Volume 4, Issue 3, March 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

SQL Injection Attacks and Countermeasures

Simaranjeet Singh, Prof. Jitendra Kr. Jindal
Department of Computer Science & Engineering,
School of Engineering & Technology,
IFTM University, Moradabad India

Abstract— SQL Injection has become a common issue with database-driven web sites. The flaw is easily detected, and easily exploited, and as such, any site or software package with even a minimal user base is likely to be subject to an attempted attack of this kind. As SQL Injection attack techniques have become more common, more ambitious, and increasingly sophisticated, so there is a deep need to find an effective and feasible solution for this problem in the computer security community. Detection or prevention of SQLIAs is a topic of active research in the industry and academia. To achieve those purposes, automatic tools and security systems have been implemented, but none of them are complete or accurate enough to guarantee an absolute level of security on web applications. Application code can be written in a manner that defeats SQL injection. However, ensuring that each and every piece of database access code is immune to SQL injection in normal size applications that also include 3rd party components is essentially impossible. One of the important reasons of this shortcoming is that there is a lack of common and complete methodology for the evaluation either in terms of performance or needed source code modification which in terms is an over head for an existing system.

Keywords— SQL Injection Attacks, encryption algorithm.SQL Injection, Tautology, SQLIA, Blind injection, piggy backing,
Countermeasures

I. INTRODUCTION

As computers become better understood and more economical, every day brings new applications. Many of these new applications involve both storing information and simultaneous use by several individuals. Thus information security had become the most thriving and fast-moving discipline. **Information security** means protecting information and information systems from unauthorized access, use, disclosure, disruption, modification, perusal, inspection, recording or destruction. Information security is concerned with the confidentiality, integrity and availability of data regardless of the form the data may take: electronic, print or other forms. Essentially, the attack is accomplished by placing a Meta character into data input to then place SQL commands in the control plane, which did not exist there before. This flaw depends on the fact that SQL makes no real distinction between the control and data planes SQL injection is a security vulnerability that occurs in the persistence/database layer of a web application. This vulnerability is derived from the incorrect escaping of variables embedded in SQL statements.

II. RECENT SQL INJECTION ATTACKS SCENARIO

A. DETAILED ANALYSIS OF ATTACK OCCURRENCES

iMPERVA, Hacker Intelligence Initiative, Monthly Trend Report #4 monitored SQL Injection attack attempts over the last nine months against a set of 30 real web applications, of all sizes, across different industries. On average, we have identified 53 SQLi attacks per hour and 1,093 attacks per day. Since July we have observed a slight increase in SQLi attacks, averaging 71 attacks per hour and 1,589 per day

Table 1: Statistics of SQLiA occurrences

		Average	Min	Max	Median	Standard Deviation
Attacks / hour	Since December 2010	53	1	7950	9	197
	Since July	71	1	4937	8	259
Attacks / day	Since December 2010	1093	44	21724	600	1909
	Since July	1589	106	8204	1162	1508

The large standard deviation indicates significant fluctuations in the number of attacks. For example, we witnessed on most days applications suffered less than 3,000 attacks and occasionally less than 500. However, there were a few days where about 8,000 SQLi attempts were concentrated against the applications. Analysing the number of hourly and daily SQLi attacks in the last nine months, we can see a trend of 50-100 attacks per hour (as shown in Figure 1) and about 1,100 daily attacks on average, with an occasional spike.

The concentration of a huge number of attack attempts during a short period of time is a clear indication that these attacks are automated.

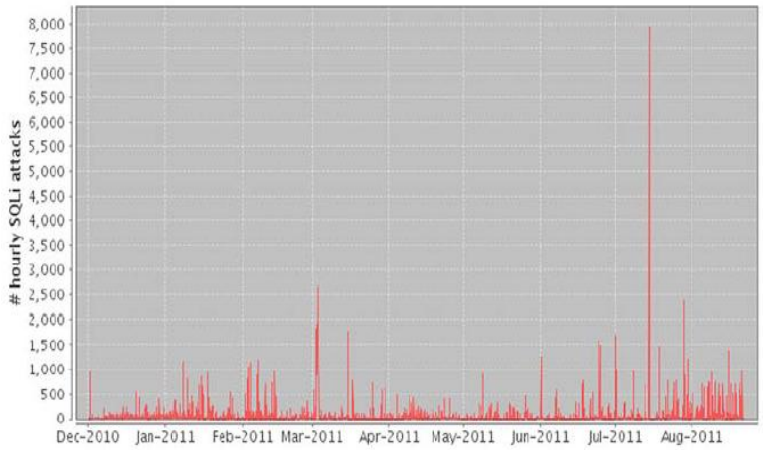

Fig1: Hourly occurrences of SQLi attacks

B. Threat to Company Data

i. McGuireWoods LLP
Julia E. Lewellen and Phillip R. Rees
USA
June 28 2011
MCGUIREWOODS

Sony recently received its third major attack by hackers. This attack was claimed by Lulz Security, which claimed to use a simple SQL injection. So, what is a SQL injection? And has your company undertaken a data protection and data security audit that would ensure precautions are taken against an SQL injection, amongst other threats?

