HUMAN-IN-THE-LOOP FOR MACHINE LEARNING IN OFFENSIVE CYBERSECURITY

1st Satida Ruengsurat
Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
Ishikawa, Japan
satida.rue@jaist.ac.jp

2nd Jaimai Eawsivigoon
Mahidol University International College
Mahidol University
Nakhon Pathom, Thailand
jaimai.eaw@student.mahidol.edu

3rd Vidchaphol Sookplang
Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
Ishikawa, Japan
vidchaphol@jaist.ac.jp

4th Karin Sumongkayothin
Faculty of Engineering
Mahidol University
Nakhon Pathom, Thailand
karin.sum@mahidol.edu

5th Prarinya Siritanawan
Graduate School of Science and Technology
Shinshu University
Nagano, Japan
prarinya@shinshu-u.ac.jp

6th Razvan Beuran
Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
Ishikawa, Japan
razvan@jaist.ac.jp

7th Kotani Kazunori
Faculty of Transdisciplinary Science
Kanazawa University
Ishikawa, Japan

***Abstract*—Penetration testing is one of the methods that is used to find the exploitable vulnerabilities so that we are able to fix those vulnerabilities. Intrusion detection system is one of the defensive systems that needs to be improved all the time to prevent the intruders or the cyber criminals from bypassing the detection system and stealing important and valuable information or data. Nowadays, there are automation tools that are used to support the adversarial attack on intrusion detection systems. However, those tools may have some errors that, even if they can not be detected by intrusion detection systems, they can be seen by human experts. Including the human experts in the development of the tool therefore is a way to improve the attack performance and decrease the errors of the tool. In this study, we developed a model that mimics normal traffic behavior while also being capable of evading existing detection systems, with human expert assistance to improve the performance. The result of the experiment shows that the model successfully decreases the detection rate and the performance of the attack is up to the attack types. Moreover, with the help from the expert in developing the model and in the attack process, the errors in the tool are reduced and the performance of the attack is increased.**

***Index Terms*—Human-in-the-Loop, Adversarial crafted traffic, Machine learning**

I. Introduction

The weakness in the identification and authentication system can cost a great deal of money to an organization. People with malicious intentions use this vulnerability to bypass the detection system which leads to loss of data or damage to the system. Therefore, it is important to detect those malicious attempts in order to prevent them from damaging the system.

With the goal of developing effective threat detection and prevention systems, artificial intelligence (AI) is implemented in cybersecurity. Systems with AI are capable of analyzing network traffic, security records, and user behavior to build baselines, spot abnormalities, and spot possible insider threats or unauthorized access attempts. AI is also useful in automating incident response procedures, enabling quick responses to isolate affected systems, stop malicious activity, and implement corrective measures.

A sizable number of pre-made algorithms are included in machine learning, which is a subset of AI that may be applied to datasets to get insightful data insights. These algorithms have been improved throughout time to operate on a wide range of various datasets [1]. ML in cybersecurity is emerging as the next generation of tools for detecting and preventing malicious actions by analyzing large volumes of data, identifying patterns, and detecting anomalies that may indicate an ongoing attack. Furthermore, leveraging these algorithms enable system to mitigate malicious action after detecting them. In addition, the use of ML in the detection system can be beneficial in various aspects as it is faster, more efficient,

and continuously operating, unlike humans who have limited energy and resources [2].

On the other hand, the malicious actor continues to develop new offensive tools to bypass the ML-incorporated detection systems [3]. ML can also be used to improve offensive strategies by using adversarial attack technique to avoid detection and exploit vulnerability [4]. While ML algorithms are effective for improving the performance of these attacks against current detection systems, they still face some challenges. One significant challenge is that these improved attacks can be detected by experts in the cybersecurity field. For instance, malicious packages in the network might be recognized by experts who are familiar with various attack methods. By involving them in the system, they can guide ML to create attacks that are less noticeable by the experts in defensive cybersecurity field. Hence, the integration of human expertise into AI systems becomes crucial.

Human-in-the-Loop (HITL) is a technique that refers to the integration of human expertise and decision-making within AI systems. For instance, in the healthcare industry, HITL is also utilized to help clinicians diagnose medical disorders with the use of AI systems, with the recommendations of the systems being verified and improved by human specialists [5]. It is emphasized in the research that the synergy produced by the use of HITL approaches, which combines human judgment with AI, improves decision-making and system flexibility by overcoming the constraints of each component acting alone [6].

