Volume 4, Issue 10, October 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Assessment of SQL Injection Solution Approaches

Jignesh Doshi, Bhushan Trivedi
MCA, LJ Institute of Management Studies MCA,
GLS Institute of Computer Technology, India

Abstract— Online banking (Net banking) has become the most common way for executing financial transactions in recent past. As a result, the web has become Centre for e-marketplace and network as a backbone. Cyber attacks have increased in numbers and became more sophisticated. Almost 60% of web resources are vulnerable. Web applications have been facing challenges against data (loss, confidentiality and integrity) and availability. In this paper, the authors performed an evaluation of Web application security solution techniques. We conclude that there is utmost urgency to use hybrid (mixed) security solutions for web applications against attacks.

Keywords— Vulnerability,SQL Injection,Attack,Security,Threat

I. INTRODUCTION

Day by day more and more transactions are taking place via the internet. Web applications have become backbone for business and economy in recent days. The web application usage and attacks are growing hand to hand. As per Netcraft survey [1], the number of attacks has increased drastically in the past few years. Almost 75% of total attacks come from web applications and 60% of web sites are vulnerable. As per CERT, Approx. 59% of cyber security incidents are related to web applications [3].

As per OWASP the top 5 attacks out of 10 are related to information theft and database security and one of the top most database attack is SQL Injection [2]. There attacks are now sophisticated and evolved. The web security Company Trust wave studied (5) computer security of the last 30 years and discovered (refer Table I).

Table I Evolution of Computer Security

Decade →	1980s	1990s	2000s
Evolution (DB Applications)	Desktop	Network Based	Web / Internet Based
Physical Attack Vector	Server room access	Weak Password	a) Sensitive Data Left in Plain View b) Unlocked Accessible Computer Systems c) Data Cabling Accessible from Public Areas
Network Attack Vector	None	Infected Network	a) Weak or Blank Administrator Passwords b) Database Servers Accessible c) ARP Cache Poisoning
Application Attack Vector	None	a) Logic Flaws b) Authorization Bypass	a) SQL Injection b) Logic Flaws c) Authorization Bypass
Asset on Risk	Files/Data	Web and Database Servers	Web and Database Servers and Data

As per OWASP, Structure Query Language (SQL) based attacks is most commonly used attacks and become under top ten attacks for past few years. An attacker can use SQL statements to retrieve or manipulate data. This paper provides an evaluation of various solutions exists to detect and prevent SQL based attacks.

This paper is organized as follows: In section 2, we discuss the importance of attacks, and suggested tools for SQL Injection attacks. In Section 3, we review techniques and tools. In section 4, we discuss finding. Conclusion and future work are provided in section 5 & 6 respectively.

II. LITERATURE SURVEY

2.1 Motivation

As per the ISC internet domain survey (July 2012), internet host count has grown 9 times in just 11 years [4][17]. More and more number of financial transactions are taking place via online banking or net banking.

As per OWASP, 2013, SQL Injection based attacks are in the top – 10 since past few years. They are easy to execute. Developers and organizations have realized the importance of web application security and it cannot be ignored.

As per Gartner statistical analysis, 75% attacks come from web applications and 2/3 web applications are vulnerable [18].

2.2 SQL Injection

2.2.1 SQL Injection- SQLI (crafted or modified SQL statements) are used most commonly in database attacks and are in top -10 attacks since 2010 [1] -[3], [5] -[7]. The Major reason why SQLI based attacks are more popular is simple in execution, SQL statement includes user input and may contain hard coded values.

2.2.2 Types of SQLI Attacks

In SQL Injection (SQLI) attacker craft or modify or suppliant SQL statements to gain access of database and retrieve data. Attacker enters queries and additions to the database via web form (application input field). There are four types of SQLI attacks: SQL manipulation, Code injection, Function call Injection and buffer overflow. SQLI can be classified into two categories: first order and second order attacks.

