Abiva Wayne

DVWA-Sytems

Inhaltsverzeichnis

[1. Brute-Force 2](#_Toc191027324)

[1.1 Information 2](#_Toc191027325)

[1.2 Graphische Darstellungen 3](#_Toc191027326)

[1.3 Code 3](#_Toc191027327)

[1.4 Tools zum Hacken und Testen 7](#_Toc191027328)

[1.5 Umsetzung 7](#_Toc191027329)

# Brute-Force

## Information

**Brute Force** ist eine Cyberkriminelle Angriffsmethode, die mit **automatisierten Tools** und **leistungsstarken Hardware**, in einer kurzen Zeit Login-Daten geknackt werden können.[[1]](#footnote-1)

**Man unterscheidet zwischen mehrere Methoden:**[[2]](#footnote-2)

**Traditonell**: Alle möglichen Kombinationen zwischen Benutzernamen und Passwörter werden ausprobiert, mithilfe leistungsstarken Hardware .

**Credential Stuffing**: Bereits bekannte Login-Daten (E-Mail und Passwort) werden auf verschiedenen Diensten getestet, um von der Wiederverwendung derselben Kombination durch Benutzer zu profitieren. Botnetze beschleunigen und verschleiern diese Angriffe.

**Credential Cracking**: Teilweise bekannte Zugangsdaten (z. B. nur der Benutzername) werden durch Erraten des Passworts ergänzt, entweder mit Passwortlisten oder durch Zufallsgenerierung. Auch hier kommen Botnetze zum Einsatz.

**Rainbow-Table Angriff**: Gehasht gespeicherte Passwörter werden mit vordefinierten Hashwerten in sogenannten Rainbow Tables abgeglichen, um das ursprüngliche Passwort zu finden.

**Dictionary Angriff**: Listen mit häufig verwendeten Passwörtern und Benutzernamen werden durchprobiert, um Zugriff auf Konten zu erhalten.

**Wie schützt man sich gegen Brute-Force:[[3]](#footnote-3)**

**Für Benutzer:**

* **Starke Passwörter**: Verwenden Sie Passwörter mit mindestens 15 Zeichen, die keine leicht erratbaren Informationen wie Wörter oder Geburtsdaten enthalten. Passwort-Manager helfen bei der Verwaltung sicherer Passwörter.
* **Zwei-Faktor-Authentifizierung (2FA)**: Aktivieren Sie immer die 2FA, um einen zusätzlichen Schutz zu gewährleisten.
* **Verschlüsselte Kommunikation**: Nutzen Sie HTTPS oder andere verschlüsselte Protokolle, um die Sicherheit der Datenübertragung zu gewährleisten.
* **Passwortlose Methoden**: Setzen Sie auf Technologien wie FIDO2, die biometrische Verfahren oder Hardware-Authentifizierung verwenden.

**Für Administratoren:**

* **Anmeldeversuche begrenzen**: Legen Sie fest, dass Konten nach einer bestimmten Anzahl fehlerhafter Logins vorübergehend gesperrt werden.
* **Standardports ändern**: Ändern Sie den Standardport (z. B. für SSH von 22 auf einen anderen zwischen 1024 und 65536), um Angreifern die Suche zu erschweren.
* **Remote Desktop-Zugriff beschränken**: Erlauben Sie den Zugang nur für ausgewählte IP-Adressen oder implementieren Sie ein Zero-Trust-Modell mit 2FA, Firewalls und Geoblocking.
* **CAPTCHA verwenden**: Verhindern Sie automatisierte Angriffe durch die Integration von CAPTCHA-Anfragen.
* **Unbenutzte Accounts löschen**: Entfernen Sie inaktive Konten, die veraltete Anmeldedaten enthalten könnten.
* **Traffic überwachen**: Überprüfen Sie Logdateien regelmäßig, um verdächtige Aktivitäten wie wiederholte Anmeldeversuche zu erkennen, und sperren Sie auffällige IP-Adressen.
* **Passwortverschlüsselung**: Speichern Sie Passwörter nur verschlüsselt (z. B. mit 256-Bit-Algorithmen), um Credential Stuffing zu verhindern.

