Available online www.jsaer.com

***Journal of Scientific and Engineering Research*, 2020, 7(4):259-267**

Research Article

ISSN: 2394-2630
CODEN(USA): JSERBR

Artificial Intelligence and Machine Learning Techniques to Control SQL Injection Attacks

Kartheek Pamarthi

Kartheek.pamarthi@gmail.com

Abstract: In the current the environment, each and every one of us is dependent on web applications. Every single day, the number of people who utilise online applications continues to increase. Databases are utilised by the majority of organisations in order to save information regarding their users or any details regarding the services that they offer. Database communication is a common application of Structured Query Language (SQL). In a SQL Injection attack, the hacker obtains access to the database by inserting malicious SQL statements into the database. Database security is at risk from SQL injection attacks because they can alter database values, erase the database entirely, or steal database content. SQL injection is a method by which malicious actors can get unauthorised access to database information. Web applications leave themselves open to SQL injection attacks if they don't validate or filter user input. We use machine learning techniques including Convolution Neural Network, Decision Tree, naive Bayes, Gradient Boosting, and Support Vector Machine to look for problems with SQL injection.

Keywords: Structured Query Language (SQL), SQL injection, Database security, Artificial intelligence.

Introduction

With the advent of new Internet technologies, the quantity of data pertaining to networks has increased exponentially. While online applications do improve our quality of life, they also pose serious risks to the safety of our Internet infrastructure. By the year's end, Qihoo 360 had checked the safety of 1,979,060,000 Chinese websites. These testing found that 46.3% of web apps had security issues. At the highest proportion, SQL injection attack (SQLIA) and cross-site scripting attack (XSS) were the vulnerabilities.

Unfortunately, SQL injection attacks are all too widespread, making them an important security concern for any network. Attacks against Sony's Play Station Network in April 2011 utilised SQL injection. Pranksters gained unauthorised access to twelve million credit cards and seventy-seven million accounts. The exposure of personal information such Sony customers' usernames, passwords, addresses, and credit card transaction details cost the company up to $170 million USD in indirect damages. In February of 2017, a hacker from Russia going by the name of "Rasputin" managed to breach the system and get extensive access to the database server by taking advantage of SQL injection flaws. A substantial amount of sensitive information was stolen from over twenty organisations and government entities in the US and UK [1].

For all intents and purposes, SQL injection attacks are possible on any system that relies on a database to power a web application. In spite of this, it is challenging for the present Web application firewall (WAF) to cover all SQL injection attack variations due to its reliance on feature matching algorithms (rule base). This occurs because SQL injection attacks are quite similar to regular user access to the system. Submitting Web forms, query strings, or page requests is all that is needed to achieve this. Even more so, it's less obvious. Hence, no one solution can stop or detect every kind of SQL injection attack. Experts in the field have recently begun exploring ways to apply AI techniques to provide more intricate answers.

Web attacks like SQL Injection have been around for a long time, but they're still a major problem and a major drain on government and corporate budgets [2]. This is particularly the case when considering that not only have some attacks changed and progressed through time, but new attack pathways are constantly being found.

The mitigation of web assaults is a topic that receives a significant amount of attention and resources from both industry and security firms. However, many of the currently available mitigation solutions have limits that are being continuously worked to overcome by current research.

Static analysis of incoming online traffic, or signature detection, has been the mainstay of previous web assault protection methods. Assembling a signature indicative of the online assault is necessary for this method. When this signature is identified, firewalls or other security appliances can block the malicious traffic. This approach has the potential to protect network resources in real time and is fast, which is a major plus. However, this approach has the drawback of just being able to identify known threats.

Another way to protect against web attacks that target SQL injection is to pay attention to the format of incoming SQL queries. The discovery of an incorrect query is known as a SQL injection attack. Protecting against web attacks is another use of this strategy. The method's reliance on familiarity with the programme and the form of "normal" queries is one of its major limitations. But it finds new attacks, such as malformed queries, and has good detection results overall. Using machine learning techniques is a newly investigated SQL injection detection solution. Investigations into this tactic are ongoing. The most common methods used in this area include neural networks, decision trees, rule-based learning, and support vector machines (SVM).