This research proposes offensive method towards ML detection models using Human-in-the-Loop (HITL) technique which can improve the attack ability, making the attack blend with benign package and less visible to the human experts.

The remainder of this paper is organized as follows. Section 2 reviews existing works related to offensive cybersecurity on machine learning and human-in-the-loop technique. Section 3 explains how our system works. Section 4 presents the metrics for evaluation. Section 5 presents the results of our experiment and Section 6 offers conclusions of our study.

II. Literature Review

Machine learning plays a vital role in user and entity behavior analytics, where they establish baselines of normal behavior and detect deviations that may indicate insider threats or compromised accounts. Furthermore, to allow the detection system to separate suspicious behavior from all actions, experts applied the classification ML algorithm to the systems [7].

However, with the trend of AI and ML being used in defensive cybersecurity, ML is being applied to attacking techniques on the Offensive side as well in order to improve efficiency and develop a new method of attacking. Moreover, to be able to bypass more robust defense systems successfully as an ML-based detection system, the attacking methods that can trick the ML anomaly detectors are developed called 'Adversarial Attacks'.

Adversarial Attacks is an attacking technique that refers to creating adjustments or modifications made to input data in order to fool or mislead machine learning algorithms which can cause the ML model to make an incorrect prediction which can lead to vulnerabilities in the system. It sometimes involves inserting precisely constructed adjustments or noise into the input data, which may be unnoticeable to humans but can have a major influence on the model's output. The goal of adversarial attacks is to exploit vulnerabilities or weaknesses in the model's decision-making process, potentially leading to security breaches or incorrect outcomes. Many research shows that adversarial attacks are able to break the ML defense system.

For instance, Ravi Chauhan at el. [8] proposes a model using Generative adversarial networks to generate adversarial DDoS attacks that can change the attack profile and can be undetected. The working mechanism of their attack is to use a generator to create an adversarial attack, then collect the detected results from the Intrusion detection system(IDS) and give the feedback to the attack generator to create a better result. Li, Heng at el. [9] proposed a new adversarial-example attack technique as a black-box attack that can evade both malware detection and adversarial detection to break through the firewall of the Android malware detection system.

As detection models advance, certain tools have been developed to deceive these detectors, making attacks appear as normal traffic. In this research, we have decided to use a tool called "Traffic Manipulator" [10] as a base to create an attacking model. Traffic Manipulator is an advanced black-box traffic mutation tool designed to skillfully and resourcefully create adversarial traffic [10]. Its primary purpose is to outsmart learning-based Network Intrusion Detection Systems (NIDS) while maintaining the original functionality of the network.

Similarly, Aritran Piplai at el. [3] proposed Generative Adversarial Network (GAN) based algorithm to generate data to train an efficient neural network-based classifier, and then break their system using adversarial attack. This shows that even a well-developed classifier can still be vulnerable to adversarial attacks.

However, creating adversarial attacks in the network domain faces significant limitations due to the restricted feature space available for mutation or modification [11]. Previous works on developing adversarial attacks often overlook these constraints, which resulted in their attacks being impractical and reducing their functionality. Therefore, having an expert to guide the model on this weakness is crucial for verifying the validity and correctness of prediction while also improving the performance of the ML.

For the professional fields that lack training data [12], human knowledge is a very effective aid as the pre-training knowledge for the machine learning model. This is where the "human-in-the-loop" term was introduced. Human-in-the-loop (HITL) is the concept of integrating human knowledge and experience into machine learning(ML) in order to improve the accuracy of the prediction using the minimum cost. In addition, HITL processes can reduce error and bias in machine

outputs, as humans can verify the results and provide feedback during learning processes to ensure the validity of the outputs [13]. A great number of research on HITL have been published and the trend is increasing every year [6]. HITL in the ML concept is being used in various systems including security systems such as fraudulent information filtering and authentication attack detection [14], [15]. Moreover, the enhancement of performance is seen in most of the models using HITL [6] which can indicate the effectiveness of human teaching performance on the machine learning results.

The combination of adversarial attacks with HITL approaches is a relatively new and challenging area for developing machine learning-based attack models. Therefore, we focus on applying HITL technique to the attacking model in order to improve the performance and validity of the attack to evade the detection system while also avoiding being detected by human experts evaluation.

