UpGuard™

HANDBOOK

Securing Your Website in 13 Steps:
The Website Security Checklist

OVERVIEW

Putting a website on the internet means exposing that website to hacking attempts, port scans, traffic sniffers and data miners. If you're lucky, you might get some legitimate traffic as well, but not if someone takes down or defaces your site first. Most of us know to look for the lock icon when we're browsing to make sure a site is secure, but that only scratches the surface of what can be done to protect a web server. Even SSL itself can be done many ways, and some are much better than others. Cookies store sensitive information from websites; securing these can prevent impersonation. Additionally, setting a handful of configuration options can protect both your website against attacks and your customer's data from compromise. Here are 13 steps to harden your website and greatly increase the resiliency of your web server.

1. ENSURE SITEWIDE SSL

The lock in the browser address bar means the site you're on is secure, right? What it really means is that you are currently using an SSL connection. But to take full advantage of SSL and verify encrypted connections, SSL should be sitewide and enforced, not a page-to-page choice that hands the client back and forth between encrypted and unencrypted connections. Every page should only be available on SSL. Information transmitted outside of SSL connections passes in plain text and can easily be intercepted by anyone willing to put the work in. A single form with sensitive information or password entry on the unencrypted side could compromise the entire site.

2. VERIFY THE SSL CERTIFICATE

When does your SSL certificate expire? Is it trusted by default in all of the major browsers? Knowing the answers to these questions will make sure the effort you put into implementing SSL isn't wasted by an overlooked certificate expiration or turned into problems for customers because they get pop-up warnings about your site. To ensure the certificate doesn't expire, some mechanism should be in place to warn relevant parties when the certificate is near expiration. Most major certificate providers are automatically trusted in all common browsers, but it's always worth verifying that the company from whom you buy your certs is keeping up with the various security changes browser manufacturers are pushing. Failure to do so can lead to situations like when Firefox and Chrome blocked sites that used a weak Diffie-Hellmann key. Major changes like this require website administrators to re-issue any affected certificates and/or update their servers' configurations.

3. USE SHA256 ENCRYPTION

Speaking of major changes, certificates using the previously standard SHA1 encryption are no longer considered secure, as SHA256 standards have taken over, drastically improving the encryption. You can view the certificate of your website and if it has a SHA256 fingerprint, then it's using modern encryption. If it only has a SHA1 fingerprint, it should be re-issued or replaced with a 2048-bit SHA256 certificate, because SHA1 support will be removed from most browsers in 2017. Encryption standards will continue to change as ways are found to crack existing standards and more secure methods are developed.

4. DISABLE INSECURE CIPHER SUITES

Even if you have the best encryption options available, that doesn't mean that other, worse, options aren't coexisting with them. Default configurations of most web servers still allow SSL cipher suites that are considered insecure, such as RC4. These should be explicitly disabled on the web server (Apache, IIS) so malicious actors can't force one of these suites and exploit it. This is crucial, not only to security, but usability, as websites allowing insecure cipher suites will be automatically blocked by some browsers.

5. OBSCURE HEADER INFO

Advertising the type and version of your web server to the internet only aides those seeking to compromise it. By narrowing the window to a specific platform or version, attackers can focus their attempts on known vulnerabilities for the specific web server you're running. This is true for X-Powered-By headers, server information headers and ASP .NET headers where available. It is recommended best practice to obscure these headers and present no identifying information to visitors. This is not the default configuration, so many production servers still have these headers available, probably unknowingly.

6. ENABLE HTTP STRICT TRANSPORT SECURITY

HTTP Strict Transport Security (Linux, Windows) ensures that browsers only communicate with a website over SSL. Non-SSL requests (http://) will be converted to SSL requests (https://) automatically. Failure to utilize this measure can result in a man-in-the-middle attack, where a malicious actor could redirect a web user to a bogus site between the non-SSL and SSL handoff.

7. USE HTTPONLY COOKIES

Protecting cookies makes sure that information your site stores on visiting systems stays private and can't be exploited by an imposter. HttpOnly cookies restrict access to cookies so that client side scripts and cross-site scripting flaws can't take advantage of stored cookies. This should be enabled so modern browsers that support HttpOnly can have the additional protection. Users with browsers that don't support it will still receive traditional cookies.