It is possible to identify new assaults using these methods, which is one of the most significant advantages of using them. However, depending on the algorithm that is utilised, there is a possibility that the processing time will be lengthened, which is a potential disadvantage of these strategies.

In order to conduct research into SQL detection techniques, including those stated, it is necessary to have access to high-quality data. The web application's or server's logs of online traffic is used in most modern studies [3]. Our proposed method aggregates data from two sources: the web application's incoming traffic and the data sent between the web app and the database server housed on the same network node as a Datiphy device. We are starting by utilising the traffic data obtained at these two locations to build our databases. Afterwards, we create a third dataset by comparing and contrasting the occurrences in the datasets derived from the two data points.

Literature Review

Machine Learning

The term "machine learning" refers to a method that is used to automatically identify patterns in data. The web application's or server's logs of online traffic is used in most modern studies [3]. Our proposed method aggregates data from two sources: the web application's incoming traffic and the data sent between the web app and the database server housed on the same network node as a Datiphy device. We are starting by utilising the traffic data obtained at these two locations to build our databases. Afterwards, we create a third dataset by comparing and contrasting the occurrences in the datasets derived from the two data points. Researchers are currently employing a wide variety of machine learning algorithms in their work. Naive Bayes and Bayes Net are two strategies that are being utilised by the authors in [4] in order to create probability models for the purpose of traffic classification. utilising a decision tree technique, the authors of [5] are analysing their results in terms of performance characteristics. Additionally, they are utilising a decision tree technique. The usage of neural networks, which will be addressed further below, is another prevalent technique in the world of current IDS research. Decision Tree Algorithms. An example of a classification algorithm is a decision tree algorithm, which uses the dataset's feature values to build a predictive model. At each step, the algorithm chooses a feature that partitions the dataset in a way that maximises information gain [6]. Data record class homogeneity or purity, as measured by information gain, is the degree to which each feature can predict the class of the data record. By iteratively exploring the dataset's properties and selecting the one with the greatest information gain and coverage, decision tree algorithms essentially reward prediction conclusions with more records. Here, the authors are using a novel decision tree technique and evaluating the results using performance measures. Their precision when dealing with SQL injection data is between 86% and 87%.

Rules-based Algorithms. Similar to decision tree algorithms, rule-based algorithms iteratively select dataset attributes that best predict the class label. A set of classification rules of the form Condition → Class is the result of rule-based algorithms, as shown in [7]:

$$(\text{Gives Birth} = \text{no}) \wedge (\text{Aerial Creature} = \text{yes}) \rightarrow \text{Birds}$$

Coverage and accuracy are the two main metrics used to assess these rules. In a dataset, coverage refers to the proportion of examples that fall under the rules' purview, while accuracy measures the proportion of instances that are part of the projected class relative to the total number of instances covered by the rule. In terms of performance, rule-based classifiers are usually on par with decision tree algorithms, but they are also known for being exceptionally easy to understand [8]. Support Vector Machine Algorithms. A statistical method for classification, Support Vector Machine (SVM) algorithms perform admirably on datasets with a large number of dimensions. Support vector machines (SVMs) use examples from training data called support vectors to try to draw a line across classes. Discovering the maximum hyper-plane spacing between instances of each class accomplishes this. When it comes to discovering globally optimal solutions, support vector machines (SVMs) are seen to be superior to rule-based and neural network approaches. Using a support vector machine (SVM) classifier, the authors of [10] try out several centrality metrics. In contrast to the text-based SVM classifier we're employing in this study, the authors in [11] use an n-gram based analysis. The authors find that support vector machine (SVM) outperforms a kNN (K-nearest neighbour) approach. The authors conclude that SVM produces the greatest results when compared to kNN and Naïve-Bayes algorithms. Neural Network Algorithms. The original idea behind Neural Network algorithms was to create a model of the human brain's structure out of a web of linked nodes, with each node standing in for a neuron. Nodes at the lowest level of a network design receive dataset attributes as inputs and nodes at the highest level receive classification results as outputs. The output nodes reflect a weighted combination of the input, and the learning process modifies the weights according to the learning rate and classification error. A Multi-Layer Perceptron is a type of Neural Network that uses hidden layers of nodes between its input and output nodes. One potential drawback of Neural Networks is how slowly they train. "Training an ANN is a time-consuming process, especially when the number of hidden nodes is large." . The difficulty in interpreting results from Neural Networks is another problem that has recently come up in the research. This difficulty can make applications susceptible to attacks. Combining MultiLayer Perceptron and Support Vector Machine (SVM) approaches, the authors of [12] demonstrate an improvement in accuracy by comparing their results to those of other algorithms, such as J48 decision tree and JRip rule-based, which are mentioned below. Ensemble Techniques. "Improving classification accuracy by aggregating the predictions of multiple classifiers" is the goal of ensemble approach. These methods start with base classifiers like the ones we just discussed, and then they use a voting process based on the predictions those classifiers made to carry out classification. In this project, random forests were utilised, which are based on decision trees and serve as a classifier. Feature Selection. In order to improve performance, feature selection is employed to lower the data dimensionality. Here we have a method to get rid of features that aren't significant to categorization and features that are redundant, meaning they give information that is already available. The authors of Kar et al. [13] employ Weka and an information gain technique to pick features. Below, we'll go over both genetic search algorithms and correlated feature sets (CFS), which we're utilising in our project.

