IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 20, Issue 2, Ver. IV (Mar. - Apr. 2018), PP 30-35
www.iosrjournals.org

A Survey on Malware & Session Hijack Attack over Web Environments

Jyoti Kushwaha[1], Dheeresh Soni[2]
[1]*RGPV, Computer Science And Engineering, SIRT, Bhopal, MP, India-462041*
[2]*RGPV, Computer Science And Engineering, SIRT, Bhopal, MP, India-462041*
Corresponding Author: Jyoti Kushwaha[1]

Abstract : *Web applications are very effective means of nitration between end-users. Today social networking based web applications are very popular due to its user friendly behavior but in recent years many social network sites have suffered from cross site scripting attacks and several business and private organization are suffered from the SQL injection attacks, which were conducted by suspicious users through inserting a vulnerable script into web form components. Network attack is a form of intelligent and sophisticated crimes in the web environment. Network based attacks are increasing at fast speed to hamper the smooth working of web resources. New trend in Network based attacks is a rising number of indirect attacks that manipulate network traffic, instead of directly entering into a network resource and damaging it. In recent time, damages from indirect attack are estimated to be more severe. In addition, the bandwidth consumption by these attacks influences the entire network performance. In network Malware authors are gradually increasingly and use specialized toolkits and obfuscation techniques to modify existing malware and avoid detection by traditional antivirus software. This paper gives an bird eye over malware and other network attack and various preservation techniques.*

Keywords *- Web environment, Network Attack, Malware attack, Session hijacks attack*

Date of Submission: 20-04-2018 Date of acceptance: 07-05-2018

I. Introduction

A web application is an application that is accessed via a web browser over a network such as the Internet or an intranet. Web application contains web pages which are written in a browser-supported language (such as JavaScript, HTML) and dependent on a web browser to turn into the application executable. Web applications offer services such as mail services, online shops, portals, social networking services. Benefits of web application can access anywhere. Web applications are popular due to the ubiquity of web browsers. The ability to update and maintain web applications without distributing and installing software on thousands of client computers is a key reason for the popularity of web application, web application also supports to cross-platform compatibility.

Most web applications are based on the client-server architecture where the client enters information while the server stores and retrieves information. Internet mail is an example of this, with companies like Yahoo and Google offering web-based email clients. The new move forward for web applications is crossing the line into those applications that do not normally need a server to store the information. Word processor, for example, stores documents on the computer, and doesn't need a server. Web applications can provide the same functionality and gain the benefit of working across multiple platforms. For example, a web application can act as a word processor, storing information and allows to downloading the document onto personal hard drive [1].

Social networking based web applications are very popular in the world; it is also known as social web sites. The functions of social networking websites are provide the interaction among the end-user. The advantage of social networking based application is that a person is in the India, could build up an online friendship with someone in Nepal or Bangladesh. A social network service focuses on building social relations among people, e.g., who share interests and/or activities. Social networking sites allow users to share ideas, activities, events, and interests within their individual networks. The most commonly worldwide used social networking sites are Facebook, Twitter, MySpace, Orkut, Hi5, Friend Finder, Yahoo! 360, etc. By their nature, Web applications often a wider meaning a very large number of potential attackers full access to the Internet as. All these factors web applications very attractive target for attackers and the coming out of new attacks. Hackers inspect a Web application and communications to understand its design, identify any potentially imperfect aspects, and use these weaknesses to break or exploit the application. There are risks associated with social networking including data theft and privacy leakage.

As the Internet is growing, the web sites become more proficient and dynamic. In order to be able to

modify the design of the web page to meet today's taste the web sites no longer use static web pages. Now web applications are used to generate dynamic web pages. End user requests a page the web application gathers the needed information and assembles it to a web page according to a request. The result of this process is then sent to the web browser of the requesting user, where it is interpreted and displayed. Embedded code which allows a more interactive web experience is executed by the web browser. This allows dynamic menus that react on mouse movements and clicks, parts of the web page that can be hidden or made visible and changing content on the web page[2,3].

