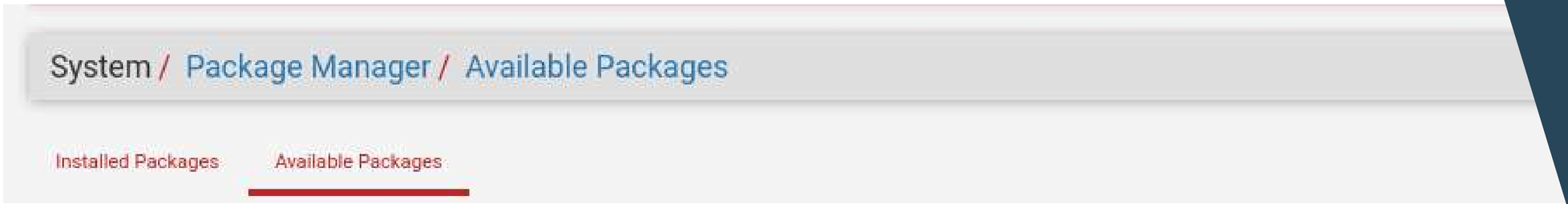
1. La **prima configurazione** imposta il gateway per interfaccia WAN dove arriva l'acceso a internet
2. La **seconda** e' un elemento importante del reindirizzamento verso la rete di interconnesione



Installazione Suricata

Installazione e configurazioen sensori IPS su PFsense Perimeter: servizio **Suricata**
Selezioniamo “**Package Manager**” dal menu’ “System”
Cerchiamo tra i “packages” disponibili il servizio “**Suricata**”

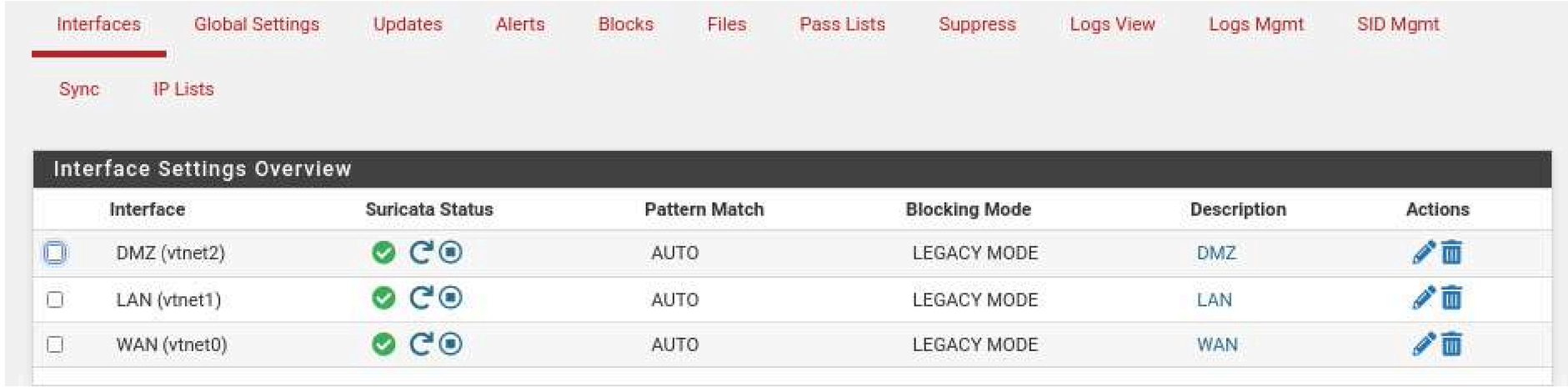
Installiamolo e configuriamolo



Configurazione IPS

Applichiamo un sensore per ogni interfaccia, uno per la **LAN**, uno per la **WAN** e uno per la **DMZ**, durante l'installazione saranno poi configurati parametri specifici, per adesso lasciamo tutto di default ricordandoci di entrare ad editare ogni sensore con queste impostazioni:
- **Interface**: selezionare interfaccia corrispondente
- **Block Offender**: ON
- **Run Mode**: Workers

NOTA BENE: selezionando "AutoFP" otterremmo un IDS, selezionando "Workers" invece avremmo un sensore proattivo IPS"