Table II: Types of SQLI [5]

Types of First Order Attack	Attack Type	Working Methods	Purpose
Tautologies	SQL Manipulation	Crafted queries	Get data / information
Logically Incorrect Queries	SQL Manipulation	Using error messages to find useful data for injection	*Get table name, column name details along with the error message*
Union Query	*Code Injection*	*Existing query is augmented with the safe union query*	*To Get data / information*
Stored Procedure	Function Call Injection	Execute built in stored procedures	*Get meta data and data*
Piggybacked Queries	Code Injection	Additional malicious queries are inserted	To perform DML operations
Inference: - Blind Injection	Code Injection and Buffer overflow	Exploiting database vulnerability based on answer true or false	Retrieve / Steal data
Inference - Timing Attacks	Code Injection and Buffer overflow	Using response time	Collect data
Alternate Encodings	SQL Manipulation	Use hex characters in place of regular characters	Collect data

2.2.3 **SQL Injection attacks Mechanisms**

Crafted / modified SQL statements can be injected into vulnerable web site using different mechanisms like: inject using user input, inject using cookies, inject through server variables or second order injection [10].

2.2.4 Intent of SQL Injection attacks

SQLI attacks can be classified based on intent of attacking. The major intent behind the attack are: 1) Database finger printing (collected metadata information), Retrieving sensitive or regular data, performing data manipulation, Performing attacks (like Denial or service, bypassing authentication etc.), or executing remote commands [10].

2.2.5 Root Cause of SQL Injection attacks

Three Major root causes of SQL injection attacks are: a) Poor Web site administration b) Weak Input validation techniques c) Non standard error reporting.

2.2.6 Impact of SQLI Attacks

The impact of SQLI may be severe to business in terms of Loss of data, Data Secrecy and data tempore. While indirectly it may cause in loss of customer trust and loyalty.

III. SQLI DETECTION AND PREVENTION TECHNIQUES

SQLI works at the application level so it is difficult to handle it easily. Researchers have suggested many solutions for SQLI. Most common approaches used to handle web application attacks are categorized into defensive coding or hardening (filtering) [10]. However, they are not enough to stop attacks [11].

For the purpose of our study, we have divided solution approaches into 4 categories; 1) Defensive Coding, 2) Detection and Prevention techniques, 3) Monitoring and 4) Hardening.

3.1 Defensive Coding Techniques

This approach, focus is to prevent the attack. The defensive coding approach can be used to mitigate the major root cause of insufficient input validation. Defensive coding is implemented using key best practices like input type checking, encoding the inputs, positive input matching, identification of all input sources, etc. [7]. Defensive coding is one of the best ways to prevent SQLI.

Key problems identified are summarized as below [7] [9] [10] [11] [12] [13]:

1) The chances of human error are always possible like developer forgot to add checks at some locations etc.
2) It requires skilled manpower to implement
3) Extra efforts and cost required
4) Can be implemented when there is a new web application development
5) Need lots work to prevent new types of attacks

3.2 Detection and Prevention

This approach, the focus is to detect and prevent the attack. Using this approach we can overcome human error and improper type checking problems of defensive coding [10]. Proposed Solutions of researchers are of different types as below:

3.2.1 Static Code Checkers

This type of solution checks static code with runtime checks queries generated dynamically i.e. perform runtime monitoring. JDBC-checker technique statically check the type correctness of dynamically runtime generated queries [19]. It mitigates risk "improper type checking". Wasserman proposed solution, which perform static analysis of checking tautology [20]. CANDID dynamically check the query structure for user input to the actual query issued to detect SQLIA [21].

Major Drawbacks of this approach are [7] -[14], [19] -[21]:

1) The scope is limited of detecting and preventing tautologies
2) Cannot catch other types of attacks

3.2.2 Static and Dynamic Analysis

This type of techniques performs static and dynamic runtime monitoring [7]. In static phase all queries which can be executed in the application are used to build a model, In dynamic phase, tool intercept queries executed and checks against the model. Queries not matching will not be executed. In SQL Guard and SQL Check verify the query structure before and after user input and allow only legal queries [22] [23]. AMNESIA performs static and runtime monitoring of queries. Here queries are intercepted before they are sent to database[24].

Major Drawback of this approach is that the Success depends on accuracy of static analysis build a model [9] -[13], [22] -[24].

3.2.3 Black-box testing dynamically

The focus of this type solutions is to test web applications [7]. This type of solutions first use web crawler to identify the points which can be used for SQLIA [9] -[13]. Secondly, it will build attacks using known patterns and lastly apply attack techniques. Improved version of this technique is penetration testing.. WAVES, SecuBat, AppScan, ScanDo, and WebInspect are in this category.

