ISSN: 2278 – 7798
International Journal of Science, Engineering and Technology Research (IJSETR)
Volume 5, Issue 9, September 2016

Framework to evade SQLIA for multiple development platforms

Santosh VG[#1], Sushma KR[#]
[#1]*Programmer, Computer Science Department, Coorg Institute of Technology, Ponnampet, Kodagu, Karnataka, India-571216*

[#2]*Asstitant Professor, Electronics and Communication Department, Coorg Institute of Technology, Ponnampet, Kodagu, Karnataka, India-571216*

***Abstract*: A software developer would think of an API / TOOL / framework that avoid attackers to inject malicious queries to his / her database with no source code modification. It's an approach to design an integrated tool to guard SQLIA for multiple development platforms using little efficient technique like Active guard filtration model and Service detector model, Data Credential Identifier, Hacker Identity etc.**

***Index Terms*: Web services, XML, XPATH.**

I. Introduction

Databases have become attractive and very lucrative targets for hackers to hack into. Attackers alter by passing SQL statements to the database as parameters and enable not only steal data from your database, but also modify and delete it. A database is vulnerable to SQLIA when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not validated properly. SQL injection attacks are also known as SQL insertion attacks. Our approach is to develop a framework that provides developer or web application programmers to implement an easy integration with no source code modification.

Information is the most important business asset in today's environment and achieving an appropriate level of Information Security. SQL-Injection Attacks (SQLIA's) are one of the top most threats for web application security models. For example financial deception, stealing confidential data, hack

website, sabotage, espionage and cyber threats. The hasty rise in fraud perpetrated over the internet has brought about the classification of nine types of frauds, developed from the data reported by Internet Crime Complaint center. The IC3 website has seen a significant increase in frauds involving the exploitation of valid online banking credentials belonging to small and medium sized businesses.

Most of the Internet threats are from software application vulnerabilities and flaw in the design of software application system. Vulnerabilities in software may allow a third party or program to gain unauthorized access to some resource. Software application vulnerability control is one of the most important parts of computer and network security. Virus program use vulnerabilities in operating system and application software to gain unauthorized access. Intruders use vulnerabilities in operating system and application software to gain unauthorized access, to attack and damage other systems. Hence, avoiding software vulnerability is a major countermeasure to protect software applications from internet threat. It is difficult to design and build a secure web application until the designer knows the possible threats in application. Hence, threat modeling is recommended to be part of the design stages in web application.

The purpose of threat modeling is to analyze the application's architecture and identify the potentially vulnerable areas. Developers must follow secure coding techniques to develop secure, robust, and hack-tough solutions. The design and development of application layer software must be supported by a secured network and hosting systems. Weak input validation is an example of an application layer vulnerability, which can result in SQL injection attack. SQL injection is a technique for exploiting web applications that uses client-supplied data in SQL queries without stripping potentially harmful characters.

I. Related Work

2.9.1 Web Vulnerability Scanning

Web vulnerability scanners move slowly and scan for web vulnerabilities by using application software mediator. These tools evade attacks against web based applications, usually in a black-box traces, and detect and avoid vulnerabilities by observing the applications' response to the injection attacks .However, without exact knowledge about the internal structure of

applications; a black-box move toward might not have enough test cases to depiction existing vulnerabilities and also have fake positives.

2.9.2 Intrusion Detection System (IDS)

Valeur and colleagues propose the use of an Intrusion Detection System (IDS) to detect SQLIA. Their IDS system is based on a device learning technique that is trained using a set of application queries. The technique builds replica of the typical queries and then monitors the application at runtime to identify queries that do not match the model in that it builds expected query Models and then checks dynamically-generated queries for compliance with the model.

2.9.3 Combined Static and Dynamic Analysis.

AMNESIA is a model-based technique that combines static analysis at runtime. In its static phase, AMNESIA uses static scrutiny to build models of the different types of queries an application can legally creates at each point of access to the database. In its dynamic phase, AMNESIA interrupts all queries before they are sent to the database and checks each query against the statically designed models.

