

Artificial Intelligence Hybrid Learning Architecture for Malware Families Classification

Yan-Ju Chen ¹, Wen-Han Kuo ¹, Sung-Yun Tsai ¹, Jiann-Liang Chen ¹, Yu-Hung Chen ¹,
Wei-Zhao Xu ²

¹ Department of Electrical Engineering, National Taiwan University of Science and Technology, Taiwan

² Cybersecurity Technology Institute for Information Industry, Taiwan

m02670261@gmail.com, skaj0626@gmail.com, su11221122su@gmail.com, lchen1215@gmail.com,
aleex830625@gmail.com, wzshi@iii.org.tw

Abstract—In recent years, the rise of the Internet of Things has led to a gradual expansion of internet services, but most people ignore the importance of information security. This study investigates the characteristics of the malicious traffic that is generated during the operation of malware, and classifies malware into families without using SSL/TLS decryption. In this work, the features of traffic include the total numbers of packets and bits, sending time, packet size, delivery intervals, and others. All of features that are obtained by extracted of traffic flows are integrated into a complex set and a model that can identify the type of malware is trained by machine learning and deep learning. This work solves the problem of imbalanced data in traffic flows using a traffic analysis mechanism and developing a multi-layer network analysis structure that improves the stability and reliability of the proposed training model, to ensure cyber security.

Keywords— Malware, Malware Traffic, Artificial Intelligence, Machine Learning, Deep Learning, Data Imbalance

I. INTRODUCTION

The rise of the Internet of Things (IoT) has expanded the range of applications of the Internet. The concept is to use Internet technology and cloud systems to enable machines to communicate with each other and to integrate and analyze data. This has changed the traditional way of information transmission. According to Gartner, 13.5 billion IoT devices will be in use by 2020. The most important part of the IoT industry is the vast user data.

IoT has changed the way information is applied, supporting breakthrough developments of every industry. However, information security is largely neglected. The main problem is that without relevant laws and regulations, Internet hackers update attack modes to outmatch current protection modes. Almost all network attacks hide or use fake network IP addresses, so identifying the real source of an attack is difficult. In the world of IoT, all devices and systems must be connected to a network, so all IoT-related devices and systems may be in danger [2]. Common cyberattacks include Phishing, the Distributed Denial of Service Attack (DDoS), the use of a malicious link and the use of malware.

In the field of information security, the threat of malware is becoming more and more serious, and victims are spread across many industries, such as finance, education, and medicine [3]. The main reason for difficult prevention is that malicious anti-detection software protects the traffic of the attacker and the

C&C (Command and Control) server through encryption. According to Cisco's statistics, more than 60% of current network traffic is SSL/TLS-encrypted [4], and malicious traffic that is generated by malicious software accounts for more than 10%. The amount of SSL/TLS-encrypted traffic has grown from number of 500 in 2014 to number of 29,000 in 2016, a 58-fold increase; moreover, the number of C&C servers involved in encrypted communication has increased by a factor of 200 [5].

According to HACKMAGEDDON information security timelines and statistics [6], 2018 has seen many more cyberattacks than 2017, as shown in Figure 1. Attacks in September 2018 can be divided into Malware/PoS Malware, Unknown, Targeted Attack, Vulnerability, Account Hijacking and DDoS, as shown in Figure 2. The statistics for attacks in September 2018 reveal that the targets of attacks include a large number of individual users, as shown in Figure 3. At present, the proportion of malicious software that uses encryption technology is increasing, but the most widely used security detection technology is still based on pattern matching. For example, the Snort organization supply 3307 rules of an Intrusion Prevention System (IPS) to detect traffic flows [4][7], but only six of them are used to detect malicious TLS, and all are limited to the detection in case of self-signed signature encryption. It lacks the ability to analyze encrypted traffic and cannot compare normal and malware after the traffic is encrypted. Encryption technology not only protects normal traffic, but also protects malicious behavior.

