International Journal of Latest Engineering and Management Research (IJLEMR)
ISSN: 2455-4847
www.ijlemr.com || Volume 02 - Issue 09 || September 2017 || PP. 09-14

A Survey on Web Application Vulnerabilities

Mr. E. K. Girisan[1], Savitha. T[2]
Assistant Professor, Department of Computer Science, Sree Narayana Guru College, K.K Chavadi, Coimbatore, Tamil Nadu, India [1]
M.Phil Scholar, Department of Computer Science, Sree Narayana Guru College, K.K Chavadi, Coimbatore, Tamil Nadu, India [2]

Abstract: Web applications are one of the most widespread platforms for information and services delivery over Internet today. As they are increasingly used for critical services, web applications become a popular and valuable target for security attacks. Although a set of techniques have been developed to strengthen web applications and mitigate the attacks toward web applications, there is little effort devoted to drawing connections among these techniques and building a big picture of web application security research. This paper performs a comparative study in the area of web application security, with the aim of regulating the existing techniques into detailed analysis that promotes future investigation. We organize the list of security threats in the tabular format. And specifically, the paper considered 3 types of attack such as Cross Site scripting, Code, data and SQL injection attacks. In addition, the existing research works on the three attack categories are discussed. In the conclusion we summarize the lessons learnt and discuss the future research opportunities in this area.
Keywords: Web Security, Vulnerabilities, Code Injection attack, Cross Site Scripting attack, Internet security.

I. Introduction

Web application has become one of the most important communication channels between various kinds of service providers and clients. It has evolved from a static medium, with user interaction limited to navigation between web pages, to a highly interactive dynamic medium performing concurrent transactions and serving up personalized content. This dynamic medium application offers a wide range of services, such as on-line stores, e-commerce, social network services [1], etc. As more and more services are provided via the World Wide Web, efforts from both academia and industry are striving to create technologies and standards that meet the sophisticated requirements of today's enterprise Web applications and users. The following are the top vulnerabilities commonly seen in web applications [2].

Table 1: Top Web application vulnerabilities

Vulnerabilities	**Description**
Cross Site Scripting (XSS)	XSS flaws occur whenever an application takes user supplied data and sends it to a web browser without first validating or encoding that content. XSS allows attackers to execute script in the victim's browser which can hijack user sessions, deface web sites, possibly introduce worms, etc. Cross-Site Scripting (XSS) is an attack technique that involves echoing an attacker-supplied code into a user's browser instance.
Injection Flaws	Injection flaws are the common vulnerability in the web applications, this occurs when the user suppliers query and command as input to the interpreter. The attacker's hostile data tricks the interpreter into executing unintended commands or changing data.
SQL Injection	SQL Injection is an attack technique used to exploit applications that construct SQL statements from user-supplied input. When successful, the attacker is able to change the logic of the SQL statements executed against the database
SSI Injection	SSI Injection (Server-side Include) is a server-side exploiting technique that allows an attacker to code into a web application, which will later be executed locally by the web server.
XPath Injection	XPath Injection is an attack technique used to exploit applications that construct XPath (XML Path Language) queries from user supplied input to query or navigate XML documents.
Malicious File Execution	Code vulnerable to remote file inclusion (RFI) allows attackers to include hostile code and data, resulting in devastating attacks, such as total server compromise. Malicious file execution attacks affect PHP, XML and any framework, which

