IEEE 38th International Conference on Information Networking (ICOIN), Ho Chi Minh City, Vietnam, 2024

Malicious URL and Intrusion Detection using Machine Learning

Amr Hamza, Farah Hammam, Medhat Abouzeid, Mohammad Arsalan Ahmed, Salam Dhou, Fadi Aloul
Department of Computer Science and Engineering
American University of Sharjah
Sharjah, UAE

Email: sdhou@aus.edu

***Abstract*—Cyberattacks are becoming increasingly sophisticated and evolving danger to the Web users. Therefore, addressing the growing threat of cyberattacks and providing automated solutions became a necessity. The purpose of this paper is to use machine learning (ML) techniques for malicious websites detection and classification, and intrusion detection. Different ML algorithms were applied, namely Decision Tree (DT), K-Nearest Neighbors (KNN), Naive Bayes (NB) and Support Vector Machine (SVM). Two datasets were utilized to train the ML models. The first dataset contains two classes of websites: "malicious" and "benign". The second dataset has six classes of different network intrusion cyber-attacks: "normal", "blackhole", "TCP-SYN", "PortScan", "Diversion", and "Overflow". Experimental results demonstrated that the ML algorithms were able to achieve high accuracy in predicting website maliciousness and intrusion detection. Using the first dataset, DT KNN, and SVM classifiers exhibited the best performance for detecting malicious URLs with accuracies over 99%. Using the second dataset, the DT classifier proved most suitable for intrusion detection, achieving an accuracy of 95%. This paper suggests the integration of ML techniques into online security systems to enhance their efficacy in detecting and preventing cyber threats.**

Keywords- *malicious URLs, machine learning, system penetration, intrusion detection*

I. INTRODUCTION

The Web is a large platform that is used by billions of people around the world. The Web has a wide range of criminal enterprises such as spam-advertised commerce, propagating malware and financial fraud via phishing [1]. One common aspect between all these cybercriminal activities is that they all have unsuspecting users visit their websites. These visits can be triggered by email, Web search results, or links from other Web pages but they all necessitate the user performing some action, such as clicking to specify the desired Uniform Resource Locator (URL). These malicious URLs could also lead to intruders accessing information stored on the users' devices such as pictures, location, emails, etc.

The idea of this paper centrally revolves around the protection of general users against malicious URLs, phishing attempts, and other security concerns. Most antivirus services provide tools that identify viruses, malware, and worms. However, they can slow down the devices that they run on. Furthermore, relying on a firewall system alone is not sufficient to prevent a network from all types of network attacks [2]. The traditional approaches for detecting malicious URLs often rely on signature-based techniques, which can be easily bypassed by polymorphic URLs [3]. Therefore, offering automated solutions using the emerging machine learning (ML) techniques, can provide a great improvement in malicious websites and intrusion detection.

The main contribution of this paper is to utilize the large datasets available nowadays and leverage the powerful ML techniques for URL maliciousness prediction and intrusion detection. Multiple machine learning techniques are utilized in this work including decision tree (DT), support vector machine (SVM), k-nearest neighbors (KNN), and Naïve Bayes (NB) classifiers. Various evaluation metrics are used to evaluate each of these classifiers such as accuracy, precision, recall, and F1-score. Moreover, the receiver operating characteristic (ROC), area under the curve (AUC) and confusion matrix are used. Two datasets were utilized, namely malicious URLs dataset and intrusion detection dataset in order to identify different evolving adversarial security concerns. The findings of this work help the cybersecurity authorities predict malicious URLs, cyber dangers, thereby improving the security of online settings for all users.

The rest of this paper is organized as follows. Section II presents the related works. Section III explains the datasets and methodology considered in this work. Section IV presents the experimental results. Section V provides a discussion of the results. Section VI concludes the paper.

II. RELATED WORK

There are numerous research papers that propose solutions to solve several security-related concerns. Justin et al. [3] explored lexical and host-based aspects of the linked URLs to identify malicious Websites using online learning techniques. Researchers found that online algorithms are especially useful when the training data is too large to be effectively processed in batch processing and when the distribution of parameters that characterize dangerous URLs is dynamic. Their proposed online algorithm achieved a classification accuracy of 99% using a balanced dataset. Another research paper [4] suggested a three-class classification system for websites into benign, phishing and malware using a learning-based technique. Without accessing the websites' content, their technique solely evaluates the URL itself which reduces the run-time latency and the chance of exposing users to browser-based security flaws.