An SQL injection, or a "Structured Query Language" injection, is a common technique used by hackers to steal data from within a database. This technique involves a hacker entering SQL commands or code into an input box, such as a login and/or password box, to obtain access to a company's database.

Typically, when a legitimate website visitor enters his or her username and password, an SQL query is generated from such information and submitted to the database. If the information is verified, the user is allowed access; otherwise, access is denied.

Unless there are mechanisms in place to block input other than names and passwords, hackers can use such input boxes to send their own requests to a database, which, for example, may store customers' and employees' confidential and private information that is subject to the Data Protection Act 1998. This may entail viewing information in the database and/or deleting it.

The simplicity of SQL injections and the use of automated tools have resulted in increased popularity in this technique and others that are similar. A privacy, data protection and security audit is necessary to identify potential gaps and weaknesses in your organization's privacy, data protection and security regime, which include vulnerability to SQL injections, and that ensure your company is complying with data protection principles.

ii. MySQL Website Falls Victim to SQL Injection Attack

Oracle's MySQL.com customer website was apparently compromised over the weekend by a pair of hackers who publicly posted usernames, and in some cases passwords, of the site's users.

By Jeremy Kirk
Mon, March 28, 2011

IDG News Service — Oracle's (ORCL) MySQL.com customer website was apparently compromised over the weekend by a pair of hackers who publicly posted usernames, and in some cases passwords, of the site's users.

III. DETAILED ANALYSIS ON CLASSIFICATION OF SQL INJECTION ATTACKS

A. Order wise Classification of SQL Injection Attacks:

a) First order Attacks:

The first order attacks are basic attacks. First-order Injection Attacks are when the attacker receives the desired result immediately, either by direct response from the application they are interacting with or some other response mechanism, such as email etc.

b) Second order Attacks:

Second Order injection Attack is the realization of malicious code injected into an application by an attacker, but not activated immediately by the application .The malicious inputs are seeded by the attacker into a system or database. This is used to indirectly trigger an SQLIA which is used at later time. The attacker usually relies on where the input will be subsequently used and thus crafts his attack. Second order injection leaves a ticklish job of detection and prevention. This is because the point of injection is different from the point where the attack actually manifests it. In the second order attacks attacker insert the malicious code into the application but not activated immediately by the application. These attacks and the SQL injection do not associate with each other in real time. The malicious user inputs are first stored in a system and then wait for being used by an application in the future.

There are various attacks classes in the second order attack.

- **Frequency based Primary Application**: Attacks in this class frequently target the other users of the primary application. For example, topmost searched items, latest popular article.
- **Frequency based Secondary Application**: This class includes application that did not initially receive the injected code, but instead process submission from an application and represent this material for statistical review. Attacks within this type targets on the system administrators.
- **Secondary Support Application:** This class includes application used to internally support primary application. Attacks within this class typically target internal application users and attack activation may be accelerated through social engineering vectors.
- **Cascaded Submission** Application: this class includes application that makes **use** of multiple client submission within single processing statement. Attacks within this class typically utilize SQL code statements to manipulate the search request and consequently target backend database resources.

c) Lateral injection:

Using Lateral SQL Injection, an attacker can exploit a PL/SQL procedure that does not even take user input. When a variable whose data type is date or number is concatenated into the text of a SQL statement, there still is a risk of injection. Lateral SQL injection is a new type of SQL injection attack. Security researcher David Litchfield first disclosed this type of attack at the Black Hat Washington conference last February, 2008s.In a SQL injection; attackers create specially crafted search terms that trick the database into running SQL commands. Previously, security experts thought that SQL injections would work only if the attacker was inputting character strings into the database, but Litchfield has shown that the attack can work using new types of data, known as date and number data types. In the past, known SQL injection only occurred at places that took user inputs. Lateral SQL injection does not require a vulnerable SQL statement to have user input parameters. It targets the SQL statements that use DATE or NUMBER data types. This type of attack is conducted by converting data format using NLS session parameters such as NLS_DATE_FORMAT or NLS_NUMERIC_CHARACTERS.

B. Classification of Attack against Database:

On the basis of attack against the database management system, SQL injection attack can be classified as:

a) Tautologies:

A SQL tautology is a statement that is always true. Tautology based SQL injection attacks are usually used to bypass user authentication or to retrieve unauthorized data by inserting a tautology into a conditional statement. 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. The general goal of a tautology-based attack is to inject SQL tokens that cause the query's conditional statement to always evaluate the true.