Attack Generation

There are mainly two approaches to collect data for SQL injection research: either by creating realistic simulated traffic or by capturing genuine web traffic entering an organisation or honeypot. There are benefits and drawbacks to both methods. Even though real-world online traffic is ideal, it's not always easy to tell which packets are malicious. Obtaining this kind of traffic can be challenging in and of itself because most organisations are wary of sharing web traffic owing to security and privacy issues. The fact that typical attack tools often use automated scans that would be easier to catch in a controlled lab environment is another problem with using a basic research honeypot. Advantages of simulated attacks include the ability to incorporate a wide variety of attacks based on current tactics, and the fact that they are controlled enough to distinguish between legitimate and malicious communications for labelling purposes. cited as [14].

Many resources are available to academics who want to test their suggested detection and mitigation systems by simulating actual attack traffic. Using the assault simulation tool Paros is recommended, as stated in [15]. According to Google Scholar searches, SQLMap is the most often utilised attack tool.

It has been found that SQLMap can be used to generate malicious traffic [16]. Also covered are SQLMap and manually written assaults, as well as other examples. In order to produce attack traffic, many researchers use manually coded SQL injection assaults, like in [17], our earlier work [18], and the present study. The most common cause of SQL injection vulnerabilities is the practice of developers passing database SQL queries constructed by concatenating strings. An attacker can modify the SQL statement by introducing SQL keywords or special symbols.

The execution leads to an assault on the system. The main cause is that we put too much faith in user-submitted data, neglect to filter it, and do not conduct reasonable server-side verification. The attackers are able to achieve their objectives, such as obtaining sensitive system data and taking control of the server, because of this. These are the broad strokes of a SQL injection attack [19], as seen in Figure 1.

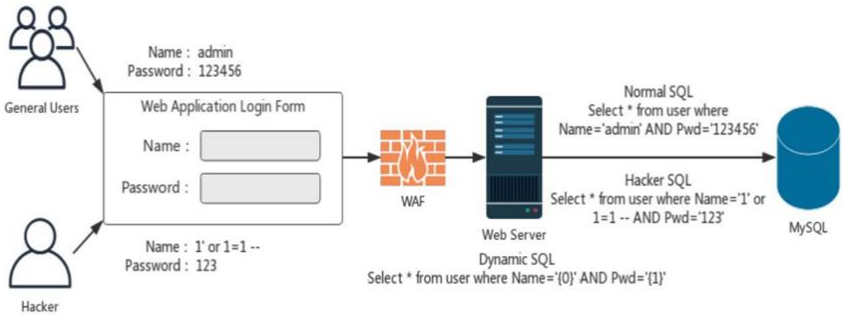

Fig. 1: Methodology for SQL Injection Attacks

From the beginning of SQL injection until now, numerous researchers have studied the detection of SQL injection from various perspectives and used various methodologies. There are now primarily two types of approaches to SQL injection prevention. Two things can be done to ensure operational security: first, developing with security in mind; and second, taking extra precautions during programme operation to identify SQL injection threats. Pattern matching, static and dynamic analysis, sequence comparison, abstract syntax trees, and black box and white box testing are all part of the second method. Some researchers have found success by bringing in research methodologies from related areas, such as data mining, machine learning, research text analysis, and SQL injection detection [20].

