International Journal of Computing and Corporate Research
ISSN (Online) : 2249-054X
Volume 5 Issue 3 May 2015
International Manuscript ID : 2249054XV5I3052015-10

A SURVEY OF SQL INJECTION ATTACK: CAUSE OF WEBSITE VUNERABILITIES

Kanika
Computer Science & Engineering Department
Universal Group of Institution College, Lalru

Sunita Rani
Computer Science & Engineering Department
Universal Group of Institution College, Lalru

Abstract:- Web applications provide end users with client access to server functionality through a set of Web pages. Web applications are currently subject to a .excess of successful attacks, such as cross-site scripting, cookie theft, session riding, browser hijacking, and the recent self-propagating worms in Web-based email and social networking sites. This paper looks at Web application vulnerabilities attack. SQL injection attack is one of the main reason of the Websites vulnerabilities.SQL injection is an attack methodology that targets the data residing in a database through the firewall that shields it. The attack takes advantage of poor input validation in code and website administration.The aim of this paper is to study about SQL injection attacks process.

Keywords:-Tautologies,Union Query, Blind Injection, Timing Attacks,Website Vulnerabilities.

I. INTRODUCTION

A security exploit in which an attacker is trying to get access to Web resources by the

insertion of structured query language code to a web form input box is called SQL injection. Most of the times what happens is that when a user his or her name and password in the log in box provided for them, those values get inserted into the SELECT query. Then they are matched with the expected values to grant or deny access. If no way is provided to prevent input other than name and password, attackers can send their own request in this box and download whole database etc in illicit ways.

Website Vulnerabilities: - Vulnerability means a flaw in a system that can put down it release to attack. Vulnerability can also refer to any type of weakness in a computer system itself, in a set of actions, or in anything that leaves information security exposed to a threat.

SQL:-SQL (Structured Query Language) is a textual language used to interact with relational Database.

The typical unit of execution of SQL is the ‘query’, which is a collection of statements that typically return a single ‘result set’. SQL statements can modify the structure of databases and manipulate the contents of databases by using various DDL, DML commands respectively. SQL Injection occurs when an attacker is able to insert a series of SQL statements into a query by manipulating data input into an application.

Definition of SQLIA Most web applications today use a multi-tier design, usually with three tiers: a presentation, a processing and a data tier. The presentation tier is the HTTP web interface, the application tier implements the software functionality, and the data tier keeps data structured and answers to requests from the application tier. Meanwhile, large companies developing SQL-based database management systems rely heavily on hardware to ensure the desired performance. SQL injection is a type of attack which the attacker adds Structured Query Language code to input box of a web form to gain access or make changes to data. SQL injection vulnerability allows an attacker to flow commands directly to web applications underlying database and destroy functionality or confidentiality.

In logic, a tautology is a formula which is true in every possible interpretation. In a tautology-based attack the code is injected using the conditional OR operator such that the query always evaluates to TRUE. Tautology-based SQL injection attacks are usually bypass user authentication and extract data by inserting a tautology in the WHERE clause of a SQL query. The query transform the original condition into a tautology, causes all the rows in the database table are open to an unauthorized user. A typical SQL tautology has the form "or <comparison expression>", where the comparison expression uses one or more relational operators to compare operands and generate an always true condition. If an unauthorized user input user id as abcd and password as anything' or 'x'='x then the resulting query will be:
select * from user_details where userid = 'abcd' and password = 'anything' or 'x'='x'
The example is already explained in "Example of a SQL injection attack"

Web applications provide end users with client access to server functionality through a set of Web pages. Web applications are currently subject to a excess of successful attacks, such as cross-site scripting, cookie theft, session riding, browser hijacking, and the recent self-propagating worms in Web-based email and social networking sites . This paper looks at Web application vulnerabilities attack. SQL injection attack is one of the main reason of the Websites vulnerabilities.SQL injection is an attack methodology that targets the data residing in a database through the firewall that shields it. The attack takes benefits of poor input validation in code and website administration. SQL Injection Attacks o comes when an attacker is able to insert a series of SQL statements in to a ‘query’ by manipulating user input data in to a web-based application, attacker can take benefits of web application programming security flaws and pass unexpected malicious SQL statements through a web application for execution by the backend Database. The aim of this research is to study about SQL injection attacks process.

