

# Explainable Malware Detection Using Predefined Network Flow

Boryau Hsupeng\*, Kun-Wei Lee\*, Te-En Wei\*, Shih-Hao Wang\*

\*Cybersecurity Technology Institute, Institute for Information Industry, Taiwan, R.O.C

bojsahsupeng@iii.org.tw, kwlee@iii.org.tw, dwyanewei@iii.org.tw, shwang@iii.org.tw

**Abstract**—As the internet has become an indispensable part of modern life, defences against cybersecurity attacks have become an important topic and a considerable number of studies have been made to provide reliable tactics to defend against cyberattacks. Flow export protocols and technologies provide several advantages in network monitoring. By using flow data aggregated from packets, the amount of data to be analysed has been significantly reduced and it is often said to be more scalable than packet-based traffic analysis. With the help of modern Artificial Intelligent algorithms, AI can be trained with flow data to predict hackers' attacks and types of malware. In this paper, we will present CSTITool, a CICFlowMeter-based flow extraction tool, to feature extraction with an aim of improving the model performance. The flow features will be used to train a machine learning-based model for hackers' attacks and malware classification. To provide interpretability, an explainable AI will be introduced to help understand the relation between the prediction and the features.

**Keywords**—Explainable AI, Cybersecurity, XGBoost, Malware Classification, SHAP

## I. INTRODUCTION

The public had no idea what the internet was at the end of the twentieth century. However, just a few years later, with the popularity and development of the Internet of things (IoT) and the Fourth Industrial Revolution (Industry 4.0), the internet has become an indispensable part of modern life. As more and more devices get connected to the internet, hackers get more and more opportunities to exploit network vulnerabilities. They unlawfully seek to break into secure networks and cyber-physical systems with the motive of stealing sensitive data, disrupting digital operations, or tampering with data.

For as long as there have been hackers, cybersecurity scientists and engineers have been proposing solutions to defend against potential attacks. Most traditional approaches use rule-based detection methods, e.g., Snort [2] and Suricata [3], to monitor the content of network packets in real time, and then identify information such as suspicious strings in the payload. Furthermore, with the rise of personal privacy awareness, encryption protocols (SSL/TLS) have been used for encrypting payload information in internet packets, which can prevent people with bad intentions from monitoring and stealing information. However, if hackers transmit malicious commands in encrypted packets, traditional methods cannot withstand such attacks. Another problem with traditional rule-based detection methods is that they are inefficient when used

with complex network environments. The number of network packets increases with the complexity of the network environment, which consumes enormous resources when compared with the rules, so that all the packet information of the monitored network environment cannot be analysed and processed immediately.

Many have proposed methods to solve the shortcomings of traditional packet detection methods. One of them is NetFlow, a concept developed by Cisco. NetFlow reduces consumption of storage space by aggregating packet information and discarding nonessential information, such as payload, then replacing original packet information with network traffic behaviour. Many have trained Artificial Intelligent (AI) with NetFlow data to build models of abnormal network behaviour for detecting anomalies (i.e. [11], [17]). By learning network traffic behaviour, AI can not only effectively identify known malicious threats, but also detect their variations. With the help of AI technology, rule-based detection methods can achieve greater performance.

As AI technology advances, many techniques and algorithms have been developed to improve the performance of AI in detecting hacker attacks. With the help of feature engineering techniques, machine learning, and deep learning, detection capability and prediction accuracy can be highly improved (i.e. [6], [8], [9], [13], [19], [20]). However, most studies focused on how to improve the model's performance in detection, but few were able to interpret the results. While this lack of information may have been acceptable in the past, today's users expect to be able to find out why and how AI has made a specific decision. This paper presents an explainable flow-data AI model to identify malicious behaviours and explain how these behaviours were classified.

## II. METHOD

This section describes the analysis methods used in this paper, which is divided into three subsections as follows: Sect. II.A presents the features extraction process from packets to network flow data along with several feature engineering techniques; Sect. II.B presents the use of a machine learning algorithm, XGBoost; and Sect. II.C introduces a model-agnostic method for interpreting predictions through SHapley Additive exPlanations (SHAP).

### A. Feature Extraction

NetFlow is widely used in network management tasks such as usage monitoring and anomaly detection. It provides a

powerful standard for flow data statistics by monitoring and recording traffic that passes in and out of an interface. While NetFlow is effective in identifying unusual network behaviours, it imposes several challenges in flow features extraction. For example, it does not record packet payload. However, a full discussion of NetFlow lies beyond the scope of this study and one can consult [12] for the details.

