ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 7, Special Issue 6, May 2018

2nd National Conference on "RECENT TRENDS IN COMPUTER SCIENCE & INFORMATION TECHNOLOGY (RTCSIT'18)"

13th-14th March 2018

Organized by

Departments of Computer Science & Information Science Engineering, Sri Krishna Institute of Technology, Bangalore, India

Prevention and Detection of SQL Injection of Attacks's

Nischay T. V[1], HareeshaH.R[2] , Rajesh V[3]
U.G Student, Department of Computer Science and Engineering, Sri Krishna Institute of Technology, Bengaluru, India[1.],

U.G Student, Department of Computer Science and Engineering, Sri Krishna Institute of Technology, Bengaluru, India[2]

Assistant Professor, Department of Computer Science and Engineering, Sri Krishna Institute of Technology, Bengaluru, Karnataka, India[3]

ABSTRACT :Recently, we are attending to the proliferationof Cloud Computing (CC) as the new trending internet-based- Platform. Thanks to the deploying paradigm, Cloud Computing is enabling many services. Software as a Service is one of those cloud-based-services. Indeed, SaaS model allows providers to reduce the cost of maintenance and management by transferring traditional on premise deployment to public Cloud. Clients can subscribe, in self-service, to SaaS services based on a pay-per- use model. However, since user data are outsourced to the Cloud, serious security breaches are rising and could harm the reputation of providers and slow down the subscription of clients. SQL injection attack (SQLIA) is one of the most critical SaaS vulnerabilities that allows attackers to violate the availability, confidentialityandintegrityofuserdata. This paper proposes SQL injection intrusion detection plan as a service for SaaS providers, SQLIIDaaS, which allows a SaaS provider to detect SQLIAs targeting several SaaS applications without reading, analyzing or modifying the source code. To achieve SQL query/HTTP request mapping, we propose an event correlation based on the similarity between literals in SQL queries and parameters in HTTP requests. SQLIIDaaS is combined and certify in Amazon Web Services (AWS). A SaaSprovider 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. SQL injection is a technique that exploits a security vulnerability occurring in the database layer of an application. The attack takes advantage of poor input validation in code and website administration. It allows attackers to obtain unauthorized access to the back-and database to change the intended application generated SQL queries. Researchers have proposed various solutions to address SQL injection problems. However, many of them have limitations and often cannot address all kind of injection problems. What's more, new types of SQL injection attacks have arisen over the years. To better counter these attacks, identifying and understanding existing techniques are very important. In this enquiry we present all SQL injection attack types and also different plans and tools which can detect or prevent these attacks.

KEYWORDS : Software as a Service; HTTP Request; SQL Query; SQL Injection Attack; Amazon Web Services; SQL injection attacks, Web application, prevention, detection.

I. INTRODUCTION

Web applications are widely using nowadays. In these web applications, most of those that are based on money transaction like on-line baking, e-shopping, on-line bill payment, Money transfer, etc. The interaction between the web applications and Database is done with Structured Query Language (SQL) and Scripting Language is used. These queries keep sensitive or personal information of various users. So it is necessary to maintain confidentiality from unauthorized access.

Cloud Computing (CC) refers to an infrastructure as a Service (IaaS), platform as a Service (PaaS), or Software as a Service (SaaS) offered by a provider through the web. Several customers share and use these services to self service, on-demand and pay-per-use characteristics. A SaaS is a service-oriented Web application hosted in a CC environment. The platform (PaaS) of anSaaS application consists of Web and database servers that should be installed on virtual

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 7, Special Issue 6, May 2018

2nd National Conference on "RECENT TRENDS IN COMPUTER SCIENCE & INFORMATION TECHNOLOGY (RTCSIT'18)"

13th-14th March 2018

Organized by

Departments of Computer Science & Information Science Engineering, Sri Krishna Institute of Technology, Bangalore, India

machines (IaaS). A SaaS provider can reduce maintenance and infrastructure-related management costs in traditional on-promise environments as well as start-up costs by gradually appointed compute resources in accordance with customer success. A client or tenant benefits from the SaaS model as a subscription, and not in terms of a license, andpays onlyinrelationtousage.A SaaS application inherits the vulnerabilities of Web applications and the service-oriented architecture (SOA) that have increasingly become victims of SQLIA [4].

