Preventing SQL-Based Attacks Using Intrusion Detection System

Manju Khari
Ambedkar Institute of Advanced Communication Technologies and Research
New Delhi, India

Anjali Karar
Ambedkar Institute of Advanced Communication Technologies and Research
New Delhi, India

Abstract: With the advancement of technology everyone is using computer and web applications. These web applications can be easily made by using rapid application development environments by developers. But they do not consider security aspect necessary in the process of providing attractive functionalities & also they are not experts in that field. This make web applications vulnerable to several attacks. Among these attacks SQL injection is considered most dangerous vulnerability. This paper describes various approaches used by authors to prevent SQL injection attack using various methods like intrusion detection, black box testing etc.

Keywords: SQL injection, Intrusion detection, anomaly detection, misuse detection

1. INTRODUCTION

Web applications are very popular today. The developers of the web applications are focused more towards adding the best possible functionalities. In this due course they often neglect security issues. Many tools have been developed to detect Web application vulnerabilities but hackers are still successfully exploiting Web applications. A possible reason is that most tools just scan Web application vulnerabilities, but few tools can automatically revise these vulnerabilities [5].

The user send request via a web browser in the form of URL, that is converted into an IP address, to the web server. Web server converts these requests to SQL commands. The result of these commands generate the response for user for the final presentation. Some rapid application development environments lead to good functionalities with vulnerabilities. These developers are not the security experts so they cannot provide security measures [1].

Nowadays, web applications are vulnerable to many attacks and injecting commands is in the top of this list [2]. Traditional network-based firewall systems offer no protection against these attacks, as the malicious (fractions of) SQL or tampered requests are located at the application layer and thus are not visible to most of these systems [3].Web-based applications are implemented using a number of server-side executable components, such as CGI programs and HTML-embedded scripting code, that access back-end systems, such as databases [4].Existing prevention systems are often insufficient to protect this class of applications, because the security mechanisms provided are either not well understood or simply disabled by the web developers "to get the job done [4]." In SQL, character constants are surrounded by apostrophes ('), semicolons (;) usually separate statements, and (--) start of comment, so the mischievous inputs will usually include at least one of those characters [6]. Typical Intrusion detection systems can prevent the use of some common malicious strings like "union", "or 1=1" [1]. Moreover they are considered as authorized users commands in databases security measures. Some SQL based attacks define earlier in [4] are as follows:

- **SQL injection**: Typing SQL keywords and control signs an intruder is able to change the structure of SQL query developed by a Web designer. Here the structure of SQL query is changed by the attacker. A query looks like:

```
uname = getAuthenticatedUser()
cctype = getUserInput()
result = sql("SELECT nb FROM creditcards
WHERE user='"
+ uname + "' AND type='" + cctype +"';")
print(result)
```

The changed query structure will be executed and database will be affected as the attacker requires.

- **Cross site scripting (XSS):** The attacker injects a script and tries to destroy relationship between web browser and web server. This script is not controlled by attacker once injected. They focus on stealing information that is

- sensitive for user like credit card details. Malicious JavaScript programs can take advantage of the fact that they are executed in a foreign environment that contains sensitive information [4].

- **Other data centric attacks:** This class focuses on particular actions taken by attacker on other query constants. If we see xxx usertype then it is considered as attack. Two-step SQL injection attack also comes under this class of attack. Here attacker inserts or deletes a string from the database. The web site periodically deletes inactive users with the following script [4]:

```
old = now() - 3 months
users = sql("SELECT uname FROM users
WHERE last_login < "+old+";")
for u in users:
sql("DELETE FROM users WHERE uname=’" +
u
```

These are some attacks which are SQL based. In this paper we are going to discuss work carried by authors for preventing malicious SQL injection attacks in different years and best possible to our knowledge we have mentioned some drawbacks. Several intrusion detection techniques are introduced along with them we will also discuss other techniques used to prevent these attacks.

