Safety Measures and Auto Detection against SQL Injection Attacks

Sandeep Choudhary, Nanhay Singh

Abstract: The SQL injection attack (SQLIA) occurred when the attacker integrating a code of a malicious SQL query into a valid query statement via a non-valid input. As a result the relational database management system will trigger these malicious query that cause to SQL injection attack. After successful execution, it may interrupts the CIA (confidentiality, integrity and availability) of web API. The vulnerability of Web Application Programming Interface (API) is the prior concern for any programming. The Web API is mainly based of Simple Object Access Protocol (SOAP) protocol which provide its own security and Representational State Transfer (REST) is provide the architectural style to security measures form transport layer. Most of the time developers or newly programmers does not follow the standards of safe programming and forget to validate their input fields in the form. This vulnerability in the web API opens the door for the threats and it's become a cake walk for the attacker to exploit the database associated with the web API. The objective of paper is to automate the detection of SQL injection attack and secure the poorly coded web API access through large network traffic. The Snort and Moloch approaches are used to develop the hybrid model for auto detection as well as analyze the SQL injection attack for the prototype system.

Keywords: Moloch, Snort, Sqlmap, SQLIA, Threats, Web API vulnerability

I. INTRODUCTION

SQLIA is a query based attack in which attacker infuse the malicious program to attack database of web applications. During this process, the attacker integrate the part of malicious statement in the actual SQL parameter and post the malicious request to targeted database server. The SQL injection on web API [15] is the common attack which is executed by the attacker. SQL injection will not need any permission to the authentic user, instead of that it will redirect the information of the database to the attacker. In year I988, Computer Emergency Response Team (CERT) [14] is developed by CERT Coordination Center at Carnegie Mellon University (CMU) which handles the security against network attacks like a worm, virus, malware, etc. The work on this paper is to study the SQL injection attack patterns and auto detect these pattern using snort method through signature mapping. All SQL injection possible patterns stored in snort rules and used to compare with all passing PCAP packet through snort. There several network attack data's are present which is compared with the free available systems. The goal is an analysis of data at runtime environment for a large task.

Revised Manuscript Received on December 30, 2019.
* Correspondence Author
Sandeep Choudhary*, Department of CSE , Ambedkar Institute of Advanced Communication Technologies & Research, Delhi (India) , Email : er.sandeepchoudhary@yandex.com
Prof. (Dr) Nanhay Singh, Department of CSE, Ambedkar Institute of Advanced Communication Technologies & Research, Delhi (India) , Email : nsingh1973@gmail.com

A. SQL Injection Overview

Mainly the SQL injection attacks are executed on client-server architecture. The web API acts as a thin-client, where the usersends the query to extract the data from the database server. The basic architecture for web API and the database server is illustrated in fig. 1.

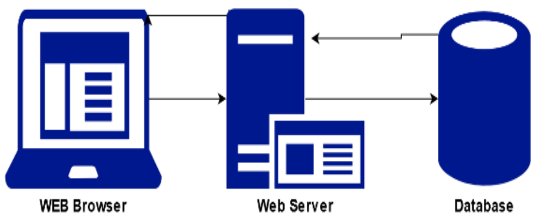

Fig. 1. Web Application Architecture

SQLIA attack rankings on the best ten listing of web API vulnerabilities as indicated by the study of OWASP (open web application security project) [22]. Although the goals of SQL injection attacks aren't just for web API but they are also able to hit on programs, which can be driven by their own database used SQL. The amount of financial loss resulting from SQL injection was very high, so it is necessary to find a scheme to prevent from these type of attack i.e. SQLIA. Attackers may inject vulnerable code through input fields of web application forms or by adding directly malicious queries in the URL of web API. SQL injection provides free cyber space to hackers, where it develops and execute the script on the network. The hacker's developed bots to check and identify the vulnerability in the websites. The bots (bots.txt) are run on the network and compromised the server machine. In most common these type of botnets are created and used by the attacker while executing the distributed denial of service attack. DDoS is the most common attack on the server machine.

B. How SQL Injection Work?

It is a query based attack was user inject the piece of code to web API. The malicious query will provide the database table information in the URL parameter. Fig. 2, represents the SQL injection on a simple website to find the records. The statement consist of SQL query associated with the code of data to be injected for information retrieval.

