Journal of Engineering Education Transformations, Volume No 36, December 2022, Special issue, eISSN 2394-1707

Secure Web Application: Rudimentary perspective

IPS Sethi[1], Sanjay Kumar Sinha[2], Neeta Chauhan[3], Deepti Khanduja [4]

[1,2,3,4] National Informatics Centre, New Delhi
[1]sethi@nic.in, [2]sanjayk.sinha@nic.in, [3]neeta.chauhan@nic.in, [4]deepti.khanduja@nic.in

Abstract: **WWW, one of the most pervasive technologies for information and service delivery over Internet with a potential to revise and preserve the web applications without dispensing and installing software on doubtlessly millions of client computers. As the web applications are increasingly used for crucial services, they have become a prominent and relevant target for any security outbreak. Software security is a methodology which guards against the malicious attacks and security failures along with an aim to increase system reliability. The prime objective of software security is to gain knowledge about the vulnerabilities in a system and foresee attacker's motive and perception.**

This paper reviews the existing techniques of web application security, with the aim of standardizing them into a bigger picture to enable the future research areas. The scrutiny of a web application attack and the attack techniques are also enclosed in details. Lastly the parameters to provide a secure hosting surrounding to the applications are indexed. The paper summarizes the security of web application in a holistic manner and provides a range of ways to ensure that it's as secure as it can be, as well as forever improving.

Keywords: Security, OWASP, SDLC, SQL Injection, Web Application Firewall.

1. Introduction

WWW, one of the most pervasive technologies for information and service delivery over Internet with a potential to revise and preserve the web applications without dispensing and installing software on doubtlessly millions of client computers. WWW has emerged from a system that used to distribute static web-pages to a platform that now supports distributed applications, known as web applications.

Web application consists of multiple layers, like web browsers, hosts (e.g. Application server, Web server and database server), data stored on the hosts, and network. Each layer of web application has its security issues which may result in vulnerability and hence must be secured.

With the Web being an open source system and web applications delivering critical services, they become one of the invaluable targets for security attacks. The security of web applications has become a paramount concern for users of such applications, especially if these applications are complex and interactive, or if they involve sensitive information exchanged in sectors such as finance, health, or banking.

The primary objective of software security is gaining knowledge about an attacker and anticipating his motives and perceptions. Its primary objective is to strengthen system reliability by guarding against cybersecurity risks and failures. In today's world, developing secure software is no longer a luxury, but a necessity for every software company. Due to the immediate to access web applications has motivated a thriving number of researchers to specialize in web application hardening and attack reduction.

Information security measures must meet the CIA security triangle's three essential functions: Confidentiality, Integrity, and Availability [2]. It is designed to serve as a tool and guide for securing computer systems, networks, and related technical assets. Due to the widespread use of information systems and networks in modern society, it is important to develop and enforce policies, procedures, and mechanisms to address security issues while also achieving the essential elements of the CIA triad.

The paper has been structured as follows. Section II gives a dip into the statistics of web service vulnerability for the year 2019-20. Then, Section III illustrates the essential security properties that a secure web application should adhere, as well as corresponding vulnerabilities and attack vectors. Section IV provides a list of monitoring parameters to provide a secure hosting environment. We conclude our survey paper in Section V.

2. Assessment of Web Application

In this section we will examine the threat landscape for web applications during the year 2019 to 2020. Security Misconfiguration vulnerabilities are the most commonly encountered in web applications, as Without the HttpOnly and Secure settings, hackers can target the user session and steal sensitive cookies. Attackers further use such flaws to execute Cross-Site Scripting (XSS) in order to capture the user's session identifier and impersonates the user in the application[11].

45 percent of web applications have a broken authentication vulnerability, namely the inability to limit

the number of authentication attempts, which can be exploited to access web applications.

It was reported by every third application that the access control was broken, which resulted in information being disclosed, modified, or destroyed to unauthorized user. Nonetheless, a web application can be developed by using the Secure Software Development Lifecycle (SSDLC) during development to minimize authentication and authorization vulnerabilities.

