INDUS NET TECHNOLOGIES

Why is Website Security Important?

www.indusnet.co.in • info@indusnet.co.in • +91 33 404644422

TABLE OF CONTENTS

WHY IS WEBSITE SECURITY IMPORTANT?

Your website is your identity, your brand, your storefront, and often your first contact with customers. If it's not safe and secure, you will lose customer faith and it is critical for business relationships. A threat can come in many forms – infecting a website with malware to spread, stealing customer information, like names and email addresses, stealing credit card and other transaction information, adding the website to a botnet of infected sites, and even crashing or hijacking the website.

A single security breach could be deathly for your business. A security breach can create huge impact on customer trust if customers find out about it.

An unprotected website is a security risk for the company as well as customers and other businesses. An infected website allows the spread and escalation of malware, attacks on other websites, and even attacks against national targets and infrastructure.

WEBSITE SECURITY CHECKLIST

When an website get live on the internet it gets exposed to hacking attempts, port scans, traffic sniffers and data miners. Most of the people look for the lock icon in the browser to see where the website is secure, but it's just the surface what can be done to protect the web server is what we are trying to explain with this article. In this article will be discussed 13 steps to harden the website and greatly increase the resiliency of the web server.

Ensure Sitewide SSL

Verify SSL Certificate

Use SHA256 Encryption

Disable Insecure Cipher Suites

Enable HTTP Strict Transport Security

Use HttpOnly Cookies

Use Secure Cookies

Secure the Web Server Processes

Ensure Forms Validate Input

Protect against SQL Injection

Protect against DoS

Ensure Sitewide SSL

The lock in the browser beside address means the website is secure using SSL connection. But to take the full advantage of SSL connection and encryption, SSL needs to be sitewide and enforced. SSL can't be a page-to-page choice that hands the client back and forth between encrypted and unencrypted connections. Every page of a website should be in SSL connection. If information transmitted outside of SSL connections then it passes in plain text can easily be intercepted by anyone who is willing to hack the website. A single form with sensitive information or password entry on the unencrypted side could compromise the entire website.

Verify SSL Certificate

The SSL certificate need to check from time to time. Every SSL certificate has an expiration date which need to be check. Also, we need to check whether the SSL certificate the website is having is trusted by default in all the major browsers. If a SSL certificate is not trusted by major browser then it leads to blocking of the website by major browser which leads loss of customer and business. A website administrator should check the website SSL certificate from time to ensure the SSL certificate validity.

Use SHA256 Encryption

A website needs encryption to make it more secure against various attacks. Previously SHA1 encryption was used but as the technology get evolved, the SHA1 encryption has become outdated and it's considered no longer secure. This bring SHA256 standard encryption which has drastically improved the encryption level. If a website has SHA1 fingerprint, it should be re-issued or replaced with a 048-bit SHA256 certificate, as most of the current browser removed SHA1 support. Encryption technology will continue to change as ways are found to crack existing standards and more secure methods are developed.

Disable Insecure Cipher Suites

Cipher suits is a concept used in Transport Layer Security network protocol. Cipher suites are considered insecure like RC4. Cipher suites have become outdated technology and it should be explicitly disabled on the web server so that no malicious actors can't force one of these suites and exploit it. Disable insecure cipher suites is not only to ensure security but also to increase usability, as websites allowing insecure cipher suites will be automatically blocked by some browsers.

Enable HTTP Strict Transport Security

By enabling HTTP Strict Transport Security we can ensure that browsers only communicate with website over SSL. By using this technique Non-SSL request (http://) will be converted to SSL requests (https://) automatically. A disabled HTTP Strict Transport Security website could attract malicious actor and could redirect a web user to a bogus site.

Use HttpOnly Cookies

Cookies is a small piece of data sent from a website and stored on the user's computer by the user's web browser while the user is browsing. Protecting cookies make sure that information of your website stores on visiting user's system stay private and it can't be exploited by an imposter. By HttpOnly cookies we are restricting access to cookies so that client side scripts and cross-site scripting flaws can't take advantage of stored cookies. HttpOnly cookies enables modern browsers that support HttpOnly to have additional protection.

Uses Secure Cookies

Unlike normal cookies secured cookies can only be transmitted across an SSL connection. This prevents cookies with potentially sensitive information from being transit between the server and the client. In case of normal cookies a third party can intercept the cookie sent to a client and impersonate that client to the web server. Secure cookies will be increased the security of website.

Secure the Web Server Processes

The web server process should not being running as root on local system. On most of the web servers in Linux systems will run as a dedicated user with limited privileges, we should always check what user it is and what permission that user has. In case of Microsoft systems, local system is the default configuration and it should be changes before production to a dedicated service account, local, unless the web server needs to access domain resources. This process prevent a compromised web server from further compromising other resources by isolating and restricting the account the web server uses.

Ensure Forms Validate Input

If the website contains a form that accept user input, then every data input mechanism should be validated so that only proper data can be entered and stored in the database. This step is the 1st barrier to SQL injection and other exploits that enter bad and unwanted data into the website.

Protect Against SQL Injection

The 2nd most important step in protection of website is protection against SQL injection attacks. By restricting the website to run stored procedures, attempts to inject SQL code into the forms will usually fail. A stored procedure enabled website only accept certain types of input and will reject anything not meeting the set criteria. Stored procedures can also run as specific users with the database to restrict access even further.