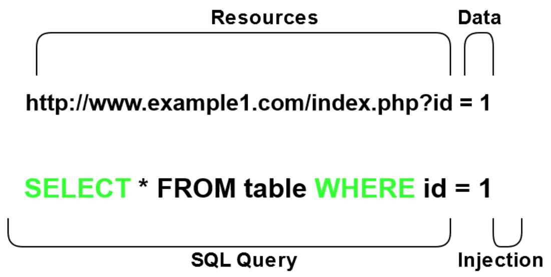
Resources
Data
http://www.example1.com/index.php?id = 1
SELECT * FROM table WHERE id = 1
SQL Query
Injection

Fig. 2. SQL injection in web API to the find the records

II. CLASSIFICATION OF SQLIA

Different ways are available to execute the SQLIA Attack [16] such as:

A. Boolean-Based Blind Injection (BBBI)

In this, the logical query is attached with the parameter and the attacker waits for some meaningful search. The malicious query will redirect some result which is related to Boolean operation (True or False). The "WHERE" operator is used to evaluate the tautology of the parameter. Let us consider a Boolean based malicious string.
http://www.sybertechnologies.com/dvwa/vulnerabilities/sqli ?id=3 AND substring (@@version, 1,2) = 5

B. Time-Based Blind Injection (TBBI)

It uses the time of the server to access the information of the database. The format for TBBI is applied on any website URL.http://www.sybertechnologies.com/dvwa/vulnerabilities/sqli/?id=1 AND User='admin' WAIT FOR DELAY '00:00:15'

C. UNION Based Injection (UBI)

It uses for merging the two different table row. The only disadvantage of UBI are (i) Tables structure are same, (ii) the Same number of row and column is present. UBI used the "ORDER BY" operator for finding the column.
http://sybertechnologies.com/index.php?id=10 ORDER BY 1 -> OK
http://sybertechnologies.com/index.php?id=10 ORDER BY 2 -> Error

III. LITERATURE REVIEW

There are various studies has been done by different researchers in the field of SQL injection and database exploitation. The attackers violate all type of security layers and protocol's to access that information. N. Singh et al. [1], discussed attacks and prevention against SQL injection. They proposed the firewall technique for the SQL server which will restrict the privilege of the unregistered users. But for using this service it needed to be the node to node signature authentication.V. K. Gudipati et al. [2], uses the Sp_executesql to execute the syntax in a specific order which replaces the QUOTENAME. It also manages the permission at the time of attacks. Kamtuo and Soomlek [3], uses the machine learning technique for analysis of attacks. It also extractsinformation for training and testing. R. Karuparthi and Zhou [4], introduced a User Defined Approach (UDA) for mapping the attribute to a specific requirement. It also checks the threshold value for any attacks. R. Dubey and H. Gupta [5], is uses introduced the filtering mechanism for sending and receiving the request. N.A. Al Sayid and D. Aldlaeen [6], introduced a firewall technique to obstruct the SQL injection attack. A.Shastri and P.N. Chatur [7], uses a security-based model for checking the signature of the authentic users. N.A. Al Sayid and D. Aldlaeen [8], proposed access control policy for user authentication and identification.P. Ghorbanzadeh et al. [9], introduced firewall and virtual private for the prevention of unwanted intrusion on mobile database. Sallam et al. [10],introduced a Role-based anomaly detection approach for an insider attack. S. Fatih [11] and F. Mouton et al. [12], introduced a web-based security approach to protect against the SQL injection. Orman [13], proposed a Blockchain concept to verify the genuine nodes of the web server.

IV. RESEARCH METHODOLOGY

After exhaustive study of several research paper, we proposed a framework for automated detection of SQL attack using Snort [18] and Moloch [17]. Fig. 3, represents the outline for the detection system.