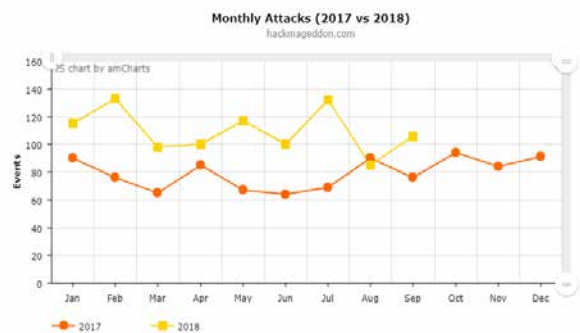


Figure 1. Monthly Attacks (2017 vs 2018)

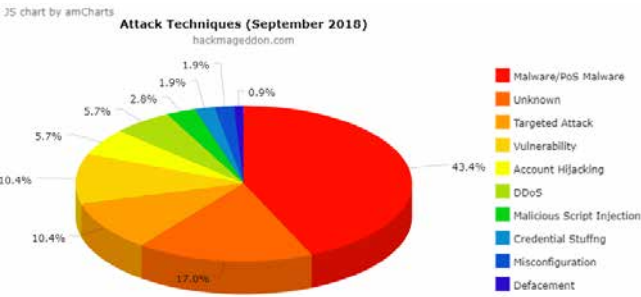


Figure 2. Attack Techniques (September 2018)

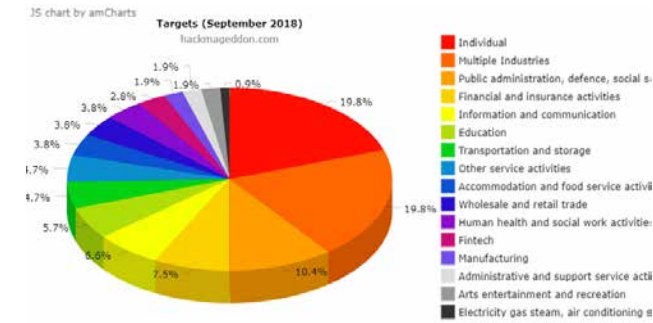


Figure 3. Targets (September 2018)

The most direct means of cracking encrypted traffic is decryption. To analyze transmitted content, the MITM (man-in-the-middle) configuration can be used as a TLS proxy [5], and a TLS connection is established with the user (client) and the remote server. However, this method requires the encryption credentials of all applications on the client side. Currently popular software uses built-in credentials, if credentials cannot be obtained, then MITM decryption cannot be conducted, and in order to maintain all encrypted connections, expensive, high-performance devices must be used. These devices limit supportable services, causing problems in service operations, such as the delivery of high-complexity JavaScript and CSS structured web pages, and self-developed special application services. Therefore, this study focuses on the relationship between malicious behavior and traffic without decryption, and classifies malware into families using machine learning and deep learning techniques and artificial intelligence (AI).

II. FEATURE EXTRACTION

The dataset that is used in this study is the CTU-13-Dataset [8] from Stratosphere Lab. The Stratosphere Lab collected pcap file of CTU-13-Dataset, and VirusTotal is used to query the hash code of datasets to generate the behavior of malicious software. To facilitate a preliminary analysis, the data are consolidated as shown in Figure 4.

Figure 4. Malware Behavior Integration Table

A. Feature Information

This study uses the open source tool NetMate-flowcalc to collect various kinds of data from the information flow that is provided by the CTU-13 dataset. The open source tool facilitates the study of data statistics and analysis. The statistics are used in training each category of malware. The data statistics are saved as a comma-separated csv file.

In this study, information of pcap file capture through NetMate is used. A pcap file includes a large amount of information. Forty-four feature data are generated. Among all of the features, only scrip (source IP address) and dstip (destination IP address) are in string format, and the other 42 features are numerical values.

The first five features (scrip, scrport, dstip, dstport, proto) are changeable information, so they are deleted in advance of feature preprocess. The remaining 39 features are retained.

1) Packet-related Features:

NetMate-flowcalc are used to calculate the values associated with features that are related to the number of packets. The numbers of packets that are sent in the forward and backward directions are counted, and the mean number of packets of sub flow that are sent in the forward and backward directions is determined, as shown in Table 1.

TABLE 1. PACKET-ORIENTED FEATURES

total_fpackets	Total packets in the forward direction
total_bpackets	Total packets in the backward direction
sflow_fpackets	The average number of packets in a sub flow in the forward direction
sflow_bpackets	The average number of packets in a sub flow in the backward direction

2) Volume-related Features:

With respect to volume-related features in the forward and backward directions, the minimum, maximum, and average standard deviations of packet size statistics are obtained. The numbers of sub-flow and the header are obtained and represented in bytes, as shown in Table 2.