CICFlowMeter is used to obtain 82 statistical network traffic features. It is an open-source network traffic flow generator which generates Biflows from pcap files, and extracts features from these flows [5]. We also introduced CSTITool, a combination of CICFlowMeter and techniques that we referred from [4], [10], to extract 53 more features. The features were then grouped into 9 types by flow data characteristics, as illustrated on Table 1. The subcolumn CICFlowMeter shows the number of features generated by CICFlowMeter in each type, while the subcolumn CSTITool shows the features that were generated by CICFlowMeter along with other techniques. One would notice that the number of Flow Descriptors features of CSTITool is much higher than CICFlowMeter. This is because we added more descriptive flow statistics to this type. Rojas et al. [7] used a similar approach to extract 7 groups of attributes. Two new network data types are also added: Bytes Distribution and Domain Knowledge. The former records the total numbers of payload bytes in packet. The same type of hacking techniques or malicious network behaviours has higher probability of sharing the same Bytes Distribution. The latter are features that store statistics defined by experts.

TABLE 1.9 TYPES OF FLOW DATA

| Flow Data Type      | Description                                                                                                                                          | Tool         |          |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|----------|
|                     |                                                                                                                                                      | CICFlowMeter | CSTITool |
| Network Identifiers | These attributes store the 5-tuple information and timestamp of Internet flow.                                                                       | 6            | 8        |
| Flow Descriptors    | These attributes hold statistics of Internet flow. For example, total packet count, forward packet length, backward minimal packet length and so on. | 34           | 76       |
| Interarrival Times  | These attributes show the information related to the interarrival times in the forward and backward directions.                                      | 15           | 15       |
| Flag Features       | These attributes store the counts for each flag contained in the header of packets.                                                                  | 12           | 12       |
| Subflow Descriptors | If subflows exist, these attributes store the counts and bytes in the forward and backward direction.                                                | 4            | 4        |
| Header Descriptors  | These attributes store the information related to the header.                                                                                        | 3            | 3        |
| Flow Time           | These attributes store the information related to the time each flow was active and idle.                                                            | 8            | 8        |
| Bytes Distribution  | These attributes store the distribution of payload bytes.                                                                                            | 0            | 1        |
| Domain Knowledge    | These attributes store statistics defined by experts.                                                                                                | 0            | 8        |

## B. Model

The eXtreme Gradient Boosting (XGBoost) model [18], an advanced gradient enhanced tree and widely used to achieve accurate performance on various machine learning challenges, is selected to implement the classification of malicious behaviours. As a Gradient Boost Decision Tree (GBDT) algorithm, XGBoost has better accuracy than traditional GBDT

algorithms because it computes second-order gradients, i.e., second partial derivatives of the loss function.

An automatic optimization process is designed to help the model determine the most suitable algorithm structure in response to different environments. This automatic optimization process, aimed at improving the accuracy and stability of our model, is divided into two steps: hyperparameter optimization and important features selection. The structure and the performance of the model highly depend on the hyperparameters of XGBoost, so appropriate tuning is essential.

For the purpose of hyperparameter optimization, the grid search algorithm has been performed. We set bounds and discretization in value spaces for hyperparameters. For each of these hyperparameters, a finite set of reasonable values are selected in the value space. Grid search then trains an XGBoost model with the Cartesian product of all hyperparameter sets, and evaluates their performance. The combination of hyperparameters with best accuracy is then chosen for building the XGBoost model.

Feature selection, another important step in the optimization process, is the process of selecting a subset of relevant features to improve a predictive model. It is a technique that assigns a score to input features to evaluate how good the features are at predicting a target variable. The features that have a score higher than the threshold are selected for use in model construction. The feature importance scores are calculated by the XGBoost model to perform feature selection.

In the automatic optimization process, a variation of k-fold called stratified k-fold is used. Both stratified k-fold and k-fold can help avoid overfitting, however, the former is chosen over latter because in the scenarios where there is class imbalance, k-fold may return training subsamples with insufficient instances, while stratified k-fold returns stratified folds that each set contains approximately the same percentage of instances of each target class as the complete set.

## C. Explainable AI

It is often heard that machine learning models are black boxes---data goes in, decisions come out. They can make good predictions but the logic behind those predictions is incomprehensible, since the process between input and output is opaque. When XGBoost constructs the boosted trees, the importance scores for each feature are calculated. The scores indicate how valuable each feature is in the construction of the boosted decision trees within the model. Although this approach can tell us which features are used to make key decisions with decision trees, it still lacks explanation for individual predictions made by the model. This is where the explainable AI comes to the rescue.