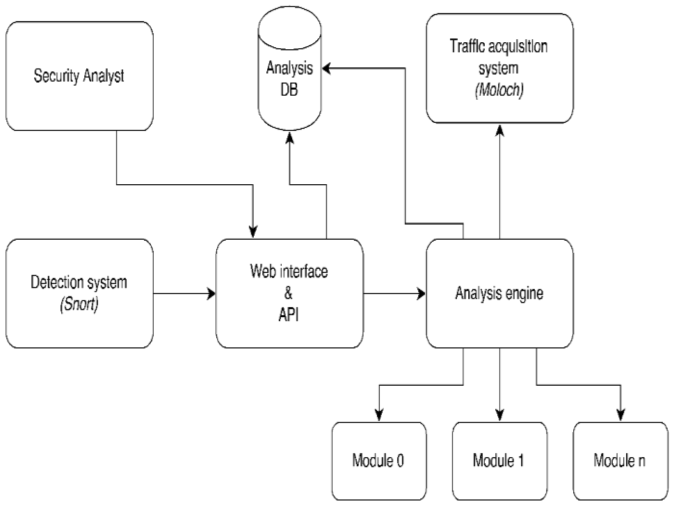

Fig. 3. Framework for Auto Detection against SQLIA

It uses the Snort technique for detection of SQL injection. It also analyzes the huge HTTP network traffic. Snort was first created by Martin Roesch in the year of 1988 [19] for network intrusion detection. Traffic acquisition system uses the Moloch as a default system to gain the visibility of SQL injection. During the attack, the several packets are not logged due to the performance reason. It uses the IPv4 packets for detection of the intrusion. The framework is divided into several components. And each component has specific work such as:

A. Detection system

It uses the Snort technique for detection of SQL injection. It also analyzes the huge HTTP network traffic. Snort was first created by Martin Roesch in the year of 1988 [19] for network intrusion detection. Fig. 4, represents the basic framework for Snort. Basically, Snort is a rule based system which used to match with the each captured packet for detection of suspicious activity pass through the system.

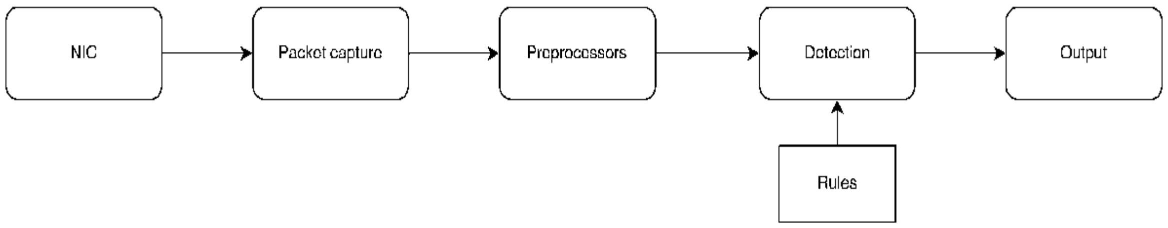

Fig. 4. Snort Framework

In packet capture is used to collect the request and response of the system. The pseudo code for packet capturing is given as follows:

```
Class Public login(req, resp)
(
String login = req;
Get.parameter("login");
String pattern = req; get.parameter( "pattern");
String qry1 = "SELECT info FROM userTable WHERE";
if ((!signin. equals("")) && (!p_word.equals("")) ) query += signin = "'+ signin+'" AND pass= "'+p_word +'"
elsequery+ = "'signin= 'Guest''";
ResultSet result1 = stmt. executeQuery(qry1) ;
if (result1 != null)
showAccount(result1) ;
else
sendAuthentacationFailed();
}
```

The genuine user is only pass through the authentication process. If there is a SQL attack then it uses the specific keywords to identify the attacks. Here is an example to detect the ICMP packet in ECHO REQUEST.

```
alert icmp $External_NET
PAC=$_HOME PSC( MSG : "PING"; i_code:00; i_type:08; class_type: m_activity; s_id:348; rev:8;)
```

The package of snort is divided in two parts Header and Body.

The Header is divided into seven different fragments:

(i) Actions
(ii) Protocol
(iii) SourceIP
(iv) SourcePort
(v) Direction
(vi) Dst IP
(vii) Dst Port

These fields are consist of variables or string to match with the database. The body uses the payload or HTTP headers of message. The alert is generated by using the fast_output modules. Example is given as:

```
[**] [1:374:5] ICMP PING [**]
[Classification: Misc activity] [Priority: 4]
03/12-02:11:09.359780 10.1.1.10 -> 10.0.1.253
ICMP TTL: 30 TOS: 0x0 ID: 38175 IpLen: 15 DgmLen: 92
Type: 7 Code: 1 ID: 32353 Seq: 5 ECHO
```

Starting statement is used for packet matching. The [**] symbol is used for starting and ending of the sequence. There are three values are present in the brackets which is colon separated.