TABLE 2. VOLUME-ORIENTED FEATURES

total_fvolume	Total bytes in the forward direction
total_bvolume	Total bytes in the backward direction
min_fpctl	The size of the smallest packet sent in the forward direction (in bytes)
mean_fpctl	The mean size of packets sent in the forward direction (in bytes)
max_fpctl	The size of the largest packet sent in the forward direction (in bytes)
std_fpctl	The standard deviation from the mean of the packets sent in the forward direction (in bytes)
min_bpctl	The size of the smallest packet sent in the backward direction (in bytes)
mean_bpctl	The mean size of packets sent in the backward direction (in bytes)
max_bpctl	The size of the largest packet sent in the backward direction (in bytes)
std_bpctl	The standard deviation from the mean of the packets sent in the backward direction (in bytes)
sflow_fbytes	The average number of bytes in a sub flow in the forward direction
sflow_bbytes	The average number of packets in a sub flow in the backward direction
total_fhlen	The total bytes used for headers in the forward direction
total_bhlen	The total bytes used for headers in the backward direction

3) Time Division-related Features:

The minimum, maximum, mean, and standard deviation of the time between transmission and traffic flow in activity or in idle state in the forward and backward directions, are obtained in microseconds, as shown in Table 3.

TABLE 3. TIME DIVISION ORIENTATION FEATURES

min_fiat	The minimum amount of time between two packets sent in the forward direction (in microseconds)
mean_fiat	The mean amount of time between two packets sent in the forward direction (in microseconds)
max_fiat	The maximum amount of time between two packets sent in the forward direction (in microseconds)
std_fiat	The standard deviation from the mean amount of time between two packets sent in the forward direction (in microseconds)

min_biat	The minimum amount of time between two packets sent in the backward direction (in microseconds)
mean_biat	The mean amount of time between two packets sent in the backward direction (in microseconds)
max_biat	The maximum amount of time between two packets sent in the backward direction (in microseconds)
std_biat	The standard deviation from the mean amount of time between two packets sent in the backward direction (in microseconds)
duration	The duration of the flow (in microseconds)
min_active	The minimum amount of time that the flow was active before going idle (in microseconds)
mean_active	The mean amount of time that the flow was active before going idle (in microseconds)
max_active	The maximum amount of time that the flow was active before going idle (in microseconds)
std_active	The standard deviation from the mean amount of time that the flow was active before going idle (in microseconds)
min_idle	The minimum time a flow was idle before becoming active (in microseconds)
mean_idle	The mean time a flow was idle before becoming active (in microseconds)
max_idle	The maximum time a flow was idle before becoming active (in microseconds)
std_idle	The standard deviation from the mean time a flow was idle before becoming active (in microseconds)

4) Flag-related Features:

The Push Function Flag, abbreviated as PSH Flag, is transmitted in a byte stream on TCP. Before transmission, the byte stream is stored in the buffer until the buffer is saturated, and then the entire byte stream is sent out. When PSH Flag is set to 1, it can be directly transmitted without waiting for the buffer to saturate. The Urgent Flag is abbreviated as URG Flag, and is represented as an emergency packet, as shown in Table 4.

TABLE 4. FLAG-ORIENTED FEATURES

fpsh_cnt	The number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP)
bpsh_cnt	The number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)
furg_cnt	The number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)
burg_cnt	The number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)

B. Feature Analysis

To evaluate the importance of the feature information in each dimension, a feature evaluation mechanism is added to reduce the complexity of the model and to prevent overfitting. The mechanism involves retaining relevant feature information. In this study, ANOVA and AutoEncoder are used to evaluate the importance of features. The following two evaluation methods are used.

1) ANOVA:

ANOVA is a method for evaluating the importance of features [9]. To find the between groups mean square (MS_B) and the within each group mean square (MS_W), the total sum of squares (SS_T) between features, the between sum of squares (SS_B), and the within sum of squares (SS_W) are calculated, as shown in Figure 5 and Eq. (1). The MS_B is divided by the MS_W to yield the F value, which quantifies the correlation between the features.

$$SS_T = \sum_{\text{All Samples}} (X_i - \bar{X}_{\text{total average}})^2 \quad (1)$$

SS_B represents the distance between each group average and the total average. A larger distance corresponds to a larger difference between each group, as shown in Eq. (2).

