Authentication

Authentication vulnerabilities can allow attackers to gain access to sensitive data and functionality. They also expose additional attack surface for further exploits

- The most common authentication mechanisms used by websites.
- Potential vulnerabilities in these mechanisms.
- Inherent vulnerabilities in different authentication mechanisms.
- Typical vulnerabilities that are introduced by their improper implementation.
- How you can make your own authentication mechanisms as robust as possible.

Three main types of authentication:

- Something you **know**, such as a password
- Something you **have**, This is a physical object such as a mobile phone or security token.
- Something you **are** or do. For example, your biometrics or patterns of behavior

Difference between authentication and authorization:

Authentication is the process of verifying that a user is who they claim to be. Authorization involves verifying whether a user is allowed to do something.

How do authentication vulnerabilities arise?

- The authentication mechanisms are weak because they fail to adequately protect against brute-force attacks
- Logic flaws or poor coding in the implementation allow the authentication mechanisms to be bypassed entirely by an attacker. This is sometimes called "broken authentication"

Vulnerabilities in password-based login

For websites that adopt a password-based login process, users either register for an account themselves or they are assigned an account by an administrator. This account is associated with a unique username and a secret password, which the user enters in a login form to authenticate themselves

the mere fact that they know the secret password is taken as sufficient proof of the user's identity. Consequently, the security of the website would be compromised if an attacker is

able to either obtain or guess the login credentials of another user.

Brute-force attacks

A brute-force attack is when an attacker uses a system of trial and error to guess valid user credentials.

Username enumeration

While attempting to brute-force a login page, you should pay particular attention to any differences in:

Status codes: During a brute-force attack, the returned HTTP status code is likely to be the same for the vast majority of guesses because most of them will be wrong. If a guess returns a different status code, this is a strong indication that the username was correct.

Error messages: Sometimes the returned error message is different depending on whether both the username AND password are incorrect or only the password was incorrect.

Response times: If most of the requests were handled with a similar response time, any that deviate from this suggest that something different was happening behind the scenes. This is another indication that the guessed username might be correct.

Flawed brute-force protection

The two most common ways of preventing brute-force attacks are:

- Locking the account that the remote user is trying to access if they make too many failed login attempts
- Blocking the remote user's IP address if they make too many login attempts in quick succession

Account locking

One way in which websites try to prevent brute-forcing is to lock the account if certain suspicious criteria are met, usually a set number of failed login attempts. Just as with normal login errors, responses from the server indicating that an account is locked can also help an attacker to enumerate usernames.Locking an account offers a certain amount of protection against targeted brute-forcing of a specific account. However, this approach fails to adequately prevent brute-force attacks in which the attacker is just trying to gain access to any random account they can.

the following method can be used to work around this kind of protection:

- Establish a list of candidate usernames that are likely to be valid.
- Decide on a very small shortlist of passwords that you think at least one user is likely to have.
- Using a tool such as Burp Intruder, try each of the selected passwords with each of the candidate usernames.

User rate limiting

Another way websites try to prevent brute-force attacks is through user rate limiting. In this case, making too many login requests within a short period of time causes your IP address to be blocked.

IP can only be unblocked in one of the following ways:

- Automatically after a certain period of time has elapsed
- Manually by an administrator
- Manually by the user after successfully completing a CAPTCHA

It is sometimes also possible to bypass this defense if you can work out how to guess multiple passwords with a single request.

HTTP basic authentication

In HTTP basic authentication, the client receives an authentication token from the server, which is constructed by concatenating the username and password, and encoding it in Base64.

token is stored and managed by the browser, which automatically adds it to the `Authorization` header of every subsequent request as follows:

```http
Authorization: Basic base64(username:password)
```

Vulnerabilities in multi-factor authentication

Verifying biometric factors is impractical for most websites. However, it is increasingly common to see both mandatory and optional two-factor authentication (2FA) based on **something you know** and **something you have**. This usually requires users to enter both a traditional password and a temporary verification code from an out-of-band physical device in their possession.

