International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:7, No:5, 2013

Web Application Security, Attacks and Mitigation

Ayush Chugh, Gaurav Gupta

***Abstract*—**Today's technology is heavily dependent on web applications. Web applications are being accepted by users at a very rapid pace. These have made our work efficient. These include webmail, online retail sale, online gaming, wikis, departure and arrival of trains and flights and list is very long. These are developed in different languages like PHP, Python, C#, ASP.NET and many more by using scripts such as HTML and JavaScript. Attackers develop tools and techniques to exploit web applications and legitimate websites. This has led to rise of web application security; which can be broadly classified into Declarative Security and Program Security. The most common attacks on the applications are by SQL Injection and XSS which give access to unauthorized users who totally damage or destroy the system. This paper presents a detailed literature description and analysis on Web Application Security, examples of attacks and steps to mitigate the vulnerabilities.

***Keywords*—**Attacks, Injection, JavaScript, SQL, Vulnerability, XSS.

I. Introduction

WEB application security deals with the protection of web services, web applications, websites and all other benefits of web. In 1995 NETSCAPE launched a script called JavaScript [2]. The major drawback of this script was that it didn't send information to server to get full web page, instead an embedded script of a downloaded page would perform tasks like input authentication and displaying parts of page. In 1996 Macromedia launched Flash to build animations in browser, and later in 1999 web application construction began with JavaScript [2]. With the progression of HTML versions (currently HTML5) it has been possible to develop applications with graphics and multimedia capabilities. A client-server environment is one where multiple clients or application users, using same database, enter information and server stores this information that needs to be secured. The web applications use two types of scripts viz. Server side script and Client side script. The former scripts (ASP, PHP etc.) include hard stuff i.e., storing and fetching the information while the latter ones, HTML, JavaScript etc., deal with the display of information.

The web applications are constructed using different programming languages such as Python, Ruby, ASP.NET, C#, VB.NET, Java, PHP, Java EE. During the past year, attacks on web applications have grown severely. The two significant techniques of attack are: Structured Query Language (SQL) Injection and Cross Site Scripting (XSS), both of which dominated in 2009. The types of vulnerabilities are shown in Table I.

TABLE I
Types of vulnerabilities [1]

Vulnerability Types	Basic Idea
Type 1	No distinction between data types of language.
Type II	No attention to type coercion.
Type III	Incorrect authorization of user input.
Type IV	Current variables are used which leads to delay in operation analysis till runtime phase.

Several online applications (Google, wikis, online retail sales), online business-based transactions, browser applications (online spreadsheets, word processors) are built in three tier model where a single tier represents a logical chunk of application. Online payment need to be safe for a satisfactory user experience. And thus, these applications require high protection or a single raid would cause severe loss. The major attack on these applications is through SQL Injection and XSS, apart from PHP injection, Java injection, Memory corruption, Path traversal and Remote Command Execution.

The use of open source frameworks like Ruby on Rails, Django has made development of web applications versatile. Nowadays, a substantial part of the population file their income tax using one these applications. The complexity of applications varies from a single message board to a spreadsheet. The applications being developed in this modern era can run on different platforms (PC, MAC) and these applications can be explored over the internet from different operating systems. The best part is that you don't need to install them; the updates are always available on the website. The dependency of people on these applications has never been so vital. They share important information via these applications. This information would always be on stake if there isn't high security. With the introduction of SaaS (Software as a Service), [23] the client need not to bother about the hardware/software compatibility to run any application. Applications can be used any time as there is direct access to them.

Attackers are always in search of vulnerable applications. They invade these applications and get access of confidential data, cause severe loss to the client and can also damage the server. There is a rise in concern for security as the documents; pictures etc. uploaded by you always remain online. The web application security is needed to prevent raid by attackers. They can cause demolition in following ways: [15]

- Access restricted content
- steal account information
- change appearance of website

Ayush Chugh is a student of Bachelor of Technology in the Department of Computer Science and Technology, School of Engineering and Technology, ITM University, Gurgaon.

Gaurav Gupta is working as an Assistant Professor in the School of Engineering and Technology, ITM University, Gurgaon (e-mail: gauravgupta@itmindia.edu).

- change credentials
- insert spam links
- insert malicious content

II. Web Application Security

Web Application Security falls under the category of Information security. The design interfaces, which the user uses to interact with backend databases, must be secure to perform the tasks over web applications. A few search links on Google or some book references can provide information to attack even the powerful applications such as WordPress, ZenCart, Joomla!, Drupal and MediaWiki.. In some severe cases attacker can attain system level permissions and direct access to database which can harm the server and execute the commands according to their choice. This accounts for the need for web application security. Generally, it can be divided into two parts: *Declarative security and Program security*. Both these security measures safeguard the data transferred over the internet.

