JICLT
Journal of International Commercial Law and Technology
Vol. 6, Issue 4 (2011)

SQL Injection-Database Attack Revolution and Prevention

Ramakanth Dorai
Computer science and engineering
S.N.S College Of Technology-Coimbatore
Playhard89@gmail.com

Vinod Kannan
Computer science and engineering
S.N.S College Of Technology-Coimbatore
fast2fortune@gmail.com

Abstract. SQL injection came with a bang and caused revolution in database attacking. In recent years, with the explosion in web-based commerce and information systems, databases have been drawing ever closer to the network and it is critical part of network security. This paper is incorporated with our research and firsthand experience in hacking the database by SQL injection. Database is the Storage Brain of a website. A hacked database is the source for Passwords and juicy information like credit card number, bank account number and every important thing that are forbidden. Importance should be given for preventing database exploitation by SQL injection. The aim of this paper is to create awareness among web developers or database administrators about the urgent need for database security. Our ultimate objective is to totally eradicate the whole concept of SQL injection and to avoid this technique becoming a plaything in hands of exploiters.

1. Introduction

Exploiting the security vulnerability found in the application's database layer and penetrating into the database using SQL codes is known as SQL injection. This code injection technique fools the database application and gains illegal access. The effect of a successful SQL injection is dreadful. Nearly 1 lakh websites were hacked in 2008 alone. Most of the sites that were compromised were government websites of many nations. We have found that nearly 65 percentage government websites of developing countries are vulnerable. This clearly depicts the need for avoiding SQL injection. Web developers and database administrators should know the devastating consequence of this attack and should consider implementing our suggested prevention methods. This paper starts with the different present forms of SQL injection by hackers and demonstrates with example how they carry out the attack. Ignorance of administrators on current protection systems have brought us to this situation. We have given our suggested prevention method in accordance with current scenario after extensive references and discussion with security experts and underground hackers.

2. Forms of SQL Injection Vulnerabilities

2.1 Incorrectly filtered escape characters

When the user input that is used in a SQL statement is not filtered for escape characters, then this form of SQL injection takes place. Consider this sample SQL code which displays us the records of the specified username,

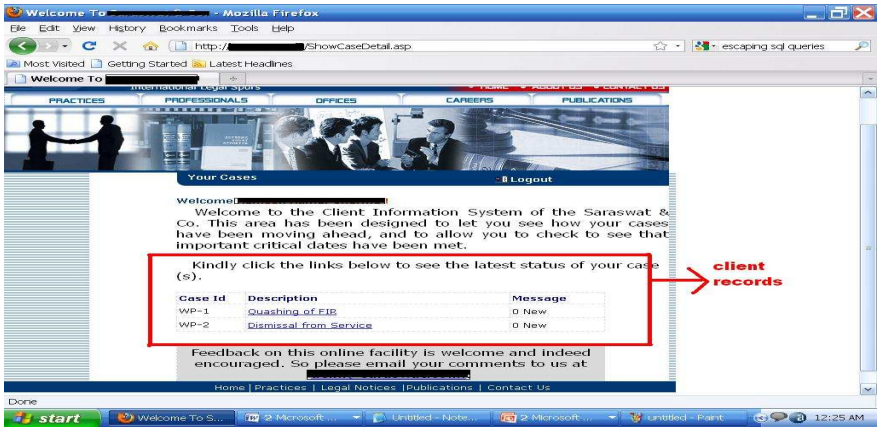
```
statement = "SELECT * FROM customers WHERE name = '" + customerName + "';"
```

JICLT
Journal of International Commercial Law and Technology
Vol. 6, Issue 4 (2011)

If the "customerName" is replaced by SQL string, like anything' or 'x'='x during authentication, the database responds to the code in same manner as the first code and displays the records. This is because evaluation of 'x'='x' is always true. The hacker inserts a series of SQL statements into a 'query' by manipulating data input. Moreover the single quote allows the code inside the quote to execute.

We took a website and injected the above string.

This injection is successful as it logged on to the account of a client of the company; so even this simple code can cause trouble.