Generator_ ID (GID) is used in alert module.

Snort_ ID (SID) is used to identify the unique alert.

Revision_ No (REV) is used to trigger the alert.

B. Traffic Acquisition System

It uses the Moloch as a default system to gain the visibility of SQL injection. During the attack, the several packets are not logged due to the performance reason. It uses the IPv4 packets for detection of the intrusion. Moloch session uses the seven tuples:

Moloch_Session = St_Time , Sp_time , Source_IP , Dstination_IP , Source_Port , Dstination_Port , Protocol

Moloch is consists of three main parts:

- *Elastic Search database:* It is used in indexing of stored sessions. It also managed the captured sessions. In real-time, the large volume data is managed by using the network traffic analysis.
- *Capture:* It separates the network between captured and network traffic.
- *Viewer:* It is used for filtering the stored session and it also exports the stored session.

Moloch is used for HTTP session for filtering. The Moloch filtering expression is given as:

```
ip.src == 10.0.0.41 && start_time >= "2019/03/05 22:11:23" && port.dst == 8080
```

C. HTTP Tag Filtering

This is the procedure for Moloch for packet filtering. The next step is the Analysis Engine. The work of analysis engine is to analyze the malicious packets. The PCAP analysis is done by modules. Whereas each module will perform certain amount of actions on PCAP to return the output from engine. The database is used to store the information (such as attack investigation) for analysis.

```
Protocol = = http && method = = GET && status = = 200 && stop_tm <= "2019/02/05 12:21:03"
```

D. Attacker IP Details

The attacker IP is very trivial to find by any IDS. Attackers often hide their IPs and location server o secure their personal information. According to CloudFlare, there are 90% of the request is came from Tor browser which is very unpredictable to find the genuine request. The IP address is is consist of:

(i) The owner IP address,
(ii) Origin of Internet Service Provider (ISP),
(iii) CIDR notation,
(iv) E-mail contact,
(v) Tor node check

All the information are retrieved from Regional Internet Registries (RIRs). All Tor node is present publically and it uses as encrypted traffic to access the HTTPs.

E. Web Server Detail

It uses to find out the details of the target server. The analysis engine doesn’t have access to analyze the web server, so it uses the pattern matching technique for finding the details. Wappalzer is an open source tool to detect the web server, content management system (CMS), and JavaScript libraries.

F. Statistical analysis

If there is a crime there also evidence, the attacker leaves the fingerprint in the form of the entry point. In statistical analysis, the PCAP pattern is observed to find the evidence of the SQL injection. The analysis engine is used to identify the outline in traffic, network endpoints.

G. Database Canary

The attacker will usually try to retrieve the information such as table name and columns. The work of Canary is to add an appropriately long string which replaces the information of the database. It also sends the SQL injection successful message to the server, but instead of the actual table, it sends the empty table to the attacker. Canary is generated by 256 bits string which is placed in the database. This will not provide the security against time-based blind injection.

H. Connections

It captures the network subnet and provides the list of the host. Using Moloch will provide this information in the API panel.

I. Analysis engine

It is used for task scheduling. There are several steps are present in the analysis engine:
(i)*Status* ***:*** It monitors the task status such as “PENDING” or “PROGRESS”.
(ii) *PCAP from Moloch*: It analyze the HTTP traffic between source and destination. The alert is generated by using (/sessions.pcap endpoint). Moloch traffic filter is done by following: port.dest == { dest_port} & protocols == http & ip_src == { src_ip} & ip_dest == { dest_ip} the Moloch has observed the initial alert such as Start_Time, Stop_Time, and expression.
(iii) *Database entry update*: It changes the entry to success or
error for task analysis engine.
(iv) *Modules* : It uses the command for analysis of module . module_ results = module(opt, pcap_ path , config) . boot strap() opt = source IP, destination IP alert.
(v) *Pcap path:* It retrieves the PCAP stored file.
(vi) *Config* ***:*** it manages the RethinkDB host and port, Celery broker. bootstrap() return the stored from the database.

