WebAppShield: an approach exploiting machine learning to detect SQLi attacks in an application layer in run-time

World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:16, No:8, 2022

WebAppShield: An Approach Exploiting Machine Learning to Detect SQLi Attacks in an Application Layer in Run-Time

Ahmed Abdulla Ashlam, Atta Badii, Frederic Stahl

***Abstract*—**In recent years, SQL injection attacks have been identified as being prevalent against web applications. They affect network security and user data, which leads to a considerable loss of money and data every year. This paper presents the use of classification algorithms in machine learning using a method to classify the login data filtering inputs into "SQLi" or "Non-SQLi," thus increasing the reliability and accuracy of results in terms of deciding whether an operation is an attack or a valid operation. A method as a Web-App is developed for auto-generated data replication to provide a twin of the targeted data structure. Shielding against SQLi attacks (WebAppShield) that verifies all users and prevents attackers (SQLi attacks) from entering and or accessing the database, which the machine learning module predicts as "Non-SQLi", has been developed. A special login form has been developed with a special instance of the data validation; this verification process secures the web application from its early stages. The system has been tested and validated, and up to 99% of SQLi attacks have been prevented.

***Keywords*—**SQL injection, attacks, web application, accuracy, database, WebAppShield.

I. Introduction

WITH the advent of the Internet of Things, social computing hyper-connectivity, and prolific telecommunication within a network-centric, service-oriented and globally interconnected environment, the opportunities for cyber-attacks are vastly increased and ever expanding. In this context, research into the cyber-attack typology and structures will enable automated means of support for network security managers in order to prevent and mitigate against cyber-attacks. In recent years, SQL injection attacks have been identified as prevalent against web applications. It challenges network security and users' data, which leads to a considerable loss of money and data every year.

Cyber threat categorisation has had a long history. The academic debate about network-enabled (malicious) behaviours began in the 1980s, when personal computers entered the mainstream, however, this coincided with the belief that skilled individuals could manipulate computer systems for personal gain or simply as a self-affirming and satisfying activity i.e., for fun. The term "hacker" was originally used to describe anyone who could manipulate a computer; it is now commonly a reference to an individual who engages in malicious cyber behaviour [1], [2]. The increasing frequency and types of cyber-attacks, including targeting critical infrastructure, have made cybercrime an area of research and national security interest. The attacker profile enables the target organisation to effectively direct resources to enhance security. However, the current hacking model does not fully take into account the multifaceted nature of the network vulnerabilities, including the emergence of social and ideological hackers [2].

In order to continue to evolve robust malware detection solutions, there is an urgent need to develop smart methods to effectively and efficiently detect malware by collecting large numbers of samples daily [3]. In this study, based on extensive analysis of exploratory data at the payload level of the application and vector optimisation of feature and Machine Learning (ML) algorithms, a SQLi attack protection with enhanced performance over the literature review has been developed and validated. A CountVectorizer is used to transform text into a vector of token counts. The comparative analysis of the performance of various models as carried out in this study has included Adaptive Boosting, Support Vector Machines (SVM), Random Forest, Decision Tree and the k-Nearest Neighbours algorithm (k-NN) to detect SQLi attacks in the two-tier architecture of the database management system (DBMS). These ML algorithms provide highly accurate predictions on unseen data. The innovation in this paper is based on the process of feature engineering performed on the payloads, also the evaluation of different ML models based on these features. Web-App auto-generated twin data structure replication. Shielding against SQLi attacks (WebAppShield) method was used regarding to Log loss method which is used as a measure for evaluating predicted probabilities. The results show that more than 99% accuracy has been achieved by the models. Moreover, other performance metrics have been established for the proposed algorithm, such as Log loss also called "logistic loss" is used as a measure for evaluating predicted probabilities and confusion matrix and sensitivity-specificity analysis. This verification process secures the database from its early stages.

Section II reviews the related work; Section III presents the methodology of the research to develop the solution of detection SQLi attacks. In Section IV, the experimental setup is described, and results analysis provided. Finally, Section VI concludes the paper.

Ahmed Abdulla Ashlam and Atta Badii, are with University of Reading, Reading, United Kingdom (e-mail: a.ashlam@pgr.reading.ac.uk, atta.badii@reading.ac.uk).

Frederic Stahl is with German Research Centre for Artificial Intelligence GmbH (DFKI) Laboratory Niedersachsen Oldenburg,Germany,26129 (e-mail: frederic_theodor.stahl@dfki.de).

A. Understanding Database Architecture in DBMS

The purpose of the Database Architecture is to illustrate the design of DBMS which helps in the development, design, implementation, and maintaining the DBMS. It also enables the database to be separated into individual components that can be modified and replaced independently. In addition, it makes the database components more comprehensible enabling the selection of the right DBMS Architecture supports for efficient management of the data. There are mainly three of DBMS Architecture forms 1-Tier Architecture, 2-Tier Architecture, 3-Tier Architecture [4].

1) A 1-Tier Architecture (Single Tier Architecture) of DBMS: It is a simple architecture where the client, server, and database all are placed on one machine.

