Appendix A
A Survey on Malware Detection from Deep Learning

This appendix discusses a survey on malware detections from deep learning. Malware detections are also an important issue due to the increasing number of malware and similar approach as IDS.

A.1 Automatic Analysis of Malware Behavior Using Machine Learning

Recently, computer security faces the increase of security challenges. Because the static analysis is perceived as vulnerable to obfuscation and evasion attack, Rieck et al. [1] tries to develop dynamic malware analysis. The primary challenge in using dynamic malware analysis is the time needed to perform the analysis. Furthermore, as the amount and diversity of malware increases, the time required to generate detection patterns is also longer. Therefore, Rieck et al. propose malware detection method to improve the performance of malware detector based on behavior analysis. In this experiment, Malheur datasets were used. These datasets were created by themselves using behavior reports of malware binaries from anti-malware vendors, Sunbelt Software. Com. Each sample was executed and monitored using CW Sandbox's analysis environment and generates 3,131 behavior reports.

In this experiment, Rieck et al. [1] used four main steps. First, malware binaries were executed and monitored in a sandbox environment. It would give system calls and arguments as the output. Then, in step 2, the sequential reports produced from the previous step were embedded into a high-dimensional vector space based on its behavioral pattern. By doing this, the vectorial representation geometrically could be analyzed, to design clustering and classification method. In step 3, the machine-learning techniques were applied for clustering and classification to identify the class of malware. Finally, incremental analysis of malware's behavior was done

K. Kim et al., *Network Intrusion Detection using Deep Learning*,
SpringerBriefs on Cyber Security Systems and Networks,
https://doi.org/10.1007/978-981-13-1444-5

by alternating between clustering and classification step. The result shows that the proposed method successfully reduces the run time and memory requirement by processing the behavior reports in a chunk. Incremental analysis needed 25 min for processing the data, while regular clustering took 100 min. Furthermore, the regular clustering required 5 Gigabytes of memory during computation while incremental analysis only needed less than 300 Megabytes. So, it can be concluded that incremental technique in behavior-based analysis gives better performance in time and memory requirement than regular clustering.

A.2 Deep Learning for Classification of Malware System Call Sequences

Nowadays, the number and variety of malware are kept increasing. As a result, malware detection and classification need to be improved to do safety prevention. This paper wants to model malware system call sequences and uses it to do classification using deep learning. The primary purpose of leveraging machine learning in this experiment is to find a fundamental pattern in a large dataset. They used malware sample dataset gathered from Virus Share, Maltrieve, and private collections. There are three main contributions in this paper. First, Kolosnjaji et al. built DNN and implemented it to examine system call sequences. Then, in order to optimize malware classification process, convolution neural networks and RNN were combined. Finally, the performance of their proposed method was analyzed by examining the activation pattern of the neural unit.

During malware classification process, Kolosnjaji et al. [2] utilized malware collection from the dataset as the input of Cuckoo Sandbox. Then, the sandbox would give numerical feature vector as the output. After that, they used TensorFlow and Theano framework to construct and train the neural networks. It would give a list of malware families as the output of the NN. The neural network consisted of two parts, convolutional part and recurrent part. Convolutional part consisted of convolutional and pooling layer. At first, convolutional layer captured the correlation between neighboring input vectors and generated new features, resulting in feature vectors. Then, the result of the convolutional layer was forwarded to the input of recurrent layer. In recurrent layer, LSTM cells were used to model the resulting sequence and sort the importance based on mean pooling. Finally, dropout and a softmax layer were used to prevent overfitting to occur in the output. As the experiment result, it showed that the combination of convolutional network and LSTM gave better accuracy (89.4%) compared to feedforward network (79.8%) and convolutional network (89.2%).

A.3 Malware Detection with Deep Neural Network Using Process Behavior

The background problem of this paper is detecting whether there is malware infection on a computer based on the data traffic. To analyze this, usually, expert knowledge is needed, and the amount of time consumed is not a little. As a result, the purpose of this paper is to propose a method to detect malware infection using traffic data by utilizing machine learning. Tobiyama et al. leveraged recurrent neural network (RNN) for feature extraction and CNN for classification. This paper is good because, during the training phase using RNN, they used LSTM. RNN is known with the error vanishing problem because of its sequential structure. Its output depends on the previous input. As a result, when the previous input is getting bigger over time, an error will occur. LSTM avoids the error problem by selecting the only required information for future output to reduce the number of data.

