INJECTION ATTACK STOPOVER OF SECURE DATA FOR WEB SERVICES

Er. Himani[1]
M.Tech Scholar

Er. Parag Rastogi[2]
Assistant Professor

Department of Computer Science & Engineering
Swami Vivekananda Subharti University, Meerut, UP ,INDIA

ABSTRACT

Now days Data prevention is most important part for an increasing amount of activities, such as Social media, banking and shopping. When user performing such activities, we entrust our personal information to these web applications and their underlying databases. Web Technology related application are often vulnerable to attacks on server or application, which can give an attacker complete right or access to the applications' underlying database. SQL Injection is an attack in which malicious code is inserted into strings that are later passed to a database for parsing and execution. When hacker SQL Injection Attack, then attacker attempts to exploit vulnerabilities in custom web applications by entering SQL code in an entry field such as a login. If successful, such an attack can give the attacker access to the data on the database used by the application and the ability to run malicious code on the Web site. Several methods have been proposed to detect and prevent SQL injection attacks. We devise a method that uses defensive coding and secure hash algorithm to prevent SQL injection attacks. This technique is illustrated by overview, diagrams and step by step procedure for implementing the technique to protect web application against SQL Injection. We show that this technique can be used effectively to prevent SQL Injection Attacks through bypass authentication in web application without degrading the system's performance. Finally, the method is implemented by using a web application and database with Apache web server, PHP script & My SQL.

Key words: Social Media, SQL Injection, Hash, Database, Web applications.

1. INTRODUCTION

These research work highlight on the analysis and resolution of the problem of SQL Injection Attacks on server, in order to protect and make reliable any vulnerable web applications. Firstly, we deeply analyzed the different SQL Injection Attack techniques and then we tried to provide a prevention technique which is implemented during development of the web applications. This technique does not let the system degrade its performance but there is a very small time increase in time while logging in to the user's account. To achieve our goal, we propose a general methodology which can be easily deployed while developing a new system.

Data prevention is a branch of technology known as information security, applied to computers. Information security is based on the general concept of protection of data against unauthorized access. The objective of computer security varies and can include protection of information from theft or corruption, or the preservation of availability, as defined in the security policy. Information/Data security is the process of preventing and detecting unauthorized use of your Device. Prevention measures help you prevent unauthorized users known as intruders, from accessing any part of your computer system. Detection helps you to determine whether or not someone attempted to break into your system, whether or not the attempt was successful, and the extent of the damage that may have been done. This makes computer security particularly challenging because it is difficult enough just to ensure that computer programs do everything that they are designed to do correctly. Nowadays most information in the world is processed through computer security. This is quite a common mistake: in fact, academically, the definition of information security includes all the processes of handling and storing information. Information/Data may be printed on paper, collect in devices, stored electronically, transmitted by post or by using electronic means, shown on films, or spoken in conversation.

2. RESEARCH FOCUS AND CONTRIBUTIONS

Nowadays, web applications are becoming increasingly popular and are poised to become a major player in the overall software market due to benefits they afford, such as visibility and worldwide access. They are, without a doubt, essential to the current and next generation of businesses and they have become part of our everyday online lives. In fact, a web application is a world wide gate accessible not only through standard personal computers but also through different communication devices such as mobile phones and PDAs.
The use of web applications is especially beneficial for a company: with just a little investment, a company can open up a marketing channel that will allow potential clients easy global to its business 24 hours a day. A typical example of a web application is an online questionnaire or user survey. The end-user client simply completes the online questions by filling in a form that is accessible worldwide through any kind of network device and submits the responses to the application than then collects and stores the data in a database on the server side.

Web applications are present in all aspects of our daily Internet use. Common examples are those applications used for searching the Internet such as "Google"; for collaborative open source projects as "SourceForge"; for public auctions such as "eBay" and many others as well as blogs, webmail, web-forums, shopping carts, e-commerce, dynamic contents, discussion boards and social networks.

