Neural Network with Principal Component Analysis for Malware Detection using Network Traffic Features

Ventje Jeremias Lewi Engel[1], Mychael Maoeretz Engel[2] and Evan Joshua[3]

[1]Department of Computer Engineering, Institut Teknologi Harapan Bangsa, Dipatiukur Street, Bandung, Indonesia

[2]Department of Informatics, Universitas Ciputra, Citraland CBD Boulevard, Surabaya, Indonesia

[3]Department of Informatics, Institut Teknologi Harapan Bangsa, Dipatiukur Street, Bandung, Indonesia

Keywords: Malware Detection, Neural Network, Network Traffic Features, Principal Component Analysis, Features Set

Abstract: Network traffic acts as a medium for sending information used by hackers to communicate with malware on the victim's device. Malware analyzed in this study will be divided into three classes, namely adware, general malware, and benign. Malware classification will use 79 features extracted from network traffic flow, and analysis of these features will use Neural Network and Principal Component Analysis (PCA). The total flow of network traffic used is 442,240 data. The evaluation of malware detection is based on Fmeasure rather than traditional accuracy metric. The literature features set (15 features) produces an Fmeasure of 0.6404, the researcher features set (12 features) produces an F-measure of 0.6660, and the PCA features (23 features) produces an F-measure of 0.7389. This result concludes that PCA can generate features that have better results for malware detection with Neural Network algorithm. Aside from the PCA result, it is shown that more features used does not mean that the accuracy of malware detection will also increase. The drawback of using PCA is the loss of interpretability. Further research is needed on the analysis of the combination of network traffic features besides using PCA.

1 INTRODUCTION

With the increasing adoption of information technology, the use of IoT (Internet of Things) devices and smartphones is growing. Security threats on IoT devices and smartphones also increased. Cyberattacks, such as taking access rights, data destruction, and the theft of important personal information, can be carried out on IoT devices and smartphones (Kaspersky, 2015). This cyber-attack is mostly entered through malicious software or malware that was successfully planted on IoT devices and smartphones.

Malware is an application that has a harmful purpose, such as corrupting data, stealing valuable information, disrupting device performance, and taking over the system (Kaspersky, 2015). This threat continues to increase every year, even in 2017 found around 3.5 million new malware only on Android smartphone devices (Lueg, 2017). One of the suspicious activities or suspicious activity of malware is the use of network traffic or network traffic. The use of network traffic can be a medium for sending confidential information in the form of PINs, bank account information, personal messages, and passwords to the perpetrators of the malware maker (Zhou and Jiang, 2012). Malware can also use network traffic as a backdoor for other malware to enter.

Network traffic on IoT devices and smartphones has the same basis as network traffic in general, which contains packets that have a header and data section (Forouzan, 2010). Data is obtained and processed at the application layer, while headers are added at each layer. Each data and header has a size that varies with the specified limits. The data included in the packet contains what you want to send from source to destination. The header consists of the destination IP address, sender's IP address, source port, destination port, and so on. Most network traffic features are time-series data. In general, malware detection classifies applications into adware, general malware, and benign (Lashkari et al., 2017). Adware force displays advertisements on top of the running software. Adware aims to increase revenue for software developers so that the advertised company will pay. Every kind of general malware can be sure to have bad goals, such as corrupting or stealing data. Benign is a regular type of application that does not have dangerous goals and runs according to what the application developer has

Engel, V., Engel, M. and Joshua, E.
Neural Network with Principal Component Analysis for Malware Detection using Network Traffic Features.
DOI: 10.5220/0009908902660271
In *Proceedings of the International Conferences on Information System and Technology (CONRIST 2019)*, pages 266-271
ISBN: 978-989-758-453-4

written in the documentation section.