About 97% of attacks against data are still due to SQLIAs [5], which can allow an attacker to gain complete control of a database [6]. The SaaS/application (39% of attacks) is the most vulnerable service model,
Followed by the PaaS/platform/OS .SQLIAs are the most common vulnerabilities in the SaaSlayer and are consistently at the top of the OWASP's list of the most critical application security risks [8].A SaaS application inherits the vulnerabilities of Web applications and the service-oriented architecture (SOA) that have increasingly become victims of SQLIA . About 97% of attacks against data are still due to SQLIAs , which can allow an attacker to gain complete control of a database . The SaaS/application (39% of attacks) is the most vulnerable service model, followed by the PaaS/platform/OS .SQLIAs are the most common vulnerabilities in the SaaSlayer and are consistently at the top of the OWASP's list of the most critical application security risks .AWS offers significant protection against traditional net- work security issues such as IP spoofing and port scanning . However, it does not provide security measures for SaaS applications deployed in its infrastructure. Force.com (PaaS provider) gives coding tips to protect against SQLIAs without providing automatic solutions .However, the implementation of IDS for SaaS applications enables SaaS providers to increase their customers by convincing them that the confi-dentiality,integrity,andavailabilityofare well ensured.
An intrusion detection system (IDS) is used to detect careful or illegal activities within an informationprocessing system. Two approaches are used forintrusion detection: misuse and anomaly detection. The misuseapproach is to build signatures of known attacks that mustbe updated regularly by an administrator. Any activity thatmatches a signature is considered as an attack. The deviationapproach is in two phases: the learning phase that modelsthe normal behavior of the entity to protect (e.g., network,system, application, service) and the detection phase thatgenerates an alert when a deviation from the normal behavioris observed.An IDS can be classified according to the sourceof information that is inspected: host (HIDS), network (NIDS).The HIDS is installed on a physical host or virtual machine(VM) to analyze data from packets.
The NIDS is installed onstrategic nodes in a network to analyze the header of packets.The most of proposed SQLIA countermeasures require costly analysis and/or modifications ofthe source code and consequently they are valid for a specificlanguage or environment (e.g., J2EE, PHP). Many proposedSQLIA countermeasures analyze the HTTP and/or SQL traffic. However, all these approachesare not available or evaluated in a public Cloud to protect. theSaaS applications. Recently, many are proposed to allow an IaaS provider to protect the infrastructure IaaS (e.g., VM Monitor) against network attacks (e.g., DoS and DDoS). These Cloud IDS focuson the infrastructure layer without ensuring the security of SaaS applications against SQLIAs, since they do not monitor the input(HTTP)and/oroutput(SQL)oftheseapplications.
In this paper, we propose a framework, SQLIIDaaS, which detects SQL injection targeting SaaS applications deployed in public Clouds (e.g., AWS). SQLIIDaaS provides the HTTP traffic (e.g., HTTP request, Web session), SQL traffic (e.g., SQLquery),andthecorrelationbetweenHTTPrequestsandSQL queries. This information is required for the detection of SQLIAs that target SaaS applications without modifying or analyzing their source code. SQLIIDaaS is integrated in AWS, which allows a SaaS provider to reduce maintenance and infrastructure costs by pooling computingresourcestoserve multiple clients or applications simultaneously (resource pooling). SQLIIDaaS is provided as a set of pre-built Amazon Machine Images (AMI). In this Internet is a widespread information infrastructure. Unaware of the security and privacy, the internet is becoming a repository of information. Information and data is the most important business asset in today's environment and achieving an appropriate level of Information Security. Applications are accessible to a variety of new security threats. One of the most threads to web application is SQL injection attack.
According to Open Web Application Security Project (OWASP) top 10 threats for web application in 2013, injection attacks stands first. For example, financial fraud, online banking, theft of private data, shopping and cyber terrorism. Web applications that are affected to SQL injection may allow an attacker to gain complete access to their essential databases. To implement security guidelines inside or outside the database the access of the sensitive databases should be monitored. Detection or prevention of SQL injection attacks is a topic of active research in the industry and

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 7, Special Issue 6, May 2018

2nd National Conference on "RECENT TRENDS IN COMPUTER SCIENCE & INFORMATION TECHNOLOGY (RTCSIT'18)"

13th-14th March 2018

Organized by

Departments of Computer Science & Information Science Engineering, Sri Krishna Institute of Technology, Bangalore, India

academia. To achieve those purposes, automatic tools and security system have been implemented, but none of them are complete or accurate enough to guarantee an absolute level of security on web application.

What is SQL injection?

Most web applications today use a multi-tier design, usually with three tiers:

presentation tier (front end**).** This is the highest level of the application. This tier displays information related to such services as browsing merchandise, purchasing cart contents. It discuss with other tiers by outputting results to the browser/client tier and all other tiers in the network.