	accepts filenames or files from users.
Insecure Direct Object Reference	A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, database record, or key, as a URL or form parameter. Attackers can manipulate those references to access other objects without authorization.
Cross Site Request Forgery (CSRF)	A CSRF attack forces a logged-on victim's browser to send a pre-authenticated request to a vulnerable web application, which then forces the victim's browser to perform a hostile action to the benefit of the attacker. CSRF can be as powerful as the web application that it attacks.
Information Leakage and Improper Error Handling	Applications can unintentionally leak information about their configuration, internal workings, or violate privacy through a variety of application problems. Attackers use this weakness to steal sensitive data, or conduct more serious attacks.
Broken Authentication and Session Management	Account credentials and session tokens are often not properly protected. Attackers compromise passwords, keys, or authentication tokens to assume other users' identities.
Insecure Cryptographic Storage	Web applications rarely use cryptographic functions properly to protect data and credentials. Attackers use weakly protected data to conduct identity theft and other crimes, such as credit card fraud.
Insecure Communications	Applications frequently fail to encrypt network traffic when it is necessary to protect sensitive communications.
Failure to Restrict URL Access	Frequently, an application only protects sensitive functionality by preventing the display of links or URLs to unauthorized users. Attackers can use this weakness to access and perform unauthorized operations by accessing those URLs directly.
Brute – Force Attack	A brute force attack is a method to determine an unknown value by using an automated process to try a large number of possible values
Insufficient Authentication	Insufficient Authentication occurs when a web site permits an attacker to access sensitive content or functionality without having to properly authenticate.
Weak Password Recovery Validation	Insufficient password recovery is when a web site permits an attacker to illegally obtain, change or recover another user’s password. Password recovery systems may be compromised through the use of brute force attacks, inherent system weaknesses, or easily guessed secret questions.
Credential / Session Prediction	Credential/Session Prediction is a method of hijacking or impersonating a web site user. If an attacker is able to predict or guess the session ID of another user, fraudulent activity is possible.
Insufficient Authorization	Insufficient Authorization results when an application does not perform adequate authorization checks to ensure that the user is performing a function or accessing data in a manner consistent with the security policy
Insufficient Session Expiration	Insufficient Session Expiration occurs when a Web application permits an attacker to reuse old session credentials or session IDs for authorization.
Session Fixation	Session Fixation is an attack technique that forces a user’s session ID to an explicit value.
Content spoofing	Content Spoofing is an attack technique that allows an attacker to inject a malicious payload, that is later misrepresented as the legitimate content of a web application.
Buffer Overflow	A Buffer Overflow is a flaw that occurs when more data is written to a block of memory, or buffer, than the buffer is allocated to hold
Format String Attack	Format String Attacks alter the flow of an application by using string formatting library features to access other memory space.
Light Weight Directory Access Protocol (LDAP) Injection	LDAP Injection is an attack technique used to exploit web sites that construct LDAP statements from user-supplied input
OS Commanding	OS Commanding is an attack technique used for unauthorized execution of operating system commands.
Directory Indexing	Automatic directory listing/indexing is a web server function that lists all the files within a requested directory, if the normal base file is not present.
Path Traversal	The Path Traversal attack technique allows an attacker access to files, directories, and commands that potentially reside outside the web document root directory
Predictable Resource	Predictable Resource Location is an attack technique used to uncover hidden web

Location	site content and functionality.
Information Leakage	Information Leakage is an application weakness where an application reveals sensitive data, such as the technical details of the web application, environment, or user-specific data.
Abuse of Functionality	Abuse of Functionality is an attack technique that uses a web site's own features and functionality to attack itself or others.
Denial of Service	Denial of Service (DoS) is an attack technique with the intent of preventing a web site from serving normal user activity. DoS attacks, which are easily normally applied to the network layer, are also possible at the application layer
Insufficient Anti-Automation	Insufficient Anti-automation occurs when a web application permits an attacker to automate a process that was originally designed to be performed only in a manual fashion
Insufficient Process Validation	Insufficient Process Validation occurs when a web application fails to prevent an attacker from circumventing the intended flow or business logic of the application.

Network Security is a fundamental form of security for Internet transmissions for blocking or filtering of data from unknown or suspect sources [3][4]. There are many approaches and techniques were developed to solve the above attacks. The common way to accomplish this is by restricting the number of open communication ports on the server, limiting inbound transmissions to those protocols supported by the few open ports. In a network, a firewall [5] can help to prevent hackers or malicious software (such as worms) [6] from gaining access to other computers through a network or the Internet. The difficulty for firewalls is distinguishing between legitimate and illegitimate traffic. If firewalls are configured correctly, they can be a reasonable form of protection from external threats, including some Denial of Service (DoS) attacks [7]. Regardless of the approaches taken to identify and protect against attacks from outside the network, there remains a larger and perhaps more insidious threat that can occur from within the network itself. So, securing the host becomes an important task within an organization.

