International Journal of Multidisciplinary Engineering in Current Research - IJMEC
Volume 9, Issue 3, March-2024, http://ijmec.com/, ISSN: 2456-4265

SQL INJECTION PREDICTION USING MACHINE LEARNING

Doma Vinay[*1], Sepoor Vinay Goud[*2], Kuruva Deeksha[*3], Mrs.M.Anusha[4]

[*]1,2,3Student, Department Of Electronics And Computer Engineering, JB Institute Of Engineering And Technology ,Moinabad,Telangana, India.

4Assistant Professor, Department Of Electronics And Computer Engineering, JB Institute Of Engineering And Technology Moinabad, Telangana, India.

ABSTRACT: The web is currently the most reliable and popular form of commercial and personal communication. On the web, users load millions of gigabytes of data every day through a variety of routes, and user input might be malevolent. As a result, security becomes a crucial component of web applications. Due to their accessibility, they are vulnerable to several flaws that, if ignored, might be harmful. These gaps are used by the attackers to engage in a variety of illicit operations that allow them to get unauthorized access. One such attack that is simple to carry out but challenging to detect due to its various forms and channels is SQL Injection. This might lead to theft, a data breach, or property loss. The suggested classifier combines a role-based access control system for detection with the Naive Bayes machine learning method. On the basis of test cases drawn from the three SQLIA attacks comments, union, and tautology the suggested model is put to the test.

INTRODUCTION

The internet is a popular platform for many enterprises and everyday transactions around the world. Relational databases, which are accessed through statements written in a unique language known as Structured Query Language (SQL), constitute the foundation of every web-based application. Using unconstrained user input parameters, the attacker injects SQL characters or keywords into a SQL statement to change the original query's logic. This injection technique is used to attack websites. A query is created each time a user-generated request is made. The user input in the query could be harmful. It's crucial to teach our web apps that user input can come from dangerous sources and comes from outside sources.

Therefore, before it is actually executed, we need to process it. The programmer is in charge of creating sophisticated code that thwarts any unauthorized entry. However, because of carelessness or ignorance, the user input is left unprocessed, giving the attacker a way in to the system. Based on the type of user input channel, the server response, how the server responds to the malicious user input, impact point, etc., there are various forms of SQL injection attacks. Some of the fundamental injection kinds include blind SQL injection, union-based assault, tautology attack, and error-based attack. The SQL query is changed in the tautology type so that the criteria always return TRUE. When in union In a query-based attack, the queries are appended with the UNION statement so that one of them executes a harmful function.

DOI: https://doi-ds.org/doilink/04.2024-79646749

METHODOLOGY

1. DATA COLLECTION

Data collection is a process in which information is gathered from numerous sources which is latterly used to develop the machine literacy models. The data should be stored in a way that makes sense for problem. In this step the data set is converted into the accessible format which can be fed into machine literacy models. Data used in this paper is a set of data with features. This step is concerned with opting the subset of all available data that you'll be working with. ML problems start with data rather, lots of data(exemplifications or compliances) for which you formerly know the target answer. Data for which you formerly know the target answer is called labelled data.

2. DATAPRE-PROCESSING

Organize your named data by formatting, drawing and testing from it.

Three common datapre-processing way are

1. Formatting

The data you have named may not be in a format that's suitable for you to work with. The data may be in a relational database and you would like it in a flat train, or the data may be in a personal train format and you would like it in a relational database or a textbook train.

2. Cleaning

Cleaning data is the junking or fixing of missing data. There may be data cases that are deficient and don't carry the data you believe you need to address the problem. These cases may need to be removed. also, there may be sensitive information in some of the attributes and these attributes may need to be anonym zed or removed from the data entirely.

3. Sampling

Sampling There may be far more named data available than you need to work with. further data can affect in much longer running times for algorithms and larger computational and memory conditions. You can take a lower representative sample of the named data that may be important faster for exploring and prototyping results before considering the whole dataset.