Application tier (Middle tier). This tier implements the software functionality by performing detailed processing, and the data tier (Backend). This tier consists of database servers, keeps data structured and solution to request from the application tiers. Three-tier is a client- server architecture in which the user interface, operations process logic, data storage and access are developed and maintained as independent modules, most often on separate platforms. SQL injection is a type of against 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.

II TYPES OF SQL INJECTION ATTACKS

The SQL injection attacks can be participate using various techniques. Some of them are specified as follows:

Tautologies: The main goal of tautology-based attack is to inject code in conditional statements so that they are always evaluated as true. Using tautologies, the attacker wishes to either bypass user authentication or insert inject-able parameters or extract data from the database. A typical SQL tautology has the form, where the comparison expression uses one or more relational operators to compare operands and generate an alwaystrue condition. Bypassing authentication page and fetching data is the most same example of this kind of attack. In this type of injection, the attacker exploits an inject-able field contained in the WHERE clause of query. He transforms this conditional query into a tautology and hence causes all the rows in the database table aimed by the query to be returned. For example, SELECT * FROM user WHERE id='1' or '1=1'-'AND password='1234'; "or 1=1" is the most commonly known tautology.

Logically incorrect query attacks: The main goal of the Illegal/Logically Incorrect Queries based SQL Attacks is to gather the information about the back end database of the Web Application. When a query is rejected, an error message is returned from the database including useful debugging fact. This error messages help attacker to find vulnerable parameters.

Error message showed: SELECT name FROM Employee WHERE id=2234\'. From the message error we can find out name of table and fields: name; Employee; id. By the gained information attacker can organize more strict attacks. The Illegal/Logically Incorrect Queries based SQL attack is accepted as the basis step for all the other techniques.

Union Query: The main goal of the Union Query is to trick the database to return the results from a table different to the one intended. By this technique, attackers join injected query to the safe query by the word UNION and then can get data about other tables from the application. This technique is mainly used to bypass authentication and extract data. For example the query executed from the server is the following: SELECT Name, Phone FROM Users WHERE Id=$id. By injecting the following Id value: $id =1 UNION ALL SELECT credit Card Number, 1 FROM Credit sys Table. We will have the following query: SELECT Name, Phone FROM Users WHERE Id=1 UNIONALLSELECT credit card Number, 1 FROM Credit sys Table. This will join the result of the original query with all the credit card users.

Stored Procedures: The main goal of the Stored Procedures SQL attack is to perform privilege escalation and try to execute the SQL procedures. SQL injection attacks of this type try to execute the SQL procedures. Stored procedure is a part of database that programmer could set an extra abstraction layer on the database. As stored procedure could be coded by coder, so this part is as inject-able as web application forms. Depend on specific stored procedure on the database there are distinct ways to attack. In the following example [4], attacker exploits parameterized stored procedure. CREATE PROCEDURE DB. is Authenticated @username Name varchar2, @pass varchar2, @pin int ASEXEC ("SELECT accounts FROM users WHERE login='" +@user Name+ "' and pass='"+@password+"'and pin="+@pin); GO For authorized/unauthorized user the stored procedure returns true/false. As an SQL injection attack, intruder input "'; SHUTDOWN; - -"for username or password. Then the stored procedure generates the following query: SELECT accounts FROM users WHERElogin='boni'ANDpass='; SHUTDOWN; -- AND pin= . After that, this type of attack works as piggy-back attack. The first unique query is executed and importance the second

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 7, Special Issue 6, May 2018

2nd National Conference on "RECENT TRENDS IN COMPUTER SCIENCE & INFORMATION TECHNOLOGY (RTCSIT'18)"

13th-14th March 2018

Organized by

Departments of Computer Science & Information Science Engineering, Sri Krishna Institute of Technology, Bangalore, India

query which is illegitimate is executed and causes database shut down. So, it is considerable that stored procedures are as vulnerable as web application code

Piggy-Backed Queries: The main goal of the Piggy- Backed Query is to execute remote commands or add or modify data. In this attack type, an attacker tries to inject additional queries along with the original query, which are said to "piggy-back" onto the real query. As a result, the database accepts multiple SQL queries for execution. Vulnerability of this kind ofattack is dependent of the kind of database [5]. For example, if the attacker inputs [';drop table users--] into the password field, the application generates the query: SELECT Login_IDFROM users_IDWHERE login_ID='john' and password='''; DROP TABLE users-' AND ID=2345 After executing the first query, the database encounters the query delimiters (;) and execute the second query. The result of executing second query would result into dropping the table users, which would likely destroy valuable information.

Inference: The main goal of the inference is to modify the behavior of a database or application. There are two well-known attack techniques that are based on inference: blind injection and timing attacks.