Fig. 1 A 1-Tier Architecture of DBMS

2) A 2-Tier Architecture of DBMS: It is a client and server database where the presentation layer runs on a client (tablets, mobiles, computers, etc.), and, data are stored on a server in a second tier.

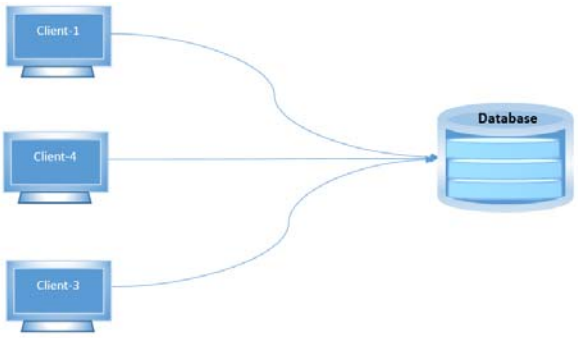

Fig. 2 A 2-Tier Architecture of DBMS

3) A 3-Tier Architecture of DBMS: It is the most common client-server architecture maintaining and developing the functional processes and logic, access to the data, store the data with an independent user interface. This type contains the following layers.

1. Presentation layer (Tablet, PC, mobile, etc.)
2. Application layer (S)
3. Database Server

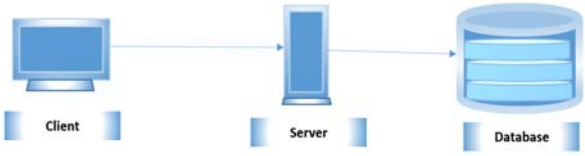

Fig. 3 A 3-Tier Architecture of DBMS

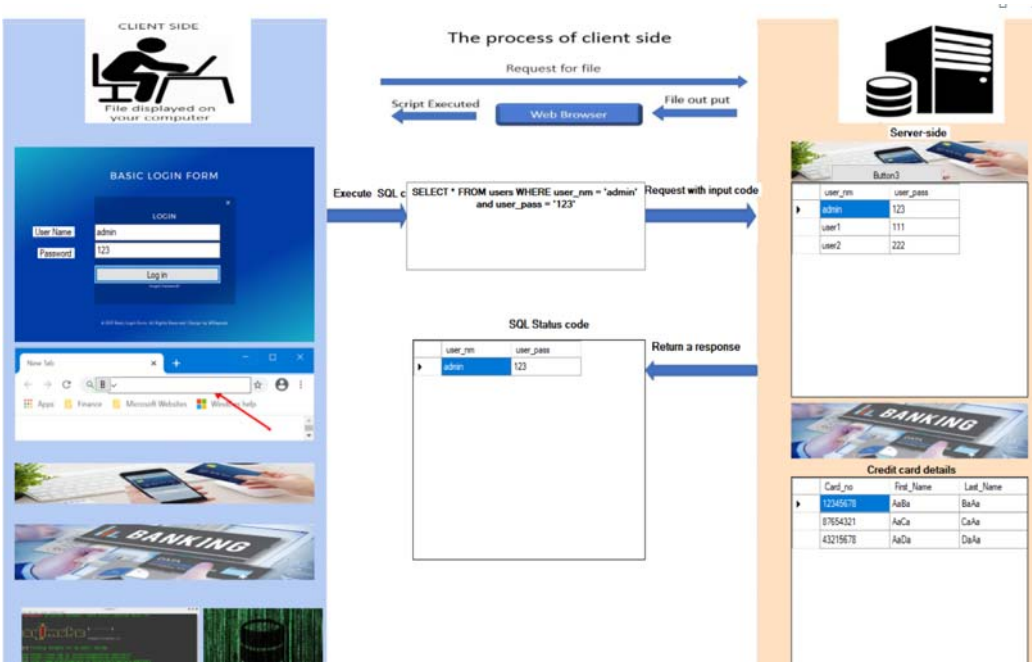

Fig. 4 Example of authorised login page

B. Understanding SQL Injection

SQL Injection is an attack that uses injection code and exploits the SQL query to access the database by gaining unauthorised access [23]. Following a simple example; we suppose that there is a banking website that allows the users to log in by entering the username and password related to the users. When entering the correct username and password, the authentication will be passed and login allowed. In the case of

an authorised login effort, the SQL statement constructed will be as follows where: Username = Admin Password= 123 SQL Query:

SELECT * FROM users WHERE U_name = 'Admin' and U_password = '123'

A user with malicious intent enters such input on the website of the username and password fields where: Username = Admin' or '1'='1', Password =' 'or '1'='1'. In this case, constructed of the SQL Query will be:

SELECT * FROM users WHERE U_name = admin' or '1'='1' and U_password =' 'or '1'='1'

