MACHINE LEARNING BASED INTRUSION DETECTION SYSTEM FOR WEB-BASED ATTACKS

Co-authored by Sushant Sharma

Pavol Zavarsky

Sergey Butakov

Project report

Submitted to the Faculty of Graduate Studies,

Concordia University of Edmonton

in Partial Fulfillment of the

Requirements for the

Final Research Project for the Degree

MASTER OF INFORMATION SYSTEMS SECURITY MANAGEMENT

Concordia University of Edmonton

FACULTY OF GRADUATE STUDIES

Edmonton, Alberta

April 2020

MACHINE LEARNING BASED INTRUSION DETECTION SYSTEM FOR WEB-BASED ATTACKS

Sushant Sharma

Approved:

Pavol Zavarsky [Original Approval on File]

Pavol Zavarsky Date: April 14, 2020

Primary Supervisor

Edgar Schmidt [Original Approval on File]

Edgar Schmidt, DSocSci Date: April 15, 2020

Dean, Faculty of Graduate Studie

Machine Learning based Intrusion Detection System for Web-Based Attacks

Sushant Sharma
Information System Security and Assurance Management
Concordia University of Edmonton
Edmonton AB, Canada
ssharm11@csa.concordia.ab.ca

Pavol Zavarsky
Information System Security and Assurance Management
Concordia University of Edmonton
Edmonton AB, Canada
pavol.zavarsky@concordia.ab.ca

Sergey Butakov
Information System Security and Assurance Management
Concordia University of Edmonton
Edmonton AB, Canada
sergey.butakov@concordia.ab.ca

***Abstract*—Various studies have been performed to explore the feasibility of detection of web-based attacks by machine learning techniques. False-positive and false-negative results have been reported as a major issue to be addressed to make machine learning-based detection and prevention of web-based attacks reliable and trustworthy. In our research, we tried to identify and address the root cause of the false-positive and false-negative results. In our experiment, we used the CSIC 2010 HTTP dataset, which contains the generated traffic targeted to an e-commerce web application. Our experimental results demonstrate that applying the proposed fine-tuned feature set extraction results in improved detection and classification of web-based attacks for all tested machine learning algorithms. The performance of the machine learning algorithm in the detection of attacks was evaluated by the Precision, Recall, Accuracy, and F-measure metrics. Among three tested algorithms, the J48 decision tree algorithm provided the highest True Positive rate, Precision, and Recall.**

***Keywords*—web-based attacks, detection, machine learning, feature extraction.**

I. INTRODUCTION

Identification of web-based attacks and prevention of the attacks by supervised and unsupervised machine learning tools has been a very active area of research. One of the key challenges in the detection of web-based attacks is feature set extraction which is essential for the development of machine learning detection models. A holistic feature set for effective detection of the attacks has not been developed. To address the gap, we present in this paper a feature set engineering as an approach to improve the detection of web-based attacks. We applied machine learning algorithms on a publicly available HTTP traffic dataset from an authentic source CSIC HTTP dataset 2010[1]. This dataset contains the generated traffic targeted to an e-commerce web application, the dataset was generated automatically and contains 36,000 normal and more than 25,000 anomalous requests [2] that include web-based attacks like SQL injection, cross-site scripting and buffer overflow that were used in our experiments. The dataset was preprocessed and cleaned using Python script.

The Python script was developed to extract features from the dataset, transform the dataset into CSV format, and distinguished between normal and anomalous traffic. After extracting useful features we applied machine learning techniques in WEKA. Feature engineering was the primary challenge and the most important component in this research. Python script parsed the dataset fields with various features like the length of the fields, user-agent length, content length, GET and POST request, cookie length.

II. RELATED WORK

G. Kaur et al [3] proposed the detection mechanism of blind cross-site scripting using machine learning, where identification of the malicious payloads that were likely to store in the database through web application was achieved in the experiments. They used the Linear Support Vector Machine classifier technique to determine the difference between stored cross-site scripting and blind cross-site scripting and to detect the attack [3].

The approach employed by Thomson [4] for exploration of SQL injection attack where they investigated areas that specialize in feature selection from the dataset. They used machine learning algorithms for experimental use to evaluate features that have greater significance than others in feature sets. Also, they performed feature extraction to parse the dataset using Python script and identified that some features have more impact on the performance of classifiers in detecting SQLi attacks [4]. Abby in [5] filtered down only the benign and SQL injection attacks from the CSIC HTTP and ECML/PKDD data set. He experimented with the dataset so as to realize higher accurate results, an iterative methodology was chosen which allows the re-evaluation during the design and planning.