3.FUTURE EXTRATION

Next thing is to do point birth is an trait reduction process. Unlike point selection, which ranks the being attributes according to their prophetic significance, point birth actually transforms the attributes. The converted attributes, or features, are direct combinations of the original attributes. Eventually, our models are trained using Classifier algorithm. We use classify module on Natural Language Toolkit library on Python. We use the labelled dataset gathered. The rest of our labelled data will be used to estimate the models. Some machine learning algorithms were used to classifypre-processed data. The chosen classifiers were Random timber. These algorithms are veritably popular in textbook bracket tasks.

4.Model evaluation

DOI: https://doi-ds.org/doilink/04.2024-68892225

Model evaluation serves as a critical checkpoint in the journey of model development, aiding in the selection of the most fitting model for representing data and projecting its future performance accurately. Relying solely on the training data for evaluation is frowned upon in data science due to its tendency to breed overly optimistic and tightly fitting models. In navigating this challenge, two primary methodologies emerge: Hold-Out and Cross-Validation.

The Hold-Out approach involves dividing the available data into two distinct sets: one for training the model and another for testing its performance. Although straightforward and computationally efficient, this method can be sensitive to the random partitioning of data.

On the other hand, Cross-Validation introduces a more intricate process. It partitions the data into multiple subsets or folds, repeatedly training the model on a subset and validating it on the remainder. By iteratively reshuffling the data, Cross-Validation offers a more robust estimate of model performance, particularly beneficial when dealing with limited data.

Crucially, both methods adhere to the principle of utilizing a separate test set unseen during model training. This ensures a realistic evaluation of the model's generalization capability, further fortifying its reliability in real-world applications.

LITERATURE SURVEY

This section briefly describes the exploration and work done so far on detecting SQL Injection attacks and precluding them effectively. The exploration work done so far on detecting SQL Injections can be astronomically classified into two types of approaches. The first approach is to securely write the source law itself to apply enough input confirmation for SQL queries. The alternate approach is to emplace fresh software to corroborate the SQL queries being passed through web operations to be executed across the database. recently some experimenters have also stressed the significance of using these two approaches in combination to achieve a further dependable SQL Injection discovery model. In this section, both the approaches and the exploration work done in those fields is bandied. Goud etal. developed a JDBC checker that's a static analysis tool to check for crimes in SQL strings and corroborate them for implicit vicious queries(3). It verifies the SQL strings forcorrectness and promises to identify and indicate implicit crimes in SQL queries. The way it works is rather of stoutly checking each query while it's generated at runtime, it statically creates a list of all implicit SQL strings that could be executed across a particular operation and also analyses all those implicit SQL strings for vicious content and semantic crimes. The problem with this approach could be multiple, including the high storehouse that will be needed for storing all the implicit SQL queries, and how could a tool induce all possible implicit queries for an operation. There's a huge possibility that it would miss out on prognosticating SQL query statements that were actually executed. To overcome this limitation of static analysis, a tool named CANDID was developed, that stands for ‘ seeker Evaluation for Discovering Intent stoutly ’,(4). It

DOI: https://doi-ds.org/doilink/04.2024-68892225

works on the conception of programmer intent. The conception of programmer intent describes how a SQL query structure should look like if it's formed exactly as was intended by the programmer of the operation. There's a pattern observed in SQL injection attacks that the vicious SQL query executed across database always has a different structure than the bone