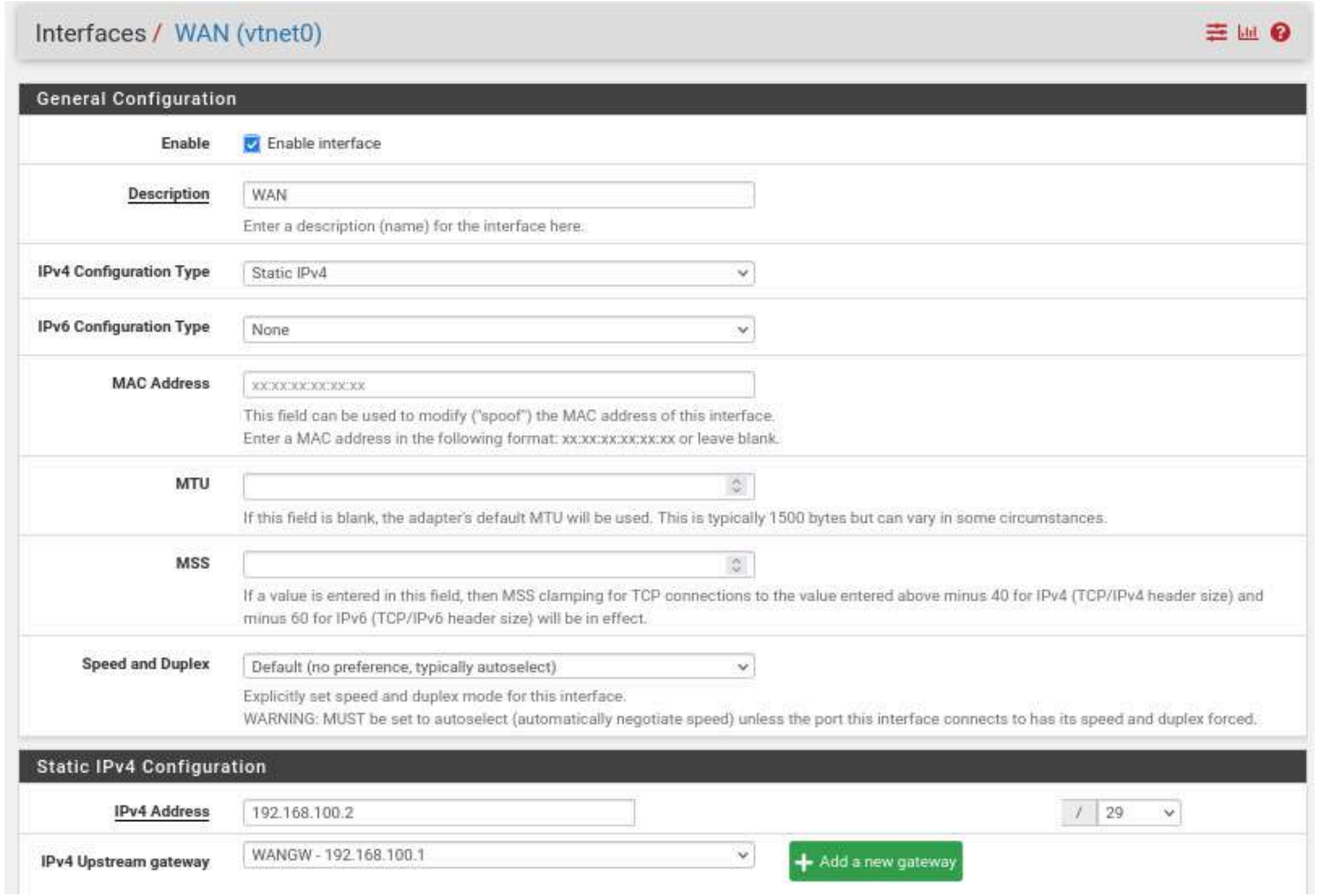
Protocollowolf

Configurazione Pfsense WebServer

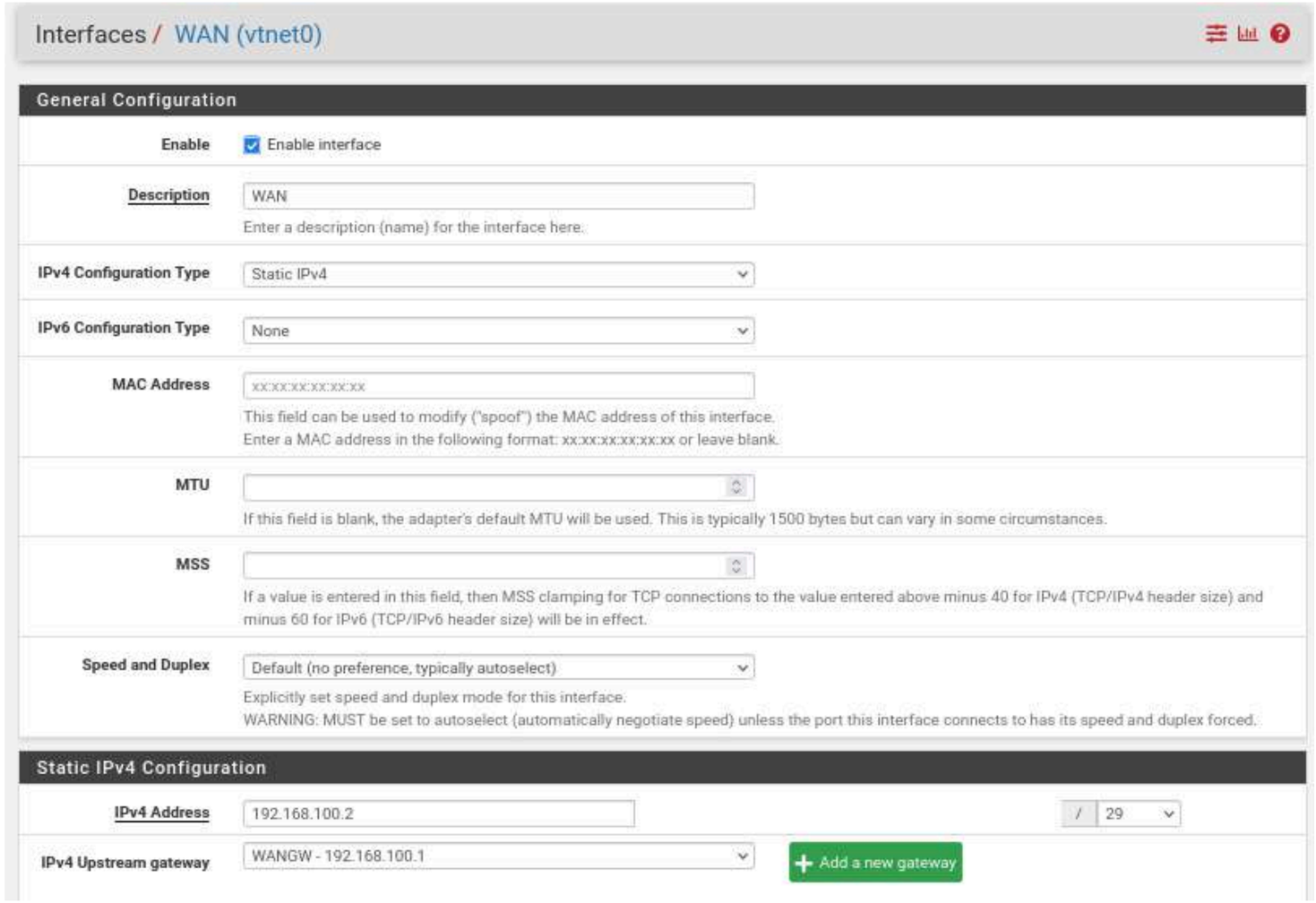
Interfaccia di rete

NOTA: per la prima installazione e configurazione della Pfsense WebServer abbiamo bisogno di connetterci ad essa, creiamo quindi momentaneamente ai fini dell'installazione una rete d'accesso che poi sarà bloccata.