III. Proposed Method

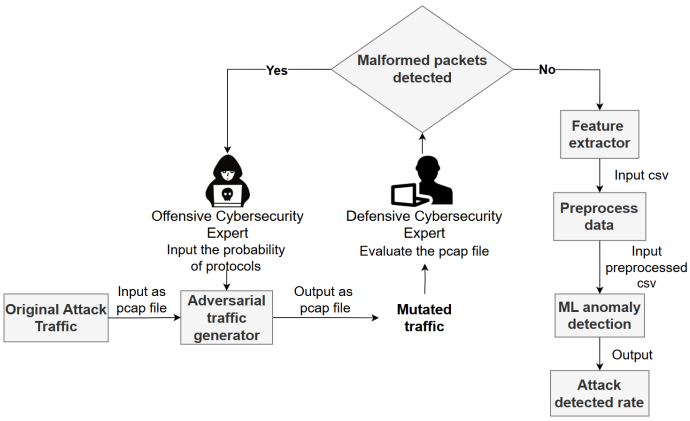

Fig. 1. The flowchart of how the system works

The proposed system workflow of this study is illustrated in Fig. 1. Firstly, we input original attack traffic into the adversarial traffic generator, producing mutated traffic as an output. We decided to use the Traffic Manipulator tool as the adversarial traffic generator. Traffic Manipulator generates adversarial examples from the original attack traffic assuming that an adversary has no internal information of the detection model. This tool uses the Particle Swarm Optimization (PSO) algorithm to search for approximate solutions in the high-dimensional discrete traffic space. The Traffic Manipulator generates the crafted packets, alters the interarrival time, protocol layer, and payload size of the crafted packets, and adds them to the original malicious traffic to create the mutated traffic. The features extracted from the mutated traffic by the extractor will be as similar as possible to the target feature set. The output of this tool is the mutated traffic, which imitates the normal traffic to bypass the anomaly detection model.

However, the Traffic Manipulator has some limitations. The protocols of the crafted packets are randomly chosen from the existing protocol layer of the original packets. The payload of the crafted packets is added randomly without considering the layout and protocol of the packet. Since the crafted packets added to the original traffic are generated with random protocols and random payload, most of the crafted packets added to the traffic are invalid packets. Although the attack of the mutated traffic may be able to bypass through the anomaly detection model, it might still be detected by human experts. In Fig. 2, the crafted packet has an error related to the invalid payload length. This error makes the crafted packet easily detectable by human experts. Therefore, we modified the Traffic Manipulator code to increase the practicability by letting the offensive cybersecurity expert input the probability of the protocol they focus on when adding the crafted packets. The protocols available are TCP, UDP, ICMP, IP, IPv6, ARP, and Ethernet.

We also added some code to categorize the original packets according to the protocol into lists and keep them as the template for the crafted packets. Lastly, we added the code so that the tool will randomly choose a template from the list according to the input protocol to create crafted packets instead of generating packets and payloads randomly. After some modifications, the crafted packets no longer appear with error warnings, making mutated malicious traffic more difficult for human experts to detect.

Then, we let the defensive cybersecurity expert evaluate the mutated traffic generated by the adversarial traffic generator. The experts check if there is any error warnings or strange packets that is visible to their eyes. If there are malformed packets, the offensive cybersecurity expert will notify the adversarial traffic generator to regenerate the mutated traffic. In this process, the offensive cybersecurity expert might change the probability of the protocol to adjust the outcome traffic. However, if no error is detected, the mutated traffic will be extracted into features.

The traffic needs to be extracted for testing the performance of the attack. The mutated traffic will be used as the input of the detection model to see the number of attacks detected compared to the original traffic. However, while the detection model needs the features in the form of a CSV file as input, the output of the Traffic Manipulator is in the form of a PCAP file. Therefore, we use CICflowmeter to extract the necessary features from mutated traffic PCAP files and convert them to CSV files. We then use the features of the CSV file to perform the data preprocessing. We preprocess the extracted mutated traffic by removing the rows containing ”NaN” or ”infinity” values, as well as excluding the columns containing solely zeros and non-numerical data. Then, we drop some features to match the input features of the detection model. The preprocessed mutated traffic is used as an input of the anomaly detection model.

