KSII TRANSACTIONS ON INTERNET AND INFORMATION SYSTEMS VOL. 14, NO. 12, Dec. 2020 4909
Copyright ⓒ 2020 KSII

Semi-supervised based Unknown Attack Detection in EDR Environment

Chanwoong Hwang[1], Doyeon Kim[1], and Taejin Lee[1*]
[1] Department of Information Security, Hoseo University
Asan, South Korea
[e-mail: hcw85123@gmail.com, ehdus3342@gmail.com, kinjecs0@gmail.com]
[*]Corresponding author: Taejin Lee

Received November 19, 2020; accepted November 21, 2020; published December 31, 2020

Abstract

Cyberattacks penetrate the server and perform various malicious acts such as stealing confidential information, destroying systems, and exposing personal information. To achieve this, attackers perform various malicious actions by infecting endpoints and accessing the internal network. However, the current countermeasures are only anti-viruses that operate in a signature or pattern manner, allowing initial unknown attacks. Endpoint Detection and Response (EDR) technology is focused on providing visibility, and strong countermeasures are lacking. If you fail to respond to the initial attack, it is difficult to respond additionally because malicious behavior like Advanced Persistent Threat (APT) attack does not occur immediately, but occurs over a long period of time. In this paper, we propose a technique that detects an unknown attack using an event log without prior knowledge, although the initial response failed with anti-virus. The proposed technology uses a combination of AutoEncoder and 1D CNN (1-Dimention Convolutional Neural Network) based on semi-supervised learning. The experiment trained a dataset collected over a month in a real-world commercial endpoint environment, and tested the data collected over the next month. As a result of the experiment, 37 unknown attacks were detected in the event log collected for one month in the actual commercial endpoint environment, and 26 of them were verified as malicious through VirusTotal (VT). In the future, it is expected that the proposed model will be applied to EDR technology to form a secure endpoint environment and reduce time and labor costs to effectively detect unknown attacks.

Keywords: Endpoint Security, EDR, Unknown Attack Detection, AutoEncoder, 1D CNN

This research was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2018R1C1B5029849)

1. Introduction

Recently, as the popularity of bring your own device (BYOD) and IoT has increased, the number of individual devices connected to an organization's network is rapidly increasing. According to the Cisco Annual Internet Report [1], the number of devices connected to the IP network is expected to be more than three times the world's population by 2023. In 2023, there will be 3.6 network devices per person, with a total of 293 billion network devices. If the device is connected to a network, it is considered an endpoint. **Fig. 1** shows a sample of what is considered an endpoint. These are the entry points for threats and malware, so endpoints like mobile and remote devices are targeted. It attempts to access the internal network through an endpoint with security flaws or vulnerabilities. Attackers who have access to the internal network can attempt various malicious actions, such as data loss or corruption.

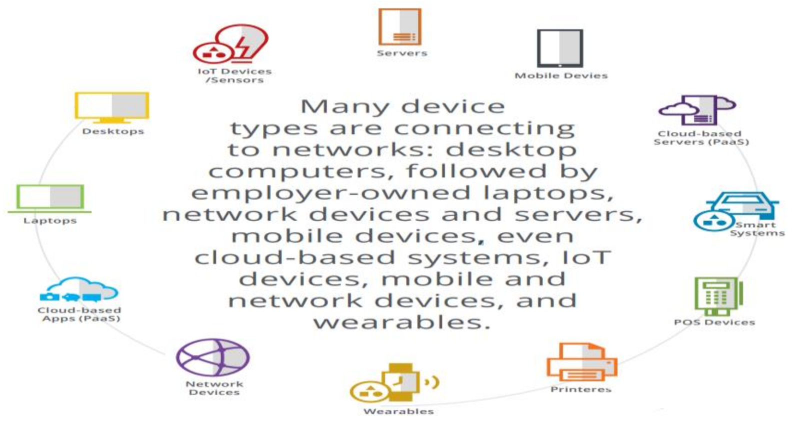

Many device types are connecting to networks: desktop computers, followed by employer-owned laptops, network devices and servers, mobile devices, even cloud-based systems, IoT devices, mobile and network devices, and wearables.
IoT Devices /Sensors
Servers
Mobile Devies
Desktops
Cloud-based Servers (PaaS)
Laptops
Smart Systems
Cloud-based Apps (PaaS)
POS Devices
Network Devices
Wearables
Printeres