Efforts to detect mobile malware have been carried out with various approaches. A behaviorbased method that uses permissions and system calls as features produced accuracy that is still relatively low, with an average of 60% (Kaushik and Jain, 2015). The result was 65.29% using Simple Logistic Regression, 65.29% using Naive Bayes, 70.31% using SMO, and 54.79% using Random Tree (Kaushik and Jain, 2015). Other research using the Neural Network (NN) method with network traffic features to detect malware on smartphones had successfully identified malware botnets with a precision level of around 88.3% (Stevanovic and Pedersen, 2015). This result is much higher compared to the naive Bayes and logistic regression methods, each of which has a value of 7% and 32% (Stevanovic and Pedersen, 2015). Besides, the neural network method successfully outperformed the Support Vector Machine (SVM) method in classifying network traffic (Zhang et al., 2012). Detecting malware through network traffic analysis which is time-series data is suitable for the neural network method.

The weakness of that research was the NN is performed on all network traffic features. Some network features have significantly more roles than other network traffic features. For example, the network destination port is more important than the length of the header contents. Second, using all network traffic features means increasing the internal errors carried in the data. Third, features with large values will automatically weigh higher; for example, the port number commonly used will be much smaller when compared to the amount of data flow across the network (Lashkari et al., 2017). On the other hand, there is a Principal Component Analysis (PCA) method for feature extraction. Research showed that in network traffic classification, PCA had faster speed, higher accuracy, and more stable than the Naive Bayes estimation method (Yan and Liu, 2014).

The differences between this study and previous research are the network traffic dataset, the combination of features, and the NN configuration iteration used. The dataset was from the Canadian Institute for Cybersecurity, University of New Brunswick (for Cybersecurity, 2017) combined with sample data collected in Harapan Bangsa computer laboratory. The set of features will be carried out based on PCA compared with features obtained from literature studies and features chosen by researchers. The iteration of the NN configuration is done by programming that pays attention to learning rate, epoch, and parameter evaluation. The purpose of this study is to investigate the combination of network traffic features that can produce high precision, recall, and F-measure.

2 METHOD

2.1 Research Framework

This part outlines the framework of thought, namely indicators, proposed methods, objectives, and measurements. The indicator explains the factors that affect the results of the objective. The number of packets of datasets analyzed is the first factor. Secondly, the number of features and features names will be used in the training and testing process. Third, neural network hyperparameters include the number of hidden neurons, the epochs, and learning rate. Then in the proposed method, there is a dataset source. Then enter the feature selection stage. The feature will then be selected by analyzing it first. Then, after obtaining features from the results of the previous analysis, training will be conducted using the neural network method and continued with the testing phase. The test results processed to produce the objectives in the form of precision, recall, and Fmeasure.

2.2 Flowchart

The steps of this research were arranged in the form of a flowchart that begins with preprocessing. Preprocessing is the normalization of the features by dividing it by the maximum value of each feature, minimizing features so as not to dominate other features. Then the learning stage used the neural network method with backpropagation algorithm, and the testing stage used feed-forward. In the initial phase, the weight will be random according to the previous provisions and stored in a weight file. Learning outcomes would give a new weight value used in the test phase. The test output was divided into three, namely benign type network traffic, adware, or general malware. The flowchart can be seen at Figure 1.

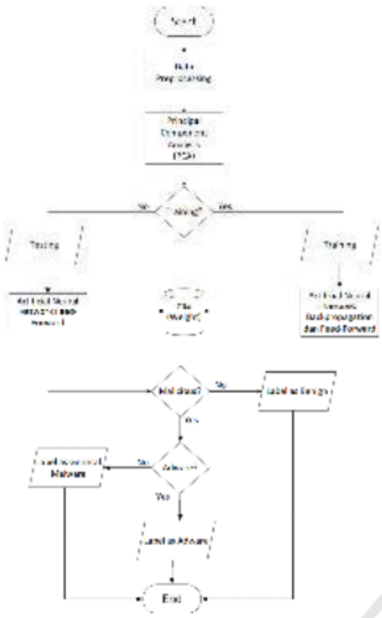

Figure 1: Research flowchart.

2.3 Neural Network Architecture

Neural network is one of machine learning techniques. Neural networks is a supervised learning with the resulting model in the form of weight (Rashid, 2016). There are three main layers in the neural network, namely the input layer, hidden layer and output layer.