Interfaccia WAN

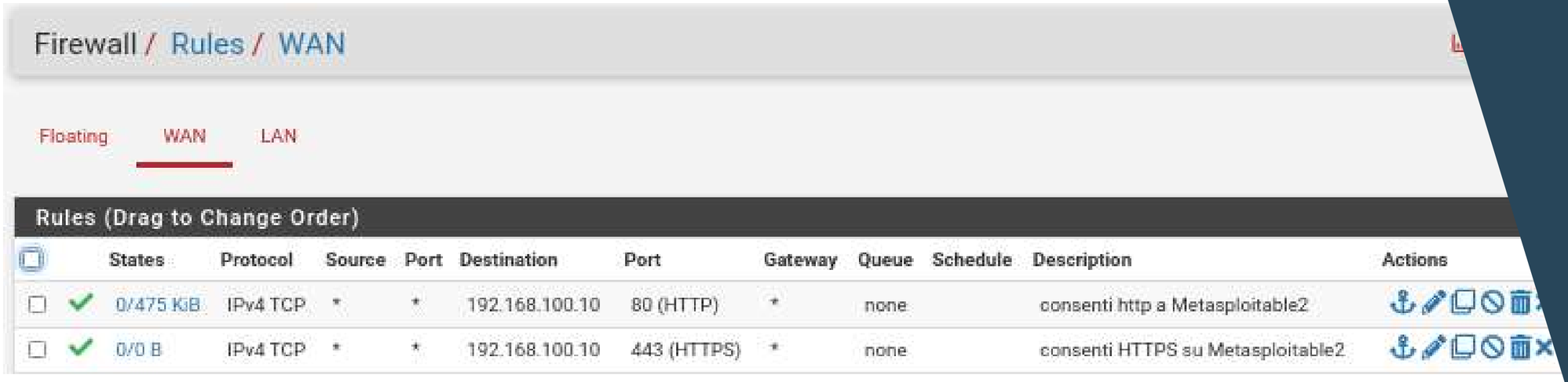


Interfaccia LAN



Firewall Rules - WAN

1. Regola che **legittima il traffico** verso il web server sulla porta 80 tramite il protocollo TCP su IPv4
2. Regola che **legittima il traffico** verso il web server sulla porta 443 tramite il protocollo TCP su IPv4



Firewall Rules - LAN

1. Regola **Anti-lockout**
2. Regola che **legittima il traffico** in uscita dal web server per poter svolgere i suoi compiti
3. Regola che presente solo in fase di installazione, poi rimossa
4. Regola che **legittima il traffico** in entrata dalla “intnet”
5. Regola che **blocca** in maniera generale il resto del traffico (implicit Deny)

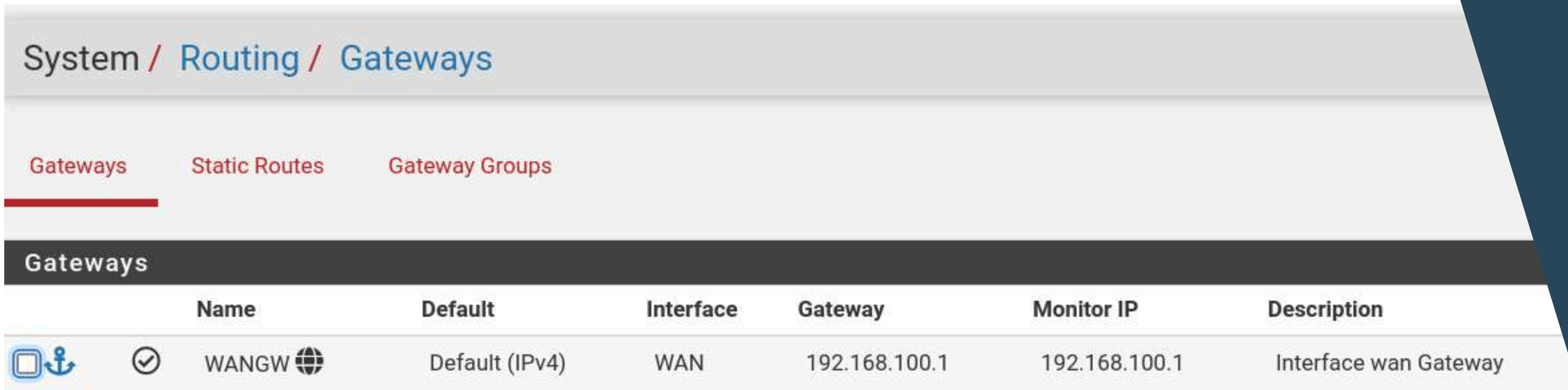
Floating WAN LAN

Rules (Drag to Change Order)

	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
	0/332 KiB	*	*	*	LAN Address	80	*	*		Anti-Lockout Rule	
	0/515 B	IPv4 *	192.168.100.10	*	*	*	*	none		uscita traffico WebServer	
	0/0 B	IPv4 TCP	192.168.100.11	*	192.168.100.10	80 (HTTP)	*	none		SOLO INSTALLAZIONE (kaliLINKtoMETA)	
	0/0 B	IPv4 TCP	192.168.110.0/24	*	192.168.100.10	80 (HTTP)	*	none		consenti kaliINT to Metasploitable2 HTTP	
	0/2 KiB	IPv4 *	*	*	*	*	*	none		blocco altro traffico LAN	

Routing

Come prima, per garantire il corretto **reindirizzamento del traffico** in questa struttura abbiamo bisogno nel **routing** di impostare i **gateway** per l'interfaccia **WAN**



Configurazione Sensori

Dopo aver installato il servizio Suricata come in Pfsense Perimetrale, sulla Pfsense WebServer abbiamo posizionato **due sensori IPS** per controllare le interfacce di rete.

La configurazione è piena di parametri utili. Lasciamo attive le opzioni di default e controlliamo le seguenti:

- **Interface**: WAN
- **Block Offenders**: abilitato
- **IPS Mode**: Legacy
- **Run Mode**: Workers (modalità sensore pro attivo)

Configurazione sensore LAN:

Ci comportiamo come il sensore WAN clonandolo e poi cambiando l’interfaccia in LAN

Configurazione Sensori

Considerazioni Finali

Abbiamo avuto successo nel progettare e configurare una rete con più segmenti isolati (LAN interna, rete di interconnessione per accedere ad una altra rete isolata, il tutto è supportato da un doppio firewall) per migliorare la postura di sicurezza complessiva.

• **Protezione del Web Server**: Il posizionamento del web server all'interno di una rete isolata, protetta da un secondo firewall dedicato, rappresenta un significativo miglioramento della sicurezza. Il traffico in entrata verso il web server è ora sottoposto a un doppio screening, riducendo drasticamente la superficie di attacco escludendo attaccanti nel caso il web server fosse compromesso.
• **Gestione Granulare del Traffico:** Attraverso l'attenta configurazione delle regole su entrambi i Firewall, abbiamo ottenuto un controllo granulare del traffico:
• **Accesso Esterno (Internet) al Web Server**: Permesso solo per le porte HTTP/HTTPS attraverso NAT e regole di passaggio precise su entrambi i firewall.
• **Accesso Interno (Intranet) al Web Server:** Consentito selettivamente, come dimostrato dalla capacità di Kali
Linux sulla rete “intnet” di raggiungere il web server via HTTP, mentre altro traffico (es. ping ecc.) è bloccato.