J. Storage and Web Interface

It uses the Rethink DB document based NoSQL database and API for real-time application. Traditional database will not provide the analysis functionality. Rethink DB will execute on separate machine using remote server or ReQL wire.
The command for NoSQL database is given as:

```
cursor = r.table(‘analyses’).filter(r.row[‘dst_ip’] ==
‘10.0.0.1’).run() for document in cursor: print(document)
```

The analysis is done by the web interface to analyze the source IP, destination IP. PCAP retrieve the Moloch information. The Snort and Moloch are both monitor host information. It supervised the running task by using the uWSGI, Celery, and Alert forwarder. The visualization of analysis is shown in the web API and it is also used to analyze the results. The individual results are offered in the form of table and map.

V. SYSTEM EVALUATION

The prerequisites for deployments are Snort 2.9.15 , winPcap4.1.3 , Proofpoint, PulledPork instances to generate the signature map. The working of Snort is to generate the intrusion logs in a binary form called unified2.
The log is consist of several alerts such as
(i) Alert-syslog
(ii) Alert-fast
(iii) Alert-full
(iv) Alert-unixsock
(v) Log-tcpdump

There are few things needed to setup the Snort log session in Moloch. The Moloch needs the subnet address as a development point, the timestamp in the UTC standard, REST API with self-signed SSL certificate, and PCAP (Packet Capture) which deletes the old unwanted session.

A. Alert forwarding

Snort is used the analysis engine to send the alert to the log processing system. The log system is constantly examined the new records. The manual implementation uses the “idstools” package in python to monitor the log record.

```
Pseudo Code: from ids_ tools import unified2;
reader1 = unified2(.)Spool_ Event_ Reader[dir,
follow,pre-fix=1(true)];
for reader event;
 # event of process;
 # ...
```

B. Signature Mapping

It is a process of setting a numeric signature ID to the textual representation. The pseudo code for the signature mapping id done by using “idstools” package.

```
import maps from ids tools
sigmap1 = maps1.signature_map()
sigmap1.ldr_sig_map(open(‘sid_msg.map’))
sigmap1.ldr_gentr_map(open(‘gid_msg.map’))
sigmap. get(g_id, s_id)
```

Additional task of Snort alert forwarding is (i) Signature filtering and (ii) Bookmarking.
In signature, filtering is used to provide the alert of SQL injection Id only. Bookmarking will help in to keep track of all the event of the analysis engine.

C. Creating Analysis

IDS will generate multiple alerts at the time of SQLIA attack. The analysis engine is used to retrieve load traffic using the API. The alerts are generated in the form of time frame and it is a cluster in one analysis. The similar alert are checked with the prescribed cluster.

There are three tuples are present in the alert. alert = (source_IP , destination_IP , source_port)

D. Task Queue

It uses the task in the distributed form means each task is executed in different processors. The Celery is an open source Python software for parallel processing. Celery is work on master and slave model, where master distributes the task for different processor and slave will run those task in parallel. The API will handle the entry of the database. The entry is consist of timestamp and task status of pending data.

VI. IMPLEMENTATION

For implementation , the experiment is conducted on linux platform using Ubuntu 18.1, MySQL database, Damn Vulnerable Web App (DVWA) and snort 2.9.15 package with winPcap 4.1.3 for capturing the packets. The Sqlmap [20] is a penetration tool used to inject the malicious query code in DVWA and Snort used for detection of these malicious code by matching with the designed rules. The workflow of experiment is divided into six steps:

Step 1: *Setup:* Sqlmap (version-1.3.12.1#dev) use as a independent penetration tool for injecting malicious SQL query and try to exploit the database of target system (DVWA). It needed some basic configuration such as: Specify the entry pointy of Target URL, HTTP header, Proxy, Tamper Scripts. It also detects the Web Application Firewall (WAF) and Protected Web Server (IWS) to access the information of operating system.

Step 2: *Enumeration:* Sqlmap also retrieves the tables and columns of the database using brute-force attack. It uses the dictionary attack for hash protected data.