II. Proposed System

The proposed technique is to develop a tool to detect and prevent SQLIA irrespective of any kind of databases used for development in various platforms. Here in our project our motive is to implement in two different databases. Our aim is to bring out this tool which can be integrated to any kind of development platforms using any kind of databases especially for open source IDEs. This would minimize the crucial part of security code practicing to the developers to guard the databases used in web applications with no source modification.

3.1 Active Guard Model

Active Guard Model in application layer build a Susceptibility detector to detect and prevent the Susceptibility characters or Meta characters to prevent the malicious attacks from accessing the data from database in various platforms.

3.2 Service Detector Model

Service Detector Model in application layer validates user input from XPATH_Validator where the Sensitive data are stored from the database. The user passed input fields data is compared with the data existed in

XPATH_Validator if it is identical then the Authenticated /legitimate user is allowed to proceed.

3.2.1 Database to XML generator

Stores the contents of database to xml file, the application loads for the first time.

3.2.2 X-PATH Validator.

It verifies the user input with the xml document contents.

3.3 Database Credential Identifier

It identifies the database server and its connection credentials used in the application.

3.4 Hacker Identity Detector

It identifies from where the hacker tries to attack and which kind of malicious activities he is performing.

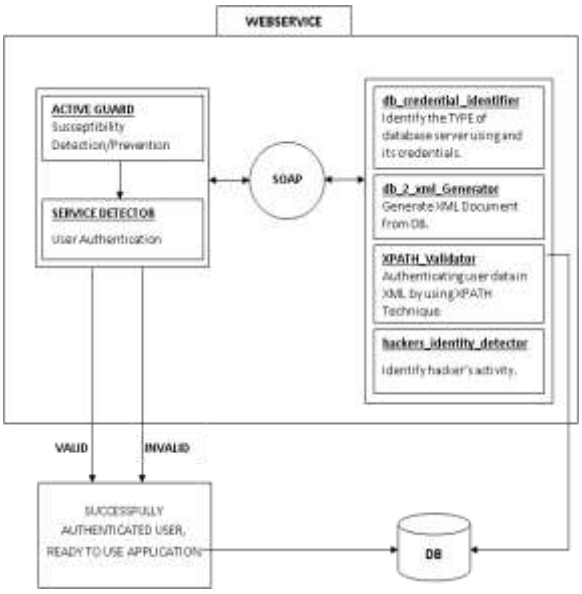
3.4.1 Get Hacker Identity

This module is used to find out the hacker's identity based on the public IP.

3.4.2 Malicious Activity

This model keeps track of malicious injects carried out by the hackers.

III. Implementation



The framework designed for multiple databases with various databases integration technologies to the development platforms. For an example, an ASP.NET programmer would need an assembly supported by .net framework whereas a java programmer would need assemblies supported by JDK. Hence the proposed system requires various IDEs for different platform to design this tool with different databases. The same piece of technique can be used for every upcoming programming technology like Android, Nodejs, AppEngines etc.

3.1 Database Credential Identifier

DCI (Database Credential Identifier) will identify what type of Database Server a developer uses in the application and its credentials as follows,

1. Create a XML File in the App_Data Path in Solution Explorer.
2. Holds Database Credential.
   I) Hostname.
   II) Database Type i.e. MySQL, SqlServer etc.
   III) Database Name.
   IV) Database Object Name holds Username/ Password fields.
3. Retrieve Database Credential and Choose Database Definition Type in the Web Service.

4.2 Active Guard

1. Read INPUT String of ARRAY.
2. Initialize str,i;
3. Find WORDS,

i) Select.

ii) –

iii) DROP

iv) ;

v) Insert

vi) Delete

vii) Xp_

viii) ‘

4. for (i = 0 to (Count Array -1))

{

str= Replace (str, ARRAY.Item (i), "",, ,CompareMethod.Text)

}

5. Return str;

4.3 **Service Detector**

1. Initialize Dataset, Data Adapter.
2. Create New Dataset.
3. Fill Data Adapter with Dataset.
4. Write All Users to the XML file.