Considerazioni Finali

• **Traffico In uscita del Web Server:** Il web server può raggiungere Internet per le sue funzionalità essenziali (aggiornamenti, DNS, ecc.), ma le sue interazioni sono mediate dal firewall dedicato.
• **Configurazione del Routing Multi-Hop:** Assicurare che il traffico potesse fluire correttamente dalla WAN esterna fino al web server, passando attraverso due firewall e una rotta statica su pfSense Perimeter, ha richiesto una comprensione approfondita del routing e del network address translation (NAT).
• **Logica delle Regole Firewall**: La definizione delle regole firewall, soprattutto su entrambi i Firewall, ha richiesto attenzione ai dettagli e continui test iterativi per garantire sicurezza e funzionalità.

NOTA BENE: Ai fini del test dimostrativi abbiamo consentito solo il traffico HTTP verso il web server e abbiamo voluto utilizzare solo indirizzi di tipo IPv4 per facilitare la comprensione e la leggibilità del materiale tecnico. Durante la reale installazione della rete saranno prese tutte le misure di sicurezza moderne per garantire la massima affidabilità.

Protocollowolf

Python Tools

Verbi HTTP

Analisi verbi HTTP

```python
import requests
from bs4 import BeautifulSoup

while True:                    # Menù per richiesta verbo HTTP da verificare
    print("Verifica dei Verbi HTTP, seleziona la richiesta HTTP da inviare:")
    print("1. Invia una richiesta HTTP di tipo GET")
    print("2. Invia una richiesta HTTP di tipo POST")
    print("3. Invia una richiesta HTTP di tipo PUT")
    print("4. Invia una richiesta HTTP di tipo DELETE")
    print("5. Invia una richiesta HTTP di tipo OPTIONS")
    print("6. Esci") # Opzione per terminare il programma

    # Qui si chiede all' utente di scegliere una delle opzioni disponibili

    try:
        scelta = int(input("\nLa tua scelta: "))
    except ValueError:
        print("\nScelta non valida. Inserisci un numero intero.")
        continue    # Ritorna all'inizio del ciclo

    if scelta == 6:
        print("\nUscita dal programma.")
        break
```

· Questa sezione gestisce il flusso principale del programma attraverso un ciclo infinito **(while True)** che continua a mostrare un menu all'utente finché non sceglie di uscire.

· Viene stampato un menu che elenca i diversi verbi HTTP che lo script è in grado di testare **(GET, POST, PUT, DELETE, OPTIONS)** e un'opzione per uscire.

· L'input dell'utente viene catturato e convertito in un numero intero. Un blocco **try-except ValueError** è utilizzato per gestire il caso in cui l'utente inserisca un valore non numerico, prevenendo crash del programma.

Analisi verbi HTTP

```python
if scelta in [1, 2, 3, 4, 5]:       # Si assicura che la scelta sia valida
    indirizzo = input("\nInserisci l'URL completo del server (es. http://example.com/): ") # Ho aggiunto un esempio
    print(f"\nConferma URL: {indirizzo}")

    try:
        response = None    # Inizializza la variabile response

        # Elaborazione degli input assegnati dall' utente

        if scelta == 1:
            print("\nHai selezionato il Verbo HTTP GET")       # Una richiesta GET recupera dati di richiesta
            response = requests.get(indirizzo)
        elif scelta == 2:
            print("\nHai selezionato il Verbo HTTP POST")
            response = requests.post(indirizzo, data={})     # Richiesta POST invia dati al server per creare una nuova risorsa
                                                             # data={} invia un corpo di richiesta vuoto per verificare se il verbo viene accettato dal server
        elif scelta == 3:
            print("\nHai selezionato il Verbo HTTP PUT")
            response = requests.put(indirizzo, data={})      # Richiesta PUT per aggiornare una risorsa
                                                             # data{} corpo vuoto per verifica
        elif scelta == 4:
            print("\nHai selezionato il Verbo HTTP DELETE")    # Richiesta DELETE per eliminare una risorsa
            response = requests.delete(indirizzo)

        elif scelta == 5:
            print("\nHai selezionato il verbo HTTP OPTIONS")
            response = requests.options(indirizzo)           # Richiesta OPTIONS per interrogare il server
```

Analisi verbi HTTP

```python
if response: # Questo blocco viene eseguito solo se la richiesta HTTP ha generato una risposta.
    print(f"\n--- Dettagli della Risposta ---")
    print(f"Codice di stato: {response.status_code}")      # Stampa il codice di stato HTTP
    print(f"URL della risposta: {response.url}")           # Stampa l'URL finale della risposta
    print(f"Tempo di risposta: {response.elapsed.total_seconds():.2f} secondi")    # Stampa il tempo

    # L'header 'Allow' è rilevante per tutti i verbi, specialmente OPTIONS e 405.
    responseAllow = response.headers.get("Allow")

    if responseAllow:
        print(f"Header 'Allow' ricevuto: {responseAllow} (Indica i metodi supportati).") # Se questo
```

· Questo blocco viene eseguito solo se è stata ricevuta una risposta valida dal server (***if response:***).
· Vengono stampati i dettagli fondamentali della risposta: il codice di stato HTTP (es. 200, 404, 405), l'URL finale della risposta e il tempo impiegato per riceverla.
· **Gestione dell'Header 'Allow'**: Viene utilizzato il metodo ***response.headers.get("Allow")*** per recuperare il valore dell'header Allow. Questo metodo è sicuro perché restituisce ***None*** se l'header non è presente, evitando errori. Se l'header Allow è presente, il suo valore viene stampato, indicando i metodi HTTP supportati dal server per la risorsa richiesta. Questo è particolarmente rilevante per le richieste OPTIONS e i codici 405.