Explainable AI (XAI) refers to a set of methods and techniques that helps data scientists better understand why a model makes certain decisions. In this paper, we utilize SHAP [14], [15], [16], an open-source tool, which provides a reliable approach to understand the relationship between the model and the data features. SHAP values are based on Shapley values, a solution concept in cooperative game theory. They are calculated by individually introducing each feature into a

conditional expectation function of the output of the model, then attributing the change produced at each step to the feature that was introduced, and then averaging this process over all possible feature orderings.

The sign of SHAP values indicates how it affects the prediction. When the value is positive, it means a feature has a positive effect, and vice versa. In this experiment, we will present three visualization methods provided by the Python SHAP package, i.e., bar plot, beeswarm plot, and force plot. Bar plot simply takes the mean absolute value of the SHAP values per feature across the data. It shows the SHAP feature importance in descending order. Beeswarm plot, also known as summary plot, gives an overview of which features are more important for a model by plotting the SHAP values of every feature for every instance. The position on the x-axis represents the SHAP value of an instance and the colour represents the value of the feature. For each feature, instances with the same SHAP value are placed along the y-axis, this gives us a sense of distribution of SHAP value. Force plot visualizes every feature as force that either increases or decreases the prediction by pushing it from the baseline for Shapley value, which is the average of the model output over training dataset, to the model output.

### III. EXPERIMENT AND RESULT

In this section, two experiments and their results will be presented. One is a hacker's attack classification model trained by CIC-IDS2017 dataset; the other is a malware classification model trained by CSTI-10 dataset. The rest of the section is organized as follows. First, we discuss the characteristics of features used in the experiments. Second, we offer the process of model construction. Third, we express the model performances. Fourth, the application results of explainable AI are described.

#### A. Dataset

Two datasets are used for modelling. One is an open-source dataset CIC-IDS2017 which records hacker attacks; the other is a combination of different datasets that record the network traffic behaviours of malwares, we refer to it as CSTI-10.

1) **CIC-IDS2017:** CIC-IDS2017 resembles real-world data which contains benign and common attacks. For this dataset, user behaviours were built based on the HTTP, HTTPS, FTP, SSH, and email protocols. We compare the classes of attack in CIC-IDS2017 with the corresponding techniques in MITRE ATT&CK framework based on the types of cyber adversary behaviour, so that the results of the model can be integrated with most products on the market. Network Service Scanning (T1046), Endpoint Denial of Service (T1499), Brute Force (T1110) and Remote Access Software (T1219), a total of four techniques and benign network flow data are selected for the follow-up experiments. The comparison and the counts of cases are shown in Table 2. Details regarding adversary techniques are listed in MITRE ATT&CK knowledge base [1].

2) **CSTI-10:** The network flow data of malwares have also been tested for experiments. Unlike the hacker attack data in

CIC-IDS2017, the network traffic behaviour of malwares is less obvious, so it is more challenging to interpret the prediction results of the model.

There are a total of ten classes of malwares in the CSTI-10 dataset. The types of the malwares are Trojan, Exploit Kits, Backdoor, Loader, and Ransomware. The packets of nine classes of these malwares are chosen from the open-source dataset CTU and a blog named Malware-Traffic-Analysis.net. The infamous ransomware (e.g., WannaCry) was also recorded and added into CSTI-10. The process of recording WannaCry packets was done by using Cuckoo Sandbox to create a virtual Windows 7 operating system, then executed the malware on the virtual environment, and used Wireshark to capture packets. Besides the traffic data of the malwares, we also collected the encrypted traffic in the general environment and added it to CSTI-10 as the benign category. After converting packets into flow data, the decrease in amount of data is more than 86%, which greatly reduces the hard disk space required for data storage.

