Vulnerability Prioritization for Web Application Security

Shreya A. Shah

San Jose State University

Author Note

Shreya Atulkumar Shah, Department of Software Engineering, San Jose State University

This paper was prepared for Technical Writing Cmpe-294 course under the guidance of Professor Robert Bruce and want to thank him for his constant feedback.

Table of Contents

Abstract 4

Introduction 5

Description 6

Goals 6

Literature Survey 7

Web Security Terminologies 8

Client. 8

Attacker. 8

Web server. 8

Types of Attacks 9

Application Vulnerabilities. 9

Buffer overflow. 9

Credentials Hijacking. 9

Cross-Site Request Forgery. 11

Cross-Site Scripting (XSS). 11

Failure to Restrict URL Access. 11

Directory Traversal. 11

SQL Injection. 12

LDAP Injection. 12

Insufficient Transport Layer Protection. 13

Error Handling. 13

Insecure Cryptographic Storage. 14

Prevention Mechanism Against Web Security Vulnerabilities 14

Web application firewall (WAF) 14

XSS Attacks 15

SQL Injections 15

Validate User Inputs. 16

Avoid using direct user input on SQL queries. 16

Access Privileges on the Database. 16

Proposed Vulnerability Prioritization 16

Remediation 17

Mitigation 18

Acceptance 18

Conclusion 19

References 20

Figures 21

Abstract

Web application is a common of communication with the users through web browser. World is tremendously using internet for all kind of purposes along with growth of web application over the years. Evolution of all these has led to increase in security vulnerabilities for the web application. Since web applications are open to users, security issues are prone to happen. Web security in a general term means layering your web application from the security threats attacked by unauthorized users. Web security can be anything from an inappropriate input by the user to cross site scripting which means injecting malicious script in the web application. The main concern is to identify, analyze and improve these security issues considering the existing studies in a web application. This paper mainly focusses on different types of security vulnerabilities, their impacts on the web application and their prioritizing parameters to reduce the potential threats on the application. This paper explores various approaches for prioritizing threats depending on the type of web applications and addresses some options to mitigate these attacks.

Keywords: *Interactive Web Application, Vulnerabilities, Unauthorized Users, Security Attacks, Injection attack, Cross-site Scripting (XSS).*

Vulnerability Prioritization for Web Application Security

Introduction

In today's era of internet of thing with everything on internet, Web application platform plays a vital role in integrating various functional aspects of the system at a common page. It is a core platform to connect individuals with each other and with various organizations located at different geographical locations. Billions of individuals all over the world use WB technologies to get information, perform financial transactions, and have fun and communicate and to socialize themselves (Rafique, Humayun, Hamid, Abbas, Akhtar, & Iqbal, 2015). Social media which has become a very popular with internet is very much because of web technologies and their platforms. But all these benefits come with some challenges. Here the main challenge to manage interactive web application is web security.

A security evaluation of application defense center, which had more than 250 e-commerce applications, online banking and the corporate sites came up with a statement that more than 85% of web applications are vulnerable to attacks (Kumar, Mahajan, Kumar, & Khatri, 2017).

There are various types of attack which are directly or indirectly associated with the web security. Some of the attacks are unrestricted input fields allowing any type of data to enter generally called as injection attacks, cross-site scripting, cookie hijacking and accessing the session data if that web application has session handling through cookies, malicious script running on the browser to avail scripting logics and providing backend access with the same called as browser attacks, less secure object references which includes access to internal objects like files, directories, database values etc. This paper will be discussing all these attacks in details with their impacts on different type of web applications. Also, the paper will focus on the different prioritizing parameters in reference with the open web application security project (OWASP).

Description

In this paper, the focus is majorly on detecting and understanding these security bundles in detail dividing their minimization effect on the type of the application. After detailing each the other major purpose is to prioritize them to remediate the impact of these security vulnerabilities on various dynamic web applications.

For the first step of the paper, I will analyze various types of web applications like applications used for e-commerce, financial web applications, social media applications, real-time application with constantly changing data like stock market applications, online gaming applications etc. After playing around with various web applications, I will list down some of the security vulnerabilities which are most important in each of the application.

