

Authentication Cheat Sheet

From OWASP

Contents

- 1 Introduction
- 2 Authentication General Guidelines
 - 2.1 Implement Proper Password Strength Controls
 - 2.1.1 Password Length
 - 2.1.2 Password complexity
 - 2.2 Implement Secure Password Recovery Mechanism
 - 2.3 Require Current Password for Password Changes
 - 2.4 Utilize Multi-Factor Authentication
 - 2.5 Authentication and Error Messages
 - 2.5.1 Authentication responses
 - 2.5.2 Incorrect responses examples
 - 2.5.3 Correct response example
 - 2.5.4 Error Codes and URL's
 - 2.6 Transmit Passwords Only Over TLS
 - 2.7 Implement Account Lockout
- 3 Session Management General Guidelines
- 4 Related Articles
- 5 Authors and Primary Editors

Introduction

Authentication is the process of verification that an individual or an entity is who it claims to be. Authentication is commonly performed by submitting a user name or ID and one or more items of private information that only a given user should know.

Session Management is a process by which a server maintains the state of an entity interacting with it. This is required for a server to remember how to react to subsequent requests throughout a transaction. Sessions are maintained on the server by a session identifier which can be passed back and forward between the client and server when transmitting and receiving requests. Sessions should be unique per user and computationally very difficult to predict.

For more information on Authentication, please see the OWASP Guide to Authentication page.

Authentication General Guidelines

Implement Proper Password Strength Controls

A key concern when using passwords for authentication is password strength. A "strong" password policy makes it difficult or even improbable for one to guess the password either by using manual or automated means. The following characteristics define strong a strong password:

Password Length

Longer passwords provide a greater combination of characters and consequently make it more difficult for an attacker to guess.

Important applications: Minimum of 6 characters in length.

Critical applications: Minimum of 8 characters in length. (consider multi-factor authentication)

Highly critical applications: Consider multi-factor authentication

Password complexity

Example

Passwords should be checked for the following composition or a variance of such:

- at least: 1 uppercase character (A-Z)
- at least: 1 lowercase character (a-z)
- at least: 1 digit (0-9)
- at least: 1 special character (!"£\$%&...)
- a defined minimum length (e.g. 8 chars)
- a defined maximum length (as with all external input)
- no contiguous characters (e.g. 123abcd)
- not more than 2 identical characters in a row (1111)

Implement Secure Password Recovery Mechanism

It is common for an application to have a mechanism that provides a means for a user to gain access to their account in the event they forget their password. Please see https://www.owasp.org/index.php/Forgot_Password_Cheat_Sheet for details on this feature.

Require Current Password for Password Changes

In order to mitigate account hijacking, it's important to require the current password for an account before updating sensitive account information such as the user's password. Without this countermeasure an attacker may be able to change the user's password through a CSRF or XSS attack, without needing

to know the user's current password. Additionally an attacker may get temporary physical access to a user's browser or by using a session riding attack. Email address changes should similarly be authenticated, since many password recovery systems send password or password reset instructions to the user's email address.

Utilize Multi-Factor Authentication

Multifactor authentication is using more than one of:

- Something you know (account details or passwords)
- Something you have (tokens or mobile phones)
- Something you are (biometrics)

to logon or process a transaction.

Authentication schemes such as One Time Passwords (OTP) implemented using a hardware token can also be key in fighting attacks such as CSRF and client-side malware.

Authentication and Error Messages

Incorrectly implemented error messages in the case of authentication functionality can be used for the purposes of user ID and password enumeration.

An application should respond (both HTTP and HTML) in a generic manner which is not unique to the error condition or authentication failure.

Authentication responses

An application should respond with a generic error message regardless if the user ID or password was incorrect. It should also give no indication to the status of an existing account.

Incorrect responses examples

- "Login for User foo: invalid password"
- "Login failed, invalid user ID"
- "Login failed; account disabled"
- "Login failed; this user is not active"

Correct response example

- "Login failed; Invalid user ID or password"

The correct response does not indicate if the user ID or password is the incorrect parameter and hence inferring a valid user ID.

Error Codes and URL's

The application may return a different HTTP Error code depending on the authentication attempt response. It may respond with a 200 for a positive result and a 403 for a negative result. Even though a generic error page is shown to a user, the HTTP response code may differ which can indicate a signature.

Transmit Passwords Only Over TLS

See: "Transport Layer Protection Cheat Sheet"

http://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet

"The login page and all subsequent authenticated pages must be exclusively accessed over TLS. The initial login page, referred to as the "login landing page", must be served over TLS. Failure to utilize TLS for the login landing page allows an attacker to modify the login form action, causing the user's credentials to be posted to an arbitrary location. Failure to utilize TLS for authenticated pages after the login enables an attacker to view the unencrypted session ID and compromise the user's authenticated session."

Implement Account Lockout

If an attacker is able to guess passwords without the account becoming disabled due to failed authentication attempts, the attacker has an opportunity to continue with a brute force attack until the account is compromised.

Automating brute-force/password guessing attacks on web applications is a trivial challenge. Password lockout mechanisms should be employed that lock out an account if more than a preset number of unsuccessful login attempts are made.

Password lockout mechanisms have a logical weakness. An attacker that undertakes a large numbers of authentication attempts on known account names can produce a result that locks out entire blocks of application users accounts.

Given that the intent of a password lockout system is to protect from brute-force attacks, a sensible strategy is to lockout accounts for a number of hours. This significantly slows down attackers, while allowing the accounts to be open for legitimate users.

See also "Reverse Brute-force" [Reverse_Brute_Force](#)

Session Management General Guidelines

Session management is directly related to authentication. The **Session Management General Guidelines** previously available on this OWASP Authentication Cheat Sheet have been integrated into

the new OWASP Session Management Cheat Sheet (https://www.owasp.org/index.php/Session_Management_Cheat_Sheet) .

Related Articles

OWASP Cheat Sheet Series

- **Authentication Cheat Sheet**
- Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet
- Transport Layer Protection Cheat Sheet
- Cryptographic Storage Cheat Sheet
- Input Validation Cheat Sheet
- XSS (Cross Site Scripting) Prevention Cheat Sheet
- DOM based XSS Prevention Cheat Sheet
- Forgot Password Cheat Sheet
- SQL Injection Prevention Cheat Sheet
- Session Management Cheat Sheet
- HTML5 Security Cheat Sheet
- Web Service Security Cheat Sheet
- Application Security Architecture Cheat Sheet

Draft OWASP Cheat Sheets

- PHP Security Cheat Sheet
- Password Storage Cheat Sheet

Cheat Sheets Project Homepage

- Cheat Sheets

Authors and Primary Editors

Eoin Keary [eoinkeary\[at\]owasp.org](mailto:eoinkeary@owasp.org)

Retrieved from "https://www.owasp.org/index.php/Authentication_Cheat_Sheet"

Category: Cheatsheets

- Powered by MediaWiki OWASP Foundation © 2011