System Design and Implementation

This study introduces a machine learning-based methodology to identifying potentially harmful or benign incoming communications. The system's confidential business chat web app runs on a distant MySQL server. When a server generates traffic, it sends requests to the webapp server, which stores the data in one location. When the webapp server sends queries to the remote database, it stores the data in another area. After analysing and correlating the two datasets, a new dataset is created that incorporates features from both datasets. Using 5-fold crossvalidation, we evaluate the algorithms used for machine learning within the Weka Machine Learning Framework for their classification accuracy and efficiency in terms of model building time and training data classification time.

The whole operation of our system is depicted in Figure 2.

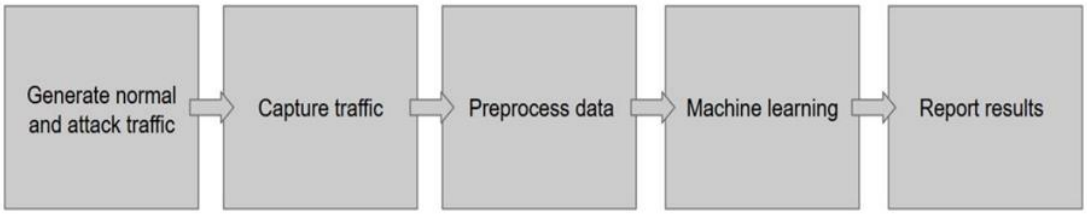

Figure 2: System Process

Architecture

Our current architecture is based on an HP server with 64 GB of RAM, two quad-core processors, and four KVM virtual machines.

This network consists of four nodes: one for web applications, one for traffic generation, one for databases, and one for Datiphy MySQL data capture. Fig. 3 and the following text provide brief descriptions of them.

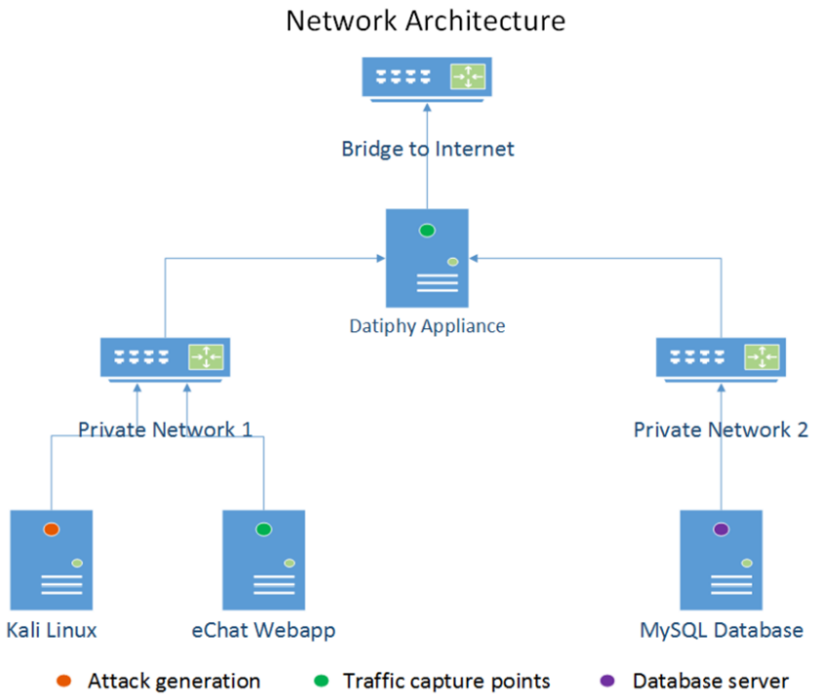

Figure 3: Network Architecture

Web Application Server. The webapp server is running Ubuntu/Apache, and the webspace hosts the custom web application. The database server is home to the MySQL backend, which is responsible for powering this webapp. This server is running Snort for both of our data acquisition points.