Fig. 1. Example of Endpoint

Endpoint security started with a traditional signature security solution. It detects changes to the file system or application that match known patterns and blocks the program from running. This made the Internet and e-commerce increasingly popular, making malware more and more frequent, complex, and difficult to detect. In addition, new security technologies are needed as fileless malware is no longer dependent on signatures. This bypasses existing signature-style security solutions without creating a file. For example, a malicious Dynamic Link Library (DLL) is injected into a normal process or a malicious VBScript is inserted into a normal Microsoft Office document. Moreover, it modifies the registry to attempt additional malicious behavior. Thus, a combination of machine learning and Artificial Intelligence (AI) can block unknown attacks from existing knowledge such as firewalls, reputations and heuristics, rather than traditional security solutions that rely on signatures.

We propose an anomaly detection approach through AutoEncoder and 1D-CNN that reflects temporal features in endpoint environments. Anomaly detection is not up-to-date and has been studied for a long time, but it has been highlighted relatively recently with big data. In other words, anomaly detection aims to find anomaly data in a given dataset. Anomaly detection approaches are used in a variety of areas, including financial fraud detection, network intrusion detection, human behavior analysis, and gene expression analysis [2-5]. Anomaly detection has also been studied in the field of time series data with time characteristics. In time series data, because a specific point in time is greatly affected by features before and after that point, it is usually necessary to select the appropriate small window size of time to proceed with the analysis. Analytical goals are categorized as finding abnormal point of time or patterns of abnormal change. In a dataset, when one point is judged to be abnormal for the other points, it is called point anomaly and when one or more data is judged to be abnormal in a particular data areas, it is called a collective anomaly. In addition, when an object within a dataset is deemed strange in a particular context, it is called contextual anomaly. It is also classified as unsupervised learning and supervised learning, depending on whether labels are available for anomaly in past datasets. Most of the existing log-based detection techniques work by using network logs to detect and update the database [6]. In recent years, effective research has been conducted on log analysis based on machine learning or large-scale logs, but it is insufficient [7-11]. Anomaly detection is different from a model that simply classifies malicious and normal. Ahmed et al. [12] explained the research challenge that publicly labeled datasets are not available despite the many techniques available for anomaly detection. Therefore, this paper proposes a technique to detect unknown attacks without prior knowledge, such as without a label. Based on semi-supervised learning, data collected for one month from a commercial endpoint environment is assumed to be normal, and normal data and out-of-bounds data are detected. It is a model that reflects continuous data in chronological order occurring at the endpoint. We expect the proposed model to make it easier for endpoints to recognize previously unseen attack behaviors, reducing the time and labor costs of dealing with new attacks.

Section 2 describes related studies on data processing that reflects common anomaly detection and sequences. Section 3 proposes an anomaly detection model at the endpoint. Section 4 provides anomaly detection results using the proposed model and discusses operational policies. Finally, Section 5 has a conclusion.

2. Related Work

Research on technology for anomaly detection of data related to ICS and EDR has been conducted [13, 14]. Alrashdi et al. [15] proposed a network-based IoT Anomaly Detection System (AD-IoT) using a Random Forest (RF) algorithm among machine learning algorithms. Attacks were detected by identifying normal traffic and abnormal traffic using False Positive Rates. Experiments were conducted using the UNSW-NB15 dataset, and **Table 1** shows the accuracy, recall, and F1-score for normal/abnormal traffic prediction using RF. An average accuracy of about 98% was derived, and recall and F1-Score were also about 98%.

Table 1. Performance of binary classification

Model	Predicted	Precision	Recall	F1-Score
RF	Normal	0.99	0.99	0.99
	Attack	0.79	0.97	0.86
	Avg/total	0.98	0.98	0.98

Kravchik et al. [16] conducted a study to detect cyberattacks on ICS using CNN. SWaT dataset was used, and the dataset includes 36 various cyberattacks. The experiment was conducted with a model combining 1D-CNN and Long Short-Term Memory (LSTM) that can be quickly trained in detecting anomalies of ICS, rather than 2D-CNN, which is frequently used for image processing. **Fig. 2** shows the 1D-CNN model. In the 1D-CNN model, Convolution-ReLU-Maxpooling was applied to each feature along the time axis. In the model that combines CNN and LSTM, the data is processed in the convolution layer and then transmitted to the LSTM layer to predict the value. **Table 2** shows the F1-score for each model tested in the study. CNN-based model showed higher accuracy when detecting anomaly than CNN and LSTM combined models. The highest accuracy was obtained when 8 convolutional layers were arranged, but the CNN composed of 4 layers was also not much different from the CNN with 8 convolutional layers in accuracy, and progressed much faster in speed.

