IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Detect SQL Injection Queries in Web Applications Using Dynamic Tainting Method

ALLU GANESH KUMAR [#1], BODALA SAI LALITHA [#2]

[#1] Assistant Professor, Department of MCA,
Sanketika Vidhya Parishad Engineering College, P.M. Palem,
Visakhapatnam, Andhra Pradesh.

[#2] MCA Student, Department of MCA,
Sanketika Vidhya Parishad Engineering College, P.M. Palem,
Visakhapatnam, Andhra Pradesh.

ABSTRACT

Data is one of the most valuable assets for modern small and big size commercial firms, and this is crucial for the growth and development of each individual. As we are all aware, there are several attackers that attempt to develop assaults on the growing amount of data by introducing certain qualities. One of the many assaults is the SQL injection attack, which is listed as the top network risk by the Open Web Application Security Project (OWASP). In this research, we attempt to develop a dynamic tainting-based SQL injection detection system that can effectively identify intrusion attempts to launch SQL assaults against the dataset and attempt to contrast it with several established detection techniques. This suggested approach attempts to detect the kind of query by first taking a sample dataset as input and then producing SQL queries from that test dataset.

KEYWORDS:

SQL Injection, Security Project, Web Application, Intrusion Detection, Privacy.

1. INTRODUCTION

The majority of research over the years has identified developers' lack of security awareness in web development to sanitised input as the root cause of SQLIA, and as a result, has moved towards code-based sanitation for their suggested remedies to solve SQLIA.

Additionally, the well-intentioned free text processing of the SQL engine itself results in the SQLIA vulnerability, which makes both legacy and cloud deployments deficient in sanitation susceptible. The ability to defend back-end databases from SQLIA in an era of large data remains a pressing concern. A search of the SQL Hall of Shame [1], which details the latest trends in data pilfering by SQLIA, reveals the ubiquity of this sort of assault. The SQLIA keywords are also in plain text, and the grammar of the SQL language is very similar to that of plain English [2]. Because of this, the SQLIA problem in a big data setting is a likely candidate for predictive analytics using a supervised learning model that was trained using both safe web request patterns and known historical attack signatures.

While legitimate web requests would take the form of anticipated data from the application, attack signatures at injection sites would contain patterns of SQL tokens and symbols as SQLIA positive. In this study, we develop a web application for predictive analytics using a large amount of learning data to train a classifier. The learning data consist of labelled vector matrices, or features of dictionary word list patterns (SQLIA negative) and SQL tokens (SQLIA positive). In order to train a supervised learning model using the Support Vector Machine (SVM) algorithm that accurately predicts SQLIA and prevents malicious web requests from reaching the target back-end database, the contributions made in this paper provide a representative data set that undergo feature hashing. Additionally, it provides context for SQLIA detection and avoidance in big data internet.

2. LITERATURE SURVEY

Literature survey is that the most vital step in software development process. Before developing the new application or model, it's necessary to work out the time factor, economy and company strength. Once all these factors are confirmed and got an approval then we can start building the application.

MOTIVATION

1) SQL Injection Detection for Web Applications Based on Elastic-Pooling CNN

Authors: XIN XIE and CHUNHUI REN

Data may be one of a company's most valuable assets and is frequently essential to its growth and survival. According to the Open Web Application Security Project, the initial danger to network applications is a SQL injection attack (OWASP). Its negative effects, universality, and dire condition are obvious. This study compares Elastic-Pooling CNN (EP-CNN)-based SQL injection detection techniques with more established ones. This technique efficiently identifies the SQL injection of web applications and can generate a fixed two-dimensional matrix without truncating data. It is more difficult to get around and can identify new assaults based on the irregular matching features.

2) GMSA: Gathering Multiple Signatures Approach to Defend Against Code Injection Attacks

Authors: HUSSEIN ALNABULSI

Invalid code processing results in security flaws and software problems, which are the targets of code injection attacks (CIAs). Hackers try to include the CIA into each new technique in an effort to get beyond the security mechanism. In this study, we provide GMSA, a tool designed to identify a number of CIAs, including file inclusion attacks, shell injection attacks, SQL injection attacks, and cross-site scripting (XSS) attacks. The latter includes both local and distant file inclusion. According to our empirical investigation, the gathering multiple signatures method (GMSA) performs a precision performance when compared to previous research (the suggested algorithm's accuracy is 99.45%). GMSA has a low false positive rate (FPR) of 0.59% when compared to other has reported.