that was intended by the programmer. Authors of this paper believed that relating this difference in structure could be a significant step towards successfully relating and precluding SQL injection attacks. The way this tool achieves its thing is that it stoutly creates the programmer intended SQL query structure when the program reaches a position where it'll induce and execute a SQL query. This generated programmer intended SQL query is also compared with the factual SQL query that was passed by the stoner, to identify and help the vicious queries from being executed. latterly a many experimenters came up with the idea of combining the use of static analysis and dynamic monitoring to efficiently help SQL Injection attacks. AMNESIA is a tool that was developed on this idea of using both static and dynamic approaches(5). In this approach, the operation law itself contains information to produce all the possible SQL queries that could be generated by the operation. All these possible licit queries are Generated and stored for comparison. At the same time dynamic monitoring is done of SQL queries generated at runtime, and each stoutly generated SQL query is compared to the list of possible SQLqueries.However, in the list of possible licit SQL queries, the query is classified as vicious and isn't allowed to be executed on the database, If no match is set up for a SQL query generated as a Result of some input from a stoner. This system has its own set of downsides, biggest one being that it isn't a hundred percent accurate and could induce a lot of false cons. Buehrer etal. used analogous approach of comparing the actually generated queries with the bone that should have been generated(Programmer intended).

Proposed System

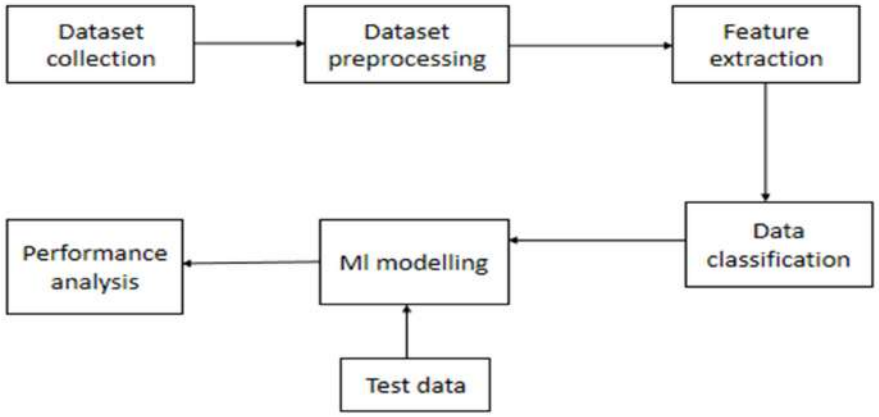
In our forthcoming system, we plan to leverage a mix of machine learning algorithms for classification, with a specific focus on integrating the Naïve Bayes algorithm. The effectiveness of our model will be assessed through extensive testing using test cases derived from three common SQL attack types: comments, union, and tautology

Advantages:

- More accuracy
- Using different algorithms we can get more training data set

System Architecture:

DOI: https://doi-ds.org/doilink/04.2024-68892225



Feasibility study

Feasibility study in the sense it's a practical approach of enforcing the proposed model of system. Then for a machine literacy systems. we generally collect the input from online websites and filter the input data and fantasize them in graphical format and also the data is divided for training and testing. That training is testing data is given to the algorithms to prognosticate the data.

1. First, we take dataset.
2. Sludge dataset according to conditions and produce a new dataset which has trait according to analysis to be done
3. PerformPre-Processing on the dataset
4. Split the data into training and testing
5. Train the model with training data also dissect testing dataset over bracket algorithm
6. Eventually you'll get results as delicacy criteria