In this research, three layers will be used, namely the input layer, hidden layer, and output layer. The input layer has several neurons according to the number of features used. In the hidden layer, only one layer will be used with the number of neurons tested, namely 4, 5, 6, and 12. The output layer will produce output in the form of 3 classes, namely benign, adware, and general malware. The test will apply several combinations of Neural Network parameters such as learning rate, hidden neurons, and the number of epochs. The learning rate tested is 0.1, 0.05, and 0.01, with the number of epoch 100, 200, and 300.

2.4 Dataset

The dataset is a pcap (packet capture) file that contains network traffic packets with a total of 79 features. The pcap file was taken from a total of 1900 android applications with a percentage of 20% malware and 80% benign. The dataset is divided into three groups, sourced from 250 adware applications, 150 general malware applications, and 1500 benign applications. In the training data, there are 2,312 network flow traffic from general malware, 149,871 for adware, and 201,609 for benign. In the testing data, there are 1,626 general malware flow, 24,271 adware flow, and 62,551 benign flow. The total flow of network traffic used is 442,240 data. The pcap file is converted to CSV file by using CICFlowMeter application, so one flow means one row of data.

2.5 Principal Component Analysis

Principal component analysis or PCA is useful for reducing high dimensions but does not eliminate the essence of the information. PCA transforms some related variables into several new variables that are not interconnected. Each variable will be checked for connectedness with other variables and will be sorted according to the most significant relationship. Mathematically, PCA looks for a linear transformation T that maximizes the equation 1

$$T^t Cov_{x-\bar{x}} T \tag{1}$$

$Cov_{x-\bar{x}}$ is the covariance matrix of data X with zero average. This linear mapping can be formed with the principal eigenvectors of the covariance matrix. Therefore, PCA solves eigenproblem in equation 2.

$$Cov_{x-\bar{x}} v = \lambda v(x) \tag{2}$$

2.6 Objectives and Evaluation

F-measure is used instead of accuracy for evaluating the model because of the class imbalance between data labeled as malware versus benign. One fifth of the data is labeled as malware and the rest of the data is labeled as benign. F-measure is used to help in drawing conclusions in which Neural Network parameters are the best. The advantage of using the F-measure for evaluation is it combines precision and recall into a single unit. Figure 2 shows the confusion matrix used to obtain the values of True Positive, False Positive, True Negative, and False Negative.

Predicted Value	**Actual Value**	
	TRUE	***FALSE***
TRUE	True Positive (TP)	False Positive (FP)
FALSE	False Negative (FN)	True Negative (TN)

Figure 2: Confusion matrix.

Equation (3), (4), and (5) are the equations to calculate precision, recall, and F-measure.

$$Presisi = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP+FP} \tag{4}$$

$$F - measure = 2x\frac{PrecisionxRecall}{Precision + Recall} \tag{5}$$

3 RESULT AND DISCUSSION

3.1 Features Combination

There are 3 sets of features combination. First, from the literature review, there are 15 features as seen in Figure 3. Second, the researcher chose 12 features from the researcher's understanding of malware behavior as seen in Figure 4. Third, from PCA, there are 23 features derived from 79 features originally.

No	Feature Name
1.	Source port
2.	Destination port
3.	L3 / L4 Protocol identifier
4.	Total number of packets
5.	Total number of bytes
6.	Mean of number of bytes per packet
7.	Standard deviation of number of bytes per packet
8.	Number of packets per second
9.	Number of bytes per second
10.	Flow duration
11.	Mean of inter-arrival time (IAT)
12.	Standard deviation of IAT
13.	Ratio of number of packets OUT/IN
14.	Ratio of number of bytes OUT/IN
15.	Ratio of IAT OUT/IN

Figure 3: Literature review features.

1.	Forward packets
2.	Total forward packets
3.	Forward packet length max
4.	Active mean
5.	Backward packets / second
6.	Forward IAT standard deviation
7.	Max packet length
8.	Total backward packets
9.	Total length of backward packets
10.	Backward IAT standard deviation
11.	FIN flag count
12.	Packet length variance

Figure 4: Researcher features.