Clickjacking is a threat to another third of online apps (User Interface Misrepresentation of Critical Information, CWE451), where the user visits an attacker's site and clicks a transparent HTML iframe, which results in an unintended action on the susceptible site [11]. In order to prevent such attacks, an HTTP header called X-Frame-Options can be used.

CSRF attacks were discovered in another third of the websites. In a CSRF attack, a hacker spoofs as a registered user into a vulnerable website/application to perform actions as that user. Typically, protection of webapp involves requiring one-time keys (CSRF tokens), verifying authenticity (with a password or OTP, for example), confirming that a request has been originated from an authorized user (using CAPTCHA), or using an additional SameSite cookie flag [1].

According to statistics, 9 out of 10 web applications are vulnerable to hacking. XSS is one of the leading cause among these attacks. Users may become infected with malware, and phishing attacks may be used to steal their passwords. In order to prevent it, it is a universal suggestion that web applications sanitize all user input that is subsequently shown in a browser, particularly HTTP request header fields such as User-Agent and Referrer.[11] It is necessary to substitute non-formatting equivalents for potentially unsafe characters on HTML pages. In addition, it is recommended to use modern web application firewalls (WAFs) that block cross-site scripting.

Breaches of significant information are the second-most dire threat to website security. In almost half of all breaches (i.e., 47%) personal data was at risk while User credentials(31%) figured prominently as well. Information has been the prime target of hackers when they target an organization.

A second-most significant threat to site security is the compromise of vital information. The vast majority of breaches (47%) have exposed personal information, while 31% exposed user credentials. Data has traditionally been a top target for hackers when they strike at an organization. According to the study, 82% of vulnerabilities are identified in the application code. Figure 1 depicts the Severity of OWASP vulnerabilities in 2020. Testing of the source code is therefore a key component of the Secure Software Development Lifecycle, which can be done independently with a code analyser or can be provided to testers. An extensive white-box security assessment is carried out simultaneously by several security experts to detect as many vulnerabilities as possible. Table 1 depicts the vulnerabilities detected by white box testing.

Table 1

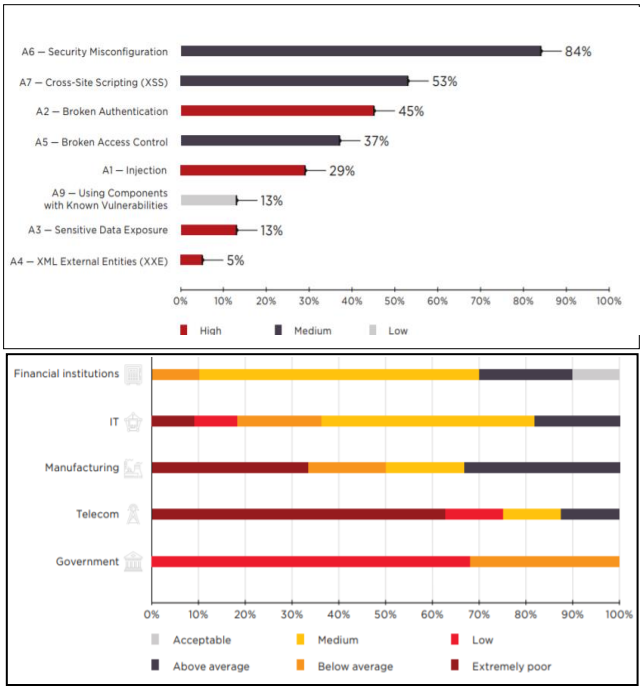
Detected Vulnerabilities

OWASP No	Name	% detected by white box testing
A4	XXE	100%
A1	Injection	76%
A7	XSS	67%

As per The Verizon 2020 Data Breach Investigation Report (DBIR) [3] data breaches are increasingly predominantly caused by attacks against web applications. The study is based on a review of 32,002 security incidents and 3,950 confirmed breaches across 81 contributors in 81 countries.