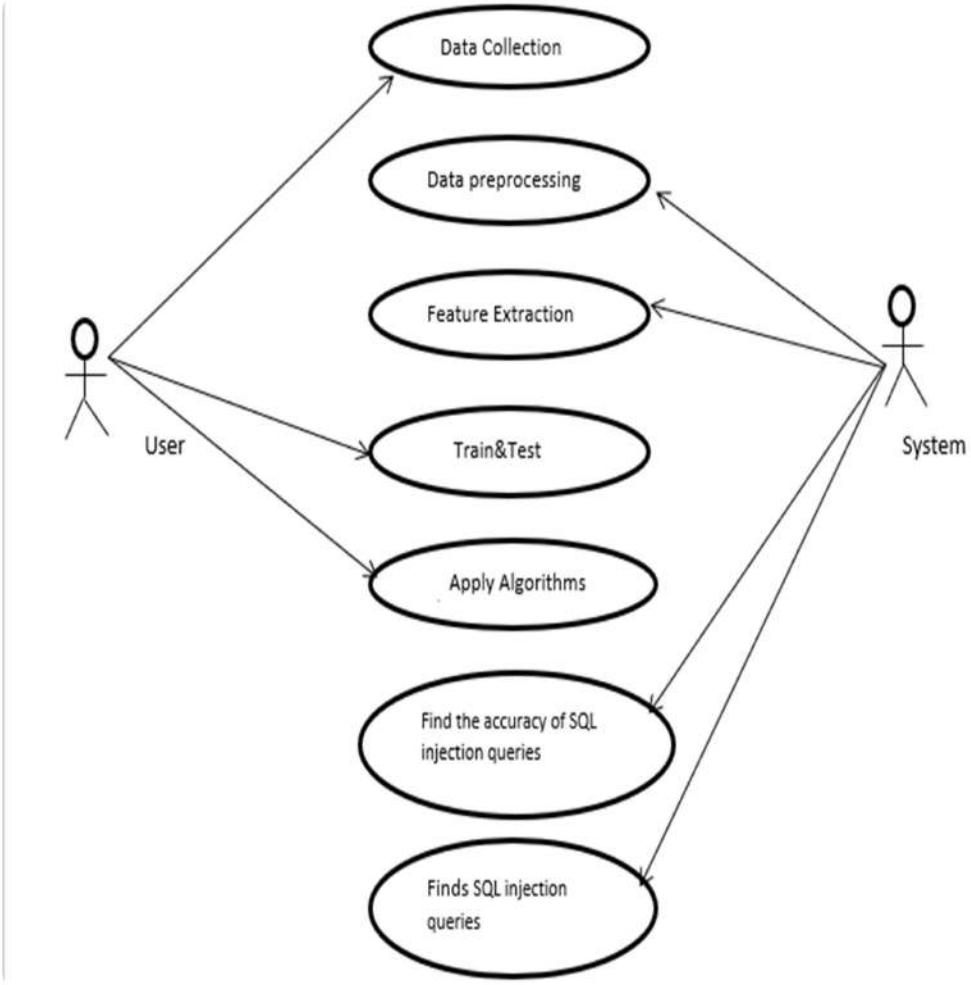
UML Diagram:

- The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the Architecture of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blueprints

Use case diagram:

A use case diagram is a way to summarize details of a system and the users within that system**.** A use case is a methodology used in system analysis to identify, clarify and organize system requirements

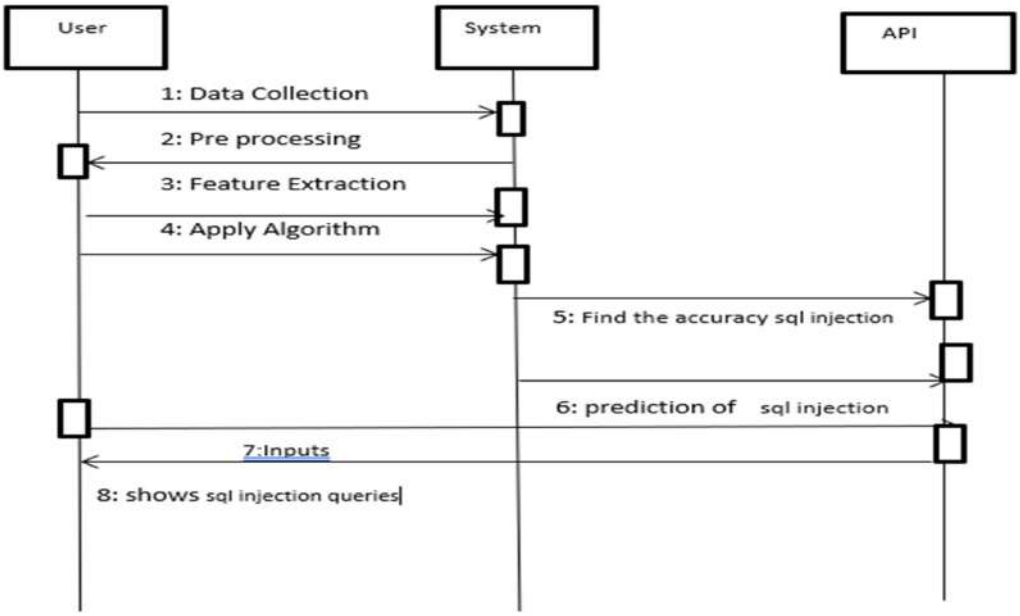
DOI: https://doi-ds.org/doilink/04.2024-68892225



