Honors Contract Research Paper
Study of Authentication for Web Applications

Saniah Safat

The University of Texas at Arlington

CSE 4344: COMPUTER NETWORKS

Dr. Yonghe Liu

December 4, 2023

Introduction

In today's ever-expanding digital landscape, where online interactions influence every aspect of our everyday lives, ensuring users' secure and verifiable identities is critical. Web authentication is the first defense against internet-based services' unwanted access and data breaches.

Brief history and evolution of web authentication

Computer authentication originated in the 1960s, when passwords were initially saved in a database, and the user's password entering had to match what was in storage. Even though authentication has developed since the 1960s and 1970s with the introduction of encryption, passwords remain the most common and widely used. Dynamic Passwords were created some 40 years ago when it was apparent that passwords were not secure for digital verification. When computers' processing power increased, creating and using one-time passwords (OTP) as single-factor or even multi-factor authentication became possible.

With the growing use of the internet by the end of the 1990s and the beginning of the 2000s, websites such as e-commerce and financial institutions were handling an unprecedented volume of sensitive data. The increased use of multi-factor authentication (MFA) has resulted from the demand for stronger authentication. The combination of two or more criteria boosted the authentication's strength. Today, besides the primary password-based method and multi-factor authentication, many other web authentication methods, such as token-based, cookie-based, biometric authentication, WebAuthn, OAuth, and OpenID Connect, are available.

Overview of the importance of authentication in web applications

There are various reasons why website authentication is necessary. For instance, it ensures user security by safeguarding essential data and preventing illegal access. Authentication works as a gatekeeper by confirming user identities, allowing only authorized individuals to access personal accounts and sensitive data. Furthermore, website authentication is critical in instilling trust and confidence in users. Users are more likely to interact with the website, share personal information, and complete transactions when they feel secure.

In addition, website authentication assists organizations in meeting data protection legislation and industry standards. Another essential advantage of website authentication is its capacity to reduce the dangers of weak passwords and password-related assaults. Authentication also ensures that data exchanged between users and websites is confidential and secure. Authentication prevents eavesdropping, data tampering, and interception by malicious actors through encryption and secure communication protocols.

Authentication Mechanisms and Security Dynamics

Authentication Methods

1. **Password-based authentication**
   Users must enter a unique combination of characters (password) associated with their account while using this technique.

   Possible drawbacks:

   - Weak passwords or password reuse can result in security flaws.
   - Passwords can be easily guessed or stolen, making them vulnerable to brute-force attacks.
   - Users may need help remembering complex passwords, leading to insecure actions such as writing them down.

2. **Multi-factor authentication (MFA)**
   To validate a user's identity, MFA combines the use of a password with another form of verification such as an OTP, knowledge-based questions, or the use of an authenticator app,

   Possible drawbacks:

   - Some MFA solutions, such as SMS-based verification, are subject to SIM swapping or code eavesdropping.
   - Implementing complexity and compatibility difficulties may develop when incorporating MFA into existing systems.
   - Adding additional authentication factors, such as hardware tokens or biometrics, can be costly and time-consuming.

3. **Token-based authentication**
   Token-based authentication validates user identity using physical (e.g., smart cards) or virtual (e.g., software tokens) tokens. During authentication, users present these tokens, which serve as unique identifiers. Physical tokens are physical objects, whereas software tokens are generated online. Access is permitted if the provided token matches the records in the system, boosting security beyond typical credentials.

Possible drawbacks:

- If tokens are lost or stolen, unauthorized parties can use them to get access.
- To prevent misuse, token distribution and management must be secure.
- Token-based authentication may be inconvenient for users because it requires carrying or holding a physical token.

4. **Cookie-based authentication**
   Users log in using their credentials, requiring the server to generate a unique session ID upon successful verification. This session ID is saved in the user's browser as a cookie. The client presents the session ID for subsequent requests, which the server compares to its data. Access is granted if they match. Logout actions terminate client and server sessions, terminating user access as soon as requested. This basic solution manages user sessions in web applications efficiently.

   Possible drawbacks:

   - Scaling becomes an issue while handling multiple users.
   - Handles sensitive information about users, making it a target for attackers.
   - XSS and CSRF attacks can exploit cookies.