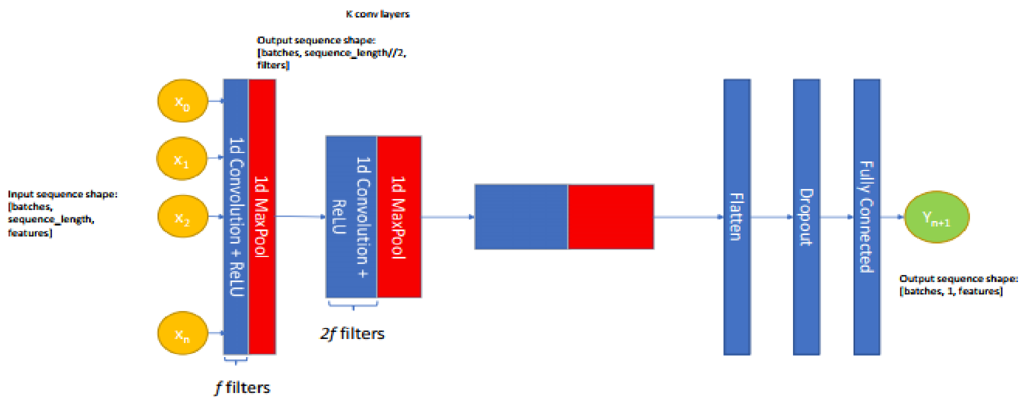

K conv layers
Output sequence shape: [batches, sequence_length//2, filters]
Input sequence shape: [batches, sequence_length, features]
x_0
x_1
x_2
x_n
1d Convolution + ReLU
1d MaxPool
1d Convolution + ReLU
1d MaxPool
f filters
2*f* filters
Flatten
Dropout
Fully Connected
Y_{n+1}
Output sequence shape: [batches, 1, features]

Fig. 2. 1D-CNN model configuration

Table 2. F1-scores per stage

Network configuration	P1	P2	P3	P4	P5	All
4 inception layer, kernel = 2, 32 filters	0.834	0.626	0.866	0.875	0.714	0.609
8 convolutional layers, kernel = 2, 32 filters	0.924	0.595	0.918	0.901	0.795	0.775
4 convolutional layers, kernel = 2, 32 filters	0.89	0.58	0.858	0.907	0.731	0.688
4 convolutional layers + 3 LSTM layers, kernel = 2, 32 filters	0.857	0.666	0.956	0.909	0.8	0.646
2 convolutional layers, kernel = 2, 32 filters	0.805	0.472	0.85	0.87	0.656	0.632
3 LSTM layers with state = 256	0.778	0.453	0.787	0.805	0.714	0.626

Kim et al. [17] detects anomalies in the event log using LOF and AutoEncoder among the anomaly detection techniques and suggests event rules generated through an attack profile. LOF was calculated based on the k-Nearest Neighbor (kNN) algorithm. The distance value between the reference point and the test data is converted into a score to determine whether it is a variant, and the larger the distance value is, the greater the difference from the normal data is. The AutoEncoder model trains independently, separating network behavior from system behavior. In addition, by analyzing the anomaly event, the malicious process and the threats that may occur are detected. Create an attack profile to create a rule that can be detected based on an attack log. Using the generated rules, users can be alerted when an attack on the endpoint occurs and it is effective in detecting an attack. **Fig. 3** shows the main attack scenario in the attack profile.