Website Vulnerabilities:-There are various types of website vulnerabilities [8]

1. Cross site scripting: - The "most prevailing and malicious" Web application security vulnerability, XSS flaw occur when an application sends user data to a web browser without first validating or encoding the content. This lets hackers execute malevolent scripts in a browser, hire them take control user sessions, disfigure Web sites, insert hostile content and conduct phishing and malware attacks.

2. Injection defect: - When user-supplied data is sent to interpreters as part of a command, hackers catch the interpreter which interprets text-based commands into executing unintentional commands. "Injection flaws allow attackers to read, update, create or delete any random data available to the application. In the worst-case situation, these flaws permit an attacker to entirely compromise the application and the necessary systems, even bypassing extremely nested firewalled environments."

3. Execution of Malicious file:-Hackers can accomplish remote code execution, totally conciliation a system. Any type of Web application is vulnerable if it accepts filenames or files from users. The vulnerability can be most widespread with PHP, a normally used scripting language for Web development.

4. Insecure direct object reference:-Attackers operate direct object references to gain unauthorized access to other objects. It happens when form parameters include references to objects such as database records or keys, directories. Banking Web sites generally use a customer account number as the primary key, and May interpretation account numbers in the Web interface. "An attacker may attack these parameters simply by guessing or searching for another valid key. Frequently, these are in arranging in nature."

5.Cross site request imitation:-"Simple and devastating," this attack takes control of victim's browser when it is logged onto a Web site, and sends malevolent requests to the Web application. Web sites are enormously susceptible, partly because they tend to authorize requests based on session cookies functionality. Banks are potential targets. "Ninety-nine percent of the applications on the Internet are susceptible to cross site request forgery."

6. Information escape and improper error handling: - Error messages that applications generate and present to users are useful to hackers when they violate privacy or accidentally leak information about the program's configuration and internal workings. Web applications will often reveal information about their internal state through detailed or debug error messages. Often, this information can be leveraged to start or even automate more potent attacks.

7. Broken authentication and session management:-User and administrative account can be hijacked when applications are unsuccessful to protect credentials and session token from starting to last. Watch out for privacy violations and the undermining of authorization and accountability controls. Imperfection in the main verification mechanism are not exceptional but weaknesses are more frequently introduced through ancillary authentication functions such as password management, and time out, remember me, secret question, log out and account update.

8. Insecure cryptographic storage: - Many Web developers are unsuccessful to encrypt responsive data in storage even though cryptography is a key part of most Web applications. When encryption is there it's frequently inefficiently designed with incompatible cipher. These flaws can go ahead to exposé of vulnerable data and compliance violations.

9. Insecure communications: - This is a failure to encrypt network traffic when it's necessary to protect sensitive communications. Attackers can be able to access unprotected conversations, with transmissions of credentials and responsive information.PCI standards require encryption of credit card information transmitted over the Internet.

10. Failure to limit URL access: - Web pages are supposed to be restricted to a small subset of confidential users such as administrators. There is no real protection of these pages, and hackers can discover the URLs by making well-informed guesses. The attacks target these liabilities are called forced browsing which encompasses guessing links and brute force techniques to find unprotected page.

2.Literature Review

Fu et al., in [12] propose a Static Analysis Framework in order to detect SQL Injection Vulnerabilities. SAFELI framework aims at identifying the SQL Injection attacks during the compile-time. This static analysis tool has two main advantages. Firstly, it does a White-box Static Analysis and secondly, it uses a Hybrid-Constraint Solver. For the Whitebox Static Analysis, the proposed approach considers the byte-code and deals mainly with strings. For the Hybrid- Constraint Solver, the method implements an efficient string analysis tool which is able to deal with Boolean, integer and string variables.