$$SS_B = \sum_{\text{each group}} n_{\text{Number of samples in each group}} * (\bar{X}_{\text{Group average}} - \bar{X}_{\text{total average}})^2 \quad (2)$$

SS_W represents the distribution of each group of samples. A larger value indicates more samples discrete, as shown in Eq. (3).

$$SS_W = \sum_{\text{All groups}} \sum_{\text{All samples in the group}} (X_{gi}(\text{Sample of the group}) - \bar{X}_{\text{Average of the group}})^2 \quad (3)$$

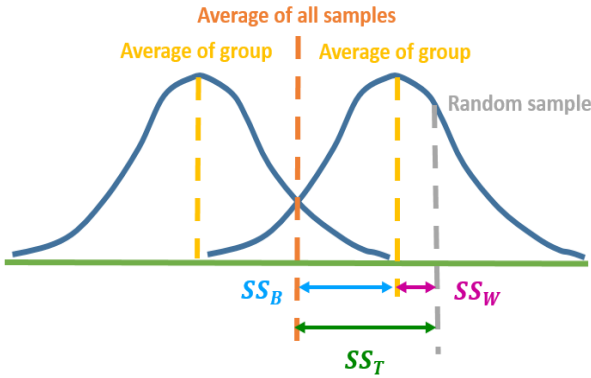


Figure 5. ANOVA Schematic Diagram

ANOVA uses the F value to determine whether there a significant difference exists between the groups. A larger F value corresponds to greater mutual independence of the features. Finally, the number of the degrees of freedom (where df_W is the number of independent pieces of information) and the mean square (MS) are used to calculate the F value, as in Eqs. (4), (5), (6), (7) and (8).

$$MS_W = \frac{SS_W}{df_W} \quad (4)$$

$$MS_B = \frac{SS_B}{df_B} \quad (5)$$

$$df_W = \text{Number of groups} - 1 = G - 1 \quad (6)$$

$$df_B = \text{Number of samples} - \text{Number of groups} = N - G \quad (7)$$

$$F = \frac{MS_B}{MS_W} \quad (8)$$

2) AutoEncoder:

Within a neural network, more neurons enable more information to be stored, but at the cost of greater complexity and training time. The main concept of the AutoEncoder [10] is to compress and restore data using an encoder and a decoder. The best weights are found in this process, so that feature information can be completely restored, reducing the noise information without loss of accuracy. The architecture diagram as shown in Figure 6.

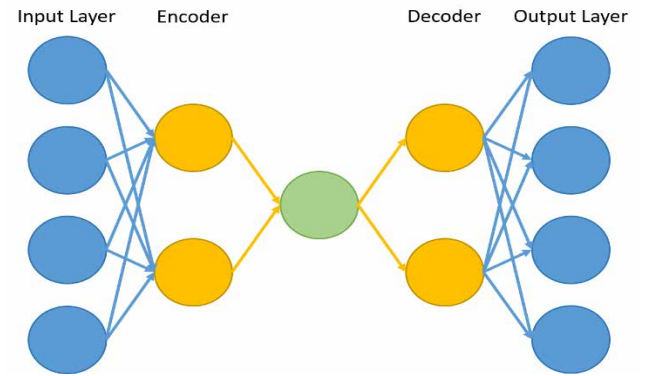


Figure 6. AutoEncoder Schematic Diagram

AutoEncoder can reduce the dimensions of the information from original data features. During the encoder and the decoder process, AutoEncoder would calculate the difference between the outputs and the original data. The difference is the loss function (Loss), as shown in Eq. (9), where $\hat{x}_i$ is the output value; x_i is the input value; and L is the loss function.

$$L(f(X)) = \frac{1}{2} \sum_{i=1}^N (\hat{x}_i - x_i)^2 \quad (9)$$

III. EXPERIMENTAL METHOD

A. Preprocess

1) Feature Filter:

To remove redundant noise from the Packet-related, Volume-related, Time Division-related and Flag-related feature, ANOVA and AutoEncoder are used to reduce the complexity of the model and speed up the speed up the overall model training time. In this study, 39 original features are used. After analysis of the results from the ANOVA and AutoEncoder, the number of features was reduced to 37. Figure 7 show the results of ANOVA and Table 5 show the table of the feature integration.