Step 3 : *Signature mapping :* It uses the fingerprints and rule based expressions of the Snort before passing to web server for crucial data .

Step 4: *Snort Detection:* it checks the entry point of the links and signature of captured packets (PCAP) by winPcap.

Step 5 : *Moloch session storage:* it used in large network traffic to gain the visibility of SQLIA and stored session of seven tuples as discussed in part B of section IV.

Step 6: *Alert :* Snort detect the malicious expressions based on the setup rules and gives an alert to log file .

VII. RESULT AND DISCUSSION

For testing, we execute the 10 sqlmap (version 1.0.5.27) attacks by injecting malicious query on DVWA and using the Snort 2.9.15 package we able to detect the SQL injection based on 3 adhoc rules designed by us. The alerts are analyzed by proper PCAP (packet capture) using winPcap 4.1.3 package. Sqlmap also identifies the canary string at the response. Snort will generate the 20 different alerts for every sqlmap attack.

In the process of designing the rule expression for snort, we used some special strings, characters or combination of both used in SQL query, these are (HAVING), (JOIN), ((LIMIT), (DEFAULT), (DATABASE), (UID) and (UNION)), (AND), (OR), (%00 to %FF) including their hexadecimal values.

Rule 1: This rule is specially designed for the detection of Boolean Based Blind Injection (BBBI) Attack.

alert tcp code = $localhost code (message:" Alert : Boolean Based Blind Injection Attack "; pcre: " /(@\%0c)||(#&/?‘’) AND*/ =x"; class_type:BBBI; uid:100; res:01;)

Injection of Malicious code to target website using Sqlmap :
python sqlmap.py –r
“http://www.sybertechnologies.com/dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit# ‘--cookies’ " —dbs

Action: Snort will now start capturing the packet (PCAP) using winPcap and signature matching of packets with the designed rules.

Table- I: Experimental Result Based on Rule 1

SQLIA	True		False (C)	Success Rate of Detection (A/A+B+C)
	Positive (A)	*Negative (B)*		
BBBI	09	01	00	0.90
TBBI	05	03	02	0.50
UBI	00	02	08	00

After detection of malicious packets with high rate of success belongs to BBBI and it will throw an exception of alert to log file.

Rule 2: This rule is specially designed for the detection of Timer Based Blind Injection (TBBI) Attack.

alert tcp code = $localhost code (message:" Alert : Timer Based Blind Injection Attack ";
pcre:":/(@\%0123456789)||(&/? ‘:’)AND+/=/x@%4f@";
classtype:TBBI; sid:200; res:01;)

Malicious code injection to target website using Sqlmap :
python sqlmap.py –r
“http://www.sybertechnologies.com/dvwa/vulnerabilities/sqli_blind/?id=1 AND user= ‘anonymous’ WAIT FOR DELAY ‘00:00:30’ ‘--cookies’ " —dbs

Action: Snort react slowly in detection of TBBI because it force the database associated with the target website to wait for certain amount of time (in seconds).

Table- II: Experimental Result Based on Rule 2

SQLIA	True		False (C)	Success Rate of Detection (A/A+B+C)
	Positive (A)	*Negative (B)*		
BBBI	06	02	02	0.60
TBBI	08	01	01	0.80
UBI	00	03	07	00

After detection of malicious packets with high rate of success belongs to TBBI and it will throw an exception of alert to log file.

Rule 3: This rule is specially designed for the detection of Union Based Injection (UBI) Attack.

The third SNORT rule is designed for UNION rule based signature and with all of its possible hexadecimal values. We try to execute this rule on relational tables having same structure in a database. We used code signature (%u%)* for UNION in our rules, which mean any characters or string between UNION of two relational table.

alert tcp code = $localhost code (message:" Alert : Union Based Injection Attack ";
pcre:"/((((\%10)|(p)|(\%25))((\%0c)|(q)|(\%1f))((\%39)|(r)|(\%65))((\%2a)|(s)|(\%3d))((\%6b)|(t)|(\%2a)))[%u%]*(((\%20)|(p)|(\%50))((\%0d)|(q)|(\%5f))((\%65)|(r)|(\%5))((\%2e)|(s)|(\%3a))((\%8b)|(t)|(\%2c))))/n"; classtype: UBI; uid:312; rev:23;)

