Prevention techniques of code injection/vulnerabilities for web applications

Bonny A. Thacker
Student, Computer Engineering (Network Security), GTU PG School, Ahmedabad, Gujarat, India

ABSTRACT

Security remains a major challenge to the entire web community. Web-applications are easily compromised due to security reason. Moreover, one of the major reason for exploitation of web application is user input i.e. user input is being processed in an unsafe manner.
SQL injection has been a threat for more than 15 years, so it is astonishing to recognize that it is still one of the top data threats to organizations. SQL Injection is an attack used to exploit user applications that build SQL statements from user-supplied input. Proper neutralization of special characters used in an SQL Command is needed to avoid the SQL injection. Existing IDS base prevention system working on predefined rules but they are not enough and stable. So, in order to prevent web application database from SQL Injection Attack, we propose a prevention technique.
In this Prevention Technique, threat logs are generated of the attacker using SNORT. Attackers request will forward and log in Honeypot. From the Honeypot, a Block list would be created; this list will be blocked by iptables in Web Application Machine. If an attacker is successful to bypass a layer of SNORT, MAC address of the attacker will be blocked in iptables and the database is already secured with ASCII which will not allow any attacker to attack the database.

Keyword: - *SQL injection, honeypot, snort, iptables, SQL injection prevention technique*

1. INTRODUCTION

1.1 SQL INJECTION:

SQL Injection attack is a kind of code injection technique which exploits the vulnerability present in the application code for gaining the unauthorized access to the data. The effects of SQL injection attacks are a modification of data, destroying some fields of data, unauthorized access to data, stealing of data, dropping down the entire database, etc. The types of SQL Injection attacks are:

- Tautologies
- Union Query
- Piggy-Backed Queries
- Illegal/Logically Incorrect Queries
- Stored Procedures
- Blind SQL Injection

1.1.1 Tautology:

This technique is inserting the string which makes the query always true. Then the query always returns upon evaluation of WHERE condition. For Example, an attacker submits " ' or 1=1 - -" for the input field. The code injected in the conditional (or 1=1) transforms the entire WHERE clause into a tautology.

1.1.2 Union query:

An attacker injects a UNION SELECT to trick the application into fetching data from a table. Attackers do this by injecting a statement of the form: UNION SELECT rest of injected query.

1.1.3 Piggy-backed query:

The original query is followed by and extra query which will harm the database. In this type of attack, the attacker injects malicious code with traditional queries and also performs data manipulation operation like lNSERT, UPDATE and DELETE clause for manipulating a record.

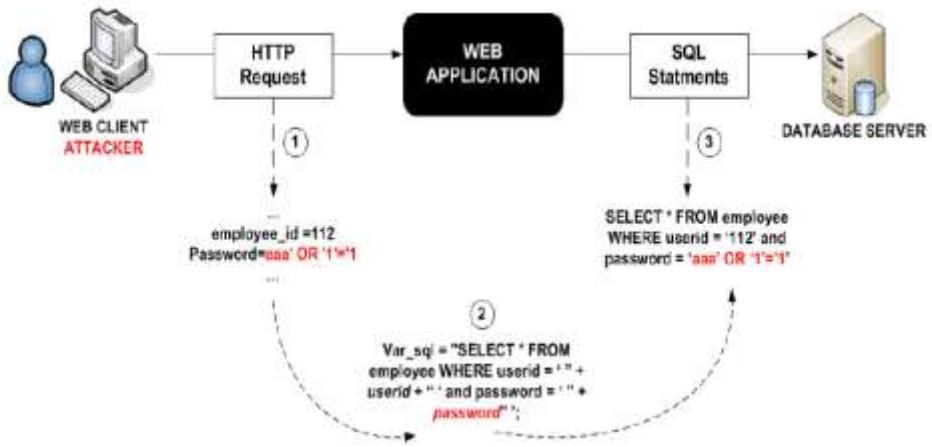

Figure 1: SQL injection Example

1.1.4 Error Based Query (Logically incorrect query):
This type of attack represents the SQL manipulation category in which attacker can get advantage from error message which is generated by the database server. In fact, the generated error messages can often reveal vulnerable/injectable parameters to an attacker.

1.1.5 Stored Procedure:
SQL Injection of this type tries to execute stored procedures present in the database. To launch an SQLIA, the attacker simply injects "'; SHUTDOWN; - -" into either the userName or password fields.

1.1.6 Blind SQL Injection:
In these types of attacks, an attacker can steal data from database asking true and false questions through SQL statement. Classical blind exploitation is based on the analysis of the true/false logical expression. If the expression is true, then the web application will return certain content, and if it is false, the application will return some other content.