Major Drawback of this approach is [9] -[13], [25]:

1) The testing cannot guarantee completeness of code and need code for testing.
2) Need to know exact knowledge of the application

3.3 Intrusion Detection Systems

This type of solutions uses training sets to build models and monitor application for queries that do not match the model [7] [26]. Mainly used for detection of SQLIA.

The drawbacks of this approach are [9] -[13], [26]:

1) Success is dependent on the quality of training set used for model building.
2) May generate a large number of false positive and negatives

3.4 Hardening (Filtering)

In this type of tools, set of input validation rules and constraints are defined to filter data [7] [13]. The tool will monitor queries executed and filters the matching one.

The drawbacks of this approach are [9] -[13], [27]:

1) The developer needs to know what data needs filtering and filtering rules to apply
2) Skill of developer is required

IV. TECHNIQUES EVALUATION

4.1 Approach Evaluations

In this section, evaluates approaches discussed in the previous section using different aspects of software engineering. We consider approaches and key aspects of software engineering for evaluation.

Table III Comparison of approaches with respect to Software Engineering Key aspects

Approach	Developer		Suitable for New Web Application	Scalable for New Attack	Source code Required	Web Server Access Required
	Skill	Effort				

			Development	**Types**		
Defensive Coding	Yes	Yes	Yes	No	Yes	-
Static Analysis	Yes	Yes	Yes	No	Yes	-
Static and Dynamic analysis	Yes	Yes	Yes	No	Yes	-
Black-box/ Penetration Testing	Yes	Yes	Yes	Yes	Yes	Yes
IDS	Yes	Yes	Yes	Yes	No(*)	Yes
Hardening	Yes	Yes	Yes	Yes	No(*)	Yes

Not: (*) – Not required for deployment of technique.

We can summarize that all techniques required skilled Developers, Development efforts and cost. Also need either source code or access to the web server.

4.2 Review Findings and Gaps

Our Findings from the study are:

1. Tools focus is towards SQL statements or input validation.
2. Every solution needs developer skills
3. Every solution proposed has led towards developer overhead like
   i. The extra effort required in the development, modification/enhancement
   ii. The extra effort required during testing
   iii. Extra efforts required if the new SQLI type of attack discovered.

Future solutions should focus on

1. Towards sensitive data exposure prevention like password, credit card number etc.
2. Prevention of static super database accounts and packages access
3. Code Fix is Not a Solution for Every Applications i.e. utility
4. Scalable solutions as attacks are never always similar but found complex and hybrid

V. CONCLUSIONS

All these techniques either attempt to keep the integrity of SQL structure or examine the correctness of the SQL statements. Also, existing security policies are not sufficient. We need a mechanism, which will check the overall application for SQLI risk and provide alerts. Which is independent of application and consisting of all types of SQLI checks.

Last but not less significant,

1. Tomorrow, if any, new type of SQLI attack is found, each suggested tool will require humongous efforts to mitigate it.
2. Unable to cover all SQL statements of applications as many third party tools are used in the application for which it will be difficult
3. In real word, 80% of total coding efforts are towards maintaining of existing code. Most of the tools suggested are good when we do new application development, but very difficult when existing system is modified, enhanced or migrated.

REFERENCES

[1] OWASP. Top Ten projects 2013.https://www.owasp.org/index.php/Category: OWASP Top Ten Project: accessed 31st May 2014.
[2] OWASP: https://www.owasp.org/index.php/Top_10_2014-A1-Injection: accessed 31st May 2014.
[3] Common Weakness Enumeration: http://cwe.mitre.org/data/definitions/89.html : accessed 3rd June 2014
[4] SQL Injection: http://www.us-cert.gov/security-publications/sql-injection: accessed 31st May 2013
[5] Puspendra Kumar, R K Pateriya: *A Survey on SQL injection attacks, Detection and Prevention Techniques,, ICCCNT 2012, 26- 28 July 2012 Coimbtore, India, IEEE-20I80*
[6] Stop SQL Injection Attacks Before They Stop You: http://msdn.microsoft.com/en-us/magazine/cc163917.aspx:, accessed 3rd June 2013
[7] A. Tajpour, M. Massrum and M. Z. Heydari, "Comparison of SQL Injection Detection and Prevention Techniques", 2nd International Conference on Education Technology and Computer (ICETC), 2012
[8] Rahul Johri and Pankaj Sharma " A Survey on Web Application Vulnerabilities(SQLIA and XSS) Exploitation and Security Engine for SQL Injection", IEEE 2012 , 978-0-7695-4692-6/12
[9] Diallo Abdoulaye and Al-Sakib Khan Pathan, " A Survey on SQL Injection: Vulnerabilities, attacks AND Prevention Techniques", IEEE 15th International Symposium on Consumer Electronics, 2011
[10] William G J Halfond, Jeremy Viegas, A Orso: A Classification of SQL Injection attacks and Countermeasures, Copyright IEEE 2006