In the second phase, I will survey different people and understand their point of view for all these applications and what they look around when they register or use any application. I will identify their security concerns and expectations from the web application. The people can be a regular customer of that application, a web developer who develops such application or business analyst who identifies the requirement of such applications before developing them.

And the last step of the paper will include the prioritization parameters for all the listed security vulnerabilities which needs to be handled while developing any web applications from the design till the development and testing phase.

Goals

The main goal of this paper is to put together all the security vulnerabilities for different web applications and prioritize them according to their remediation factor.

Literature Survey

Rafique, Humayun, Hamid, Abbas, Akhtar, & Iqbal, 2015) summary article studied a systematic mapping to view and understand web applications implications and their vulnerability detection over the years with the parameters of OWASP Top 10 security vulnerabilities. In this project, they discussed their findings regarding the new advancement in the web security vulnerabilities and analyzing each individual method of detection for the same.

Kumar, Mahajan, Kumar & Khatri, (2017) in their paper reviews several parameters of web security: 1. Reasons for the Occurrence of the Attacks 2: Method for the detection of the attacks 3: Approaches or mechanism to minimize these security vulnerabilities. In today's era lot of website gets hacked by anonymous individual. And at some point, they get success too in breaching this privacy of the application because of the open-ended pain points in the applications.

Huang, Zhang, Cheng, & Shieh, (2017) in their paper discussed about countermeasure of these security attacks and they have developed a mechanism for secure implementation and defense deployment using firewalls in the web applications. The paper also discussed some of the methods for testing like automatic penetration and manual penetrations to evaluate the user inputs and locate the security breaches in the applications. The paper clearly mentions that amount of security threat soon is going to increase with discovery of new innovations. The vulnerabilities level is going rise every year and mechanism are getting developed to minimize the effects, but prioritization plays a very important role in handling such attacks and challenging the hackers. Test coverage of the application serves a good entry point to take preventive measure against all these security risk.

From the literature survey, I understood the impact of these web vulnerabilities and a need of strong mechanism for detection, analysis, prevention and prioritization of these security issues

in the web applications. When there are more clients associated with the application it is very important to secure their session and develop a faster pattern to reduce the attacks majorly SQL injections and cross-site scripting requests.

Technical Aspects

Before understanding the vulnerabilities of the web application, we will understand the basic components of the web applications and their terminologies.

Web Security Terminologies

Client. Client for the applications are the one who uses and interacts with the application. In web application the client connects with the application through web browser which makes request for a page whenever accessing the website from the server and expects an appropriate and relevant response from the web servers. There can be N number of middleware between client and the web server which generally client is unaware of.

Attacker. Attacker is an unauthorized user. Typically, this kind of attacker would be a proficient programmer or engineer with enough technical knowledge to understand the weak points in a security system. (Kumar, Mahajan, Kumar, & Khatri, 2017)

Web server. A web server is another application that receives client request, processes the input given by the client and responds to that request in the form of web pages which gets translated by their web browser. The protocol used to communicate between client and server is called as Hypertext transfer protocol (HTTP).

After understanding the common terms for the web application, we will perform a detailed study on the type of security vulnerabilities, how does it impact and what kind of the applications are majorly influenced by such security breaches.

Types of Attacks

Application Vulnerabilities. A system's weakness or flaws always leads to exploitation of the security in any application. Once if the attacker finds the application pain points or vulnerabilities and understands the execution of the application, there are high chances of exploitation of the application vulnerabilities leading to cyber security issues called as cyber-crime. These crimes target the confidentiality, integrity, or availability (known as the "CIA triad") of resources possessed by an application, its creators, and its users. (Application security vulnerabilities: Code flaws retrieved from https://www.veracode.com/security/application-vulnerability.)

Buffer overflow. This is a security vulnerability which occurs when client request for more data which then added to the buffer with less capacity then the incoming data. The buffer is a sequential storage block for memory allocations in the web applications. The attacker purposefully overflows the buffer if not handled in a proper way leading to system crash or unexpected behavior of the application. These types of vulnerabilities can be handled with the software code and proper testing by constantly checking and fixing the buffer. Some of the languages also support exceptional handling to avoid the buffer overflow.

