International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 05 | May-2018 www.irjet.net p-ISSN: 2395-0072

An Intrusion Detection System for SQL Injection Attack on SaaS applications

Prashanth C[1], Nithin R[2], Prajwal Naresh[3], Shobhitha G[4]

[1,2,3 &4]*Department of Computer Science and Engineering Global Academy of Technology, Bangalore, Karnataka, India.*

Abstract - *At the present time where there is a huge increase in demand for computation and storage of data, Cloud Computing has emerged as the front runner in providing a very inexpensive and easy solution. It is because of the Software as a Service (SaaS), it has eradicated physical servers and applications being stored on-site. This basically means that server management responsibility is with an outsourced provider and we just have to subscribe to the SaaS applications to use it. But since user data will be in the hands of the outsourced provider we are having a huge number of attack. The most sought-after method used to expose the vulnerabilities is the SQL injection attack (SQLIA). SQL injection attacks allows attackers to spoof their identity, tamper with the existing data, allow the complete disclosure of all data on the system, destroy the data or make it otherwise unavailable, and also become administrator of the database. The proposed framework acts as a filter mechanism between the SaaS application and the database server. An intrusion detection system(IDS) is established which checks the query retuned by the application is malicious(Backlist) or not. To achieve this query validation, we propose an algorithm to check the query for proper syntax, proper grammar and it checks for all known SQLIA queries. The IDS is deployed in the live cloud, so any SaaS applications can just use this service as a plugin.*

***Key Words*: Software as a Service(SaaS), SQLIA, IDS, Blacklist, Cloud**

1. INTRODUCTION

The computing with rapid changes in the whole processing and storage technologies and the success in the communication networks such as Internet has decreased the overall cost of the computing resources, more efficient and even more ubiquitously available. This technological modification has enabled the development of a new computing paradigm known as cloud computing, in which resources are shared by multiple system over the communication network. Cloud computing is nothing but a set of services that are provided to a the end customers over a medium maybe network on a rental basis and with the power to scale up or down their service necessities. Cloud computing services are delivered by a 3rd party provider who owns the complete infrastructure. Cloud computing has turned up as a brand-new model for hosting and delivering services over the net. Cloud computing becomes an attraction within the business world because it doesn't need to set up a provision and business may be established with a small amount. In cloud computing, services are shared over the communication network that is. Internet. In conception of cloud computing multiple computers will access services and data which is stored on the Cloud. So that the overall expense for installation of the software and cost for maintenance is thus reduced because there is no need of installing on each end user's computer and might be accessed from different places.[1] The age old definition of cloud computing that it is a model for enabling convenient and on-demand network access to shared pool of configurable computing resources for example, networks, applications, servers, storage, and services that can be rapidly provisioned and also released with minimal management effort or any service provider interaction. Let us consider, an associate of any organization needs to make sure that every worker of the company has the right software and hardware to do their job. To shop for new computers and install software on each system is expensive. Hence, using a service cloud is an easy way to store all required resources of the company so that each of the employees can access the resources whenever in need.

The glory of net and its merits are being extremely cloaked by the downside associated with it. Of all the major threats is the Internet Vulnerability which leads to data modification and data thefts. Many web applications store the data in these databases and retrieve and update information whenever needed. Hence these applications are extremely vulnerable to several varieties of attacks, one of them being SQL injection Attacks(SQLIA). SQL injection attack occurs when an attacker causes the web application to generate SQL queries that are functionally different from what the user interface programmer intended. For example, consider an application which deals with author details.

select ID, fname, lname from authors;

This statement will retrieve the 'id', 'fname' and 'lname' columns from the 'authors' table, returning all rows in the table. The 'result set' could be restricted to a specific 'author' using 'where' clause.

select id, fname, lname from authors where fname = 'Mark' and lname = 'Lisa';

An important point to note here is that the string literals 'Mark' and 'Lisa' are delimited with single quotes. Here the literals are given by the user and so they could be modified. They become the vulnerable area in the application. Now, to

drop the table called 'authors', a malicious literal can be injected into the statement like

fname: Mar'; drop table authors-- lname:

Now the statement becomes,

select id, fname, lname from authors where fname = 'Mar'; drop table authors-- and lname = ' ';

and this will be the query executed

Since the first name is ending with a delimiter ' and - - is given at the end of the input, all the remaining commands following - - is neglected as it will be commented out. The output of this command will be the deletion of the table named 'authors', which is not the exact intended result from the server database.

The main objective of this paper is to handle SQL injection attack in any form. SQLIA can be done by using the input form or even from the URL, considering all the new techniques used for evading signatures.