A. Declarative Security

It comprises of mechanisms used in an application, and shown in declarative syntax in a deployment descriptor (DD). A DD is an .xml file that shows application security structure including access controls and authorization requirements. The security information is placed into annotations/metadata by declarative security syntax. The metadata includes:

- Description of the assembly i.e., identity, name and type.
- Details of types.
- Attributes.

Approaches to Declarative Security:

1) Basic Authentication
2) Form-based Authentication

1. Basic Authentication

This includes a very basic procedure. Firstly put all the protected/sensitive data in a protected directory such as xyz.xml. To access this information there you would be required to enter a username and password as asked by server through a pop up window. The steps [25] involved are:

1. Create a list of users, password and roles in the .xml file as shown:

<user username=”adm” password=”password” roles=”admin”/>

3. Specify the use of BASIC Authentication

<login-config>
<auth-method>BASIC</auth-method>
<realm-name>BASIC Validation
</realm-name></login-config>

4. Specify URLs that should be password protected.

5. List all the possible roles.

6. List all the URLs that need SSL (Secure Sockets Layer).

SSL apply private and public key to encrypt data passed over SSL connection. The disadvantage of basic authentication is user defined GUI can’t be constructed, and hence, no modification.

2. Form-Based Authentication

The server collects information about user through a login page which is necessary to identify a single user among millions of user. The following are the steps for Form-based authentication:

- Set the all possible users, their password and role.
- List the user of Form Based Authentication using a different tag.
- Create a login page demanding username and password.
- Similarly, create a login failure page.
- Specify all password protected file paths.
- List all the other different roles granted access to privileges.
- List all URLs require SSL.

TABLE II
Basic vs. Form-Based [25]

Basic	Form-based
Credentials are entered in browser provided dialog box.	Credentials are entered using a customized page.
Credentials can only be collected.	Customized data can be collected.
HTTP Authentication header is used to convey credentials.	Form data is used to convey the credentials.

B. Program Security

In this security technique, user has full control on authentication and access control. This is because it is done through programs. Consequently, it makes all the components portable as there is involvement of server related components. Also, there is no need to create configuration files. The programs can be written according to the need. For better authentication and user friendly experience, a customized program can be made. The steps [25] involved in Programming Security are:

- Use a data structure to store usernames and passwords. Use a function to match these credentials.
- Use sessions to keep a check on currently logged in users. For Example: String loginName = request.getParameter (“username”).
- Enable SSL by using https instead of http. Use request secure to check working of SSL. It will return either true or false.

While designing the program one should always remember that user inputs may act as your enemy. All measures are to be taken in consideration so that there is no attack on database via SQL injections. Each JSP (Java Server Page) needs to be authenticated before proceeding further. In Java EE, a java class named Servlet is responsible for responding to HTTP.

In the program you need to direct the request to https URL so that SSL can protect data over the network. Due to this, a unique connection is built between client and server. Entire information is passed in a private mode. Though this communication is slow, it is a safe process. The programs give a flexible experience in the development of web applications. All these aspects help in building dynamic web pages whose content varies according to the arguments passed. Thus you can build a secure login page using servlet code [6].

III. SQL Injection

SQL stands for Structured Query Language. Here Injection refers to insertion of SQL query through the input data from client to application and is executed by back end database. SQL Injection attack (SQLIA) is one of the top 10 web application vulnerabilities and the reason behind it is the inadequate authentication of input [13] i.e., adding malicious keywords into an SQL query [7]. It can assist an attacker to access all the high privileged data which was hidden earlier. Today a number of websites are susceptible to SQL Injection vulnerabilities. The attacker's input is an SQL query which becomes a SQL code [3]-[9] i.e., by injecting a SQL statement with help of escape characters like')'OR'1'='1'; which is always true, an attacker can intrude the web application and can update/delete/modify the database and even perform administrative tasks.

