



International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS)

(Open Access, Double Blind Peer-reviewed, Refereed and Indexed Journal)

www.iasir.net

Taxonomy of Web Application Vulnerabilities

Navdeep Kaur, Karandeep Singh

Department of Computer Engg. & Tech., Guru Nanak Dev University,
Amritsar-143005, Punjab, India

Abstract: This paper is an effort of categorizing various recognized vulnerabilities of web applications. It has been seen that in the speedy race of developing web applications in minimum budget and time, security is given least importance as consequence of which web applications are developed and hosted with number of vulnerabilities in them. This race helps the attackers to take advantage of vulnerabilities existing in technology for financial gain and theft of intellectual property. In this paper various vulnerabilities are discussed with examples. This paper categorized the vulnerabilities on the basis of their causes. All the vulnerabilities are explained along with their countermeasures to prevent the system from these vulnerabilities.

Keywords: Web Application; Security; Secure Software; Taxonomy; Vulnerabilities;

I. Introduction

Accessing of remote data becomes easier day by day with the development in network software. It becomes easy and simple with new software. Besides this, new software also gave rise to new vulnerabilities in software/web application. According to statistics, daily more than 10 vulnerabilities are found in open source or commercial software [1]. So it is inevitable to build secure software without any vulnerability.

There are many reasons which cause increase of vulnerabilities in web applications. Main reason behind their increase is attack toolkits which make everything easier for attacker to do. Anyone can find various ways to attack web applications, everything is available online. Hackers do various things with web applications to get money, information or fame. Very sensitive data is present at remote positions. Hackers can trace credit card number, pin number or bank details and make transactions on behalf of user. Hackers can also get sensitive information like username or password to get all the privileges of legal user.

There are various reasons behind leaving flaws in web applications, such as programmer has to complete their within limited period. There might be a problem of budget, due to which they are not able use good tools for testing and validation of software.

II. Related Work

Efforts in the context of secure implementation have focused on studying implementation vulnerabilities and devising secure coding methods to avoid them [7,8]. Some studies are present that have studied various vulnerabilities and their discovery rate [11,12,13]. These studies treat every vulnerability as equally important entity. Various taxonomies has proposed to get the universal taxonomy of vulnerability, for categorizing vulnerabilities by vulnerability type [14,15]. Focus of these vulnerabilities taxonomies on goals like: Exhaustiveness, mutual exclusiveness, repeatability, and taxonomy provided by these qualities will be unambiguous. Vulnerability taxonomy is a developing area of research; even so, there are various organizations like, NVD (National Vulnerabilities Database) and MITRE Corporation [10] which identify the taxonomy and shown to be acceptable [9]. CVSS (Common Vulnerabilities scoring system) is another taxonomy include the classification of various vulnerabilities on the basis of their severities. NIST and NVD adapted CVSS, and it provides vulnerabilities severity classification standard, this system give each vulnerability a value from 1 to 10, the lower number means the less severe vulnerability, vice versa [16].

III. Categorization of Software Vulnerabilities

Vulnerabilities [2] are weak spot of application, through which attacker can cause harm to user's data or application. Below 13 vulnerabilities are discussed, which are mainly categorized in 5 ways:

- Input validation
- Session management
- Authorization
- Authentication
- Cryptography

A. Input Validation

Input validation[19] becomes security vulnerability if a hacker/attacker traces that your application makes untraced assumptions about the format, size, range and type of input data. The hacker can then carefully transfer

vulnerable input that can harm your application. Entry points at network and host level are secured fully, then the interface exposed to the public is the only way to attack. Your application's input is a way to execute hacker's code. Your application is susceptible to the following vulnerabilities if your application blindly trust input:

- Buffer overflow
- Cross-site scripting
- SQL injection[17,18,19,20]

These vulnerabilities are discussed in detail as follow:

- **Buffer Overflows**

Buffer overflow vulnerabilities can cause code injection[2] or disaffirmation of service attacks[9]. Code injection changes the execution address of program to run hacker's injected code. Example of buffer overflow is discussed below. The program given below allow the user to fill password ,if the password entered by user is "buffer" then the root privileges granted to user, otherwise not.

```
#include <stdio.h>
#include <string.h>
int main(void)
{
    char buf[15];
    int pass = 0;
    printf("\n Password : \n");
    gets(buf);

    if(strcmp(buf, "buffer"))
    {
        printf("\n Wrong Password \n");
    }
    else
    {
        printf("\n Correct Password \n");
        pass = 1;
    }
    if(pass)
    {
        /* privilege to user*/
        printf("\n Root privileges provided to the user  \n");
    }
    return 0;
}
```

If user enter 'buffer' as a password then following output will be shown:

Enter the password :

buffer

Correct Password

Root privileges provided to the user

This code works as expected, root privileges given to user when the password is correct. But possibility of buffer overflow is there in this program. The function gets() do not checks length of string or array bound. The user can enter the input which is greater than the size of buffer. Example is given by following to show what attackers can do by overflowing the buffers.

Example :

Enter the password :

hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh

Wrong Password

Root privileges provided to the user

As shown in example even after entering the wrong password, program worked as user given a 'buffer' password. Reason behind this behavior of program is that when we enter the input greater than size of buffer which cause buffer overflow and overwrote the value of integer 'pass', due to this value of 'pass' become non-zero and hacker got the privileges. Good programming practices are advised to avoid the buffer overflow. For example:

- Memory auditing[q] should properly done in the program by using various utilities.
- Use fgets() instead of gets().
- Use strncpy() instead of strcpy(),strncpy() instead of strcmp(), etc

- **Cross-Site Scripting**

Cross-site scripting[17] target the user of application not the application itself. An XSS attack can run any code in browser of the user while that browser has opened a trusted website. Internet explorer's security cannot trace this type of attack because the code is downloaded from the trusted website by the browser, so due to this reason there is no method to check that code is vulnerable. Hacker's code can access cookies related to that trusted website and those cookies are stored on user's computer, so hacker can also access the authentication cookies of user which is typically main target of hacker.

Example of Cross-Site Scripting

To initialize the cross-site scripting attack, attacker must convince the user to click on hyperlink, like attacker can give attractive offers, or provide the email notification or any other information that can attract user to click hyperlink. This link directs user toward malicious page. For example:

Authorized link:

`www.trustedwebsite.com/logon.aspx?username=abc`

Malicious link:

`www.trustedwebsite.com/logon.aspx?username=<script>alert('hacker code')</script>`

This example display the pop up message, instead of this hacker can use other script which can easily extract the authentication cookies of user and use that authentication on target website to access that website as a authenticated user.

Countermeasures:

- Proper input validation must be applied and it should be ensured that query strings must be valid for the application.
- **HTMLEncode** and **URLEncode** functions should be used to encode any output that include user input.

- **SQL Injection**

In SQL injection[18,20], hacker uses such inputs which construct dynamic SQL statement, which enable the hacker to access the database. Hacker can access the unauthorized data or can alter the database. For example:

Suppose hacker enters 21 or 1=1 in the user id's text box. then the SQL will execute as following:

`SELECT Id, UserName, Password FROM Login WHERE Id = 21 or 1=1`

The SQL statement written above is valid, and returns all the rows of table Login. In this way hacker could access the unauthorized information (login ids and passwords).

Countermeasures:

- Proper input validation should be applied.
- Use the accounts with least privilege[q] to connect the database.
- It should be ensured that user inputs are not generating any dynamic SQL statement.

B. Session Management

Session management is done under application layer. Session avoid repeatedly authentication of user, if user login once there is no need to login again until user logout. Security of the session of application is a crucial task. Various vulnerabilities related to session management are:

- Session hijacking
- Session replay

- **Session hijacking**

Session is said to be hijacked when hacker got the authentication token by using some network monitoring tool. Mainly the authentication tokens are in form of cookies, hacker trace that cookies and then use them for authentication. Hacker got the same access as legal user.

Remedies to prevent session hijacking:

- Secure socket layer (SSL) [3] should be used for secure communication channel.
- Logout function should be implemented so that user can end the session, and authentication should be required for another session.
- Session cookies should be expired after some period.

- **Session Replay**

Session replay is a problem which occurred when a hacker intercept token of user's session and bypass the mechanism of authentication. Session token can be in any form like cookie, URL. The hacker can use all privileges of user by hijacking tokens of session.