2. RELATED WORK

The SQLIA is divided into three types which are the in band, out of band and the inference attack [8]. Here in the in-band attack, it is nothing but the process of extraction of information over similar channels in between server and client. The out of band attacks are the ones which do not work on the same channel but instead they have distinct channels to retrieve the data. The Inference attack involves in the process of intentionally waiting for an error message from the server side or either by invoking an error message as a return parameter from the server, thus giving the complete information of the server and also giving access to the complete database. In this paper, the author has discussed how the data could be collected using the inference method.

The usual Static code checkers in the current world like the JDBC-checker [7] is a unique method for statically checking all the typo mistakes of a dynamically generated SQL query. But the main drawback of this is it is going to find out only a single SQL vulnerability which is caused by the improper type checking mechanism of the input.

There is another model called the combined static and dynamic analysis for example, AMNESIA [4]. It is considered as a model-based technique. AMNESIA combines both the static analysis of the query and the runtime-monitoring. The motivation to build this approach is mainly because of two reasons. It is because the source-code contains information which is enough to infer models of the expected, syntactically correct SQL queries generated within the application. And, because the SQL injection Attack, by injecting the extra SQL statements into the queries, will violate the whole model. The AMNESIA works with its static part deployed in constructing a model of correct queries that could possibly be generated by the application by using various techniques of program analysis. Whereas the dynamic part, the AMNESIA monitors all the dynamically generated queries at run-time and validates the queries for compliance with the statically generated model.

Even with vast variety of the SQL detection system the rise of SQL injection attacks has not been cut down. Recently in India, A famous French Security researcher by the name Elliot Anderson exposed BSNL. This was an SQL Injection Attack and it exposed data of 47000 employees. This is one of the latest attack which was exposed on March 4th, 2018 [16]. Hence it is correct to say that almost all the dangerous threats out there involve some sort of SQL injection attack down the line.

The existing system has several limitations such as security measures are provided only for PAAS and SAAS, but not for IAAS [5], all of them requires a costly analysis and modifications of the source code, and solution applies only for specific language or environment

3. PROPOSED SYSTEM

In this paper, a framework called as SQL Intrusion Detection System, is proposed which detects SQL injection targeting software applications deployed in public IaaS cloud providers. Reusable by any IaaS providers, proposed solution is language independent and the solution doesn't require any modification to the source code.

3.1 System Architecture

Any application running on IaaS will have its own database. The application will be used by both users and the attacker. A framework SQLIIDaaS is used to provide security to this Application. This framework is accessed through a Web Service.

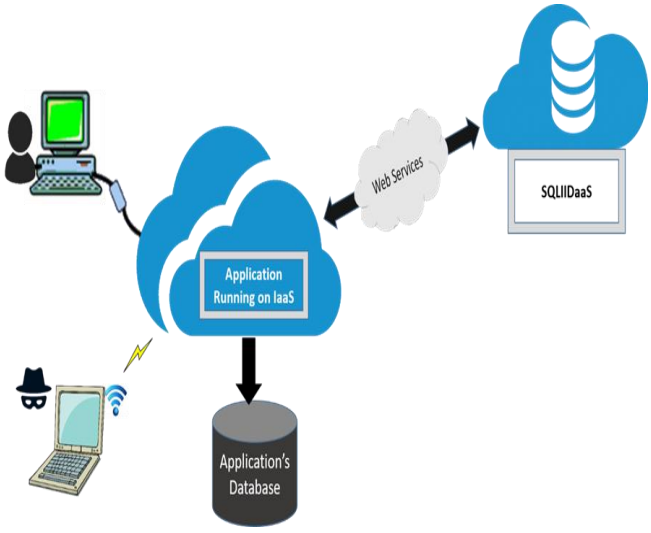

Fig-1: System Architecture

3.2 System Design

Major divisions in this paper are

A. Data Access Layer

Data access layer is the one which exposes all the possible operations on the data base to the outside world. It will contain the DAO classes, DAO interfaces, POJOs, and Utils as the internal components. All the other modules of this project will be communicating with the DAO layer for their data access needs

B. Account Operations

Account operations module provides the following functionalities to the end users of our project.

Step 1: Register a new seller/ buyer account

Step 2: Login to an existing account

Step 3: Logout from the session

Step 4: Edit the existing Profile

Step 5: Change Password for security issues

Step 6: Forgot Password and receive the current password over an email

Step 7: Delete an existing Account

Account operations module will be re-using the DAO layer to provide the above functionalities.