In this paper, Tobiyama et al. [3] used 81 malware log files and 69 benign process log files for training and validation. The dataset was generated by using Cuckoo Sandbox to run malware files in an emulated environment. Then, they traced malware process behavior to determine generated and injected processes. The methodology of this paper is as follows: (1) generating log files during behavior process monitoring; (2) feature extraction using RNN, based on the log files from step 1; (3) converting the extracted features into image features; (4) training the CNN by using the image features; and (5) evaluating the process of validation using trained model. The result shows that the proposed model achieved 92% of detection accuracy. The drawback is the dataset was too small; they performed 5 min logging for 10 times, so the proposed system is not tested for large-scale data.

A.4 Efficient Dynamic Malware Analysis Based on Network Behavior Using Deep Learning

Nowadays, malware detection methods can be divided into three categories: static analysis, host behavior-based analysis, and network behavior analysis. Static analysis method can be evaded using packing techniques. Host behavior-based analysis can be deceived by using code injection. As a result, network behavior analysis becomes the spotlight because it does not have those vulnerabilities, and the need to communicate between the attacker and the infected host makes this method effective. One main challenge that hinders the use of network behavior analysis in malware detection is the analysis time. Various malware samples need to be collected and analyzed for a long period because the user does not know when the malwares start their activity. The main idea proposed by Shibahara et al. of this paper is aimed at two characteristics of malware communication, the change in communication purpose and common latent function. For the dataset, firstly they collected malware samples from VirusTotal, which were detected as malware

by antivirus program. Then, they used malware samples in VirusTotal that have different sha1 hash compared to previously collected malware samples. They used 29,562 malware samples in total for training, validation, and classification.

Their methodology consists of three main steps: feature extraction, neural network construction, and also training and classification label. First, Shibahara et al. [4] extracted features from communications, which are collected with dynamic malware analysis. Then, these features are used as inputs of recurrent neural network (RNN). Then, in NN construction phase, the change in communication purpose was captured. During the training and classification, the feature vectors of root nodes were calculated according to VirusTotal. Then, based on the vectors, it gave the classification result. During the experiment, the analysis time and time reduction between the proposed method and the regular continuation method were compared. The result shows that the proposed method reduces 67.1% of analysis time and keeps the range of covered URL to 97.9% compared to full analysis method.

A.5 Automatic Malware Classification and New Malware Detection Using Machine Learning

The development of malware is going rapidly. Various malwares keep appearing, increasing the diversity of malware family. As a result, traditional static-based malware analysis cannot detect these new kinds of malwares. Because of that reason, Liu et al. [5] propose a machine-learning-based malware analysis system. In order to support this research, a significant amount of malware information was collected by using ESET NOD32, VX Heavens, and Threat Trace Security from their campus network. Approximately 21,740 malware samples that belong to 9 families, including viruses, worms, Trojan, backdoor, etc., were collected. A total of 19,740 samples were used for training, and 2,000 samples were used for testing.

The methodology consists of three main modules: data processing, decision-making, and malware detection. Data processing module includes gray-scale images, Opcode n-gram, and import function to extract the malware features. Decision-making module does the classification process and identifies the malware based on the features extracted from data processing module. It consists of Classifier1, Classifier 2, Classifier 3, until Classifier N. Finally, clustering process in malware detection module is used to discover new kind of malware by using Shared Nearest Neighbor (SNN) clustering algorithm. The result of clustering is either unambiguous or ambiguous. For the ambiguous one, SNN was implemented to identify it. In the experiment, the performance of the proposed method was tested by using several classifiers like random forest, K-Nearest Neighbor (kNN), gradient boosting, naive Bayes, logistic regression, SVM, and decision tree. The result of the experiment shows the average accuracy of the proposed method is 91.4%, while the best accuracy is 96.5% by using the random forest classifier.

A.6 DeepSign: Deep Learning for Automatic Malware Signature Generation and Classification

Recently, a variety of malwares keep growing, including a new variant of malwares that are undetected by antivirus software. Several methods have been proposed to overcome this problem, for instance, by using signatures based on specific vulnerabilities, payloads, and honeypots. However, all these methods have one big problem; they target a specific aspect of malware. As a result, if an adversary modifies small parts of their malware, it will not be detected by them. Because of that background, David et al. [6] proposes a method for a signature generation that does not depend on a particular part of malware, so it can be resistant to code modifier. In this research, David et al. used a dataset that consists of six categories of malware, such as Zeus, Carberp, Spy-Eye, Cidox, Andromeda, and DarkComet. They used a total of 1800 samples, which is 300 samples for each category.