TABLE 2. ATT&CK TECHNIQUE AND CIC-IDS2017 COMPARISON

| ATT&CK Technique                   | CIC-IDS2017 Label      | Counts    |
|------------------------------------|------------------------|-----------|
| -                                  | Benign                 | 2,273,097 |
| T1046 - Network Service Scanning   | PortScan               | 158,930   |
|                                    | DoS Hulk               |           |
|                                    | DDoS                   |           |
| T1499 - Endpoint Denial of Service | DoS GoldenEye          | 380,688   |
|                                    | DoS slowloris          |           |
|                                    | DoS Slowhttptest       |           |
|                                    | FTP-Patator            |           |
| T1110 - Brute Force                | SSH-Patator            | 15,342    |
|                                    | Web Attack Brute Force |           |
| T1219 - Remote Access Software     | Bot                    | 1,966     |

#### B. Modelling

1) **Data Preprocessing:** There are missing values in the data, such as NAN and Inf. In each category, the proportion of missing values is between 0.01% and 0.5%. Since the amount of missing values is negligibly small and have little impact on the experiment, they can be removed from the data. In order to verify the effectiveness of the trained model, the data are divided into training set and test set at a ratio of 7:3. The segmentation is processed according to the proportions of each category, that is, the training and test sets have the same proportion of categories as in the original data.

2) **Model:** XGBoost is the core algorithm of the experiment. It is trained to learn the characteristics of the data for building a multiclass classification model. When optimizing model parameters and selecting important features, Stratified K-fold is used, where K is set to 5. In addition, in order to reduce resource consumption and improve modelling efficiency, an early termination standard is set for model training. In the optimization process, in order to complete the optimization faster, early\_stopping\_round is set to 10. In the final model training, the early\_stopping\_round is set to 100 for higher accuracy in model prediction.

For the purpose of hyperparameter optimization, grid search is applied to find the most suitable combination of

hyperparameters for the model. Hyperparameter `max_depth` is set to be 2 and 5, while `n_estimators` is set to be 100, 300, 500, 700, and 900. Grid search then trains an XGBoost model with each of the 10 combinations of hyperparameters to find the most suitable combination.

To select important features, the feature importance score of the XGBoost model is used to perform feature selection. In the experiments, 'gain' is chosen as the feature importance type. The features whose gain value is 0 are removed, while remaining features are important features for constructing the XGBoost model.

For feature selection, we tested 79 features from the CIC-IDS2017 dataset (Source IP and Destination IP were not included). 69 features were considered as important features. We also applied feature selection to the CSTI-10 dataset and created two variations, CSTI-10-A and CSTI-10-B. For CSTI-10-A, we tested features obtained from CICFlowMeter and 68 were identified as important features. As for CSTI-10-B, a total of 377 features were obtained from CSTITool. A part of the increase in number of features (from 79 to 377) was contributed by One-Hot Encoding. After the feature selection process, only 142 features remain in CSTI-10-B.

**3) Performance:** Both experiments are multiclass classification problems. We choose Precision and Recall to evaluate performance. Precision can help us determine how many of the model classification results are correct; Recall can help us determine how many of the original categories are successfully classified by the model.

Table 3 shows the confusion matrix and performance metrics of the model trained with CIC-IDS2017 dataset. It can be found that the values of Precision and Recall are close to 1 regardless of the predicted category. For the category T1219, which has the least data among all categories, the Precision and Recall are 97.97% and 98.81% respectively. The confusion matrix indicates that the model for detecting hacker attacks which is trained with network flow data has good performance in identifying T1110, T1499, T1046, and T1219.

TABLE 3. PERFORMANCE OF CIC-IDS2017 MODEL

| True Value                         | Predicted Value |       |         |        | Metrics |               |
|------------------------------------|-----------------|-------|---------|--------|---------|---------------|
|                                    | Benign          | T1110 | T1499   | T1046  | T1219   |               |
| Benign                             | 687,210         | 1     | 111     | 2      | 12      | 1.0000 0.9998 |
| T1110 - Brute Force                | 3               | 4,599 | 0       | 0      | 0       | 0.9991 0.9993 |
| T1499 - Endpoint Denial of Service | 17              | 3     | 113,901 | 0      | 0       | 0.9990 0.9998 |
| T1046 - Network Service Scanning   | 6               | 0     | 5       | 47,630 | 0       | 1.0000 0.9998 |
| T1219 - Remote Access Software     | 7               | 0     | 0       | 0      | 580     | 0.9797 0.9881 |

Two malware classification models are trained based on CSTI-10-A and CSTI-10-B respectively, and illustrated the performance metrics in Figure 1. Both flow-data based models performed well in most classes with Precision and Recall higher than 90%. The CSTI-10-B model is expected to achieve better performance as it was built with more features. The extra features in CSTI-10-B help improve the performance metrics of most categories; for instance, Rig's Precision and Recall are increased by 1.23% and 1.59% respectively, and Recall value of Emotet is increased by 1.15%. Although the Recall value of Nuclear is reduced by 2.7% to 97.3%, the Precision and Recall value of Macro in CSTI-10-B model are higher than CSTI-10-A by 0.1 to 0.2%.