II. Literature Survey

A. Cross - site scripting:

Among the many attacks on Web applications, cross-site scripting (XSS) is one of the most common [8]. An XSS attack involves injecting malicious script into a trusted website that executes on a visitor's browser without the visitor's knowledge and thereby enables the attacker to access sensitive user data, such as session tokens and cookies stored on the browser. With this data, attackers can execute several malicious acts, including identity theft, key-logging, phishing, user impersonation, and webcam activation. The typical scenario of a persistent cross-site scripting (XSS) attack is shown in Fig 1.0. Each time a user visits a webpage injected with malicious script, the stored script exploits the user's browser privileges to access sensitive information. Cross site scripting is categorized into three types like **Persistent XSS, Non- Persistent XSS and DOM-based XSS.**

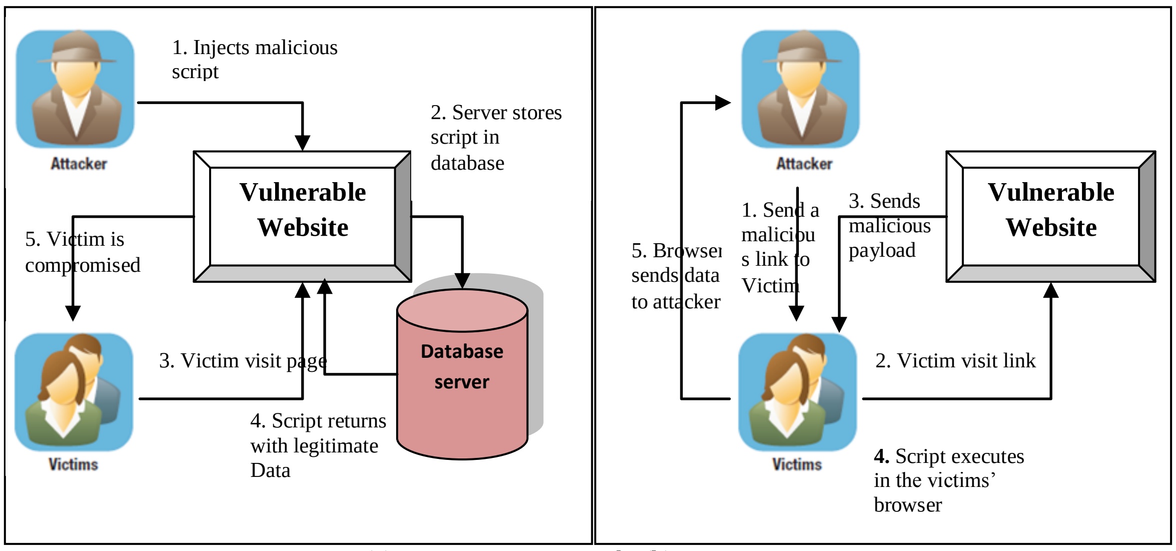
Attacker
1. Injects malicious script
2. Server stores script in database
Vulnerable Website
5. Victim is compromised
3. Victim visit page
Database server
4. Script returns with legitimate Data
Victims
Attacker
1. Send a maliciou s link to Victim
3. Sends malicious payload
Vulnerable Website
5. Browser sends data to attacker
2. Victim visit link
Victims
4. Script executes in the victims' browser

Fig 1.0 (a) Persistent XSS attack, (b) Non-persistent XSS

Persistent XSS is often difficult to detect and is considered more harmful than the other two attack types. Because the malicious script is rendered automatically, there is no need to target individual victims or lure them to a third-party website [9].

The next type is the non-persistent, or reflected, XSS attack, which occurs when a website or Web application passes invalid user inputs. Usually, an attacker hides malicious script in the URL, disguising it as user input, and lures victims by sending emails that prompt users to click on the crafted URL. When they do, the harmful script executes in the browser, allowing the attacker to steal authenticated cookies or data.

DOM-based, or Type-0, XSS attack executes in the same manner as a non-persistent XSS attack. To update the structure and style of webpage content dynamically, all Web applications and websites interact with the DOM, a virtual map that enables access to these webpage elements. Compromising a DOM will cause the client-side code to execute in an unexpected manner.

Recent Works on Cross site scripting:

There are several approaches and techniques were introduced to prevent, detect and mitigate the XSS attacks. The detections techniques are statistics based, dynamic based and hybrid based. In statistics based approaches includes the analysis of string, bounded model checking, software testing approaches, Taint approaches, and using unstructured scripts. The dynamic approaches include Browser-Enforced Embedded Policies Approach, Syntactical Structure Approach, Proxy-based Approach and Interpreter-based Approaches. In the prevention mechanisms falls under client side, server side and hybrid process, which includes firewall, testing tools, and others.

