Application for Detecting and Preventing SQL Injection Attacks using Web Service

Anuja A. Patil[1], Ketaki H. Pangu[2]

[1,2]Department of Computer Science, Bharati Vidyapeeth College of Engineering, Kolhapur, Maharashtra, India
anujapatil294@gmail.com
ketakipangu04@gmail.com

Abstract: ***Every cyber attack mostly targets the Databases through the firewalls that shield it. Like that SQL injection attacks can target the databases. This type of attacks occurs due to poor input validation code, poor website administration as well as it takes advantage of less security to the databases. In this case the attacker can take advantage of web services applications and can pass a series of unwanted, malicious SQL statements as input and try to execute the back end of the system. So, the proposed model gives methodology to detect and prevent the SQLIAs in databases using XPATH validation systems which is most efficient way.***

Keywords: Database security, web application security, SQL injection attacks, Web Service

1. Introduction

Database security takes the highest priority so, in today's life the information is important and that is also residing in databases. So database security is a big deal for today's business assets and for that SQLIAs are most vulnerable threats of database security through web services. e.g. financial frauds, confidential information hacking, sabotage, cyber terrorism etc. The SQLIAs is the hacking technique where attacker can input the number of malicious SQL statements from which it can executes the database as well as the confidential information in it. So, now a days the SQLIAs are the topmost threats in web security point of view. Compromises of these applications represent a status threat to organizations. In this attacker can direct attack on a databases of the system and leak the confidential information residing in it. The increasing of such a web applications can occurs an increments of such attacks. It is very difficult to implement and enforce a rigorous defensive coding discipline. So, detect and prevent the SQL attacks by using XPATH validation[1] can helps a lot.

2. Techniques of SQLIA'S

Most of the attacks can takes place in sequential manner, depending of the attackers.

2.1. Tautologies

The tautology based attack is to inject code in one or more conditional statements so that they always evaluate the true.
e.g. Attacker can submits the query like " 'or 1==1--"
so it can like as SELECT * FROM user_info WHERE loginID=" or 1=1-- AND password="
The code injected in the conditional ('OR 1=1) transforms

the entire WHERE condition into tautology query to execute each row and column in table and returns all of them because the query after "- -" is considered as comment into the SQL.

2.2. Union

In union query an attacker can attack by injecting a statement in the form of UNION SELECT <rest of injected query> because an attacker are completely control on the injected query i.e. second part of the query. The result of this attack is that the database returns a dataset that is the union of the results of the original first query and the result of the second injected query.
e.g.

"select * from great where pin=222 union Select * from great1 where 1=1".

2.3. Stored Procedure

Once an attacker determines which backend database is in use, SQLIAs can be crafted to execute stored procedures provided by that specific database, including procedures that interact with the operating system. It is a common misconception that using stored procedures to write Web applications renders them invulnerable to SQLIAs.
e.g.

```
'create or alter procedure test'
 @uid varchar(30)
 as
 begin
 select pwd from login where unm=''
 union select pwd from login
 where unm=@uid-- and  pwd=''
 end
 To run above query we use following syntax:
 exec test 'unm' .
```

2.4. Extended Stored Procedures

There is several extended stored procedures that can cause permanent damage to a system. This Attack is used to stop the service of the web server of particular Web application. Stored procedures primarily consist of SQL commands, while XPs can provide entirely new functions via their code.

Extended stored procedure can be executed by using login form with an injected command as the,
LoginId:'; execmaster..xp_xxx; -- Password:[Anything]
LoginId:'; execmaster.xp_cmdshell'iisreset';--
Password:[Anything]
select password from user_info where LoginId='';
exec master..xp_cmdshell 'iisreset'; --' and Password=''

2.5. Alternate Encoding

In this attack, the injected text is modified so as to avoid detection by defensive coding practices and also many automated prevention techniques. This attack type is used in conjunction with other attacks. In other words, alternate encodings do not provide any unique way to attack an application; they are simply an enabling technique that allows attackers to evade detection and prevention techniques and exploit vulnerabilities that might not otherwise be exploitable.
e.g.
SELECT * FROM user_info WHERE loginID='secret';
exec (char (0x73687574646f776e)) -- AND pass1=''

3. Related Work

3.1 Intrusion detection system

An intrusion detection system (IDS) is a device or software application that monitors network or system activities for malicious activities or policy violations and produces reports to a management station. Some systems may attempt to stop an intrusion attempt but this is neither required nor expected of a monitoring system. Intrusion detection and prevention systems (IDPS) are primarily focused on identifying possible incidents, logging information about them, and reporting attempts. In addition, organizations use IDPSes for other purposes, such as identifying problems with security policies, documenting existing threats and deterring individuals from violating security policies. IDPSes have become a necessary addition to the security infrastructure of nearly every organization. The technique builds models of the typical queries and then monitors the application at runtime to identify queries that do not match the model. In their evaluation, Velour and colleagues have shown that their system is able to detect attacks with a high rate of success.