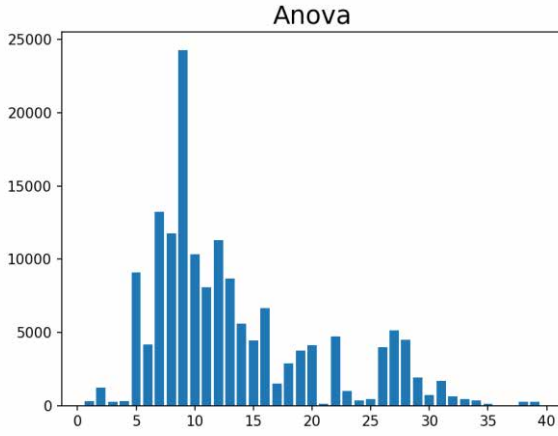


Figure 7. ANOVA

TABLE 5. FEATURE INTEGRATION

Features			
<i>total_fpackets</i>	<i>std_bpktl</i>	<i>mean_active</i>	<i>fpsh_cnt</i>
<i>total_fvolume</i>	<i>min_fiat</i>	<i>max_active</i>	<i>bpsh_cnt</i>
<i>total_bpackets</i>	<i>mean_fiat</i>	<i>std_active</i>	<i>total_fhlen</i>
<i>total_bvolume</i>	<i>max_fiat</i>	<i>min_idle</i>	<i>total_bhlen</i>
<i>min_fpktl</i>	<i>std_fiat</i>	<i>mean_idle</i>	
<i>mean_fpktl</i>	<i>min_biat</i>	<i>max_idle</i>	
<i>max_fpktl</i>	<i>mean_biat</i>	<i>std_idle</i>	
<i>std_fpktl</i>	<i>max_biat</i>	<i>sflow_fpackets</i>	
<i>min_bpktl</i>	<i>std_biat</i>	<i>sflow_fbytes</i>	
<i>mean_bpktl</i>	<i>duration</i>	<i>sflow_bpackets</i>	
<i>mean_bpktl</i>	<i>min_active</i>	<i>sflow_bbytes</i>	

2) Data Imbalance:

To balance the sample number of each malware family from dataset, the synthetic minority over-sampling technique (SMOTE) is used [11]. The theory of SMOTE is focus on the fewer categories in the dataset. In these fewer categories, SMOTE find the data points of the same categories in the vicinity. Any two data points are connected by a line, and a new data point is random generated on the line, as shown in Figure 8 and Eq. (10).

$$x_{new} = x_i + (x' - x_i) \times \delta \quad (10)$$

x_i is a random data point of the fewer dataset; x' is the nearest data point to x_i , and δ is a value between 0 and 1.

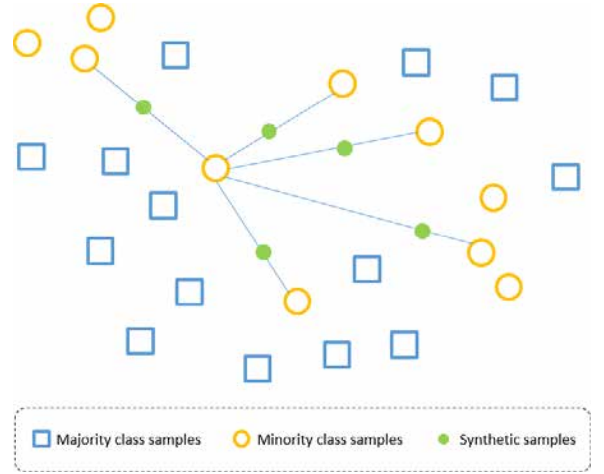


Figure 8. SMOTE Schematic Diagram

B. Machine Learning

1) eXtreme Gradient Boosting:

eXtreme Gradient Boosting (XGBoost) is based on the concept of the Gradient Boosting Decision Tree (GBDT), which is an iterative decision tree algorithm. Unlike the Random Forest algorithm, GBDT weights all results to produce a final output, whereas Random Forest uses a majority vote to output the final result.

First, XGBoost generates t-tree (t is the number of tree), when constructing to the t-th tree, the difference between the predicted value and the actual value which obtained from the training result of the former t-1 tree needs to be fitted [12]. Whenever a new tree is generated by fitting, view all of the generated trees, and selected the tree with the smallest objective function (cost), which represents the smallest error value, as shown in Eq. (11):

$$L^{(t)} = \sum_{i=1}^n \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \quad (11)$$

Where $g_i = \partial_{y^{(t-1)}} l(y_i, y^{(t-1)})$ and $h_i = \partial_{y^{(t-1)}}^2 l(y_i, y^{(t-1)})$.