Credentials Hijacking. This is a type of attack which breaches the username/password of the client and gets the full access to the user accounts. Once the attacker enters the system, it can change any data, delete any important information or use that information for transactions, insert malicious script to access database via the web server and steal secure and critical information. In four ways attacker can perform credentials hijacking:

Unprotected Storage. Occurs when passwords are stored in browser local storage or cookies in a plain-text format.

Hard-coded Password. Occurs when an application security is very poor and hard-coded passwords are used on the client request for authentications or while communicating or redirecting authentication to some third-party applications.

Insecure Protection policies. Occurs when a very poor encryption algorithm are used to protect the password or send in a channel to the web server. The protection policies may result in hijacking of information showing unauthorized retrievals.

Hard-coded Cryptographic Keys. Occurs when the key used to encrypt and decrypt the passwords are hard-coded and easily accessible to the client for the conversions. Once the cryptographic key is available to the attacker, the attacker can easily encrypt the credentials permitting to access the user accounts.

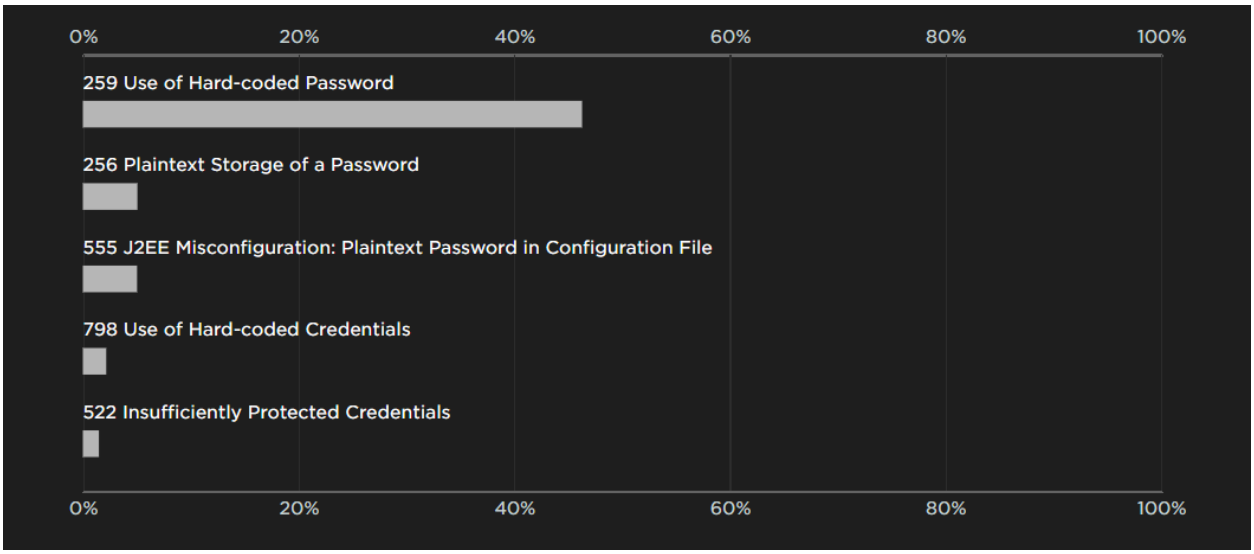

Figure 1. Credential Hijacking Incident Comparison

Note. The comparison analysis of the credential hijacking in the figure is from Common Application Vulnerabilities: Retrieved from https://www.veracode.com/security/web-application-vulnerabilities

Cross-Site Request Forgery. This is an attack listed in OWASP Top 10 vulnerabilities in which an attacker sends a request from a malicious website to the authenticated website a user already visited and stored their credentials on the browser for the autologin. Since the request is automatically routed to the authenticated target website through browser, the web applications show successful authentications and provide full access of the applications which can be dangerous.

Cross-Site Scripting (XSS). This is the most popular attacks on the web application as it deals with client-side machine and browser. In any web applications, some scripts are embedded on the page and executes whenever browser runs that application or when the page is loaded. The attacker can access those scripts and can modify the code causing malicious insertion of the threat. XSS is a threat which lowers down the strength of the web application in terms of internet security.

Failure to Restrict URL Access. These attacks deal with the unrestricted URL on the web application and allowing access to each URL to all the users. Security can be compromised if an attacker retrieves sensitive data from the web browser requesting web pages or files through forced browsing. Using a brute force method, attacker can fairly access the pages which are unprotected and contains sensitive information. This generally happens when there is a lack of access control on the web server side and lack of role-based policies.