Traffic Generation Server. It is possible to generate both legitimate and malicious traffic on this server, which is running the Kali Linux system. Python and shell scripts utilising the Beautiful Soup Python libraries are used to produce both legitimate and malicious traffic.

Database Server. Linux Mint with MySQL is the operating system on this server. Every MySQL request for the webapp goes via this server because it is configured to receive database access requests from the webapp server's chat application.

Datiphy MySQL Data Capture Server. Datiphy Inc. has made available for research purposes a Datiphy appliance virtual machine (VM) on this server. This tool is being utilised by the project since it allows for the visibility of SQL traffic as well as other types of database activity. The Datiphy web interface allows full visibility into all traffic between the webapp and the MySQL database server by routing all traffic via this appliance.

Data Preprocessing

These three steps make up our data generating process: traffic generation, capture, and preparation. Following is a quick discussion of them.

Traffic Generation. As stated before, the scripts kept on the traffic generation server are what actually cause our project's simulated normal and malicious traffic. First, this server initiates a connection between the webapp server and the database server by sending HTTP POST requests to the chat webapp. This causes MySQL traffic to flow between the two servers. The chat web app's typical traffic consists of simulated user interactions.

As we'll see later on, malicious traffic fluctuates and sometimes includes SQL injection attempts that are hand-coded. To create an environment that is statistically close to real-life online conversation, the text is created at random but yet has a rather realistic level of word and sentence repetition.

Traffic Capture. The data is stored in two places: the webapp server and the Datiphy appliance. The traffic is being captured on the webapp server using the Snort intrusion detection system (IDS) tool. Files called PCAP contain the saved data. All data sent and received by the web app while interacting with the remote MySQL server is stored on the Datiphy appliance node. After that, a report is generated from this traffic and saved in CSV format using the Datiphy online interface.

Data Pre-processing. Most data pre-processing methods are conducted on the webapp server, which is also where the PCAP files generated by Snort data capture are stored. We use TShark, which is a command line interface to Wireshark [20], to process the PCAP data. With TShark, we can get useful information out of webapp communication packets by analysing PCAP files.

The output is a CSV file, which is subsequently prepared for additional processing by use of shell scripts. Shell scripts are used in a similar fashion to process data collected at the Datiphy appliance. The last step is to use shell scripts to combine the two datasets into a single file, resulting in the correlated dataset. Tokens, which take the shape of distinct text strings, are introduced into the dataset in order to facilitate correlation. Once events are shown to be associated, these tokens are removed.

Web Application

An enterprise chat application simulation built from scratch is the web app we're utilising in our system. It uses the PHP programming language and is hosted on the Apache web server. An HTML form is utilised by the programme, which is set up to store the simulated chat messages in a remote MySQL database backend. One particular place in our webapp where we intentionally left room for SQL injection is in the INSERT statement.

Experiment And Results

As shown in Table 1, we conducted trials using a variety of machine learning techniques. Model Time refers to the amount of time spent building the machine learning models, Testing Time to the amount of time spent classifying the testing dataset using 5-fold cross-validation, and Accuracy to the amount of accuracy each method achieves in classification. Figure 2 displays the classification accuracy, while Table 2 provides the F-scores.

