A Compendium on Vulnerabilities in Web Application and Related Prevention

Mohit Dayal[1], Jitender kumar[2]
[1]Department of Computer Science and Engineering, Ambedkar Institute of Advanced Communication Technologies and Research, New Delhi, India.
[2]Rawal Institute of Engineering and Technology, Faridabad Haryana, India.
E-Mail: mohitdayal.md@gmail.com

Abstract
A secure web application is a challenge in the present era for every application developer. Despite several efforts, there may be chances for loopholes in the security of web applications. These are called vulnerabilities and for this reason, web applications are prone to attacks. The evolution of the internet was a big leap for connectivity among entities around the world. It provides a platform for billions of applications and services. In this paper, we are doing a study of five vulnerabilities and methods for their prevention. These vulnerabilities are SQL Injection, Broken Authentication and Session Management, Cross-Site Scripting, Insecure Direct Object References, Security Misconfiguration. In this paper, we will explore more about these vulnerabilities and their prevention. This work will also be helpful for further research in the above-mentioned area and will provide a better understanding of work.
Keywords: Vulnerabilities, Prevention, Web Application, Cross-Site Scripting

Introduction

The evolution of internet was a big leap for connectivity among entities around the world. It provides a platform for billions of applications and services. There was a time when this platform only contained static web pages providing information in the form of several textual and graphical formations. As the internet platform is widening with time, these static entities become dynamic. Nowadays the websites are more dynamic and structured in nature. Web applications are now the way for access to services and also alluring for each kind of business-standard. At present online transactions, personal networks, health care and solutions, business organizations, banks, and almost every system depends on web applications. With the help of these web applications, one can connect to others with just one click. These applications are used for managing documents, sharing data and information. So we can say that the internet is becoming an integral part of our day to day life. With this much importance, it comes to us with some problems. With so much to do with the internet and web-based applications, one must assure security for all this process. A safe and benign entourage is an imperative need for this process. The vulnerability can be a nightmare for users of web applications. Using a web application which is prone to attacks may endanger the privacy and integrity of user's data. Broken Authentication and Session Management, Insecure Direct Object References, Security misconfiguration, SQL Injection, and Cross-Site Scripting are considered as top 5 web application vulnerabilities for 2013 cited by the Open Web.

Figure 1: 5 Vulnerabilities cited by Open Web

SQL Injection

SQL language is the web-based vulnerability that allows a hacker to obtain access to the database information and other sensitive information such as credit card numbers and other monetary information by manipulating logic of SQL command and thus allowing an attacker to delete and update the data stored in the database.

A. SQL injection Impacts

Following are the SQL injection impacts.

1. Loss of Confidentiality: As hacker gains access on the database and other sensitive information.
2. Loss Of authentication: As attacker without providing the authentic user name and password could successfully obtain access over the network.
3. Loss of authorization: As attacker leaks complete information present on the system.
4. Lack of Integrity: As hacker obtain access on the database information and other sensitive information

B. SQL Injection Types

S. No.	SQL Injection Type	Types	Description
1	Simple SQL Injection	a) UNION SQL Injection b) Error Based SQL injection	A SQL query script is built by concatenating hard coded string with the string entered by the user In this attacker launches the SQL injection using UNION command. The attacker checks the vulnerability by adding a tick at the end of a ".php? id=”file. The attacker uses database –level error messages for making vulnerability exploit requests.
2	Blind SQL Injection	-------	Blind SQL injection can be used to obtain access over sensitive information present in the database by asking series of true or false questions through SQL statements

Table 1 SQL injection types

C. SQL Injection Prevention

1. Minimizing the Privileges

Security aspects should be prioritized and adequate steps must be taken during development stage only instead of leaving those matters to the end of the development cycle.

2. Implementation of consistent coding standards

Well planned security infrastructure and set of standards and policies must be laid down. Developers use any method to access the data resulting into multitude of data access method which is the serious security concern.

3. SQL server firewalling

SQL server should be firewalled so that only trusted clients can be contacted. Administrative network and the web server need to be connected with the SQL server.

Cross Site Scripting Attack (XSS)

Cross-Site Scripting attack (XSS) is the computer security threat that allows the attacker to get access over sensitive information when JavaScript, VBScript, ActiveX, Flash or HTML which is embedded in the malicious XSS link gets executed. There could be many ways through which XSS could be launched some of them are URL XSS which contains the malicious code embedded in the URL and gets executed when the user login, or input fields which allows inserting data which will remain on the website. To encode the malicious code attacker uses different techniques so that it appears authentic to the user, while it is not.