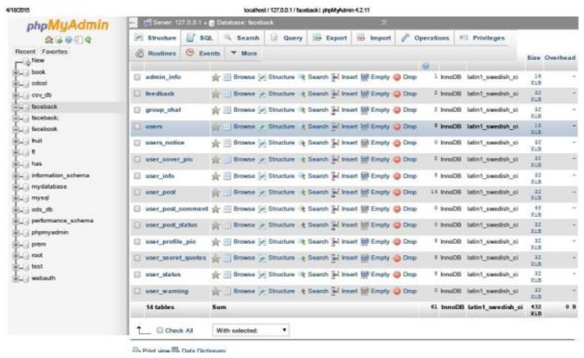
3. PROPOSED TECHNIQUE

Our research work proposes the technique, SOL Injection Attack Prevention for Web Application (SIAPWA). In the papers [1][3] encryption methods have been used in conjunction with adding some extra columns in a Login table to avoid SQL injection attacks. These techniques require much more extra space, when we have a large number of users in a Login table and also introduces a significant amount of delay during comparison of credentials. However, our proposed technique requires no extra columns in a login table, due to which no extra space will be required and comparison delay is also negligible. In this technique the credentials provided by the user are directly stored into a Login table in encrypted form.For the encryption of credentials (username, password) we use SHA2(secure hash algorithm 2). SHA2 is more stronger than SHA1,the latter has been already broken by hackers. The hash values of username and password are calculated and stored in Login Table when the user's account is first time created with the web application. Whenever user wants to login to database his/her identity is checked by comparing username and password entered by the user with the already stored values in the Login table. Whenever comparison returns to be true, the user able to access the database
otherwise database access denied. These hash values are calculated at runtime using stored procedure when user wants to login into the database.

If only username and password are used for authentication, and the attacker enters Username = ' OR 1=1 – – and Password = pwd;

The query becomes like this:

SELECT* from Users where username='OR 1=1=1' and password=pwd;



4/18/2015 localhost / 127.0.0.1 / faceback | phpMyAdmin 4.2.11

Server: 127.0.0.1 » Database: faceback

Table	Action	Rows	Type	Collation	Size	Overhead
admin_info	Browse Structure Search Insert Empty Drop	1	InnoDB	latin1_swedish_ci	16 KiB	-
feedback	Browse Structure Search Insert Empty Drop	2	InnoDB	latin1_swedish_ci	32 KiB	-
group_chat	Browse Structure Search Insert Empty Drop	2	InnoDB	latin1_swedish_ci	32 KiB	-
users	Browse Structure Search Insert Empty Drop	8	InnoDB	latin1_swedish_ci	16 KiB	-
users_notice	Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	32 KiB	-
user_cover_pic	Browse Structure Search Insert Empty Drop	2	InnoDB	latin1_swedish_ci	32 KiB	-
user_info	Browse Structure Search Insert Empty Drop	8	InnoDB	latin1_swedish_ci	32 KiB	-
user_post	Browse Structure Search Insert Empty Drop	14	InnoDB	latin1_swedish_ci	32 KiB	-
user_post_comment	Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	48 KiB	-
user_post_status	Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	32 KiB	-
user_profile_pic	Browse Structure Search Insert Empty Drop	8	InnoDB	latin1_swedish_ci	32 KiB	-
user_secret_quotes	Browse Structure Search Insert Empty Drop	8	InnoDB	latin1_swedish_ci	32 KiB	-
user_status	Browse Structure Search Insert Empty Drop	8	InnoDB	latin1_swedish_ci	32 KiB	-
user_warning	Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	32 KiB	-
14 tables	Sum	61	InnoDB	latin1_swedish_ci	432 KiB	0 B

Figure 5.1 Query without using hash values

There the user will be to bypass authentication. Whereas using PSIAW approach, the query for authentication will become like this:

SELECT* from Users where username=hashvalue of('OR 1=1=1') and password=hashvalueof(pwd);

Select*from Login where Hash_value_username='hash_valueof(OR 1=1--)'

Figure 5.2 Query using hash values

Thus using encrypted values for password and username, the hacker cannot bypass authentication as attacker does not know the hash values of username and password and hence can't access the database of the web application. Thus, web application is secured. The error messages generated by application should not show that any hash values are calculated at the back end and it's getting matched with the entered one. This prevents the attacker from accessing database as he is not aware of any hash values used and does not know the hash values of username and password as hash values are calculated at runtime. Only two text boxes are provided at the interface for entering username and password, he will not be able to enter hash values from anywhere. Hence, the attacker will not be able to attack database and web application is secured.