Directory Traversal. Some of the application allows access to the client directory structure to select some files or folder without any restrictions on the upload/downloads. The attacker can upload any kind of virus or unwanted scripts on the computer causing system crash or unexpected shutdown. Attacker also gets the access of the private files and folders through directory access.

There are two security mechanisms that web servers use to restrict user access: root directory and Access Control Lists (ACLs). The root directory is the top-most directory on a server

file system. User access is confined to the root directory, meaning users are unable to access directories or files outside of the root. Administrators use Access Control Lists to define user access rights and privileges for viewing, modifying and executing files. (Application security vulnerabilities: Code flaws retrieved from https://www.veracode.com/security/application-vulnerability)

SQL Injection. This is again a most common attacks and this occurs due to careless nature of the developer coding the web application. Since web applications are open-ended to the customer, all the input fields accepting from the end users must be protected and restricted with some validations. If not protected, then attacker can enter any scripts or SQL database command in the input fields which leads to database violations or sometimes alters the data in the database.

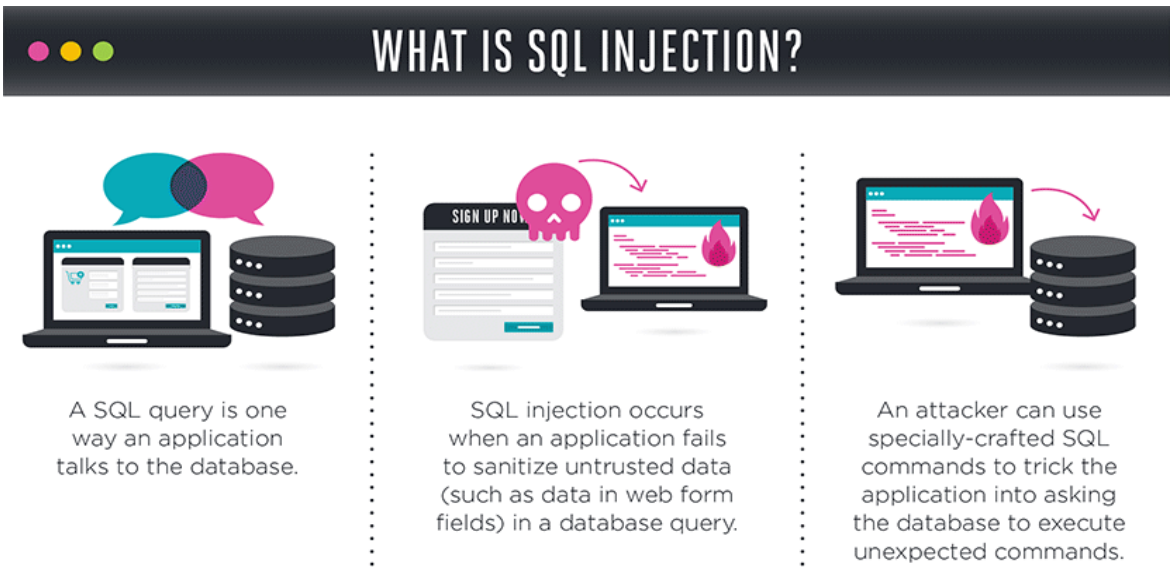

Figure 2. SQL Injection Flow diagram

Note. The flow diagram in the figure is from Common Application Vulnerabilities: Retrieved from https://www.veracode.com/security/web-application-vulnerabilities

LDAP Injection. This attack occurs when user credentials are not protected, and no security policies are applied on the LDAP server. The attacker may change the construction of the

user credentials when they are not sanitized properly on the webserver side. This can lead to serious threats if no right policies are applied on the LDAP tree. As an example, attackers can retrieve all the usernames and credentials stored on the LDAP server by executing some malicious code while sending data onto that server.

Insufficient Transport Layer Protection. This is a type of attack where web security layer is not protected to manage network traffic. While authentication and authorization checking, applications generally used secure socket layer (SSL) to provide a secure and encrypted communication with web server, but some applications often fails to secure the web application at other part or further in the system exploiting all the other data along with session-IDs. Intercepting these data can cause a very serious vulnerabilities providing access to all sensitive data.