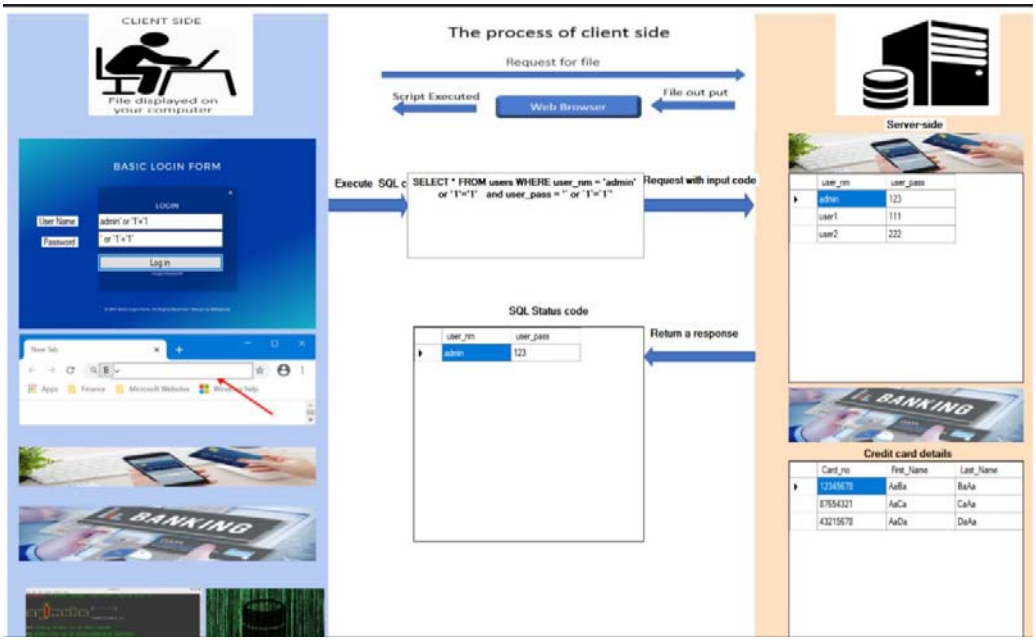

Fig. 5 Example of unauthorised login page

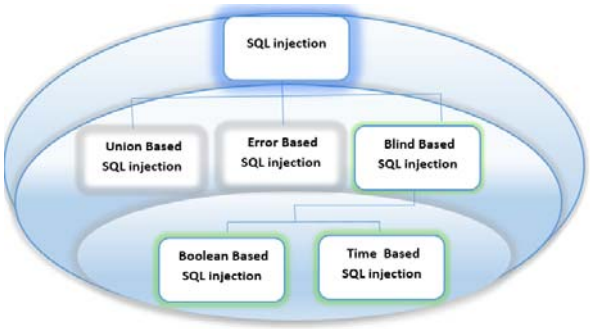

Fig. 6 Types of SQL Injection

Because 1 = 1 always will be true, the user will always be allowed to log in. Unauthorised access to details of a user's account can be obtained which could result in serious consequences for the user whose account was attacked. This is a case of stealing and data privacy infringement. This was an example of a SQLi attack, and this type of attack could be prevented on most websites and web applications these days. However, there are different types and more complicated forms of SQLi attacks, which will be defined later in detail. The target of the attackers is to exploit the database using SQLi. In order to defend the sensitive data stored in such databases, it is extremely important to protect them against SQLi attacks which include deleting tables, stealing important data.

The following are three categories of SQLi Attacks broadly classified: Union Based SQL Injection, Error Based SQL Injection and Blind SQL Injection [5], [6].

1) Union Based SQL Injection: In a SQL statement (query), the operator UNION is used to integrate two SQL statements or queries. An advantage of this feature is to return the desired results of the database token by Union Based SQL Injection. This is achieved by using the UNION keyword at the beginning of the SQL statement and injecting another query in place of the plain text. A simple example would be searching for Card details in a database. When entering the First_Name in the finding field, the following query is created. The entered value: AaBa

SQL Query: SELECT * FROM Card_details WHERE F_name= 'AaBa'

However, an attacker can be exploiting the database by entering the following in the Card_details search field. The entered value: AaBa' UNION DROP TABLE Card_details

SQL QUERY: SELECT * FROM Card_details WHERE F_name= 'AaBa' UNION DROP TABLE Card_details

Here the attacker runs two queries at the same time and the UNION keyword is used to combine both the queries. This might result in deleting the entire Card_details table. This

approach is used to perform an unauthorised action on the database, made possible by using the second part of the query.

2) Error Based SQL Injection: This works if there is a thereby triggering an error in the database by an invalid input in the SQL statement. It can be achieved by forcing the database into the subsequent action causing an error. Therefore, the errors generated by the database enable the attacker to gain information also exploiting the SQL query to obtain further manipulate the database.
3) Blind SQL Injection: This type of SQLi attack has been classified as the most intractable type. Because the state of the database is unknown when it returns generic errors, for example "Error in Syntax". It can be achieved when the attacker asking about the database decides on the further course of action based on the returned answers. Blind SQL Injection attacks are categorised into Boolean Based SQL Injection attacks and Time-Based SQL Injection attacks.

a) Boolean Based SQL Injection: This is a technique that relies on the database receiving an SQL query that forces the application to return results depending on whether the query returns the result TRUE or FALSE. Following the result, the content within the response will change or be the same. The attacker can infer whether the used payloads returned false or true, even if there is no returned data from the database. The need to enumerate a database char by char makes this attack typically slow.
b) Time Based SQL Injection Attacks: This is a technique that relies on the database waiting for a particular amount of time (in seconds) when it receives an SQL query before responding from the database. The attacker can deduce from the response time whether the result is TRUE or FALSE. The response delay depends on the result. Therefore, information about the database and application can be inferred by the attacker.