2. LITERATURE SURVEY

Fonseca et al. [1] proposed an intrusion detection system (IDS) at database level. This IDS is based on anomaly detection technique and detects the database operations that are malicious. According to them it is best to place a layer at database level which will be an additional layer for intrusion detection. The purpose served here will be the detection of insider attacks and malicious SQL attacks. They had done an offline analysis. The proposed architecture in [1] is:

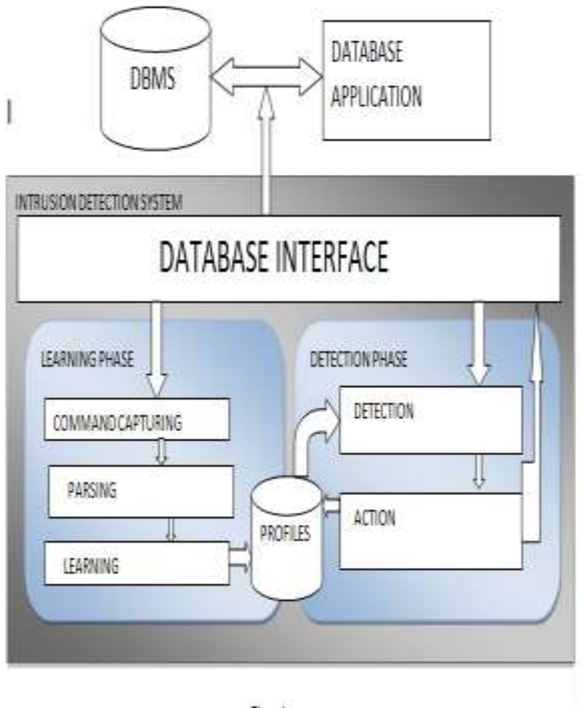
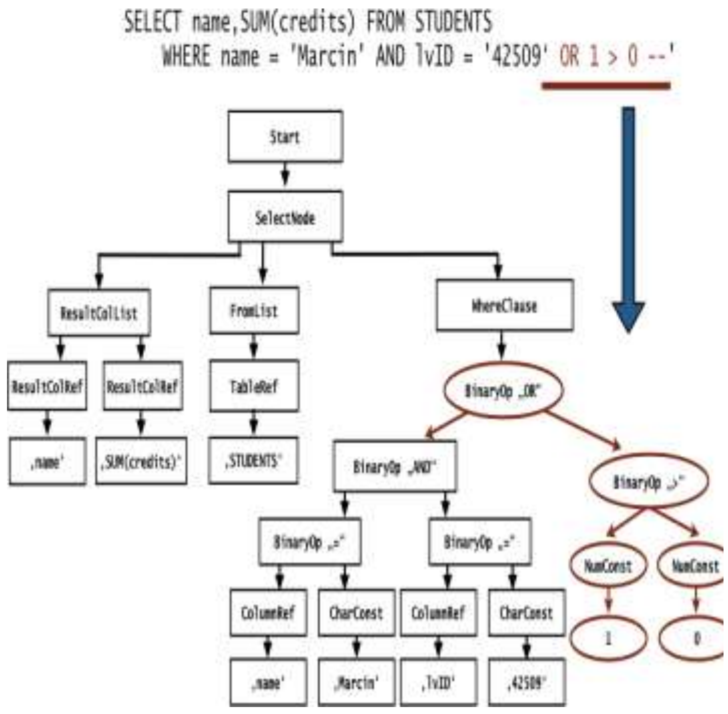


Figure 1

Bockermann et al. [3] proposed technique that used machine learning algorithm ‘internal self organizing maps’ to detect the malicious behavior. The approach incorporates the parse tree structure of SQL queries as characteristic e.g. for correlating SQL queries with applications and distinguishing benign and malicious queries [3]. This paper followed the approach used in [7]:



```
SELECT name,SUM(credits) FROM STUDENTS
    WHERE name = 'Marcin' AND lvID = '42509' OR 1 > 0 --'
```

Figure.2

If the query matches the above diagrammatic representation then it is normal behavior else it will be considered malicious behavior. They used Apache Derby database & generated off a grammar file using tools javacc giving description of grammar. They collected data of the popular Typo3 content management system, create a set of different queries & added attacks that closely relate to SQL injections.

