OWASP TOP 10

A2- Broken Authentication and Session Management

CYBER SECURITY &
PRIVACY FOUNDATION

A2-Broken Authentication and Session Management

The Application functions that are related to the authentication and the session management are most often not implemented correctly, this allows the attackers to compromise the passwords, keys, or session tokens, or to even exploit other implementation flaws to assume other user's identities.

Session Management

- HTTP/S protocol generally do not provide tracking of a user's session.
- Most of the times developers make the mistake of inventing their own session tracking mechanism.

A Session ID

- Which is unique to the User
- Which is used for only one authenticated session
- Which is generated by the server
- The user is expected to send back the same ID in the next request.

Session Hijacking

- Session ID is disclosed or is guessed.
- An attacker using the same session ID has the same privileges as the real user.
- Especially useful to an attacker if the session is privileged.
- Allows initial access to the web application to be combined with other attacks.

Broken Authentication Management

Most of the times even the valid authentication schemes can be undermined by flawed account management functions such as:

- Account update.
- Change Password.
- Password Forget Recovery or Reset.

Causes:

- Credentials can be guessed or overwritten through weak account management functions .
- The Credentials of the user's authentication aren't protected during the time of storing using hashing or any encryption mechanism.
- Credentials are sent over unencrypted connections.
- User's sessions or authentication tokens aren't properly invalidated during the time of logging out.
- Session IDs being exposed in the URL, hence possible of URL Rewriting.

Consequences

- The data an attacker can compromise depends on various factors like his abilities, the system design and the vulnerabilities in the application's authentication and session management system.
- Such flaws may allow some or even all accounts to be compromised. Once succeeded, the attacker can do anything with the victim's profile.
- Removing the user's online identity (if the user has linked multiple accounts using email IDs as data storage, using information for one service provider as a recovery information for another to compromise more.)
- Gain access to the data contained in the victim's system as user accounts usually have privileges on the database.

Session Management: Protection

- Use of long complex random session ID which cannot be guessed.
- Protecting the transmission and storage of the Session ID to prevent disclosure and hijacking.
- A URL query string should not be used for Session ID or any User/Session information.
- Entire session should be transmitted via HTTPS to prevent disclosure of the session ID.
- Protect any session information transmitted to/from the client.
- Session ID should expire and/or time-out **on the Server** when the user is idle or on logout.
- Regenerating a new session upon successful authentication or privilege level change would be a better choice.

Broken Authentication and Session Management: Protection

- **Forgotten Password Controls** - if forgotten passwords are emailed to users, they should be required to re-authenticate whenever they attempt to change their email address.
- **Password Strength** –developers should take care that the user's password should require at least 7 characters, with letters, numbers, and special characters both upper case and lower case.
- **Password Change Controls** - require users to provide both old and new passwords.
- **Man-in-the-middle attacks** - are still possible with SSL if users disable or ignore warnings about invalid SSL certificates.
- **Replay attacks -** Transformations such as hashing on the client side provide little protection as the hashed version can simply be intercepted and retransmitted so that the actual plain text password is not needed.

Broken Account and Session Management: Protection cont'd

- **Password Storage** – developers should ensure that never store passwords in plain text. Passwords should always be stored in either hashed (preferred) or encrypted form.
- **Protecting Credentials in Transit** - to prevent "man-in-the-middle" attacks the entire authenticated session / transaction must be encrypted.