8. USE SECURE COOKIES

Secure cookies can only be transmitted across an SSL connection. This prevents cookies with potentially sensitive information from being sniffed in transit between the server and the client. Failure to use secure cookies would allow a third party to intercept a cookie sent to a client and impersonate that client to the web server. Obviously to use secure cookies, you should already have ensured sitewide SSL, as cookies will no longer be delivered over unencrypted connections.

9. SECURE THE WEB SERVER PROCESSES

The web server process or service itself should not being running as root or Local System. On Linux systems, most web servers will run as a dedicated user with limited privileges, but you should double check what user it is and what permissions that user has. On Microsoft systems, chances are Local System is the default config and as such should be changed before production to a dedicated service account, local, unless the web server needs to access domain resources. This user should not be an administrator (or worse a domain admin) and should have file access only to what is necessary. Doing this prevents a compromised web server from further compromising other resources by isolating and restricting the account the web server uses.

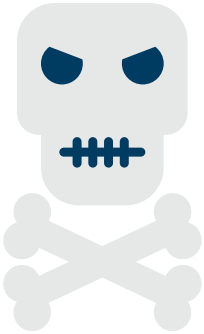
10. ENSURE FORMS VALIDATE INPUT

If you have forms that accept user input, every data input mechanism should be validated so that only proper data can be entered and stored in the database. This is the first step to protect against SQL injection and other exploits that enter bad data into a form and exploit it. This step must be taken on the development end, so it should be rolled into standard procedures if it isn't a part of them already.

11. PROTECT AGAINST SQL INJECTION

The second and most important step to protect yourself against SQL injection attacks is to utilize well-implemented stored procedures rather than open queries to perform database functions. By restricting your web application to run stored procedures, attempts to inject SQL code into your forms will usually fail. Stored procedures only accept certain types of input and will reject anything not meeting their criteria. Stored procedures can also be run as specific users within the database to restrict access even further. Again, since this is structural, it should be a best practice during the development and updating of the website backend.

12. PROTECT AGAINST DENIAL OF SERVICE



Denial of Service (DoS) attacks flood servers with connections and/or packets until they are overloaded and can't respond to legitimate requests. There's no way to absolutely prevent these types of attacks, because they use legitimate connectivity lanes, but there are measures you can take to resist them if they happen. Utilizing a cloud mitigation provider such as Akamai or CloudFlare will almost certainly prevent DoS attacks from causing you an issue. These solutions leverage the huge resources of distributed cloud architecture to offset the load of a DoS attack, as well as having identification and blocking mechanisms for malicious traffic. Alternatively, you can set up mitigation in-house, which operates on similar principles, but will be limited to the resources of whatever hardware your solution runs on.

13. REGULARLY TEST CONFIGURATIONS

When does your SSL certificate expire? Is it trusted by default in all of the major browsers? Knowing the answers to these questions will make sure the effort you put into implementing SSL isn't wasted by an overlooked certificate expiration or turned into problems for customers because they get pop-up warnings about your site. To ensure the certificate doesn't expire, some mechanism should be in place to warn relevant parties when the certificate is near expiration. Most major certificate providers are automatically trusted in all common browsers, but it's always worth verifying that the company from whom you buy your certs is keeping up with the various security changes browser manufacturers are pushing. Failure to do so can lead to situations like when Firefox and Chrome blocked sites that used a weak Diffie-Hellmann key. Major changes like this require website administrators to re-issue any affected certificates and/or update their servers' configurations.

CONCLUSION

There are many other steps that can be taken to protect against threats to a web server, but by following these 13, you should be resilient against all of the most common vulnerabilities. Furthermore, by integrating these practices into development and operations duties, companies can build a habit of security. Finally, by routinely testing configurations, companies can track changes and address security problems before they are exploited. UpGuard's free external risk grader analyzes websites for most of these security measures. Take a look at how secure your favorite websites are. How does yours hold up? Better yet, how secure is ***your*** website?

Check your website right now