We investigated into three categories with the least accuracy in performance, i.e., Emotet, TrickBot, and Rig. 71 out of 1743 Emotet were misclassified as TrickBot, while 102 out of 5039 TrickBot were labelled as Emotet. It is found that Emotet is often used together with TrickBot, causing the model to learn TrickBot's behaviours and is responsible for the misclassification. Having only 186 cases, Rig's Recall is also low. It is because data insufficiency will not only make the model unable to learn the characteristics of this type of malware, but also reduce the performance metrics with a small amount of misclassification. In this experiment, there were only 189 Rig cases, 25 of which were incorrectly classified, resulting in only 86.24% of Recall value.

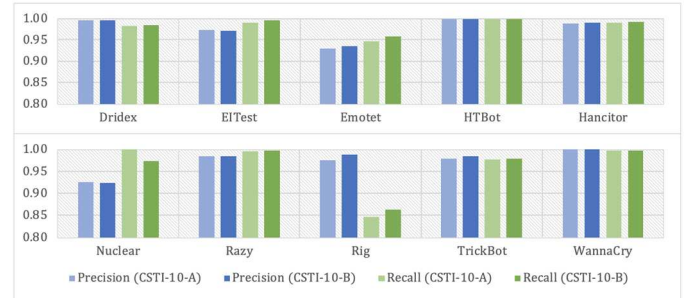


Figure 1. Performance metrics of CSTI-10-A and CSTI-10-B model

### C. Explainable AI

The experiments suggest that our models trained with network flow data are able to achieve good performance in both hacking and malware classification. In addition to the effectiveness, however, the users also expect transparency, interpretability, and explainability of the model. Here the open-source tool SHAP is applied to achieve this goal.

**1) CIC-IDS2017:** We calculate SHAP value of all CIC-IDS2017 features that are used to construct hacker's attacks classification model. The features with the top 10 SHAP values are depicted in a bar chart in Figure 2. In this plot the impact of a network flow feature on 5 classes (Benign, T1046, T1110, T1219, T1499) are stacked to create the feature importance plot.

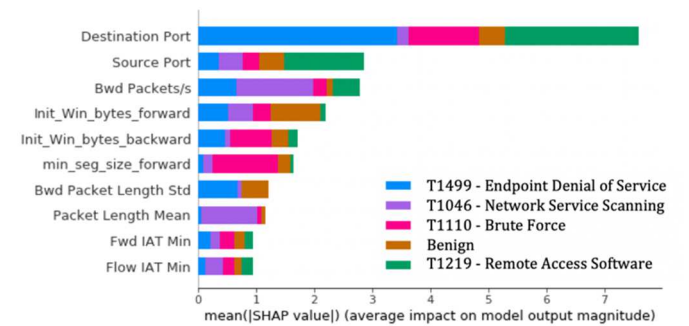


Figure 2. SHAP feature importance plot

According to the bar chart, Destination Port is the most influential feature. It is also the most important feature for classifying Endpoint Denial of Service, Remote Access Software and Brute Force. Even though Destination Port is the most influential feature for the model, it is insignificant for

Network Service Scanning. The second most important feature is Source Port, which is also the second most important feature that influences predicting Remote Access Software. We presume that Destination Port is influential to Remote Access Software due to the fact that this hacking technique is often implemented by a Botnet tool called ARES, which by default is executed by using port 8080. To support this presumption, we have to look at SHAP dependence plot, a scatter plot that illustrates the effect a single feature has on the predictions. The dependence plot (Figure 3) shows the interaction between the SHAP value for Destination Port, Destination Port, and Source Port. The cases with 8080 Destination Port have high SHAP values, validating our presumption. The bar chart also shows that Network Service Scanning can be detected by Backward Packet Per Second and Packet Length Mean and Brute Force can be effectively identified by Minimal Forward Segment Size.

