DVWA

Julian Barfus

4aWI

Table of Contents

[Brute Force 3](#_Toc188005366)

[1.1 Definition: 3](#_Toc188005367)

[1.2 Beliebte Brute Force Tools: 3](#_Toc188005368)

[1.3 Code 3](#_Toc188005369)

[Unsicherer Code 3](#_Toc188005370)

[Sicherer Code 4](#_Toc188005371)

[1.4 Grafik 5](#_Toc188005372)

[Brute Force Tools 6](#_Toc188005373)

[Aircrack-ng 6](#_Toc188005374)

[John the ripper 7](#_Toc188005375)

[Hydra 8](#_Toc188005376)

[Burpsuite 9](#_Toc188005377)

[Ncrack 10](#_Toc188005378)

# Brute Force

## 1.1 Definition:

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.

[Literaturhinweis](https://www.proofpoint.com/de/threat-reference/brute-force-attack#:~:text=Ein%20Brute%20Force%20Angriff%20(Brute,nach%20durch%2C%20bis%20eines%20funktioniert.) 13.01.2025

## 1.2 Beliebte Brute Force Tools:

* Aircrack-ng
* John the Ripper
* Klassische Programierung
* Hydra
* Burpsuite
* Ncrack

[Literaturhinweis](https://www.proofpoint.com/de/threat-reference/brute-force-attack#:~:text=Ein%20Brute%20Force%20Angriff%20(Brute,nach%20durch%2C%20bis%20eines%20funktioniert.) 13.01.2025

## 1.3 Code

### Unsicherer 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);

}

?>

#### Warum ist dieser Code unsicher

Mann kann so viele passwort versuche machen wie man möchte

### Sicherer Code

<?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

* Check Loginversuche!!
* CSRF-Token
* Verhinderung über Zeiteinstieg!!
* 2FA
* Check auf mensch als Bediner(Captures!!!)

## 1.4 Grafik



# Brute Force Tools

## Aircrack-ng

**Beschreibung:**

* Ein Paket von Tools zum Testen der Sicherheit von WLAN-Netzwerken.
* Hauptsächlich verwendet für:
  + Überwachung von Netzwerken.
  + Angriff auf WLAN-Verschlüsselungen wie WEP und WPA/WPA2-PSK.
  + Capturing und Cracking von Handshakes.

**Hauptfunktionen:**

* **Airodump-ng**: Netzwerk-Sniffing und Paketerfassung.
* **Aircrack-ng**: Passwort-Cracking basierend auf den erfassten Daten.
* **Aireplay-ng**: Packet-Injection für Replay-Angriffe.
* **Airdecap-ng**: Entschlüsselung von erfassten WLAN-Paketen.

**Unterstützte Betriebssysteme:**

* **Linux**: Native Unterstützung, z. B. in Kali Linux.
* **Windows**: Unterstützt, aber eingeschränkte Funktionalität.
* **macOS**: Unterstützt über Homebrew.
* **BSD-Systeme**: Teilweise Unterstützung.

## John the ripper



**Beschreibung:**

* Ein Passwort-Cracking-Tool zur Überprüfung der Passwortsicherheit.
* Kann verschiedene Hash-Typen (z. B. MD5, SHA-1, bcrypt) knacken.
* Unterstützt Wörterbuch-Angriffe, Brute-Force und hybride Angriffe.

**Hauptfunktionen:**

* Flexible Konfiguration für verschiedene Hash-Typen.
* Unterstützung von Multi-Core-Prozessoren und GPU-basiertem Cracking.
* Erweiterbar durch Plugins und benutzerdefinierte Regeln.

**Unterstützte Betriebssysteme:**

* **Linux**: Native Unterstützung.
* **Windows**: Unterstützt (über Cygwin oder vorkompilierte Versionen).
* **macOS**: Unterstützt.
* **UNIX und BSD**: Unterstützt.

## Hydra



**Beschreibung:**

* Ein Passwort-Cracking-Tool für Online-Dienste und Protokolle.
* Unterstützt zahlreiche Protokolle wie HTTP, FTP, SSH, RDP, MySQL, etc.

**Hauptfunktionen:**

* Sehr schnell und parallelisiert.
* Unterstützt Brute-Force- und Wörterbuch-Angriffe.
* Flexibel und erweiterbar durch benutzerdefinierte Module.

**Unterstützte Betriebssysteme:**

* **Linux**: Native Unterstützung (z. B. in Kali Linux).
* **Windows**: Unterstützt über Cygwin.
* **macOS**: Unterstützt über Homebrew.
* **UNIX-Systeme**: Teilweise unterstützt.

## Burpsuite



**Beschreibung:**

* Ein umfassendes Tool für Penetrationstests von Webanwendungen.
* Ermöglicht das Identifizieren von Sicherheitslücken wie XSS, SQL-Injection, CSRF und mehr.

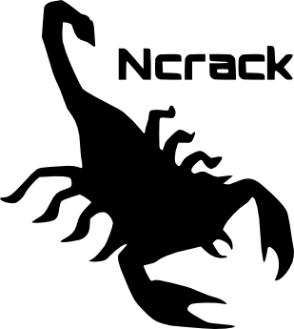
**Hauptfunktionen:**

* **Proxy**: Interceptieren und Modifizieren von HTTP/S-Anfragen.
* **Scanner**: Automatisches Scannen nach Schwachstellen.
* **Intruder**: Automatisierte Angriffe auf Parameter und Formulare.
* **Repeater**: Manuelles Testen von HTTP/S-Anfragen.

**Unterstützte Betriebssysteme:**

* **Linux**: Unterstützt.
* **Windows**: Unterstützt.
* **macOS**: Unterstützt.
* **Java-basiert**: Funktioniert auf jedem Betriebssystem mit einer Java Runtime Environment (JRE).

## Ncrack



**Beschreibung:**

* Ein Tool zum Cracking von Netzwerk-Authentifizierung.
* Speziell entwickelt, um hochsichere Passwörter in Protokollen wie SSH, RDP, FTP, Telnet, und HTTP zu knacken.

**Hauptfunktionen:**

* Schnelle und parallele Authentifizierungsprüfungen.
* Unterstützung für passwortgeschützte Dienste (z. B. VNC, POP3).
* Erweiterbar durch benutzerdefinierte Module.

**Unterstützte Betriebssysteme:**

* **Linux**: Native Unterstützung.
* **Windows**: Unterstützt.
* **macOS**: Unterstützt.
* **BSD-Systeme**: Unterstützt.