**Security protocols for web application**

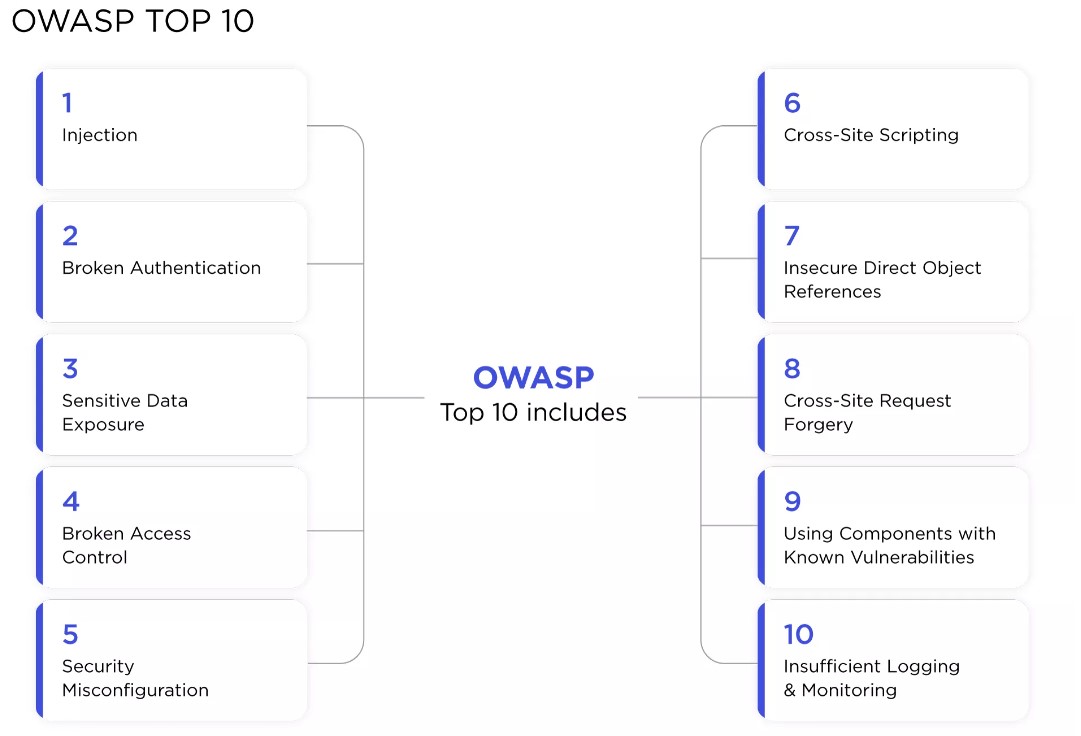
**Definition:**

Web application security is a central component of any web-based business. The global nature of the Internet exposes web properties to attack from different locations and various levels of scale and complexity. Web application security deals specifically with the security surrounding websites, web applications and web services such as APIs.

# Common Web Application Security Vulnerabilities

## OWASP Top 10

The Open Web Application Security Project (OWASP) is an open community of engineers and security IT professionals whose goal is to make the web safer for users and other entities.



Injection Flaws:

Injection flaws are when an attacker uses unfiltered and often [malicious data to attack databases or directories connected to your web apps](https://owasp.org/www-community/Injection_Flaws). Two common injection attacks often get used. First, SQL injection gets used to attack your databases. Second, LDAP injection gets used to attack directories.

**How to prevent injection flaws:**

There are ways we can help to prevent injection attacks. Adding filters to your inputs is the best defense. With SQL databases, we can first use prepared statements that can help prevent attackers from manipulating queries. Second, with LDAP injections, we can use protocols like escape variables to prevent characters used with injection attacks from being passed to manipulate the directory.

### Broken Authentication

Authentication helps apps identify and validate users. Therefore broken authentication can allow attackers to access and have the same permissions as the targeted user, creating severe web app vulnerabilities. Issues with authentication can give an attacker unfettered access to your data and wreak havoc on your web application.

**How to prevent broken authentication vulnerabilities:**

Protecting your web application from authentication vulnerabilities can be a simple fix. Using multi-factor authentication can help verify the correct user. Creating strong passwords with periodic password updates can keep from common password use. Finally, properly configuring timeouts and password security within your database will prevent authentication issues.

### Sensitive Data Exposure

Sensitive data gets transported or stored without any encryption or other protection, leaving information vulnerable to various attacks.

**How to prevent sensitive data exposure:**

Preventing the exposure of your sensitive data is vital to the security of your app. Due to data vulnerabilities in motion, HTTPS, and perfect forward secrecy (PFS), ciphers need to get implemented for incoming data to your site. Disabling data caching that may store sensitive information is another way to help protect data.

### Missing Function Level Access Control

When server-side authorization is misconfigured, broken, or missing, vulnerabilities will occur that can leave your back-end open to attacks.