In the paper [10], authors focused on the detection and prevention of stored XSS and DOM based XSS. The authors reviewed and concluded that there is very less possibility to detect the XSS, so the author recommended prevention technique using hierarchical approach. This technique applies the security testing using black box and white box in the initial level, the second level in the hierarchy utilized the XSS testing tool Burp. The attack is prevented when the author used the first level. However, the prevention is not fully performed in this paper.

B. Code injection including SQL injection:

Code injection attacks are more common and widespread attack in the web applications. The attacker spreads some vulnerable program on the target system and then performs the data control in the memory layout. As like code injection SQL injection is a technique which passes the SQL queries to access or destroy the database.

Web applications work with a back-end database to store or retrieve data. Many Web applications store the data in the data base and retrieve or update information as needed. These applications are highly vulnerable to many types of attacks, one of them being SQL injection Attacks (SQLIA). Input data given by the user is dynamically embedded into the HTML code and sent to the server as a request. The server processes this request and provides the response back to the user. If this input data is malicious, then this input will in some way try to extract sensitive information from the database. This type of attack where the database is compromised using SQL statements is called the SQL injection attack (SQLIA). An SQL injection attack occurs when an attacker causes the web application to generate SQL queries that are functionally different from what the user interface programmer intended.

```
Var UserID;
UserID= Request.form ("UserID");
var InfoUser = "select * from UserInfo where UserID = '" + UserID + "'";
```

If the user fills the field with correct information of his UserID (F827781), after the script execution the above SQL query will look like

```
SELECT * FROM UserInfo WHERE UserID = 'F827781'
```

Consider a case when a user fills the field with the below entry.

```
F827781; drop table UserInfo--
```

After the execution of the script, the SQL code will look like

```
SELECT * FROM UserInfo WHERE UserID = ' F827781';drop table UserInfo--
```

This will ultimately result in deletion of table UserInfo.

Recent Works on Injection attacks:

In paper [12], authors proposed a fine grained randomization based approach to mitigate a code injection based attack, which is named as return oriented programming (ROP), this ROP changes the sequence of instructions. The authors mitigated the code reuse attacks using random shuffling of function blocks in the target binary. This approach could effectively deny the attacker. The detection and prevention of such attacks has different types of characteristics. However, the Marlin randomized prototype is difficult and unable to perform certain binary rewriting. Authors in [13] developed a *Polynomial-based Compromise-Resilient En-route Filtering scheme (PCREF)* for cyber physical network system. This technique can filter false injected data easily. The technique proposed with high resilience against code injection attacks. The approach finds the compromised nodes and performs node localization. Polynomials are used to verify the reports, and this is capable on mitigating node impersonating attacks. The technique achieved better filtering results. The authors proved that the scheme resilience to a large number of compromised nodes in comparison with the existing schemes. However, the PCREF incurs high verification overhead when comparing with the other technique.

In the paper [14], a detection scheme for stealthy false data injection attack is proposed. Unlike earlier works, the authors concentrated on the intelligent attackers, where they can design a sequence of data injection into the sensors. This process was performed by the attacker to be undetected. So, authors developed a coding matrix to convert the real sensors outputs to improve the performance under code and data injection attacks. Authors proposed a heuristic algorithm to decide the time interval. Based on the time interval, the system updates the matrix. This work is the recent one which also performed in the sensor networks. But the authors failed to explore the coding scheme for structural constraints.

The various techniques used to prevent SQL injections such as Parameterized query, Stored procedure, Regular expression to discard input string, Quoteblock function, Don't show detailed error messages to the user concepts. And Have a less privileged user/role of your application in database. Authors in [15] implemented a system named, WebSSARI, that detects input-validation-related errors using information flow analysis. In this approach, static analysis was used to check taint flows against preconditions for sensitive functions. One technique to detect when tainted input has been used to construct an SQL query has been proposed by Authors in [16] proposed information flow technique. It gets the vulnerability specifications from the user and uses this as static analyzer point. This technique detects SQLIA, XSS and Hyper Text Transfer Protocol (HTTP) splitting attacks. An approach called SQLRand based on instruction-set randomization was developed [16]. It allows developers to create queries using randomized instruction instead of normal SQL keywords. This approach specifically designed for Common Graphical Interface (CGI) application. To retain the portability and security, a reverse proxy was used. This proxy-filter intercepted the queries to the database and de-randomized the keywords.