The detection rate was considered as TPR and false positives as FPR. Tree-Kernels were used for SQL grammars & query analysis. Typo3 were represented by dots and malicious modifications as squares. This resulted into highly structured query language, high detection rates and speed.

The author mentioned the drawback here as computational-overhead as computation of kernel matrix took a few more time than necessary. This was due to use of tree-kernel approach.

Pinzon et al. [8] used case based reasoning (CBR) engine which is collaboration of advanced algorithms that can easily allow classification of malicious codes. The agent used here is named as CBRid4SQL. Here combination of CBR system, artificial neural network (ANN) & support vector machine (SVM) gave advantage of learning and adaption and query detection ion best possible way.

CBR related SQL query has three steps namely ; problem description, solution for performing some action & final state after solution. The steps retrieval, reuse, revise, retain were steps followed where main learning phase was completed and machine learning algorithm were used.

Method		**Method**		**Method**	
BayesNet	638	Naive Bayes	666	AdaBoostM1	665
Bagging	684	DecisionStump	598	J48	689
JRIP	692	LMT	693	Logistic	688
LogitBoost	680	MultiBoostAB	666	OneR	622
SMO	685	Stacking	437	CBRid4SQL	**698**

Table 1. Performance of different classifiers

This table 1 shows that the highest-performance system is CBRid4SQL, which has a success rate of 698/705 [8]. The proposed agent is capable of low error rates compared to other existing systems of that time, robustness, decision mechanism and flexibility in queries review.

Valeur et al. [4] focused on mimicry based SQL attacks by developing anomaly based system. The tool can be deployed on a host that contains custom-developed server-side programs and are able to automatically derive models of the manner in which these programs access a back-end database [4]. Here profiles for normal databases access are developed and models are obtained during training phase. The anomalies are detected by the help of profiles made earlier in detection phase.

They used several models in order to characterize the normal behavior of web applications for mimicry attacks. Training phase is divided into data feeding of models for profile building process and anomaly score calculation. If an anomaly score exceeds the maximum anomaly score seen during training by a certain tunable percentage, the query is considered anomalous and an alert is generated. This anomaly based detector had less false positives and little overhead.

Skaruz et al. [9] used recurrent neural network (RNN) which was trained by back propagation time algorithm (BPTT). They divided SQL queries statements into tokens. In training phase, activations of all neurons are computed. Next, an error of each neuron is calculated. These steps are repeated until last token has been presented to the network [9].

They took different tokens and defined their indexes in table 2 [9] as follows:

They used datasets as DATASET I AND DATASET II.
The second dataset showed the scope of use of re-

evaluated data. This division of statements into tokens led to clear line of distinction between an attack and a authorized statement.

Skaruz et al. [10] in 2010, used neural networks to detect SQL attacks and gene expression programming (GEP).jut like they did earlier this time also they divided SQL problem to time series prediction and classification problem. The statement of SQl were again divided into tokens and RNN was used and trained by BPTT.

Each data subset had a corresponding network. The trained network was examined for both attacks and normal SQL queries. They evaluated 2 coefficients that were used as threshold for RNN output. They used two approaches that is RNN and GEP for detection of SQL based attacks. RNN with classification rules is able to predict sequences of 10 tokens with false alarms rate below 1%. We also showed how the number of SQL queries used for setting the coefficients affects the number of false alarms. Classification accuracy received from GEP depicts great efficiency for SQL queries constituted from 10 to 15 tokens. For longer statements the averaged FP and FN equals to about 23%. [10]

Lee et al. [11] provided a framework and named it as DIDAFIT (detecting intrusions through fingerprinting transactions). This system consists of known fingerprints that are compared to every database access and hence fate it as intrusive or normal activity. DIDAFIT is a database intrusion detection system that identifies anomalous database accesses by matching database transactions with a set of legitimate transaction fingerprints. This is database IDS at application level. It can also be be classified as misuse –signature based IDS. This technique deals best with incomplete training datasets and has lower false negatives rate.
This framework deduces the missed SQL fingerprints, the statements were easily converted into fingerprints, high risk SQL statements were detected in training sets.