Analisi verbi HTTP

```
# Questa sezione interpreta i codici più comuni per determinare se il verbo è abilitato
if 200 <= response.status_code < 300:
    print(f"\nRisultato: Il verbo HTTP selezionato è probabilmente ABILITATO (Codice {response.status_code}).")     # Se il codice è tra 200 e 299 la richiesta è stata ricevuta

    # Adesso il programma si occuperà di parsificare il contenuto, cioè estrapolerà e analizzerà il contenuto di HTML

    content_type = response.headers.get('Content-Type', '').lower()          # Così non si creano problemi per il case-sensitivity, dato che il titolo può essere scritto in var
    if 'text/html' in content_type or 'application/xml' in content_type or 'text/xml' in content_type:      # Controllo del conten_type perchè BeautifulSoup non è in grado di pa
        print("\nAnalisi del contenuto HTML: ")       # Informa l'utente che il programma ha iniziato l'analisi
        # Inizializza BeautifulSoup con il testo della risposta
        soup = BeautifulSoup(response.text, 'html.parser')        # Inizializzazione di BeautifulSoap e utilizzo del parser che utilizzerà per l'analisi per interpretare

        # Ottenere il titolo della pagina con BeautifulSoap
        title = soup.find('title')
        if title:
            print(f"\n  Titolo della pagina: {title.get_text().strip()}")      # Se è stato trovato il titolo stampa il testo del titolo e rimuove gli spazi bianchi extra
            if "phpmyadmin" in title.get_text().lower():
                print("\n  -> Il titolo della pagina suggerisce una risposta attesa per phpMyAdmin.")
            else:
                print(f"\n  -> Il titolo della pagina ({title.get_text().strip()}) non è quello atteso, potrebbe essere una pagina di errore mascherata o un reindirizzamento.")

        # Trovare tutti i link della pagina: cerca i tag <a>S
        links = soup.find_all('a')

        if links:
            print(f"  Trovati {len(links)} link. Primi 5:\n")
            # Itera fino ad un massimo di 5 link
            for i, link in enumerate(links[:5]):                # Slicing della lista di link con numerazione
                href = link.get('href')                         # Estrae il valore di href (URL del link)
                text = link.get_text().strip()                  # Estrae il testo visibile del link
                print(f"    - [{text}]({href})")                # Link in formato leggibile

        # Controllo per messaggi di successo o errore nel contenuto della pagina
        success_message_found = False
        error_message_found = False


         # Tiene traccia se sono stati trovati messaggi di successo o errore
        success_message_found = False
        error_message_found = False
```

Analisi verbi HTTP

```
# Questa sezione interpreta i codici più comuni per determinare se il verbo è abilitato
if 200 <= response.status_code < 300:
    print(f"\nRisultato: Il verbo HTTP selezionato è probabilmente ABILITATO (Codice {response.status_code}).")
```

Codici 2xx (Successo): Se il codice di stato è tra 200 e 299, indica un successo. In questo caso, viene stampato che il verbo è "probabilmente ABILITATO".

```
elif response.status_code == 405:                # Il codice 405 "Method Not Allowed" indica esplicitamente
    print(f"\nRisultato: Il verbo HTTP selezionato NON è ABILITATO (Codice 405 - Method Not Allowed).")
    # L'header 'Allow' è già stampato all'inizio dei dettagli della risposta se presente.
elif response.status_code == 404:                # Il codice 404 "Not Found" indica che la risorsa richiesta
    print(f"\nRisultato: Risorsa non trovata (Codice 404). Verifica l'URL o il path.")
else:                                            # Per qualsiasi altro codice di stato la response è inattes
    print(f"\nRisultato: Risposta inattesa o errore (Codice {response.status_code}).")
    print("Controlla il codice di stato e il contenuto della response per maggiori dettagli.")
```

Codice 405 (Method Not Allowed): Questo codice indica esplicitamente che il server ha compreso la richiesta, ma non permette il metodo HTTP richiesto per quella risorsa, confermando che il verbo non è abilitato.
Codice 404 (Not Found): Indica che la risorsa richiesta non esiste sull'URL specificato. Il verbo potrebbe essere abilitato, ma la destinazione non è valida.

Analisi verbi HTTP

```
content_type = response.headers.get('Content-Type', '').lower()          # Così non si creano problemi per il ca
if 'text/html' in content_type or 'application/xml' in content_type or 'text/xml' in content_type:      # Control
    print("\nAnalisi del contenuto HTML: ")       # Informa l'utente che il programma ha iniziato l'analisi
    # Inizializza BeautifulSoup con il testo della risposta
    soup = BeautifulSoup(response.text, 'html.parser')        # Inizializzazione di BeautifulSoap e utilizzo del
```

Viene inizialmente controllato l'**header Content-Type** della risposta per determinare se il contenuto è HTML, XML o testo XML.

Se il contenuto è compatibile, BeautifulSoup viene inizializzato con il testo della risposta, creando un oggetto **soup** che permette la navigazione e l'estrazione di elementi.

```
else:                                   # Se il content-type non è HTML/XML bs4 non sarà in g
    print("\nIl contenuto della risposta non è HTML/XML; impossibile analizzare.")
    print(f"Contenuto (primi 500 caratteri):\n{response.text[:500]}...")          # Stampa la
```

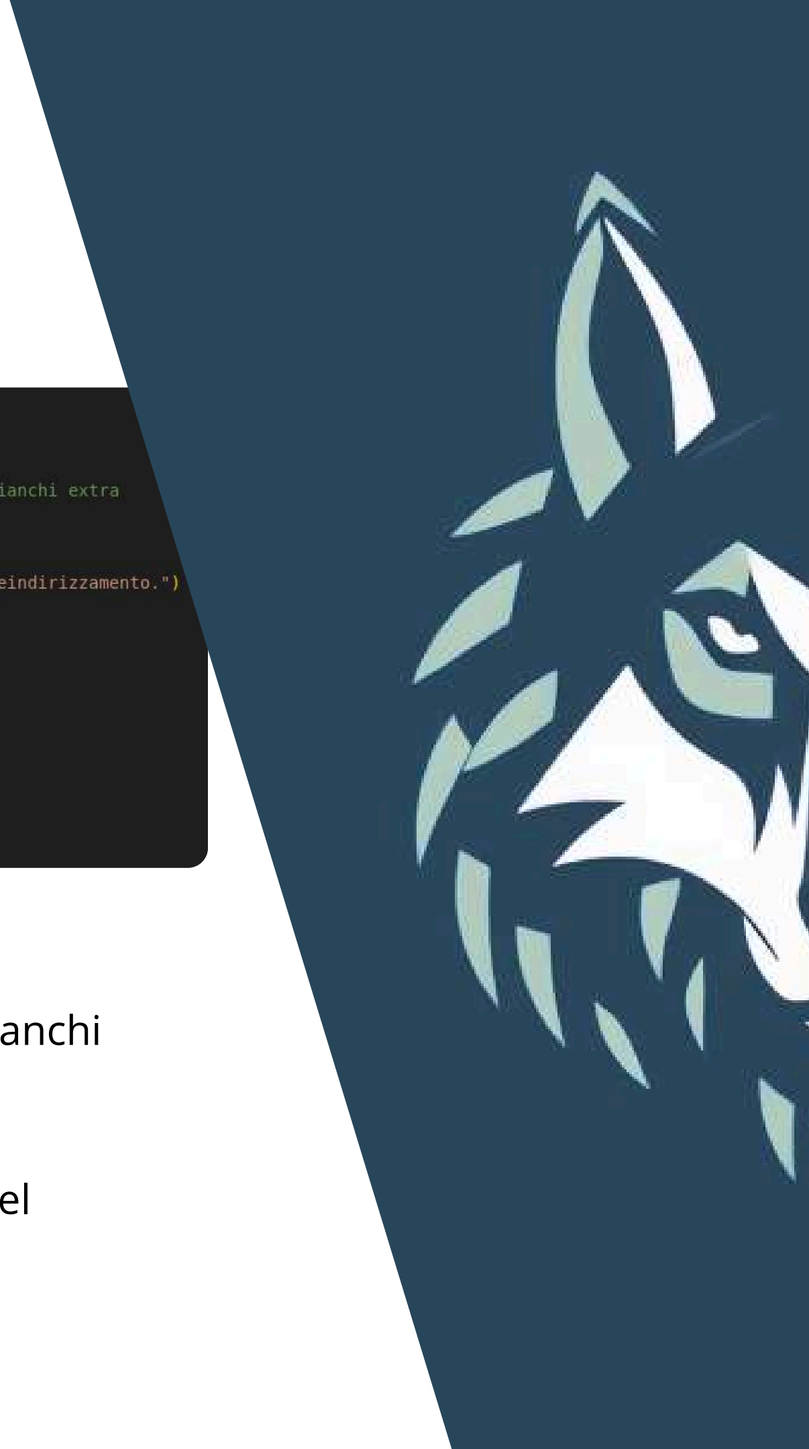
Nel caso in cui il **Content-Type** non sia HTML/XML, il contenuto non viene analizzato in dettaglio, ma vengono stampati i primi 500 caratteri per permettere un'ispezione manuale da parte dell'utente.