Table 1: Results With 20000 Records

Dataset	Algorithm	Accuracy	Model Time	Testing Time
Webapp	JRip	94.740%	3m30.85s	2.70s
	J48	95.630%	1m42.65s	2.55s
	RF	96.525%	5m30.20s	30.60s
	SVM	94.025%	2m35.80s	1m10.45s
	ANN	96.715%	46m03.55s	3.35s
Datiphy	JRip	95.980%	6m10.90s	3.35s
	J48	96.995%	1m59.30s	2.45s
	RF	97.210%	6m10.20s	33.50s
	SVM	95.190%	2m11.50s	1m3.45s
	ANN	97.285%	41m23.00s	2.95s
Correlated	JRip	97.150%	11m50.80s	2.10s
	J48	97.295%	2m01.80s	2.05s
	RF	98.055%	4m22.55s	35.45s
	SVM	95.715%	3m30.85s	1m5.90s
	ANN	97.615%	47m25.25s	3.80s

Table 2: F-score for 20000 Record Data

Dataset	Algorithm	True Pos.	False Pos.	True Neg.	False Neg.
Webapp	JRip	9117	169	9831	883
	J48	9385	259	9741	615
	RF	9487	182	9818	513
	SVM	9134	329	9671	866
	ANN	9501	158	9842	499
Datiphy	JRip	9291	95	9905	709
	J48	9538	139	9861	462
	RF	9757	46	9954	243
	SVM	9226	188	9812	774
	ANN	9613	156	9844	387
Correlated	JRip	9623	193	9807	377
	J48	9656	197	9803	344
	RF	9860	32	9986	140
	SVM	9457	314	9686	543
	ANN	9644	121	9879	356