We use the detection model that we trained from the CICIDS2017 dataset to test whether the method of generating adversarial crafted traffic decreases the chance of being detected by the anomaly detector. We preprocess the dataset, perform feature importance calculations, and identify highly correlated pairs to select the features used for training. The

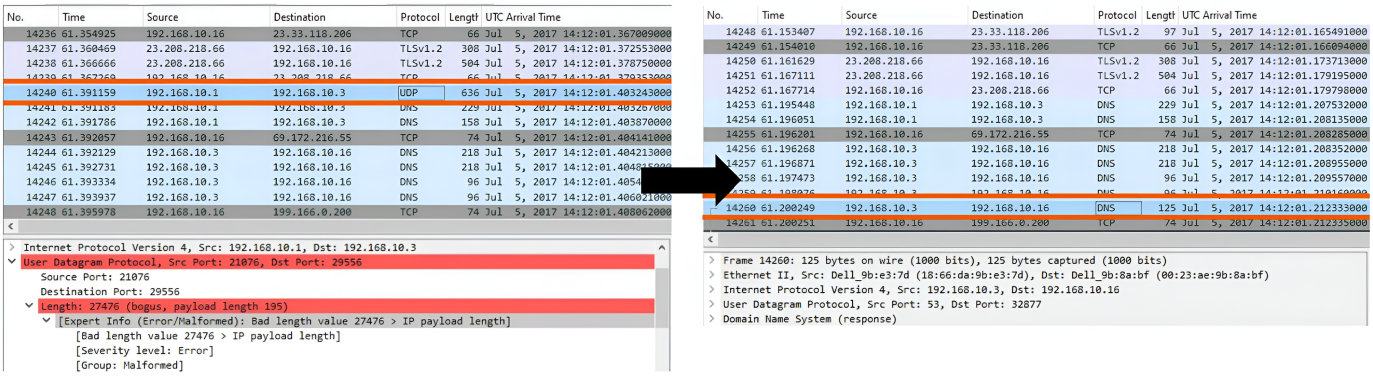

Fig. 2. Mutated traffic before and after modifying Traffic Manipulator

highlighted features in Fig. 3 are those selected for use.

no.	Feature name	no.	Feature name	no.	Feature name	no.	Feature name
1	Flow ID	22	Flow Packets/s	43	Fwd Packets/s	64	Fwd Avg Packets/Bulk
2	Source IP	23	Flow IAT Mean	44	Bwd Packets/s	65	Fwd Avg Bulk Rate
3	Source Port	24	Flow IAT Std	45	Min Packet Length	66	Bwd Avg Bytes/Bulk
4	Destination IP	25	Flow IAT Max	46	Max Packet Length	67	Bwd Avg Packets/Bulk
5	Destination Port	26	Flow IAT Min	47	Packet Length Mean	68	Bwd Avg Bulk Rate
6	Protocol	27	Fwd IAT Total	48	Packet Length Std	69	Subflow Fwd Packets
7	Timestamp	28	Fwd IAT Mean	49	Packet Length Variance	70	Subflow Fwd Bytes
8	Flow Duration	29	Fwd IAT Std	50	FIN Flag Count	71	Subflow Bwd Packets
9	Total Fwd Packets	30	Fwd IAT Max	51	SYN Flag Count	72	Subflow Bwd Bytes
10	Total Backward Packets	31	Fwd IAT Min	52	RST Flag Count	73	Init_Win_bytes_forward
11	Total Length of Fwd Packets	32	Bwd IAT Total	53	PSH Flag Count	74	Init_Win_bytes_backward
12	Total Length of Bwd Packets	33	Bwd IAT Mean	54	ACK Flag Count	75	act_data_pkt_fwd
13	Fwd Packet Length Max	34	Bwd IAT Std	55	URG Flag Count	76	min_seg_size_forward
14	Fwd Packet Length Min	35	Bwd IAT Max	56	CWE Flag Count	77	Active Mean
15	Fwd Packet Length Mean	36	Bwd IAT Min	57	ECE Flag Count	78	Active Std
16	Fwd Packet Length Std	37	Fwd PSH Flags	58	Down/Up Ratio	79	Active Max
17	Bwd Packet Length Max	38	Bwd PSH Flags	59	Average Packet Size	80	Active Min
18	Bwd Packet Length Min	39	Fwd URG Flags	60	Avg Fwd Segment Size	81	Idle Mean
19	Bwd Packet Length Mean	40	Bwd URG Flags	61	Avg Bwd Segment Size	82	Idle Std
20	Bwd Packet Length Std	41	Fwd Header Length	62	Fwd Header Length	83	Idle Max
21	Flow Bytes/s	42	Bwd Header Length	63	Fwd Avg Bytes/Bulk	84	Idle Min

Fig. 3. Features in the CICIDS2017 dataset where the highlighted are selected features

We used 3 detection models: the Deep Neural Network (DNN) model, the Random Forest model, and the Support Vector Machine (SVM) model. The reason we decided to use these models is that we wanted to test the performance of the attack on both the neural network model and the classical models. The output of the detection model is the classification result if the records are classified as abnormal or benign.