A. Technical Impact

Malicious XSS link gets executed on the user's browser to hijack the user session, to damage websites, to add a wrathful comment, to redirect the user to the attacker server, etc.

B. Business Impact

It includes the business value affected because of the affected system and data which is affected during the XSS exploit.

C. Types of XSS are:

1) Domed based XSS.

Domed based XSS is also known as type 0 XSS. It is a client-side attack, In this attack, user-provided data is written to the document object model (DOM) by the web application client-side script.

2) Stored XSS

Stored XSS is also known as persistent XSS or type 1 XSS. In this attack, attacker embedded a malicious script which gets permanently stored on the server. One of the most common examples of this type is an XSS attack in a comment field of a blog or forum post.

3) Reflected XSS

Reflected XSS is also known as non-persistent or type 2 XSS. In this attack attacker first builds the malicious link. After creating the malicious link it sends the URL to the user via email and persuades them to click on it. The user then sends the request to the server to give access to the required page. The genuine server handles the request made by the user and sends the response page containing the malicious code

D. XSS Prevention

Following presents the way through which XSS can be prevented:
Rule 0: "Never insert untrusted data except in allowed location"
Rule 1: "HTML escape before inserting untrusted data into HTML element content"
Rule 2: "Attribute escape before inserting untrusted data into HTML common attributes"
Rule 3: "JavaScript escape before inserting untrusted data into HTML JavaScript data values"
Rule 4: "CSS escape before inserting untrusted data into HTML style property values"
Rule 5: "URL escape before inserting untrusted data into HTML URL parameter values"

Broken Authentication and Session Management

In this attack, the attacker uses the flaw present in the authentication or session management function to imitate the user. In this, a user first sends the credentials, sites rewrites the URL by inserting session ID in the URL. Once the user clicks on the link, the hacker checks the refers logs and finds a user's JSESSION ID. Using JSESSION ID, the attacker takes over the victim's account. For example, on an airplane reservation website that supports URL rewriting, an authenticated user wants his friend to know about the sale and mails the link containing session ID, a friend finds the user's JSESSION ID and takes over the account to extract sensitive information like credit card number and password. In this attack, an attacker can hijack user account, administrative account, authorization, and accountability controls and can cause privacy violation.

A. Broken authentication and session management Vulnerabilities

1. Credentials should always be protected when stored using hashing or encryption.
2. Credentials should not be guessed or overwritten through weak account management functions
3. Are sessions IDs exposed in the URL rewriting?
4. Are password, session IDs and other credentials sent over TLS connection?

B. Impacts

Technical impact: flaws in the area such as logout, password management, timeouts, remember me, secret questions may allow some or even all accounts to be attacked. Once done, the attacker can do anything could victim do.

Business impact: This attack could affect the business data or business application function at large. An attacker could hijack user account, administrative account, authorization, and accountability controls and can cause a privacy violation

C. Broken authentication and session management preventive measures

1. An inbuilt session management mechanism must be used.
2. A single authentication mechanism must be used.
3. The login process should not be started from an encrypted page.
4. Every page should have a logout link.
5. Time out period must be used.

Insecure Direct Object References

In this attack, an attacker who is an authorized system user changes a parameter value that directly refers to a system object to another object, the user isn't authorized for. For generating a web pages application, the website uses the name or key of an object and does not verify whether the user is authorized for the target object or not. This results in an insecure direct object reference flaw. Through manipulating, parameter values testers could detect this flaw and code analysis could be used to check whether an authorization is done properly or not.

A. Insecure Direct Object References Vulnerabilities

To check whether the application is vulnerable toward Insecure Direct Object References Vulnerabilities or not, verify that all object references should have appropriate defenses. This could be achieved through the following ways:

For direct references to restricted resources, the application must verify whether the user authorized to access the resource they have requested or not.

For indirect references, the mapping to the direct reference must be limited to values authorized for the current user.
By reviewing the code of an application it could be verified whether an approach is implemented safely or not.

B. Impacts
The attacker could access all the data referenced by the parameter.
C. Insecure Direct Object References prevention
Following are the ways through which insecure direct object references can be prevented:
1. An approach must be selected for protecting each user-accessible object. Session indirect object per user must be used to prevent an attacker from directly targeting unauthorized resources.
2. Access control must verify that the user is authorized to access the object for which he requested.

Cross site Request Forgery (CSRF)
In this attack, the attacker creates forged HTTP request and tricks the authenticated user to gain access over sensitive information. An attacker could forge the HTTP requests that are difficult to distinguish from an actual one by using credentials like session cookie sent by the browser.

A. CSRF Vulnerabilities
Following are the CSRF vulnerabilities
Check whether the links contain unpredictable token because, without such token, an attacker cannot forge malicious HTTP requests.
Check the link which contains state-changing functions as these are the easiest to target.