II. Literature Review

A pre-requisite of cyber-attack modelling is to examine the malware domain and previous research efforts focused on Malware Analysis and Modelling. In the field of computer security, many studies have been conducted using ML and various data mining methods [24] to detect known and unknown malware including studies that have made the following notable contributions relevant to the research focus, as follows:

- Comparison of the use of ML techniques with the traditional signature-based method for detecting malware [7].
- Proposal of a control flow graphs based multiclass malware discovery framework that consequently classified the malware into their individual families utilising Bi-normal segregation as a feature scoring metric [8].
- Introduction of an unsupervised method for detecting abnormal behaviour in computer and network logs, and comparing the effectiveness of standard and bidirectional repetitive neural network language models in detecting malicious activity in network log data [9].
- A method for unsupervised deep learning online that can detect abnormal network activity in real-time from system logs, which can analyse the degree of deviation in the contribution characteristics of individual user behaviour to increase interpretability and help analysts review potential cases of insider threats [10].
- A language model to predict the communication between two IPs using prediction error as a measure of the typical range or degree of fear of the observed communication that used the repetitive neural networks of long-term memory cells to capture the complex relationships and nuances of this language [11].
- A behaviour-based bot discovery map called BotGraph, which combines a sitemap and convolutional neural network to discover the internal behaviour of a bot [12].
- Discussion of all approaches to the pioneering model of deep learning and neural networks for intrusion detection systems [13].
- Evaluation of the effectiveness of unsupervised batch processing and streaming detection algorithms flowing through gigabytes of source traces recorded on four different operating systems to determine whether they can reliably and effectively detect attacks [14].
- The design of an intelligent system method using ML that can accurately and reliably classify malware under adversarial conditions [15].
- Consideration of the presence of spatial and temporal features in the network traffic data, and a proposed hierarchical neural network which will synchronously learn the input traffic data with increasing accuracy so that the spatial and temporal characteristics of the data can be extracted effectively [16].
- An overview of the current malware detection systems performance [17].

TABLE I
Summary of Malware Detection Systems [17]

Feature Type	Classification Method	Strengths	Weaknesses
Static	Gain ratio, Fisher Score, ANN, DT, NB, BNB, BDT, SVM	Accuracy, Imbalance problem, Unknown Malware	Packed Executables
	Information gain, DT, KNN, SVM, RF, NB, Bayesian Network		
	Mutual Information, TF, cosine similarity	Detects Malware variants families	Pack executable Accuracy
	Game Theory, Genetic Algorithm, SVM	Dimensionality problem, False positive	Time
Hybrid	DT, KNN, SVM, RF, NB, Bayesian Network.	Hybrid approach, Scalable, Automation, Robust Defend	Manifest Single program behaviour.
	SVM ensemble with bagging	Polymorphism & Metamorphism	Large size data
Dynamic	Behavioural analysis using Phytogenic trees	Flexible, Automation, Zero day malware	False Positive
	Behaviour analysis using SVM, DT, IBI, RF + KNN, NB	Reduces runtime and memory overheads.	Incomplete picture malware activity
	Behaviour Graph Matching	False positive, Fast generation of behaviour graphs	Accuracy, Latest malwares Testing time

A. Supervised ML

ML algorithms are broadly categorised into Supervised Learning algorithms and Unsupervised Learning algorithms. Supervised learning is a ML type that works in the simplest form as in the following manner. The present dataset is called a training dataset and each individual component of this dataset is labelled. The relationship between the data and the label can be basically learned by a supervised learning model, and then new data are classified is deployed to classify new data from the test dataset that the model had not seen previously. The supervised learning algorithms are broadly classified as Regression Algorithms and Classification Algorithms as illustrated in Fig. 7.

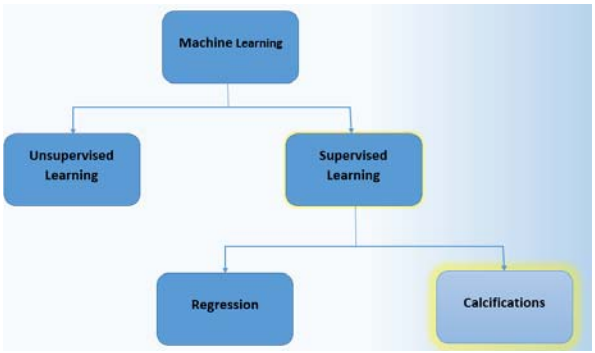

Fig. 7 Supervised ML

Regression Algorithms are used to predict the value of an individual component of data, such as shop value predicting or stocks predicting. Commonly, the regression algorithms predict the values as quantitative or numerical. The individual data components are classified by using classification algorithms e.g., predicting whether the vehicle is a truck or car another form of transport or predicting if it is going to rain or not on a specific day. Classification algorithms are also used to predict qualitative values. Fig. 7 represents a derived hierarchy of classification and regression algorithms. This paper has focused on classification algorithms in ML.