The framework of proposed system is as follows:

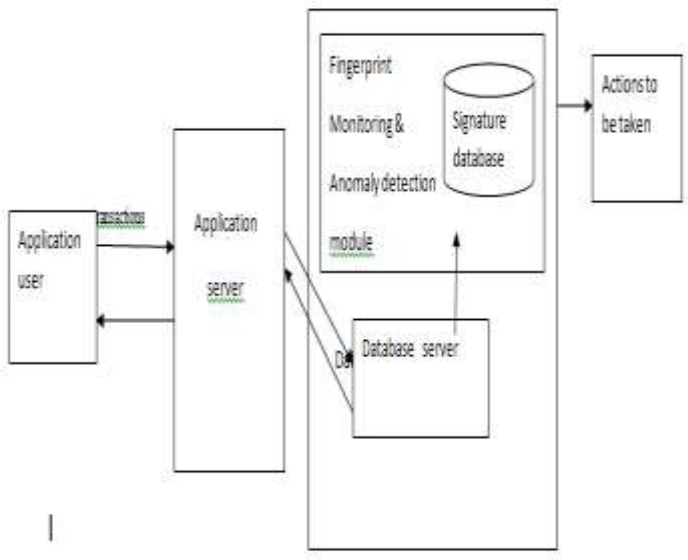


Figure 3

But not to forget that every misuse anomaly detection has drawback that it cannot detect new attacks as no existing fingerprint will match the attack and system will be left vulnerable.
Kiani et al. [12], used an anomaly detection approach. They called the model as same character comparison (SCC) model where HTTP request's section were divided on the basis of character. It follows the approach of FCD (frequency character distribution) model and tried to overcome its limitation. For example, given the extracted query section 'id=444', the frequency count for the characters would be 3 for the character '4', and the frequency count would be 1 for the characters 'i', 'd', '=', and zero for all other characters. Here query section is taken from HTTP requests directly [12].
In training phase frequency is evaluated. The cumulative characteristic count is calculated after all requests are processed. Expected values are evaluated then. In testing phase, anomaly score is calculated using Chi-square test and threshold is determined. This threshold decides whether SQl query is a intrusive one or not. If anomaly score is above the threshold defined alert is triggered.
The approach operates by parsing the query section of HTTP requests and creates profiles for each file. It requires no access to the source code, or modification of existing software modules [12]. Moreover large training datasets were used. Here we got reduced false alerts, no user interaction and UNION attacks and tautology attacks were detected.

Aswami et al [13] proposed the architecture given below:

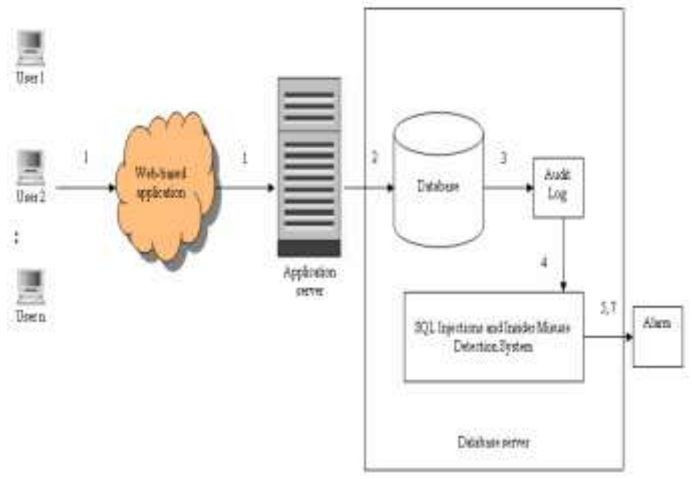


Figure.4

This paper proposes a system that will detect both insider attack and SQL injection attack. It is names as SIIDMS (SQL injection and insider misuse detection system) architecture. This paper has presented a description on the threats in database security and the intrusions from both external and internal attacks against database systems [13]. This is because in many instances, the insiders do have authorized access to their database system but often misuse their rights.

Razzaq et al. [14] introduced the defense mechanism for application level. They provided multi layered defense mechanism capable of detecting both classes of known and unknown attacks. They showed results for high detection rate and low false positives using graphical representation. They focused on XSS attacks and SQL injections. System is evaluated against the existing data mining techniques, attribute length, character distribution, or inference structure used by different models in anomaly detection [14]. First layer, filter out the special tags from malicious input through Filter. Second layer, *Detection module* detects malicious input through positive, negative and anomaly components and lastly syntactical and semantically validation through Analyzer & Validation module [14].

Ciampa et al. [15] proposed a tool named V1p3R.Unlike other exsisting tools it didn't generate SQL queries rather it performed penetration testing. The proposed approach worked on following steps for a web application:

- It determined hyperlinks structure & its input forms
- It was seeded with already known SQL attacks for reporting error(Standard attacks consist in a set of query strings that are not dependent on the Web application.[15])
- Then every access to web application is compared to regular expressions in the database related to error messages.
- It continues the attack using text mined from the error messages with the objective of identifying likely table of field names, until it is able to retrieve (part of) the database structure[15].

This approach worked out for 12 real web applications in different fields.

Boyd et al. [16] used one of the Instruction-Set Randomization application. To create complications for attacker, the SQL standard keywords are appended with a random integer. Therefore, any malicious user attempting an SQL injection attack would be thwarted, for the user input inserted into the "randomized" query would always be classified as a set of non-keywords, resulting in an invalid expression.

They called it SQLrand system and its architecture is given as:

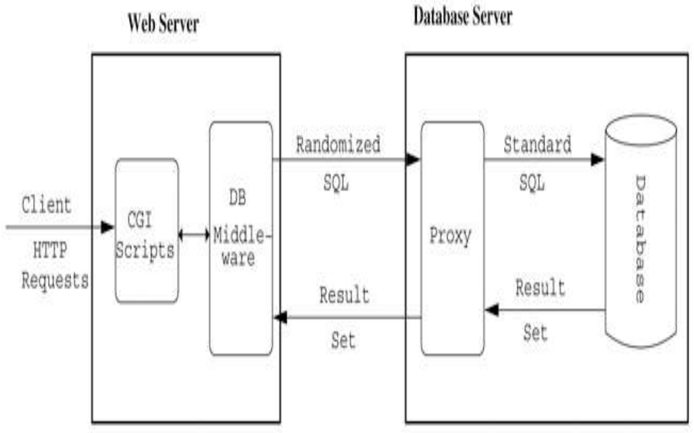


Figure.5

3. CONCLUSIONS AND FUTURE WORK

This work is carried out for details in intrusion detection and SQL based attacks. This result will help for database and IDS work together. This contains work since 2002 to 2011 with some drawbacks and advantages suggested. Including SQL injection we have discussed about some XSS attacks and mimicry attacks. This paper will help people looking forward to perform research work in IDS and SQl based attacks field.

4. REFERENCES

[1] José Fonseca, Marco Vieira, and Henrique Madeira, "Detecting malicious SQL," ESTG-ISUC, 2007.

[2] Open web application security project.The toplist of most severe web application vulnerabilities, 2004.