Because of using the ML approach, their system achieved 97.53% accuracy in identifying dangerous websites which outperformed blacklisting services in terms of generality and coverage. Another method for identifying dangerous websites that prioritizes privacy protection was done by Wu et al. [5]. They employed structural partitioning and singular value decomposition (SVD) to protect the private information. Afterwards, an evaluation was conducted using SVM. Their method was able to identify a significant number of rogue websites by their URLs. Lakshmanarao et al. [6] proposed an ML-based solution for detecting malicious websites using different ML techniques, namely LR, KNN, DT and RF. In addition, they made use of different feature extraction methods. The researchers concluded that using the hashing vectorizer and RF classifier achieved the highest accuracy of 97.5%. This model was used in a mobile app for detecting malicious URLs.

For intrusion detection solutions, Wu et al. [7] utilized the KDD intrusion detection dataset to evaluate several models, namely J48, RT, Random Tree, Decision Table, Multilayer Perceptron (MLP), NB and Bayes Network classifiers. The Bayes network classifier had the greatest value for properly identifying the regular packets. The RF classifier has the lowest RMSE value, lowest false positive rate and the greatest accuracy rate of 93.77%. Except for the false negative parameter, the RF classifier offers adequate performance parameters.

Furthermore, Choi et al. [8], Vanhoenshoven et al. [9] Kaddoura et al. [10] and Prieto et al. [11] adopted various novel methodologies and perspectives in detecting and categorizing malicious web links and websites, utilizing different ML techniques and datasets. These works are the most similar to the work proposed in this paper. Choi et al. [8] presented a method that detects malicious URLs and identifies specific types of threats they pose. In a similar work, Vanhoenshoven et al. [9] delve into the use of ML techniques for detecting malicious URLs. Further, Kaddoura et al. [10] explored the classification of websites based on their malicious or benign nature. The study specifically leverages network features in conjunction with supervised ML algorithms, providing a distinct methodological approach from the previous studies. Lastly, Prieto et al. [11] proposed a knowledge-based approach to identify potentially risky websites. While the details were not given, their work signifies an interesting perspective that deviates from the typical ML-centric methodology and integrates a knowledge-based approach for risk detection. The papers discussed above are closely related to the proposed work since they make use of a similar approach and utilize datasets that contain features similar to the ones used in this work. They were also able to acquire high accuracies using ML models similar to ours. Table 1 summarizes the papers discussed above.

Table 1: Summary of Literature Review Studies

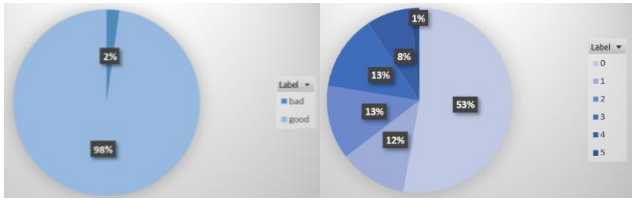
Reference	**Type of Attack Targeted**	**Dataset Used**	**Classifier Used**	**Accuracy**
Choi et al. [8]	attack types and malicious URLs	Real life dataset collected by the authors	SVM	93% for attack types and 98% for malicious URLs
Vanhoenshoven et al. [9]	malicious URLs	Public dataset (2.4 million URLs)	RF	97.69%
Kaddoura et al. [10]	malicious URLs and network features	Public dataset (1,782 URLs)	SVM	96%
Prieto et al. [11]	domains with malicious content	Generated dataset	LR	89%

III. METHODOLOGY

A. *Datasets Description*

In this paper, two publicly available datasets that relate to detecting malicious URLs as well as intrusion detection were utilized. The first dataset used is "Dataset of Malicious and Benign Webpages" [12], which will be referred to as dataset ***A***. This dataset contains 10 features such as URL, URL length, IP address, geographic location and others. The dataset consists of 1.52 million records that are split into a training set that contains 1.2 million records, and a testing set that contains 362k records. Each record represents a webpage that is either labeled as benign (good) or malicious (bad). The dataset is highly imbalanced with 98% of the data belong to the benign class and the rest (2%) belong to the malicious class.