C. Web Service Module

The SQL intrusion injection detection process will be implemented as a web service. This module will be used by the admins of SQLiiD admins to get the endpoint of the SQLiiD service. The admins can then use this module for performing email campaigning to various customers. The customers will be receiving an email with the end URL to access the SQLiiD web service along with their passcode.

D. Pay as you go

This module is used by the SQLiiD admins for billing the customers based on their usage on the SQLiiD web service. The SQLiiD webservice, whenever invoked by the customer will be checking for the authorization. If it passes, it executes the core algorithm to analyze the inputted SQL string to determine if it's safe to execute on the customer database or not. The entry will be made on the pay as you go module database indicating that the user has invoked the service.

E. Results and Reports

This module will be used by the SQLiiD admins to view the results and reports of various webservice calls to the SQLiiD by the customers. The results and reports include the client identifier who invoked the service, the timestamp at which the service was invoked, the SQL string against which the algorithm was been invoked, and the detailed report indicating whether it's safe to execute the SQL sting on the customer database.

F. Sample Application

This application is developed only for demonstration purpose to show how the customers will be using our SQLiiD web services to analyze their SQL string for SQL injection detection.

4. IMPLEMENTATION

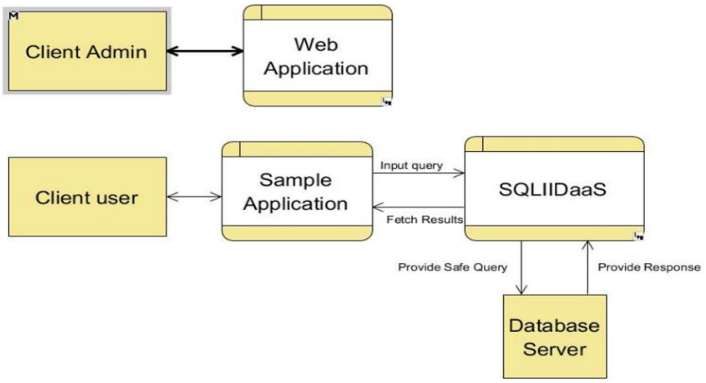

Fig-2: Data Flow of the application

We basically have two applications one being the client-side end user application and the other is the administrator application.

4.1 Implementation of the administrator module

Fig-3: Pay as You Go

Here the client through the Web Application will access all the basic Profile operations. Firstly, Registration and authentication is done to get the data of the client who is going to use the service. Then the admin can invite other clients to use the service. With the help of Pay as You go and

reports the admin can track all the usage statistics of his/her end users.

4.2 Implementation of the client module

Here the invite is received from the admin and we get a portal access to this IDS. So, whatever the input query given in the client application will be reviewed once by this application and then given the result whether the query is safe or not. Only after the query is validated it gives access to the database.

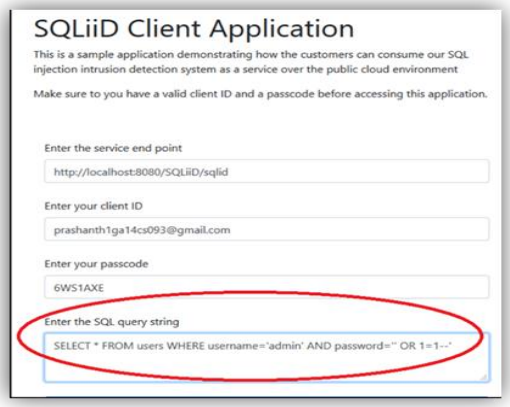

Fig-4: Malicious Query

4.3 PsuedoCode

We basically have three filter mechanisms in the algorithm.

First is the grammar checking phase wherein we check for the syntax correctness of the sql query. It makes sure the query inputted is always starting with parenthesis and quotes. If the query doesn't comply with it, it rejects.

Then we have the Whitelist checking phase. Here the query is matched for syntactic errors in a sql query. Let us consider the example

SELECT DISTINCT City FROM Customers;

Here **"SELECT", "DISTINCT" and "FROM"** are whitelist literals. So, it checks if these literals are present. If it is present, then it passes on to Black List checking phase.

The Black List checking phase is the one which checks for malicious queries present in the query. If there are any uncommon letters used or if there is any special character, it rejects. Let us consider the following example

SELECT * FROM users WHERE username='admin' AND password='' OR 1=1--'

Here the username is accepted but the password is made true "1=1" for all the values. Hence whatever may be the real password, it retrieves all the data.

There is also method wherein the code present inside the database is commented out. Hence the whole code will be altered. So, keeping all the usual hacking techniques we have devised a blacklist which contains all the possible hacking techniques by query mapping.