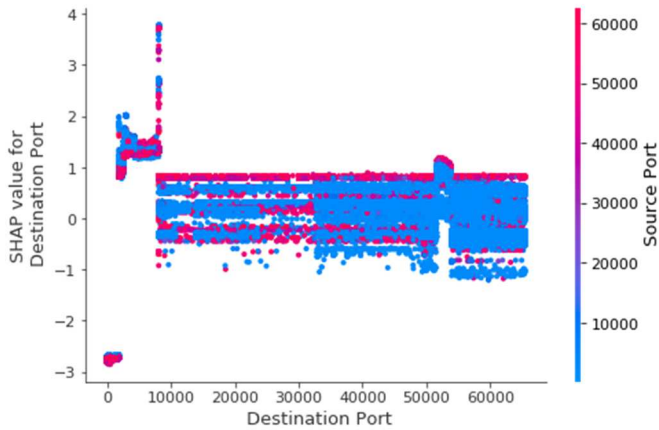


Figure 3. SHAP dependence plot for Destination Port

To further understand how each individual feature affects the classification, that is, the relation between the magnitude of a certain feature and the impact on the prediction, we use SHAP beeswarm plot as shown in Figure 4. Note that we only present the top 10 important features of the four hacking techniques.

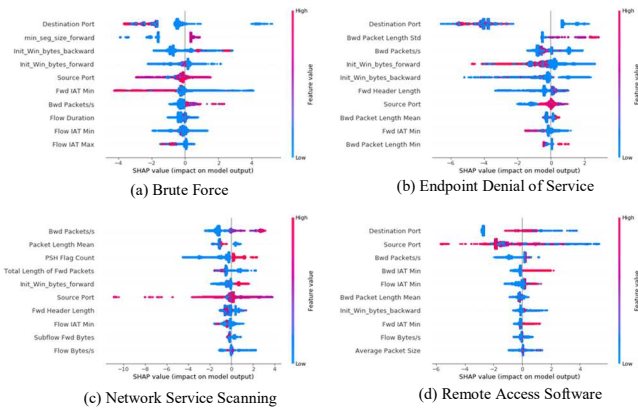


Figure 4. SHAP beeswarm plot for four hacking techniques

Figure 4a and Figure 4b show that an attack with small Fwd IAT Min value has a higher chance of being classified as Brute Force or Endpoint Denial of Service. IAT is the average

interval time between two packets. As Brute Force and Endpoint Denial of Service are both types of attacks that attempt to connect to the target many times in a short period of time, they usually have a small Fwd IAT value.

By means of port scans and vulnerability scans, a hacker may attempt to get a listing of services running on the targeted hosts. CIC-IDS2017 records Network Service Scanning behaviours which adversaries utilized Nmap to implement Port Scan on the remote hosts. One of the port scan techniques supported by Nmap is Xmas scan, which sets the FIN, PSN, and URG flags. This may be responsible for the high PSN Flag Count in Network Service Scanning class.

Contrary to Brute Force and Endpoint Denial of Service, the connection frequency of Remote Access Software is lower, that is, the IAT value should be smaller. As a result, the larger the value of Bwd IAT Min and Fwd IAT Min, the higher chance it will be classified as Remote Access Software.

The beeswarm plot combines feature importance with feature effects, giving us preliminary understanding of the relationship between the value of a feature and the impact on the prediction. To display explanations for individual predictions, we can use SHAP force plot.

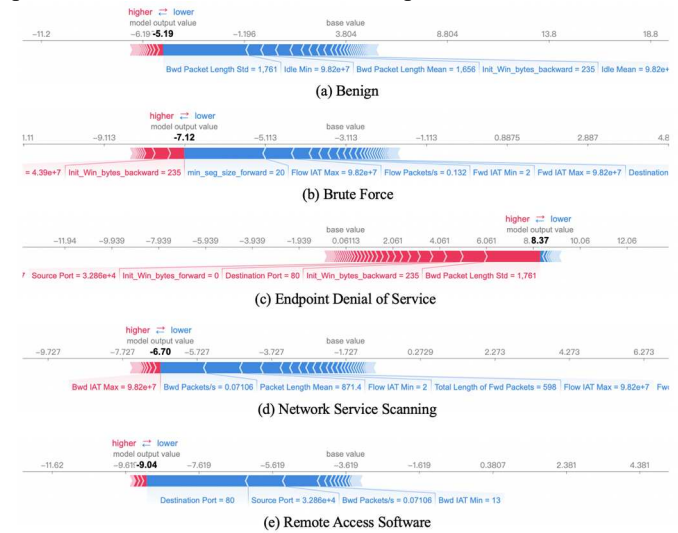


Figure 5. SHAP force plot

Next, we want to see how features contribute to the prediction for a specific observation. For multi-class classification, SHAP will return a separate matrix for each class for an input. It means that we expect five SHAP force plots for one prediction. Figure 5 demonstrates the SHAP force plots for an Endpoint Denial of Service case from CIC-IDS2017 dataset. Each force plot shows the model output value for each class. Figure 5c presents a positive output value in Endpoint Denial of Service class, while the output values in Figure 5a, 5b, 5d and 5e are negative; this indicates that the case is classified as Endpoint Denial of Service. The three most important features that increase the model output value for Endpoint Denial of Service are Bwd Packet Length Std (1761), Init\_Win\_bytes\_backward (235) and Destination Port (80). We notice that the average value of Bwd Packet Length Std of Benign cases is 123, while more than half of the Destination

Port of Remote Access Software cases are 8080. As a result, Bwd Packet Length Std (1761) results in a negative contribution for Benign; and Destination Port (80) highly decreases the output value for Remote Access Software.