II. Related Works

This section gives an extensive literature survey on the existing code and SQL injection attack prevention technique. We study various research paper and journal and know about code and SQL injection attack in web environment. All methodology and process are not described here. But some related work in the field of web security discuss by the name of authors and their respective title.

Kasra Amirtahmasebi, Seyed Reza Jalalinia and Saghar Khadem [4] present the deep study of SQL injection attack. SQL Injection Attack (SQLIA) is an approach by which it is possible for the attackers to achieve direct access to the database and culminates in getting sensitive information from the database of any organization. This survey paper presents and analyzes six different SQL Injection prevention techniques which can be used for securing the data storage over the Internet. The survey starts by presenting Variable Normalization and will continue with AMNESIA, Prepared statements, SQL DOM, SQL and SQLIA prevention in stored procedures respectively. Finally this paper presents the various technique pay attention to standardization of SQL statement in order to getting unauthorized access of an organization's database. It shows the most common attacks as well as advanced. The paper discusses the suitable SQL prevention techniques for a certain attacks. The author believed that each prevention technique cannot provide complete protection against SQL Injection Attacks but a combination of the presented mechanisms will cover a wide range of SQL injection attack.

Ntagw Abira Lambert & Kang Song Lin [5] proposed a method to detect SQL injection attacks using Query tokenization which has implemented by the Query Parser method. Query Parser which must be execute before the query runs in the application. When attacker makes SQL injection he probably uses a space, single quotes or double dashes in his input. Our method consists of tokenizing original query and a query with injection separately, the tokenization is performed by detecting a space, single quote or double dashes and all strings before each symbol constitute a token. Now check the valid input to deter attacker from injection attack.The method to detect one of the above symbols (space, single quote or double dashes) has been discussed in paper. The technique consists of tokenizing original query and a query with injection. After that if it founds additional symbols have been used in user input, the injection has detected.

Aurelien Francillon, Claude Castelluccia [6] has described how an attacker can take control of a wireless sensor networks. The authors stated that this attack can be used to silently eavesdrop on the data that is being sent by a sensor, to modify its configuration, or to turn a network into a botnet. The work focuses on proving the feasibility of permanent code injection into Harvard architecture based sensors. The attack combines several techniques, such as fake frame injection and return-oriented programming, in order to overcome all the barriers resulting from sensor's architecture and hardware. Authors have also described how to transform their attack into a worm.

Raju Halder and Agostino Cortesi [7] proposed obfuscation/ de-obfuscation based technique to detect the presence of possible SQL Injection Attacks (SQLIA) in a query before submitting it to a DBMS. This technique combines static and dynamic analysis. In the static phase, the queries in the application are replaced by queries in obfuscated form. Finally, a de-obfuscation step is performed to recover the original query before submitting it to the DBMS. It has the following advantages: the verification for the presence of possible SQLIA is performed at atomic formula level and only on those atomic formulas which are tagged as vulnerable. This scheme does not depend on the static models or the private key. It depends only on the accuracy of the dynamic verifier at atomic formula level.

Adam Kie˙ zun, Philip J. Guo, Karthick Jayaraman and Michael D. Ernst [8] present a technique for searching security vulnerabilities in Web applications. SQL Injection (SQLI) and cross-site scripting (XSS) attacks are widespread forms of attack in which the attacker crafts the input to the application to access or modify user data and execute malicious code. In the most serious attacks (called second-order, or persistent, XSS), an attacker can corrupt a database so as to cause subsequent users to execute malicious code. They have presented a technique for creating SQL injection and cross-site scripting (XSS) attacks in Web applications and an automated tool, Ardilla, that implements the technique for PHP. Our technique is based on input generation, dynamic taint propagation, and input mutation to find a variant of the input that exposes vulnerability. Using a novel concrete symbolic database to store taint, Ardilla can actively and accurately find the most damaging type of input-based Web application attack: stored (second-order) XSS. A novel attack checker that compares the

output from running on an innocuous input and on a candidate attack vector allows Ardilla to detect vulnerabilities with high accuracy. In this experiments, Ardilla found 68 attack vectors in five programs, each exposing a die rent vulnerability, with few false positives.