The second dataset used was "Network Intrusion Detection" [13], which will be referred to as dataset ***B***. This dataset contains 5000 records of features extracted from Network Port Statistics to protect modern-day computer networks from cyber-attacks. The dataset contains 31 features such as switch ID, Port Number passed, Received Packets, Sent Bytes, Sent Packets, and others. The dataset consists of six classes: 0 (Normal), 1 (Blackhole), 2 (TCP-SYN), 3 (PortScan), 4 (Diversion) and 5 (Overflow). Fig. 1 shows the percentage of records belonging to different the classes in each of dataset ***A*** and dataset ***B***.



(a) (b)

Fig. 1. Percentage of records of different classes in (a) dataset ***A***, and (b) dataset ***B***

B. *Data Preprocessing and Model Selection*

Several data pre-processing techniques were applied to the datasets before ML algorithms were used. For dataset ***A***, irrelevant features were dropped from the dataset such as ID, URL, IP address and content. This results in a total of 8 features used for classification. Moreover, the binary features were encoded using ordinal encoding. Due to the fact that the dataset is highly imbalanced, under-sampling was applied by taking a random subset of 150K from the benign records while considering all the 8063 malicious records present in the dataset.

For dataset ***B***, dimensionality reduction was applied by merging the 'Packets Looked Up' field with the 'Packets Matched' field into a single new field named 'Packets not Found'. Additionally, the samples belonging to class 5 were dropped because the class has very few samples (1% of the dataset). Dimensionality reduction was also applied by dropping irrelevant features from the dataset such as the Table_ID, and Max_Size. Additionally, the features that had no variance in the values for all samples were also dropped, such as the 'is_valid' field. This process results in a total of 26 features left to be used for classification. Moreover, ordinal encoding and scaling were applied to features such as 'received packets' and 'bytes', and the 'sent packets' and 'bytes'. For model selection, holdout method were used where 20% of the dataset is used for testing while the rest of the dataset is used for training.

C. *Classification Algorithms Used*

Decision Tree (DT) algorithm: creates a tree-like model by learning basic decision rules from training data [10]. The root of the decision tree represents the entire dataset and each internal node represents a decision rule based on one of the input features. The branches represent the possible values of the feature, and the leaf nodes represent the predicted value of the target variable. To make a prediction for a new data point, the input features of the testing dataset are compared against the decision rules represented by the internal nodes of the tree, and the predicted value is obtained by following the appropriate branch to a leaf node. The metrics that decision trees rely on to determine the best feature to split the dataset based on at each internal node are impurity measures such as Entropy or Gini Index, defined as follows:

$$Gini = 1 - \sum_{i=1}^{c}(p_i)^2 \quad (1)$$

$$Entropy\ (S) = \sum_{i=1}^{c} -p_i \log_2 p_i \quad (2)$$

where p_i is the relative frequency of class *i* at a specific node.

K-nearest neighbors (KNN): a supervised non-parametric ML algorithm. It assigns a class label to an instance based on the class labels of its K nearest neighbors in the training data. Using a distance metric (e.g., Euclidean distance, Manhattan distance), the algorithm determines the distance between the instance and each training sample. The K closest neighbors are identified, and the class title is selected by majority vote. The predictions are affected by the choice of K. In addition, the KNN classifier avoids the time-consuming training process, and, more importantly, bypasses the need to learn individual program profiles separately [17]. Thus, the cost of learning program behavior is significantly decreased. Euclidean distance is defined by equation (3).

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (3)$$

where *x* and *y* are samples being compared, x_i and y_i represent feature *i* of each of the samples *x* and *y*, respectively, and *n* is the number of features describing each sample.

Naive Bayes (NB): an algorithm based on Bayes theorem, which predicts the label of a data point based on the probability of a hypothesis (or label) given some observed evidence (or features). The algorithm classifies the new data point by calculating the posterior probability of each class given the observed evidence and assigning that point to the class with the highest probability. The mathematical formula (4) calculates the probability of a sample to belong to a specific class given a feature vector, where *y* represents the class label and *X* represents the feature values.

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)} \quad (4)$$

The NB algorithm assumes conditional independence among features. Furthermore, it can work with missing feature values.