The above objective function is used to evaluate the fitness of a tree, and its structure is decomposed in Eq. (11). One layer of the tree is optimized at a time. To find the best segmentation point, the root node must be divided into two leaf nodes, and select the feature with the highest Gain, according to Eq. (12):

$$Gain = \frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma \quad (12)$$

IV. EXPERIMENTAL RESULTS & ANALYSIS

Whether the feature analysis architecture affects the results of model training is interest. In the experiment, the main XGBoost algorithm is used to evaluate the accuracy of 20 malware family classification models. Owing to the imbalance of the original dataset, SMOTE is used to balance the data and determine whether dataset balancing improving the overall accuracy of the malware classification model.

A confusion matrix is used to analyze performance of prediction results on a categorization, it counts a number of true and wrong predictions to summarize with count values by each category. In this study, the number of dataset is about 900 thousand data in 20 malware families. Using the 3-fold cross validation to train the malware classification model with XGBoost, and reduce the dimensions by ANOVA and AutoEncoder.

A. Performance Analysis

In this study, the accuracy of XGBoost with ANOVA is 99.73%, as shown in Figure 9. And the accuracy of XGBoost with AutoEncoder is 99.13%, as shown in Figure 10.

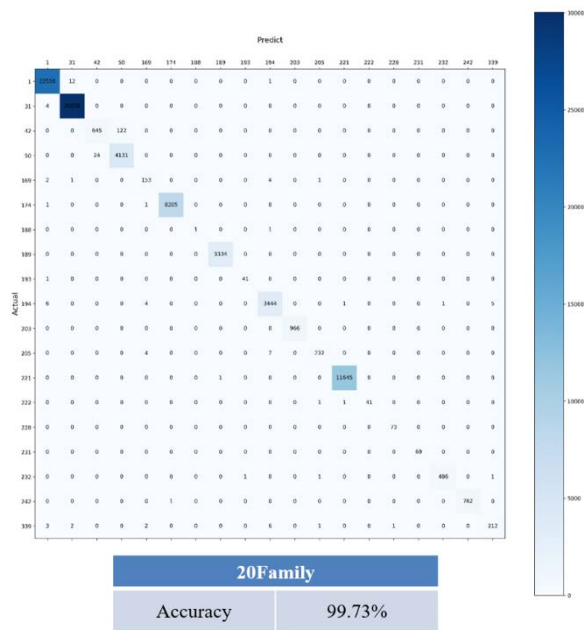


Figure 9. Accuracy of XGBoost with ANOVA



Figure 10. Accuracy of XGBoost with AutoEncoder

B. Data Imbalance Analysis

This study has been increased the reliability of a performance analysis of the classification by SMOTE. Figure 11 and Figure 12 demonstrate the performance analysis of confusion matrix which show the approximate quantities in the same categorization between prediction and actual results.

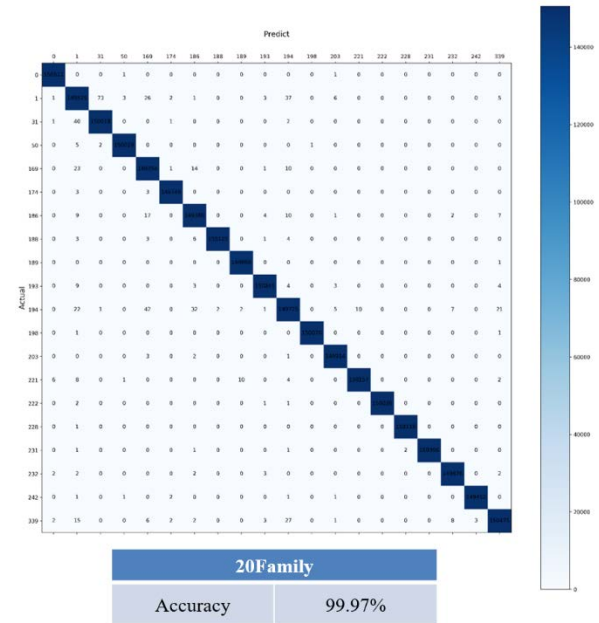
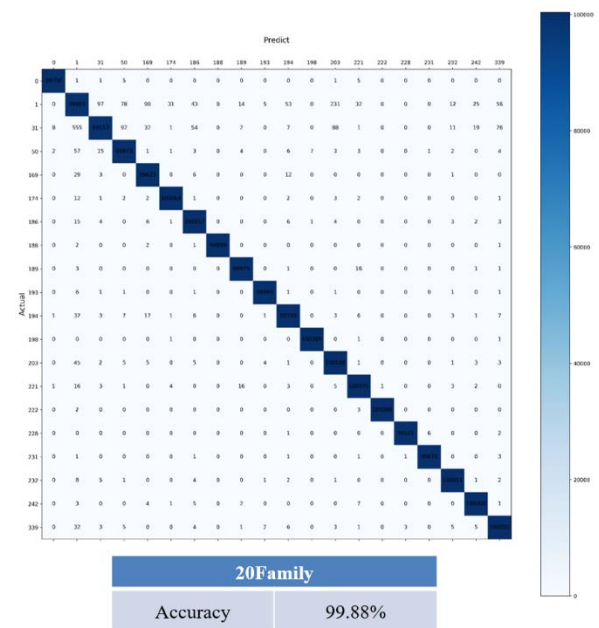