4.4 **X-PATH Validator**

1. Read User Name, Password.
2. Initialize count=0;
3. Read XML File Content (User Name, Password).
4. If(Username, Password) found

Count=1;

Else GOTO Step 3;

IV. Conclusion

SQLIA detection and evading SQLIA tool is developed in the motive that this will help developers to reuse it for any kind of Database Server used for development. Web Service Oriented XPATH Technology standardize the user application by means of

its effective characteristics like, xml database is faster compared to other databases. The feature of this tool like, Web Service Oriented XPATH Technology Authentication Technique checks the user input with valid database and do not affect database directly, then the validated user input field is allowed to access the web application.

A developer can integrate this tool with no source modification to prevent from SQLIA. We use few efficient modules to prevent SQLIA's that is Active guard filtration model and Service detector model, Data Credential Identifier, Hacker Identity.

V. References

[1] William G.J. Halfond and Alessandro Orso, "AMNESIA: Analysis and Monitoring for Neutralizing SQLInjection Attacks", ASE'05, November 7–11, 2005

[2] William G.J. Hal fond and Alessandro Orso, "A Classification of SQL injection attacks and countermeasures",proc IEEE int'l Symp. Secure Software Engg., Mar. 2006.

IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.1, January 2011 205

[3] Muthuprasanna, Ke Wei, Suraj Kothari, "Eliminating SQL Injection Attacks - A TransparentDefenceMechanism",

SQL Injection Attacks Prof. Jim Whitehead CMPS 183. Spring 2006, May 17, 2006

[4] William G.J. Hal fond, Alessandro Orso, "WASP: Protecting Web Applications Using Positive Tainting and Syntax-Aware Evaluation IEEE Software Engineering, VOL. 34, NO.1January/February 2008

[5] K. Beaver, "Achieving Sarbanes-Oxley compliance forWeb applications", http://www.spidynamics.com/support/white papers/, 2003

[6] C. Anley, "Advanced SQL Injection In SQL Server Applications," White paper, Next Generation Security Software Ltd., 2002.

[7] W. G. J. Halfond and A. Orso, "Combining Static Analysis and Runtime Monitoring to Counter SQL Injection Attacks," 3rd International Workshop on Dynamic Analysis,2005, pp. 1- 7

[8] Z. Su and G. Wassermann, "The Essence of Command Injection Attacks in Web Applications," 33rdACM SIGPLAN-SIGACT Symposium on Principles of

Programming Languages, 2006, pp. 372-382.

[9] 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, 2004.

[10] P. Finnigan, "SQL Injection and Oracle - Parts 1 & 2," Technical Report, Security Focus, November 2002.

http://securityfocus.com/infocus/1644

[11] F. Bouma, "Stored Procedures are Bad, O'kay," Technical report, Asp.Net Weblogs, November 2003.

http://weblogs.asp.net/fbouma/archive/2003/11/18/38178.aspx.

[12] E. M. Fayo, "Advanced SQL Injection in Oracle Databases," Technical report, Argeniss Information Security,

Black Hat Briefings, Black Hat USA, 2005.

[13] C. A. Mackay, "SQL Injection Attacks and Some Tips on How to Prevent them," Technical report, The Code Project,

January 2005. http://www.codeproject.com/cs/database/qlInjectionAttacks.asp.

[14] S. McDonald. SQL Injection: Modes of attack, defense, and why it matters. White paper, GovernmentSecurity.org, April 2002.

[15] V. B. Livshits and M. S. Lam. Finding Security Errors in C# Programs with Static Analysis. In Proceedings of the

14th Usenix Security Symposium, pages 271–286, Aug. 2005.

[16] F. Valeur and D. Mutz and G. Vigna "A Learning-Based Approach to the Detection of SQL Attacks," In Proceedingsof the Conference on Detection of Intrusions and Malware Vulnerability Assessment (DIMVA), July 2005.

[17] Kals, S., Kirda, E., Kruegel, C., and Jovanovic, N. 2006. SecuBat: a web vulnerability scanner. In Proceedings of the 15th International Conference on World Wide Web. WWW '06. ACM Press, pp. 247-256.

****