Support Vector Machines (SVM): A powerful and widely used supervised ML algorithm. Its goal is to find a hyperplane in an N-dimensional space to separate the data points belonging to different classes [10]. The hyperplane is selected in such a way that the data points are separated into distinct regions, one for each class, and the margin between the regions is maximized. SVM can still work even if the data is not linearly separable by using the kernel trick to map the data into a higher-dimensional space where it becomes separable by a hyperplane. This allows the SVM algorithm to handle complex and nonlinear relationships between the features and the target variable.

All four classifiers mentioned above are applied to each of dataset ***A*** and dataset ***B***, and their results are compared.

D. *Evaluation Metrics*

Several evaluation metrics are used to assess the performance of each of the ML models in classifying the target variable. In this work, accuracy, precision, recall, F1-score, receiver operating characteristic (ROC), area under the curve (AUC) and confusion matrix are used.

Accuracy is the simplest and commonly used metric for classification tasks. It is defined as the ratio of correct predictions to the total number of predictions as defined in formula (4):

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (4)$$

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives and FN is the number of false negatives.

However, accuracy may not be the best metric to use especially when dealing with imbalanced datasets. Precision and recall are two metrics that are commonly used for classification tasks and they work well in the case of imbalanced datasets. Precision and recall formulas are provided as follows:

$$Precision = \frac{TP}{TP+FP} \quad (5)$$

$$Recall = \frac{TP}{TP+FN} \quad (6)$$

F1-score is a harmonic mean of precision and recall. It is a good metric to use when dealing with imbalanced datasets. It is defined as follows:

$$F1\ Score = \frac{2 \cdot Precision \cdot Recall}{Precision+Recall} \quad (7)$$

The ROC is a graphical representation of the trade-off between the true positive rate and false positive rate for different classification thresholds [16]. The AUC can be calculated as the area under the ROC curve. The AUC is a popular metric for evaluating binary classification models. It measures the performance of the model across all possible thresholds, which can be useful when dealing with imbalanced datasets.

Lastly, a confusion matrix is a table that summarizes the performance of a classification model. It contains four values: true positives, false positives, true negatives, and false negatives [16]. The values in the confusion matrix can be used to calculate various metrics, including accuracy, precision and recall as mentioned in equations (4) – (6).

IV. Experimental Results

A. *Results using Dataset **A***

Different ML algorithms were applied to dataset ***A***. The following subsections show the results of each of the ML models.

- Decision Tree (DT)

DT algorithm was applied on dataset ***A***. Several tests were run to find the optimal tree size that results in the highest prediction accuracy. The size of the optimal tree was 19 nodes and it resulted in an accuracy of 99%. This classifier achieved a recall value of 99% and a precision value of 96% for the 'malicious' class. The classifier also achieved an F1-score of 98%. The AUC score obtained was 0.99.

Fig. 2 shows the confusion matrix resulting from applying the DT classifier on dataset ***A***. As can be seen from the figure, most of the samples were correctly classified.

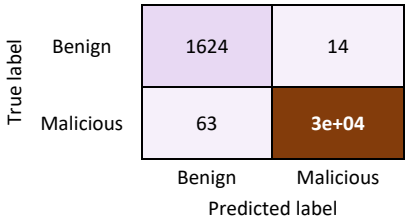


Fig. 2. Confusion matrix resulting from applying DT classifier on dataset ***A***

- K-Nearest Neighbors (KNN)

One of the important factors in prediction accuracy for KNN is the number of neighbors used. It was found that the number of nearest neighbors for dataset ***A*** that yielded the highest accuracy (99.88%) was 100 neighbors. The precision of the malicious class is 100%. However, the recall was about 95% and the F-score was 97%. The AUC of the ROC curve was 99.86%.

Fig. 3 shows the confusion matrix resulting from applying the KNN classifier on dataset ***A***. As can be seen from the figure, most of the samples were correctly classified.

- Naïve Bayes (NB)

Using the NB model, an accuracy of 99.75% was obtained. The precision of the malicious class was 100% while the recall was approximately 89%. The F1-score was 94%, which was lower than that of the decision tree and KNN models. The AUC of the ROC curve was 99.69%.