Countermeasures:

- Sessions should be expired properly, which include expiration of all tokens and cookies.
- "Do not remember me" option should be created so that no data would store on client side.
- In case of critical functions re-authenticate is must.

C. Authentication

Authentication verifies that the person or an entity is valid to use privileges. Authentication is mainly done by checking and verifying the username and password entered by user in addition to some personal information which is known only by user. Various vulnerabilities to authentication are:

- Network eavesdropping
- Brute force attacks
- Credential theft

- **Network sniffing**

Network sniffing is a attack in network layer. In this vulnerability, hacker capture the data packets which are sent from client to server contains username or password. The hacker can use that information to get authentication.

Countermeasures:

- Username and password should be encrypted before transmitting.
- Use such authentication mechanism, in which password is not transmitted over the network, like windows authentication.

- **Brute Force Attacks**

A brute force attack is a process of checking the hidden data of web application. The hacker tries to find the username or password by sending GET and POST request to server. In this attack, hackers configure predetermined values, and send them to server and then check response of server.

Countermeasures:

- Use strong passwords.
- Hashed passwords should be used with salt (cryptography)[4].

- **Credential Theft**

Credential can be anything which is used to authenticate user or device, like username and password. When you use your credential, it may save in your system in various ways such as cookies. If someone able to access this saved data then he/she can login with your credentials.

Countermeasure:

- Use hashed passwords with salt (cryptography)[4].
- Account should be locked after a certain number of retrying attempts.
- Option should be there to select that user want to save credentials or not.

D. Authorization

Various authorizations, privileges to certain resources and services are given to user according to his/her identity.

Various vulnerability related to authorization are listed below:

- Elevation of privilege
- Disclosure of confidential data
- Data tampering

- **Elevation of Privilege**

In any system, hacker wants to get the privileges of most powerful account of the system. So that hacker can get control over the whole system. For example: In ASP RevertToSelf API allow to run the local system with all privileges.

Countermeasures:

- Don not provide all privileges to all users.
- Make use of least privileged process.

- **Disclosure of Confidential Data**

Disclosure of confidential data means hacker can view very sensitive data. Sensitive data includes credit card/debit card number, personal details of user, financial records and other information which user do not want to disclose. Access to these credentials should be limited to trustable administration. Database, files that contain the information must be secured, only authenticated and authorized user can access them.

Countermeasure:

- Encrypt sensitive data before storing in database and files.
- Check roles before giving access to sensitive data.

- **Data Tampering**

In data tampering vulnerability, hacker can alter the data. It can be done in various ways, such as modification in cookies, http headers, URLs.

Countermeasures:

- Give access to authorized user only, to alter data.
- Rule based security should be used, to differentiate between user who can only view data and those who can modify data.

E. Cryptography

Cryptography is technique for securing communication or data by changing it from meaningful state to meaningless state. Various vulnerabilities in cryptography are:

- Poor key generation
- Weak or custom encryption

- **Poor Key Generation**

Cryptography is done with the help of key, both encryption and decryption require to change the state of data. If the key is easy to guess then hacker may find it and change the state of data to meaningful state.

Countermeasures:

- Do not generate key which is guessable.
- Key should be expired regularly.
- Random keys should be generated with the help of strong random key generation functions.

- **Weak Encryption**

There is no security of data if the encryption is cracked and it is very much vulnerable to brute force attack. Custom algorithm should be tested, because they are vulnerable. User must use well known algorithm for cryptography which are not vulnerable and secure.

Countermeasures:

- Avoid using custom algorithms.
- Keep on checking that algorithm is cracked or not, and also know about techniques to crack them and handle them.

IV. Conclusion

To develop secure software is very much necessary, as vulnerabilities in software security increasing day by day. Reviewing various vulnerability classifications, this paper identified and briefly described 18 most common vulnerabilities. Then the main reasons for creation of those vulnerabilities along with their countermeasures were mentioned. It is very important that web developers keep themselves up to date with new risks which their applications could face and also understand their nature to ensure that the mitigations implemented are effective. In the future, we can expect more security solutions and more awareness of their implementation.