[3] Christian Bockermann, Martin Apel, and Michael Meier, "Learning SQL for Database Intrusion Detection Using Context-Sensitive Modelling(Extended Abstract)," Artificial Intelligence Group, Information Systems and Security Group ,Department of Computer Science Technische Universit¨at Dortmund,2009.

[4] Fredrik Valeur, Darren Mutz, and Giovanni Vigna, "A Learning-Based Approach to the Detection of SQL Attacks," Reliable Software Group, Department of Computer Science, University of California, Santa Barbara , 2005.

[5] Jin-Cherng Lin, Jan-Min Chen and Hsing-Kuo Wong, "An Automatic Meta-revised Mechanism for Anti-malicious Injection," The Dept. of Computer Sci & Eng, Tatung University, Taipei 10451, Taiwan, The Dept. of Information Management, Yu Da College of Business Miaoli 36143,Taiwan, Chung-shan Institute of Science and Technology, 2007.

[6] Orlando Karam and Svetlana Peltsverger, "Teaching with security in mind," Department of Computer Science and Software Engineering School of Computing and Software Engineering Southern Polytechnic State University Marietta, GA 30060, 2009

[7] Lee, S.-Y., Low, W.L., Wong, P.Y.: "Learning fingerprints for a database intrusion detection system." In: Gollmann, D., Karjoth, G., Waidner, M. (eds.) ESORICS 2002. LNCS, vol. 2502, pp. 264–280. Springer, Heidelberg , 2002.

[8] Cristian Pinzón, Álvaro Herrero, Juan F. De Paz, Emilio Corchado, and Javier Bajo, CBRid4SQL: "A CBR Intrusion Detector for SQL Injection Attacks," HAIS 2010, Part II, LNAI 6077, pp. 510–519, 2010. © Springer-Verlag Berlin Heidelberg, 2010.

[9] Jaroslaw Skaruz, Franciszek Seredynski, and Pascal Bouvry, "Tracing SQL attacks via neural networks," PPAM 2007, LNCS 4967, pp. 549–558, 2008._c Springer-Verlag Berlin Heidelberg, 2008.

[10] Jaroslaw Skaruz, Jerzy Pawel Nowacki, Aldona Drabik, Franciszek Seredynski, and Pascal Bouvry, "Soft computing techniques for intrusion detection of SQL based attacks," ACIIDS 2010, Part I, LNAI 5990, pp. 33–42, 2010. c_ Springer-Verlag Berlin Heidelberg 2010.

[11] Sin Yeung Lee, Wai Lup Low, and Pei Yuen Wong, "Learning fingerprints for database Intrusion detection system," ESORICS 2002, LNCS 2502, pp. 264–279, 2002. c_Springer-Verlag Berlin Heidelberg, 2002.

[12] Mehdi Kiani, Andrew Clark and George Mohay, "Evaluation of Anomaly Based Character Distribution Models in the Detection of SQL Injection Attacks," 0-7695-3102-4, 2008 IEEE, 2008.

[13] Aziah Asmawi, Zailani Mohamed Sidek and Shukor Abd Razak, "System Architecture for SQL Injection and Insider Misuse Detection System for DBMS," Faculty of Computer Science and Information System Universiti Teknologi Malaysia 978-1-4244-2328-6, IEEE, 2008.

[14] Abdul Razzaq, Ali Hur, Nasir Haider, Farooq Ahmad, "Multi-Layered Defense against Web Application Attacks," NUST School of Electrical Engineering and Computer Sciences, Pakistan, 978-0-7695-3596-8, © 2009 IEEE DOI 10.1109/ITNG.2009.77, 2009.

[15] Angelo Ciampa, Corrado Aaron Visaggio and Massimiliano Di Penta, "A heuristic-based approach for detecting SQL-injection vulnerabilities in Web applications," ACM, 978-1-60558-965-7, 2010.

[16] StephenW. Boyd and Angelos D. Keromytis, SQLrand: "Preventing SQL Injection Attacks," Springer-Verlag Berlin Heidelberg, 2004.