Sometimes, the SQL Server implementations such as php's mysql_query do not allow multiple statements to be executed with one call for security reasons. This prevents hackers from injecting entirely separate queries, but doesn't stop them from modifying queries. The following value of "userName" in the statement below would cause the deletion of the "users" table as well as the selection of all data from the "data" table (in essence revealing the information of every user):

*a';DROP TABLE users; SELECT * FROM data WHERE name LIKE '%*

This input renders the final SQL statement as follows:

*SELECT * FROM* users *WHERE* name = *'a'*;DROP *TABLE* users; *SELECT * FROM DATA WHERE* name *LIKE '%'*;

2.1.1 Incorrect type handling

When a user supplied field is not strongly typed or is not checked for type constraints this type of SQL injection occurs. This could take place when a numeric field is to be used in a SQL statement, but the programmer makes no checks to validate that the user supplied input is numeric. For example consider a statement

"SELECT * FROM data WHERE customer_id = " + variable + ";"

From this statement it is clear that the author intended "variable" to be a number mutually relating to the "customer_id" field. However, if it is in fact a string then the end user may manipulate the statement as they choose, thereby bypassing the need for escape characters. For example, we can replace "variable" with the

1;DROP TABLE Customers

It will delete (DROP) the " Customers " table from the database, since the SQL statement is executed as follows

SELECT * FROM DATA WHERE Customer_id=1;DROP TABLE Customers;

1)***magic string****:* It is a sample SQL string which is used in login pages to get access to the database.

The magic string is 'OR"=' . When you use this string in the login page you will be logged in as the user in the database.

2.2 Vulnerabilities inside the database server

Sometimes vulnerabilities can exist in the underlying OS, the Web application or the database system itself, as was the case with the MySQL server's mysql_real_escape_string() function. Hackers typically test for SQL injection vulnerabilities by sending the application input that would cause the server to generate an invalid SQL query. If the server then returns an error message to the client, it allows an attacker to perform a successful SQL injection attack based on error even if the user's input is being escaped.

2.2.1 Blind SQL Injection

When a web application is vulnerable to SQL injection but the results of the injection are not visible to the attacker, this type of Injection is used. The content of the page is displayed differently depending on the result of logical statement injected into the legitimate SQL statement called for that page. An attacker checks for the vulnerability by injecting Malicious SQL codes and if the server returns an error message, the attacker can interrupt the error message and perform successful SQL injection attack. The program accepts data from a client and executes SQL queries without validating the input of the client. Then an attacker can modify, add, or delete the content from the database. This type of attack can become time-intensive because a new statement must be crafted for each bit recovered. Many tools are available to perform this attack automatically without manual interaction once the target information and the vulnerability location is found

a) Conditional Responses

A type of blind SQL injection forces the database to evaluate a logical statement on an ordinary application screen. Consider a sample SQL statement:

SELECT Customer_name FROM Customer WHERE Customer_Id = '125' AND 1=1

The above statement will result in a normal page while

JICLT
Journal of International Commercial Law and Technology
Vol. 6, Issue 4 (2011)

SELECT Customer_name FROM Customer WHERE Customer_Id = '125' AND 1=2

will likely give a different result if the page is vulnerable to a SQL injection. This will help the attacker to know the web application is vulnerable and blind SQL injection is possible, then the attacker will devise statements that evaluate to true or false depending on the contents of a field in another table. Whereas if an application is secure it will reject this request because the user's input would be treated as a value, and the value "125 AND 1=1" would cause a type mismatch error.

b) Conditional Errors

In this type of SQL injection an statement which results in error is evaluated and it will returns an error only if the WHERE condition is true, For example,

SELECT 1/0 FROM Customers WHERE Customer_id='125'
the division by zero will only be evaluated and result in an error if Customer 125 exists.

c) Time Delays

This is a type of blind SQL injection a SQL query which takes a long running time during execution is injected. Based on the time taken for execution the attacker can measure the time the page takes to load to determine if the injected statement is true.

We tested a website , did a type of blind SQL injection and obtained the details of the user at the bottom of the page. The below SQL code is used to obtain the details from the database

http://doxxxxxx.com/view_faq.php?id=147+union+Select+concat(admin_id,0x3a,admin_email_address,0x3a,admin_password),2+from+admin--

1:PROMARKINFOTECH@YAHOO.COM:9FCC265199B6BEFD223E117A8ACAEC78:803:
SANTOSHRUK@DOLIOSIS.COM:B302B49DB439BE28CDBB6E23D67D516B:D5

This was the thing found in the website after attack. It contains the *admin_id*, *admin_email_address*, *admin_password* respectively. this reveals the username of the admin and the password which is md5 encrypted. It can be easily decrypted.

.

