Contents

[Web Security 1](#_Toc118416004)

[Web Security Best Practice 2](#_Toc118416005)

[ SecDevOps 2](#_Toc118416006)

[ CSP 2](#_Toc118416007)

[ HSTS 2](#_Toc118416008)

[ Cookies security 2](#_Toc118416009)

[ X-XSS-Protection 3](#_Toc118416010)

[ HTTP Public Key Pinning 3](#_Toc118416011)

[ Resource loading 3](#_Toc118416012)

[ CSRF Protection 4](#_Toc118416013)

[ Referrer Policy 4](#_Toc118416014)

[ Subresource Integrity 5](#_Toc118416015)

[ Denial-of-Service Attack 5](#_Toc118416016)

## Web Security

Application without security is like a house without door. Web security is critical for preventing hackers and cyber-thieves from gaining access to sensitive information. Businesses run the risk of malware infection, network attacks, and other IT infrastructure attacks without a proactive security strategy.

No web technology has unquestionably demonstrated its invulnerability to this point. Every day, new threats emerge that necessitate at least some adjustment or improvement in the way countermeasures are put in place and overall web-focused security. Web application developers should always try to implement security measure to raise the overall quality of web applications.

## Web Security Best Practice

### SecDevOps

SecDevOps is very similar to DevSecOps but security is placed as the first priority of the process. It means that security should not only be tested during CI/CD but should be the first priority in software development. SecDevOps helps to eliminate security issues before they appear. Everyone involved in software creation should be aware of security risks introduced by vulnerabilities: all developers, operations engineers, product owners, administrators, etc. Then, you can implement verification, for example, by including a vulnerability scan in your CI/CD builds.

### CSP

Content Security Policy (CSP) is an additional layer of security that aids in the detection and mitigation of specific types of attacks such as Cross-Site Scripting (XSS) and data injection attacks. These attacks are used for a variety of purposes, including data theft, site defacement, and malware distribution.

### HSTS

The HTTP Strict-Transport-Security response header (often abbreviated as HSTS) informs browsers that the site should only be accessed via HTTPS and that any future HTTP attempts will be automatically converted to HTTPS.

### Cookies security

Cookies are mainly used for three purposes:

|  |  |  |
| --- | --- | --- |
| **Session Management** | **Personalization** | **Tracking** |
| Logins, shopping carts, game scores, or anything else the server should remember. | User preferences, themes, and other settings. | Recording and analysing user behaviour. |

Cookie’s lifetime should be defined. Permanent cookies are deleted at a date specified by the Expires attribute, or after a period of time specified by the Max-Age attribute. Cookies are intended to be sent securely and not accessed by unintended parties or scripts in one of two ways: with the Secure and HTTPOnly attributes.

### X-XSS-Protection

XSS means cross-site scripting. Cross-site scripting (XSS) is a security flaw that allows an attacker to inject malicious client-side code into a website. This code is executed by victims and allows attackers to circumvent access controls and impersonate users. When sites implement a strong Content-Security-Policy that disables the use of inline JavaScript ('unsafe-inline'), these protections are largely unnecessary in modern browsers.

### HTTP Public Key Pinning

HTTP Public Key Pinning must be enabled on high-risk sites (HPKP). A user agent is instructed by HPKP to bind a site to a specific root certificate authority, intermediate certificate authority, or end-entity public key. This prevents certificate authorities from issuing unauthorized certificates for a given domain that browsers would still trust. An active attacker could use these fraudulent certificates to MitM and impersonate a website, intercepting credentials and other sensitive data.

HPKP must be implemented with caution. Having backup key pins, testing on a non-production domain, testing with Public-Key-Pins-Report-Only, and finally doing initial testing with a very short-lived max-age directive are all part of this. HPKP is not recommended for the majority of websites due to the risk of causing a self-denial-of-service attack and the very low risk of issuing a fraudulent certificate.

### Resource loading

All resources, whether shared or not, should be loaded over secure channels. Browsers will block secure (HTTPS) websites that attempt to load active resources such as JavaScript insecurely. As a result, users will see degraded user interfaces and "mixed content" warnings. Although less dangerous, attempts to load passive content (such as images) insecurely will result in degraded UIs and can allow active attackers to deface websites or phish users. Despite the fact that modern browsers alert users when websites load resources insecurely, these errors still occur on a regular basis. To avoid this, developers should ensure that all resources are securely loaded prior to deployment.

### CSRF Protection

Cross-site request forgeries are a type of attack in which an unauthorized command is sent to a website by a trusted user. They appear to be validly issued commands because they inherit the user's cookies (and thus session information). When a user visits a page that contains that HTML fragment, the browser will make a GET request to that URL. If the user is logged in, the browser will provide session cookies, and the attempt to delete the account will be successful.

While there are several mitigation strategies available, including Origin/Referrer checking and challenge-response systems (such as CAPTCHA), the most common and transparent method of CSRF mitigation is the use of anti-CSRF tokens. Anti-CSRF tokens guard against CSRF attacks by requiring the presence of a secret, one-of-a-kind, and unpredictable token on all destructive changes. These tokens can be set to be valid for the duration of a user session, rotated on a regular basis, or created uniquely for each request. Although Same Site cookies are the best CSRF defence, they are not yet fully supported in all browsers and should be used in conjunction with other anti-CSRF defences.

### Referrer Policy

When a user navigates to a site via a hyperlink or a website loads an external resource, browsers use the HTTP Referer (sic) header to inform the destination site of the origin of the requests. Although this can be useful for a variety of purposes, it can also jeopardize users' privacy. HTTP Referrer Policy gives sites fine-grained control over how and when browsers send the HTTP Referer header.

### Subresource Integrity

Subresource integrity is a recent W3C standard that protects against attackers modifying the contents of JavaScript libraries hosted on content delivery networks (CDNs) in order to create vulnerabilities in all websites that make use of that hosted library.

For example, JavaScript code on jquery.org that is loaded from mozilla.org has access to the entire contents of everything of mozilla.org. If this resource was successfully attacked, it could modify download links, deface the site, steal credentials, cause denial-of-service attacks, and more.

Subresource integrity locks an external JavaScript resource to its known contents at a specific point in time. If the file is modified at any point thereafter, supporting web browsers will refuse to load it. As such, the use of subresource integrity is mandatory for all external JavaScript resources loaded from sources not hosted on Mozilla-controlled systems.

Note that CDNs must support the Cross Origin Resource Sharing (CORS) standard by setting the Access-Control-Allow-Origin header. Most CDNs already do this, but if the CDN you are loading does not support CORS, please contact Mozilla Information Security. We are happy to contact the CDN on your behalf.

### Denial-of-Service Attack

Denial-of-service (DoS) attacks overwhelm a server’s resources with API requests to slow, break, or crash the web server. Often, these attacks are made from multiple malicious sources simultaneously — a distributed denial-of-service (DDoS) attack. So proper security measures should be implemented to stop DoS attack.