Thomas et al., in [13] suggest an automated prepared statement generation algorithm to remove SQL Injection Vulnerabilities (SQLIVs). They implement their research work using four open source projects namely: (i) Net-trust, (ii) ITrust, (iii) WebGoat, and (iv) Roller. Based on the experimental results, their prepared statement code was able to successfully replace 94% of the SQLIVs in four open source projects. However, the experiment was conducted using only Java with a limited number of projects. Hence, the wide application of the same approach and tool for different settings still remains an open research issue to investigate.

In [14], Haixia and Zhihong propose a secure database testing design for Web applications. They suggest a few things; firstly, detection of potential input points of SQL Injection; secondly, generation of test ca1ses automatically, then finally finding the database vulnerability by running the test cases to make a simulation attack to an application. The proposed methodology is shown to be efficient as it was able to detect the input points of SQL Injection exactly and on time as the authors expected. However, after analyzing the scheme, we find that the approach is not a complete solution but rather it needs additional improvements in two main aspects: the detection capability and the development of the attack rule library

In [15] Ruse et al. propose a technique that uses automatic test case generation to detect SQL Injection Vulnerabilities. The main idea behind this framework is based on creating a specific model that deals with SQL queries automatically. In addition, the approach identifies the relationship (dependency) between sub-queries. Based on the results, the methodology is shown to be able to specifically identify the causal set and obtain 85% and 69% reduction respectively while experimenting on few sample examples. Moreover, it does not produce any false positive or false negative and it is able to detect the real cause of the injection.

In [16], Roichman and Gudes, in order to secure Web application databases, suggest using a fine-grained access control to Web databases. They develop a new method based on fine-grained access control mechanism. The access to the database is supervised and monitored by the built-in database access control. This approach is efficient in the fact that the security and access control of the database is transferred from the application layer to the database layer.

In [17], Shin et al. suggest SQLUnitGen, a Static-analysisbased tool that automate testing for identifying input manipulation vulnerabilities. They apply SQLUnitGen tool which is compared with FindBugs, a static analysis tool. The proposed mechanism is shown to be efficient (483 attack test cases) as regard to the fact that false positive was completely absent in the experiments. However for different scenarios, false negatives at a small number were noticed. In addition to that, it was found that due to some shortcomings, a more significant rate of false negatives may occur “for other applications”. Hence, the authors talk about concentrating on getting rid of those significant false negatives and further improvement of the approach to cover input manipulation vulnerabilities as their future works.

In [4] Gary McGraw explains the types of security as Software Security with the term Application Security. The main difference between them is that Application Security is the process of protecting the software after it has been completed and deployed by finding and fixing the security problems after they have occurred, while Software security is the process of building a secure software by designing, planning, coding and implementing taking in consideration common security threats [4].

3.Sql Injection

Sql injection is a software liability that occurs when data entered by users is sent to the sql predictor as a part of an SQL query. There are different types of attack that depending on the goal of attacker are performed together.

Types of various attack

I. Tautologies: Tautology is a formula which will be true in every situation. In such case conditional OR operator is used such that query will always give the result as true. In such a method, query is generated using WHERE clause. Query will change the unique situation into a tautology. It makes every row n the database to open to unauthorized user access. One or more operators are used to compare operands and results are forever true[1]. Suppose attacker gives input as ‘abcd’ as log in id and ‘anything’ as password, then query generated becomes:

Select * from user_data where userid ‘abcd’ and password=’anything’

4. Logically Incorrect Queries

Its main purpose is to identification of inject able parameters. It is also used to extract data and to identify database. In this method attacker tries to gather all the data about the structure of database. If the query sent to database is not correct, some application server returns the error message. Attacker uses it as advantage and injects code vulnerable parameters. They will create syntax and type conversion along with logical error. Type error helps in identifying certain column data types. Table names and column names are exposed more often by the logical error.

This type of attacks makes use of UNION query. Resultant will give a dataset that is return of result of original first query and the result of query that was injected. Basic rules used in such a query are as follows:

- Number of columns along with their order must be the same.
- 2Data types of columns should be same.
- First query is returned by column names.