References

- [1] John Wilander, Jens Gustavsson, "Security Requirements - A Field Study of Current Practice", Symposium on Requirements Engineering for Information Security (SREIS'05), In conjunction with RE 05 - 13th IEEE International Requirements Engineering Conference, Paris, France, August 29th, 2005.
- [2] M.A. Hadavi, H. M. Sangchi, V. S. Hamishagi, H. Shirazi " Software Security; A Vulnerability-Activity Revisit", The Third International Conference on Availability, Reliability and Security, IEEE DOI 10.1109/ARES, 2008.
- [3] Mr. Pradeep Kumar Panwar, Mr. Devendra Kumar, "Security through SSL", *International Journal of Advanced Research in Computer Science and Software Engineering*, Volume 2, Issue 12, December 2012.
- [4] Pritesh N. Patel, Jigisha K. Patel, and Paresh V. Virparia, A Cryptography Application using Salt Hash Technique.
- [5] Mihir Bellare, Ran Canetti, Hugo Krawczyk, "Message Authentication using Hash Functions- the HMAC construction", Appears in RSA Laboratories' CryptoBytes, Vol. 2, No. 1, Spring 1996.
- [6] Yi-Min Wang, Om P. Damani, Woei-Jyh Lee, "Reliability and Availability Issues In Distributed Component Object Model (DCOM)", Available at: <http://www.cse.iitb.ac.in/~damani/papers/ut/dcom.pdf>.
- [7] M. Howard and D. LeBlanc, *Writing Secure Code*, 2nd ed. Redmond: Microsoft Corporation, 2003.
- [8] Hoglund G., McGraw G., "Exploiting Software: How to Break Code", Pearson Higher Education. 2004.
- [9] National Vulnerability Database, <http://nvd.nist.gov/>.
- [10] MITRE Corp, Common Vulnerabilities and Exposures, <http://www.cve.mitre.org/>.
- [11] O. H. Alhazmi, Y. K. Malaiya, "Quantitative vulnerability assessment of system software", Proc. Annual Reliability & Maintainability Symposium, pp. 615–620, Jan. 2005.
- [12] O. H. Alhazmi, Y. K. Malaiya, I. Ray, "Security vulnerabilities in software systems: A quantitative perspective", Proc. Ann. IFIP WG11.3 Working Conference on Data and Information Security, pp. 281–294, Aug. 2005.
- [13] O. H. Alhazmi, Y. K. Malaiya, "Modeling the vulnerability discovery process", Proc. 16th Intl Symp. On Software Reliability Engineering, pp. 129–138, Nov. 2005.
- [14] T. Aslam, and E. H. Spafford, "A taxonomy of security faults", Technical report, Carnegie Mellon, 1996.
- [15] C. R. Seacord, and A. D. Householder, "A structured approach to classifying vulnerabilities", Technical Report CMU/SEI-2005-TN-003, Carnegie Mellon, 2005.
- [16] Common Vulnerabilities Scoring System, <http://www.first.org/cvss>.
- [17] Navdeep Kaur and Parminder Kaur. "Mitigation of SQL Injection attacks using Threat Modeling", *ACM SIGSOFT Software Engineering Notes*, Vol.39 no.6, pp.1-6, <http://doi.acm.org/10.1145/2674632.2674638>, November 2014.
- [18] Navdeep Kaur and Parminder Kaur. "SQL Injection - Anatomy and Risk Mitigation", *CSI Communications Magazine*, Vol. 38 issue 3, pp. 27-28, ISSN 0970-647X, June 2014.
- [19] Navdeep Kaur and Parminder Kaur. "Input Validation Vulnerabilities in Web Application". *Journal of Software engineering (Science Alert)*, Vol. 8 Number 3, pp. 116-126, ISSN 1819-4311, DOI: 10.3923/jse.2014.116.126, April 2014.
- [20] Navdeep Kaur and Parminder Kaur. "Modeling a SQL Injection attack", Published in IEEE Xplore Digital Library, Presented at 3rd International Conference on "Computing for Sustainable Global Development" INDIACOM-2016, IEEE Conference ID: 37465 2016, Bharati Vidyapeeth's Institute of Computer Applications and Management (BVICAM), New Delhi, 16th - 18th March, 2016, ISSN 0973-7529; ISBN 978-93-80544-20-5, pp.95-100.