## Brute-Force-Attacke: Definition und Funktionsweise | MyraGraphische Darstellungen

[[4]](#footnote-4)

## Code

**Low Brute Force**

<?php

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

// Get username

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

// Get password

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

$pass = md5( $pass );

**Es gibt keine Mechanismen, die verhindern, dass ein Angreifer beliebig viele Login-Versuche durchführt. Brute-Force-Angriffe sind daher leicht möglich.**

// Check the database

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

**Es gibt keine Überprüfung oder Sanitisierung der Eingaben. Dadurch wird der Code anfällig für SQL-Injection.**

$result = mysqli\_query($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);

**Impossible Brute Force:**  
<?php

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

// Check Anti-CSRF token

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

// Sanitise username input

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

$user = stripslashes( $user );

$user = ((isset($GLOBALS["\_\_\_mysqli\_ston"]) && is\_object($GLOBALS["\_\_\_mysqli\_ston"])) ? mysqli\_real\_escape\_string($GLOBALS["\_\_\_mysqli\_ston"], $user ) : ((trigger\_error("[MySQLConverterToo] Fix the mysql\_escape\_string() call! This code does not work.", E\_USER\_ERROR)) ? "" : ""));

// Sanitise password input

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

$pass = stripslashes( $pass );

$pass = ((isset($GLOBALS["\_\_\_mysqli\_ston"]) && is\_object($GLOBALS["\_\_\_mysqli\_ston"])) ? mysqli\_real\_escape\_string($GLOBALS["\_\_\_mysqli\_ston"], $pass ) : ((trigger\_error("[MySQLConverterToo] Fix the mysql\_escape\_string() call! This code does not work.", E\_USER\_ERROR)) ? "" : ""));

$pass = md5( $pass );

// Default values

$total\_failed\_login = 3;

$lockout\_time = 15;

$account\_locked = false;

// Check the database (Check user information)

$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();

// Check to see if the user has been locked out.

if( ( $data->rowCount() == 1 ) && ( $row[ 'failed\_login' ] >= $total\_failed\_login ) ) {

// User locked out. Note, using this method would allow for user enumeration!

//echo "<pre><br />This account has been locked due to too many incorrect logins.</pre>";

// Calculate when the user would be allowed to login again

$last\_login = strtotime( $row[ 'last\_login' ] );

$timeout = $last\_login + ($lockout\_time \* 60);

$timenow = time();

/\*

print "The last login was: " . date ("h:i:s", $last\_login) . "<br />";

print "The timenow is: " . date ("h:i:s", $timenow) . "<br />";

print "The timeout is: " . date ("h:i:s", $timeout) . "<br />";

\*/

// Check to see if enough time has passed, if it hasn't locked the account

if( $timenow < $timeout ) {

$account\_locked = true;

// print "The account is locked<br />";

}

}

// Check the database (if username matches the password)

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

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

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

$data->execute();

$row = $data->fetch();

// If its a valid login...

if( ( $data->rowCount() == 1 ) && ( $account\_locked == false ) ) {

// Get users details

$avatar = $row[ 'avatar' ];

$failed\_login = $row[ 'failed\_login' ];

$last\_login = $row[ 'last\_login' ];

// Login successful

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

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

// Had the account been locked out since last login?

if( $failed\_login >= $total\_failed\_login ) {

echo "<p><em>Warning</em>: Someone might of been brute forcing your account.</p>";

echo "<p>Number of login attempts: <em>{$failed\_login}</em>.<br />Last login attempt was at: <em>{$last\_login}</em>.</p>";

}

// Reset bad login count

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

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

$data->execute();

} else {

// Login failed

sleep( rand( 2, 4 ) );

// Give the user some feedback

echo "<pre><br />Username and/or password incorrect.<br /><br/>Alternative, the account has been locked because of too many failed logins.<br />If this is the case, <em>please try again in {$lockout\_time} minutes</em>.</pre>";

// Update bad login count

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

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

$data->execute();

}

// Set the last login time

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

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

$data->execute();

}

// Generate Anti-CSRF token

generateSessionToken();

?>

Entscheidende Punkte für sicheren Code sind:

1. **Check Login versuche!!**
2. **CSRF-Token**
3. **Verhinderung über Zeiteinstieg**
4. **Check auf MENSCHEN ALS Bedient (Captures)**
5. **2FA**

## Tools zum Hacken und Testen

* Hydra
* JohnTheRipper
* Burpsuite
* Klassisches Programmierung: Java, JavaScript, node …

## Umsetzung

Für die Umsetzung haben wir die Sicherheit auf Low eingestellt. Die Tools die benutzt wurden ist einfaches JavaScript Code mit Node.js.

1. [Brute Force Angriffe: Methoden, Tools und Schutzmaßnahmen | Proofpoint DE](https://www.proofpoint.com/de/threat-reference/brute-force-attack#:~:text=Ein%20Brute%20Force%20Angriff%20(Brute,nach%20durch%2C%20bis%20eines%20funktioniert.), 17.01,2025 [↑](#footnote-ref-1)
2. <https://www.myrasecurity.com/de/brute-force-attacke/>, 17.01.2025 [↑](#footnote-ref-2)
3. <https://www.experte.de/it-sicherheit/brute-force-angriff>, 17.01.2025 [↑](#footnote-ref-3)
4. <https://www.myrasecurity.com/de/brute-force-attacke/>, 24.01.2025 [↑](#footnote-ref-4)