A huge 43% of security breaches have been attributed to web application attacks - which is more than double the results from last year. Figure 2 illustrates the vulnerabilities by industries. Data breaches are predominantly motivated by illicit financial gain (86%) - a significant increase from 71 % in 2019 - while two thirds (67%) are caused by breaching credentials, human mistake, or social engineering attacks.

More than a quarter 27% of malware incidents covered by the study were attributed to ransomware.



3. A Strategy For Securing Web Applications

The security of Web applications is technology-centric and influenced by organization rules and regulations, legal policies, and the practices of the people involved in deploying, developing, and maintaining Web applications.

In this article, we cover a wide range of concepts that together constitute the basis for data security. As we look at web application security holistically, we offer multiple options to ensure that it is as secure as it can be, as well as continuously improving.

A. Securing at Software Development Stage

Incorporating a security layer throughout all phases of software development will provide software users with a safe cyber environment. The Secure SDLC(SSDLC) is a set of best practices aimed at the enhancement of security within the standard SDLC[4]. From requirement gathering to deployment and maintenance, building a secure SDLC process demands dedicated effort at each step.

a. By keeping all security testing until the end of the SDLC, you're increasing the risk of having to break the build at a late stage or allowing flaws to leak into the application. With agile environments paving the way for how all organizations will run in the near future, secure coding is essential for the longevity of any organization to be viable.

b. Treating functionality and performance bugs with a higher regard than security bugs. Performance and functionality are important aspects of any application and your users deserve high-quality products[5]. Yet security needs to be considered equally – and we can no longer afford to compromise security for some sparkly feature. Do not neglect security in favour of speed or number of features in your application.

c. Test the app before each new release. Each new release offers new code to attackers to find flaws to exploit. Do not deploy small updates in your applications without scanning the code changes. Don't skimp on security testing future releases, no matter how small the added changes are. Ensuring that libraries are called correctly, added components secure, and new code free of vulnerabilities needs to be done each time you update an application.

With incremental scanning available in the newer SAST tools, testing for security flaws with each new update doesn't need to cause delays. Testing only the newly implemented code and their dependencies, incremental scanning can save lots of headaches and resources caused when security testing slows down the SDLC.

B. System Hardening

System hardening is the process of configuring an asset in line with security to reduce its vulnerability to cyber-attacks. The process involves reducing the 'attack surface' of the asset by disabling unnecessary services, user accounts, and ports.[7] It provides protection in layers, i.e., protecting at the host level, the application level, the operating system level, the user level, the physical level and all the sublevels in between. Each level requiring a unique method of security.

Hardening activities for a computer system include the following:

- Keeping security patches and hot fixes updated
- Monitoring security bulletins that are applicable to a system's operating system and applications
- Installing a firewall
- Closing certain ports such as server ports
- Not allowing file sharing among programs
- Installing virus and spyware protection, including an anti-adware tool so that malicious software cannot gain access to the computer on which it is installed
- Keeping a backup, such as a hard drive, of the computer system
- Disabling cookies
- Creating strong passwords
- Removing unnecessary programs and user accounts from the computer
- Using encryption where possible
- Hardening security policies, such as local policies relating to how often a password should be changed and how long and in what format a password must be in.

C. OS Hardening

Hardening of the OS is the act of configuring an OS securely, updating it, creating rules and policies to help govern the system in a secure manner, and removing unnecessary applications and services[6]. This is done to minimize a computer OS's exposure to threats and to mitigate possible risk.

While different operating systems have their own intricacies, there are numerous recommendations such as configuring system and network components properly, deleting unused files and applying the latest patches.

D. Server Hardening

Every vulnerability management programme should include hardening servers while guaranteeing server security. Attackers could take advantage of web server weaknesses to obtain access to the systems that host web servers and perform undesired actions[8].

The steps to acquire server hardening involves:

- Disable the signature: The server signature, commonly known as the "server footer," can be disabled to prevent the server name, server version number, and other information from showing on the computer. It is possible to secure web servers by including the commands "ServerSignature Off" and "ServerTokens Prod" in the server configuration file.
- Disable HTTP Trace and Track requests: Cross-site scripting attacks are a common exploitation method, in which attackers can capture the sessions cookies and traffic connections from normal traffic, as well as any data in transit using the HTTP TRACE and TRACK methods.