2) **CSTI0-10:** We have mentioned that the CSTI-10-B model is expected to achieve better performance than the CSTI-10-A model due to the extra features it uses. We sum the mean absolute Shapley values per feature across the data of each malware, and then sort the features by decreasing importance. We found that among the top 20 important features of these ten malwares, on average at least 7 are additional features that we added, which means that the additional features have a considerable contribution in identifying malware.

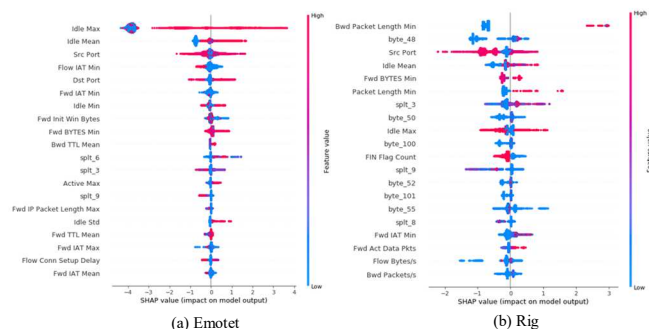


Figure 6. Emotet and Rig SHAP beeswarm plot

As shown in Figure 6, we use Emotet and Rig data, which have better improvement in performance, to plot SHAP beeswarm plot. Time-related features such as Interarrival Time and Flow Time have a stronger impact on Emotet. For example, if the value of idle max or idle std is higher, the case is easier to be classified as Emotet. Cases with a low value of Flow IAT Min also tend to be labelled as Emotet. Besides, features related to Flow Descriptors and Flags are more influential to Rig class. For example, cases with a large Fwd Act Data Pkts, large Fwd Act Data Pkts, or low FIN Flag Count have a higher chance to be classified as Rig.

The SHAP tool provides an approach to understand the impact of the network flow features on the malware classification. However, much uncertainty still exists about the causal relationship between the features and the malwares. For example, although the idle and IAT related features are influential to Emotet class, and Flow Descriptors and Flags are important to classify Rig class, a theoretical explanation for their relation is still lacking. This indicates a need for further research on how the important features impact on each malware.

#### IV. CONCLUSIONS

In this paper, we have shown an explainable flow-data classification model for hacker's attacks and malwares. The flow data used for training the model were converted from packets by CICFlowMeter. This process greatly shrank the size of data which reduced the requirement for data storage. We then introduced CSTITool to feature extraction processes with an aim of improving the model performance. We compared the XGBoost models that were trained with features obtained by

CICFlowMeter and CSTITool. The result showed that the additional descriptive flow statistics created by CSTITool can help improve the performance metrics. In order to understand how the model made predictions, we used SHAP to further investigate the relation between cyberattacks and network flow features. The experiences showed that our model achieved good accuracy in both hacker's attack and malware classification. It provided interpretability in hacker's attacks. However, there is still no sufficient theoretical explanation for the causal relationship between the features and the malwares, and further research is needed.

#### ACKNOWLEDGMENT

This research is supported by industry and institute in Taiwan, and the authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the contributions of this paper.

