Forgot Password Cheat Sheet

Introduction

In order to implement a proper user management system, systems integrate a **Forgot Password** service that allows the user to request a password reset.

Even though this functionality looks straightforward and easy to implement, it is a common source of vulnerabilities, such as the renowned user enumeration attack.

The following short guidelines can be used as a quick reference to protect the forgot password service:

- **Return a consistent message for both existent and non-existent accounts.**
- **Ensure that the time taken for the user response message is uniform.**
- **Use a side-channel to communicate the method to reset their password.**
- **Use URL tokens for the simplest and fastest implementation.**
- **Ensure that generated tokens or codes are:**
    - **Randomly generated using a cryptographically safe algorithm.**
    - **Sufficiently long to protect against brute-force attacks.**
    - **Stored securely.**
    - **Single use and expire after an appropriate period.**
- **Do not make a change to the account until a valid token is presented, such as locking out the account**

This cheat sheet is focused on resetting users passwords. For guidance on resetting multifactor authentication (MFA), see the relevant section in the Multifactor Authentication Cheat Sheet.

Forgot Password Service

The password reset process can be broken into two main steps, detailed in the following sections.

Forgot Password Request

When a user uses the forgot password service and inputs their username or email, the below should be followed to implement a secure process:

- Return a consistent message for both existent and non-existent accounts.
- Ensure that responses return in a consistent amount of time to prevent an attacker enumerating which accounts exist. This could be achieved by using asynchronous calls or by making sure that the same logic is followed, instead of using a quick exit method.
- Implement protections against automated submissions such as CAPTCHA, rate-limiting or other controls.
- Employ normal security measures, such as SQL Injection Prevention methods and Input Validation.

User Resets Password

Once the user has proved their identity by providing the token (sent via an email) or code (sent via SMS or other mechanisms), they should reset their password to a new secure one. In order to secure this step, the measures that should be taken are:

- The user should confirm the password they set by writing it twice.
- Ensure that a secure password policy is in place, and is consistent with the rest of the application.
- Update and store the password following secure practices.
- Send the user an email informing them that their password has been reset (do not send the password in the email!).
- Once they have set their new password, the user should then login through the usual mechanism. Don't automatically log the user in, as this introduces additional complexity to the authentication and session handling code, and increases the likelihood of introducing vulnerabilities.
- Ask the user if they want to invalidate all of their existing sessions, or invalidate the sessions automatically.

Methods

In order to allow a user to request a password reset, you will need to have some way to identify the user, or a means to reach out to them through a side-channel.

This can be done through any of the following methods:

- URL tokens.
- PINs
- Offline methods
- Security questions.

These methods can be used together to provide a greater degree of assurance that the user is who they claim to be. No matter what, you must ensure that a user always has a way to recover their account, even if that involves contacting the support team and proving their identity to staff.

General Security Practices

It is essential to employ good security practices for the reset identifiers (tokens, codes, PINs, etc.). Some points don't apply to the offline methods, such as the lifetime restriction. All tokens and codes should be:

- Generated cryptographically secure random number generator.
    - It is also possible to use JSON Web Tokens (JWTs) in place of random tokens, although this can introduce additional vulnerability, such as those discussed in the JSON Web Token Cheat Sheet.
- Long enough to protect against brute-force attacks.
- Linked to an individual user in the database.
- Invalidated after they have been used.
- Stored in a secure manner, as discussed in the Password Storage Cheat Sheet.

URL Tokens

URL tokens are passed in the query string of the URL, and are typically sent to the user via email. The basic overview of the process is as follows:

1. Generate a token to the user and attach it in the URL query string.
2. Send this token to the user via email.
3. Don't rely on the Host header while creating the reset URLs to avoid Host Header Injection attacks. The URL should be either be hard-coded, or should be validated against a list of trusted domains.
4. Ensure that the URL is using HTTPS.
5. The user receives the email, and browses to the URL with the attached token.
6. Ensure that the reset password page adds the Referrer Policy tag with the `noreferrer` value in order to avoid referrer leakage.
7. Implement appropriate protection to prevent users from brute-forcing tokens in the URL, such as rate limiting.
8. If required, perform any additional validation steps such as requiring the user to answer security questions.
9. Let the user create a new password and confirm it. Ensure that the same password policy used elsewhere in the application is applied.

Note: URL tokens can follow on the same behavior of the PINs by creating a restricted session from the token. Decision should be made based on the needs and the expertise of the developer.

PINs

PINs are numbers (between 6 and 12 digits) that are sent to the user through a side-channel such as SMS.

1. Generate a PIN.
2. Send it to the user via SMS or another mechanism.
3. Breaking the PIN up with spaces makes it easier for the user to read and enter.
4. The user then enters the PIN along with their username on the password reset page.
5. Create a limited session from that PIN that only permits the user to reset their password.
6. Let the user create a new password and confirm it. Ensure that the same password policy used elsewhere in the application is applied.

Offline Methods

Offline methods differ from other methods by allowing the user to reset their password without requesting a special identifier (such as a token or PIN) from the backend. However, authentication still needs to be conducted by the backend to ensure that the request is legitimate. Offline methods provide a certain identifier either on registration, or when the user wishes to configure it.

These identifiers should be stored offline and in a secure fashion (*e.g.* password managers), and the backend should properly follow the general security practices. Some implementations are built on hardware OTP tokens, certificates, or any other implementation that could be used inside of an enterprise. These are out of scope for this cheat sheet.

Backup Codes

Backup codes should be provided to the user upon registering where the user should store them offline in a secure place (such as their password manager). Some companies that implement this method are Google, GitHub, and Auth0.

While implementing this method, the following practices should be followed:

- Minimum length of 8 digits, 12 for improved security.
- A user should have multiple recovery codes at any given time to ensure that one of them works (most services provide the user with ten backup codes).
- A process should be implemented to allow the user to invalidate all existing recovery codes, in case they are compromised by a third party.

- Rate limiting and other protections should be implemented to prevent an attacker from brute-forcing the backup codes.

Security Questions

Security questions should not be used as the sole mechanism for resetting passwords due to their answers frequently being easily guessable or obtainable by attackers. However, they can provide an additional layer of security when combined with the other methods discussed in this cheat sheet. If they are used, then ensure that secure questions are chosen as discussed in the Security Questions cheat sheet.

Account Lockout

Accounts should not be locked out in response to a forgotten password attack, as this can be used to deny access to users with known usernames. For more details on account lockouts, see the Authenication Cheat Sheet.