- Create non-root users: For basic administrative and management tasks, you need to create and use non-root accounts. This is a best practice measure for web servers, but it also applies to other operating systems.
- Restrict IP access: In the case that your web server is only used for limited purposes such as internal organizational information sharing, hosting a static website or testing and development, you can restrict access to specific IP addresses.
- Disable SSLv2 and SSLv3: Despite being known to have security problems, most web servers still run SSL 2.0/3.0 and TLS 1.0/1.1 protocols by default. This compromises the security of any data transferred over these protocols. Thus, SSLv2 and SSLv3 need to be disabled, as well as TLS 1.0 and 1.1, and in their place, we should enable TLS 1.2.
- Disable directory listing: Directory listing can also be disabled in the same way as web server signatures. If there is no index.html file in the root directory, web servers display the content of the documents and files there by default.
- Eliminate unused modules: If the web server is configured using the default operating system configuration, there's a high probability that several dispensable modules are running. Because the more services and functions a web server runs, the more opportunity a potential hacker has to exploit your network, disabling and turning off any superfluous services, ports, or functions. A simple best practise. Is that any internal or exterior ports and modules that aren't in use should be disabled.
- Install Mod_evasive and Mod_security: Mod security also referred to as ModSec, is a web server supplemental firewall(WAF) that allows you to monitor traffic in real time while also preventing host connections if the module detects any brute-force password attempts. Mod evasive (an Apache module) is used to help prevent DDOS assaults by terminating connections if too many requests arrive into a website too soon, if a child process request tries to make too many concurrent requests, or if any host IP tries to contact the web server despite being banned.
- Constantly check for patches: You should check for updates and patches on your server on a regular basis. Even after installation, this should not be a one-time function.
- Prevent Lateral attack within the same VLAN: Set Up an Iptables Firewall to Protect Traffic Between your Servers.

E. Prevent SQL Injection

The insertion or "injection" of a SQL query by the entered data from the client to the application program is referred to as a SQL injection attack. A successful SQL injection exploit includes reading sensitive information from the database, modification of database data (Insert/Update/Delete), performing database administration operations (such as shutting down the DBMS), recovering the content of a given file on the DBMS file system, and rarely, issuing commands to the operating system.

- Instead of string concatenation, use parameterized queries or stored procedures to access a database. Parameterized queries(prepared statements) require you to first specify all of the SQL code before passing each argument to the query. This enables the database to distinguish between code and data, regardless of the type of user input provided. In the same way that parameterized queries require you to create the SQL code upfront and then pass in the parameters afterwards, stored procedures do the same. A stored procedure differs from a regular procedure in that the SQL code for it is defined and saved in the database, then invoked from the application.
- Protect user input by "whitelisting" the characters that are permitted. Allow just the letters a-z and A-Z if you're asking for a name. Limit characters to 0-9 for phone numbers. To do this, use the proper validation mechanism allowed by your database.
- Use the character escaping strategy set up by your database to escape user-supplied input. By escaping special characters, you're telling your database that the characters in your query are data rather than code. The database will not confuse user-supplied input with SQL code you've written if all user-supplied input is then escaped using the right scheme.
- Don't make it easier for attackers. Ensure that error messages do not reveal information that might be exploited against the site later.

F. Auditing & Logging

The auditing and logging of security-relevant events and system abnormalities are key elements in the after-the fact detection of, and recovery form, security breaches [10]. Even when implemented systematically and full application coverage has been achieved, there are several guidelines that should be considered

- No sensitive information or technical details should be disclosed in error messages presented to the user. This is to prevent an attacker from gaining a better understanding of the internal implementation details or the supporting infrastructure
- Standard HTTP error codes (e.g. 404, 500, etc.) should be handled by the application and never be returned to users. Most development frameworks provide functionality to supply alternative error handlers.
- Logs must be stored in high-integrity remote destinations. They should not be stored within the web server. Access to the log storage should be secure physically.
- Log data should be transmitted to the remote storage in an encrypted and authenticated fashion. Consider using

write-once read-many physical supports such as tapes for log storage.