And for all the checking phase we increment our Pay as You go Module because the customer would have anyhow used the service. The time is also recorder to give the reports for the admin in the later stage.

Black List Checking ()

```
begin
read (SQl Query);
res<-compare (SQLQuery, RE);      /*Check for some
                                    malicious  SQL statement*/
if (res==true) then
PayAsYouGo<- PayAsYouGo+1;
insert (time recorded, SQL syntax, status) in report;
return false;
break;
end if
else if (res==false) then
PayAsYouGo<- PayAsYouGo+1;
insert (time recorded, SQL syntax, status) in report;
return true;
break;
end if
end
```

5. CONCLUSION AND FUTURE WORK

The proposed SQL injection intrusion detection framework allows a SaaS provider to detect sql Injection attack targeting several SaaS applications without reading, analyzing or modifying the source code. A SaaS provider can subscribe to this framework and launch its own set of virtual machines, which holds on-demand self-service, resource pooling, rapid elasticity, and measured service properties

Future work would be to make SQLIIDaaS fully as a service from tenant's point of view by ensuring on-demand self-service and pay-as-you-go characteristics. We could have an option of sending an alert notification to the client whenever this service is being used. A provision could be made to make

the intrusion detection system fully as a service from tenant's point of view by ensuring on-demand self-service and pay-as-you-go characteristics with the help of API's we can have a lesser complexity and more visibility of availing the service

REFERENCES

[1] P. Mell and T. Grance, "The nist definition of cloud computing," National Institute of Standards and Technology.Special Publication 800- 145, pp. 1–7, 2011.

[2] V. Varadharajan and U. Tupakula, "Security as a service model for cloud environment", Network and Service Management, IEEE Transactions on, vol. 11, no. 1, pp. 60–75, 2014.

[3] S. Curtis, "Barclays: 97 percent of data breaches still due to sql injection," http://www.techworld.com/news/security/barclays-97-percentof- data-breaches-still-due-sql-injection-3331283/, 2015.

[4] W. G. J. Halfond and A. Orso, "Preventing SQL injection attacks using AMNESIA," Proceeding of the 28th international conference on Software engineering - ICSE '06, pp. 1–4, 2006.

[5] Yassin M., Ould-Slimane H., Talhi C. & Boucheneb H. (2017, June). "SQLIIDaaS: A SQL injection intrusion detection framework as a service for SaaS providers". In Cyber Security and Cloud Computing (CSCloud), 2017 IEEE 4th International Conference on (pp. 163-170). IEEE.

[6] S. Bandhakavi, P. Bisht, P. Madhusudan, and V. Venkatakrishnan, "Candid: Preventing sql injection attacks using Dynamic candidate evaluations," In Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS), no. October, pp. 12–24, 2007.

[7] C. Gould and P. Devanbu, "JDBC checker: a static analysis tool for SQL/JDBC applications," Proceedings. 26th International Conference on Software Engineering, pp. 697–698, 2004

[8] David Litchfield, (2005) "Data-mining with SQL Injection and Inference", Next Generation Security software Ltd., White Paper.

[9] Zhendong Su and Gary Wassermann, (2006) "The Essence of Command Injection Attacks in Web Applications", Proc. ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages '06, pp.372-382.

[10] Chip Andrews, "SQL Injection FAQs", http://www.sqlsecurity.com/FAQs/SQLInjectionFAQ/tabid/56/Default.aspx

[11] Yonghee Shin and Laurie Williams, (2008) "Toward A Taxonomy of Techniques to Detect Cross-site Scripting and SQL Injection Vulnerabilities", NC state Computer science: Technical report.

[12] Stephen Kost, (2004) "An introduction to SQL injection attacks for Oracle developers", Integrity Corporation, White paper.

[13] Burp suite, http://portswigger.net/burp/

[14] https://www.cisco.com/c/en/us/about/security-center/sql-injection.html

[15] https://en.wikipedia.org/wiki/SQL_injection

[16] https://www.itwire.com/security/81964-french-researcher-finds-flaws-in-site-of-indian-state-owned-telco.html.

BIOGRAPHIES

Prashanth C
B.E. (Computer Science)
Global Academy of Technology
Research Area: Cloud Computing and Web Development

Nithin R
B.E. (Computer Science)
Global Academy of Technology
Research Area: Business Intelligence and Big Data

Prajwal Naresh
B.E. (Computer Science)
Global Academy of Technology
Research Area: Networking

Shobhitha G
B.E. (Computer Science)
Global Academy of Technology
Research Area: Business Analysis