1.2 HONEYPOT:

The honeypot project was started with kept in mind to observe the activities of the attackers. A honeypot is nothing but a system which is created to emulate the services that are executed on the server in order to observe the patterns of the attacks. A honeypot is used for the purposes like production or researchers.

1.3 ASCII:

ASCII stands for American Standard Code for Information Interchange. The computer can only interact with numbers, so to identify A or $ or any other character or symbol or any events like opening a file. Use of ASCII value in the normal database is like add 2 new columns of the username ASCII value and password ASCII value. These columns will give prevention for an injection attack.

2. PROBLEM STATEMENT

One of major issue with a web application is its security. Nowadays, web applications are easily compromised due to security reason. Moreover, one of the major reason for exploitation of web application is user input i.e. user input is not filtered properly. All the data provided by a user must be treated as untrustworthy.

Major threats are being produced in database domain and one of the major issues is SQL injection. Proper filtration of special characters used in an SQL query is needed to avoid the SQL injection. There are different techniques/process such as hashing, query transmission inputs from the users, header sanitization, an algorithm for analyzing, etc are being used by the developer. However, all these techniques are not fully protective.

One of the key issues is IDs rules. Automatic updations of IDS rule are not possible. To update the IDS rule, there are manual interventions required. A technique needs to be evolved to overcome this issue as well as also a time-saving approach to reducing manual work.

3. PROPOSED WORK

Snort IDS need to be placed. Initially, a request will be triggered to the proxy server while attacking (by web attacker) attempts to enter into a web server. As shown in below, the IDS will match the existing attack as per received request. On basis of request and matching signature, IDS will take a decision regarding dropped or allowed to access the real web server.

In a genuine database server, when the user enters login credentials that values are converted into ASCII value and compared with stored ASCII signature in the database server. In the event that the match found the user will be permitted to get access to the genuine database server and web application, otherwise, that attacker's request will be sent to the honeypot server.

In honeypot, that attacker's request will be tracked, monitored and logs will be generated. If any SQL injection is encountered here, Using those logs, IP address and MAC address of the attacker will be extracted and stored in the database. Those IP address and MAC address of the attacker will be blocked using iptables. Then new snort signature is generated and the snort rule will be updated to the IDS.