III. Methodology

Accordingly, this study is motivated by the need to contribute to research and innovation in tackling the challenges of SQLi attacks protection; specifically, the system was designed and implemented as illustrated in Fig. 8 which presents the workflow of the proposed method: Data Collection, Pre-processing, Feature Extraction, Model Classification, Performance Analysis, WebAppShield, and Results.

After a detailed study of the various algorithms as deployed in ML in order to obtain the most accurate predictions; and based on the percentage of errors in results of models deployed to-date, an algorithm enabling active learning feedback was developed for detection and prediction of SQL injection attacks with reduced false positives and false negatives (WebAppShield). The approach here has focused on re-training the system based on its errors regarding the Log loss method which is used as a measure for evaluating predicted probabilities so as to achieve some performance improvement upon the original model.

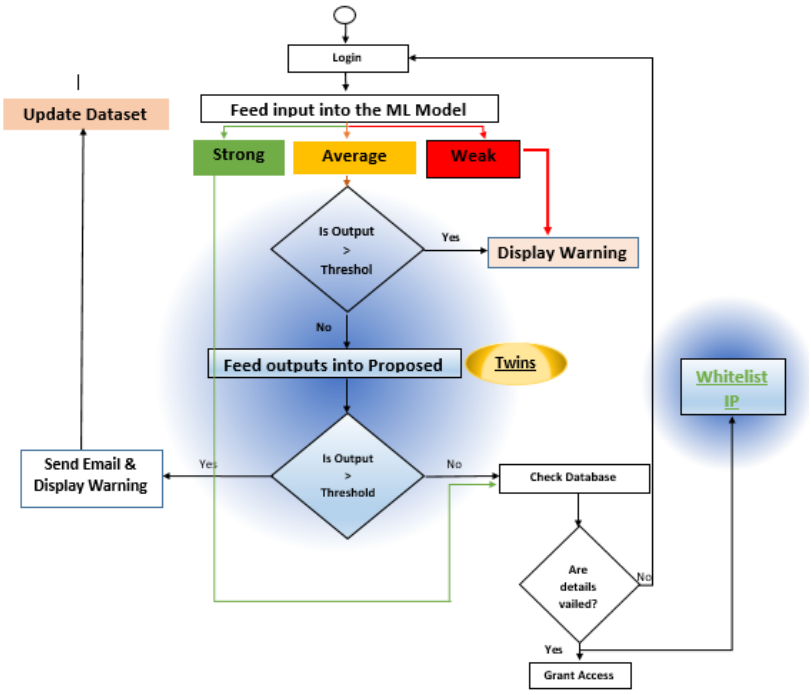

Fig. 8 Workflow of the proposed method in application layer

A. Data Collection

This section describes data collection methods as deployed in cybersecurity applications. Specifically, data collection can be performed in two different ways: i) through processing

system calls from host-based operating systems, and ii) through extracting packet headers from the transmission control protocol (TCP) communications stack to extract packet headers and payloads from network and application traffic packets [18]. Accordingly, packet capture and NetFlow protocol are the two main methods used to collect network traffic as follows:

1) Packet capture enables more detailed data collection from the network as it includes the extraction of the entire network packet (including the packet headers) for all the information to be transmitted. In particular, the data collected from these packets include protocol type, flow header, source, packet size, flags, and destination IP addresses, source, and destination port numbers [19].
2) NetFlow enables the collection of brief information or some pre-defined features related to packet flow in the network. Examples of attributes that can be extracted include the number of packets during a certain period of time or the amount of data sent over the network. Only summary data are taken into account and new types of features cannot be extracted to meet the new requirements. However, compared to data collection through packet capture, the data collection efficiency for the whole NetFlow is higher [18], [20]. The Python library named Lib-injection and the National Vulnerability Database comprises all types of SQLi attacks. In addition, normal SQL queries have been added to discriminate normal text.

B. Pre-Processing

The categorical variables need encoding before the variables in the dataset are fed into the ML models. There are many ways to encode categorical variables. Possibly the dataset possesses many categorical features with high cardinality in which case the number of the features after an encoding process would be massive, thus requiring a lot of computational power and memory. Thus, performing a feature selection is the best way to cleanse the dataset for efficient and effective data input into the ML model.

C. Feature Engineering

A range of normalisation and encoding techniques are applied to deal with data sparsity, extrema data-points and categorical variables in (CountVectorizer) [21]. The development of a classifier model requires feature extraction of Malware feature space and epistemological analysis with an ontological commitment to the particular attack-tree-theoretic ontology of the pre-conditions and the triggers for specific attack types [22].