Figure 11. Accuracy of XGBoost with ANOVA and SMOTE



1) Feature Distribution Information

According to the results in this research, the datasets from same family of malware, there will be similar distribution of characteristic data, through the data visualization technology, it can identify the same family of malware information by the image of distribution. The results as show in Figures 13, 14, 15, and 16.

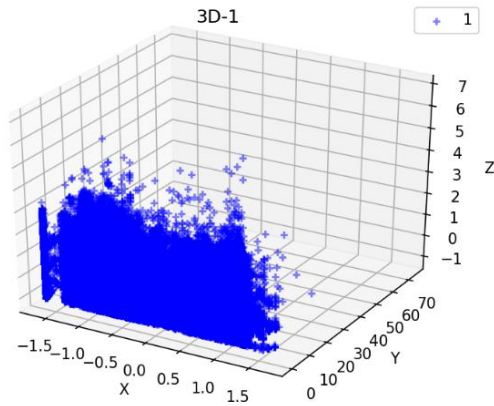


Figure 13. Data Distribution

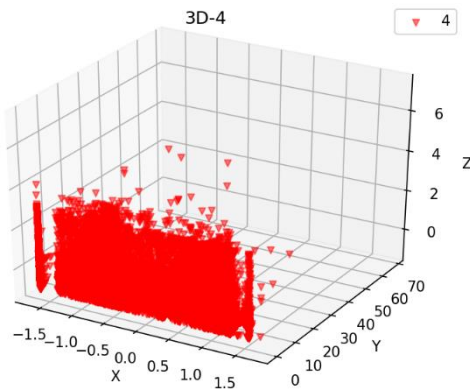


Figure 14. Data Distribution

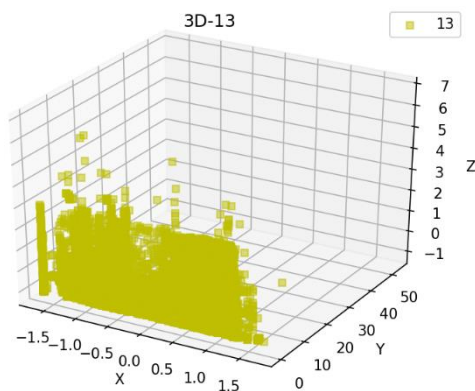


Figure 15. Data Distribution

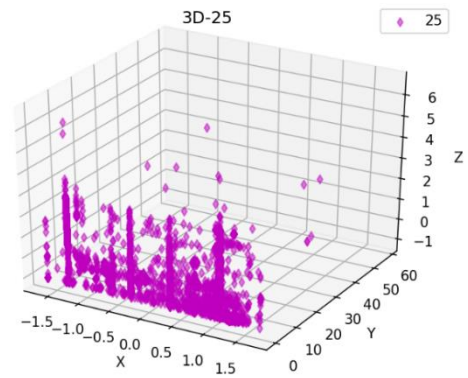


Figure 16. Data Distribution

V. CONCLUSION

This work has been classified malware into 20 families, and used imbalance learning with dimension reduction to increase the reliability of a performance analysis of the categorization. Currently, a network simulation environment is being developed, and attempts are being made to trigger unknown malware traffic flow to classify the malware with similar traffic distributions into 20 families or become a new family.

