 Protokoll

|  |  |  |
| --- | --- | --- |
| Thema: | **DVWA** | Klasse: 4aWI |
| Datum: |  | Lehrer: Schwärzler |
| Name: | Nemanja Katanic | Fach: SESD |

Inhaltsangabe:

Inhalt

[Brute Force 1](#_Toc160432798)

[Command Injection 4](#_Toc160432799)

[CSRF 5](#_Toc160432800)

[File Inclusion 7](#_Toc160432801)

[File Upload 8](#_Toc160432802)

[Insecure CAPTCHA 11](#_Toc160432803)

[SQL Injection 12](#_Toc160432804)

[SQL Injection (Blind) 13](#_Toc160432805)

[Weak Session IDs 16](#_Toc160432806)

[XSS (DOM) 17](#_Toc160432807)

[XSS (DOM) - Snippet 1: 17](#_Toc160432808)

[XSS (DOM) - Snippet 2: 17](#_Toc160432809)

[Empfehlungen: 17](#_Toc160432810)

[XSS (Reflected) 18](#_Toc160432811)

[XSS (Stored) 19](#_Toc160432812)

[CSP Bypass 21](#_Toc160432813)

[JavaScript 23](#_Toc160432814)

[Authorisation Bypass 24](#_Toc160432815)

[Open HTTP Redirect 25](#_Toc160432816)

# Brute Force

Theorie:  
Brute-Force-Angriffe bedeuten, dass ein Angreifer versucht, Zugang zu einem System zu erlangen, indem er wiederholt verschiedene Kombinationen von Benutzernamen und Passwörtern ausprobiert.

Beispiel 1:

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

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Brute Force PHP Snippet:

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.

Beispiel 2:

Bei der Authentifizierung könnte ein Angreifer versuchen, sich Zugang zu einem Konto zu verschaffen, indem er Listen gängiger Benutzernamen und Passwörter durchprobiert. Wenn es keine strengen Passwortregeln gibt, könnte dies erfolgreich sein. Sicherheits-Tools wie der Php-Brute-Force-Attack Detector helfen dabei, solche Angriffe zu erkennen.

**Verbessertes PHP Snippet:**

**POST-Parameter für Benutzeranmeldeinformationen:**

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

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

Sicherheit: Die Verwendung von POST-Parametern ist sicherer als GET, da POST-Daten nicht in der URL sichtbar sind.

**Verwendung von PDO und vorbereiteten Statements:**

Sicherheit: PDO und vorbereitete Statements schützen vor SQL-Injektionen, indem sie sicherstellen, dass Eingaben als Daten und nicht als Teil der Abfrage interpretiert werden.

**Implementierung von Sicherheitsmaßnahmen:**

Überprüfung von Anti-CSRF-Token und Begrenzung fehlgeschlagener Anmeldeversuche: Das Snippet enthält Funktionen wie checkToken() und die Überwachung fehlgeschlagener Anmeldeversuche, um die Anwendung vor Brute-Force-Angriffen zu schützen.

**Detaillierte Rückmeldung an den Benutzer:**

Das Snippet gibt detaillierte Informationen über den Kontostatus und mögliche Sicherheitsbedrohungen zurück, was dem Benutzer hilft, sein Verhalten anzupassen und Sicherheitsmaßnahmen zu beachten.

Sicherheits-Tipp:

Starken Passwörtern kommt eine enorme Bedeutung für die Computersicherheit zu. Oft wählen Nutzer jedoch aus Bequemlichkeit einfache Passwörter. Password-Cracker sind Tools, die entwickelt wurden, um diese schwachen Passwörter zu knacken. Systemadministratoren sollten die Benutzer über die Wichtigkeit starker Passwörter aufklären und deren Verwendung sicherstellen. Regelmäßige Passwort-Updates und die Verwendung von Passwort-Prompten sind wichtige Maßnahmen.