UNION by default eliminates the duplicate rows. But to prevent this you can use ALL keyword along with the UNION [5].

5. Stored Procedure

These types of attacks try to run stored procedures of the database. Stored procedures are usually in the database to extend the functions of database. It also gives interaction with operating system. Attacker may use other injection techniques in the beginning to know the type of database [6]. Once it is known what kind of database is used in the backend then by the use of injected code attacker may execute various procedures. Developers write all the stored procedures that is the reason why they don't make database open to injection attacks. Suppose an attacker is injecting '; SHUTDOWN; -- into the user id or password field then following code is generated in SQL:

Select * from user_data where userid='abcd' and password=''; SHUTDOWN;--'

This command will make the system shut down.

6.Piggy Backed Queries

It is used to modify dataset, extract data, remote command execution, service denial. In this original query is mixed up with additional query. First query will be valid one and following queries are injected ones. System might allow multiple statements in one query. Suppose attacker writes abcd in userid and drop table xyz—in password field. After completion of first query, injected query is run.

7.Blind Injection This method asks questions regarding true or false statements to database and based on the response given by the application answer is determined [7]. It is same as normal injection. When web page doesn't get data from database, attacker steals data by using sequence of true or false questions.

VI. Timing Attacks

Here database will pause for some time. After that it returns the results. That tells user that query was executed successfully. An attacker will enumerate each letter using following logic:If the first is 'A', waiting time is 10 seconds.If the first letter is 'B', waiting time is 10 seconds etc.

CONCLUSION

In this paper various types of sql injection attacks are discussed. To prevent these attacks it is necessary to know about them. By knowing how they work and operate, preventive measures can be developed for them. They are dangerous because they can be easily automated. Attacker can access whole database.

REFERENCES

[1] Bhavani M. Thuraisingham, Chris Clifton, Amar Gupta, Elisa Bertino, Elena Ferrari. “Directions for Web and E-Commerce Applications Security.” Proceedings of the 10th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (pp.200-204). 2001.

[2] Bojan Jovicic, Dejan Simic. “Common Web Application Attack Types and Security Using ASP.NET. “ ComSIS Consortium. 2006.

[3] Cosmin Striletchi, Mircea-Florin Vaida. “Enhancing the Security of Web Applications.” Proceedings of the 25th International Conference on Information Technology Interfaces (pp. 463-468) . 2003.

[4] Gary McGraw. "Software Security." IEEE Security & Privacy (vol. 2, pp. 80-83). 2004.

[5] J.D. Meier. "Web Application Security Engineering." IEEE Security &Privacy (vol. 4, no. 4, pp. 16–24). 2006.

[6] J.D. Meier, Alex Mackman, Michael Dunner, Srinath Vasireddy, Ray Escamilla and Anandha Murukan. “Improving Web Application Security: Threats and Countermeasures.” Microsoft Corporation. 2003.

[7] Jeff Zadeh, Dennis DeVolder. “Software Development and Related Security Issues.” Proceedings of IEEE Southeastcon. 2007.

[8] John R. Maguire, Gilbert Miller. “Web-Application Security: From reactive to proactive.” IT Professional (vol. 12, no. 4, pp. 7-9). 2010.

[9] Mark Curphey, Rudolph Araujo. “Web application security assessment tools.” IEEE Security & Privacy (vol. 4, no. 4, pp. 32-41). 2006.

[10] Myat Myat Min, Khin Haymar Saw Hla. “Security on Software Life Cycle using Intrusion Detection System.” 6th AsiaPacific Symposium on Information and Telecommunication Technologies APSITT 2005 Proceedings. 2005.

[11] Shin-Jer Yang, Jia-Shin Chen. “A study of security and performance issues in designing web-based applications.” IEEE International Conference on e-Business Engineering. 2007.

[12] Fu, X., Lu, X., Peltsverger, B., Chen, S., Qian, K., and Tao, L., A Static Analysis Framework for Detecting SQL Injection Vulnerabilities. Proc. 31st Annual International Computer Software and Applications Conference 2007 (COMPSAC 2007), 24-27 July (2007), pp. 87-96.