Overview of Authentication Methods

During the authentication phase, the entity who wants to authenticate sends an `identification string` that could be an ID, a username, email, along with additional data. The most common type of data that an authentication process requires to be sent together with the identification string is a password string. That being said, the type of additional data can vary between implementations.

Multi-Factor Authentication

`Multi-Factor Authentication`, commonly known as `MFA` (or `2FA` when there are just two factors involved), can result in a much more robust authentication process.

Factors are separated into three different domains:

- `something the user knows, for example, a username or password`
- `something the user has, like a hardware token`
- `something the user is, usually a biometric fingerprint`

When an authentication process requires the entity to send data that belongs to more than one of these domains, it should be considered an MFA process. Single Factor Authentication usually requires something the user `knows`:

- `Username + Password`

It is also possible for the requirement to be only something the user `has`.

Think about a corporate badge or a train ticket. Passing through a turnstile often requires you to swipe a badge that grants you access. In this case, you need no PIN, no identification string, or anything else but a card. This is an edge case because company badges, or train multi-cards, are often used to match a specific user by sending an ID. By swiping, both authorization and a form of authentication are performed.

Form-Based Authentication

The most common authentication method for web applications is `Form-Based Authentication` (FBA). The application presents an HTML form where the user inputs their username and password, and then access is granted after comparing the received data against a backend. After a successful login attempt, the application server creates a session tied to a unique key (usually stored in a cookie). This unique key is passed between the client and the web application on every subsequent communication for the session to be maintained.

Some web apps require the user to pass through multiple steps of authentication. For example, the first step requires entering the username, the second the password, and the third a `One-time Password` (`OTP`) token. An `OTP` token can originate from a hardware device or mobile application that generates passwords. One-time Passwords usually last for a limited amount of time, for example, 30 seconds, and are valid for a single login attempt, hence the name one-time.

It should be noted that multi-step login procedures could suffer from business logic vulnerabilities. For example, Step-3 might take for granted that Step-1 and Step-2 have been completed successfully.

HTTP Based Authentication

Many applications offer `HTTP-based` login functionality. In these cases, the application server can specify different authentication schemes such as Basic, Digest, and NTLM. All HTTP authentication schemes revolve around the `401` status code and the `WWW-Authenticate` response header and are used by application servers to challenge a client request and provide authentication details (Challenge-Response process).

When using HTTP-based authentication, the `Authorization header` holds the authentication data and should be present in every request for the user to be authenticated.

From a network point of view, the abovementioned authentication methods could be less secure than FBA because every request contains authentication data. For example, to perform an HTTP basic auth login, the browser encodes the username and password using base64. The `Authorization header` will contain the base64-encoded credentials in every request. Therefore, an attacker that can capture the network traffic in plaintext will also capture credentials. The same would happen if FBA were in place, just not for every request.

Below is an example of the header that a browser sends to fulfill basic authentication.

HTTP Authentication Header

HTTP Authentication Header

```
GET /basic_auth.php HTTP/1.1
Host: brokenauth.hackthebox.eu
Cache-Control: max-age=0
Authorization: Basic YWRtaW46czNjdXIzcDQ1NQ==
```

The authorization header specifies the HTTP authentication method, Basic in this example, and the token: if we decode the string:

HTTP Authentication Header

```
YWRtaW46czNjdXIzcDQ1NQ==
```

as a base64 string, we'll see that the browser authenticated with the credentials: `admin:s3cur3p455`

Digest and NTLM authentication are more robust because the data transmitted is hashed and could contain a nonce, but it is still possible to crack or reuse a captured token.

Other Forms of Authentication

While uncommon, it is also possible that authentication is performed by checking the source IP address. A request from localhost or the IP address of a well-known/trusted server could be considered legitimate and allowed because developers assumed that nobody but the intended entity would use this IP address.

Modern applications could use third parties to authenticate users, such as SAML. Also, `APIs` usually require a specific authentication form, often based on a multi-step approach.

Attacks against API authentication and authorization, Single Sign-On, and OAuth share the same foundations as attacks against classic web applications. Nevertheless, these topics are pretty broad and deserve their own module.

Login Example

A typical scenario for home banking authentication starts when an e-banking web application requests our ID, which could be a seven-digit number generated by the e-banking web application itself or a username chosen by the user. Then, on a second page, the application requests a password for the given ID. On a third page, the user must provide an OTP generated by a hardware token or received by SMS on their mobile phone. After providing the authentication details from the above two factors (2FA case), the e-banking web application checks if the ID, password, and OTP are valid.

Previous Next

Mark Complete & Next

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left