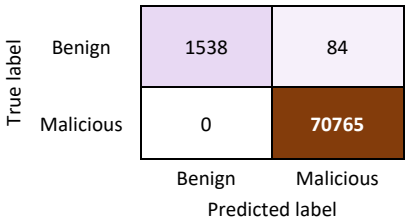


Fig. 3. Confusion matrix resulting from applying KNN classifier on dataset ***A***

- SVM with Linear Kernel

An SVM with linear kernel model was utilized in this work. Several tests were run to select the best hyperparameters for the model. The best accuracy obtained was 99.73% at iteration 5000. The precision of the malicious class was 100%. The recall, on the other hand, was 89% similar to that achieved using the NB model. The achieved F1-score was 94%. This demonstrates that the linear SVM model and the NB model are both less accurate models compared to KNN and DT classifiers.

- SVM with Polynomial Kernel

An SVM with polynomial kernel model was utilized in this work. Several tests were run to select the best hyperparameters for the model. The polynomial degree selected was 6 which achieved an accuracy of 99.75%. Fig. 4 shows the confusion matrix resulting from applying the SVM with polynomial kernel classifier on dataset ***A***. As can be seen from the figure, most of the samples were correctly classified.

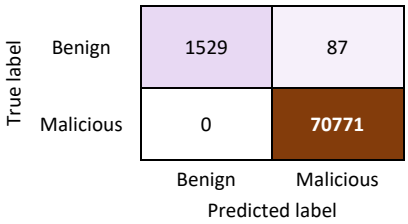


Fig. 4. Confusion matrix resulting from applying the SVM classifier with polynomial kernel on dataset ***A***

The precision of the malicious class was 100% while the recall was 95%. The combination of the two measures yielded an F1-score of 97%. The results of all the ML models applied to dataset ***A*** are summarized in Table 2.

Table 2. Applicability metrics of all ML models applied to dataset ***A***

Class	Metric	DT	KNN	NB	Linear Kernel SVM	Poly Kernel SVM
Malicious	*Precision*	96%	100%	100%	100%	100%
	Recall	99%	95%	89%	89%	95%
	F1-score	98%	97%	94%	94%	97%
Benign	*Precision*	100%	100%	100%	100%	100%
	Recall	100%	100%	100%	100%	100%
	F1-score	100%	100%	100%	100%	100%
Overall	*Accuracy*	99.76%	99.88%	99.75%	99.73%	99.75%
	AUC	99.98%	99.86%	99.69%	99%	99%

B. *Results using Dataset **B***

Different ML algorithms were applied to dataset ***B***. The following subsections show the results of each of the ML models.

- DECISION TREE (DT)

Several tests were run to select the ideal size of the DT that results in the maximum prediction accuracy. The testing accuracy was at its highest (95%) when the tree had 49 nodes. Fig. 6 shows the confusion matrix resulting from applying DT on dataset ***B***. As can be seen from the figure, both the TCP_SYN (class 2) and Port Scan (class 3) attacks have remarkably similar features which resulted in false negatives and false positives between these classes in the confusion matrix.

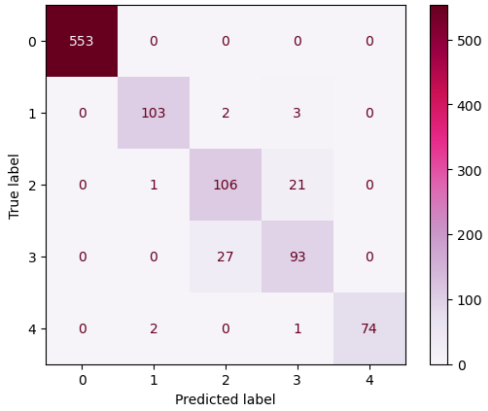


Fig. 6. Confusion matrix resulting from applying the DT classifier on dataset ***B***

The DT classifier had precision values ranging between 79% and 100% for the different classes as seen in Table 3. The recall and F1-score values ranged between 78% and 100%.

- KNN

KNN was applied on dataset ***B***. Several tests were run to find the number of neighbors that maximizes the accuracy. It was found that 13 neighbors is the best number of nearest neighbors. The accuracy of the KNN model was comparatively lower than that of the DT classifier, which was around 85.19% as can be seen in Table 3. The KNN classifier was mainly not able to differentiate between the TCP_SYN and Port scan classes, as well as misclassifying a hefty 39 samples from the Diversion type class as a blackhole. The precision values ranged between 58% and 99 %, and recall values ranged between 42% and 100%. The generated F1-scores ranged between 51% and 99%, as can be seen in Table 3. Due to these scores, it can be concluded that this classifier is not as accurate as the DT classifier.