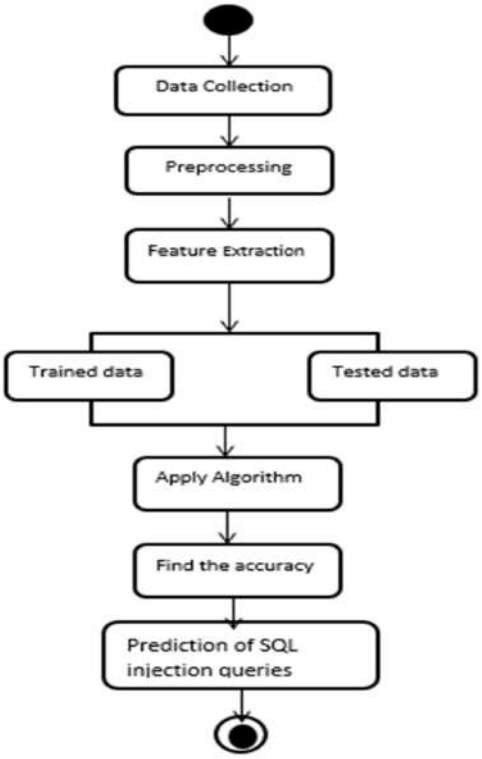
Sequence Diagram:

Sequence Diagram Represent the objects sharing the commerce horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a protestation of an offered geste Each use case specifies some geste of the subject without reference to its internal structure. These conduct, involving relations between the actor and the subject, may affect in changes to the state of the subject and dispatches with its terrain. A use case can include possible variations of its introductory geste and error running.

DOI: https://doi-ds.org/doilink/04.2024-68892225



ACTIVITY Diagram



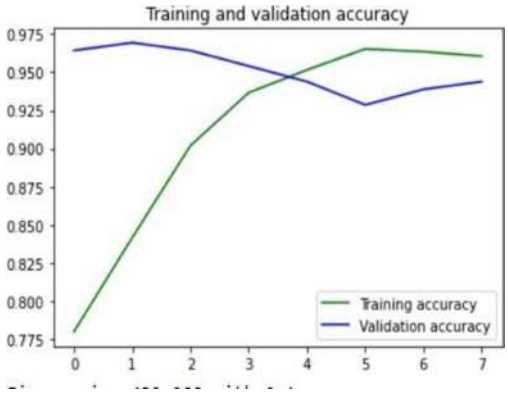
DOI: https://doi-ds.org/doilink/04.2024-68892225

