IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 7, Issue 5, May 2018

Systematic Review of Web Application Security Vulnerabilities

Miss. Rohini Padmakar Pawar

Software Test Engineer, Kanaka Software Consulting Private Limited, Pune, India

Abstract: Increasing web technologies in collaboration with changing business environment means that web applications are becoming more popular today in corporate, public and government services. It is essential to understand the vulnerabilities commonly found in web applications. The number of reported web application vulnerabilities is increasing dramatically. The most of vulnerabilities result from improper input or improper security layer implementation in web application development this paper presents a new approach to vulnerability analysis which incorporates the different security vulnerabilities in web application and how they occur and how we can prevent from attack.

Keywords: Web Application, Vulnerabilities analysis, SQL Injection, XSS, Insecure Direct Object Reference, Failure to restrict URL Access.

I. INTRODUCTION

Increasing web technologies in collaboration with changing business environment means that web applications are becoming more popular today in corporate, public and government services. Although web applications can provide convenience and efficiency, there are also many new security threats, which can potentially create significant risks to an organization's information technology infrastructure if it is not done properly, more than a decade By the time, the organization's role to protect its IT infrastructure To protect the structure has been dependent on security measures on the periphery of the network. However, traditional network security measures and technologies are not enough to protect web applications from new threats because the attacks are now specifically targeting security flaws in the design of web applications. New security measures need to be implemented with the development of web applications for both technical and administrative to deal with the dangers related to these new application services, it is necessary to understand the vulnerabilities found in web applications. This article discusses important web application weaknesses and how they can be addressed during various stages of the System Development Life Cycle. Tips for surfing the internet safely are also provided to end users as these web apps may have the weakest link in information security.

II. VULNERABILITIES INWEB APPLICATION

In general, there are three kinds of security vulnerabilities among web applications at completely different levels: [1] input validation vulnerability at the single request level, [2] session management vulnerability at the session level and [3] application logic vulnerability at the extent of the whole application. In what follows, description of the above three kinds of vulnerabilities are presented and the common attacks that exploit these vulnerabilities.

A. *Cross-Site Scripting (XSS)*

XSS is one of the most common web applications layer attacks. The XSS vulnerability target scripts are embedded in a page that runs on client-side (in the user's web browser) instead of server-side. XSS is a threat that is brought into the client-side scripting languages by Internet security vulnerabilities, such as manipulating the concept of HTML and JavaScript XSS client-side script of the web application to execute the desired way by a malicious user. This kind of impact can embed a script in a page that which is executed every time the page is loaded in browser, or whenever ansupplementary event is done. XSS is the most common security vulnerability in the software today. This should not be the case because it is easy to find and fix XSS. The results of XSS vulnerabilities can be in the form of tampering and sensitive data theft. An XSS vulnerability arises when web applications take data from users and without dynamically incorporate it into web pages, without properly validating data XSS vulnerabilities allowed an attacker to execute arbitrary commands and display arbitrary content in the victim's user's browser. A successful XSS attack leads to an attacker controlling the account on the victim's browser or the weak web application. Although XSS is enabled by weaker pages in a web application, XSS is user of victim application of attack, not the application itself. The power of the XSS vulnerability lies in the fact that malicious code executes the context of the victim's session, so that the attacker can bypass the general security restriction.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 7, Issue 5, May 2018

1. Example Attack Scenarios: The application gives permission to user to submit a state-changing request that does not include anything secrete.
   For example: The application uses untrusted data in the construction of the following HTML snippet without validation or escaping

```
(String) page += "<input name='creditcard' type='TEXT' value='" + request.getParameter("CC") + "'>";
```

The attacker modifies the 'CC' parameter in their browser to:

```
'<script>document.location= 'http://www.attacker.com/cgi-bin/cookie.cgi ?foo='+document.cookie</script>'.
```

This causes the user's session ID to be sent to the attacker's website, allowing the attacker to hijack the user's current session[2].