Analisi verbi HTTP

```
# Ottenere il titolo della pagina con BeautifulSoap
title = soup.find('title')
if title:
    print(f"  Titolo della pagina: {title.get_text().strip()}")      # Se è stato trovato il titolo stampa il testo del titolo e rimuove gli spazi bianchi extra
    if "phpmyadmin" in title.get_text().lower():
        print("  -> Il titolo della pagina suggerisce una risposta attesa per phpMyAdmin.")
    else:
        print(f"  -> Il titolo della pagina ({title.get_text().strip()}) non è quello atteso, potrebbe essere una pagina di errore mascherata o un reindirizzamento.")

# Trovare tutti i link della pagina: cerca i tag <a>
links = soup.find_all('a')

if links:
    print(f"  Trovati {len(links)} link. Primi 5:")
    # Itera fino ad un massimo di 5 link
    for i, link in enumerate(links[:5]):                # Slicing della lista di link con numerazione
        href = link.get('href')                         # Estrae il valore di href (URL del link)
        text = link.get_text().strip()                  # Estrae il testo visibile del link
        print(f"    - [{text}]({href})")                # Link in formato leggibile
```



La sezione **if title** si occupa di estrarre e interpretare il titolo della pagina web.
Se il titolo è presente, viene stampato il testo estratto, rimuovendo eventuali spazi bianchi superflui.

La sezione **if links** ha il compito di identificare e presentare i link (tag <a>) presenti nel documento HTML.

Analisi verbi HTTP

```
# Controllo per messaggi di successo o errore nel contenuto della pagina
success_message_found = False
error_message_found = False


 # Tiene traccia se sono stati trovati messaggi di successo o errore
success_message_found = False
error_message_found = False

# Cerca nel testo della pagina la parola "success" per avere la certezza del funzionamento
if soup.find(string=lambda s: s and "success" in s.lower()):
    success_message_found = True
    print("  -> Trovato messaggio di successo nel contenuto della pagina.")

# Stesso principio per rilevare errori
if soup.find(string=lambda s: s and ("error" in s.lower() or "denied" in s.lower())):
    error_message_found = True
    print("  -> Trovato messaggio di errore/accesso negato nel contenuto della pagina.")

# Indicazione più precisa della riuscita a livello applicativo oltre il codice 200
if success_message_found and not error_message_found:
    print("  -> L'analisi del contenuto HTML suggerisce un'esecuzione funzionale riuscita.")
# Se c'è il codice 200, ma viene indicato un messaggio di errore alla pagina
elif error_message_found:
    print("  -> L'analisi del contenuto HTML suggerisce un fallimento funzionale, nonostante il codice di stato HTTP.")
```

Questa sottosezione analizza il testo del contenuto HTML per individuare indicatori di **successo o fallimento** a livello applicativo, che potrebbero non essere sempre riflessi nel codice di stato HTTP.

Analisi verbi HTTP

· **else per if response:** Questo blocco viene eseguito se, per qualche motivo, la variabile response rimane **None**
· **Gestione degli Errori di Richiesta**: Il blocco **except** cattura diversi tipi di errori che possono verificarsi durante la fase di richiesta HTTP:
- requests.exceptions.ConnectionError: Indica problemi di connettività di rete
- requests.exceptions.MissingSchema: Si verifica se l'URL fornito non include lo schema http:// o https://.
- requests.exceptions.RequestException as e: È una classe base per la maggior parte degli errori della libreria requests, catturando qualsiasi altro problema generico che possa insorgere durante la richiesta e stampandone i dettagli.

```
    # Gestione errori
    except requests.exceptions.ConnectionError:              # Assenza connettività alla rete o server offline
        print("ERRORE: Impossibile connettersi all'URL. Controlla l'indirizzo, la connettività di rete o che il server sia attivo.")
    except requests.exceptions.MissingSchema:                # Assenza di "http://" o "https://"
        print("ERRORE: L'URL deve includere lo schema ('http://' o 'https://').")
    except requests.exceptions.RequestException as e:        # Errore generico. "e" è una variabile che contiene l'oggetto dell'errore e lo stamperà pe
        print(f"ERRORE GENERALE DURANTE LA RICHIESTA: {e}")

else:
    print("Scelta non valida. Per favore, inserisci un numero tra 1 e 6.")
```