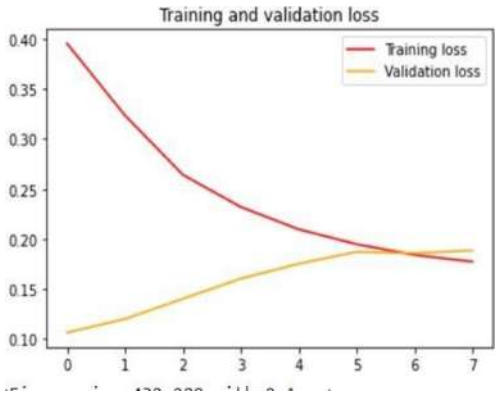
RESULTS AND DISCUSSION

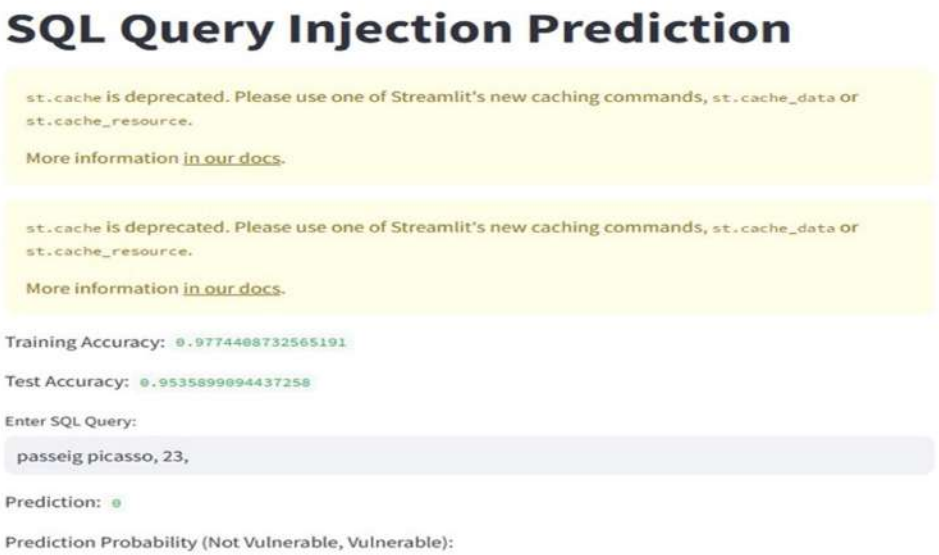
We implemented and evaluated the classifier for SQLIAs detection using the machine learning Naive-Bayes Algorithm. In the learning process, the training dataset is read by the application from text files and puts each data to the learning method of the classifier. The classifier generates feature vectors from received data by blank separation and tokenizing method and learns it by machine learning method. The feature vector also includes role of the user which is used for classification based on Role Based Access Control mechanism. In the classification process, the application reads the test dataset from text files and puts each data to the classification method of the classifier. From the generated feature vector

classification is done. The classification results of the Naïve Bayes machine learning approach is analyzed by Precision and Recall in the application.

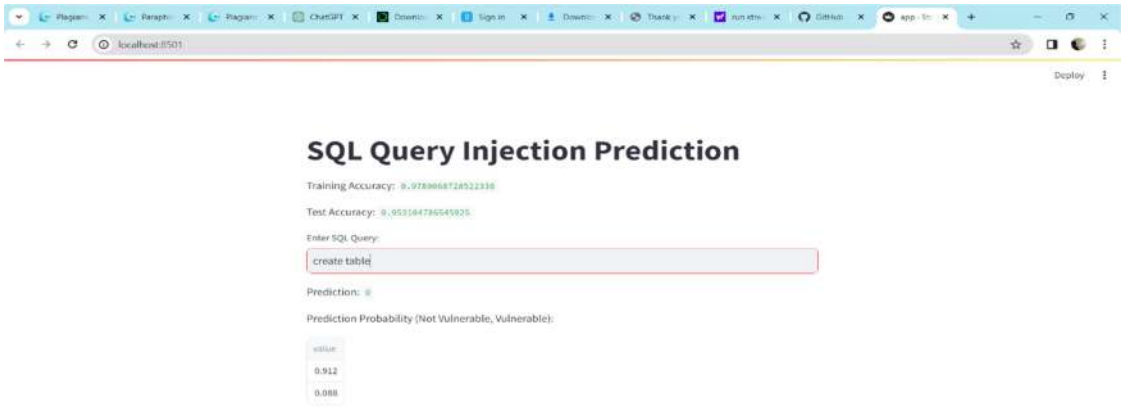
Output:







DOI: https://doi-ds.org/doilink/04.2024-68892225



CONCLUSION

The SQL Injection Prediction project has provided valuable insights into the detection and prevention of SQL injection attacks. The findings of this project can have significant implications for enhancing the security of web applications. Key conclusions from the project include:

- **Machine Learning Models:**

  Machine learning models can effectively predict SQL injection attacks with high accuracy and can be integrated into existing security systems.

- **Feature Importance:**

  Certain features, such as the length of input, presence of special characters, and the number of SQL keywords

- **False Positives:**

  While machine learning models can accurately predict SQL injection attacks, they may also generate false positives

REFERENCES

DOI: https://doi-ds.org/doilink/04.2024-68892225

1. J. Abirami, R. Devakunchari and C. Valliyammai, "A top web security vulnerability SQL injection attack — Survey," 2015 Seventh International Conference on Advanced Computing (ICoAC), Chennai, 2015, pp. 1-9.

2. Diallo, A. K., Al-sakib, K. P.: A survey on SQL injection:vulnerabilities, attacks, and prevention techniques.2011. Retrieved from http://irep.iium.ed u.my/769/1/ISCE2011_paper323.pdf and accessed on 10th June, 2017.

3. C. Gould, Zhendong Su and P. Devanbu, "JDBC checker: a static analysis tool for SQL/JDBC applications," Proceedings. 26th International Conference on Software Engineering, Edinburgh, UK, 2004, pp. 697-698.

4. Prithvi Bisht, P. Madhusudan, and V. N. Venkatakrishnan, "CANDID: Dynamic candidate evaluations for automatic prevention of SQL injection attacks", ACM Transactions on Information and System Security (TISSEC), v.13 n.2, p.1-39, February 2010.

5. William G. J. Halfond , Alessandro Orso, "AMNESIA: analysis and monitoring for NEutralizing SQL-injection attacks", Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering, November 07-11, 2005.

6. Gregory Buehrer , Bruce W. Weide , Paolo A. G. Sivilotti, "Using parse tree validation to prevent SQL injection attacks", Proceedings of the 5th international workshop on Software engineering and middleware, September 05-06, 2005

7. A. Joshi and V. Geetha, "SQL Injection detection using machine learning," 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), Kanyakumari, 2014, pp. 1111-1115.

8. William GJ. Halfond and Alessandro Orso," Preventing SQL Injection Attacks Using AMNESIA" ICSE'06, May 20-28, 2006, Shanghai, China ACM 06/0005.

9. Takeshi Matsuda, Daiki Koizumi, Michio Sonoda, Shigeichi Hirasai, "On predictive errors of SQL injection attack detection by the feature of the single character" SQL Injection Detection Using Machine Learning 42 Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on 9-12 Oct 2011, On Page 1722-1727.

10. Angelo Ciampa, Corrado Aaron Visaggio, Massimiliano Di Penta :"A heuristic-based approach for detecting SQL-injection vulnerabilities in Web applications". 11. P. Joshi, Dissecting Bias vs. Variance Tradeoff In Machine Learning, 2015 [Online] Available: https://prateekvjoshi.com/2015/10/20/dissecting-bias-vs-variance-tradeoff-in-machine-learning/

12. Xristica, What is the difference between Bagging and Boosting?, 2016 [Online] Available: https://quantdare.com/what-is-the-difference-between-bagging-and- boosting/

13. Nick Galbreath, 'libinjection', 2012. [Online] Available: https://github.com/client9/libinjection.git

14. foospidy/payloads, libinjection_bypasses.txt, 29 June 2007. [Online] Available: https://github.com/foospidy/payloads.git

DOI: https://doi-ds.org/doilink/04.2024-68892225