3) SQL Injection Detection for Web Applications Based on Elastic-Pooling CNN**.**

Authors: Chunhui Ren

Data may be one of a company's most valuable assets and is frequently essential to its growth and survival. According to the Open Web Application Security Project, the initial danger to network applications is a SQL injection attack (OWASP). Its negative effects, universality, and dire condition are obvious. This study compares Elastic-Pooling CNN (EP-CNN)-based SQL injection detection techniques with more established ones. This technique efficiently identifies the SQL injection of web applications and can generate a fixed two-dimensional matrix without truncating data. It is more difficult to get around and can identify new assaults based on the irregular matching features.

3. EXISTING SYSTEM AND ITS LIMITATIONS

Today's online apps can be accessible through the Internet using any Web browser, regardless of operating system or architecture. Because of their simplicity, adaptability, accessibility, and interoperability, they have spread like wildfire. Unfortunately, a number of brand-new security risks can also affect Web apps. Among these dangers, SQL Injection Attacks (SQLIAs) are one of the most serious. Because they may provide attackers full access to the databases that support Web applications, SQLIAs are becoming more prevalent and constitute very significant security dangers. The constraints brought on by this SQLIAs are listed below.

LIMITATION OF PRIMITIVE SYSTEM

The following are the limitations of the existing system.

1) In the existing days there is no automated approach for dynamic detection and prevention of SQLIAs.

2) There is no concept to identify the difference between “Trusted “Strings and Injected Strings.

3) There is no mechanism which can identify the injection of some special operators or keywords during data insertion in a dynamic manner.

4) In the existing days there is no method like “DYNAMIC TAINTING” , which marks and tracks certain data in a program at runtime.

5) All the existing schemes failed to detect the SQL injections for web applications in accurate manner.

4. PROPOSED SYSTEM AND ITS ADVANTAGES

In this scenario, the user has two options for searching SQL queries: one in which the queries contain no special characters or strings, and the other in which the questions contain some special strings. We can easily recognise the sql queries that comprise both false positive and false negative queries by using the dynamic tainting approach. By running several tests on our proposed model, we were able to demonstrate that it is effective and feasible for detecting sql-injection threats over secure connections.

The following are the advantages of the proposed system. They are as follows:

1) In the proposed system we try to design an automated approach for dynamic detection and prevention of SQLIAs.
2) There is a advanced concept to identify the difference between “Trusted “ Strings and Injected Strings.
3) In this proposed mechanism we can able to identify the injection of some special operators or keywords during data insertion in a dynamic manner.
4) We try to develop “DYNAMIC TAINTING” , which marks and tracks certain data in a program at runtime.
5) The proposed method utilizes the facility to detect the SQL injections for web applications in accurate and efficient manner.

5. IMPLEMENTATION PHASE

Implementation is the stage where the theoretical design is converted into programmatically manner. In this stage we will divide the application into a number of modules and then coded for deployment. The front end of the application takes JSP,HTML and CSS and as a Back-End Data base we took SQL Injection dataset. Here we are using Java as Programming Language to Implement the current application. The application is divided mainly into following 3 modules. They are as follows:

1) User Module
2) Admin Module
3) Data Storage Module

1) User Module

The user here is general web browser how retrieves the information from web where it was connected to a backend database. The user doesn’t need any login access. He simply fires the query to get the necessary information from that particular domain .

2) Admin Module

Admin the one who responsible for loading the data into the database and maintain the database from identifying SQL injection attacks that cannot be recognized by traditional methods. And also use as an auxiliary method of traditional SQL detection methods.Admin also has an login access to database

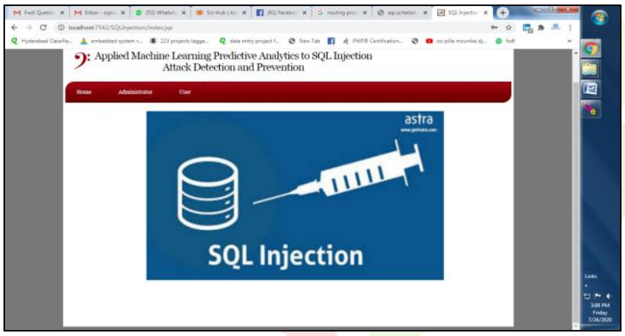
3) Data Storage Module

In this project we don't use any database tables for storing and retrieving the records. Instead of physical tables we try to load the dataset which is required to access the data to and from the application. Here we try to load a SQL dataset as input and try to pre-process the dataset using WEKA tool and then check the type of query from that loaded dataset**.**