2. Preventions for Attack: Preventing XSS requires separation of untrusted data from active browser content.

- The preferred option is to run all unreliable data properly on the basis of HTML references (body, attribute, JavaScript, CSS, or URL), which will be kept in data. For details on techniques to avoid essential data, see OWASP XSS Prevention Cheatsheet [2].
- Positive or "Whitelist" server-side input validation is also recommended as it helps to protect against XSS, but there is no complete security because many applications require special characters in their input. Before accepting the input, the validity of this type of verification should be, the length, character, format, and business rules on that data.
- For rich content, consider auto-sanitization libraries like OWASP's AntiSamy or the Java HTML Sanitizer Project[1].

3. Technical Business Impact

- When attackers succeed in exploiting XSS vulnerabilities, they can gain access to the account credentials. They can also spread webworms or reach the user's computer and users can see the business value of affected data or application functions. Do not imagine that the users are aiming to take these actions on the impact of your reputations Consider browser history or browser remotely after receiving control over the hunting system, attackers can also analyse and use other intranet applications. The attacking victims can victimize victims to execute the changing operation in any state, which is authorized to hunt, For Example: updating account details, making a purchase, log out and even login.

B. Insecure Direct Object References

Insecure Direct Object References occurs when an application delivers direct access to the object based on user-supplied input. As a result of this vulnerability attackers, bypassing the authority in the system and using resources, for example database records or files. In insecure direct object references, the attacker allows to bypass the authorization and access the resources by modifying the value of the parameters used to directly point to the object.[2] Such resources may have database entries related to other users, files in the system, and more. This is due to the fact that the app provides input to the user and uses it to recover an object without checking enough authorization. When preparing web pages, apps often use the real name or key of an object. Apps do not always verify that the user is authorized for the target object. This causes unprotected direct object reference defects. In order to detect such defects, the testers can easily manipulate the parameter values. Code analysis quickly identifies whether the authorization is verified properly or not.

1. Example Attack Scenario

```
http://test.com/changepassword?user=someusers
```

In the above case, the value of the user parameter is used to say the web application for which user it should change the password. In many cases, this step will be a part of a wizard or a multi-step operation. In the first step the application will get a request stating for which user's password is to be changed, and in the next step, the user will provide a new password (without asking for the current one)[1]. The user parameter is used to directly reference the object of the user for whom the password change operation will be performed. To test for this scenario of the case, the tester should try to provide a different test username than the one currently logged in user and check whether it is possible to change the password of another user.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 7, Issue 5, May 2018

```
http://test.bar/showImage?img=img045
```

In this case, the value of the file parameter is used to say the application what file the user intends to retrieve. By providing the name or identifier of a different file (for example file=image0045.jpg) the attacker will be able to retrieve objects or details belonging to other users.

2. Preventions for Attack:
Avoidinginsecure direct object references requires selecting a method for defending each user available object.

- Use indirect object reference per user or session.This prevents the attackers from directly targeting unauthorized resources. For example, instead of using the resource key of the resource, a drop-down list of six resources authorized for the current user, to use numbers 1 to 6, to indicate which value the user chose The application has to map back the non-user contextual reference to the actual database key on the server.
- OAWAS's ESAPI includes sequential and random access reference maps that developers can use to eliminate direct object references [1].Role Based access in use of a direct object reference from an untrusted source must include an access control check to ensure the user is authorized for the requested object.

3. Technical Business Impact:

- Such flaws can compromise with all data that can be referred by the parameter. Unless the object references are unpredictable, the attacker is easy to access all available data of that type. Consider the business value of the exposed data, and also consider the effect of the business of public exposure at risk.

