|  |
| --- |
| 4aWI |
| DVWA |
| Beer Martin |

|  |
| --- |
| Beer Martin  10.1.2025 |

Inhalt

[1.1 Brute Force 0](#_Toc188001041)

[1.1.1 Information 0](#_Toc188001042)

[1.1.2 Graphische Darstellung 0](#_Toc188001043)

[1.2 Command Injection 1](#_Toc188001044)

[1.3 CSRF 3](#_Toc188001045)

[1.4 File Inclusion 3](#_Toc188001046)

[1.5 File Upload 3](#_Toc188001047)

[1.6 Insecure CAPTCHA 3](#_Toc188001048)

[1.7 SQL Injection 3](#_Toc188001049)

[1.8 SQL Injection (blind) 3](#_Toc188001050)

[1.9 Weak Session IDs 3](#_Toc188001051)

[1.10 XSS (DOM) 3](#_Toc188001052)

[1.11 XSS (Reflected) 3](#_Toc188001053)

[1.12 XSS (Stored) 3](#_Toc188001054)

[1.13 CSP Byepass 3](#_Toc188001055)

[1.14 JavaScript 3](#_Toc188001056)

[1.15 Authorisation Byepass 3](#_Toc188001057)

[1.16 Open HTTP Redirect 3](#_Toc188001058)

[1.17 Cryptography 3](#_Toc188001059)

[1.18 TOOLS 4](#_Toc188001060)

# Brute Force

## 1.1.1 Information

Ein Brute Force Angriff (Brute Force Attack auf Englisch) ist eine Methode, die Cyberkriminelle anwenden, um Passwörter und andere Zugangsdaten zu knacken. Bei einem Brute Force Angriff greift ein Angreifer auf eine Liste an häufigen Wörtern zurück und probiert sie der Reihe nach durch, bis eines funktioniert. Wenn diese Liste zu keinem Ergebnis führt, versucht er es mit verschiedenen Buchstabenkombinationen. Es braucht manchmal tausende von Versuchen, bis ein Passwort geknackt ist. Deshalb kommen für Brute Force, Automatisierungstools zum Einsatz, die es einem Angreifer ermöglichen, sehr viele Versuche möglichst schnell durchzuführen.[[1]](#footnote-1)

Durch einige Rechenbeispiele soll das Zusammenwirken von Länge und verwendeter Zeichen für die Sicherheit eines Passworts veranschaulicht werden. In den Rechenbeispielen wird mit einer Generierung von 2 Milliarden Schlüsseln pro Sekunde gerechnet, da dies ungefähr der Geschwindigkeit eines sehr starken Einzelrechners entspricht.

Beim Erstellen eines Passworts stehen Ihnen in der Regel folgende Zeichen zur Verfügung:

* Zahlen (10 verschiedene: 0-9)
* Buchstaben (52 verschiedene: A-Z und a-z)
* Sonderzeichen (32 verschiedene).[[2]](#footnote-2)

## 1.1.2 Graphische Darstellung

[](https://www.google.com/url?sa=i&url=https%3A%2F%2Frublon.com%2Fblog%2Fbrute-force-dictionary-attack-difference%2F&psig=AOvVaw0SeKz697a1BzNO8LdCtqlc&ust=1737191516353000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCLDYzuy0_IoDFQAAAAAdAAAAABAK)

Abbildung 1 BruceForce Attacke

## 1.1.3 Code

**Low Brute Force Code:**

<?php

if( isset( $\_GET[ 'Login' ] ) ) {

// Get username

$user = $\_GET[ 'username' ];

// Get password

$pass = $\_GET[ 'password' ];

$pass = md5( $pass );

// Check the database

$query = "SELECT \* FROM `users` WHERE user = '$user' AND password = '$pass';";

$result = mysqli\_query($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>' );

if( $result && mysqli\_num\_rows( $result ) == 1 ) {

// Get users details

$row = mysqli\_fetch\_assoc( $result );

$avatar = $row["avatar"];

// Login successful

echo "<p>Welcome to the password protected area {$user}</p>";

echo "<img src=\"{$avatar}\" />";

}

else {

// Login failed

echo "<pre><br />Username and/or password incorrect.</pre>";

}

((is\_null($\_\_\_mysqli\_res = mysqli\_close($GLOBALS["\_\_\_mysqli\_ston"]))) ? false : $\_\_\_mysqli\_res);

}

?>

**Unbegrenzte Login-Versuche**: Das Skript blockiert den Angreifer nicht nach einer bestimmten Anzahl falscher Passworteingaben.

**Schnelles Durchprobieren**: Da es kein künstliches Delay (z. B. sleep) gibt, können automatisierte Tools sehr schnell alle möglichen Kombinationen testen.