D. Classification Model

Five ML algorithms including KNeighbours, AdaBoost, SVM, Random Forest, and Decision Tree, were implemented using Python to train the dataset. After the classification, learners were re-trained, the accuracy of each classifier was studied to identify the most accurate classifier. Accordingly, the accuracy of the obtained results has been improved from 94% to 99% by WebAppShield to enhance the performance of the resulting model. In addition, when compared with results achievable using SVM, the proposed approach shows performance improvement in terms of enhanced sensitivity and specificity i.e., lower number of False Positives and False Negatives.

Each of the features is to be examined and information value analysis is performed to indicate the feature engineering steps as required.

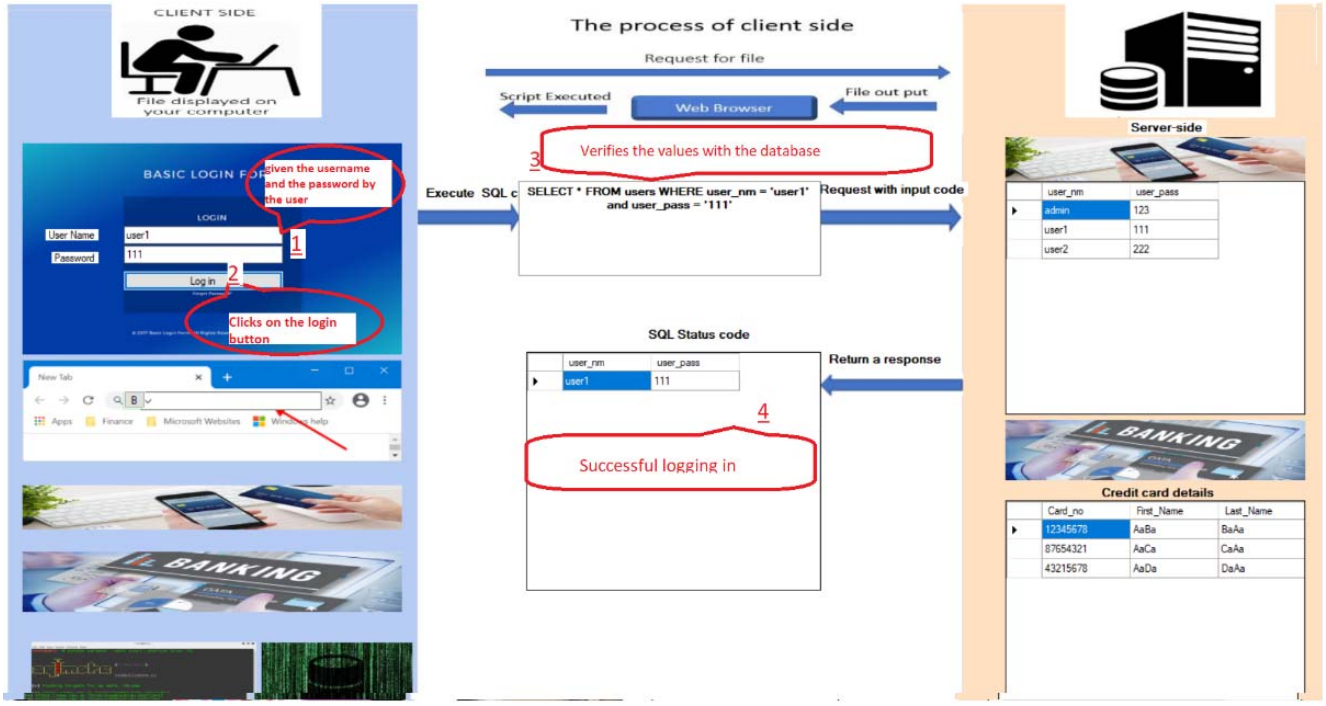

Fig. 9 Login Page

IV. Results and Discussion

Initial statistical analysis of the pattern distribution of the dataset indicated that it is feasible to source a balanced dataset. For any of the selected ML models, a 99% prediction accuracy is regarded as the baseline that should set a qualifying basis to continue to pursue solution refinement at all levels in seeking to arrive at the optimal performance. These methods increase reliability and accuracy of results in terms of making decisions about whether an operation is an attack or a valid operation. The work as reported here began with the preparatory steps including studying the research literature, identifying the benchmarking data sets, the cardinalities, feature spaces and pattern distribution of the selected datasets available. This has comprised data normalisation and encoding techniques and the injection pipeline setup to establish the input space for selected mining, modelling and ML tools/algorithms. The implementation has included the selection of the dataset for the planned work, the exploratory data analysis consisting of statistical pattern distribution analysis of the selected dataset and completion of the data processing pipeline for the experimental scenarios. The purpose of this approach is to prevent the SQLi attacks by using a two-tier architecture, which will: i) strictly repel unauthorised users when they try to access the database, and ii) make the attacker incapable of producing any useful result, for instance, accessing, deleting, or modifying the data in the database that may lead to permanent changes in the content or behaviour of the applications. Testing Web-app Invisible Needs Accessing Safely (WebAppShield) method can easily be applied with all sorts of databases or languages. The next steps are there to explain how SQLi attacks (malice Query) are possible when using blind variables.

1) Fig. 9 is the login page where the username and the password are given by the user. They then click on the login button, verify the values with the database, and if the credentials are correct the user will be successful in logging in.