3.2 Discussion

The implementation and testing environment is carried out in cloud computing because the CSV data that must be processed is quite large, both for training and testing. The weight configuration on the Neural Network was generated randomly for the first training, then the weight is updated. The training was rerun until the epoch was finished. Then the testing was run with feed-forward algorithm. Figure 5 shows a comparison of Neural Network results with literature review features, researcher features, and PCA. Complete test results for each set of features are given in the supplement of this article.

The highest F-measure was achieved for hidden neurons equal 12. These results are consistent with Stevanovich's research (Stevanovic and Pedersen, 2014) (Stevanovic and Pedersen, 2015), which stated that the more hidden neurons used, the performance of Neural Networks tends to be better until finding a saturation point. The highest F-measure also achieved for epoch equals 300, the maximum configuration. These results are different for the learning rate.

Increasing learning rate does not guarantee that the results of the F-measure will also be better. In researcher features and PCA features, the best results are achieved when the learning rate is 0.05 only. The comparison of learning rate and epoch for the combination of each feature set in hidden neurons 12 can be seen in Figure 6. The literature features achieve the best results with the maximum configuration of Neural Network parameters (learning rate = 0.1 and epoch = 300).

No	Features Set	Number of Feature	Hidden Neuron	Learning Rate	Epoch	Training Duration	Testing Durati on	Precision	Recall	F-measure
1	Literature	15	12	0.1	300	33m 50s	9s	47.58%	97.88%	0.6404
2	Resear cher	12	12	0.05	300	32m 43s	7s	55.89%	82.39%	0.6660
3	PCA	23	12	0.05	300	31m 21s	6s	69.47%	78.92%	0.7389

Figure 5: Neural Network results.

Learning rate is how much change given to the weight based on the error value, while epoch shows the number of iterations performed by the computer. A learning rate that is too big or small can make the new weight more distant than the expected results.

From Figure 6, it appears that for the combination of researcher features, some learning processes produce a value of 0 for precision, recall, and Fmeasure. This happened because the model created with these parameters is underfitting when the learning rate is 0.01 and 0.1.

Features Set	Learning Rate	Epoch	Precision	Recall	F-measure
Literature	0.1	100	47.34%	98.07 %	0.6386
		200	47.44%	98.14 %	0.6396
		300	47.58%	97.88 %	0.6404
	0.05	100	63.83%	49.85 %	0.5598
		200	64.46%	49.78 %	0.5618
		300	65.30%	49.75 %	0.5648
	0.01	100	70.69%	42.65 %	0.532
		200	71.30%	42.57 %	0.5331
		300	71.68%	42.38 %	0.5326
Researcher	0.1	100	0.00%	0.00%	0
		200	0.00%	0.00%	0
		300	0.00%	0.00%	0
	0.05	100	53.93%	85.13 %	0.6603
		200	54.94%	83.45 %	0.6626
		300	55.89%	82.39 %	0.6660
	0.01	100	0.00%	0.00%	0
Features Set	Learning Rate	Epoch	Precision	Recall	F-measure
		200	0.00%	0.00%	0
		300	0.00%	0.00%	0
PCA	0.1	100	55.92%	90.29 %	0.6907
		200	56.35%	90.33 %	0.694
		300	56.58%	90.19 %	0.6954
	0.05	100	68.68%	78.43 %	0.7323
		200	68.65%	79.28 %	0.7358
		300	69.47%	78.92 %	0.7389
	0.01	100	77.23%	67.18 %	0.7186
		200	78.52%	66.45 %	0.7198
		300	78.75%	66.59 %	0.7216

Figure 6: Comparison of Neural Network result of features set in hidden neuron = 12.

PCA features set gives the best result (F-measure = 0.7389). This result is significantly higher than the other two sets. The nature of PCA calculating the covariance between feature, generates set of features with the highest F-measure. Interestingly, the training duration only took 31 minutes and 21 seconds, the lowest between the three sets, even though the number of features is the highest (23 features). The PCA features seemingly make the algorithm running more efficient or the results converging faster.