When user changes password, encrypted value of old password supplied as well as new password is calculated. Encrypted value of old password must match with the stored encrypted value and new value is stored with the new password in the Login Table.

Every time database is accessed, encrypted values of supplied parameters are calculated and matched with the stored one. Whenever it does not match it simple generates the message, username and password do not match. So the attacker does not get to know about the encrypted values concept.

4. METHODOLOGY

A social networking website named 3D Facebook has been developed for evaluating proposed technique. The website is developed using PHP 5.6.0 and MySQL 5.6.12 as DBMS. Total 14 tables are created including 'users' and 'admin_info'.

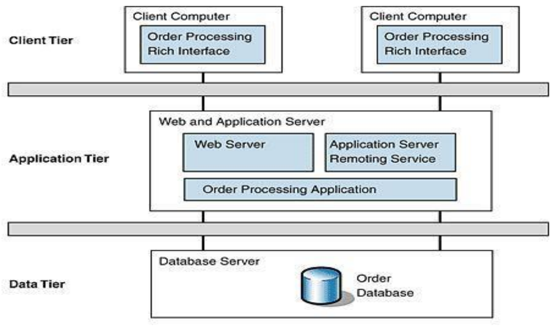

Client Tier
Client Computer
Order Processing Rich Interface
Client Computer
Order Processing Rich Interface
Application Tier
Web and Application Server
Web Server
Application Server Remoting Service
Order Processing Application
Data Tier
Database Server
Order Database

Figure Design of users Table

Data is added to these tables. Two stored procedures named Registration and Log user have been created. Registration stored procedure calculate hash values of username and password at the registration time and stores all entries of the user along with hash values in Registration Table table. Log user separately stores the username, password, hash values of username and password into the Login table.

```
mysql> CREATE
    PROCEDURE
    registration
    (OUT param1 INT)
    -> BEGIN
    ->
    SELECT
    COUNT(*
    ) INTO
    param1
    FROM t;
    ->
    END//

Query OK, 0 rows affected (0.00
sec)

mysql> delimiter ;

mysql> CALL simpleproc(@a);

Query OK, 0 rows affected (0.00
sec)
```

```
create procedure
[dbo].[Registration]
(@fn nchar(10), @ln
nchar(10),
@uid nvarchar(50),@pwd
nvarchar(10),
@eid nchar(30),@desi
nchar(30),
@exp_yr int, @exp_mn int,
@jdate date,@ms
nchar(10),
@cn nchar(10),@qual
nvarchar(50)) as
insert into
RegistrationTable values
(@fn,@ln,@uid,@pwd,@eid,@de
si,@exp_yr,
@exp_mn,@jdate,@ms,@cn,@qua
l )
```

5. ARCHITECTURE

Architecture for SQL Injection Attack prevention in Web Application (SIAPWA) technique, consists of three components: User Login Interface, SQL Query Component and User Account Table.

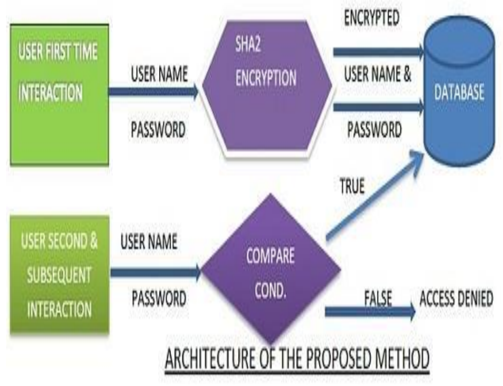


Fig: Web Application Architecture for proposed technique

. Here, user login interface is just the user entry form containing two columns for username and password. Main component of SIAPW is SQL Query Component. SQL Query component is the component where hash value of username and password is calculated. These values are then combined with username and password using AND operator. Every time the user enters username and password, their hash values are calculated. The query formed is then sent to database. Subcomponents of SQL Query component are username hash value, username and password and password hash value

User account table is the component where username, password, hash values for user and passwords are stored here.