For example,

Select * from Employee where EmpName = „ ‟ or 1=1 -- „ and

Password= „xxxxx‟.

The "or 1=1" is the most commonly known tautology

A SQL tautology is a statement that is always true. Tautology-based SQL injection attacks are usually used to bypass user authentication or to retrieve unauthorized data by inserting a tautology into a conditional statement

b) Inference:

In this attack, the query is being modified into the form of an action which is executed based on the answer to a true/-false question about data values in the database. In this type of injection, attacker try to attack a site that is enough secured not to provide acceptable feedback via database error messages when an injection has succeeded. The attacker must use a different method to obtain the response from the database since database error messages are unavailable to him. In this situation, the attacker injects commands into the site and then observes how the function/response of the website changes. By carefully observing the changing behaviour of the site, attacker can extrapolate not only vulnerable parameters, but also additional information about the values in the database. By this type of attack, intruders change the behaviour of a database or application. There are two well known attack techniques that are based on inference: blind injection and timing attacks.

c) Proposed Architecture – SQLAvoid

Below is the proposed architecture on which our designed algorithm to prevent SQLIAs will run. We had named the architecture as SQLAvoid .There can be moderate changes made in it based on the needs of algorithm.

SQLAvoid is implemented in J2EE platform and consists of an

- HTTP request interceptor module
- Thread-local storage module
- SQL interceptor module
- SQLIA detector module
- SQL lexer module

As illustrated in Figure below, the original data flow (HTTP request to web application to JDBC driver to database) is modified when SQLAvoid is deployed into a web server.

First, the references to the program objects representing incoming HTTP requests are saved into the current thread-local storage.

Second, the SQL statements composed by web applications are intercepted by the SQL interceptor and passed to the SQLIA detector module.The detection module then retrieves the corresponding HTTP request from thread-local storage and examines the request to determine whether it contains an SQLIA. If so, the SQL interceptor prevents the malformed SQL statement from being submitted to the database.

All main modules of SQLAvoid are shown in Figure, and are explained below:

- **HTTP Request interceptor** is implemented as a servlet filter- a component type introduced in Java Servlet specification version 2.3. This module intercepts HTTP requests and stores an internal reference to the object representing the intercepted HTTP request in the corresponding thread-local storage. The stored reference is retrieved later by the SQLIA detector module when it processes the intercepted SQL statements.
- **Thread-local storage** is static or global memory local to a thread-each thread gets a unique instance of thread-local static or global variables. Given that web servers are commonly implemented as multi-threaded processes that handle multiple concurrent HTTP requests at the same time, the SQLIA detector module needs a way to find the corresponding HTTP request for each intercepted SQL statement. Since both request handling and query generation are processed in the same thread, the thread-local storage provides an adequate mechanism for a one-to-one mapping between an HTTP request and the corresponding SQL statement.
- **SQL interceptor** extends P6Spy. This open-source module intercepts and logs SQL statements issued by web-application programming logic before they reach the JDBC driver. We have extended P6Spy to invoke the SQLIA detector when SQL statements are intercepted.
- **SQLIA detector** takes an intercepted SQL statement as input, retrieves the corresponding HTTP request object from the thread-local storage, passes the intercepted SQL statement to the SQL lexer for tokenization, and then performs detection according to Algorithm which is under development. If an SQLIA is identified the detector indicates this fact to the SQL interceptor, which throws a necessary security exception to the web application, instead of letting the SQL statement through.
- **SQL lexer i**s implemented as a lexical analyzer. This module converts a sequence of characters into a sequence of tokens. The SQL lexer module is used to perform lexical analysis of intercepted SQL statements. Given an SQL statement,the SQL lexer generates a set of tokens with the corresponding token types.

For example, by giving the following SQL statement as an input: “UPDATE books SET book name='SQLIA', price=100 WHERE book id=123", the SQL lexer will generate the following set of tokens and the corresponding token types

No.	Token	Token Type
1.	UPDATE	[IDENTIFIER]
2.	Books	[IDENTIFIER]
3.	SET	[IDENTIFIER]
4.	book name	[IDENTIFIER]
5.	=	[OPERATOR – EQUALS]
6.	'SQLIA'	[LITERAL - STRING]
7.	,	[COMMA]
8.	Price	[IDENTIFIER]
9.	=	[OPERATOR - EQUALS]
10.	100	[LITERAL - INTEGER]
11.	WHERE	[IDENTIFIER]
12.	book id	[IDENTIFIER]
13.	=	[OPERATOR - EQUALS]
14.	123	[LITERAL - INTEGER]

The SQL lexer is used by the SQLIA detector module to find a set of literal types in the intercepted SQL statement, such as LITERAL - STRING in line 6 and LITERAL - INTEGER in line 10 and line 14.