**Keine Account-Sperre**: Selbst nach vielen Fehleingaben wird der Login nicht für eine bestimmte Zeit gesperrt, sodass ein automatisiertes Ausprobieren nicht gestoppt wird.

**Impossible Brute Force Code:**

<?php

if (isset($\_POST['Login'], $\_POST['username'], $\_POST['password'])) {

// 1) Anti-CSRF prüfen

checkToken($\_REQUEST['user\_token'], $\_SESSION['session\_token'], 'index.php');

// 2) Eingaben validieren/escapen

$user = mysqli\_real\_escape\_string($GLOBALS["\_\_\_mysqli\_ston"], stripslashes($\_POST['username']));

$pass = md5(mysqli\_real\_escape\_string($GLOBALS["\_\_\_mysqli\_ston"], stripslashes($\_POST['password'])));

// 3) Lockout-Einstellungen

$failed\_limit = 3;

$lockout\_time = 15; // Minuten

$account\_locked = false;

// 4) Ist der Account schon gesperrt?

$data = $db->prepare('SELECT failed\_login, last\_login FROM users WHERE user = :user LIMIT 1;');

$data->bindParam(':user', $user, PDO::PARAM\_STR);

$data->execute();

$row = $data->fetch();

if ($data->rowCount() && $row['failed\_login'] >= $failed\_limit) {

if (time() < strtotime($row['last\_login']) + ($lockout\_time \* 60)) {

$account\_locked = true;

}

}

// 5) Prüfung der Logindaten

$data = $db->prepare('SELECT \* FROM users WHERE user = :user AND password = :pass LIMIT 1;');

$data->bindParam(':user', $user, PDO::PARAM\_STR);

$data->bindParam(':pass', $pass, PDO::PARAM\_STR);

$data->execute();

// 6) Erfolgreicher Login?

if ($data->rowCount() && !$account\_locked) {

// a) Fehlversuche zurücksetzen

$upd = $db->prepare('UPDATE users SET failed\_login = 0 WHERE user = :user LIMIT 1;');

$upd->bindParam(':user', $user, PDO::PARAM\_STR);

$upd->execute();

echo "Willkommen, $user";

} else {

// a) Verzögerung bei falschem Login

sleep(rand(2, 4));

// b) Fehlversuche hochzählen

$upd = $db->prepare('UPDATE users SET failed\_login = failed\_login + 1 WHERE user = :user LIMIT 1;');

$upd->bindParam(':user', $user, PDO::PARAM\_STR);

$upd->execute();

echo "Fehler: Ungültige Login-Daten oder Account gesperrt.";

}

// 7) Letzte Loginzeit aktualisieren

$upd = $db->prepare('UPDATE users SET last\_login = NOW() WHERE user = :user LIMIT 1;');

$upd->bindParam(':user', $user, PDO::PARAM\_STR);

$upd->execute();

}

// 8) Anti-CSRF-Token erzeugen

generateSessionToken();

?>

**Entscheidende Punkte für sicheren Code sind:**

**Anti-CSRF-Token**: Verhindert, dass Angreifer Logins von fremden Seiten auslösen.

**Eingabesanitierung**: Schützt vor SQL-Injections.

**Lockout**: Nach totalfailedlogintotal\_failed\_logintotalf​ailedl​ogin Fehlversuchen wird der Account für lockouttimelockout\_timelockoutt​ime Minuten gesperrt.

**Zeitverzögerung**: sleep(rand(2, 4)) erschwert automatisierte Angriffe.

**Zurücksetzen und Inkrementieren der**: Erlaubt dem System, erfolgreiche Logins wieder zu normalisieren und bei Fehlern konsequent zu erhöhen.

**Update der letzten Login-Zeit**: Dient zur Berechnung der Sperrzeit und zur Protokollierung.

## 1.1.4 Tools zum Hacken und Testen

* Hydra
* Burpsuite
* OWASP ZAP
* Klassische Programmierung: Java, JavaScript, node
* Sqlmap (Testing-Tool)

# Command Injection

# CSRF

# 1.4 File Inclusion

# 1.5 File Upload

# 1.6 Insecure CAPTCHA

# 1.7 SQL Injection

# 1.8 SQL Injection (blind)

# 1.9 Weak Session IDs

# 1.10 XSS (DOM)

# 1.11 XSS (Reflected)

# 1.12 XSS (Stored)

# 1.13 CSP Byepass

# 1.14 JavaScript

# 1.15 Authorisation Byepass

# 1.16 Open HTTP Redirect

# 1.17 Cryptography

# 1.18 TOOLS

[Abbildung 1 BruceForce Attacke 1](#_Toc188001649)

1. <https://www.proofpoint.com/de/threat-reference/brute-force-attack> [↑](#footnote-ref-1)
2. https://www.password-depot.de/de/know-how/brute-force-angriffe.htm [↑](#footnote-ref-2)