Here, by giving the SQL Injection query in the username's field and a random password, as can be seen, the user is able to log in successfully.

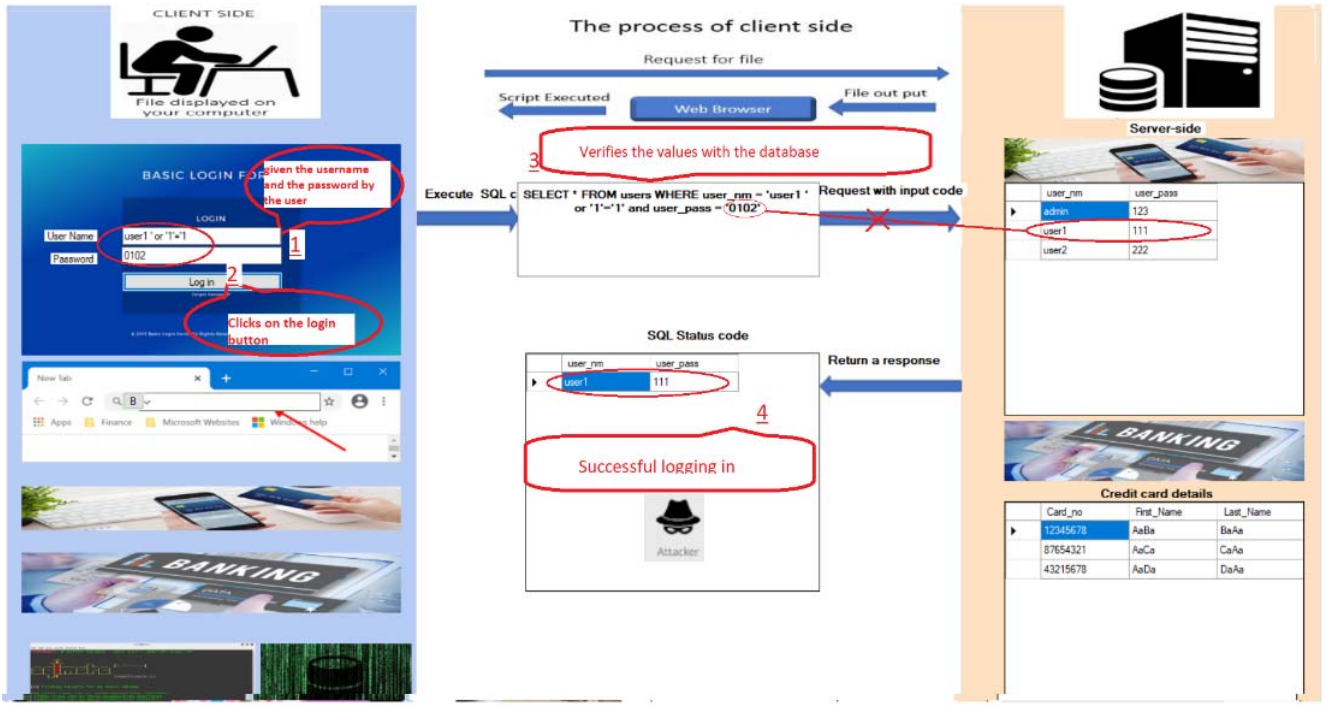

Fig. 10 SQL Injection Query is injected and Log in Page gets accessed

2) *Ultimately*, SQLi attacks can vary depending on the expertise and imagination of the attacker. Examples of some of the most common ones found are discussed in the following two scenarios.

A. An ML algorithm (SVM) has recognised the Injection query. Fig. 11 shows how the database is being protected from SQL Injection.

B. Although the accuracy of the selected ML module was 94%, the ML algorithm has not recognised this Injection Query. Consequently, it has become necessary to consider the unprotected remaining 6%. Therefore, WebAppShield repelled the injection query, as presented in Fig. 12.

As can be seen, instead of using a single quote, the attacker used a double quote, leading to the ML module being unable to recognise this new injection code. However, through performance analysis and Iterative optimisation of the SQLI Detection model the database is being protected.

Fig. 11 ML algorithm has recognised the Injection query

Fig. 12 Iterative optimisation of the SQLI Detection model

V. Conclusion

SQL injection attacks have been identified as a common attack for web applications. It challenges network security and users' data, which leads to the loss of a considerable amount of money and data every year. This study provides an overview of the development of relevant previous research work and in particular approaches to network traffic data analysis, pre-processing, and malware feature engineering to establish the requisite techniques and tools to support the planned research. Accordingly, the work as reported here began with the preparatory steps including studying the research literature, identifying the benchmarking data sets, the cardinalities, feature spaces, and pattern distribution of the selected datasets available. This is comprised of data normalisation and encoding techniques, and, the injection pipeline setup to establish an

input space for selected mining, modelling, and ML tools/ algorithms. The implementation has included the selection of the dataset for the planned work, the exploratory data analysis consisting of statistical pattern distribution analysis of the selected dataset, and the completion of the data processing pipeline for the experimental scenarios for the next phase of the project. This research study selected SQLi attacks as the focus given that these are increasing sharply unrestricted access to databases with some malicious content. In order to obtain the best possible results and distinguish injections queries from normal text, the data were subjected to a set of ML methods, in which SVM achieved the highest accuracy rate of 0.94. Subsequently, a method (WebAppShield) was proposed to enhance the accuracy of the results in the application layer in real-time. This provided an accuracy of more than 99%. WebAppShield has been used for detecting and preventing SQLi attacks in the application layer in real-time. The proposed future work is improvement using (WebAppShield) in the database layer in real-time, looking at up different types of cybersecurity attacks and continuous improvement, committed to security solutions, research, and practical testing.