Poorly implemented two-factor authentication can be beaten, or even bypassed entirely, just as single-factor authentication can.

Two-factor authentication tokens

It is also common for websites to use a dedicated mobile app, such as Google Authenticator,

On the other hand, some websites send verification codes to a user's mobile phone as a text message.

Bypassing two-factor authentication

If the user is first prompted to enter a password, and then prompted to enter a verification code on a separate page, the user is effectively in a "logged in" state before they have entered the verification code.

In this case, it is worth testing to see if you can directly skip to "logged-in only" pages after completing the first authentication step.

Flawed two-factor verification logic

Sometimes flawed logic in two-factor authentication means that after a user has completed the initial login step, the website doesn't adequately verify that the same user is completing the second step.

the user logs in with their normal credentials in the first step as follows

```http
POST /login-steps/first HTTP/1.1
Host: vulnerable-website.com
...
username=carlos&password=qwerty
```

They are then assigned a cookie that relates to their account

```http
HTTP/1.1 200 OK
Set-Cookie: account=carlos
GET /login-steps/second
HTTP/1.1 Cookie: account=carlos
```

When submitting the verification code, the request uses this cookie to determine which account the user is trying to access:

```http
POST /login-steps/second HTTP/1.1
Host: vulnerable-website.com
```

```
Cookie: account=carlos
...
verification-code=123456
```

In this case, an attacker could log in using their own credentials but then change the value of the `account` cookie to any arbitrary username when submitting the verification code.

Brute-forcing 2FA verification codes

As with passwords, websites need to take steps to prevent brute-forcing of the 2FA verification code. This is especially important because the code is often a simple 4 or 6-digit number. Without adequate brute-force protection, cracking such a code is trivial.

Some websites attempt to prevent this by automatically logging a user out if they enter a certain number of incorrect verification codes. This is ineffective in practice because an advanced attacker can even automate this multi-step process by creating macros for Burp Intruder.

Vulnerabilities in other authentication mechanisms

Users can typically change their password or reset their password when they forget it. These mechanisms can also introduce vulnerabilities that can be exploited by an attacker.

Resetting user passwords

Some users will forget their password, so it is common to have a way for them to reset it. As the usual password-based authentication is obviously impossible in this scenario, websites have to rely on alternative methods to make sure that the real user is resetting their own password.

Few different ways that this feature is commonly implemented

Sending passwords by email :

Some websites generate a new password and send this to the user via email.

Resetting passwords using a URL :

A more robust method of resetting passwords is to send a unique URL to users that takes them to a password reset page. Less secure implementations of this method use a URL with an easily guessable parameter to identify which account is being reset

some websites fail to also validate the token again when the reset form is submitted. In this case, an attacker could simply visit the reset form from their own account, delete the token, and leverage this page to reset an arbitrary user's password

Password Reset Poisoning

If the URL in the reset email is generated dynamically, this may also be vulnerable to password reset poisoning. In this case, an attacker can potentially steal another user's token and use it change their password.

Changing user passwords

changing your password involves entering your current password and then the new password twice. These pages fundamentally rely on the same process for checking that usernames and current passwords match as a normal login page does. Therefore, these pages can be vulnerable to the same techniques.

Authentication vulnerabilities

authentication vulnerabilities are easy to understand. However, they are usually critical because of the clear relationship between authentication and security.

- The most common authentication mechanisms used by websites.
- Potential vulnerabilities in these mechanisms.
- Inherent vulnerabilities in different authentication mechanisms.
- Typical vulnerabilities that are introduced by their improper implementation.
- How you can make your own authentication mechanisms as robust as possible.

How to secure your authentication mechanisms

- **Take care with user credentials**
- **Don't count on users for security**
- **Prevent username enumeration**
- **Implement robust brute-force protection**
- **Triple-check your verification logic**
- **Don't forget supplementary functionality**
- **Implement proper multi-factor authentication**