A successful SQL Injection attack is due to defective coding of web applications. All the features such as forums, login page, search pages, user credentials can be exploited via SQL Injection. This caused by the non-validation of input parameters by applications. This can be explained by an example: username "Scott" and password "tiger" are the default credentials in oracle. An attacker can pass "grant dbato Scott" as a parameter in one of the SQL queries which in first go gives error but actually, database has already gone ahead and granted permission [4]. Now "Scott" who was earlier a low privileged user is now granted administrative privileges. Another way of raiding is by the use of tools whose perquisites are that you just need to know the URL and the database to be attached and then you are on the go. In a few seconds an attacker can carry out desired proceedings. Now the question is how to write the SQL queries which would betray the database. The required query is:

- select * from users where username = ' ' OR 'z' = 'z';'z'='z' guarantees truth, regardless of the first part. Similarly another login query can be written as "select profile from users where username = 'anything' and password = 'anything' OR '1' = '1';"

Another way of bypassing the authentication of a login page would be by use of escape character in username section. When you open a specific website you need to provide username and password to login. In order to gain unauthorized access attacker may enter 'test' or 1=1-- which will open the id profile of the first user on the top of the list [8]. Tools like Burp Suite can be used for extraction of rows from database automatically.

A. Blind SQL Injection

It is used when a web application is vulnerable to SQLIA but the results are not apparent to attacker. The resultant page would depend on the statement entered with the corresponding SQL query. This may demand new query for every new bit. This can be encountered by the use of a tool which will automate the whole attack. For example [18]:

- select * from employee where dept='5' AND '1'='1';
- select * from employee where dept='5' AND '1'='3';

If '1'='1', it returns the original page and if '1'='3', error or blank page is returned. Thus the website is prone to the attack. There can be problems of incorrect type handling in which the attacker appends the SQL query with the desired query taking help of data types. It is known that age is a numeric data. Then to perform an administrative task,[10] "Scott" user can perform as:

```
<?php
$sqlStatement="Select * from 'users' where age=30;
truncate table 'users'";
?>
{/code}
```

The value would always be an integer. If the programmer doesn't check if the value is really an integer, then the attacker can add one or more statements and ultimately harm your database. The various schemes [1] related to SQL injection are: Amnesia [5], SQLrand [11], Automated approaches [5], SQLDOM [17], webSSAR [13], [21], [22], SQLIPA [26], DIWeDa [14], SQLGuard [19], CANDID [19].

B. Mitigation of SQL Injection

1. *Use Parameterized Query:*You should not supply the values directly, instead the values can be passed as parameters as shown below.
2. *Automatic and Dynamic Access Control List profiling:*This should be preferred so as to continuously monitor the application behavior and adapt to the application changes.
3. *Take steps to discard invalid input:*You can use the following code to discard non-integer value [10]:

```
if(!eregi("^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,3})$", $email))
```

4. This will allow only an integer value to be entered. Hence, it does not give chance to attacker to hamper the database.
5. *Use quote blocking function:*The function will not allow attackers to intrude inside the database [24]. A check is maintained on the values given by user.
6. *Avoid detailed error message:*The error message can provide the attacker hint regarding the parameters. Though not descriptive, a detailed error message is sufficient for a skilled attacker. So turn off the error reporting or just type something in error message which is of no use to attacker.
7. *Impart limited permissions to users:*There is no need to give extra privileges to people who don't need them. All the irrelevant privileges should be removed and granted later if the need be.
8. *Use firewalls:*dotDefender can be used to detect intrusion and keep the network secure.

IV. Cross Site SCRIPTING (Xss/css)

It is one of the most common application layer hacking techniques. XSS attacks have been taking place since 1990s. The major websites which are targeted are social networking websites such as Facebook, Twitter, and Orkut. It is a four stage process:

1. Attacker inserts client side script into vulnerable web server or web pages.
2. A user (victim) visits this server.
3. The code enters the victim's browser.
4. The code executes with web server privileges.

The visitor is tricked as the web application looks legitimate. This confirms XSS attack taking place when data enters an application via a suspicious source and data is sent to web user along with the dynamic content. The web applications are dynamic nowadays. The content changes after a few seconds which gives enough privileges for an attacker to attack on the application. The applications get data from a number of sources and after that this data is filtered to produce the resultant webpage. It contains pictures, text, scripts. The most common way used by attacker is to write script in the comment section. Every visitor who sees this script will download it and execute it on the browser. This will cause undesirable results. For example if there is link in your Gmail inbox, then it may show undesirable behavior if not filtered by Gmail server. The links, if not filtered, can contain malicious content which will give access to the cookies of the user to the attacker and he can now login to the website as a legitimate user. The credentials might be changed or misused leading to severe results. The types of XSS [12], [20] are:

A. Reflected XSS

This occurs when data supplied by a web client in the form of HTML query parameters or HTML form submission, is not filtered and can be used by server directly to scan and display a set of resultant pages. Not very unlike when you enter something in the search engine and the result is the items which contain your keywords and also the vector representing what you are searching for. It is also called Non-persistent XSS. The URL which you receive in your e-mail might contain a XSS vector.