5. **Biometric authentication**
   Biometric authentication confirms identity by utilizing distinguishing biological features such as fingerprints, facial features, or iris patterns. It evaluates unique physiological or behavioral factors rather than traditional credentials, providing a high level of security. Fingerprint scanning, facial recognition, and iris scanning are common ways of delivering robust identification verification.

   Possible drawbacks:

   - Once compromised, biometric data cannot be updated in the same way that a password can. It raises worries about privacy and the security of biometric data stored in the cloud.
   - False negatives or positives may occur, leading to denied or illegal access.
   - Because biometric authentication methods may require specialized hardware or software support, their availability and compatibility may be limited.

6. **Single Sign-On (SSO)**
   SSO lets users authenticate once and access numerous systems or applications without re-entering their credentials.

   Possible drawbacks:

   - A compromised SSO account might grant access to several systems, thus magnifying the effect of a security breach.

   - Implementation complexity can be increased, particularly when integrating several systems or dealing with multiple authentication protocols.

   - Using a centralized authentication method creates a single point of failure.

Security Threats

1. **Brute force attacks**
   Brute force attacks involve systematically attempting all possible password combinations until the correct one is found. Attackers use automated methods to guess passwords, taking advantage of weak or easily guessable credentials. There are various types of brute force attack methods, such as simple brute force attacks, dictionary attacks, hybrid brute force attacks, reverse brute force attacks and credential stuffing. To prevent brute force attacks, implement account lockout policies, utilize strong password policies, and use multi-factor authentication.

2. **Man-in-the-middle attacks**
   An unauthorized entity intercepts and potentially modifies communication between two parties without their knowledge in man-in-the-middle attacks. This can result in data interception, illegal access, or malicious content injection. To prevent eavesdropping and tampering, employ encryption technologies such as HTTPS, secure channels such as VPNs, and validate digital certificates.

3. **Session hijacking**
   Session hijacking happens when an attacker takes a user's session identifier, allowing the attacker unauthorized access to the victim's active session. This can result in impersonation and unauthorized activities on behalf of the victim.
   Use secure session management methods, HTTPS to encrypt session data, and routinely regenerate session identifiers to reduce attackers' window of opportunity.

4. **Credential stuffing**

Credential stuffing is gaining unauthorized access to user accounts on numerous sites by using previously leaked usernames and passwords. Attackers take advantage of users' proclivity to reuse credentials across multiple sites. Encourage users to use different passwords for each account, keep an eye out for unusual login habits, employ multi-factor authentication, and regularly audit and change password policy.

Authentication Protocols

OAuth

OAuth (Open Authorization) is an open standard protocol for authorizing an application to use user information. In general, it allows a third-party application access to user-related information such as name, DOB, email, or other required data from an application such as Facebook, Google, and others without requiring the user to provide the third-party app with the user's password.

Some of the key functionalities of OAuth are:

- Authorization: Users can grant permissions to third-party applications, granting them access to designated resources on a service (Provider) without directly disclosing their credentials.

- Token-Based Access: Rather than using traditional usernames and passwords, OAuth uses access tokens to provide temporary and limited access to the defined resources. This approach improves security by reducing exposure.

- Delegated Access: OAuth allows users to grant third-party applications access to their resources without jeopardizing the security of their login credentials.

OAuth is used in a variety of circumstances, demonstrating its adaptability. It is widely used in social networking platforms, allowing users to connect their accounts to third-party programs such as social media dashboards or analytics tools, hence facilitating data access. OAuth is essential in Single Sign-On scenarios, allowing users to log in to different services using credentials from a single authentication source. Furthermore, the OAuth Authorization Code procedure is recommended for mobile applications. Because mobile apps are considered public clients, the Authorization Code Flow with the PKCE extension is highly recommended. Additionally, web services such as Google, Facebook, and GitHub use OAuth to ensure secure and restricted access to their APIs by third-party applications, improving interoperability.

Securing an OAuth implementation necessitates the application of essential best practices.

- Tokens with short expiration periods improve security by reducing the dangers associated with long token lifetimes.
- Preventing potential misuse by clearly specifying and requesting only appropriate rights (scopes) tailored to specific use cases.

- Using secure storage technologies to protect access tokens on the client side avoids illegal access or leakage.

- Integrating refresh tokens simplifies access renewal without requiring user reauthentication, increasing user experience while maintaining stringent security standards.