The methodology consists of four main parts. First, malware program was run in Cuckoo Sandbox, an emulated environment. The output of the sandbox was sandbox log file. Then, the log file was converted into a binary bit string. The bit string was used as the input of Deep Belief Neural Network (DBN) to produce 30 sized vectors as its output layer. By doing that step, malware signature was generated. The question here is how they convert sandbox files to fixed size input. In order to do that, first, David et al. extracted all unigrams for each sandbox file in the dataset. Then for each unigram, the number of files that it appears was counted. After that, they selected top 20,000 with the highest frequency and finally converted each sandbox file to a 20,000 sized bit string. During the experiment, they used 1,200 samples for training (200 samples for each malware category) and 600 samples for testing (100 samples for each category). The number of features generated is 30, and the accuracy is 98.6%. The input noise was also set as 0.2 and learning rate as 0.001.

A.7 Selecting Features to Classify Malware

Since the development of machine learning several years ago, there have been various ideas to implement machine learning as an engine in malware detection software. However, because there is a time delay between malware landing on user's system and the signature generation process, it can harm the user. Because of that reason, Raman et al. utilized data mining to identify seven key features in Microsoft PE file format that can be used as input to the classifier. The seven features would be used in machine-learning algorithm to do malware classification. Raman et al. generate their dataset from PE files. Firstly, they write a parser to extract features from PE file. They use their experience in malware analysis to select a set of 100 features from the initial 645 features. Finally, they created a dataset of 5,193 dirty files and 3,722 clean files to evaluate those 100 features.

The focus of the methodology is feature extraction and feature selection, with the addition of combining intuitive method and machine-learning method during feature selection. First, Raman et al. [7] used their knowledge to reduce the number of the feature from 645 to 100. Then, random forest algorithm was utilized to choose 13 features. Finally, four classifiers (J48Graft, PART, IBk, and J48) were used to check the accuracy of each feature and choose the highest seven features. Those features are debug size that denotes the size of the debug directory table, image version that denotes the version of the file, debugRVA that denotes the relative virtual address of the import address table, ExportSize that denotes the size of the export table, ResourceSize that denotes the size of the resource section, VirtualSize2 that denotes the size of the second section, and NumberOfSections that denotes the number of section.

A.8 Analysis of Machine-Learning Techniques Used in Behavior-Based Malware Detection

The main problem in this paper is about the increase of malware varieties, which lead to vulnerable manual heuristic malware detection method. In order to cope with this problem, Firdausi et al. proposed an automatic behavior-based malware detection method utilizing machine learning. In this research, they used datasets in the format of Windows Portable Executable. The dataset consisted of 250 benign instances collected from System 32 of Windows XP 32 bit SP2. They also collected 220 malware samples from various resources. For monitoring process, Firdausi et al. [8] used Anubis, a free online automatic dynamic analysis service to monitor both malware and benign samples. Then, the performance of five classifiers was compared using their proposed method. The five classifiers were kNN, naive Bayes, J48 DT, SVM, and Multi-Layer Perceptron (MLP).

For the methodology of the research, firstly Firdausi et al. did data acquisition from community and virology.info. Then, the behavior of each malware was analyzed on an emulated sandbox environment. Anubis Sandbox was chosen for API hooking and system call monitoring. After that, the report would be processed into sparse vector model. The report was generated in xml format. For the next step, XML file parsing, feature selection, and feature model creation were done based on that XML file. Finally, the last step was doing classification based on that model. Firdausi et al. applied machine-learning tools, did parameter tuning, and finally tested their scheme. The result of this experiment shows that feature extraction reduces the attributes from 5,191 to 116 attributes. By performing feature selection, the time consumed to train and build the model became shorter. The overall best performance was achieved by J48 classifier with 94.2% true positive rate, 9.2% False Positive Rate (FPR), 89.0% precision, and 92.3% accuracy.

A.9 Malware Detection Using Machine-Learning-Based Analysis of Virtual Memory Access Patterns

For several years, people know that traditional malware detection can be divided into the static and dynamic method. These methods are usually implemented in antivirus software. Static method uses signature database to detect malware; on the other hand, dynamic method runs the suspicious program to check its behavior, whether it is a malware or not. However, there is a problem here. The software is vulnerable to malware exploit during infection, and it can be disabled by malware. So, in this paper, Xu et al. proposed an idea of hardware-assisted detection mechanism which is not vulnerable to such disabling problem. However, this idea relies on expert knowledge of the executable binary and its memory layouts. So, as a solution, they decided to use machine learning to detect a malicious action of malware. The primary purpose of this paper is to learn one model for each application that separates malware-infected execution from legitimate execution. Furthermore, Xu et al. will do the classification based on memory access pattern.