3. Importance of Database Security

On August 17, 2009, the United States Justice Department charged an American citizen for theft of 130 million credit card numbers using an SQL injection attack reportedly "the biggest case of identity theft in American history". About 500,000 pages which use Microsoft's IIS web server and SQL server were attacked between April and August 2008 using SQL injection. In July 2008, Kaspersky's Malaysian website was hacked using SQL injection. Several other major sites that were hacked by SQL injection were:

- Kaspersky's American website, Bitdefender Portugal site and several American government websites.
- Pakistan hackers hacked Indian railways and several government websites of India. The Indian hackers hacked back most of all Pakistan government website during the time of Mumbai attack resulting in a cyber war. Eventually many websites were hacked on both sides.

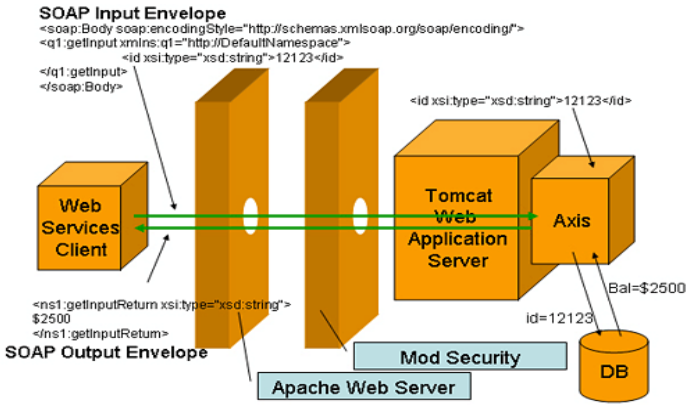
We tested for vulnerabilities inside government websites of some prime nations and the result astounded us as most of the websites were not secure enough and we were able to access all confidential documents.

4. Preventing SQL Injection

4.1 Interpret at web server level

We need to prevent SQL injection at web server level itself. It is one of the best ways of preventing SQL injection. We suggest the use of Mod –Security, an open source web application firewall that can be installed in the server and it alerts the host whenever the keyword specified is encountered.

First, it will *canonicalize* the input by sequence of anti-evasive method and other functions like removing multi-forward slashes and self-referenced directories, treating forward and backward slash equally, decoding URL and replacing null bytes (%00) with spaces. validation of URL encoding and Unicode encoding and also byte range verification(only specific character values are allowed as part of a request).A single page request may provoke many static requests for multimedia contents, cascading style sheets, JavaScript libraries and various other contents. Mod-security is intelligent enough to differentiate real website request and other functions are changing server signatures and internal *chroot*. It also helps to protect the application without modifying the source code of the application. This long time well known application is reliable, free, stable and well documented. it can be deployed as a network gateway by combining with apache operating as reverse proxy. There are numerous security benefits available with mod security and it is perfect.



4.2 To interpret at language level

The main, basic and essential prevention of SQL injection is by writing secured source code. so when strong secured Source code is written, the intrusion of database is very difficult. This can be done by escaping in PHP.

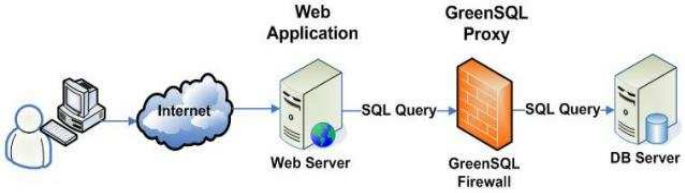
Generally, we escape characters in php by replacing a single quote(‘) in a parameter by double quote to make it a valid SQL statement..Web developers regularly use mysql_real_escape_string() function for escaping special characters. It adds backslashes to characters \x00, \n, \r, \, ', " and \x1a before sending the query to MySQL. Using parameterized statements in PHP, Instead of embedding user inputs in the statement parameterized statements are used with parameters where user inputs are bounded to the parameter. Here, the structure of the query is pre-defined and includes placeholders or bind variables for real data. Casting the variable can be done for validating integer input. Data type validation is the basic prevention method. But many are unaware that this regularly used method for preventing SQL injection is ineffective and it can be easily bypassed by an intelligent hacker. So what must be done is to take extra care and create a code like this in a separate function.