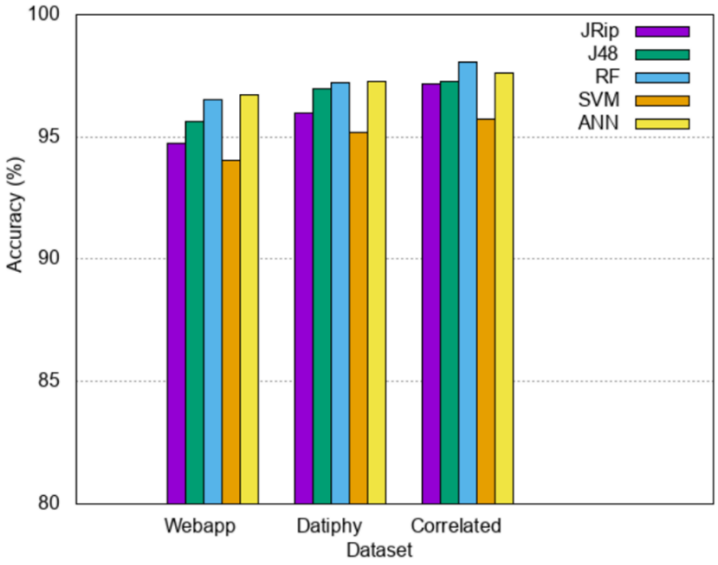

Figure 4: Classification Accuracy

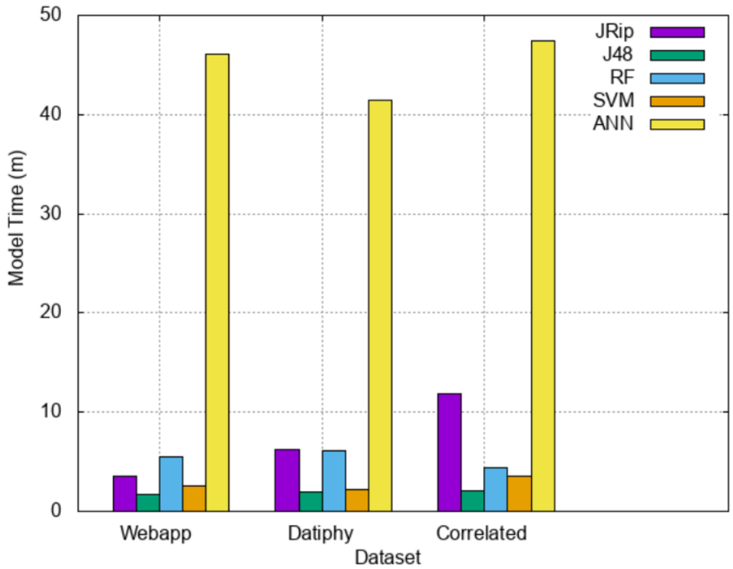

Figure 5: Modeling Time

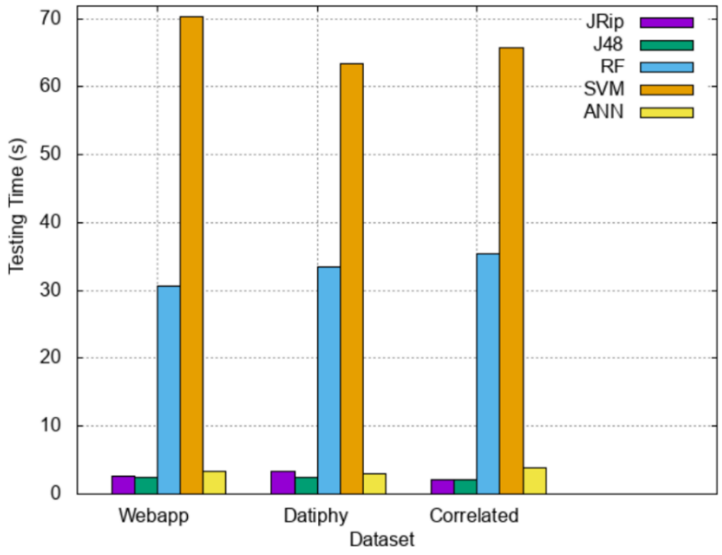

Figure 6: Testing Time

Analysis

The correlated dataset combines two datasets, thus there is more data accessible. Therefore, the results gained should be higher in terms of classification accuracy. This is based on my intuitive knowledge of the process. As previously stated, this has been the outcome of our studies thus far, with JRip utilising features from both datasets to generate rules. For example, the best-performing rules of the JRip algorithm take advantage of both the http.content_length and SQL Length datasets from the webapp and Datiphy, respectively. Results from trials utilising JRip and J48 algorithms on datasets containing 2000 records formed the basis of our earlier conclusions. Our findings which are shown in figure 4,5,6 are in line with the trend of consistently higher results when using the associated datasets compared to each dataset alone. Consistently better results across all parameters have also been achieved with larger datasets. Machine learning algorithms seem to be able to construct a model from the data that falls into certain categories, and their capacity to do so increases as they are exposed to more data, despite the fact that our attack creation is random. The SVM, Random Forest, and ANN algorithms have been implemented into this project and performance measurements have been reported.

Notable among the performance metrics is the decision tree algorithm's commendable relative performance throughout our trials. Although decision tree algorithms are known to be resource heavy, we hypothesised that they would outperform other classification algorithms when the data is typically easy to categorise using a small number of carefully chosen features. This is how we arrived at the results of the current project. From our trials, we can deduce that the quicker algorithms produce results on the correlated dataset that are almost identical to our MultiLayer Perceptron results, but take substantially less time to execute. These trials suggest that using data from several sources can enhance classification results just as much, if not more so, as using techniques that are known to be more accurate.

Conclusion and Future Work

In the face of SQL injection attacks and other web-based attacks, the security of sensitive information, such as financial and health documents, remains a primary concern. As the internet becomes increasingly prevalent in societal activities, this issue is becoming increasingly urgent. We have proposed a system for analysing data from multiple sources in order to enhance the precision of SQL injection attack detection in this project. We have also demonstrated that a variety of algorithms, such as decision tree and rule-based algorithms, can classify testing data with a level of performance that is comparable to that of Neural Networks, while necessitating noticeably less time to construct models. Future projects include the modification of this system to identify additional web-based attacks. Additional items on the list include the collection of additional data, such as traffic that exits the web application and enters the browser, the analysis of larger datasets to determine whether they enhance performance, and the evaluation of additional machine learning techniques for accuracy and performance.

References

[1]. OWASP Top 10, 2013, "Top 10 2013-A1-Injection", https://www.owasp.org/index.php/Top_10_2013-A1-Injection

[2]. Application Defense Center (ADC), 2015, "2015 Web Application Attack Report (WAAR)", http://www.imperva.com/docs/HII_Web_Application_Attack_Report_Ed6.pdf

[3]. M. Moh, S. Pininti, S. Doddapaneni, T-S. Moh, Detecting Web Attacks Using Multi-Stage Log Analysis, 2016 IEEE 6th International Conference on Advanced Computing (IACC), Bhimavaram, 2016, pp. 733-738.