Blind Injection: Sometimes developers hide the mistakes details which help attackers to compromise the database. In this situation attacker face to a generic pages provided by developer, instead of an error message. So the SQLIA would be more difficult but not impossible. An attacker can still steal data by asking a series of True/False questions through SQL statements. Consider two possible injections into the login field: For example, SELECT accounts FROM users WHERE id= '1111' and 1 =0 -- AND pass = AND pin=0 SELECT accounts FROM users WHERE login= 'doe' and 1 = 1 -- AND pass = AND pin=0 If the application is protected, both queries would be unsuccessful, because of input validation. But if there is no input validation, the attacker can try the chance. First the attacker submits the first query and receives an error information because of "1=0 ". So the intruder does not understand the error is for input validation or for logical error in query. Then the attacker submits the second query which always true. If there is no login error message, then the attacker finds the login field vulnerable to injection.

Timing Attacks: A timing attack lets an attacker gather information from a database by observing timing delays in the database's responses. This technique by using if-then statement cause the SQL engine to execute a long running query or a time delay statement depending on the logic injected. This attack is same to blind injection and attacker can then measure the time the page takes to load to cause if the injected statement is true. This technique uses an if-then statement for injecting queries. WAITFOR is a keyword along the branches, which causes the database to delay its response by a specified time.

III. MAIN CAUSE OF SQL INJECTION

In this section various cause of SQL injection presented those are:

1. Invalidated Input: Any SQL query consist some parameters such as INSERT, UPATE, ALTER and some SQL control character such as semicolon and quotation mark. If there is no checking for web applications so it can be abused in SQL injection.
2. Generous Privileges: Privileges are some rules for accessing some database for some object and by object which actions going to be perform. SELECT, INSERT, DELETE are actions of executing SQL queries that included typical privileges. web application is use for accessing any specific information from database. By bypass authentication an attacker gain privileges.
3. Uncontrollable variable size: If any variable is using storage for large amount of data so some time can be possible that attacker may enter faked input values. 2.4 Error Message: Error message generates when wrong input values is inserted in web application. Attacker may get the script structure or information about database .so that attacker may create its own attack.
4. Clint side only control: If input validation is implemented in client side scripts only, then by using cross site scripting security function of script at client side can be override and attacker can invalidate input or accessing database.
5. Stored Procedure: Stored Procedure is small program with some function that calls multiple times in execution. When these functions become calls so that stored procedure become calls in place of that functions. These stored procedures become storedin database. The problem with the stored procedure is that an attacker can execute and damage database.
6. Into Outfile support: A text file of containing SQL query result may get by manipulating SQL query. This can be possible by using condition of INTO OUTFILE clause that is benefit of Some Relational Database

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 7, Special Issue 6, May 2018

2nd National Conference on "RECENT TRENDS IN COMPUTER SCIENCE & INFORMATION TECHNOLOGY (RTCSIT'18)"

13th-14th March 2018

Organized by

Departments of Computer Science & Information Science Engineering, Sri Krishna Institute of Technology, Bangalore, India

IV. SQL ATTACK INJECTION PREVENTION TOOLS

1. **JDBC Checker** [11] - It is uses for dynamically generated query string on basis of mismatching. As we know that most of the SQLIAs consist of syntactically and type correct queries so this technique would not catch more general forms of attacks.
2. **WAVES** [6] - WAVES a black-box technique that uses a Web crawler to identify all points in a Web application that can be used to inject SQLIAs. It target on a specified list of patterns and attack techniques. WAVES then monitors the application's response to the attacks and uses machine learning techniques to improve its attack methodology.
3. **SECURITYFly** [2] is tool that is implemented for java. Here check string in place of character for any suspicious information and try to sanitize query strings. This tool has a drawback that is numeric fields cannot stop by this approach. Difficulty of identifying all sources of user input is the main limitation of this approach.
4. **SECURITY GATWAY** [2] - It works on the filtering system that forces the input validation. By using Security Policy Descriptor Language (SPDL), developers provided specify transformation that is applied to the parameters of web application.
5. **SQL DOM** [6]- It is an object model for proposing a solution for building a secure communication environment for accessing relational databases from the OOP (Object-Oriented Programming) Languages because they mainly focus on identifying the obstacles in the interaction with the database via Call Level Interfaces.
6. **WebSSARI** [32]– A WebSSARI tool use for sanitizing input that passed through predefined set of filters. Here static analysis to check taint flows against preconditions for sensitive functions. The drawback of this approach is that it is not necessary preconditions for sensitive function accurately expressed. WebSSARI's system architecture is presented.