B. Persistent XSS

This occurs when the script provided by an attacker is stored by server and passed to the logged in user during browsing session. Therefore it is also known as Stored XSS. It is the most significant of all XSS as there is no involvement of third part website to direct the user to it. Take an example of the Facebook Timeline section. If you have posted a new photograph then attacker can comment on it and write a malicious script which is invisible to you. This script executes when you visit the comment section.

C. Document Object Model Based

Dom-based vulnerabilities occur due to changes in DOM environment of victims' browser. Scripts can be inserted in the error messages supplied to the user. HTTP response remains the same but client code runs unexpectedly.

Aspects of XSS Attacks [12]

1) **Stealing Cookies:** Cookies are parts of text which are saved by the websites in your browser. The attacker injects the script that reads the cookies of a specific website. These scripts send desired information to the attacker. The attacker can log in on the website by knowing the credentials of victim, thus performing tasks at the name of the victim.

```
//JavaScript code
<script>
varimg = new Image();
img.src="http://xyz.com/log_cookie.php?"+
document.cookie
</script>
```

2) **Defacement:** Attacker injects the script which changes the appearance of a specific website. The script takes victims to another website. Religious and government website are usually targeted by attacker. Script example:

```
//JavaScript code
<script>
document.location="http://xyz.com";
</script>
```

3) **Phishing**: Attacker creates his own page which is the exact copy of the webpage with which the victim is familiar. The user is trapped when he enters the credentials on the fake login page. The data is sent to the attacker as written in the injected script.
4) **Run Exploits:** Exploit is a tool run by attackers on the system to perform offensive attacks to exploit the weakness in software over internet. Zero-Day is a common type of exploit. Due to this, a malware is installed on user's system and then attacker can again render your security useless.
5) **Privacy Violation:** The attacker's script can provide the list of websites visited by the victim. This would allow attacker to select the websites asking for the credentials from victim. Now, the attacker can make a duplicate website and perform phishing attacks.

Mitigation: To mitigate XSS attack, you must prefer escaping of string input. The script should include the location of the unwanted strings to be placed in a HTML document. The tools, which look like OWASP (Open Web Application Security Project) antisamy, [16] can be used to secure the application against faulty HTML input. You must employ additional security measures when running a website functioning on cookie based authentication. Use plug-ins which could block untrusted websites. Blocking the scripts would be beneficial if you have knowledge about them. Many defensive technologies are available which guarantees minimum XSS. Those technologies are Mozilla's Content

Security Policy, JavaScript Sandbox and Auto-escaping templates.

V. Conclusion

Web application security is the demand of the hour. It cannot be neglected; it is a serious issue like network security. With the advancement of technologies, it has spread like wild fire. Online business is most affected by web application vulnerabilities. Due to lack of security, attackers always prefer to attack web applications.

The main problem is that most of the web applications don't have testing of desktop software. Therefore there is always a growing concern for security. These applications must support all different browsers and operating systems which demand high security and better code writing. You have to be cautious about the data you enter in an application as it has the probability of getting copied or misused. The development of web applications is a long term task as they are very complex. You must include security in your coding and never assume that you would always get the correct input. There must be a way to deal with the unexpected input from a browser, unhandled error messages, unchecked database call. You can use content management system like Joomla! to construct a website with dynamic content. For the e-commerce security you can rely upon either Zencart or Magento. There can also be a way to prevent injections which is to avoid characters which have a distinct meaning in SQL. Single quote (') can be replaced by two single quotes ('') to form an accurate SQL string literal. You must assign limited permissions to the database and extend these when the need arises.

Unnecessary privileges can cause serious threats to your database. You must prefer PHP over any other language as PHP is moving towards object oriented architecture and there is use of PDO classes which give privilege to make prepared statements to prevent SQL Injections. You will find more code built in Open Source world using PHP. All major CMS (Content Management System) like Drupal and Joomla! are built in PHP. You can also use Aspect-Oriented Programming (AOP) Paradigm to mitigate SQL Injection and XSS attacks. AOP focuses on making a relation between user requests. Its remote address acts as an AOP advice where advice is the additional code you want to apply to your existing model like a log code process of users who executed the update statement.