**How to prevent missing function level access control vulnerabilities:**

Fixing this flaw is simple. All server-side authentications needs to be active and configured to prevent unwanted access.

### Security Misconfiguration

Often web applications are misconfigured, leaving an array of vulnerabilities for attackers to capitalize. Security misconfigured vulnerabilities can include unpatched flaws, unused pages, unprotected files or directories, outdated software, and running software in debug mode.

**How to prevent security misconfigurations:**

Preventing security configuration vulnerabilities is simple. For instance, using a deployment protocol to continuously develop and deploy updates inside a secure environment or segmented application architecture will help prevent security vulnerabilities. Automatic your deployment will also keep your applications up to date and prevent attacks.

### Cross-Site Scripting XSS

Cross-site scripting uses malicious code injected into benign sites to attack a user’s web browser. An attacker will insert the code through a link and, together with social engineering, will lure the user to clicking the link and executing the code. Attackers using JavaScript for XSS vulnerabilities can access a user’s webcam, location, and other sensitive data and functions.

**How to prevent XSS:**

Ultimately XSS vulnerabilities can be fixed by sanitizing input. Sanitizing input will help stop user input from manipulating vulnerabilities and injecting them into websites. Also, validating and escaping user input will help prevent malicious injection.

### Cross-Site Request Forgery

Cross-site request forgeries (CSRF) use social engineering to trick authenticated users into clicking a link, as an example and take control of their sessions. Due to having authenticated sessions, the attacker can perform changes to the state of an app vs. data theft.

**How to prevent CSRF:**

There are several preventative measures to help stop CSRF attacks. Using secret tokens or cookies can help with authenticating real requests vs. malicious ones. Also, utilizing POST requests only and eliminating GET requests can help keep the URL information from getting compromised.

### Using Components with Known Vulnerabilities

Due diligence needs to get done when considering using a third-party code or component in your web application. Many security issues can come with using unfettered code from sources you aren’t familiar with.

**How to prevent using components with vulnerabilities:**

The best way to prevent using vulnerable code is to know where and who it’s coming from.

### Invalidated Redirects & Forwards

Invalidated redirects and forwards is another input manipulation vulnerability again using parameters like GET requests to execute the attacks.

**How to prevent invalidated redirects and forwards:**

By eliminating redirects, you can eliminate the issue of redirect attacks. If necessary, keep redirects and forwards static, not allowing users to input URLs.

**What is Access control?**

Access control is a fundamental component of data security that dictates who’s allowed to access and use company information and resources. Through authentication and authorization, access control policies make sure users are who they say they are and that they have appropriate access to company data. Access control can also be applied to limit physical access to campuses, buildings, rooms, and datacenters.

**How does access control work?**

Access control identifies users by verifying various login credentials, which can include usernames and passwords, PINs, biometric scans, and security tokens. Many access control systems also include multifactor authentication (MFA), a method that requires multiple authentication methods to verify a user’s identity.

Once a user is authenticated, access control then authorizes the appropriate level of access and allowed actions associated with that user’s credentials and IP address.

**There are four main types of access control:**

**Discretionary access control (DAC)**: In this method, the owner or administrator of the protected system, data, or resource sets the policies for who is allowed access.

**Mandatory access control (MAC):** In this nondiscretionary model, people are granted access based on an information clearance. A central authority regulates access rights based on different security levels. This model is common in government and military environments.

**Role-based access control (RBAC):** RBAC grants access based on defined business functions rather than the individual user’s identity. The goal is to provide users with access only to data that’s been deemed necessary for their roles within the organization. This widely used method is based on a complex combination of role assignments, authorizations, and permissions.

**Attribute-based access control (ABAC):** In this dynamic method, access is based on a set of attributes and environmental conditions, such as time of day and location, assigned to both users and resources.

**Why is access control important?**

Access control keeps confidential information such as customer data, personally identifiable information, and intellectual property from falling into the wrong hands. It’s a key component of the modern zero trust security framework, which uses various mechanisms to continuously verify access to the company network. Without robust access control policies, organizations risk data leakage from both internal and external sources.

Access control is particularly important for organizations with hybrid cloud and multi-cloud cloud environments, where resources, apps, and data reside both on premises and in the cloud. Access control can provide these environments with more robust access security beyond single sign-on (SSO), and prevent unauthorized access from unmanaged and BYO devices.

**What is single sign-on (SSO)?**

Single sign-on (SSO) is an authentication capability that lets users access multiple applications with one set of sign-in credentials. Enterprises typically use SSO authentication to provide simpler access to a variety of web, on-premises, and cloud applications for a better user experience. It can also give IT more control over user access, reduce password-related help desk calls, and improve security and compliance.