In our study, human experts contribute to improving the performance of the system. Firstly, the expert will be in the process of modifying the Traffic Manipulator tool to guarantee that the content of the crafted packets is according to the protocol and that the crafted packets will not be invalid. The expert will determine the probability of protocol for the crafted packets corresponding to each type of attack, as each type requires a specific combination of protocols. Lastly, the expert will check through the packets of a mutated traffic PCAP file generated by the Traffic Manipulator to see if the traffic looks normal or not. The experts in our study must be the offensive or defensive cybersecurity experts who specialize in network security.

IV. Evaluation Metrics

The behavior of the mutated traffic detected by the detection system will be categorized into normal and abnormal behavior. If the behavior invalidates the network traffic features, it will be labeled as abnormal.

In this study, we use the attack-detected rate, which is the rate of attack records detected compared to the total attack records in the original traffic, as the measurement to evaluate the performance of our model. If the attack-detected rate of mutated traffic is lower than the attack-detected rate of the original traffic, it can be interpreted as our model successfully mimicking normal traffic behavior and evading the detection system. The attack-detected rate can be written as Equation 1:

$$r = \frac{p}{t} \cdot 100 \tag{1}$$

where r is attack-detected rate; p is predicted attack count and t is total attack count.

V. Experiment Results

This section initializes testing phase of the detection models and presents the results of testing the modified Traffic Manipulator tool against these models.

A. Initial test of the detection models

We assess the performance of the detection models using unmodified datasets to validate their effectiveness, demonstrating the capabilities of Random Forest, SVM, and DNN models. Firstly, we trained and tested the Random Forest model, achieving an accuracy of 0.999, a precision of 0.999, a recall of 0.990, and an f1-score of 0.999.

Next, we trained and tested the SVM model, achieving an accuracy of 0.954, a precision of 0.955, a recall of 0.954, and an f1-score of 0.954.

Finally, we trained and tested the Deep Neural Network (DNN) model, achieving an accuracy of 0.989, a precision of 0.984, a recall of 0.994, and an f1-score of 0.989.

Judging from the various metrics, the Random Forest model outperformed the other models, followed by the Deep Neural Network and then the SVM.

B. Testing the effect of the probability of protocol proportion to output of the Traffic Manipulator

The output of the Traffic Manipulator tool is tested using the detection models. Given that the models have already

demonstrated strong performance, as indicated in Section A. We evaluated the tool's effectiveness by comparing the predicted labels from the detection models for both the original data and the tool's output, categorized by attack type. The paper of Traffic Manipulator primarily focused on pattern-based attacks so we picked DoS Goldeneye for verification. To expand our research, we also tested two protocol-based attacks: SQL Injection and Heartbleed. The objective is to determine the best probability of protocol proportion for each attack type.

For the DoS Goldeneye attack, there are 5,148 records in the original traffic dataset. From the original traffic, the DNN detection model predicts 1,743 attack records (33.86% of the original traffic), the Random Forest model predicts 2,789 attack records (54.18% of the original traffic), and the SVM predicts 639 attack records (12.41% of the original traffic).

In Table1, the best probability of protocol proportion for the DNN model is 70% TCP, 15% UDP, and 15% IP, with 444 attack records predicted (8.62% of the original traffic). For the Random Forest model and SVM model, the best protocol proportion is 34% TCP, 33% UDP, and 33% IP. The Random Forest model predicts 246 attack records (4.78% of the original traffic), while the SVM predicts 463 attack records (8.99% of the original traffic).

In conclusion, the results indicate that the mutated traffic from the Traffic Manipulator can evade the detection models effectively compared to original traffic.

For the SQL Injection attack, there are 21 records in the original traffic dataset. From this traffic, the DNN detection model predicts 64 attack records (304% of the original traffic), the Random Forest model predicts 6 attack records (28.57% of the original traffic), and the SVM predicts 205 attack records (976% of the original traffic).