Malicious code injection to target website using Sqlmap :
Before injecting malicious code to target using sql map it is required to retrieve the following information from the vulnerable website:

- Find the no. of columns using order by clause in relational tables .
- Find out and check the union function exist in database.
- Retrieve name of table & columns names.
- Fire the malicious query to exploit the database of target website using sqlmap :
  python sqlmap.py –r "http://www.sybertechnologies.com/dvwa/vulnerabilities/sqli/?id=4%20union%20all%20select%201,2,3,4/'--cookies' " —dbs
  python sqlmap.py –r "http://www.sybertechnologies.com/dvwa/vulnerabilities/sqli/?id=4 AND 1=1 '--cookies' " —dbs

Action: Union based injection is not easier to target as it required the no. of malicious query execution to retrieve the information related to relational tables in database .
Malicious query used in initial stage for UBI to retrieve information is also detected by our set rule 1 & 2. When we try to inject malicious code containing the UNION clause, our designed rule 3 start to work and snort detect the UBI attack by matching the signature with captured packets.

Table- III: Experimental Result Based on Rule 3

SQLIA	True		False (C)	Success Rate of Detection (A/A+B+C)
	Positive (A)	Negative (B)		
BBBI	06	02	02	0.60
TBBI	05	01	04	0.50
UBI	07	02	01	0.70

After detection of malicious packets with high rate of success belongs to UBI and it will throw an exception of alert to log file.
we have observed different results on various SQLIA attack based on design three adhoc rules by snort. All rules executed on basis of tautology and signature matching of malicious code with the rules. we tries to customize rules to reach malicious packet detection up to the possible maximum success rate but snort is not deal efficiently on temper scripts like base64 conversion. Boolean based blind SQLIA achieved maximum detection up to 90 % by snort using rule 1, Timer based blind SQLIA achieved maximum detection up to 80% by snort using rule 2 and UBI achieved 70 % by snort using rule 3.All rules have executed concurrently on each malicious query passing through snort to target website i.e DVWA and output of these query detection belong to different category true positive (TP), true negative (TN) and failed to detect under false category . We consider only the true positive (TP) to evaluate the success rate of detection of malicious captured packets passing through the Snort.

VIII. CONCLUSION

SQLIA attack is the most hazardous threat for poorly coded web API and till now there are number of techniques have been purposed for detection and prevention from SQLIA attacks. Most often, attackers found the vulnerability in web API to bypass these techniques and solutions. The temper scripts like base64 conversion is not be detected by using Snort. In this paper, author presents a novel automated detection system using Snort and Moloch and system can be implemented in the large network traffic against SQLIA where thousands of system is communicated rapidly to web application based architecture. Sqlmap is used for the code injecting process and our methodology is to find out the successful SQL injection against these malicious code. This system use the combination of existing technology for analysis and collects the information automatically in real time. It is signature-based system which has some limitations at the time of detection. Further we are going to enhance the proposed methodology to make a hybrid model for auto detection of cross site scripting (XSS) attack for poorly coded web applications in real time.

ACKNOWLEDGMENTS

I heartiest thanks to my mentor **Prof. Nanhay Singh** for guiding me towards the right path. I also thanks to faculty members from department of computer science & engineering, AIACT&R for their valuable support and inputs.