**Example of single sign-on:**

The user signs in only one time, hence the name of the feature (Single Sign-on). For example**, if you log in to a Google service such as Gmail, you are automatically authenticated to YouTube, AdSense, Google Analytics, and other Google apps.**

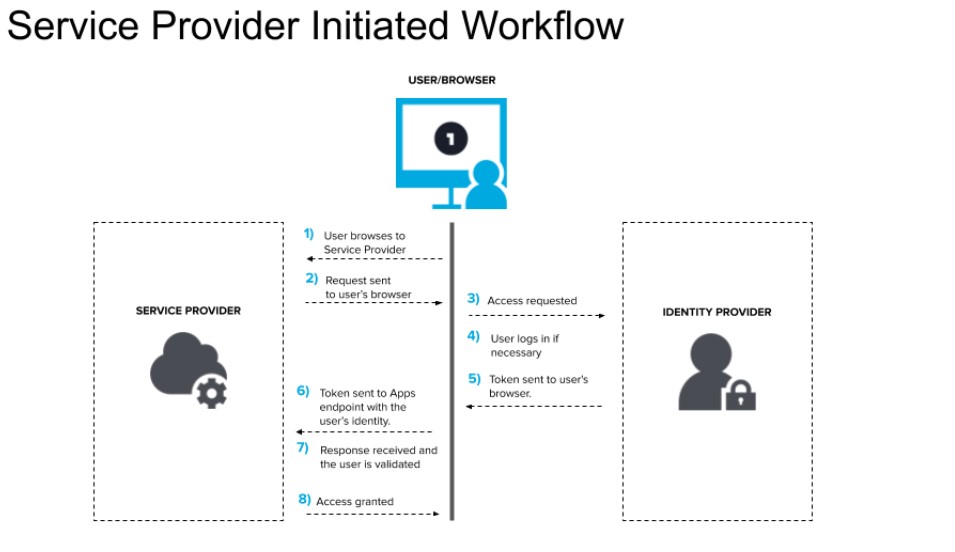
## How Does SSO Work?

SSO works based upon a trust relationship set up between an application, known as the service provider, and an identity provider, like One Login. This trust relationship is often based upon a certificate that is exchanged between the identity provider and the service provider. This certificate can be used to sign identity information that is being sent from the identity provider to the service provider so that the service provider knows it is coming from a trusted source. In SSO, this identity data takes the form of tokens which contain identifying bits of information about the user like a user’s email address or a username.

The login flow usually looks like this:

1. A user browses to the application or website they want access to, aka, the Service Provider.
2. The Service Provider sends a token that contains some information about the user, like their email address, to the SSO system, aka, the Identity Provider, as part of a request to authenticate the user.
3. The Identity Provider first checks to see whether the user has already been authenticated, in which case it will grant the user access to the Service Provider application and skip to step 5.
4. If the user hasn’t logged in, they will be prompted to do so by providing the credentials required by the Identity Provider. This could simply be a username and password or it might include some other form of authentication like a [One-Time Password (OTP)](https://www.onelogin.com/learn/otp-totp-hotp).
5. Once the Identity Provider validates the credentials provided, it will send a token back to the Service Provider confirming a successful authentication.
6. This token is passed through the user’s browser to the Service Provider.
7. The token that is received by the Service Provider is validated according to the trust relationship that was set up between the Service Provider and the Identity Provider during the initial configuration.
8. The user is granted access to the Service Provider.

When the user tries to access a different website, the new website would have to have a similar trust relationship configured with the SSO solution and the authentication flow would follow the same steps.



**What are the benefits of single sign-on?**

SSO offers benefits to both users and IT. From a user perspective, SSO alleviates password fatigue, making it easier and faster to access applications.

For IT, SSO can help reduce the number of password-related support calls. And automated credential management alleviates the burden of manually managing employees’ access to apps and services. SSO also makes it easier for IT to quickly provision and roll out SaaS applications to employees.

Additionally, from a security perspective, SSO can reduce the threat of cyber attacks, like phishing, by reducing the number of credentials at risk. It’s critical, however, to also implement multi-factor authentication as a backup in case passwords do become compromised.

**What are the types of SSO?**

* Federated Identity Management (FIM)
* OAuth (specifically OAuth 2.0 nowadays)
* OpenID Connect (OIDC)
* Security Access Markup Language (SAML)
* Same Sign On (SSO)

**SSO vs SAML**

Both the authentication protocols serve a similar function to connect users and allow them to access the requested resource. **SAML is an umbrella standard that covers federation, identity management and single sign on (SSO)**. SAML activates single Sign On (SSO) for browser based applications.

**What is FIM authentication?**

Federated identity management (FIM) is **an arrangement between multiple enterprises or domains that enables their users to use the same identification data (digital identity) to access all their networks**. These partners are also known as trust domains.

**SSO Vs FIM**

The key difference between SSO and FIM is while SSO is designed to authenticate a single credential across various systems within one organization, federated identity management systems offer single access to a number of applications across various enterprises.

**Top 10 Alternatives to SSO:**

* Rippling.
* JumpCloud.
* LastPass.
* Keeper Password Manager.
* Okta.
* Duo Security.
* OneLogin.
* Microsoft Azure Active Directory.

Why these 2