6. EXPERIMENTAL RESULTS

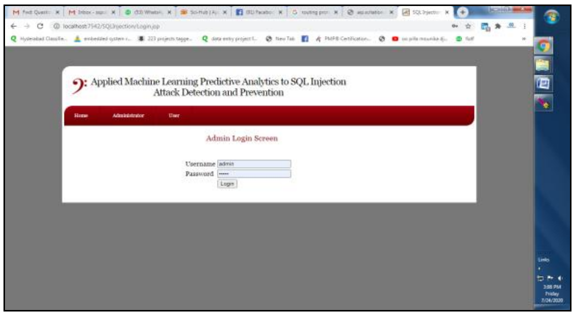
In this section we try to design our current model using Java as programming language and we used J2EE as working environment for executing the application. Now we can check the performance of our proposed application as follows:

MAIN PAGE


Applied Machine Learning Predictive Analytics to SQL Injection
Attack Detection and Prevention
Home
Administrator
User
astra
SQL Injection

The above window clearly represent the main page of our proposed application.

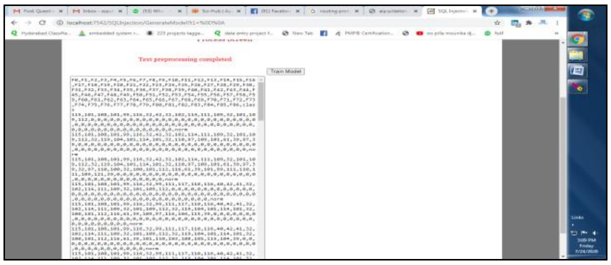
ADMIN LOGIN PAGE


Applied Machine Learning Predictive Analytics to SQL Injection
Attack Detection and Prevention
Home
Administrator
User
Admin Login Screen
Username
Password
Login

Username: admin

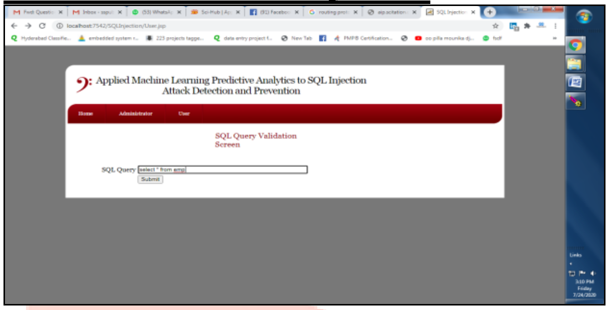
From the above window we can see admin is trying to enter his account with his valid username and password.

ADMIN UPLOAD THE DATASET AND PRE-PROCESS

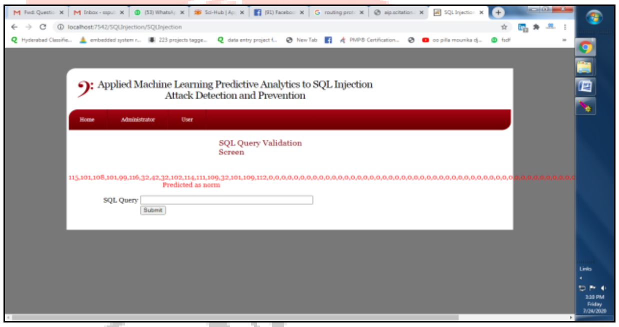


From the above window we can clearly see the pre-processing of input dataset is done successfully.

USER SEARCH FOR QUERY


Applied Machine Learning Predictive Analytics to SQL Injection
Attack Detection and Prevention
Home
Administrator
User
SQL Query Validation
Screen
Submit

SQL Query: select * from emp

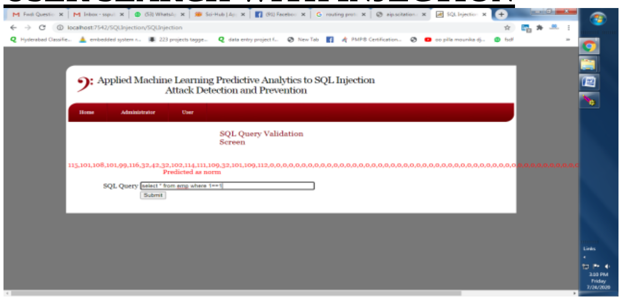

Applied Machine Learning Predictive Analytics to SQL Injection
Attack Detection and Prevention
Home
Administrator
User
SQL Query Validation
Screen
Submit

115,101,108,101,99,116,32,42,32,102,114,111,109,32,101,109,112,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
Predicted as norm

SQL Query

From the above window we can clearly see the user search query is normal and successful.