In Table2, for the DNN detection model, there are multiple best combinations, each resulting in 56 predicted attack records (266% of the original traffic). However, the probability of protocol proportion appears to be insignificant for the Random Forest model where the number of attack records detected in the mutated traffic remains unchanged compared to the original traffic. For the SVM model, the best combination is 25% UDP, 25% IP, and 50% ARP, reducing the predicted attack records to 109 (519% of the original traffic).

In summary, while the Traffic Manipulator can mutate SQL Injection traffic and evade detection models, it is less effective compared to its performance with DoS Goldeneye.

For the Heartbleed attack, there are 11 records in the original traffic dataset. From this traffic, the DNN detection model predicts 462 attack records (4200% of the original traffic), the Random Forest model predicts 31 attack records (281.81% of the original traffic), and the SVM predicts 2,114 attack records (19218.18% of the original traffic).

In Table3, the best protocol proportion combination for both the DNN and Random Forest models is 15% TCP, 15% UDP, and 70% IP, with the DNN predicting 859 attack records (7809.09% of the original traffic) and the Random Forest predicting 23 attack records (209% of the original traffic). For the SVM, the best combination is 50% TCP, 25% UDP, and 25% IP, with 1,142 attack records predicted (10381.81% of the original traffic).

Aside from the DNN, the number of predicted attack records in the mutated traffic is reduced compared to the original.

The number of attacks predicted by the models for SQL Injection and Heartbleed exceeds the number of traffic records in the original dataset. This may be due to the limited number of attack records for certain attack types in the training set. The distribution of attack records in the CICIDS2017 dataset is unbalanced, with only a small number of records for some attack types, such as SQL Injection and Heartbleed, and a larger number of records for others, like DoS attacks. This imbalance leads to poor model performance for certain types of attacks.

VI. Conclusion

In the field of offensive cybersecurity, autonomous attack generators are designed to evade detection by machine learning models. Despite this, human experts are often capable of identifying anomalies in the generated traffic. Our goal is to develop a model which generates attacks that simulate normal traffic behavior, incorporating human assistance to enhance its performance.

The original tool struggles because its crafted packets are easily spotted by humans. On the contrary, our improved model uses original traffic content to create more convincing packets, thereby reducing detectability by human observers. Additionally, our model allows experts to adjust the probability of different protocol proportions of the mutated packets.

The model is evaluated using the CICIDS2017 dataset, employing Random Forest, SVM, and DNN detection models. The testing process starts with our adversarial traffic generator mutating the original traffic into a mutated traffic. This mutated traffic is then assessed against the detection models, which have been trained on the dataset, to determine whether the traffic is classified as an attack. We measure the model's performance based on the attack-detected rate written in Equation 1.

Our test results indicate that the output from our adversarial traffic generator no longer contains invalid packets detectable by experts, unlike the original version. Our adversarial traffic generator demonstrates a success in evading detection models for pattern-based attacks, as evidenced by the reduction in detection rates. For the DNN model, the attack-detected rate reduces from 33.86% to 8.62%. For the Random Forest model, the attack-detected rate reduces from 54.18% to 4.78% and for the SVM model, the attack-detected rate reduces from 12.41% to 8.99%. However, it is less effective against command-based attacks. Moreover, with expert involvement in tuning the probability protocols proportion, the performance of our adversarial traffic generator can be further improved.

References

[1] D. Wankhede, "Artificial intelligence and its subsets: Machine learning and its algorithms, deep learning, and their future trends," vol. 9, pp. page no.i112–i117, 06 2022.

TABLE I
BEST PROTOCOL PROPORTION FOR DOS GOLDENEYE ON DIFFERENT DETECTORS

Detector	**Protocol (%)**							**Attack Records Detected in Original Traffic**	**Attack Records Detected in Mutated Traffic**
	TCP	UDP	ICMP	IP	IPv6	ARP	Ethernet		
DNN	70	15	0	15	0	0	0	1743	444
Random Forest	34	33	0	33	0	0	0	2789	246
SVM	34	33	0	33	0	0	0	639	463