#### REFERENCES

- [1] (2021) The MITRE ATT&CK website. [Online]. Available: <https://attack.mitre.org/>
- [2] (2021) The snort website. [Online]. Available: <https://www.snort.org/>
- [3] (2021) The Suricata website. [Online]. Available: <https://suricata.io/>
- [4] A. Fast, "Deep Packet Dynamics for Encrypted Traffic." [Online]. Available: [https://static1.squarespace.com/static/5a70743a90badee73423db1f/t/5f5c54a9ef8cdeb769c691eb85/1606765216821/CFAT\\_TB\\_DPD\\_1.0.pdf](https://static1.squarespace.com/static/5a70743a90badee73423db1f/t/5f5c54a9ef8cdeb769c691eb85/1606765216821/CFAT_TB_DPD_1.0.pdf)
- [5] A. Habibi Lashkari, G. Draper Gil, M. S. Mamun, and A. A. Ghorbani, "Characterization of Tor traffic using time based features," Proceedings of the 3rd International Conference on Information Systems Security and Privacy, 2017.
- [6] B. Anderson and D. McGrew, "Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity," Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1723–1732, Aug. 2017.
- [7] J. S. Rojas, Á. R. Gallón, and J. C. Corrales, "Personalized service degradation policies on OTT applications based on the consumption behavior of users," Computational Science and Its Applications – ICCSA 2018, pp. 543–557, 2018.
- [8] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, "DISCLOSURE: Detecting botnet command and control servers through large-scale NetFlow analysis," Proceedings of the 28th Annual Computer Security Applications Conference on - ACSAC '12, 2012.
- [9] P. Amini, R. Azmi, and M. A. Araghizadeh, "Botnet detection using NetFlow and clustering," Advances in Computer Science: an International Journal, vol. 3, no. 2, Mar. 2014.
- [10] P. Benko, G. Malicsko, and A. Veres, "A large-scale, passive analysis of end-to-end tcp performance over gprs," IEEE INFOCOM 2004, 2004.
- [11] P. Prasse, L. Machlica, T. Pevný, J. Havelka, and T. Scheffer, "Malware detection by analysing encrypted network traffic with neural networks," Machine Learning and Knowledge Discovery in Databases, pp. 73–88, 2017.
- [12] R. Hofstede, P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow monitoring explained: From packet capture to data analysis with netflow and ipfix," IEEE Communications Surveys & Tutorials, vol. 16, no. 4, pp. 2037–2064, 2014.
- [13] S. Andropov, A. Guirik, M. Budko, and M. Budko, "Network anomaly detection using artificial neural networks," 2017 20th Conference of Open Innovations Association (FRUCT), Apr. 2017.
- [14] S. M. Lundberg, B. Nair, M. S. Vavilala, M. Horibe, M. J. Eisses, T. Adams, D. E. Liston, D. K.-W. Low, S.-F. Newman, J. Kim, and S.-I. Lee, "Explainable machine-learning predictions for the prevention of hypoxaemia during surgery," Nature Biomedical Engineering, vol. 2, no. 10, pp. 749–760, 2018.
- [15] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee, "From local

- explanations to global understanding with explainable ai for trees,” Nature Machine Intelligence, vol. 2, no. 1, pp. 56–67, 2020.
- [16] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions Share on,” Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 4768–4777, Dec. 2017.
- [17] S. Rezaei and X. Liu, “Deep learning for encrypted traffic classification: An overview,” IEEE Communications Magazine, vol. 57, no. 5, pp. 76–81, 2019.
- [18] T. Chen and C. Guestrin, “XGBoost: a scalable tree boosting system,” Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016.
- [19] X. Liu, Z. Tang, and B. Yang, “Predicting network attacks with CNN by constructing images from Netflow data,” 2019 IEEE 5th Intl Conference on BigDataSecurity, HPSC and IDS, 2019.
- [20] Y. Abuadlla, G. Kvascev, S. Gajin, and Z. Jovanovic, “Flow-based anomaly intrusion detection system using two neural network stages,” Computer Science and Information Systems, vol. 11, no. 2, pp. 601–622, 2014.



Boryau Hsupeng received his M.S. degree in the Department of Space Science and Engineering from National Central University, Taoyuan, Taiwan, in 2015. He is currently an engineer in Cybersecurity Technology Institute, Institute for Information Industry, Taiwan. His current and previous research interests include magnetospheric physics, space plasma physics, numerical simulation, and machine learning.



Kun-Wei Lee received his M.S. degree in statistics from National Chengchi University, Taipei, Taiwan, in 2017. He is currently an engineer in Cybersecurity Technology Institute, Institute for Information Industry, Taiwan. His research interests include data mining, machine learning, deep learning, and generative adversarial network.



Te-En Wei received his Ph.D. degree in the Department of Computer Science and Information Engineering at National Taiwan University of Science and Technology, Taipei, Taiwan. He is currently an acting director in Cybersecurity Technology Institute, Institute for Information Industry, Taiwan. His research includes Network Security, Application Security, and Financial Forecasting.



Shih-Hao Wang received his post-graduate degree of Management Information Systems from National Pingtung University of Science and Technology. He is a section manager at Institute for Information Industry. He has conducted research and design in cyber security, network communication, and fleet management.