Qian XUE and Peng he [9] introduced the mechanism of SQL injection attack. Differing from the works of the predecessors, the authors categorize the injection attacks according to the Characteristics of the injection codes. SQL injection attack aims at imprecise programming during application development process, so this kind of attack is 'legitimate' for most firewalls. The solution to the problem could only depend on the improving of the programming. For the type of web databases with SQL Server as the backend, a DDL (Detection- Defense-Log) Model against SQL injection is created. Both the client computer and the server are included in the model. The model is intended to prevent as many attacks as possible and record the dangerous attack actions by deploying some smart program on the client computer and the server respectively, which can check the length and data type of the submitted variables, and detect the injection-sensitive characters and keywords. There aren't many tools specifically for SQL injection attacks, and W poison is helpful for the development with the use of asp, php. The model of DDL in this article is regarded as a whole solution, which could also be used for other relational database with some change in the code.

Li Shan, Dong Xiaorui and RaoHong [10] describe SQL injection is an attack technique used to exploit code by altering back-end SQL statements through manipulating input. And proposes a novel methodology of preventing SQL injection attacks by building a protective adaptive shell. The protective shell is located between the application and the back-end database and has three layers to block illegal SQL statements. It could be adaptive after training and reduce the possibility of data leakage to protect the database system.

Jeom-Goo Kim [11] describe a large percentage of web security incidents are SQL Injection attacks which are a serious security threat to databases with potentially sensitive information. Therefore, much research has been done to detect and prevent these attacks and it resulted in a decline of SQL Injection attacks. And proposes a SQL Injection detection method by comparing the static SQL queries with the dynamically generated queries after removing the attribute values. Furthermore, they evaluated the performance of the proposed method by experimenting on a vulnerable web application. They also compared there method with other detection methods and confirmed the efficiency of the proposed method. The proposed method simply removes the attribute value in the SQL queries for analysis which makes it independent from the DBMS. Complex operations such as Parse trees or particular libraries are not needed in the proposed method. Though this method cannot detect all attacks on vulnerabilities on web applications, since it uses both static analysis and dynamic analysis, it can effectively detect SQL Injection attacks on web applications.

M. Muthuprasanna, Ke Wei and Suraj Kothari [12] describe SQL-Injection Attacks are a class of attacks that many of these systems are highly vulnerable to, and there is no known fool-proof defense against such attacks. In this paper, we propose a technique, which combines static application code analysis with runtime validation to detect the occurrence of such attacks. The deployment of this technique eliminates the need to modify source code of application scripts, additionally allowing seamless integration with currently-deployed systems. We provide various optimizations improving overall efficiency, and also preliminary evaluation of prototype developed. Authors also provided preliminary evaluation results of prototype developed against various performance metrics affecting web server performance. Thus addressing these critical security issues in web applications would help transition easily towards next generation web services.

III. Security Challenges In Social Networking Sites

In social networks people share personal information, date of birth, email address, home address, photos and family relations. Some information may not be important, but a clear depiction of all about a person's feelings and attackers such as credit card fraud or identity theft information required to perform other attacks may be doing. Any real life can be made more targeted attacks and for additional information about the intended victim through the effective use. Social network sites persuade users to share information. Once information is posted or uploaded onto a Social network site, it should no longer be considered private. Even if the Social network site has powerful privacy settings, that privacy is completely dependent on the protection of the web application. Some attackers may also aggregate information from multiple sites to gain access to private information (e.g., online banking records, email). For example, personal information posted to social networking sites could be used to compromise security credentials (e.g., password, pin, security questions) for that site or other sites, giving an attacker access to private information[3].

The majority of people using these sites do not pretense a threat; malicious people may be drawn to them because of the accessibility and amount of personal information that's available. The more information malicious people have about the end user from the social sites. The personal information also is used to conduct social attacks. Using information that provide about location, hobbies, interests, and friends, a malicious person could impersonate a trusted friend or convince that they have the authority to access other personal or financial data.