Sikha Bagui et al [6] used a hybrid feature selection process and classification techniques to classify cyber-attacks in the UNSW-NB15 dataset [7]. They employed two techniques (J48, Naïve Bayes) based on a correlation-based feature selection, where Naïve Bayes outperform with classification accuracy for most attacks but J48 did not perform any better with feature selection. Naïve Bayes classifier gave an attack detection rate for shellcode attack which is an injection attack at 91.53% with feature selection. Whereas, J48 was not able to detect attacks like shell code, Backdoor, Analysis, and Worms with feature selection.

III. EXPERIMENTATION

The performed experimentation in three phases is outlined in Fig 1.

Phase 1 focuses on data preparation of the dataset and it's pre-processing to identify sub-categories of anomalous traffic for a fine-tuned training set that helped us to identify the missing features for common attacks. The data set contains some errors, Latin characters, errors, blank space and rows at the end of the lines in the dataset which were fixed with the help of Python scripting language before being processed for the next phase.

Phase 2 was dedicated towards the feature extraction from the dataset that contains fields like GET, POST request, content-length, keywords count for SELECT, DROP,

UNION, DELETE, MODIFY, content that contains field, number of special characters, login name, and password. The cleaned dataset was put into the CSV file before feeding it to Weka 3.8, an open-source tool for machine learning and data mining.

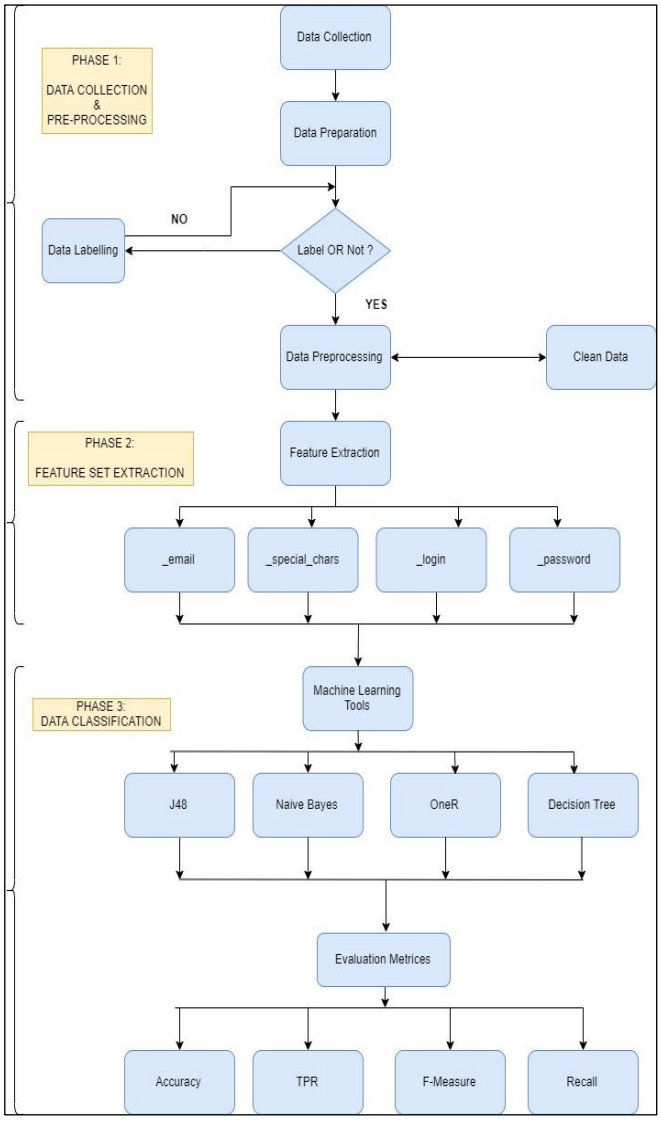

Fig 1: Flow chart outlines three phases

The dataset set was pre-processed at a finite level using Pyhton script. The feature extraction is the most important component of the research. The script successfully parse the dataset fields with various features like length of the fields, user-agent length, content length, GET and POST requests length, cookie. The steps used to parse the dataset using Python script are explained below:

Step 1 - Script infrastructure to process multiple files, where we go through multiple files and produce a single output file.
Step 2 - If we process multiple input files, we used a Global variable to write the header only once.
Step 3 - We choose the columns of the processed CSV files to parse. These keywords are searched in input files to extract their text. For example, "Host:" will be searched, and that line will be extracted and input to Python dictionary.
Step 4 - We not only extracted selected features into selected CSV columns, but added our processed features. These were denoted by CSV columns preceded with an underscore for e.g. "_keywords". The "_keywords" column was input after analyzing the GET request to search for features like SQL injection.
Step 5 - "refine_results()" function was used to further analyzer the input data. We counted the number of characters of the GET request, and stored the result into a CSV column called "_len_of_GET". We parsed out four additional fields from within the "content" field, namely: login, password, nombre, and email. We found selected features from GET, POST, or content fields, such as those that indicated SQL injection, buffer overflow, or XSS script attacks. Below is the script to count the number of characters of the GET request.

```
# Create GET request length
    if od['GET'] == 'null':
        od['_len_of_GET'] = '0'
    else:
        od['_len_of_GET'] = '{0}'.format(len(od['GET']))

    content = od['content']

    _login = parse_field(content, 'login')
    od['_login'] = (_login)

    _password = parse_field(content, 'password')
    od['_password'] = (_password)

    _nombre = parse_field(content, 'nombre')
    od['_nombre'] = (_nombre)

    _email = parse_field(content, 'email')
    od['_email'] = (_email)
```

Step 6 - We read all lines from each input file, and search for the chosen CSV columns from step 3. The "Content-Length" text is special. Upon reading this line, we extract the number of bytes of content, and then read that many bytes from the file.
Step 7 - When there is an empty line, that means it is the end of one GET or POST request. At this time, we write the parsed data of GET and POST to the output file.

```
csv_columns = [
    "GET",
    "POST",
    # ##################################################################
    # These do not seem to differ between anamomous and normal data
    #
    #"User-Agent:",
    #"_user_agent_len",
    #
    #"Pragma:",
    #"Cache-control:",
    #"Accept-Encoding:",
    #"Accept-Charset:",
    #"Accept-Language:",
    "Host:",
    #"Cookie:",
    #"_cookie_len",
    #"Connection:",
    "Content-Length:",
    "content",
    # ##################################################################
    # These are our own extractions from the data
    #
    "_label",
    "_keywords",
    "_len_of_GET",
    #"_register",
    "_login",
    "_password",
    "_nombre",
    "_email",
    "_special_chars",
]
```

Fig 2: Extraction using keywords

Step 8 - When parsing a selected CSV column, we altered some input. We changed all commas to underscore and stripped newline characters.

In Phase 3, the data was fed to machine learning methods using algorithms and classifiers that are pre-defined in the Weka with multiple features extracted from phase 2. The final phase consists of four procedures which were data collection, data preprocessing, feature extraction and classification of the data. Firstly, cleaning of the data was done from errors which came across while feeding it to Weka. Secondly, Weka gave a leverage to transform text/string into nominal type using built-in filter during pre-processing step. Thirdly, feature extraction using a python script was conducted that helped in distinguish between the most relevant features in the detection of web-based attacks in the HTTP traffic from the dataset.

IV. Experimental Results and Discussion

A detail of extracted features and experimental results using evaluation metrics for the machine learning classifiers can be seen in table I and table II.

We have used three ML classifier for the research., J48 from the decision tree, One rule (OneR) from rule system and Naïve Bayes from bayesian ML models. Firstly, J48 is used to classify different applications and perform accurate results of the classification. J48 algorithm is one of the best machine learning algorithms to examine the data categoriacally and continuously that is used to measure the improved performance and produce higher rate of accuracy. The decision tree consists of internal nodes that denote different attributes, the branches depict the possible values of the attributes between the nodes, and terminal node tell the final value. Secondly, OneR classifier is one of the simple and most effective callsifier algorithm in ML. It depends upon the classification rules based on the single predictor's value and not rely on the frequency of the target, then select the rule with the smallest total error as its "one rule". To create a rule for a predictor, one can construct an evaluation/frequency table for each predictor against the target. It is easy to interpret the rules for each predictor, its value and the target. Thirdly, Naïve Bayes is a high-bais, low-variance classifier that mainly calculates the probability of an item belonging to a certain class depending on the metrics obtained from the training data [8]. Also, it requires a small amount of training data to estimate the parameters. Being an intuitive method, it uses the conditionl probabilities of each attribute belonging to each class to make a predicition.

The preprocessing of the data was done at a granular level using a python script to identify missing features for common attacks. Initially, raw data was cleaned by removing errors before testing it on the Weka 3.8. Further, using in-built filters from the Weka 3.8 made the transformation of type from string to nominal followed by the attribute selection process for desired results. Lastly, feature extraction was carried using Python script to extract the most relevant features that contributed to identifying the web-based attacks. The script successfully parsed the dataset fields by distinguishing normal and anomalous traffic, along with special features that define the type of web-attacks. The metrices used to evaluate the accuracy of the classifier in the research.

The features extracted using the Python script were (1) content_length, (2) content, _(3) label (norm & anom), (4) keywords, (5)_length_of_GET, (6)_login_count, (7) special_chars, (8) Pragma, (9) cookie_len, (10) host, (11) connection, (12) cache_control, (13) accept_language, (14) accept_encoding, (15) _email, (16) user-agent, (17) nombre, (18) password count, (19) _special chars, (20) login.