Protect Against Denial of Service (DoS)

Dos attacks can flood servers with connections and/or packets until they are overloaded and can't respond to legitimate request. There are no way to prevent these types of attack, because the attacker use legitimate connectivity lanes, but there are measures we can take to resist them if it happens. We can use cloud mitigation provider which will almost certainly present DoS attacks from causing an issue. These cloud mitigation platforms uses huge resources on a distributed cloud architecture to offset the load of a DoS attack, as well as having identification and blocking mechanisms for malicious traffic. We can create an in-house mitigation system but it will cost a lot and it has the limitation of resources.

	Blog	Information Website	eCommerce	BFSI
Ensure Sitewide SSL	No	Yes (it depends on the type of information the website have)	Yes	Yes
Verify SSL Certificate	No	Yes (it depends on the type of information the website have)	Yes	Yes
Use SHA256 Encryption	Yes	Yes	Yes	Yes
Disable Insecure Cipher Suites	Yes	Yes	Yes	Yes
Enable HTTP Strict Transport Security	Yes	Yes	Yes	Yes
Use HttpOnly Cookies	Yes	Yes	Yes	Yes
Use Secure Cookies	Yes	Yes	Yes	Yes
Secure the Web Server Processes	Yes	Yes	Yes	Yes
Ensure Forms Validate Input	Yes (if the blog have any form)	Yes	Yes	Yes
Protect against SQL Injection	No (if the blog have a db then yes)	No (if the blog have a db then yes)	Yes	Yes
Protect against DoS	No	No	Yes	Yes

WEBSITE SECURITY TEST CHECKLIST

Deployment

1. Test for missing security updates
2. Test for unsupported or end-of-life software versions
3. Test for HTTP TRACK and TRACE methods
4. Test for extraneous functionality
5. Test the server using the Server Security Test Checklist

Information Disclosure

1. Test for extraneous files in the document root
2. Test for extraneous directory listings
3. Test for accessible debug functionality
4. Test for sensitive information in log and error messages
5. Test for sensitive information in robots.txt
6. Test for sensitive information in source code
7. Test for disclosure of internal addresses

Privacy & Confidentiality

1. Test for sensitive information stored in URLs
2. Test for unencrypted sensitive information stored at the client-side
3. Test for sensitive information stored in (externally) archived pages
4. Test for content included from untrusted sources
5. Test for caching of pages with sensitive information
6. Test for insecure transmission of sensitive information
7. Test for non-SSL/TLS pages on sites processing sensitive information
8. Test for SSL/TLS pages served with mixed content
9. Test for missing HSTS header on full SSL sites
10. Test for known vulnerabilities in SSL/TLS
11. Test for weak, untrusted or expired SSL certificates
12. Test for the usage of unproven cryptographic primitives
13. Test for the incorrect usage of cryptographic primitives

State Management

1. Test for client-side state management
2. Test for invalid state transitions

Authentication & Authorization

1. Test for missing authentication or authorization
2. Test for client-side authentication
3. Test for predictable and default credentials
4. Test for predictable authentication or authorization tokens
5. Test for authentication or authorization based on obscurity
6. Test for identifier-based authorization
7. Test for acceptance of weak passwords
8. Test for plaintext retrieval of passwords
9. Test for missing rate limiting on authentication functionality
10. Test for missing re-authentication when changing credentials
11. Test for missing logout functionality

User Input

1. Test for SQL injection
2. Test for path traversal and filename injection
3. Test for cross-site scripting
4. Test for system command injection
5. Test for XML injection
6. Test for XPath injection
7. Test for XSL(T) injection
8. Test for SSI injection
9. Test for HTTP header injection
10. Test for HTTP parameter injection
11. Test for LDAP injection
12. Test for dynamic scripting injection
13. Test for regular expression injection
14. Test for data property/field injection
15. Test for protocol-specific injection
16. Test for expression language injection

Sessions

1. Test for cross-site request forgery (CSRF)
2. Test for predictable CSRF tokens
3. Test for missing session revocation on logout
4. Test for missing session regeneration on login
5. Test for missing session regeneration when changing credentials
6. Test for missing revocation of other sessions when changing credentials
7. Test for missing Secure flag on session cookies
8. Test for missing HttpOnly Flag on session cookies
9. Test for non-restrictive domain on session cookies
10. Test for non-restrictive or missing path on session cookies
11. Test for predictable session identifiers
12. Test for session identifier collisions
13. Test for session fixation
14. Test for insecure transmission of session identifiers
15. Test for external session hijacking
16. Test for missing periodic expiration of sessions

File Uploads

1. Test for storage of uploaded files in the document root
2. Test for execution or interpretation of uploaded files
3. Test for uploading outside of designated upload directory
4. Test for missing size restrictions on uploaded files
5. Test for missing type validation on uploaded files

Content

1. Test for missing or non-specific content type definitions
2. Test for missing character set definitions
3. Test for missing anti content sniffing measures

XML Processing

1. Test for XML external entity expansion
2. Test for external DTD parsing
3. Test for extraneous or dangerous XML extensions
4. Test for recursive entity expansion

Miscellaneous

1. Test for missing anti-clickjacking measures
2. Test for open redirection
3. Test for insecure cross-domain access policy
4. Test for missing rate limiting on e-mail functionality
5. Test for missing rate limiting on resource intensive functionality
6. Test for inappropriate rate limiting resulting in a denial of service
7. Test for application- or setup-specific problems