[4]. K. Ross, M. Moh, T-S. Moh, J. Yao, Poster: Multi-Source Data Analysis For SQL Injection Detection, 38th IEEE Symposium on Security and Privacy (IEEE S&P), San Jo-se, CA, 2017.

[5]. S. Djanali, F.X. Arunanto, B.A. Pratomo, H. Studiawan, S.G. Nugraha, "SQL injection detection and prevention system with raspberry Pi honeypot cluster for trapping attacker," in Technology Management and Emerging Technologies (ISTMET), 2014 International Symposium on, vol., no., pp.163-166, 27-29 May 2014

[6]. D. Kar, S. Panigrahi, "Prevention of SQL Injection attack using query transformation and hashing," in Advance Computing Conference (IACC), 2013 IEEE 3rd International, vol., no., pp.1317-1323, 22-23 Feb. 2013

[7]. D. Kar, A.K. Sahoo, K. Agarwal, S. Panigrahi, M. Das, "Learning to detect SQLIA using node centrality with feature selection," 2016 International Conference on Computing, Analytics and Security Trends (CAST), Pune, 2016, pp. 18-23.

[8]. I. Lee, S. Jeong, S. Yeo, J. Moon, "A novel method for SQL injection attack detection based on removing SQL query attribute values," Mathematical and Computer Modelling, vol. 55, no. 1, 2012, pp. 56-68.

[9]. M. Mueter, F. Freiling, T. Holz, J. Matthews, (2008). "A generic toolkit for converting web applications into high-interaction honeypots," University of Mannheim, 280.

[10]. G.K. Sadasivam, C. Hota, "Scalable Honeypot Architecture for Identifying Malicious Network Activities," 2015 International Conference on Emerging Information Technology and Engineering Solutions, Pune, 2015, pp. 27-31.

[11]. R. Tiwari, A. Jain, 2012. "Improving network security and design using honeypots." In Proceedings of the CUBE International Information Technology Conference (CUBE '12). ACM, New York, NY, USA, 847-852.

[12]. S. Kulkarni, M. Mutalik, P. Kulkarni, T. Gupta, "Honeydoop - a system for on-demand virtual high interaction honeypots," in Internet Technology and Secured Transactions, 2012 International Conference for, vol., no., pp.743-747, 10-12 Dec. 2012

[13]. B. Hanmanthu, B. R. Ram, P. Niranjan, "SQL Injection Attack prevention based on decision tree classification," 2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO), Coimbatore, 2015, pp. 1-5.

[14]. C. Pinzón, J. F. De Paz, J. Bajo, Á. Herrero, E. Corchado, "AIIDA-SQL: An Adaptive Intelligent Intrusion Detector Agent for detecting SQL Injection attacks," 2010 10th International Conference on Hybrid Intelligent Systems, Atlanta, GA, 2010, pp. 73-78.

[15]. T, Gu, B. Dolan-Gavitt, S. Garg, "Badnets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain", eprint arXiv:1708.06733, 8/2017, https://arxiv.org/abs/1708.06733

[16]. J. Choi, H. Kim, C. Choi, P. Kim, "Efficient Malicious Code Detection Using N-Gram Analysis and SVM," 2011 14th International Conference on Network-Based In-formation Systems, Tirana, 2011, pp. 618-621.

[17]. R. Komiya, I. Paik, M. Hisada, "Classification of malicious web code by machine learning," 2011 3rd International Conference on Awareness Science and Technology (iCAST), Dalian, 2011, pp. 406- 411.

[18]. Datiphy Data Sheet, Dec. 2016, Datiphy Inc., San Jose, CA, The Snort Project, August 28, 2015, http://datiphy.com/resources/data-sheet/

[19]. "SNORT® Users Manual 2.9.7.3", https://www.snort.org/documents/snort-users-manual

[20]. Angela Orebaugh, Gilbert Ramirez, Jay Beale, Joshua Wright. 2007. Wireshark & Ethereal Network Protocol Analyzer Toolkit. Syngress Publishing.