OpenID Connect

OpenID Connect is a layer of identity built on top of the OAuth 2.0 protocol. Whereas OAuth 2.0 allows a service user to grant a third-party application access to their data hosted by the service without disclosing their credentials to the application, OpenID Connect allows a third-party application to obtain a user's identity information managed by a service. This feature allows developers to easily authenticate users across websites and apps without having to own and manage their passwords. Google Plus Sign-In is one platform that developers may utilize to provide a secure social login experience for their users. It is built on OpenID Connect and OAuth 2.0.

Besides, some other features of OpenID connect enable to carry out Single Log Out (SLO), where a user can logout and get all sessions terminated without needing to logout individually from each client. This mechanism ensures there is no active session left from a SSO session that can be targeted for a hijack.

SAML (Security Assertion Markup Language)

SAML stands for Security Assertion Markup Language, an authentication data format based on XML that offers authorization between an identity provider and a service provider. While OAuth is a protocol for authorization, SAML (Security Assertion Markup Language) is a federated authentication mechanism for enterprise security. It is intended for usage in single sign-on (SSO) settings, where a user can log in to several linked systems and services with a single ID and password.

SAML was designed by the technology industry to streamline the authentication process when consumers required to access different, independent web apps across domains. Prior to SAML,

single sign-on (SSO) was possible but required cookies that could only be used within the same domain. This is accomplished by centralizing user authentication using an identity provider. Web applications can then utilize SAML to offer access to their users via the identity provider. This SAML authentication method eliminates the need for users to remember numerous usernames and passwords. It also benefits service providers by increasing the security of their own platform, especially by eliminating the need to retain (often weak and insecure) passwords and dealing with forgotten password difficulties.

Emerging Technologies

WebAuthn

WebAuthn is an API standard that enables servers, applications, websites, and other systems to manage and validate registered users using password-less authentication methods such as biometric or possession-based device authenticators. This protocol, developed by the World Wide Web Consortium (W3C), covers common web browsers such as Chrome, Microsoft Edge, Firefox, Safari, and their mobile equivalents. WebAuthn secures the registration, management, and authentication of devices and accounts with the relevant servers by utilizing public-key cryptography.

The first significant advantage of WebAuthn is increased security. Going passwordless protects both enterprises and individual customers against password-based harm. WebAuthn is also supported by a wide range of common browsers, operating systems, and devices, so it is not limited to a small set of systems. This solution also offers consumers a better overall user experience, including ease (they can log in faster without a password) and choice (they can authenticate using various methods, devices, and operating systems). WebAuthn is also adaptable, allowing users and companies to choose between strong single-factor logins and multi-factor authentications that can be tailored to the system being accessed.

FIDO (Fast Identity Online)

FIDO (Fast IDentity Online) is a collection of open, standardized authentication methods designed to eventually replace passwords, which are often useless and antiquated in terms of security. To secure user authentication, FIDO protocols employ basic public key cryptography techniques. All messages are encrypted, and private keys never leave users' devices, reducing the possibility of them being discovered during transmission. Furthermore, if biometric information is utilized to authenticate, it is also saved on users' devices, making authentication processes more robust and more secure.

To implement FIDO in web applications, FIDO client libraries must be integrated into the front end, allowing connection with FIDO-compliant devices such as security keys or biometric sensors. Web applications register users using the Web Authentication API (WebAuthn) by producing unique public-private key pairs saved on the server and the FIDO device, respectively. During authentication, the FIDO device verifies the user's identity using biometrics or a PIN by signing a challenge, which the server validates using the saved public key. Cross-browser compatibility, fallback options for non-FIDO users, a seamless user experience, and tight security standards are critical factors for a successful FIDO deployment. This method improves authentication security, lowers dependency on passwords, and contributes to a more secure online environment.

Web Authentication Effectivity

Web authentication effectiveness varies depending on use case, necessary security level, and user preferences. There is no single authentication mechanism that is generally "most effective" in all cases. To improve security, it is sometimes recommended to use a combination of different authentication factors or to use a layered approach.

Any authentication method's efficiency is partly determined by its execution, user awareness, and overall security measures in place. When deciding on the best authentication method for a web service or system, consider user experience, scalability, compatibility, and the specific security landscape.