TABLE II
BEST PROTOCOL PROPORTION FOR SQL INJECTION ON DIFFERENT DETECTORS

Detector	**Protocol (%)**							**Attack Records Detected in Original Traffic**	**Attack Records Detected in Mutated Traffic**
	TCP	UDP	ICMP	IP	IPv6	ARP	Ethernet		
DNN	0	25	0	50	0	25	0	64	56
	0	25	0	25	0	50	0	64	56
	0	70	0	15	0	15	0	64	56
Random Forest	Same for all protocol proportion combination							6	6
SVM	0	25	0	50	0	25	0	205	109

TABLE III
BEST PROTOCOL PROPORTION FOR HEARTBLEED ON DIFFERENT DETECTORS

Detector	**Protocol (%)**							**Attack Records Detected in Original Traffic**	**Attack Records Detected in Mutated Traffic**
	TCP	UDP	ICMP	IP	IPv6	ARP	Ethernet		
DNN	15	15	0	70	0	0	0	462	859
Random Forest	15	15	0	70	0	0	0	31	23
SVM	50	25	0	25	0	0	0	2114	1142

[2] N. Kühl, M. Goutier, L. Baier, C. Wolff, and C. Wolff, "Human vs. supervised machine learning: Who learns patterns faster?" *Cognitive System Research*, vol. 76, pp. 78–92, 2022.

[3] A. Piplai, S. S. L. Chukkapalli, and A. Joshi, "Nattack! adversarial attacks to bypass a gan based classifier trained to detect network intrusion," in *2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, 2020, pp. 49–54.

[4] M. Ozay, I. Esnaola, F. T. Y. Vural, S. R. Kulkarni, and H. V. Poor, "Machine learning methods for attack detection in the smart grid," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 8, pp. 1773–1786, aug 2016. [Online]. Available: https://doi.org/10.1109%2Ftnnls.2015.2404803

[5] A. Ganesan, A. Paul, G. Nagabushnam, and M. J. J. Gul, "Human-in-the-loop predictive analytics using statistical learning," *Journal of Healthcare Engineering*, vol. 2021, 2021.

[6] X. Wu, L. Xiao, Y. Sun, J. Zhang, T. Ma, and L. He, "A survey of human-in-the-loop for machine learning," *Future Generation Computer Systems*, vol. 135, pp. 364–381, 2022.

[7] G. Abdiyeva-Aliyeva, J. Aliyev, and U. Sadigov, "Application of classification algorithms of machine learning in cyber-security," *Procedia Computer Science*, vol. 215, pp. 909–919, 2022, 4th International Conference on Innovative Data Communication Technology and Application. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050922021640

[8] R. Chauhan and S. Shah Heydari, "Polymorphic adversarial ddos attack on ids using gan," in *2020 International Symposium on Networks, Computers and Communications (ISNCC)*, 2020, pp. 1–6.

[9] H. Li, S. Zhou, W. Yuan, J. Li, and H. Leung, "Adversarial-example attacks toward android malware detection system," *IEEE Systems Journal*, vol. 14, no. 1, pp. 653–656, 2020.

[10] D. Han, Z. Wang, Y. Zhong, W. Chen, J. Yang, S. Lu, X. Shi, and X. Yin, "Evaluating and improving adversarial robustness of machine learning-based network intrusion detectors," *IEEE Journal on Selected Areas in Communications*, pp. 1–1, 2021.

[11] R. Sheatsley, N. Papernot, M. J. Weisman, G. Verma, and P. D. McDaniel, "Adversarial examples in constrained domains," *CoRR*, vol. abs/2011.01183, 2020. [Online]. Available: https://arxiv.org/abs/2011.01183

[12] X. Zhang, S. Wang, J. Liu, and C. Tao, "Towards improving diagnosis of skin diseases by combining deep neural network and human knowledge," *BMC Medical Informatics and Decision Making*, vol. 18, 07 2018.

[13] O. Gómez-Carmona, D. Casado Mansilla, D. López-de Ipiña, and J. Garcia-Zubia, "Human-in-the-loop machine learning: Reconceptualizing the role of the user in interactive approaches," *Internet of Things*, vol. 25, p. 101048, 04 2024.

[14] D. Odekerken and F. Bex, *Towards Transparent Human-in-the-Loop Classification of Fraudulent Web Shops*, 12 2020.

[15] J. Brown and M. Anwar, "Blacksite: human-in-the-loop artificial immune system for intrusion detection in internet of things," *Hum.-Intell. Syst. Integr.*, vol. 3, pp. 55–67, 2021.