3.2 Combined static and dynamic analysis

AMNESIA[2] is a model-based technique that combines static analysis and runtime monitoring. In its static phase, AMNESIA uses static analysis to build models of the different types of queries an application can legally generate at each point of access to the database. In its dynamic phase, AMNESIA intercepts all queries before they are sent to the database and checks each query against the statically built models. Queries that violate the model are identified as SQLIAs and prevented from executing on the database. In their evaluation, the authors have shown that this technique performs well against SQLIAs. The primary limitation of this technique is that its success is dependent on the accuracy of its static analysis for building query models. Certain types of code obfuscation or query development techniques could make this step less precise and result in both false positives and false negatives.

4. Proposed Techniques

This Technique is used to detect and prevent SQLIA's with runtime monitoring [3]. The solution insights behind the technique are that for each application, when the login page is redirected to our checking page, it was to detect and prevent SQL Injection attacks without stopping legitimate accesses.

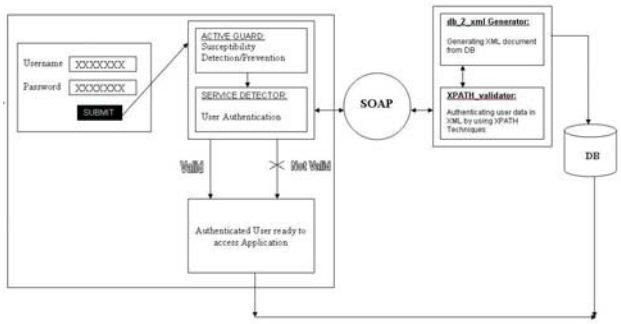

Figure 1. Proposed Architecture

The technique consists of three models namely:

4.1. Active Guard Filtration Model

In this module, susceptibility characters or meta characters are detected to prevent the malicious attacks from accessing the database.
Following patterns are detected by it:

4.1.1. Tautology

a. 'or 1=1- -
b. 'or 1=1- -
c .'or 1=1 or 'a'
d. 'or 1=1/*

4.1.2. Union Query

a. Union select * from tablename where columnname=value - - -

4.1.3. Delete Query

a. a ',delete from tablename - -

4.1.4. Drop Query

a. a 'drop table tablename - -
b. a' drop database databasename - -

4.2. Service Detector Filtration Model

It validates user input from X_path validator where the sensitive data's are stored from the database at the second level filtration model.

4.3. Web Service Layer

Web service builds two types of execution process that are DB_2_Xml generator and XPATH_ Validator. DB_2_Xml generator is used to create a separate temporary storage of Xml document from database where the Sensitive data's are stored in XPATH_ Validator, The user input field from the Service Detector compare with the data existed in XPATH_ Validator, if the data's aresimilar XPATH_ Validator sends a flag with the count iterator value = 1 to the Service Detector by signifying the user data is valid.

5. Evaluation

Both the protected and unprotected web Applications are tested using different types of SQLIA's; namely use ofTautologies, Union, Piggy-Backed Queries, Inserting additional SQL statements, Second-order SQL injection and various other SQLIA s. Table 1 shows that the proposed technique prevented all types of SQLIA s in allcases.

Table 1: SQLIA'S Prevention Accuracy

SQL Injection Types	*Unprotected*	*Protected*
1.tautologies	Not Protected	Prevented
2.union	Not Protected	Prevented
3.stored procedure	Not Protected	Prevented
4.alternate encoding	Not Protected	Prevented

6. Conclusion

SQL Injection Attacks attempts to change the web application parameters to alter the SQL database. There are some procedures that they prevent these attacks and increase the strength of coding. This is one of the good attempts to prevent the SQL Injection Attacks.

7. Acknowledgement

I express my deep sense of gratitude and appreciation towards my research guide Prof. A. M. Patravale for his continuous inspiration and valuable guidance in throughout my dissertation work.

8. References

[1] Indrani Balasundaram and Dr. E. Ramaraj , "An Approach to Detect and prevent SQL Injection Attacks using Web Service".
[2] William G . J . Hal fond and Alessandro Orso , "AMNESIA: Analysis and Monitoring for Neutralizing SQL Injection Attacks".
[3] W. G. J. Halfond and A. Orso, "Combining Static Analysis and Runtime Monitoring to Counter SQL Injection Attacks".

Author Profile

Anuja Patil, the final year student of the Bharati Vidyapeeth's College of Engineering, Kolhapur, Maharashtra, India; persuing Computer Science and Engineering degree(2013).

Ketaki Pangu, the final year student of the Bharati Vidyapeeth's College of Engineering, Kolhapur, Maharashtra, India; persuing Computer Science and Engineering degree(2013).