- NAÏVE BAYES (NB)

NB classifier was applied to dataset ***B***. As can be seen in Table 3, the accuracy of NB was lower than that of the DT, but similar to the KNN model, sitting at about 84.69%. This classifier had difficulty distinguishing between several samples in the TCP_SYN and Port scan classes too, as well as misclassifying 38 samples from the blackhole type class as Diversions. This model yielded F1-scores for its classes ranging between 60% and 100%, and precision scores ranging between 52% and 100% and recall scores ranging between 48% and 100%, as shown in Table 3. It can be concluded that the scores achieved by KNN classifier are not as high as the ones achieved by the DT model.

- SVM WITH LINEAR AND POLYNOMIAL KERNELS

SVM models with linear and polynomial kernels were also applied to dataset ***B***. The results achieved were low compared to other models. Even after performing hyperparameter tuning on both models, the performance did not vary significantly. Moreover, due to their poor performance in all metrics as can be seen in Table 3, SVM models with linear and polynomial kernels were considered unsuitable for dataset ***B***.

Table 3. Applicability metrics of all ML models applied to dataset ***B***

Class	**Metric**	**DT**	**KNN**	**NB**	**SVM Linear**	**SVM Poly**
Normal	*Precision*	100%	99%	100%	86%	58%
	Recall	100%	100%	100%	91%	100%
	F1-score	100%	99%	100%	89%	74%
Blackhole	*Precision*	97%	58%	78%	85%	30%
	Recall	95%	72%	54%	28%	2%
	F1-score	96%	64%	64%	42%	5%
TCP-SYN	*Precision*	79%	73%	61%	47%	75%
	Recall	83%	79%	96%	45%	13%
	F1-score	81%	75%	75%	46%	22%
PortScan	*Precision*	79%	76%	99%	25%	25%
	Recall	78%	70%	48%	28%	5%
	F1-score	78%	73%	65%	27%	9%
Diversion	*Precision*	100%	67%	52%	44%	22%
	Recall	96%	42%	72%	68%	3%
	F1-score	98%	51%	60%	54%	5%
Overall	*Accuracy*	95%	85%	85%	67%	57%

V. DISCUSSION

This work employed four ML methods, namely DT, NB, SVM and KNN, to detect harmful websites and identify system intrusions. As shown in Table 2, all ML algorithms demonstrated excellent performance in detecting fraudulent websites especially the KNN model applied on dataset ***A***. This suggests that these algorithms can be valuable in identifying and preventing cyber threats, particularly in the context of website and system security. Furthermore, the DT algorithm exhibited a high F1-score of 98% when applied on dataset ***B***, indicating its proficiency in recognizing patterns and making accurate predictions on new, unseen data. Furthermore, it can be concluded that the preprocessing step including feature selection and dimensionality reduction that were applied on each of the datasets allowed the respective models to be able to perform well in detecting malicious URLs and intrusion. Table 4 compares the proposed work to other works that use similar datasets. As can be seen, the proposed work outperformed the previous ones.

Table 4. Models' Performance Comparison with previous works

Detected Attack Categories	**Dataset Used**	**Classifier Used**	**Results**
Attack types and bad URLs [8]	Real life dataset collected by the authors	SVM	Accuracy: 93% (Attack types), 98% (Malicious URLs)
Malicious URLs [9]	Public dataset (2.4 million URLs)	RF	Accuracy: 97.69%
Malicious URLs and network features [10]	Public dataset (1,782 URLs)	SVM	Accuracy: 96% F1-Score: 92%
Domains that contain malicious content [11]	Built a dataset	LR	Accuracy: 89%
Malicious URLs	Dataset ***A***	KNN	Accuracy: 99.88% F1-Score: 97%
TCP_SYN and Port scan	Dataset ***B***	Decision Tree	Accuracy: 95.09% F1-Score: 78% - 100%

VI. CONCLUSION

In summary, this paper revealed that using ML classification techniques was highly effective in detecting malicious URLs, achieving accuracies over 99%. For intrusion detection, the DT classifier proved to be the most suitable with an accuracy of 95%. These findings highlight the significant potential of ML techniques in the field of cybersecurity. Implementing these techniques can enhance website security and effectively defend against harmful cyberattacks. The study underscores the need for a diverse range of ML algorithms to improve the accuracy and dependability of security systems. By leveraging the strengths of different algorithms, comprehensive and robust security solutions can be developed to combat evolving cyber threats. In the future, the aim is to extend the work to consider more types of security attacks. This can be done by training the ML models on datasets that include other attacks. Moreover, the proposed method can be implemented and used in different security systems to provide real-time protection of general users against malicious URLs and other security concerns.