C. ***Insecure Direct Object References***

SQL injection refers to an injection attack, in which an attacker malicious SQL statement (commonly referred to as a malicious payload) can execute a web application's database server (usually relational database management systems - RDBMS [3], Since SQL injection vulnerability can affect any website or web application Which uses SQL-based databases, vulnerability is the oldest, most prevalent and most dangerous of web application vulnerabilities By taking advantage of the SQL injection vulnerability, under the right conditions, an attacking web application can use it to bypass authentication and authorization mechanisms and retrieve the contents of the entire database. SQL injection can be used to add, modify and delete records in the database, thereby affecting data integrity. To such extent, SQL injection can provide access to sensitive data, including an invader, customer data, personally identifiable information (PII), trade secret, intellectual property and other sensitive information. SQL is a standardized language that is accessed and used by the database [3].SQL queries are used to execute commands for creating customizable data views for each user, such as data retrieval, updating and removing records, various SQL elements implement these tasks.

1. Example Attack Scenario
In order to run malicious SQL queries against a database server, an attacker must first find an input within the web application that is included inside of an SQL query. In order for an SQL Injection attack to take place, the vulnerable website needs to directly include user input within an SQL statement. An attacker can then insert a payload that will be included as part of the SQL query and run against the database server [3].

The following server-side pseudo-code is used to authenticate users to the web application.

```
# Define POST variables
uname = request.POST['UNAME']
passwd = request.POST['PASSWORD']

# SQL query vulnerable
sql = "SELECT id FROM tblUser WHERE username='" + UNAME + "' AND PASSWORD='" + passwd + "'"

# Execute the SQL statement
database.execute(sql)
```

The above script is a simple example of authenticating a user with a username and a password against a database with a table named tblUser, and a username and password column.The above script is vulnerable to SQL Injection because an attacker could submit malicious input in such a way that would alter the SQL statement being executed by the database server. A simple example of an SQL Injection attack payload could be somewhat as simple as setting the password field to password' OR 1=1.This would result in the following SQL query being run against the database server.

```
SELECT id FROM tbl_users WHERE username='username' AND password='password' OR 1=1'
```

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 7, Issue 5, May 2018

An attacker can also comment out the rest of the SQL statement to control the execution of the SQL query further.

```
-- MySQL, MSSQL, Oracle, PostgreSQL, SQLite
' OR '1'='1' --
' OR '1'='1' /*
-- MySQL
' OR '1'='1' #
-- Access (using null characters)
' OR '1'='1' %00
' OR '1'='1' %16
```

Once the query executes on the server, the result is returned to the application to be processed, resulting in an authentication bypass. In the event of authentication bypass being possible, the application will most likely log the attacker in with the first account from the query result — the first account in a database is generally of an administrative user [1].

2. Prevention for Attack:

- Preventing injection requires keeping untrusted data separate from commands and queries.
- The preferred option is to use a safe API which avoids the use of the interpreter entirely or provides a parameterized interface. Be careful with APIs, such as stored procedures that are parameterized, but can still introduce injection under the hood [1].
- If a parameterized API is not available, you should carefully escape special characters using the specific escape syntax for that interpreter. OWASP's ESAPI provides many of these escaping routines[1].
- Positive or "white list" input validation is also recommended but is not a complete defense as many applications require special characters in their input. If special characters are required, only approaches 1. And 2. Above will make their use safe. OWASP's ESAPI has an extensible library of white list input validation routines[1].

3. Technical-Business Impact

- SQL Injection can effect in data loss or fraud, lack of accountability, or denial of access. SQL Injection can sometimes lead to complete host takeover. Consider the business value of the affected data and the platform running the interpreter [1]. All data could be stolen, modified, or deleted. SQL is used to delete records from a database. An attacker could use an SQL Injection vulnerability to delete data from a database. Even if an appropriate backup strategy is employed, deletion of data could affect an application's availability until the database is restored [6].

D. Failure to Restrict URL Access

If your app fails to properly restrict access to URL access, security can be compromised through a technique called compulsory browsing.