Additionally, because of the popularity of these sites, attackers may use them to distribute malicious code. Sites that offer applications developed by third parties are particularly susceptible. Attackers may be able to create customized applications that appear to be innocent while infecting user's computer without their knowledge.

These social networking sites most likely to suffer from code injection attack. But security issues and privacy issues are entirely two different beasts. A security issue occurs when a hacker gains unauthorized access to a site's protected coding or written language. Privacy issues, those involving the unwanted access of private information, don't necessarily have to involve security breaches. Someone can gain access to confidential information by simply watching what user types the password. But both types of breaches are often intertwined on social networks, especially since anyone who breaches a site's security network opens the door to easy access to private information belonging to any user. But the potential harm to an individual user really boils down to how much a user engages in a social networking site, as well as the amount of information they're willing to share.

IV. Code Injection Attacks

Code Injection is a term used when code/script is injected into a program/web application from an outside source, for example input field which is provided by the web application to take input from the end-user. Code Injection attack is a type of attack in which malicious code injects in to a program or web application such an attack may be performed by adding strings of characters into a cookie or argument values in the URI. This attack makes use of lack of accurate input/output data validation. The concept of Code Injection is to add malicious code into an application, which then will be executed. Added code is a part of the application itself, if successful would result in either damage to an asset, or undesirable operation. Attack can be performed within software, web application etc in which the weakness is present. This term applies to mistakes regardless of whether occur in implementation, design, or other phases of the software development life cycle (SDLC). Weakness contribute to the introduction of vulnerabilities within that software or web applications, vulnerability can be used by the attacker to exploits the web applications to gain access unintended data, denial of services, or perform incorrect operations.

V. Brute Force Attack

A brute force attack is a strategy used to break the encryption of data. It involves traversing the search space of possible keys to crack the password until the correct key is found. It determines an unknown value by using an automated process to try a large number of possible values. That means a type of password attack that does not attempt to decrypt any information but simply continue to try different passwords. For example, a brute-force attack may have a dictionary of all words and/or a listing of commonly used passwords. To gain access to the account using a brute-force attack, the program would try all the available words it has to gain access to the account. Another type of brute-force attack is a program that runs through all letters and/or letters and numbers until it gets a match.

The attack takes advantage of the fact that the entropy of the values is smaller than perceived. For example, while an 8 character alphanumeric password can have 2.8 trillion possible values; many people will select their passwords from a much smaller subset consisting of common words and terms.

Brute force attacks common to web applications.

- Brute Forcing Log-in Credentials
- Brute Forcing Session Identifiers
- Brute Forcing Directories and Files
- Brute Forcing Credit Card Information

VI. SQL Injection

SQL Injection is an attack technique used to exploit applications that construct SQL statements from user-supplied input. When successful, the attacker is able to change the logic of SQL statements executed against the database.

Structured Query Language (SQL) is a specialized programming language for sending queries to databases. The SQL programming language is both an ANSI and an ISO standard, though many database products supporting SQL do so with proprietary extensions to the standard language. Applications often use user-supplied data to create SQL statements. If an application fails to properly construct SQL statements it is possible for an attacker to alter the statement structure and execute unplanned and potentially hostile commands. When such commands are executed, they do so under the context of the user specified by the application executing the statement. This capability allows attackers to gain control of all database resources accessible by that user, up to and including the ability to execute commands on the hosting system.

VII. SQL Injection Using Dynamic Strings

A web based authentication form might build a SQL command string using the following method:

```
SQLQuery = "SELECT Username FROM Users WHERE Username = '"
SQLQuery = SQLQuery & strUsername
SQLQuery = SQLQuery & "' AND Password = '"
SQLQuery = SQLQuery & strPassword
SQLQuery = SQLQueryd & """
strAuthCheck = GetQueryResult(SQLQuery)
```

Example 1 - Dynamically built SQL command string

In this code, the developer combines the input from the user, str User Name and str Password, with the logic of the SQL query. Suppose an attacker submits a login and password that looks like the following:

Username: xyz
Password: india OR '1'='1

The SQL command string built from this input would be as follows:

```
SELECT Username FROM Users WHERE Username = 'xyz'
AND Password = 'india' OR '1'='1
```

This query will return all rows from the user's database, regardless of whether "xyz" is a real user name or "india" is a legitimate password. This is due to the OR statement appended to the WHERE clause. The comparison '1'='1' will always return a "true" result, making the overall WHERE clause evaluate to true for all rows in the table. If this is used for authentication purposes, the attacker will often be logged in as the first or last user in the Users table.