A code walker consists of a lexer, a parser, an AST (abstract syntax tree) maker, and a program abstractor. The program abstractor asks the AST maker to generate a full representation of a PHP program's AST. The AST maker uses the lexer and the parser to perform this task, handling external file inclusions along the way. By traversing the AST, the program abstractor generates a control flow graph and a symbol table. Based on the prelude files, the verification engine moves through the Control Flow Graph and references the ST to generates type qualifiers for variables and preconditions and post conditions for functions.
This routine is repeated until no new information is generated. The verification engine then moves through the control flow graph once again, this time performing type state tracking to determine insecure information flow. It outputs insecure statements (with line numbers and the invalid arguments). For each variable involved in an insecure statement, it inserts a statement that secures the variable by treating it with a sanitization routine. The insertion is made right after the statement that caused the variable to become tainted. Sanitization routines are stored in a prelude, and users can supply the prelude with their own routines. The program abstractor asks the AST maker to generate a full representation of a PHP program's AST.
The AST maker uses the lexer and the parser to perform this task, handling external file inclusions along the way. By traversing the AST, the program abstractor generates a control flow graph and a symbol table. Based on the prelude files, the verification engine moves through the Control Flow Graph and references the ST to generates type qualifiers for variables and preconditions and post conditions for functions.

V. CONCLUSION

Injection attack is major headache for developers for securing web applications. Various tools have been developed for detection and prevention. In these most of the tools can attack all types of SQLIAs except Stored procedure. In this paper a survey of various detection and prevention techniques of SQL Injection Attack is enlist. Started from Tautology checker that on only check tautology attack then CANDID [26] that can detect more attacks then CSSE [29] by using string evaluation mechanism it made possible to detect all type of attack except Stored Procedure, then AMNEISA [25] which is based on combination of static and runtime monitoring. Other newly method of detection by

ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 7, Special Issue 6, May 2018

2nd National Conference on "RECENT TRENDS IN COMPUTER SCIENCE & INFORMATION TECHNOLOGY (RTCSIT'18)"

13th-14th March 2018

Organized by

Departments of Computer Science & Information Science Engineering, Sri Krishna Institute of Technology, Bangalore, India

comparing static MAC value and Dynamic MAC value is done in detection block model [1]. So this is broad area where new tools can be developed for detection as well as prevention of SQL Injection Attacks.

REFERENCES

[1] Diksha G. Kumar, MadhumitaChatterjee "Detection Block Model for SQL Injection Attacks" I.J. computer Network and Information Security, 2014

[2] BojkenShehu, AleksanderXhuvani "A literature Review and comparaative analysis on SQL injection: Vulnerabiities, attacks and their detection and prevention Techniques" International Journal of Computer Science Issues, Vol 11,Issue 4, no1 2014

[3] GeogianaBuja, Dr. Kamarularifin Bin AbdJalil, Dr. Fakariah Bt. HjMohd Ali, The Faradilla Abdul "Detection model for SQL Injection Attack: An approach for preventing a web application from the SQL injection Attack"IEEE Symposium on Computer Applications and Industrial Electronics, April 2014

[4] NunoSeixas, Marco Vieira, Jose Fonseca, Henrique Madeira "Analysis of field data on web security vulnerabilities "IEEE Transactions on Dependable and secure computing Vol. 11 No.2 March/Aril 2014

[5] HossaianShahriar, Mohammad Zulkernine, "Information Theoretic Detection of SQL Injection Attacks" International Symposium on high-Assurance systems Engineering, IEEE 2014 [6] Hussein AlNabulsi, IzzatAlsmadi,, Mohammad AlJarrah "Textual Manipulation for SQL Injection attack" I.J. computer Network and Information Security, 2014

[7] Monali R. Boradel, Neeta A. Despande "Extensive Review of SQLIA's Detection and Prevention Techniques" International Journal of Emerging Technology and Advanced Engineering ISSN 2250- 2459, ISO 9001:2008 Certified Journal, Volume 3, Issue 10, October 2013

[8] Shelly Rohilla, Pradeep Kumar Mittal "Database Security by Preventing SQL Injection Attacks in Stored Procedure" Journal of Advanced Research in Computer Science and software Engineering Volume 3, Issue 11 November 2013.

[9] JaskanwalMinhas Raman Kumar "Blocking of SQL Injection attack by Comparing Static and Dynamic queries" International Journal of computer network and Information Security 2013

[10] Mihir Gandhi, JwalantBaria "SQL INJECTION Attacks in Web application"International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-6, January 2013"