1. Example Attack Scenario

Attackers can use a direct way to access and relate with hidden / unlinked pages on a website, the most common attack known as "forced browsing". Forced browsing attacks can take place when an attacker is able to correctly guess the URL of or use brute force to access an unprotected page. "If there is any flaw in this page's access control policy, then this process is very easy on the attacker."These flaws typically include hidden pages with guessable URLs, applications that permit access to pages that are meant to be hidden/restricted, outdated access-control policy code, and a lack of server-side access-control policy.

2. Preventions for Attack:

To prevent unauthorized URL access, it is essential for each page to select an approach to require proper authentication and proper authorization [1]."Frequently, such protection is provided by one or more components external to the application code. Regardless of the mechanism(s), all of the following are recommended:

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 7, Issue 5, May 2018

- The authentication and authorization policies are role-based, to minimize the effort required to maintain these policies.
- "The policy should be highly configurable, so that any difficult coded aspects of the policy can be reduced."
- "The enforcement mechanism should deny access to all by default, explicit access to specific users and roles are required for access to each page[1]."
- "If the page is included in a workflow, check to ensure that conditions are in the proper position to allow access."

3. Technical-Business Impact

- Such flaws can allow attacking some or all of the accounts. If successful, an attacker can do anything he can do.Privileged accounts are frequently targeted.

E. Clickjacking

A web framing attack knows as "clickjacking" uses a transparent iframe (HTML tags to specify an inline frame that is used to embed another document in the original HTML document) to hijack the users clicks[1]. In a typical clickjacking attack scenario, a malicious web page is constructed by an attacker in order to trick the victim into clicking on elements of the web page within an invisible iframe to perform unintended actions. Recently, click jacketing strikes have a lot of interest and many redressed techniques are proposed, however, it is still unclear whether these defence mechanisms have been implemented effectively or not. Clickjacking is a web-based attack that was first reported by Jeremiah Grossman and Robert Hansen in 2008. It is a technique in which the user is induced to click on an element of a web page which is designed by the attacker.[1]

1. Example Attack Scenario

As mentioned above, such attacks are often planned to allow user actions on a site targeted by an attacker site, even if the anti-anti anti-CSRF is being used. Therefore, for the CSRF attack, it is important to isolate the web pages of the targeted site that take input from the user. We have to know that if we are testing on the website, then there is no security against clickjacking attacks or if the developers have implemented certain types of security if these techniques are liable for bypass. Once we know that the website is weak, we can make "proof of concept" to take advantage of the vulnerability. The first step to finding out is that if a website is unsafe, it is to check whether the target web page can be loaded in the iframe. To do this, you need to create a simple web page that includes the frame with the target web page. The HTML code is displayed in the following snippet to create this test web page:

```
<html>
<head>
<title>Clickjack testing page</title>
</head>
<body>
<p>Website should be vulnerable to clickjacking!</p>
<iframesrc="http://www.target.site" width="500" height="500"></iframe>
</body>
</html>
```

Result Expected:
If you can see both texts "Website is unsafe for clicking!" At the top of the page and your targeted web page has been successfully loaded in the frame, your site is weak and there is no protection against click-linking attacks. Now you can make a "proof of concept" directly to demonstrate that an attacker can take advantage of this vulnerability.

2. Prevention for Attack:

Clickjacking can be prevented using a host of client-side browser plugins such as:

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 7, Issue 5, May 2018

- Send a header to the appropriate Content Protection Policy (CSP) Frame-Ancestors Guidelines, which instruct the browser to allow framing from other domains. (This replaces the old X-Frame-Option HTTP header.)
- This explains that plug-ins are recommended for daily browsing and can also protect users against additional client-side attacks, such as XSS (cross-site scripting). Below are plug-in client-side protection techniques that should be taught to all user applications; However, steps should also be taken from the developer's end[1].
    1. No Script – http://noscript.net
    2. Web Protection Suite – http://www.comitari.com/Web_Protection_Suite

3. Technical-Business Impact

- In potential risks of clickjacking and its underlying impact, it provides a moderate risk problem in most sensitive applications, such as financial or sensitive data handling applications. The reason why it is a medium risk and not a high-risk issue is down to the delivery method of attack and its execution vectors. This vulnerability requires a user's contact and as a result of the victims (generally more technically naïve), an element of social engineering has to voluntarily interact with the malicious page. An example of an attack on a financial application could consist of sending out emails to authenticated users of the application.