In order to secure your web applications you must be aware about the above said attacks and should be prepared for any kind of new attacks. Thus, you must have knowledge about progression of technologies. Use web application vulnerability evaluation tools in order to check the current security level of your web application or website. Educate developers, security professionals, testers regarding the risks and steps to mitigate the attacks. You must be sure how you pass the data and don't expose your logic. A single loop hole can lead to the destruction of your whole database. You must always have a backup and keep it safe.

Acknowledgment

We greatly acknowledge the research assistance provided by Mr. Bhavyanshu Parasher.

References

[1] Diallo Abdoulaye Kindy and Al-Sakib Khan Pathan, A Detailed Survey on Various Aspects of SQL Injection: Vulnerabilities, Innovative Attacks, and Remedies.
[2] http://en.wikipedia.org/wiki/Web_application.
[3] Tajpour, A., Masrom, M., Heydari, M.Z., and Ibrahim, S., "SQL injection detection and prevention tools assessment," *in Proc. 3rd IEEE International Conference on Computer Science and Information Technology*, China, 2010, pp. 518-522.
[4] http://www.youtube.com/watch?v=M2N-uDMCot4&feature=pyv
[5] Junjin, M., "An Approach for SQL Injection Vulnerability Detection. Sixth International Conference on Information Technology," *in Proc. New Generations*, 27-29 April (2009), pp. 1411-1414.
[6] http://stackoverflow.com/questions/10981191/create-a-sample-login-page-using-servlet-and-jsp.
[7] Kindy, D.A. and Pathan, A.-S.K., "A Survey on SQL Injection: Vulnerabilities, Attacks, and Prevention Techniques" *in Proc. 15th IEEE Symposium on ConsumerElectronics*,Singapore, 2011, pp. 468-471.
[8] http://www.youtube.com/watch?v=PB7hWlqTSqs.
[9] https://www.owasp.org/index.php/SQL_Injection.
[10] http://www.tanzilo.com/2008/11/14/sql-injection-prevention-protection-in-php-mysql-with-example/.
[11] Boyd S.W. and Keromytis, A.D., "SQLrand: Preventing SQL Injection Attacks," *in Proc. 2nd Applied Cryptography and Network Security Conference*, China, 2004, pp. 292–302.
[12] https://it.ucsb.edu/system/files/websecurity.ppt.
[13] Halfond W. G., Viegas, J., and Orso, A., "A Classification of SQL-Injection Attacks and Countermeasures" *in Proc. of the Intl. Symposium on Secure Software Engineering*,U.SA, 2006, pp.
[14] Roichman, A., and Gudes, E., DIWeDa - Detecting Intrusions in Web Databases. Atluri, V. (ed.) DAS 2008. LNCS, vol. 5094, Springer, Heidelberg (2008), pp. 313-329.
[15] http://www.applicure.com/solutions/web-application-security.
[16] https://www.owasp.org/index.php/Category:OWASP_AntiSamy_Project
[17] McClure, R.A. and Kruger, I.H., "SQL DOM: Compile time checking of dynamic SQL statements,"*in Proc. 27th International Conference on Software Engineering*, St. Louis, MO, U.S.A, 2005, pp. 88- 96.
[18] http://en.wikipedia.org/wiki/SQL_Injection.
[19] Buehrer, G., Weide, B.W., and Sivilotti, P.A.G., "Using Parse Tree Validation to PreventSQL Injection Attacks" *in Proc. 5th International Workshop on Software Engineering and Middleware*, Portugal, 2005,pp. 106–113.
[20] http://en.wikipedia.org/wiki/Cross-site_scripting.
[21] Ruse, M., Sarkar, T., and Basu. S., "Analysis & Detection of SQL Injection Vulnerabilitiesvia Automatic Test Case Generation of Programs" *in Proc. 10th Annual International Symposium on Applications and the Internet*, Seoul, Korea,2010, pp.31-37.
[22] Huang, Y.-W., Yu, F., Hang, C., Tsai, C.-H., Lee, D.-T., and Kuo, S.-Y., "Securing Web Application Code by Static Analysis and Runtime Protection,"*inProc. 13th International Conference on World Wide Web*, New York,2004, pp. 40-52.
[23] http://www.pssuk.com/AdvantagesWebApplications.htm.
[24] http://www.helpspot.com/helpdesk/index.php?pg=kb.page&id=186
[25] http://www.cs.cityu.edu.hk/~jia/cs4273/L10_SecurityProgramming.ppt
[26] Ali, S., Shahzad, S.K., and Javed, H., "SQLIPA: An Authentication Mechanism AgainstSQL Injection," *European Journal of Scientific Research*, vol. 38, no. 4,pp. 604-611, 2009.