VIII. SQL Injection In Stored Procedures

It is common for SQL Injection attacks to be mitigated by relying on parameterized arguments passed to stored procedures. The following examples illustrate the need to audit the means by which stored procedures are called and the stored procedures themselves.

```
SQLCommand = "exec LogonUser '" + strUserName + "','" + strPassword + """
```

Example 2 - SQL Injection in stored procedure execute statement

Using a stored procedure does not imply that the statement used to call the stored procedure is safe. An attacker could supply input like the following to execute additional statements:

Username: xyz
Password: '; DROP TABLE Users--

The generated SQL Command string would be:

```
exec Log on User 'xyz','; DROP TABLE Users--'
```

On a Microsoft SQL server, using the above SQL command string will execute two statements: the first will likely not identify a user to log in, and the second would remove the Users table from the database. It should be noted that attempts to escape dangerous characters are not sufficient to address these flaws, even within stored procedures.

IX. Blind SQL Injection

Blind SQL Injection is used when a web application is vulnerable to an SQL injection but the results of the injection are not visible to the attacker. The page with the vulnerability may not be one that displays data but will display differently depending on the results of a logical statement injected into the legitimate SQL statement called for that page. This type of attack can become time-intensive because a new statement must be crafted for each bit recovered. There are several tools that can automate these attacks once the location of the vulnerability and the target information has been established.

One type of blind SQL injection forces the database to evaluate a logical statement on an ordinary application screen.

```
SELECT sname FROM student WHERE sname = 'pqr' AND 1=1;
```

This statement will result in a normal page while

```
SELECT sname FROM student WHERE sname = 'pqr' AND 1=2;
```

This will likely give a different result if the page is vulnerable to a SQL injection. An injection like this may suggest to the attacker that a blind SQL injection is possible, leaving the attacker to devise statements that evaluate to true or false depending on the contents of another column or table outside of the SELECT statement's column list.

X. Conclusions

Many web servers have vulnerabilities that can result in unauthorized access to a system by means of submitting malicious requests. Code injection attacks, especially SQL injection attack is one of the well-known issue of vulnerabilities. Controlling the malicious SQL code/script on the web application and maintaining the end user privacy is still a key challenge for the web developer. These issues must be considered seriously by the web developers involved in developing websites using databases. Many web applications have weak points that can result in unauthorized access to a database by inserting malicious code. In this research work we have described how an attacker can exploit the web application by using SQL injection attack to get confidential information from a database.