F.Input Sanitization and Validation

Input validation routines aid as the first line of protection for a Web application. Input verification attacks are where an attacker deliberately sends unusual input into the expectations of confusing the application.

1. Example Attack Scenario

- Attacker Enters Malicious Inputs such as: http://www.testbank.com/index.php?id= 1 UNION ALL SELECT creditCard_Number,1,1, FROM CreditCardTable by using this Attacker obtain other customers credit card information so Application sends that modified query to the database i.e. SELECT Emp_Name, MobileNo, Address FROM Users WHERE Id=1 UNION ALL SELECT creditCardNumber 1,1 FROM CreditCardTable, which executes it.

- By using following attack user also done the Input attack i.e. HTML Injection, File Injection Attack, Server Pages Injection Attack, Script Injection.

2. Prevention for Attack:

- Check input validation on every layer and when crossing trust boundaries and also validate the user input with the special character or HTML code.

3. Technical-Business Impact

It provides a moderate risk problem in most sensitive applications, such as financial or sensitive data handling applications. Or by using file injection attack hacker hack the system by uploading the .exe file or .bat file Strongly typed at always, Length validation checked and fields length minimized Range checked if a numeric, Unsigned unless required to be signed, Syntax or grammar should be checked prior to first use or inspection.

G Default Deny Principle:

The default deny means does not allow something to the user if they don't have any access to respective web area or something. Default deny security attack treats everything specific and not allowed as suspects. So in a web application it's very necessary to implement the role based privileges authentication means role wise user screen allocation or access.

1. Example Attack Scenario

- By using the following attack hacker may try to directly access the respective unauthorized URL to get the access.
- The user also inspects the web page then changes the given URL of changes to the desired URL on any button click.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 7, Issue 5, May 2018

2. Prevention for Attack:

- Block the inspect element view from browser.
- Implement the privilege authorization access to a user in server-side/Coding

3. Technical-Business Impact
It provides a moderate risk problem in most sensitive applications, such as financial or sensitive data handling applications and the hacker can see all most secure data.

III.CONCLUSION

The Web applications are becoming popular and have wide spread interaction medium in our daily lives. But at same point using vulnerabilities the user sensitive data also disclosed regularly. This paper surveys the different web application vulnerabilities based on the security properties that web application should be preserved. However, we enforce to have a pen test, vulnerability assessment of the web application for discussed vulnerabilities which reduces chances of the occurrence of the vulnerabilities. However, vulnerability assessment tools are automated one which saves time and money and also defends the web applications from modern threats. At the last the newly advanced security attacks are always emerging, requires the security professional to have positive security solution without putting a huge number of web applications at risk.

REFERENCES

[1] https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
[2] https://www.owasp.org/index.php/Top_10_2013-A3-Cross-Site_Scripting_(XSS)
[3] https://www.owasp.org/index.php/Top_10_2010-A4-Insecure_Direct_Object_References
[4] https://www.acunetix.com/websitesecurity/sql-injection/
[5] http://airccse.org/journal/iju/papers/3412iju01.pdf
[6] Sagar Joshi, ―SQL injection attack and defense: Web Application and SQL injection, ‖ 2005
[7] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, IEEE Std. 802.11, 1997.

BIOGRAPHY

Miss. Rohini P. Pawar Done BE in Information Technology from North Maharashtra University, Jalgaon. She is Researcher in Computer Science, website security and currently working as Software Test Engineer in Kanaka Software Consulting Private Limited. Pune, Maharashtra and her research interests are web application security, information security and software testing