The list of special extracted features is shown in Table I.

TABLE I – EXTRACTED FEATURES DESCRIPTION

Feature Name	Description
_len_of_GET	Number of Characters in getting request
_keywords	To identify the type of web-based attack.
content_length	A number of bytes of the content.
password_count	A number of characters in the password field.
_login count	A number of attempts to log in to the database.
_cookie_len	The amount of time it will hold information on to the user system.
_special_chars	To identify the special characters in the request field.

Results of the performance of machine learning algorithms in the detection of malicious traffic are summarized in table II. J48 classifier gave the highest detection rate of 94.5% that helped in detecting the type of attack from the HTTP traffic dataset with cross-validation of n=10 folds using Weka 3.8. Whereas, Naïve Bayes classifier gave the highest F-measure rate of 93.9% among the three-classifier used in our research.

TABLE II: EXPERIMENTAL RESULTS

Classifier	TP Rate	Precision	Recall	F-Measure
J48	94.5	94.7	94.5	93.4
Naïve Bayes	94.0	94.5	94.0	93.9
OneR	92.5	93.1	92.5	89.8

The attack detection rate concluded by Sikha Bagui et al [6] in their research for Naïve Bayes classifier was averaging 90% with feature selection for cyber-attacks whereas our research brought results from Naïve Bayes at 94% true positive rate with feature selection. The F1-Measure for NB turns out to be the best at 93.9% among the three classifiers. However, J48 outperform other classifiers with a True positive rate of 94.50% having the best attack detection rate as compared to Sikha Bagui et al [6] of 0% attack detection rate for cyber-attacks.

Thomson [4] in the research thesis referred to 10 features extracted and selected by the CFS from the CSIC-2010 dataset. However, we extracted 20 features that contributed essentially in detecting web-based attacks like SQLi, XSS, Buffer Overflow. The TP rate for SQL-Injection of Thomson [4] was 91.2% without feature selection and 53.7% with general feature selection compared to our results for 94.5% TP rate for J48 classifier in detecting the web-based attack.

V. Conclusion

While performing experiments, we cleaned and labeled CSIC HTTP 2010 dataset that helped in distinguish between normal and anomalous traffic. The preprocessing of the data was done at a granular level using a python script to identify missing features for a common attack. Further, feature extraction from the dataset played a key role in identifying malicious behavior and the attack types like SQL injection (SQLi), Cross-Site Scripting (XSS) and Buffer Overflow by using various Machine Learning classifiers like J48, Naïve Bayes, OneR, Decision table that use evaluation metrics to find the accuracy using Weka 3.8. Also, by applying fine-tuned feature set engineering resulted in 20 features extracted with improved detection of web-based attacks, thereby increasing the true positive rate. Lastly, our experimental results with three machine learning algorithms (J48, Naïve Bayes, OneR) demonstrates the reliability in the detection of web-based attacks, and the J48 decision tree algorithm was depicted to be the best performing algorithm with the best attack detection rate of 94.5%.

VI. Acknowledgment

I am very thankful to Preetpal Kang from Sibros Tech, California USA for his supreme assistance in developing Python script.

References

[1] R. Martinez, 2018. "Web Application Attacks-Datasets" GSI / web-application-attacks-datasets," *GitLab*. [Online]. Available: https://gitlab.fing.edu.uy/gsi/web-application-attacks-datasets.

[2] *HTTP DATASET CSIC 2010*. [Online]. Available: https://www.isi.csic.es/dataset/.

[3] G. Kaur, Y. Malik, H. Samuel, F. Jaafar, "Detecting Blind Cross-Site Scripting Attacks Using Machine Learning," *Proceedings of the 2018 International Conference on Signal Processing and Machine Learning - SPML 18*, 2018.

[4] A. Thomson, 2016. A "Exploration of SQL Injection (SQLi) Detection using Machine Learning" Edinburgh Napier University.

[5] P. Aaby, 2016." Evaluating Web App Datasets towards Detection of SQL Injection Attacks with Machine Learning Techniques" Edinburgh Napier University.

[6] S. Bagui, E. Kalaimannan, S. Bagui, D. Nandi, and A. Pinto, "Using machine learning techniques to identify rare cyber-attacks on the UNSW-NB15 dataset," *Security and Privacy*, vol. 2, no. 6, Oct. 2019.

[7] *The UNSW-NB15 data set description*. [Online]. Available: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/.

[8] R. M. GmbH, "Naive Bayes (RapidMiner Studio Core)," Naive Bayes - RapidMiner Documentation. [Online]. Available: https://docs.rapidminer.com/latest/studio/operators/modeling/predictive/bayesian/naive_bayes.html.