REFERENCES

[1] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: Learning to detect malicious web sites from suspicious URLs," Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., pp. 1245–1253, 2009.

[2] P. Sangkatsanee, N. Wattanapongsakorn, and C. Charnsripinyo, "Practical real-time intrusion detection using machine learning approaches," Computer Communications, vol. 34, no. 18, pp. 2227–2235, 2011.

[3] Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. 2009. Identifying suspicious URLs: an application of large-scale online learning. In Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09). Association for Computing Machinery,NewYork,NY,USA,681–688.

[4] Haotian Liu, Xiang Pan and Zhengyang Qu, "Learning based Malicious Web Sites Detection using Suspicious URLs", Proc. of the 34th Internationai Conference on Software Engineering, 2009.

[5] M. Wu and M. Yang, "Privacy Preservation for Detecting Malicious Web Sites from Suspicious URLs," 2011, International Conference on Business Computing and Global Informatization, Shanghai, China, 2011, pp. 400-403, doi: 10.1109/BCGIn.2011.106.

[6] A. Lakshmanarao, M. R. Babu, and M. M. Bala Krishna, "Malicious URL detection using NLP, machine learning and Flask," 2021 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES), 2021.

[7] M. Almseidin, M. Alzubi, S. Kovacs, and M. Alkasassbeh, "Evaluation of machine learning algorithms for Intrusion Detection System," 2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY), 2017.

[8] H. Choi, B. Zhu, and H. Lee, "Detecting Malicious Web Links and Identifying Their Attack Types," in Proceedings of the 2nd USENIX Conference on Web Application Development (WebApps '11), Portland, OR, USA, 2011, pp. 4-4.

[9] F. Vanhoenshoven, G. Napoles, R. Falcon, K. Vanhoof, and M. Koppen, "Detecting malicious urls using machine learning techniques," 2016 IEEE Symposium Series on Computational Intelligence (SSCI), 2016. doi :10.1109/SSCI.2016.7850079

[10] S. Kaddoura, "Classification of malicious and benign websites by network features using supervised machine learning algorithms," 2021 5th Cyber Security in Networking Conference (CSNet), Abu Dhabi, United Arab Emirates, 2021, pp. 36-40, doi: 10.1109/CSNet52717.2021.9614273.

[11] J. C. Prieto, A. Fernandez-Isabel, I. M. De Diego, F. Ortega, and J. M. Moguerza, "Knowledge-based approach to detect potentially risky websites," IEEE Access, vol. 9, pp. 11633–11643, 2021.

[12] A. K. Singh, "Dataset of malicious and benign webpages," Kaggle, 04-Apr-2020.[Online].Available: https://www.kaggle.com/datasets/aksingh2411/dataset-of-malicious-and-benign-webpages?resource=download&select=Webpages_Classification_train_data.csv. [Accessed: 07-Feb-2023].

[13] G. Dutt, "Network Intrusion Detection," 2020, Kaggle. [Online]. Available: https://www.kaggle.com/datasets/gauravduttakiit/network-intrusion-detection?resource=download. [Accessed: April 29, 2023].

[14] I. Ul Hassan, R. H. Ali, Z. Ul Abideen, T. A. Khan, and R. Kouatly, "Significance of machine learning for detection of malicious websites on an unbalanced dataset," Digital, vol. 2, no. 4, pp. 501–519, 2022.

[15] Y. Liao and V. R. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection," Computers & Security, vol. 21, no. 5, pp. 439–448, 2002.

[16] S. Pradhan and S. K. Nayak, "An Analysis of SVM and NN Classifiers for Large Scale Datasets," 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), 2019, pp. 475-480.

[17] G. S. Handelman, H. K. Kok, R. V. Chandra, A. H. Razavi, S. Huang, M. Brooks, M. J. Lee, and H. Asadi, "Peering into the black box of artificial intelligence: Evaluation metrics of machine learning methods," American Journal of Roentgenology, vol. 212, no. 1, pp. 38–43, 2019.