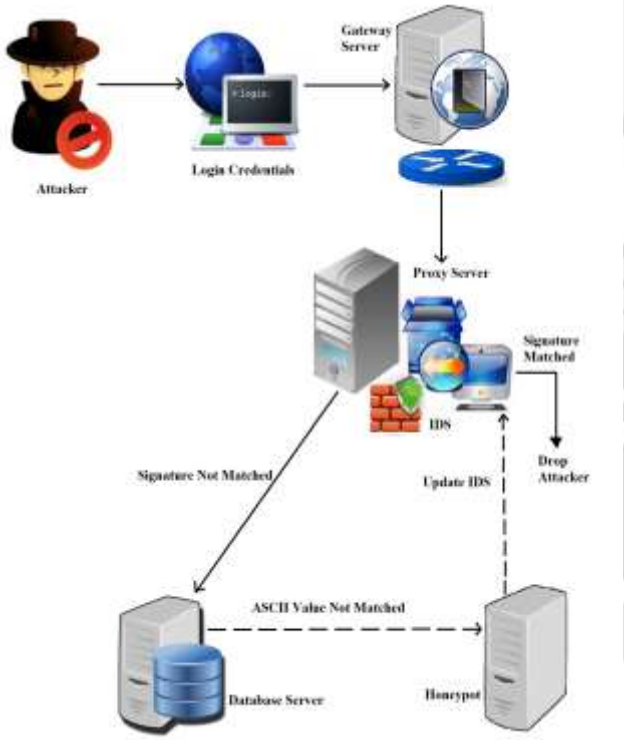

Figure 2: Proposed prevention technique

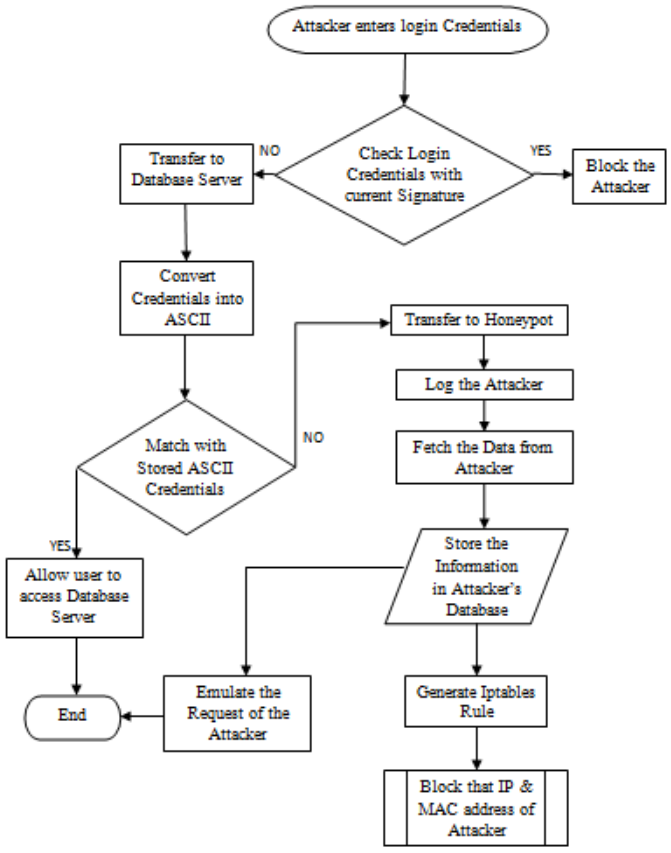

Figure 3: Flow chart of proposed prevention technique

4. EXPERIMENTAL SCENARIO

We have created a replica of web server1 to execute the process of SQL injection. Login credentials are stored in ASCII value. SNORT is implemented here which has the rule to alert about SQL Injection from any IP. Below Figure shows that SQL injection is encountered here and the attacker is logged in.

Figure 4: Logged on Screen by SQL Injection

Attack Detection - In the bellow figure the alerts generated by snort IDS is shown as a result of SQL injection attack on the web server.

```
GNU nano 2.2.6                File: myrule.rules

alert tcp any any -> $HOME_NET $PORT_HTTP
(msg: "SQL Injection Attempt - and 1=1";
content: "POST"; http_method; uricontent: "' or 1=1 --";
nocase; classtype:web-application-attack; sid:5000001; rev:1;)
```

Figure 5: SQL injection-Tautology Query detected by snort rule

```
[**] [1:10001:0] SQL Injection Attempt - or 1=1-- [**]
[Priority: 1]
5/9 11:38:23.103633 192.168.92.2:80 ->192.168.92.254:80
TCP TTL:64 TOS:0x0 ID:17259 IpLen:20 DgmLen:587 DF
***AP*** Seq: 0xEE2F694E Ack: 0xF551D9B6 Win: 0xB7 TcpLen: 32
TCP Options (3) => MSS: 1460 SackOK TS: 104385 0 NOP WS: 5
```

Figure 6: Log of Snort detecting SQL Injection

We have created another web server2 for Glastopf installation to prevent real database. After SQL Injection detection, the attacker's request will be sent to the honeypot. Attacker's request is tracked and monitored in the honeypot. In the screenshot of the logs, it can be seen that the attack was attempted by the IP address 192.168.92.128 which is the IP address of the attacker machine.

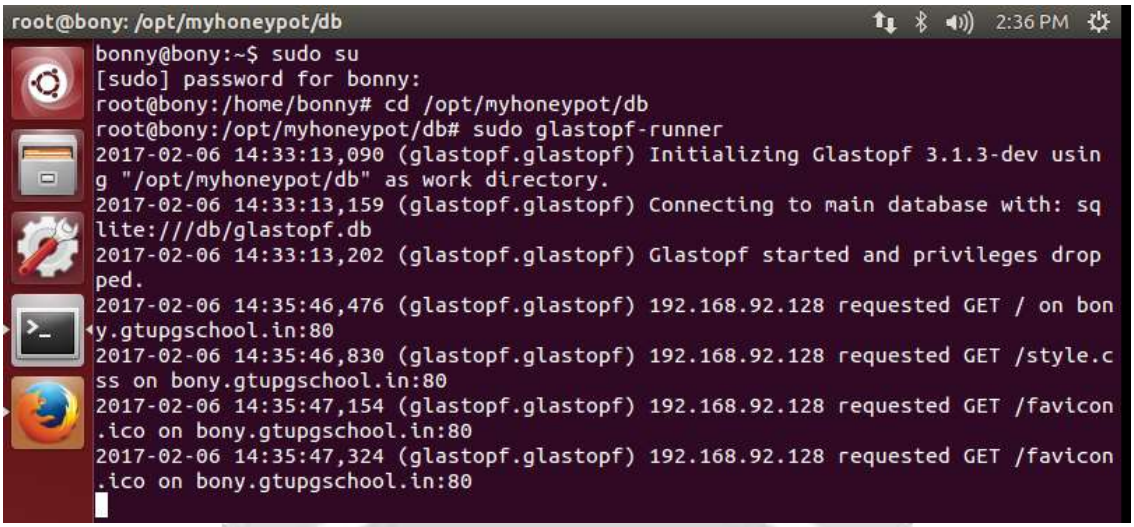

Figure 7: Logs from Glastopf

The attacker's IP address and MAC address are extracted and stored in the database using the python script. The outcome of the script is shown in below figure.