[11] A. Tajpour, M. JorJor Zade Shooshtari, "Evaluation of SQL Injection Detection and Prevention Techniques", 2010 International Conference on Computational Intelligence, Communication systems and Networks, 978-0-7695-4158-7/10

[12] A. Tajpour, M. Masrom and M. Z. Heydari, Suhaimi Ibrahim: "SQL Injection Detection and Prevention Tools Assessment", 978-1-4244-5540-9/10 @ 2010 IEEE

[13] A S Yeole, B B Meshram: Analysis of Different Technique for Detection of SQL Injection", International Conference and Workshop on Emerging Trends in Technology (ICWET 2011), ACM 978-1-4503-0449-8/11/02

[14] You Yu, Yuanyuan Yang, Jian GU and Lian Shen:"Analysis and Suggestions for the Security of Web Applications", 2011 International conference on Computer Science and Network Technology, 978-1-4577-1587-7/11 IEEE

[15] Kasra Amirtahmasebi, Syed Reza Jalalinia and Saghar Khadem: A Survey of SQL Injection Defense Mechanism, 2009 The Institute of Electrical and Electronics Engineers, Inc

[16] Raju Halder, Agostino Cortesi: Obfuscation-based Analysis of SQL Injection Attacks; IEEE 2010, 978-1-4244-7755-5/10

[17] Gartner Press Releases: http://www.gartner.com

[18] Netcraft, Total Sites Across All domains August 1995 – 2010, http://news.netcraft.com

[19] C. Gould, Z. Su, and P. Devanbu. JDBC Checker: A Static Analysis Tool for SQL/JDBC Applications. In *Proceedings of the 26th International Conference on Software Engineering (ICSE 04) –Formal Demos*, pages 697–698, 2004

[20] G. Wassermann and Z. Su. An analysis framework for security in web applications. In Proceedings of the FSE Workshop on Specification and Verification of Component-Based Systems (SAVCBS 2004), pages 70–78,October 2004.

[21] Bisht, P., Madhusudan, P., and Venkatakrishnan, V.N., CANDID: Dynamic Candidate Evaluations for Automatic Prevention of SQL Injection Attacks. *ACM Transactions on Information and System Security*, Volume 13 Issue 2, (2010), doi>10.1145/1698750.1698754.

[22] G. Buehrer, B.W. Weide, P.A.G. Sivilotti, Using Parse Tree Validation to Prevent SQL Injection Attacks, in: 5th International Workshop on Software Engineering and Middleware, Lisbon,Portugal, 2005, pp. 106–113

[23] Z. Su and G. Wassermann. The Essence of Command Injection Attacks in Web Applications. In *The 33rd Annual Symposium on Principles of Programming Languages (POPL 2006)*, Jan. 2006

[24] William G.J. Halfond, Allesandro Orso,"AMNESIA: Analysis and Monitoring for NEutralizing SQL Injection Attacks", ACM, USA,2005, pp 174-183

[25] MeiJunjin: An approach for SQL injection vulnerability detection, 2009 Sixth International Conference on Information Technology: New Generations, 978-0-7695-3596-8/09 $25.00 © 2009 IEEE

[26] Kemalis, K. And T. Tzouramanis. SQL-IDS: A Specification-based Approach for SQL Injection Detection. SAC'08. Fortaleza, Ceará, Brazil, ACM (2008), 2153 2158