- The log storage should be appended only. It should not be possible to delete records or overwrite existing entries
- Read permission to the log should be granted carefully. If an attacker manages to get access to the application's log, then very sensitive information may be disclosed
- Exclude sensitive data such as passwords from the logs and ensure that logs are backed up regularly and that a copy is kept in a safe off-site location
- Beware of log rotation mechanisms. Ensure that all your logs are backed up prior to allowing any logs to be rotated.

G. Access Control

The fundamental goal of the access control module is to regulate which resources and operations can be performed by which users within an application. Access control mechanism must be rigorously enforced throughout the application. Each module in the application must be protected by an authorization filter where, every request is matched against the authorization framework without exception.

Access controls defines security policies and enforce the defined security policy on the authorized parties. The objective of access control is to preserve and safeguard the integrity and confidentiality of data. The various access control mechanisms are discussed that will be used while designing and implementing access control framework in application.

- Mandatory Access Control (MAC): The security policy administrator defines the access control policy that is strictly enforced in the system. The policy states the resources and operations permitted for each user and user cannot alter the policy rules. The strict system-imposed rules mandate what processes and threads are permitted to do with resources such as files and TCP connections.
- Discretionary Access Control (DAC): Each resource in the application is assigned to an owner. The owner of a resource can decide to grant privileges to interact with the resource to other users.
- Role-based Access Control (RBAC): Access to resources is mandated through the use of groups defined by a business role (e.g. Finance, Accounting, Guests, etc.). Authenticated users can be a part of multiple groups and thus access resources and functionality accordingly.

In any web application with multiple privilege levels, at least two different types of access control need to be implemented:

- Vertical access controls ensure that users of a lower privilege level cannot perform actions or access resources reserved to higher privilege accounts.
- Horizontal access controls prevent users from accessing or performing actions on resources that belong to other users with at the same privilege level.

H. WAF

A 'web application firewall (WAF)' is an HTTP application firewall[9]. An HTTP interaction is subjected to a set of rules. These rules, in general, protect against common attacks like Cross-site Scripting (XSS) and SQL Injection. WAFs defend servers rather than clients, like proxies do. A web application firewall (WAF) is used to safeguard a single web application or a group of web apps. A reverse proxy may be thought of as a WAF.

WAFs provide a powerful protection since online applications are continually evolving and every new modification has the danger of creating a new vulnerability. A WAF must not just detect and prevent known threats at the application and business logic levels to be effective. It must also detect zero-day vulnerabilities and protect users from cyber-attacks.

WAFs can be deployed on-premise, in the cloud, or a hybrid of the two, depending on the requirement, infrastructure, and other factors. As more businesses migrate their apps and data to the cloud, it's critical to consider your security requirements. Adding a solution with a dedicated security team to your selection criteria is a good idea. To adequately secure your assets, security teams can roll out timely security updates.

4. Hosting Environment Security

The application development team apart from the knowledge of development platform, should have working knowledge of server hardening (OS, database), network infrastructure and its security so that the application can be hosted in a secure environment. The following parameters should be monitored.

1. VA of servers (both OS and database) should be done periodically
2. All servers should be tightened with respect to security and monitored regularly
3. Additional security measures may be implemented for intrusion prevention.
4. Application security audit should be renewed periodically even though there is no change in the code
5. Regularly check audit logs for authenticated traffic.
6. Antivirus should be enabled on all the server
7. All uploaded files should be scanned by AV to the application
8. All public sites should run on SSL
9. All logs to be enabled in the servers for capturing footprints.