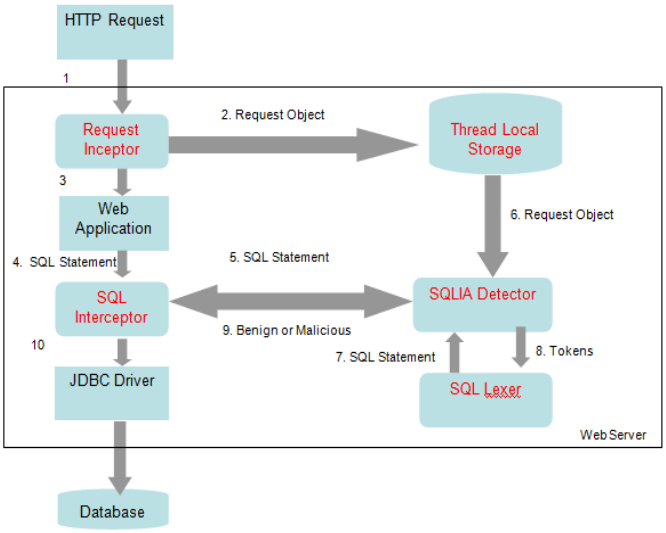

Fig2: SQL Avoid Architecture

IV. CONCLUSIONS

Nowadays, many businesses and organizations use web applications to provide services to users. Web applications depend on the back-end database to supply with correct data. However, data stored in databases are often targets of attackers. SQL injection is a predominant technique that attackers use to compromise databases. During the project, we will conduct a survey of different types of SQL injection attacks, and will built applications and Oracle database environment to illustrate how they work. Many countermeasures against SQL injection attacks have been proposed and implemented by academic researchers, developers and databases venders. In this project, we will able to get a good insight into the existing techniques and approaches used to secure and protect web applications and database systems. We will also identify techniques that can be used to secure my applications and database system against SQL injection attacks and will apply them. After understanding and identifying the working mechanisms as well as advantages and disadvantages of current techniques, we will try to implement an algorithm which eliminates the need of source code modification and provide an improved overall efficiency and will benefit future work in this area.

REFERENCES

1. secerno.com,"SQL Injection Attack: A Security Threat", http://www.secerno.com/?pg=SQL-Injection#2
2. IBM, IBM Internet Security SystemsX-Force 2009 Trend & Risk Report, Jan 2010, http://www-935.ibm.com/services/us/iss/xforce/trendreports/xforce-2008-annual-report.pdf
3. Wikipedia, "SQL injection" http://en.wikipedia.org/wiki/SQL_injection
4. Book- Silberschatz, Korth, Sudarshan. Database system concepts, 4th Edition
5. Book - SQL Injection Attacks and Defense by Justin Clarke
6. AMNESIA: Analysis and Monitoring for NEutralizing SQLInjection Attacks by William G.J. Halfond and Alessandro Orso ,College of Computing Georgia Institute of Technology
7. A Survey On Sql Injection: Vulnerabilities, Attacks, And Prevention Techniques By Diallo Abdoulaye Kindy and Al-Sakib Khan Pathan Department of Computer Science, International Islamic University Malaysia, Malaysia Published at : 2011 IEEE 15th International Symposium on Consumer Electronics ; 46-471
8. SQL Injection Detection and Prevention Tools Assessment By Atefeh Tajpour CASE Center University Technology Malaysia Kuala ,Lumpur, Malaysia ; Mohammad Zaman Heydari ,IT & Management Dep UCSI University Kuala Lumpur, Malaysia ; Maslin Masrom ,CASE Center University Technology Malaysia Kuala ,Lumpur, Malaysia , Suhaimi Ibrahim ,CASE Center UTM University Kuala Lumpur, Malaysia Published at : 978-1-4244-5540-9/10/$26.00 ©2010 IEEE ; 518-52
9. Shielding Against SQL Injection Attacks Using ADMIRE Model By Prof (Dr.) Sushila Madan Supriya Madan ,Department of Computer Science ,Lady Shri Ram College ,University of Delhi ; Supriya Madan ,Department of Information Technology,Vivekananda Institute of Professional Studies (Affiliated to GGSIP University) Published at 2009 First International Conference on Computational Intelligence, Communication Systems and Networks ; 978-0-7695-3743-6/09 $25.00 © 2009 IEEE ; 314-320
10. A Survey of SQL Injection Defense Mechanisms By Kasra Amirtahmasebi, Seyed Reza Jalalinia and Saghar Khadem, Chalmers University of Technology, Sweden Presented at: Institute of Electrical and Electronics Engineers in 2009