Analisi verbi HTTP (Test)

```
Verifica dei Verbi HTTP, seleziona la richiesta HTTP da inviare:
1. Invia una richiesta HTTP di tipo GET
2. Invia una richiesta HTTP di tipo POST
3. Invia una richiesta HTTP di tipo PUT
4. Invia una richiesta HTTP di tipo DELETE
5. Invia una richiesta HTTP di tipo OPTIONS
6. Esci

La tua scelta: 1

Inserisci l'URL completo del server (es. http://example.com/): http://192.168.20.100/phpMyAdmin/

Conferma URL: http://192.168.20.100/phpMyAdmin/

Hai selezionato il Verbo HTTP GET

--- Dettagli della Risposta ---

Codice di stato: 200

URL della risposta: http://192.168.20.100/phpMyAdmin/

Tempo di risposta: 0.08 secondi

Risultato: Il verbo HTTP selezionato è probabilmente ABILITATO (Codice 200).

Analisi del contenuto HTML:

  Titolo della pagina: phpMyAdmin

  -> Il titolo della pagina suggerisce una risposta attesa per phpMyAdmin.
  Trovati 2 link. Primi 5:

    - [](http://www.phpmyadmin.net)
    - [mcrypt](http://php.net/mcrypt)
```

Analisi verbi HTTP (Test)

```
Verifica dei Verbi HTTP, seleziona la richiesta HTTP da inviare:
1. Invia una richiesta HTTP di tipo GET
2. Invia una richiesta HTTP di tipo POST
3. Invia una richiesta HTTP di tipo PUT
4. Invia una richiesta HTTP di tipo DELETE
5. Invia una richiesta HTTP di tipo OPTIONS
6. Esci

La tua scelta: 5

Inserisci l'URL completo del server (es. http://example.com/): http://192.168.20.100/phpMyAdmin/themes/original/img/logo_right.png

Conferma URL: http://192.168.20.100/phpMyAdmin/themes/original/img/logo_right.png

Hai selezionato il verbo HTTP OPTIONS

--- Dettagli della Risposta ---

Codice di stato: 200

URL della risposta: http://192.168.20.100/phpMyAdmin/themes/original/img/logo_right.png

Tempo di risposta: 0.01 secondi

Header 'Allow' ricevuto: GET,HEAD,POST,OPTIONS,TRACE (Indica i metodi supportati).

Risultato: Il verbo HTTP selezionato è probabilmente ABILITATO (Codice 200).

Il contenuto della risposta non è HTML/XML; impossibile analizzare.
Contenuto (primi 500 caratteri):
...
Verifica dei Verbi HTTP, seleziona la richiesta HTTP da inviare:
1. Invia una richiesta HTTP di tipo GET
2. Invia una richiesta HTTP di tipo POST
3. Invia una richiesta HTTP di tipo PUT
4. Invia una richiesta HTTP di tipo DELETE
5. Invia una richiesta HTTP di tipo OPTIONS
6. Esci

La tua scelta:
```

Protocollowolf

Python Tools

Port Scanner

Port Scanner

```python
import socket

ip = input("Inserisci l'indirizzo IP da scansionare: ")
start_port = int(input("Porta iniziale: "))
end_port = int(input("Porta finale: "))

print(f"\nScansione di {ip} sulle porte da {start_port} a {end_port}...\n")
```

Abbiamo usato la libreria **socket** che serve per lavorare con le connessioni di rete ed è utilizzata per creare socket e fare prove di connessioni TCP alle porte del server da scansionare.

Ci basterà definire l'input di un utente che deve digitare sulla tastiera ovvero indirizzo IP e porte da scansionare usando con i comandi **ip , start port e end port .**
Con print stampiamo un semplice messaggio di avviso prima di iniziare la scansione.

Port Scanner

Il processo prevede la creazione di un socket TCP IPv4, che viene poi utilizzato in un ciclo **for** per tentare una connessione a ciascuna porta di un intervallo specificato, impostando un timeout di un secondo per ogni tentativo di risposta del server.

AF_INET - specifica che stiamo usando indirizzi IPv4.
SOCK_STREAM - indica che stiamo usando il protocollo TCP.

```
import socket

ip = input("Inserisci l'indirizzo IP da scansionare: ")
start_port = int(input("Porta iniziale: "))
end_port = int(input("Porta finale: "))

print(f"\nScansione di {ip} sulle porte da {start_port} a {end_port}...\n")


for port in range(start_port, end_port + 1):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.settimeout(1)
    result = s.connect_ex((ip, port))
```

Port Scanner

```python
ip = input("Inserisci l'indirizzo IP da scansionare: ")
start_port = int(input("Porta iniziale: "))
end_port = int(input("Porta finale: "))

print(f"\nScansione di {ip} sulle porte da {start_port} a {end_port}...\n")


for port in range(start_port, end_port + 1):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.settimeout(1)
    result = s.connect_ex((ip, port))

    if result == 0:
        print(f"Porta {port}: APERTA")
    else:
        print(f"Porta {port}: CHIUSA")

    s.close()
```

connect ex è il comando che serve per connettersi all'indirizzo IP e alla porta corrente e Tramite **result** provando a connettersi alla porta ci dirà:
Se result == 0, la connessione è riuscita ovvero porta aperta.
Altrimenti, la porta è chiusa o inaccessibile.

Port Scanner (Test)

```python
ip = input("Inserisci l'indirizzo IP da scansionare: ")
start_port = int(input("Porta iniziale: "))
end_port = int(input("Porta finale: "))

print(f"\nScansione di {ip} sulle porte da {start_port} a {end_port}...\n")


for port in range(start_port, end_port + 1):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.settimeout(1)
    result = s.connect_ex((ip, port))

    if result == 0:
        print(f"Porta {port}: APERTA")
    else:
        print(f"Porta {port}: CHIUSA")

    s.close()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Inserisci l'indirizzo IP da scansionare: 127.0.0.1
Porta iniziale: 20
Porta finale: 25

Scansione di 127.0.0.1 sulle porte da 20 a 25...

Porta 20: CHIUSA
Porta 21: CHIUSA
Porta 22: CHIUSA
Porta 23: CHIUSA
Porta 24: CHIUSA
Porta 25: CHIUSA
```

Avviando il programma ci basterà inserire un indirizzo IP e il range di porte da voler scansionare e come risultato il programma scansionerà l'indirizzo ip e darà come output lo stato delle porte se sono **APERTE o CHIUSE**

Protocollowolf

Python Tools

Packet sniffing

Packet Sniffing

importiamo la funzione **sniff**() da **Scapy**, che serve per catturare pacchetti di rete in tempo reale

Successivamente definiamo una funzione in cui con Il parametro pacchetto riusciamo ad avere tutte le informazioni sul pacchetto catturato e viene eseguita ogni volta che arriva un pacchetto.
successivamente Stampiamo un riassunto del pacchetto catturato.

```
from scapy.all import sniff
```

```
from scapy.all import sniff

def mostra_pacchetto(pacchetto):
    print(pacchetto.summary())
```