REFERENCES

[1] C. Meyers, S. Powers, and D. Faissol, "Taxonomies of cyber adversaries and attacks: a survey of incidents and approaches," Lawrence Livermore National Lab. (LLNL), Livermore, CA (United States), 2009.

[2] R. Seebruck, "A typology of hackers: Classifying cyber malfeasance using a weighted arc circumplex model," *Digital investigation,* vol. 14, pp. 36-45, 2015.

[3] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A survey on malware detection using data mining techniques," *ACM Computing Surveys (CSUR),* vol. 50, no. 3, pp. 1-40, 2017.

[4] R. Peterson. "Database Architecture in DBMS: 1-Tier, 2-Tier and 3-Tier." https://www.guru99.com/dbms-architecture.html (accessed.

[5] S. Mishra, "SQL injection detection using machine learning," 2019.

[6] I. Tasevski and K. Jakimoski, "Overview of SQL Injection Defense Mechanisms," in *2020 28th Telecommunications Forum (TELFOR)*, 2020: IEEE, pp. 1-4.

[7] M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo, "Data mining methods for detection of new malicious executables," in *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001*, 2000: IEEE, pp. 38-49.

[8] A. Kapoor and S. Dhavale, "Control Flow Graph Based Multiclass Malware Detection Using Bi-normal Separation," *Defence Science Journal,* vol. 66, no. 2, 2016.

[9] A. Tuor, R. Baerwolf, N. Knowles, B. Hutchinson, N. Nichols, and R. Jasper, "Recurrent neural network language models for open vocabulary event-level cyber anomaly detection," *arXiv preprint arXiv:1712.00557,* 2017.

[10] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," *arXiv preprint arXiv:1710.00811,* 2017.

[11] B. J. Radford, L. M. Apolonio, A. J. Trias, and J. A. Simpson, "Network traffic anomaly detection using recurrent neural networks," *arXiv preprint arXiv:1803.10769,* 2018.

[12] Y. Luo, G. She, P. Cheng, and Y. Xiong, "BotGraph: Web Bot Detection Based on Sitemap," *arXiv preprint arXiv:1903.08074,* 2019.

[13] P. Poornachandran and S. KP, "A Compendium on Network and Host based Intrusion Detection Systems," *arXiv preprint arXiv:1904.03491,* 2019.

[14] G. Berrada *et al.*, "A baseline for unsupervised advanced persistent threat detection in system-level provenance," *Future Generation Computer Systems,* 2020.

[15] S. M. Devine and N. D. Bastian, "Intelligent Systems Design for Malware Classification under Adversarial Conditions," *arXiv preprint arXiv:1907.03149,* 2019.

[16] P. Wu and H. Guo, "LuNet: A Deep Neural Network for Network Intrusion Detection," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2019: IEEE, pp. 617-624.

[17] S. Ranveer and S. Hiray, "Comparative analysis of feature extraction methods of malware detection," *International Journal of Computer Applications,* vol. 120, no. 5, 2015.

[18] D. Gümüşbaş, T. Yıldırım, A. Genovese, and F. Scotti, "A Comprehensive Survey of Databases and Deep Learning Methods for Cybersecurity and Intrusion Detection Systems," *IEEE Systems Journal,* 2020.

[19] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar, "Towards the deployment of machine learning solutions in network traffic classification: A systematic survey," *IEEE Communications Surveys & Tutorials,* vol. 21, no. 2, pp. 1988-2014, 2018.

[20] I. Birzniece, "Security Analytics: Dispelling the Fog," in *BIR Workshops*, 2018, pp. 160-169.

[21] P. Vishnu, P. Vinod, and S. Y. Yerima, "A Deep Learning Approach for Classifying Vulnerability Descriptions Using Self Attention Based Neural Network," *Journal of Network and Systems Management,* vol. 30, no. 1, pp. 1-27, 2022.

[22] A. Badii and D. Patel, "Evolving features-algorithms knowledge map to support NIDS data intelligence and learning loop architecting–a generalised approach to NIDS pattern feature space knowledge processing," 2008.

[23] Pallam, R., Konda, S. P., Manthripragada, L. & Noone, R. A. 2021. Detection of Web Attacks using Ensemble Learning. learning, 3, 5.

[24] Tang, P., Qiu, W., Huang, Z., Lian, H. and Liu, G., 2020. Detection of SQL injection based on artificial neural network. Knowledge-Based Systems, 190, p.105528.