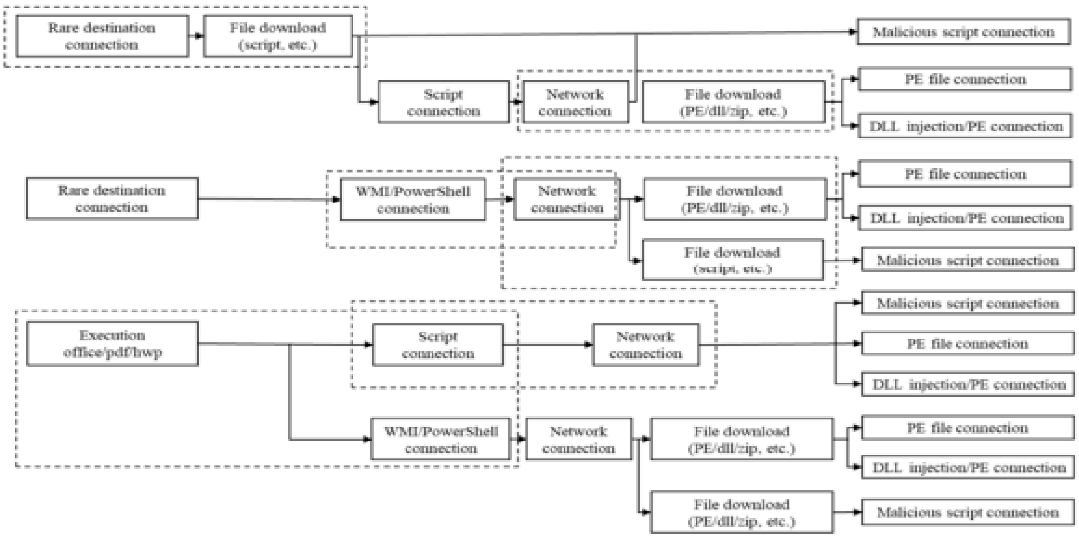

Rare destination connection
File download (script, etc.)
Malicious script connection
Script connection
Network connection
File download (PE/dll/zip, etc.)
PE file connection
DLL injection/PE connection
Rare destination connection
WMI/PowerShell connection
Network connection
File download (PE/dll/zip, etc.)
PE file connection
DLL injection/PE connection
File download (script, etc.)
Malicious script connection
Execution office/pdf/hwp
Script connection
Network connection
Malicious script connection
PE file connection
DLL injection/PE connection
WMI/PowerShell connection
Network connection
File download (PE/dll/zip, etc.)
PE file connection
DLL injection/PE connection
File download (PE/dll/zip, etc.)
Malicious script connection

Fig. 3. Main attack scenario in the attack profile

Independent Identically Distribution (IID) data means that random variables are distributed independently and equally, and Sequential data means that data is generated in a same stream in sequential order. Time Series means that data is generated from a stream with the same spacing between successive objects. There are Point anomaly, Contextual anomaly, and Collective anomaly to detect anomaly, and Deep Anomaly Detection (DAD), which detects all three types of anomaly. Point anomaly is classified as an anomaly as the distance between the single instance and the other instance is farther, and is mainly used for credit card fraud detection. Contextual anomaly detects abnormal signs considering context and is commonly used in time series data. When anomalous values exist between constant contexts (values), abnormal signs are judged. Collective anomaly, unlike point anomaly, detects anomaly for multiple instances rather than a single instance.

Andrade et al. [18] proposed a method for detecting anomaly in sequential data. For detecting anomaly, three data mining techniques and various algorithms were proposed: Unsupervised, Semi-supervised, and Supervised. The unsupervised learning is based on untrained data to detect anomaly and can be grouped into statistics, clustering, distance-based and density-based techniques. The supervised learning detects anomaly through learning the neural network. Semi-supervised learning combines a small amount of data with a label and a large number of data without a label to learn through machine learning. Training data with label is trained as supervised learning, and training data without label is trained as unsupervised learning. The anomaly detection using time series data uses a data set performed at the same time interval among continuously observed data, and generally measures a specific period. One example is the Symbolic Aggregate Approximation (SAX) algorithm. The SAX algorithm is an algorithm that reduces dimensions and duplication of time series data, which has the advantage of using less storage space. However, sliding windows are required as parameters to increase the accuracy of the algorithm. But, Recursive Ray Acoustics (RRA) algorithm improves this problem. Unlike the SAX algorithm, it is possible to detect anomaly without a time limit if there is no need for a sliding window value and only an initial value. **Fig. 4** shows the results of detecting anomaly with the RRA algorithm using an electrocardiography (EGC) dataset. Of the two, the figure above shows the location of the abnormal ECG, and the figure below shows the abnormal signs appearing at the location shown above using the regular density curve.