Packet Sniffing

con **sniff** avviamo lo sniffing dei pacchetti, prn=mostra_pacchetto: chiama la funzione mostra_pacchetto() per ogni pacchetto. e con count=0: cattura illimitatamente, finché l'utente non ferma il programma

```
from scapy.all import sniff

def mostra_pacchetto(pacchetto):
    print(pacchetto.summary())

print("In ascolto dei pacchetti... (CTRL+C per interrompere)")
sniff(prn=mostra_pacchetto, count=0)
```

Packet Sniffing

Sta rispettando il Three-Way Handshake TCP ovvero :
Client chiede al Server: S (**SYN**)

Server chiede al Client: SA (**SYN** + **ACK**)

Client risponde al Server: A (**ACK**)

Dopo questo scambio, la connessione TCP è stabilita e i dati possono iniziare a fluire.

```
┌──(kali㉿kali)-[~]
└─$ sudo python3 sniffer.py
[sudo] password for kali:
In ascolto dei pacchetti... (CTRL+C per interrompere)
Ether / ARP who has 192.168.50.1 says 192.168.50.101
Ether / ARP who has 192.168.50.1 says 192.168.50.101
Ether / ARP who has 192.168.50.1 says 192.168.50.101
Ether / IP / TCP 192.168.50.101:41772 > 192.168.50.100:http S
Ether / ARP who has 192.168.50.101 says 192.168.50.100 / Padding
Ether / ARP is at 08:00:27:8b:48:76 says 192.168.50.101
Ether / IP / TCP 192.168.50.100:http > 192.168.50.101:41772 SA
Ether / IP / TCP 192.168.50.101:41772 > 192.168.50.100:http A
Ether / IP / TCP 192.168.50.101:41772 > 192.168.50.100:http PA / Raw
Ether / IP / TCP 192.168.50.100:http > 192.168.50.101:41772 A
Ether / IP / TCP 192.168.50.100:http > 192.168.50.101:41772 PA / Raw
Ether / IP / TCP 192.168.50.101:41772 > 192.168.50.100:http A
Ether / ARP who has 192.168.50.1 says 192.168.50.101
Ether / ARP who has 192.168.50.1 says 192.168.50.101
Ether / ARP who has 192.168.50.1 says 192.168.50.101
Ether / ARP who has 192.168.50.1 says 192.168.50.101
Ether / ARP who has 192.168.50.1 says 192.168.50.101
Ether / ARP who has 192.168.50.100 says 192.168.50.101
Ether / ARP is at 08:00:27:d5:c1:ce says 192.168.50.100 / Padding
Ether / IP / TCP 192.168.50.101:41772 > 192.168.50.100:http PA / Raw
Ether / IP / TCP 192.168.50.100:http > 192.168.50.101:41772 A
Ether / IP / TCP 192.168.50.100:http > 192.168.50.101:41772 A / Raw
Ether / IP / TCP 192.168.50.101:41772 > 192.168.50.100:http A
```

Protocollowolf

Python Tools

Socket di rete

Socket Test

```
--- Risposta HTTP Grezza Ricevuta dal Socket ---
HTTP/1.1 200 OK
Date: Fri, 25 Jul 2025 11:51:10 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: private, max-age=10800, pre-check=10800
Set-Cookie: phpMyAdmin=a2220807a29c5a49cae076a83f364191ebaebe83; path=/phpMyAdmin/; HttpOnly
Set-Cookie: pma_lang=en-utf-8; expires=Sun, 24-Aug-2025 11:51:10 GMT; path=/phpMyAdmin/; httponly
Set-Cookie: pma_charset=utf-8; expires=Sun, 24-Aug-2025 11:51:10 GMT; path=/phpMyAdmin/; httponly
Set-Cookie: pma_collation_connection=deleted; expires=Thu, 25-Jul-2024 11:51:09 GMT; path=/phpMyAdmin/; httponly
Set-Cookie: pma_theme=original; expires=Sun, 24-Aug-2025 11:51:10 GMT; path=/phpMyAdmin/; httponly
Last-Modified: Tue, 09 Dec 2008 17:24:00 GMT
Content-Length: 4145
Connection: close
Content-Type: text/html; charset=utf-8

<!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" dir="ltr">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <link rel="icon" href="./favicon.ico" type="image/x-icon" />
    <link rel="shortcut icon" href="./favicon.ico" type="image/x-icon" />
    <title>phpMyAdmin </title>
    <link rel="stylesheet" type="text/css" href="phpmyadmin.css.php?lang=en-utf-8&amp;convcharset=utf-8&amp;token=a4bdb047bb6e5d1d3fa18601e868205
&amp;nocache=2457687151" />
    <link rel="stylesheet" type="text/css" href="print.css" media="print" />
    <meta name="robots" content="noindex,nofollow" />
<script type="text/javascript">
```

Socket Test

```
Log in</legend>

        <div class="item">
            <label for="input_username">Username:</label>
            <input type="text" name="pma_username" id="input_username" value="" size="24" class="textfield"/>
        </div>
        <div class="item">
            <label for="input_password">Password:</label>
            <input type="password" name="pma_password" id="input_password" value="" size="24" class="textfield" />
        </div>
        <input type="hidden" name="server" value="1" />    </fieldset>
    <fieldset class="tblFooters"><input type="hidden" name="phpMyAdmin" value="a2220807a29c5a49cae076a83f364191ebaebe83" />
        <input value="Go" type="submit" id="input_go" />
    <input type="hidden" name="lang" value="en-utf-8" /><input type="hidden" name="convcharset" value="utf-8" /><input type="hidden" name="token" value="a4bdb047bb6
5d1d3fa18601e8682059" />    </fieldset>
</form>

    <div><div class="warning">Cannot load <a href="http://php.net/mcrypt" target="Documentation"><em>mcrypt</em></a> extension. Please check your PHP configuration.
/div><div class="notice">Cookies must be enabled past this point.</div></div></div>
<script type="text/javascript">
// <![CDATA[
function PMA_focusInput()
{
    var input_username = document.getElementById('input_username');
    var input_password = document.getElementById('input_password');
    if (input_username.value == '') {
        input_username.focus();
    } else {
        input_password.focus();
    }
}

window.setTimeout('PMA_focusInput()', 500);
// ]]>
</script>
    </body>
</html>

--- Fine Risposta ---

Socket chiuso.
```

Il team

Protocollowolf

Samuel Lukac

Jacopo Sarvello

Marco Potenza

Gioele Parla

Membro Onorario

Francesco Livrieri

Gennaro Del Giudice

Alessandro Castello