Another worth mentioning result is the Fmeasure of 12 researcher features, which is higher than the 15 literature features (0.6660> 0.6404). This shows that using more features does not necessarily increase the accuracy of malware detection with the Neural Network. It appears that the two sets of feature combinations do not intersect but have slightly different F-measure values. This phenomenon happened because the more features used, the more internal errors are involved in the learning process. Each feature has internal errors, such as errors due to measurement or errors due to rounding values (Lim et al., 2015). Another factor is that each feature has its contribution to malware detection, and the features which are combined may have the effect of eliminating each other so that the detection accuracy decreases (Celik et al., 2015).

4 CONCLUSIONS

Detection of cyber malware based on network traffic features using Neural Network results in different Fmeasure values for various combinations of features. The literature features set (15 features) produces an F-measure of 0.6404, the researcher features set (12 features) produces an F-measure of 0.6660, and the PCA features (23 features) produces an F-measure of 0.7389. This concludes that PCA can generate features that have better results for malware detection with Neural Network algorithm. Aside from the PCA result, it is shown that more features used do not mean that the accuracy of malware detection will also increase.

This research used a dataset with a total of 442,240 data, which is a combination of existing datasets and the results of laboratory experiments. It is recommended that the Neural Network model result can be used for real-time malware detection on IoT devices and smartphones. Also, further research is needed on the analysis of the combination of network traffic features besides using PCA. The drawback of using PCA is the loss of interpretability. Without domain expertise and a lot of guessing, it is difficult to know the meaning of features derived from the PCA method.

ACKNOWLEDGEMENTS

This research is partially funded by Ministry of Research, Technology and Higher Education of Republic of Indonesia.

REFERENCES

Celik, Z. B., Walls, R. J., McDaniel, P., and Swami, A. (2015). Malware traffic detection using tamper resistant features. In *MILCOM 2015-2015 IEEE Military Communications Conference*, pages 330–335. IEEE.

for Cybersecurity, C. I. (2017). Android adware and general malware dataset.

Forouzan, B. A. (2010). Tcp/ip protocol suite. copyright© 2010 by the mcgraw-hill companies.

Kaspersky (2015). Mobile malware.

Kaushik, P. and Jain, A. (2015). Malware detection techniques in android. *International Journal of Computer Applications*, 122(17).

Lashkari, A. H., Kadir, A. F. A., Gonzalez, H., Mbah, K. F., and Ghorbani, A. A. (2017). Towards a network-based framework for android malware detection and characterization. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pages 233–23309. IEEE.

Lim, H., Yamaguchi, Y., Shimada, H., and Takakura, H. (2015). Malware classification method based on sequence of traffic flow. In *2015 International Conference on Information Systems Security and Privacy (ICISSP)*, pages 1–8. IEEE.

Lueg, C. (2017). 8,400 new android malware samples every day, april 2017.

Rashid, T. (2016). *Make your own neural network*. CreateSpace Independent Publishing Platform.

Stevanovic, M. and Pedersen, J. M. (2014). An efficient flow-based botnet detection using supervised machine learning. In *2014 international conference on computing, networking and communications (ICNC)*, pages 797–801. IEEE.

Stevanovic, M. and Pedersen, J. M. (2015). An analysis of network traffic classification for botnet detection. In *2015 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, pages 1–8. IEEE.

Yan, R. and Liu, R. (2014). Principal component analysis based network traffic classification. *Journal of computers*, 9(5):1234–1240.

Zhang, J., Xiang, Y., Wang, Y., Zhou, W., Xiang, Y., and Guan, Y. (2012). Network traffic classification using correlation information. *IEEE Transactions on Parallel and Distributed systems*, 24(1):104–117.

Zhou, Y. and Jiang, X. (2012). Dissecting android malware: Characterization and evolution. In *2012 IEEE symposium on security and privacy*, pages 95–109. IEEE.