> $sql=preg_replace(sql_regcase("/(from|select|insert|delete|where|drop table|like|show tables|\'|'\|
> |=|-|;|,|\|'|<|>|#|*|--|\\\\)/"), "" ,$sql);
> Preg_replace is a PCRE function(perl Compatible Regular Expressions) in PHP. it searches and replaces.

Using this helps us to locate specified characters and replaces it with double quotes (“”). We advice the web developers to migrate to latest versions of PHP for security purpose. Dedicated care should be given for securing the coding.

4.3 To interpret with SQL firewall

SQL injection can be prevented to some extent by using GreenSQL, which is an Open Source database firewall that works as a proxy and inbuilt MySQL support. It blocks Db administrative commands (signature-based subsystem) and uses risk scoring matrix to analyse commands. The application first connects to the Green SQL server first instead of usual method of connecting to the real database server (MySQL server).This Green SQL acts as an interpreter between the real server and application and interprets the queries. It approves the queries to be passed to the real server working of Green SQL. The picture below describes the whole process.



It operates in the following different modes:

- **Intrusion Detection System(IDS):** It uses the risk scoring matrix engine to find suspicious queries and notify the database admin by the GreenSQL Management Console.
- **Intrusion Prevention System(IPS):**It checks whether the suspicious query belongs to the white list(these contains list of queries specified by the admin at first).if it is the former, it directs it to the MySQL server but if it is not found in white list, then it will generate an empty result set. Unknown queries are always blocked.
- **Learning Mode:** It is advised to use the learning mode at initiation because in this mode, all queries (only queries that are required) are added to the white list making the task easier for the admin.

GreenSQL uses strong heuristics for detecting illegal queries.(anomaly detection subsystem)

4.4 User Privileges

The administrative power should be subdivided to admin user accounts. The *superuser* concept (the super admin who can edit, create and drop any table) should be avoided. One admin user account should be limited to one or two operations so that even if one account is compromised, it vastly reduces the possible damage.

*4.5 Direct SQL*Plus Access*

An attacker looks for direct SQL*Plus Access because the Oracle Application Server is secure enough. Sometimes an intruder gets away with the application server password. In this case, when he tries to login with the compromised password, he will fail because the external session or the IP address will not be the expected one by system predicate. It will hide all the data and becomes a road block for the intruder.

4.6 Encrypting Data

For high security, encrypting the data should be the must. It is another layer of security for sensitive data .The keys of the encrypted data can be stored in separate table or even in separate server and can be related in a secret fashion. Various efficient encryption techniques can be used and incorporated into the database, but anything can be decrypted so that the secret lies in the key management and inaccessibility. We have to create a very difficult maze for hackers to break into. The choice of encrypting algorithm and encrypting packages is with the admin but it should be efficient and intelligent. Devoted time plus extra effort is really worth for securing sensitive details. We need to modify the widely available methods in internet into a more reliable one and use it wisely. It is not only about encryption but also testing. Proper testing prior to implementation is known to correct critical errors.

4.7 Other precautions

The database should be developed in such a way that the whole security implementation should be known only to certain extent to the database admin or staffs. It should be fully known only by a trusted developer. We need to be informed regularly about the access detail. Administrators should monitor the access details to check for intruders. It is best to update the database software regularly with latest releases.

4.8 Advantages and Disadvantages

Our procedure for preventing SQL injection is cost effective. All recommended software is open source. All methods are tested to be secure. Data theft can be prevented to a greater extent because common hackers find it impossible for breaking into the security structure. Our paper suggests that it is necessary for government and business organizations and educational institutions to consider implementing our procedure and fill the security hole existing in today's internet world. We have provided the way to prevent and confuse the hacker. Though the password is compromised, damages can be controlled. Awareness is needed to be spread quickly among the web developers and organizations about the important need for securing information. Unfortunately, even though the suggested methods are secure, there is a risk that hackers will find loopholes and we need to be prepared for this.

5. Conclusion

We have shown the importance of information security and present condition about how insecure are major websites. This *attack SQL injection* can be eliminated and its threat can be minimized to greater extent only if government and private organization are informed about the seriousness of the security. In some countries,

cyberlaw is not developed or strictly implemented, allowing hackers to take advantage of the system. Using existing technology and security measures intelligently by major sites will lead us to the internet with minimal vulnerability to database theft.

References

1. http://www.greensql.net/
2. http://www.modsecurity.org/
3. Litchfield, David (2005) The Database Hacker's Handbook: Defending Database Servers. John Wiley & Sons © 2005
4. http://www.oracle.com