In order to monitor the memory access, Xu et al. [9] did epoch-based monitoring. A monitoring method divided program execution into epochs. Then, each epoch was separated by inserting a sign in a memory stream. They found that for most malicious behavior, the deciding feature was the location and frequency of memory accesses rather than their sequence. Then, after the monitoring had finished, the classifier was trained by using summary histograms for epochs. During training, the program was executed, and each histogram was labeled either malicious or benign. After the training model had been defined, the binary signature was verified, and the model was loaded into hardware classifier. Finally, the last step was hardware execution monitoring. If malware were detected, an authenticated handler would be launched automatically. During the experiment, they used three classifiers (SVM, random forest, and logistic regression) and compared their performance. The best performing classifier was random forest, with 99% true positive rate and less than 1% FPR.

A.10 Zero-Day Malware Detection

Lately, the variety of malware keeps increasing and threating the security of our computer system. People cannot rely on traditional signature-based antivirus anymore. Zero-day malware will easily bypass regular antivirus because their signature is not in the antivirus database yet. To solve this problem, Gandotra et al. [10] proposed a combination of static and dynamic malware analysis with machine-learning algorithm for malware detection and classification. However, there were several problems with this scheme. First, this scheme had high FP and False Negative Rate (FNR). Second, it took time to build the classification model because of the large dataset. As a result, early malware detection was not possible with this scheme.

Knowing these problems, Gandotra et al. concluded that the challenge here was to select the relevant set of features so the building time could be reduced and the accuracy would be improved. The dataset that is used in this experiment was taken from VirusShare. About 3,130 portable executable files, which include 1,720 malicious and 1,410 clean files, were utilized. All files were executed in a sandbox to get their attributes and then were used to build the classification model using WEKA.

The methodology consisted of six main steps. The first step was data acquisition. In this step, they collected malware samples targeting Windows OS from VirusShare database. They also collected clean files manually from system directories of Windows. The second step was automated malware analysis. In this phase, they used modified Cuckoo Sandbox to execute the specimen and generate the result as Java Script Object Notation (JSON) file. The third step was feature extraction. In this step, the JSON reports generated by Cuckoo Sandbox were parsed to obtain the various malware features. The result was a feature set of 18 malware attributes which can be used to build the classification model. The next step was feature selection. They selected seven top features by using IG method. IG method is an entropy-based method for feature evaluation which is broadly used in machine learning. The last step was classification. They used the selected seven features to build the classification model using machine-learning algorithm in WEKA library. The used seven classifiers are IB1, naive Bayes, J48, random forest, bagging, decision table, and multilayer perceptron. The result of their experiment showed that random forest gave the best accuracy with 99.97% and the time to build model was 0.09 s.

References

1. K. Rieck, P. Trinius, C. Willems, and T. Holz, "Automatic analysis of malware behavior using machine learning," *Journal of Computer Security*, vol. 19, no. 4, pp. 639–668, 2011.
2. B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep learning for classification of malware system call sequences," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2016, pp. 137–149.
3. S. Tobiyama, Y. Yamaguchi, H. Shimada, T. Ikuse, and T. Yagi, "Malware detection with deep neural network using process behavior," in *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual*, vol. 2. IEEE, 2016, pp. 577–582.
4. T. Shibahara, T. Yagi, M. Akiyama, D. Chiba, and T. Yada, "Efficient dynamic malware analysis based on network behavior using deep learning," in *Global Communications Conference (GLOBECOM), 2016 IEEE*. IEEE, 2016, pp. 1–7.
5. L. Liu, B.-s. Wang, B. Yu, and Q.-x. Zhong, "Automatic malware classification and new malware detection using machine learning," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, no. 9, pp. 1336–1347, 2017.
6. O. E. David and N. S. Netanyahu, "Deepsign: Deep learning for automatic malware signature generation and classification," in *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, 2015, pp. 1–8.
7. K. Raman *et al.*, "Selecting features to classify malware," *InfoSec Southwest*, vol. 2012, 2012.

8. I. Firdausi, A. Erwin, A. S. Nugroho, *et al.*, "Analysis of machine learning techniques used in behavior-based malware detection," in *Advances in Computing, Control and Telecommunication Technologies (ACT), 2010 Second International Conference on*. IEEE, 2010, pp. 201–203.
9. Z. Xu, S. Ray, P. Subramanyan, and S. Malik, "Malware detection using machine learning based analysis of virtual memory access patterns," in *Proceedings of the Conference on Design, Automation & Test in Europe*. European Design and Automation Association, 2017, pp. 169–174.
10. E. Gandotra, D. Bansal, and S. Sofat, "Zero-day malware detection," in *Embedded Computing and System Design (ISED), 2016 Sixth International Symposium on*. IEEE, 2016, pp. 171–175.