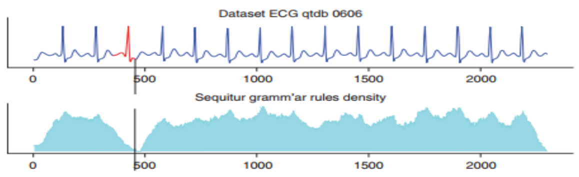


Fig. 4. Anomaly discovery in electrocardiography (EGC) dataset

3. Proposed Model

3.1 Overview

We use the feature difference between data that happened in the past and what is happening to detect anomaly at the endpoint. The larger the difference between normal data and new data that has not occurred in the past, the more it can be identified as anomalous. This section proposes an anomaly detection method for six fields collected from endpoints using AutoEncoder and 1D-CNN. **Fig. 5** shows the overall structure of the proposed model. First, we collect the logs collected from the endpoints in the internal network. Based on the collected data, it extracts the features for learning the anomaly detection engine. AutoEncoder and 1D-CNN are applied to calculate anomaly score that represent data differences. Since both models are deep learning algorithms, they return a loss value indicating the difference from the training model during the test. The returned loss value is used statistically. Computes the Cumulative Distribution Function (CDF) value expressed as a number between 0 and 1 using a standard normal distribution. The CDF value thus generated is used as the final anomaly score. Anomaly score is used to identify anomaly. These anomaly engines can access suspicious Internet Protocol addresses (IPs) or detect suspicious processes and classify data by process for detailed analysis. If anomaly are detected using anomaly score, you can also consider collective anomaly using flow data collection [19, 20]. Detected abnormal data is verified through VirusTotal (VT) [21]. VirusTotal shows the number of engines that have determined that more than 70 antiviruses are malicious. Anomaly data determined to be not malicious can be quickly excluded from verification through the allowlist policy.

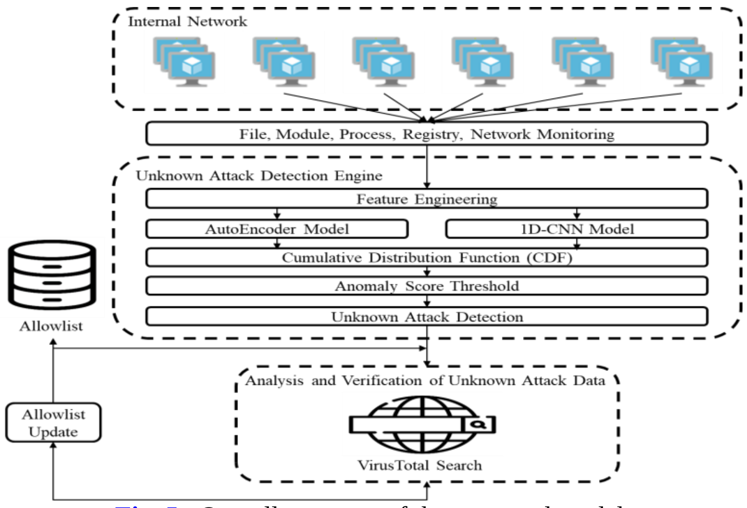

Internal Network
File, Module, Process, Registry, Network Monitoring
Unknown Attack Detection Engine
Feature Engineering
AutoEncoder Model
1D-CNN Model
Cumulative Distribution Function (CDF)
Anomaly Score Threshold
Unknown Attack Detection
Allowlist
Allowlist Update
Analysis and Verification of Unknown Attack Data
VirusTotal Search

Fig. 5. Overall structure of the proposed model

3.2 Unknown Attack Detection Engine

3.2.1 Feature Engineering

Only 6 specific fields are used from the collected endpoint log data. Extract 22 features using 6 fields. The 6 fields are categorized into process path, destination IP, event time, process type, event type, and file type. String data, such as process path fields, must be converted to vectors in order to train the model. We generate a fixed length vector through feature hashing. Before the feature hashing, strings are divided into n lengths consecutively using n-gram. Apply the generated n-gram list to the feature hashing function. The pseudo-code of the feature hashing function is shown in **Fig. 6**.

```
Function1 – Feature Hashing
Description. This function converts the strings of field to fixed size, V.

SET V : Fixed Vector Size
SET X : Array of Extracted n-gram

1.  Array Vector[V]=0
2.  FOR string In X :
3.      Point=HASH(string) Mod V
4.      Vector[Point]=Vector[Point]+1      # Count Operation
5.  Return Vector
```