Finally, companies should undertake a thorough risk assessment and examine their specific needs to choose the most effective web authentication technique that balances security, usability, and practicality for their specific circumstances.

Website authentication development precautions

Avoiding harmful behaviors in the development landscape of website authentication is critical for building a durable and trustworthy system. To improve security, avoid practices like storing passwords in plain text and instead use effective hashing algorithms with salt values. Enforcing strict password regulations, thoroughly checking user inputs, and putting safeguards in place to prevent brute-force attacks all contribute to a more robust authentication process. Furthermore, a safe authentication system must communicate login credentials securely through protocols such as HTTPS, eliminate hard-coded credentials in source code, and provide robust password recovery procedures. Developers should set up robust logging and monitoring systems to improve detection capabilities. Regular updates and patch management, combined with routine security audits and testing, strengthen the authentication system's resistance to potential flaws.

By avoiding these traps, developers assure a reinforced authentication system that emphasizes security while also instilling user trust in the digital experience.

Conclusion

Web authentication has evolved from the fundamental username-password combinations to cutting-edge technology such as biometrics and public key cryptography, adapting to the dynamic difficulties given by an interconnected world. This paper investigates the complicated fabric of web authentication, delving into its historical roots, important approaches, and the critical role it plays in safeguarding sensitive information, preserving user privacy, and enhancing the robustness of digital ecosystems.

References

Blog, I. (2023, January 11). *What is authentication? [the essentials of digital authentication]*. https://www.incognia.com/the-authentication-reference/what-is-authentication

IEMLabs, I. (2023, May 18). *Web authentication : What is it and why is it important in web development?*. IEMLabs Blog. https://iemlabs.com/blogs/web-authentication-what-is-it-and-why-is-it-important-in-web-development/

Roy, W. by: S. (2023, October 27). *Popular authentication methods for web apps*. Baeldung on Computer Science. https://www.baeldung.com/cs/authentication-web-apps

GeeksforGeeks. (2023, November 1). *What is OAuth (Open Authorization) ?*. GeeksforGeeks. https://www.geeksforgeeks.org/what-is-oauth-open-authorization/

An overview of OAUTH2 concepts and use cases. Ory. (n.d.). https://www.ory.sh/docs/oauth2-oidc/overview/oauth2-concepts

OAuth Best Practices. Square Developer. (n.d.). https://developer.squareup.com/docs/oauth-api/best-practices

Cobb, M., & Mann, S. (2020, February 7). *What is oauth and how does it work?*. App Architecture. https://www.techtarget.com/searchapparchitecture/definition/OAuth

SAML explained in plain English. OneLogin. (n.d.). https://www.onelogin.com/learn/saml#:~:text=SAML%20is%20an%20open%20standard,the%20service%20provider%20(SP).

Magnusson, A. (2023, February 7). *What is WebAuthn? web authentication explained*. StrongDM. https://www.strongdm.com/blog/webauthn

Curity. (2020, February 25). *OpenID Connect Single Logout*. Curity Identity Server. https://curity.io/resources/learn/openid-connect-logout/

Passwordless authentication. Ping Identity. (n.d.). https://www.pingidentity.com/en/resources/identity-fundamentals/authentication/passwordless-authentication.html

What is a brute force attack?: Definition, Types & How It Works. Fortinet. (n.d.). https://www.fortinet.com/resources/cyberglossary/brute-force-attack

What is a man in the middle (MITM) attack?. SentinelOne. (2023, September 4). https://www.sentinelone.com/cybersecurity-101/what-is-a-man-in-the-middle-mitm-

attack/?utm_source=gdn-paid&utm_medium=paid-display&utm_campaign=nam-pmax-brand-ppc&utm_term=&campaign_id=19502097988&ad_id=&gad_source=1&gclid=Cj0KCQiAyKurBhD5ARIsALamXaEjneJXC3BWwb2pzASHFmcmy39DAKiqYsDDkr2-6SKbhQuycvjeqtAaArxYEALw_wcB

Venafi. (n.d.). *What is session hijacking & how does it work?* https://venafi.com/blog/what-session-hijacking/

Credential stuffing vs. brute force attacks - Cloudflare. (n.d.). https://www.cloudflare.com/learning/bots/what-is-credential-stuffing/