USER SEARCH WITH INJECTION


Applied Machine Learning Predictive Analytics to SQL Injection
Attack Detection and Prevention
Home
Administrator
User
SQL Query Validation
Screen
Submit

115,101,108,101,99,116,32,42,32,102,114,111,109,32,101,109,112,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
Predicted as norm

SQL Query: select * from emp where 1==1

From the above window we can clearly see the user search query is **ABNORMAL**.

7. CONCLUSION

In this study, using a dynamic tainting technique, we create and use predictive analytics for the first time for SQLIA detection and prevention in a big data scenario. This method is generally the best for dynamically recognising false positive and false negative data, and this is the first time that SQL queries have been run to find any abnormalities that may be present. The simulation results clearly show that our programme is the best at accurately differentiating between regular queries and sql injection queries, and that it will classify them into two separate lists that seem to be absent in all other enterprise-level solutions right now.

8. REFERENCES

[1] M. Qbea'h, M. Alshraideh, and K. E. Sabri, ``Detecting and preventing SQL injection attacks: A formal approach," in Proc. Cybersecur. Cyber-forensics Conf. (CCC), Aug. 2016, pp. 123_129.

[2] S. O. Uwagbole, W. J. Buchanan, and L. Fan, ``Applied machine learning predictive analytics to SQL injection attack detection and prevention," in Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM), May 2017,pp. 1087_1090.

[3] P. R. McWhirter, K. Kifayat, Q. Shi, and B. Askwith, ``SQL injection attack classification through the feature extraction of SQL query strings using a gap-weighted string subsequence kernel," J. Inf. Secur. Appl., vol. 40,pp. 199_216, Jun. 2018.

[4] M. Lodeiro-Santiago, C. Caballero-Gil, and P. Caballero-Gil, ``Collaborative SQL-injections detection system with machine learning," in Proc. 1st Int. Conf. Internet Things Mach. Learn., 2017, Art. no. 45.

[5] B. Hanmanthu, B. R. Ram, and P. Niranjan, ``SQL injection attack prevention based on decision tree classi_cation," in Proc. IEEE 9th Int.

[6] J. Santoso, E. M. Yuniarno, and M. Hariadi, ``Large scale text classi_cation using map reduce and Naive Bayes algorithm for domain specified ontology building," in Proc. 7th Int. Conf. Intell. Hum.-Mach. Syst. Cybern.,vol. 1, Aug. 2015, pp. 428_432.

[7] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, ``A survey of the recent architectures of deep convolutional neural networks," 2019,arXiv:1901.06032. [Online]. Available: https://arxiv.org/abs/1901.06032.

[8] The Ten Most Critical Web Application Security Risks, Top OWASP 10, Toronto, ON, Canada, 2013.

[9] N. Singh, M. Dayal, R. S. Raw, and S. Kumar, ``SQL injection: Types, methodology, attack queries and prevention," in Proc. 3rd Int. Conf. Comput. Sustain. Global Develop. (INDIACom), Mar. 2016, pp. 2872_2876.

[10] D. Kar, S. Panigrahi, and S. Sundararajan, ``SQLiGoT: Detecting SQL injection attacks using graph of tokens and SVM," Comput. Secur., vol. 60,pp. 206_225, Jul. 2016.

[11] K. Kamtuo and C. Soomlek, ``Machine Learning for SQL injection prevention on server-side scripting," in Proc. Int. Comput. Sci. Eng. Conf.(ICSEC), Dec. 2016, pp. 1_6.

[12] S. M. Darwish, ``Machine learning approach to detect intruders in database based on hexplet data structure," J. Elect. Syst. Inf. Technol., vol. 3, no. 2,pp. 261_269, 2016.

About the Authors

ALLU GANESH KUMAR is currently working as an Assistant Professor in Department of MCA at Sanketika Vidhya Parishad Engineering College, P.M. Palem, Visakhapatnam, Andhra Pradesh. He has more than 12 years of teaching experience. His research interest includes Java, Python, Cloud Computing, and Deep Learning.

BODALA SAI LALITHA is currently pursuing her 2 years MCA in Department of Computer Science and Applications at Sanketika Vidhya Parishad Engineering College, P.M. Palem, Visakhapatnam, Andhra Pradesh.Her area of interest includes Python, Java, C, C++.