Error Handling. This type of attacks occurs when the application shows the entire stack trace or error logs on the user interface. Error handling is an important part of any applications since it provides a meaningful insight of the application status at different times. But when these insights are displayed on the web client then attackers can easily misuse these error state of the applications exploiting the web application. The attacker easily gets the process of accessing the application and navigation patterns which may lead to exposure of sensitive data and information including session and passwords of the users.

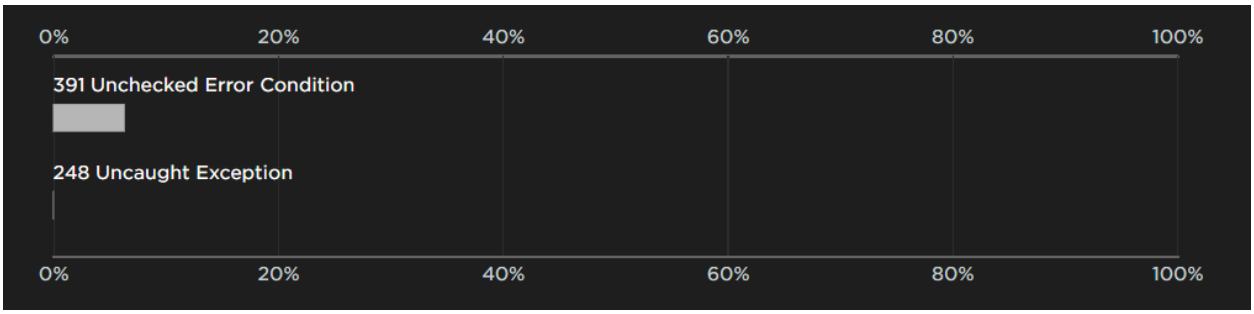

Figure 3. Error Handling Incident Comparison

Note. The error handler percentage comparison in the figure is from Common Application Vulnerabilities: Retrieved from https://www.veracode.com/security/web-application-vulnerabilities

Insecure Cryptographic Storage. This type of attacks occurs when sensitive information is not stored or communicated securely. Many times, developers assume that the storage data will not be accessed or analyzed by the end users. But then misconceptions lead to such type of vulnerabilities. To prevent from such attacks, some basic steps like making sure data is transported in a secure channel and stored in an encrypted and secured environment with right privacy policy controls, making sure the algorithms used are appropriate for that information avoiding security hijack. Also make sure you don't implement your own cryptographic algorithm and use if not tested properly. It is very easy for the attackers to alter the sensitive data and access system information.

Prevention Mechanism Against Web Security Vulnerabilities

Web application firewall (WAF)

A software solution to protect against the application security threats. Firewalls are extensively used to restrict the incoming and outgoing traffic accessing the application. We can configure which incoming traffic should be allowed to enter the system and which traffic needs to be blocked. This provides an initial layer of web security reducing sanitization deficiencies.

Currently many cloud providers like Amazon Web Services (AWS), Google cloud etc. provides this layer of security when configuring application on their platform. There is concept of virtual private cloud in which there are two types of security layer: Public subnet and private subnet. Whenever your application is in private network only restricted users can access the application. It has in-built firewall mechanism with appropriate access control list to block network traffic.

XSS Attacks

The cross-site scripting attacks usually occurs with the HTML and scripting contents on the web page. These type of attacks major impacts JavaScript and ActiveX. The attacker inserts a malicious script on the requested HTML page and hijacks important or sensitive data stealing the credentials from the user on the web page. XSS attacks also impacts cookies or browser storage if not handled properly. Such scripts can be stored on some remote servers and attackers provide links to such scripts, which looks harmless in the beginning. But once such links are invoked, scripts stored at those links runs at client side that may cause security breach. (Sonewar & Thosar, 2016)

Whenever developing any web application, individual developer must be responsible for understanding the outcome of the code and type of attack are possible on their developed code. There should not be any open-ended code written that accepts third party scripts to run the web browser. If you need to include this script, then it always preferable to manual download this scripts and cache into the system. There are two type of XSS attacks: Reflective XSS and Persistent XSS in which one occurs when injected malicious code and another is when malicious script is executed in the form of link which performs action once clicked. The important steps to avoid such attacks are: Escaping, Validating and Sanitizing user inputs. Penetration testing, also called white-hat evaluation, can help web developers discover and locate system vulnerabilities (Huang, Zhang, Cheng & Shieh, 2017)