10. WAF (Web application firewalls) must be implemented in all servers.
11. Patch Management Solutions (PMS) to be implemented regularly in the servers.
12. Firewall policies may be strengthened. No unwanted ports should be left open.
13. Applications should not be hosted with root privileges.
14. Creating strong passwords and keep changing passwords after an interval.
15. If users are limited for a public site, white list IP range for authorized users.
16. Prepare proper diagram of the interaction if the software is exchanging data and information with other softwares to validate and secure communication
17. Regularly check Inbound Traffic, Outbound Traffic, traffic at port 80.
18. Synchronize your servers with NTP server
19. Regularly check IPS logs which gives the report about most attacking clients, clients using highest consuming bandwidth and detect and prevent identified threats.
20. If any suspicious Ip is found, get its details from the network team and capture the data packet which is being sent to it or accessed by it. It will clarify any vulnerability in your software or lack in hardening.
21. If logs are studied properly then it can act as major breakthrough in finding ambiguities if any, which otherwise go unnoticed and undetected. For example, if you define a server entry in /etc/sysconfig/network file and the same entry is missing in /etc/hosts , then to resolve the server IP, it will keep hitting DNS server which is undesirable.

5. Conclusions

However, the fraction of online apps with significant vulnerabilities is steadily decreasing year after year. It's not simple to achieve and maintain excellent web application security on a constant basis. There are two ground rules, however:

- Fix any detected flaws as soon as possible
- Make processes automatic wherever possible

Preventive techniques such as web application firewalls (WAFs) offer a powerful protection since online applications are continually evolving and every new modification carries the danger of creating a new vulnerability.

It must also identify zero-day exploits, protect users from assaults, and analyse and correlate events in order to discover attack chains.

The concept of building websites to work as planned, even when they are under attack, is web application security. To protect its properties from potentially malicious agents, the definition requires a series of security controls built into a web application. Online applications invariably involve bugs, like all apps. Any of these bugs represent real vulnerabilities that can be abused, putting organizations at risk. Protection for web applications protects against such defects. Throughout the life cycle of software development, it includes exploiting safe development practices and enforcing security controls, ensuring that design-level vulnerabilities and implementation-level bugs are addressed.

References

[1] D. H. G. B. ,. P. L. A. P. Suzanne Widup, "2020 Verizon Data Breach Investigations Report," 2020.

[2] R. M. N. K. S. K. K. Sandeep Kumar, "A study on web application security and detecting security vulnerabilities," in *2017 6th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)*, 2017.

[3] S. a. H. M. a. H. B. a. A. A. a. A. M. a. I. K. Rafique, "Web application security vulnerabilities detection approaches: A systematic mapping study," in *2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, 2015.

[4] X. a. X. Y. Li, "A Survey on Server-Side Approaches to Securing Web Applications," *ACM Comput. Surv.*, p. 29, 2014.

[5] N. P. a. M. R. B. a. M. H. Khan, "Software Security Issues: Requirement Perspectives," *International Journal of Scientific & Engineering Research*, 2014.

[6] A. a. J. S. Dalai, "Evaluation of web application security risks and secure design patterns," in *Proceedings of the 2011 International Conference on Communication, Computing & Security*, 2011.

[7] D. a. P. S. Banerjee, "Research on software security awareness: problems and prospects," *ACM SIGSOFT Software Engineering Notes*, pp. 1-5, 10 2010.

[8] J. a. W. S. Andress, "The Basics of Information Security: Understanding the Fundamentals of InfoSec in Theory and Practice: Second Edition," pp. 1-217, 01 2014.

[9] M. Al-ibrahim, "The Reality of Applying Security in Web Applications in Academia," *International Journal of Advanced Computer Science and Applications*, 2014.

[10] O. M. Y. a. R. I. Alhazmi, "Security Vulnerabilities in Software Systems: A Quantitative Perspective," in *Data and Applications Security XIX*, Berlin, Heidelberg, 2005.

[11] P. technologies, "www.ptsecurity.com".

Acknowledgement

This research was supported by National Informatics Centre. We thank our colleagues from NIC who provided insight and expertise that greatly assisted the research. We would also like to show our gratitude for sharing their pearls of wisdom with us during the course of this research.