In paper [17], authors implemented a system named StringBorg, that breaks the barriers in cross language programming, where one language needs to construct sentences in another language. This system embeded the grammar of the guest language into the host language. For example, if the guest language is SQL and the host language is Java, then the construction of SQL query can be done by Java, whenever needed. It used an Application Programming Interface (API) to build sentences, and this process is called construction. It is very helpful for the programmers without good programming experience. The limitation of this system is that it cannot detect Semantic injection attack.

Conclusion

The main idea for the future research work focuses on detecting and mitigating the SQL injection attacks and cross site scripting attacks on Web Application. To materialize this, the challenges and the issues concerning the attacks were reviewed and studied in this paper. Several strategies were studied in literature, which will be effectively analyzed to identify the vulnerable points in the code and to detect and mitigate an attack. The experiment results showed significant improvement. In this paper, we discussed the general approaches to detect the vulnerabilities in web applications and the existing solutions for injection vulnerabilities in web applications and its limitations.

References:

[1]. Leon Shklar and Rich Rosen, "Web Application Architecture, Principles, Protocols and Practices", Second Edition, Wiley Publication, 2012.
[2]. Stuttard, Dafydd, and Marcus Pinto, "The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws", John Wiley and Sons, 2008.
[3]. Munro, Ken, "Android Scraping: Accessing Personal Data on Mobile Devices", Network Security, Vol. 2014, no. 11, pp. 5-9, 2014.
[4]. W. Halfond, J. Vigeas and A.Orso, "A Classification of SQL Injection Attacks and Counter Measures", IEEE International Symposium on Secure Software Engineering, vol. 1, pp. 13-15, 2006.
[5]. Puppy, Rain Forest, "NT web technology vulnerabilities", Phrack Magazine, Vol. 8, no. 54, 1998.
[6]. Litchfield, D., "Data-mining with SQL Injection and Inference", White Paper, Next Generation Security Software Ltd., 2005.

[7]. Huang, Y.-W., Huang, S.-K., Lin, T.-P., & Tsai, C.-H., " Web Application Security Assessment by Fault Injection and Behavior Monitoring", 12th International ACM Conference on World Wide Web, pp. 148-159, 2003.

[8]. Gould, C., Su, Z., & Devanbu, P. T., "JDBC Checker: A Static Analysis Tool for SQL/ JDBC Applications", 26th International Conference on Software Engineering, pp. 697–698, 2004.

[9]. Huang, Y.-W., Yu, F., Hang, C., Tsai, C.-H., Lee, D.-T., & Kuo, S.- Y, "Securing Web Application Code by Static Analysis and Runtime Protection", 13th International Conference on World Wide Web, pp. 40–52, 2004.

[10]. Shrivastava, Ankit, Santosh Choudhary, and Ashish Kumar. "XSS vulnerability assessment and prevention in web application." *Next Generation Computing Technologies (NGCT), 2016 2nd International Conference on*. IEEE, 2016.

[11]. Ahmed, Moataz A., and Fakhreldin Ali. "Multiple-path testing for cross site scripting using genetic algorithms." *Journal of Systems Architecture* 64 (2016): 50-62.

[12]. Gupta, Aditi, Javid Habibi, Michael S. Kirkpatrick, and Elisa Bertino. "Marlin: Mitigating code reuse attacks using code randomization." *IEEE Transactions on Dependable and Secure Computing* 12, no. 3 (2015): 326-337.

[13]. Yang, Xinyu, Jie Lin, Wei Yu, Paul-Marie Moulema, Xinwen Fu, and Wei Zhao. "A novel en-route filtering scheme against false data injection attacks in cyber-physical networked systems." *IEEE Transactions on Computers* 64, no. 1 (2015): 4-18.

[14]. Miao, Fei, Quanyan Zhu, Miroslav Pajic, and George J. Pappas. "Coding schemes for securing cyber-physical systems against stealthy data injection attacks." *IEEE Transactions on Control of Network Systems* 4, no. 1 (2017): 106-117.

[15]. Livshits, Benjamin V., and Monica S. Lam, "Finding Security Errors in Java Programs with Static Analysis", Technical Report, 2005.

[16]. Boyd, Stephen W., and Angelos D. Keromytis, "SQLrand: Preventing SQL Injection Attacks", Applied Cryptography and Network Security, pp. 292-302. Springer Berlin Heidelberg, 2004.

[17]. Bravenboer, Martin, Eelco Dolstra, and Eelco Visser, "Preventing Injection Attacks with Syntax Embeddings", 6th ACM International Conference on Generative Programming and Component Engineering, pp. 3-12, 2007.