References

[1]. Haron, H. "Cyber stalking The social impact of social networking technology" , IEEE 2010, pp 237 - 241
[2]. Limsaiprom and Prajit, "Social network anomaly and attack patterns analysis", IEEE 2010, pp 1 – 6.
[3]. Xuelei Wu Coll. of Transp. & Manage., DaLian Maritime Univ., Dalian Jia Chen and Bilan Rong, "Web Service Architecture and Application Research", IEEE 2009, pp 1–5.
[4]. Kasra Amirtahmasebi, Seyed Reza Jalalinia and Saghar Khadem, "A Survey of SQL Injection Defence Mechanisms", IEEE 2009, pp 1-6.
[5]. Ntagw Abira Lambert & Kang Song Lin, "Use of Query Tokenization to detect and prevent SQL Injection Attacks", IEEE 2010, pp 438-440.
[6]. Aurelien Francillon, Claude Castelluccia, "Code Injection Attacks on Harvard-Architecture Devices" Alexandria, Virginia, USA: ACM, 2008.
[7]. Raju Halder Agostino Cortesi, "Obfuscation-based Analysis of SQL Injection Attacks", IEEE 2010, pp 931-938.
[8]. Adam Kie˙ zun, Philip J. Guo, Karthick Jayaraman and Michael D. Ernst, "Automatic Creation of SQL Injection and Cross-Site Scripting Attacks", IEEE 2009, pp 199-209.
[9]. Qian XUE and Peng HE, "On Defence and Detection of SQL SERVER Injection Attack", IEEE 2011.
[10]. Li Shan, Dong Xiaorui and RaoHong, "An Adaptive Method Preventing Database from SQL Injection Attacks", IEEE 2010, pp 352-355.
[11]. Jeom-Goo Kim, "Injection Attack Detection using the Removal of SQL Query Attribute Values", IEEE 2011.
[12]. M. Muthuprasanna, Ke Wei and Suraj Kothari, "Eliminating SQL Injection Attacks - A Transparent Defense Mechanism", IEEE 2006.
[13]. R. Ezumalai and G. Aghila. Combinatorial Approach for Preventing SQL Injection Attacks. IACC, 2009.
[14]. MeiJunjin. An approach for SQL Injection vulnerability detection. IEEE, 2009.
[15]. ManjuKaushik, GazalOjha,"Attack Penetration System for SQL Injection",InternationalJournal of Advanced Computer Research, Volume-4 Number-2 Issue-15 June-2014.
[16]. Lakhtaria K. (2011) Protecting computer network with encryption technique: A Study. International Journal of u- and e-service, Science and Technology 4(2).
[17]. Chhajed, Urmi, and Ajay Kumar. "Detecting Cross-Site Scripting Vulnerability and performance comparison using C-Time and E-Time." International Journal of Advanced Computer Research (IJACR) 4 (2014): 733-740.
[18]. Stalling, W. (2005) Cryptography and network security principles and practices, 4th edition Prentice Hall.
[19]. Shannon, C. E. (1948) Communication Theory of secrecy systems. Bell System Technical Journal.
[20]. Ashutosh Kumar Dubey,Animesh Kumar Dubey, MayankNamdev, Shiv Shakti Shrivastava,"Cloud-User Security Based on RSA and MD5 Algorithm for Resource Attestation and Sharing in Java Environment", CONSEG 2012.
[21]. Ashutosh Kumar Dubey, Animesh Kumar Dubey, VipulAgarwal, YogeshverKhandagre, "Knowledge Discovery with a Subset-Superset Approach for Mining Heterogeneous Data with Dynamic Support",Conseg-2012.
[22]. PreetiKhare, Hitesh Gupta, "Finding Frequent Pattern with Transaction and Occurrences based on Density Minimum Support Distribution", International Journal of Advanced Computer Research (IJACR), Volume-2 Number-3 Issue-5 September-2012.
[23]. Elia, I.A.; Fonseca, J.; Vieira, M., "Comparing SQL Injection Detection Tools Using Attack Injection: An Experimental Study," Software Reliability Engineering (ISSRE), 2010 IEEE 21st International Symposium on , vol., no., pp.289,298, 1-4Nov. 2010.
[24]. Kai-Xiang Zhang; Chia-Jun Lin; Shih-Jen Chen; Yanling Hwang; Hao-Lun Huang; Fu-Hau Hsu, "TransSQL: A Translation and Validation-Based Solution for SQL-injection Attacks," Robot, Vision and Signal Processing (RVSP), 2011 First International Conference on , vol., no., pp.248,251, 21-23 Nov. 2011.
[25]. Jagnere, P., "Vulnerabilities in social networking sites," Parallel Distributed and Grid Computing (PDGC), 2012 2nd IEEE International Conference on, pp.463, 468, 6-8 Dec. 2012.

IOSR Journal of Computer Engineering (IOSR-JCE) is UGC approved Journal with Sl. No. 5019, Journal no. 49102.

* Jyoti Kushwaha1 A Survey on Malware & Session Hijack Attack over Web Environments." IOSR Journal of Computer Engineering (IOSR-JCE) 20.2 (2018): 30-35