6. ADVANTAGES

This technique has been tested with all popular know SQL Injection Strings. This technique was fully capable to protect authentication against SQL Injection. Attacker is not able to bypass authentication, so this technique is very helpful. This technique needs to search two additional columns (user hash value and password hash value) other than username and password. It takes about 2ms more time in logging in than the website which is developed without using hash values. This 2ms extra time overhead is acceptable as it is protecting against hacking. Thus performance is not degraded. So, this technique protects against SQL Injection without degrading system's performance.

time in logging in than the website which is developed without using hash values. This 2ms extra time overhead is acceptable as it is protecting against hacking. Thus performance is not degraded. So, this technique protects against SQL Injection without degrading system's performance.

this technique on the existing website.

FUTURE WORK

This Research motive is security awareness to people. SQL Injection is a common technique hacker employ to attack these web-based applications. These attacks reshape the SQL queries, thus altering the behavior of the program for the benefit of the hacker. In our research work, we have presented a technique for protecting authentication against SQL Injection. This technique can only protect authentication mechanism. Rest of the SQL Injection techniques can't be prevented using this technique. So, in future, we will try to improve the technique by making it efficient for other types of SQL Injection Attacks also. Then, this technique will be able to prevent SQL Injection Attack completely.

7. LIMITATIONS

When a website is developed, database is first created. Then tables are created. In this technique, two columns huser and hpass are added in addition to username and password in the users table. After that stored procedure are created. Then, the website is developed. This means that any website, that is already developed, cannot implement this technique. If any website that is already developed needs to implement this technique, then it has to be reengineered.

REFERENCES

[1] Ravindra Kumar, Neha singh, "SQL INJECTIONS – A HAZARD TO WEB APPLICATIONS" International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 6, June 2012.

[2] Sruthi Bandhakavi, Prithvi Bisht, P. Madhusudan, CANDID: Preventing SQL Injection Attacks using Dynamic Candidate Evaluation. Proceedings of the 14th ACM conference on Computer and communications security. ACM, Alexandria,Virginia, USA.page:12-24.

[3] Ms. Zeinab Raveshi, Mrs. Sonali R. Idate, Efficient Method to Secure Web applications and Databases against SQL Injection Attacks, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 5, May 2013.

[4] P. Bisht, P. Madhusudan. CANDID: Dynamic Candidate Evaluations for Automatic Prevention

of SQL Injection Attacks. ACM Transactions on Information and System Security Volume: 13, Issue: 2, 2010.

[5] S. W. Boyd and A. D. Keromytis. SQLrand: Preventing SQL Injection Attacks. In Proceedings of the 2nd Applied Cryptography and Network Security (ACNS) Conference, pages 292–302, June 2004.

[6] G. T. Buehrer, B. W. Weide, and P. A. G. Sivilotti. Using Parse Tree Validation to Prevent SQL Injection Attacks. In International Workshop on Software Engineering and Middleware (SEM), 2005.

[7] W. R. Cook and S. Rai. Safe Query Objects: Statically Typed Objects as Remotely Executable Queries. In Proceedings of the 27th International Conference on Software Engineering (ICSE 2005), 2005.

[8] Mayank Namdev , Fehreen Hasan, Gaurav Shrivastav, Review of SQL Injection Attack and Proposed Method for Detection and Prevention of SQLIA, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 7, July 2012.

[9] Sonam Panda, Ramani , “Protection of Web Application against Sql Injection Attacks”, International Journal of Modern Engineering Research (IJMER) Vol.3, Issue.1, Jan-Feb. 2013 pp-166-168 ISSN: 2249-6645.