

Cebrail Demir 4aWi

DVWA-Protokoll

[Brute Force 3](#_Toc159223568)

[Command Injection 3](#_Toc159223569)

[C SRF 3](#_Toc159223570)

[File Inclusion 3](#_Toc159223571)

[File Upload 3](#_Toc159223572)

[Insecure CAPTCHA 3](#_Toc159223573)

[SQL Injection 3](#_Toc159223574)

[SQL Injection (Blind) 3](#_Toc159223575)

[Weak Session IDs 3](#_Toc159223576)

[XSS (DOM) 3](#_Toc159223577)

[XSS (Reflected) 3](#_Toc159223578)

[XSS (Stored) 3](#_Toc159223579)

[CSP Bypass 3](#_Toc159223580)

[Javascript 3](#_Toc159223581)

[Authorisation Bypass 3](#_Toc159223582)

[Open HTTP Redirect 3](#_Toc159223583)

# Brute Force

// 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();

Theorie:

Brute-Force-Angriffe beinhalten den Versuch eines Angreifers, Zugang zu einem System zu erlangen, indem er wiederholt verschiedene Kombinationen von Benutzernamen und Passwörtern ausprobiert.

Beispiel 1:

Der Angreifer könnte eine Webseite attackieren, indem er bekannte Seiten ausprobiert und überprüft, ob sie existieren.

Brute Force PHP-Schnipsel:

GET-Parameter für Benutzeranmeldeinformationen:

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

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

Unsicherheit: Die Verwendung von GET-Parametern ist riskant, da sie in der URL sichtbar sind und leicht abgefangen oder manipuliert werden können.

Direkte SQL-Zusammensetzung:

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

Unsicherheit: Die direkte Zusammensetzung von SQL-Abfragen macht die Anwendung anfällig für SQL-Injektionen, da Angriffe durch Einfügen von schädlichem SQL-Code in die Eingabefelder möglich sind.

Fehlen von Sicherheitsprüfungen:

Es gibt keine Überprüfung auf Anti-CSRF-Token oder Begrenzung fehlgeschlagener Anmeldeversuche, was die Anwendung anfälliger für Brute-Force-Angriffe macht.