REFERENCES

1. N. Singh, M. Dayal, R. S. Raw,and S. Kumar, "SQL injection: Types, methodology, attack queries and prevention," *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, 2016, pp. 2872-2876.
2. V. K. Gudipati, T. Venna, S. Subburaj, and O. Abuzaghleh, "Advanced automated SQL injection attacks and defensive mechanisms," 2016 Annual Connecticut Conference on Industrial Electronics, Technology & Automation (CT-IETA), Oct. 2016.
3. K. Kamtuo and C. Soomlek, "Machine Learning for SQL injection prevention on server-side scripting," 2016 International Computer Science and Engineering Conference (ICSEC), Dec. 2016.
4. R. P. Karuparthi and B. Zhou, "Enhanced Approach to Detection of SQL Injection Attack," 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), Dec. 2016.
5. R. Dubey and H. Gupta, "SQL filtering: An effective technique to prevent SQL injection attack," 2016 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Sep. 2016.
6. N. A. Al-Sayid and D. Aldlaeen, "Database security threats: A survey study," 2013 5th International Conference on Computer Science and Information Technology, Mar. 2013.
7. A. A. Shastri and P. N. Chatur, "Efficient and effective security model for database specially designed to avoid internal threats," 2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy,and Materials (ICSTM), May 2015.
8. N. A. Al-Sayid and D. Aldlaeen, "Database security threats: A survey study," *2013 5th International Conference on Computer Science and Information Technology*, Amman, 2013, pp. 60-64. doi: 10.1109/CSIT.2013.6588759.
9. P. Ghorbanzadeh, A. Shaddeli, R. Malekzadeh, and Z. Jahanbakhsh, "A survey of mobile database security threats and solutions for it," The 3rd International Conference on Information Sciences and Interaction Sciences, Jun. 2010.
10. A. Sallam, Q. Xiao, E. Bertino, and D. Fadolalkarim, "Anomaly Detection Techniques for Database Protection Against Insider Threats (Invited Paper)," 2016 IEEE 17th International Conference on Information Reuse and Integration (IRI), Jul. 2016.
11. S. Fatih; KOÇAK, "A second pre-image attack and a collision attack to cryptographic hash function lux," Communications Faculty Of Science University of Ankara Series A1Mathematics and Statistics, vol. 66, no. 1, pp. 254–266, 2017.
12. F. Mouton, M. M. Malan, L. Leenen, and H. S. Venter, "Social engineering attack framework," 2014 Information Security for South Africa, Aug. 2014.

13. H. Orman, "Blockchain: the Emperors New PKI?" IEEE Internet Computing, vol. 22, no. 2, pp. 23–28, Mar. 2018.
14. H. Meyer, "A computer emergency response team policy," Computers & Security, vol. 15, no. 4, p. 320, Jan. 1996.
15. J. Clarke, "Exploiting SQL Injection," SQL Injection Attacks and Defense, pp. 137–218, 2009.
16. E. Pollack, "Protecting Against SQL Injection," Dynamic SQL, pp. 31–60, Dec. 2018.
17. J. Uramova, P. Segec, M. Moravcik, J. Papan, T. Mokos, and M. Brodec, "Packet capture infrastructure based on Moloch," 2017 15th International Conference on Emerging eLearning Technologies and Applications (ICETA), Oct. 2017.
18. Z. Zhou, Chen Zhongwen, Zhou Tiecheng, and Guan Xiaohui, "The study on network intrusion detection system of Snort," 2010 International Conference on Networking and Digital Society, May 2010.
19. Martin Roesch, "Snort - Lightweight Intrusion Detection for Networks," In *Proceedings of the 13th USENIX conference on System administration* (LISA '99). USENIX Association, Berkeley, CA, USA, 229-238, 1999.
20. S.-D. AXINTE, "SQL Injection Testing in Web Applications Using SQLmap," International Journal of Information Security and Cybercrime, vol. 3, no. 2, pp. 61–68, Dec. 2014.
21. Antunes, N. and M. Vieira, "Defending against Web Application Vulnerabilities." Computer, 2012. 45(2): p. 66-72.
22. (OWASP), "O.W.A.S.P. Top 10 Vulnerabilities."; Available from : https://www.owasp.org/index.php /Top_10 2013.

AUTHORS PROFILE

Sandeep Choudhary, received his B.Tech. degree in computer science and engineering from G.G.S.I.P.U, New Delhi and M.Tech. (P) degree in information security from AIACT&R , Delhi (India). He is having more than 06 yrs. of experience in web API development, cloud solutions and IT security administration. His area of interest includes cyber security, cryptography, cloud computing, IOT and advanced web technologies.

Dr. Nanhay Singh, working as professor and HOD in department of computer science & engineering, AIACT&R, Delhi (India). He received his Ph.D degree in computer science and technology and M.Tech. degree in computer science and engineering from Kurukshetra University, Haryana . He is having more than 20 yrs, of rich experience in teaching and research. His area of research includes data mining, web engineering, and distributed computing, cloud computing, IoT and mobile ad-hoc network.