SQL Injections

These attacks are the most commonly seen attacks in the web applications. Malicious code can easily be injected in the applications if the user inputs are not properly validated. The attacker injects harmful SQL query in the inputs which leads to alter in database and retrieval of sensitive

information through malicious database queries. There are various ways to prevent such injections on the applications. Some of them are:

Validate User Inputs. Never assume that the user inputs will always be right or expected for the box. Always consider user inputs to be wrong and keep it validated. Use functions to identify unwanted and dangerous characters and either flush them or restrict to move further in the system. Sanitization is the best option to filter out the content. Some examples like phone number must contains numbers and some special characters if entered other than show validation message and not move them forward, similarly email have some specific character restriction with fixed format etc.

Avoid using direct user input on SQL queries. The SQL queries are very sensitive when executing database operations. If user inputs any harmful character, database can be on risk. Hence always construct the SQL queries with formats like Prepared Statement which sanitize the input in a specific format and then fill ups the user inputs.

Access Privileges on the Database. To perform operations that alters the construction of the database like delete, update etc. must have access privileges. Only user with some administrator rights can perform these operations. Using such access control mechanism not all the user can perform dangerous and harmful operations. Even if attackers change the user inputs the database will not allow such operation coming from the respective user and the application will be still in a stable stage.

Proposed Vulnerability Prioritization

Vulnerability prioritization is important to understand the effects of these attacks on the individual applications based on their respective domains. Various Hybrid approaches are recommended to incorporate these security vulnerabilities.

The first approach would be to target the number of attacks per application and prioritizing them with the most impact on the application. The most vulnerable attack would be from malware/virus, ransomware, worms and various system impacted threats.

The second approach would be to prioritize based on the criticality of the vulnerability in the application. The example would be based on the exploitation level, impact, easily identifiable on the application etc.

The third approach would be to prioritize based on the vulnerability which is directly or indirectly associated with the critical aspects of the application. The one associated directly will be given the maximum priority than the indirection one and last the less critical assets associations.

According to the previous study of all the vulnerabilities, the second approach is generally the most suitable one and recommended. For every application, identify the various requirements while designing. The applications which are more real time and requires user interactions and contains numerous third-party libraries requires maximum number of high-to-medium vulnerabilities to be fixed. While the financial applications contain the similar priority. But the one which deals marketing applications need not require high prioritization. They can prioritize the attacks in the medium-to-low ranges.

According to the report in (Not All Vulnerabilities Are Created Equal Retrieved from https://www.veracode.com/blog/managing-appsec/not-all-vulnerabilities-are-created-equal) there are some outlines which addresses these vulnerabilities:

Remediation The solutions/fixes of the vulnerabilities made with understanding the codebase and changing the code to minimize the threat controls. Sometimes configuration changes also impact the system like instead of hard-coded environment configuration it is always preferable to use

environment variables to access global security keys. Patching is also one of the ways in a tightly couple application. Writing extra layer of logic provides security from the threats.

Mitigation Whenever the coding environment is not in the control, then some layer of security like firewalls to allow/block network traffic, concept of private subnets, role-based access controls/privileges on the database are used to minimize the effect of vulnerability on the applications.

Acceptance For some of the applications, it is not possible to apply any of the above approaches to reduce the impact of vulnerabilities. In that case, acceptance risk is a suitable option if approved by the Information security team of the organization. But constant monitoring and analysis is required in such case to identify the threats appearance.

The Open Web Application Security Project analyzed these security risk every 4 years and provides a top ten vulnerabilities found in the applications these days and needs to be handles with utmost priority.