VI. REFERENCE

- [1] Gartner (2015, November 10). Gartner Says 6.4 Billion Connected "Things" Will Be in Use in 2016, Up 30 Percent from 2015 [Online forum comment]. Retrieved from <http://www.gartner.com/newsroom/id/3165317>.
- [2] McAfee, McAfee Labs. McAfee Labs 2017 Threats Predictions. Retrieved from <https://www.mcafee.com/tw/resources/reports/tp-threats-predictions-2017.pdf>
- [3] SECURELIST, KASPERSKY Lab. KASPERSKY SECURITY BULLETIN: OVERALL STATISTICS FOR 2016. Retrieved from <https://kasperskycontenthub.com/>
- [4] Anderson, B., Paul, S., & McGrew, D. (2016). Deciphering Malware's use of TLS (without Decryption). ArXiv.
- [5] Symantec, "The 5 Things You Need to Know About Managing Encrypted Traffic", 2017.
- [6] HACKMAGEDDON, September 2018 Cyber Attacks Statistics. Retrieved from <https://www.hackmageddon.com/2018/10/11/september-2018-cyber-attacks-statistics/>
- [7] Snort (2016). Snort: Community Rules [Online forum comment]. Retrieved from <https://www.snort.org/downloads>.
- [8] New dataset, CTU-13-Extended, now includes pcap files of normal traffic (<https://www.stratosphereips.org/blog/2015/7/17/new-dataset-ctu-13-extended-now-includes-pcap-files-of-normal-traffic>).
- [9] Z. H. Bohari, R. Ghazali, N. N. Atira, M. F. Sulaima, A. A. Rahman and M. K. Nor, "Building energy management saving by considering lighting system optimization via ANOVA method," Proceedings of the IEEE ICCTA, pp.216-220, 2018.
- [10] Q. Xu, Z. Wu, Y. Yang and L. Zhang, "The Difference Learning of Hidden Layer Between Autoencoder and Variational Autoencoder," Proceedings of the 29th Chinese Control and Decision Conference (CCDC), pp.4801-4804, 2017.

- [11] M. Ahsan, R. Gomes and A. Denton, "SMOTE Implementation on Phishing Data to Enhance Cybersecurity," Proceedings of the IEEE International Conference on Electro/Information Technology (EIT), pp.531-536, 2018.
- [12] Z. Chen, F. Jiang and Y. Cheng, "XGBoost Classifier for DDoS Attack Detection and Analysis in SDN-Based Cloud," Proceedings of the IEEE BigComp, pp.251-256, 2018.

Yan-Ju Chen

Yan-Ju Chen received the M.S. degree in Electrical Engineering of National Taiwan University of Science and Technology, Taipei, Taiwan. His current researches are directed at malware analysis and artificial intelligence for cyber security.



Wen-Han Kuo

Wen-Han Kuo received the M.S. degree in Electrical Engineering of National Taiwan University of Science and Technology, Taipei, Taiwan. His current researches are directed at traffic analysis, and artificial intelligence for cyber security.



Sung-Yun Tsai

Sung-Yun Tsai received the M.S. degree in Electrical Engineering of National Taiwan University of Science and Technology, Taipei, Taiwan. Her current researches are directed at malicious traffic detection, and artificial intelligence for cyber security.



Jiann-Liang Chen

Jiann-Liang Chen was born in Taiwan on December 15, 1963. He received the Ph.D. degree in Electrical Engineering from National Taiwan University, Taipei, Taiwan in 1989. Since August 2008, he has been with the Department of Electrical Engineering of National Taiwan University of Science and Technology, where he is a distinguished professor now. His current research interests are directed at cellular mobility management, personal communication systems, artificial intelligence, and cyber security.



Yu-Hung Chen

Yu-Hung Chen received the M.S. degree in Electrical Engineering of National Taiwan University of Science and Technology, Taipei, Taiwan. His current researches are directed at phishing detection, and artificial intelligence for cyber security.

Wei-Zhao Xu

Wei-Zhao Xu received the M.S. degree in Computer Science of National Chiao Tung University, Hsinchu, Taiwan. He is currently a Senior Engineer in Institute for Information Industry-Cybersecurity Technology Institute, Taipei, Taiwan. His current research include information security and malware analysis.