B. CSRF Prevention
CSRF can be prevented through the following ways:
Unpredictable tokens should be added in the URL body which should be unique per session and per requests
The unique token should be added to a hidden field.

Comparative Analysis
Following table presents the comparative analysis among top 5 web application vulnerabilities for 2013 cited by the Open Web.

Figure 2: Attack Mechanism

Criteria	Injection	Cross Site Scripting	Broken authentication and session management	Insecure Direct Object References	Cross site request forgery
Threat Agents	Anonymous external attackers	Attacker who knows javaScript, VBScript, ActiveX, Flash or HTML	Anonymous external attackers	User having partial access to certain type of system data	Anyone who can trick user for submitting request on your forged website
Attack Vectors	hacker executes malicious SQL queries on the database server through a web application to either gain access over the sensitive information or on the database	URL XSS which contains the malicious code embedded in the URL and gets executed when the user login or input fields which allows inserting data which will remain on the website.	Attacker uses flaws present in the authentication	Attacker who is an authorized system user, changes a parameter value that directly refers to a system object to another object, the user isn't authorized for	Attacker could forge the HTTP requests that are difficult to distinguish from actual one by using credentials like session cookie send by the browser
Security Weakness	hacker manipulates the logic of SQL command to obtain access on the database and other sensitive information	XSS flaws occur when application includes user supplied data in page sent to the browser.	Attacker uses the flaw present in the authentication or session management function to imitate user	For generating web pages application, website uses name or key of an object and does not verify whether the user is authorized for the target object or not	CSRF occurs when application allow attacker to predict all the information.
Technical Impacts	Attacker could hijack user account, administrative account, authorization and accountability controls and can cause privacy violation	Malicious XSS link gets executed on the user's browser to hijack the user session, to damage websites, to add wrathful comment, to redirect user to the attacker server etc.	Flaws in the area such as logout, password management, timeouts, remember me, secret questions may allow some or even all accounts to be attacked. Once done, attacker can do anything could victim do	the attacker could access all the data referenced by the parameter	Attacker could hijack user account, administrative account, authorization and accountability controls and can cause privacy violation
Business Impacts	This attack could affect the business data or business application function at large	It includes the business value affected because of the affected system and data which is affected during the XSS exploit.	This attack could affect the business data or business application function at large. Attacker could hijack user account, administrative account, authorization and accountability controls and can cause privacy violation	This attack could affect the business data or business application function at large	This attack could affect the business data or business application function at large.

Preventive Measure	Sending email alerts, blocking offending IP, strong design	Strong design, Never inserts untrusted data except in allowed location.	Inbuilt session management mechanism must be used. Single authentication mechanism must be used. Login process should not be started from an encrypted page. Every page should have a logout link. Time out period must be used.	An approach must be selected for protecting each user accessible objects. Session indirect object per user must be used to prevent attacker from directly targeting unauthorized resources. Access control must verify that user is authorized	Unpredictable tokens should be added in the URL body which should be unique per session and per requests. Unique token should be added in a hidden field.

Table 2 Comparative analysis among top 5 vulnerabilities

Conclusion

The vulnerability can be a nightmare for users of web applications. Using a web application which is prone to attacks may endanger the privacy and integrity of user's data. Broken Authentication and Session Management, Insecure Direct Object References, Security Misconfiguration, SQL Injection, and Cross-Site Scripting are considered as top 5 web application Vulnerabilities for 2013 cited by the Open Web. In this paper, we will explore these vulnerabilities and their prevention. This work will also be helpful for further research in the above-mentioned area and will provide a better understanding of work.

Acknowledgement

We would like to thank the Ambedkar Institute of Advanced Communication Technologies and Research, New Delhi for helping us out in carrying the research work successfully.

References

1. R. P. Mahapatra, Subhi khan, "A survey on SQL injection Countermeasures," International Journal of computer Science and engineering survey, vol 3, pp. 55-74, June 2012.
2. Lwin Khin Shar, Hee Beng Kaun Tan, " Defeating SQL injection," IEEE computer society, vol 46, pp. 69-77, March 2013.
3. A. Sadeghian, M. Zamani, A. A. Manaf, "A Taxonomy Of SQL Injection Detection and Prevention," In Proceedings of the Conference on Informatics and Creative Multimedia (ICICIM), IEEE, pp. 53-56, September 2013.
4. Gundy, M.V., "Noncespaces: Using randomization to defeat cross-site scripting attacks",Computer and security, Elsevier,2012.
5. Open web application security project, OWASP

Source of Support: Nil
Conflict of Interest: None