OWAPS TOP 10 - 2007	OWAPS TOP 10 - 2010	OWAPS TOP 10 - 2013	OWAPS TOP 10 - 2017
A1 - Cross Site Scripting (XSS)	A1 – Injection	A1 – Injection	A1 – Injection
A2 - Injection Flaws	A2 – Cross Site Scripting (XSS)	A2 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Management
A3 - Malicious File Execution	A3 – Broken Authentication and Session Management	A3 – Cross-Site Scripting (XSS)	A3 – Sensitive Data Exposure
A4 - Insecure Direct Object Reference	A4 – Insecure Direct Object References	A4 – Insecure Direct Object References	A4 – XML External Entity (XXE)
A5 - Cross Site Request Forgery (CSRF)	A5 – Cross Site Request Forgery (CSRF)	A5 – Security Misconfiguration	A5 – Broken Access Control
A6 - Information Leakage and Improper Error Handling	A6 – Security Misconfiguration (NEW)	A6 – Sensitive Data Exposure	A6 – Security Misconfiguration
A7 - Broken Authentication and Session Management	A7 – Insecure Cryptographic Storage	A7 – Missing Function Level Access Control	A7 – Cross-Site Scripting (XSS)
A8 - Insecure Cryptographic Storage	A8 – Failure to Restrict URL Access	A8 – Cross-Site Request Forgery (CSRF)	A8 – Insecure Deserialization
A9 - Insecure Communications	A9 – Insufficient Transport Layer Protection	A9 – Using Components with Known Vulnerabilities	A9 – Using Components with Known Vulnerabilities
A10 - Failure to Restrict URL Access	A10 – Invalidated Redirects and Forwards (NEW)	A10 – Invalidated Redirects and Forwards	A10 – Using Components with Known Vulnerabilities

Figure 4. OWASP Top 10 Vulnerabilities from past years

Note. OWASP Top 10 Vulnerabilities is from OWASP Top 10-2017: The Ten Most Critical Web Application Security Risks, 2017 Retrieved from https://www.owasp.org/index.php/Top_10-2017_Top_10

Conclusion

Based on the vulnerabilities we identified by examining different type of web application, the impact of threats is severe and hence we discussed all those threats in a complete detailed form to understand how they are arising, what is the severity and how much harmful they are for the applications. This paper also discussed some of the preventive measures to avoid such type of attacks whether it is into codebase, transport layer or database layer. The vulnerability prioritization is very much crucial for the web application to take immediate steps for the high priority security threats and neglect some of the acceptance risk is easily monitorable.

It is rightly said in the paper, Generally, our logics and crucial codes resides at client side that is our browser that exposes programmer concepts. Thus, for attackers it becomes easy to intercept the logics and cause total damage to the server-side state of the application. and have always been an inspiration for achieving greater heights. (Kumar, Mahajan, Kumar & Khatri, 2017)

References

AppSec Knowledge Base: Application Vulnerabilities (n.d), [online] Available: https://www.veracode.com/security/web-application-vulnerabilities

Huang, H., Zhang, Z., Cheng, H., & Shieh, S. W. (2017). Web Application Security: Threats, Countermeasures, and Pitfalls. Computer, 50(6), 81-85. doi:10.1109/mc.2017.183

Kumar, S., Mahajan, R., Kumar, N., & Khatri, S. K. (2017). A study on web application security and detecting security vulnerabilities. 2017 6th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO). doi:10.1109/icrito.2017.8342469

Kumar, R. (2011). Mitigating the authentication vulnerabilities in Web applications through security requirements. 2011 World Congress on Information and Communication Technologies. doi:10.1109/wict.2011.6141435

OWASP Top 10-2017: The Ten Most Critical Web Application Security Risks, 2017. [online] Available: https://www.owasp.org/index.php/Top_10-2017_Top_10

Rafique, S., Humayun, M., Hamid, B., Abbas, A., Akhtar, M., & Iqbal, K. (2015). Web application security vulnerabilities detection approaches: A systematic mapping study. 2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD). doi:10.1109/snpd.2015.7176244

Sonewar, P. A., & Thosar, S. D. (2016). Detection of SQL injection and XSS attacks in three tier web applications. 2016 International Conference on Computing Communication Control and Automation (ICCUBEA). doi:10.1109/iccubea.2016.7860069

SQL Injection Bypassing WAF, 2016, [online] Available:

www.owasp.org/index.php/SQ_Injection_Bypassing_WAF.

Figures

Figure 1: Credential Hijacking Incident Comparison 10

Figure 2: SQL Injection Flow diagram 12

Figure 3: Error Handling Incident Comparison 13

Figure 4: OWASP Top 10 Vulnerabilities from past years 18