Figure 7: Result of the script

The List of the IP address and MAC address of the attacker will be sent to the web server1 where snort is implemented. Those IP address and MAC address of the attacker will be blocked using iptables. Then new snort signature is generated and the snort rule will be updated to the IDS.

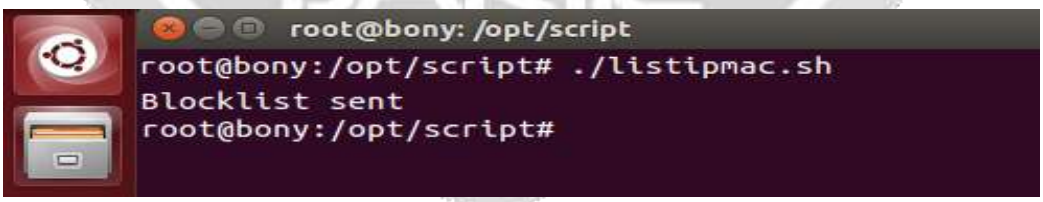

Figure 8: Send block list to the web server

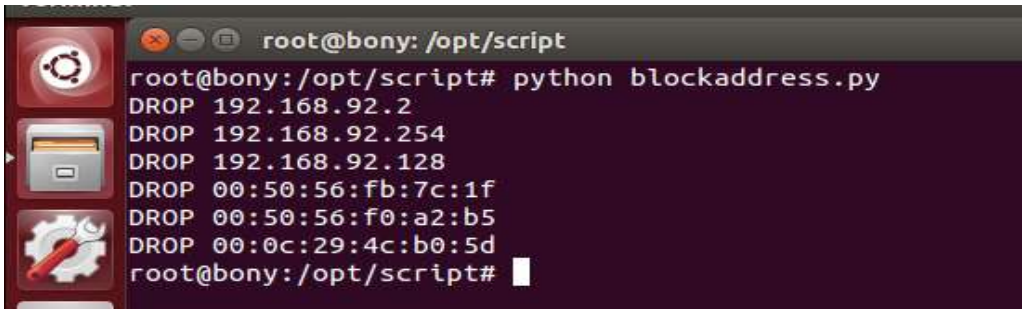

Figure 8: Script to block the IP and MAC addresses

5. CONCLUSION

Security remains a major challenge to the entire web community. One of the major reason for exploitation of web application is user input i.e. user input is being processed in an unsafe manner. SQL injection has been a threat for more than 15 years, so it's astonishing to recognize that it's still one of the top data threats to organizations. SQL injection vulnerability allows an attacker to flow commands directly to a web application's underlying database and destroy functionality or confidentiality. The proposed work will secure the database server from the attacker. Honeypot will lure the attackers and generate the database of the attacker. If the attacker can bypass the IDS, the ASCII protected database still will be safe. From the Honeypot a Block list is generated on the web server, which will block by iptables rules. Another layer of the security is added by blocking IP address and MAC address of the attacker. So, Layers of security can maintain confidentiality and integrity of secure web application.

6. REFERENCES

[1] Rui Silva, "Testing Snort with SQL Injection Attacks", 2016 ACM

[2] Hussein Alnabulsi, School of Comp & Mathematics, Charles Sturt University, Albury, NSW, Australia, "Detecting SQL Injection Attacks Using SNORT IDS", 2014 IEEE

[3] Mahima Srivastava, "Algorithm to Prevent Back End Database against SQL Injection Attacks", 2014 IEEE

[4] FX Arunanto, Baskoro Adi Pratomo, "Aggressive Web Application Honeypot for Exposing Attacker's Identity", 2014 1st International Conference on Information Teclmology, Computer and Electrical Engineering (ICITACEE), 2014 IEEE

[5] Hudan Studiawan, FX Arunanto, Baskoro Adi Pratomo, "SQL Injection Detection and Prevention System with Raspberry Pi Honeypot Cluster for Trapping Attacker", 2014 International Symposium on technology management and emerging technologies(ISTMET), May 27-29, 2014 IEEE