Fig. 6. Feature hashing pseudo-code

After converting the string divided by n-gram into a unique number through the hash function, the remainder after the modular operation is counted. As a result, a field consisting of a string can generate a feature vector of a desired size through feature hashing. The destination IP field also creates a feature vector by feature hashing with a string including dots. Also, in order to add suspicious local IP address of the accessed IP address, class A and B are separated and Min-Max Scaling is applied. This is for network behavior only, and the system behavior is assigned a value of 0 because the value of the destination IP field does not exist. The event time field was divided into day of the week, weekday, and weekend, and features were created with different values every 3 hours. Also, the process type, event type, and file type fields are created with different values of features depending on the corresponding type. **Table 3** shows the processing method and input example of Feature Engineering. There are four major process types. First, processes such as *powershell.exe* or *cmd.exe* are classified as shell-based processes. Second, processes such as *wscript.exe* or *cscirpt.exe* are classified as script-based processes. Third, processes such as *WINWORD.EXE*, *EXCEL.EXE*, and *POWERPNT.EXE* are classified as word-based processes. Finally, all other processes are classified as general processes. File types are classified into *PE*, *script*, *ZIP*, *DOC*, *IMAGE*, and *MEDIA*. Event types are classified into *network*, *file*, *module*, *process*, *and registry*.

Table 3. Example of processing method and input of feature engineering

Used Field	Method of Feature Extraction	Input Example
Process Path	SHA256(2-gram(Process Path)) mod 10	C:\Program Files(x86)\Microsoft Office\POWERPNT.EXE
Destination IP	SHA256(2-gram(IP)) mod 4	127.0.0.1
	MinMaxScaling(IP.A_Class, IP.B_Class)	127.0
Event Time	If (EventTime = Moday): feature = 1	1564519160346

	else if (EventTime = Tuesday):
	feature = 2
	else if (EventTime = Sunday):
	feature = 7	

	If (EventTime =Weekday):
	feature = 1
	else if (EventTime =Weekend):
	feature = 2	

	If (EventTime = 0~3 time):
	feature = 0
	else if (EventTime = 3~6 time):
	feature = 1
	else if (EventTime = 21~24time):
	feature = 8	

Process Type	If (ProcessType = Normal):
	feature = 1
	else if (ProcessType = Shell):
	feature = 2
	else if (ProcessType =Word):
	feature = 3
		POWERPNT.EXE

Event Type	If (EventType = file):
	feature = 1
	else if (EventType = module):
	feature = 2
	else if (EventType = process):
	feature = 3
		file

File Type	If (FileType = PE):
	feature = 1
	else if (FileType = Script):
	feature = 2
	else if (FileType = Zip):
	feature = 3
		PE

3.2.2 AutoEncoder based unknown attack model

The collected endpoint log data is based on unsupervised learning because no label exists. We use the AutoEncoder model to calculate anomaly score and detect anomaly. AutoEncoder is a neural network that simply copies input to output as shown in **Fig. 7**.

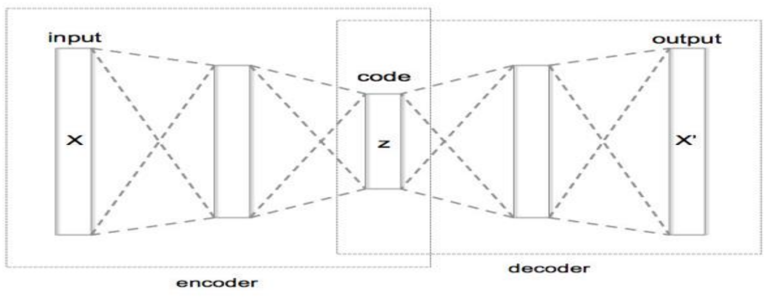


Fig. 7. AutoEncoder configuration

Using these AutoEncoder features, training the normal event log in the past makes use of the fact that the loss value is large when predicting abnormal data [22, 23]. The parameters of the AutoEncoder are learned in the direction of minimizing the loss, and it is judged that the greater the loss generated by the AutoEncoder model learned during testing, the greater the anomaly. The loss occurring at this time uses the Mean Squared Error (MSE). The loss value using the MSE function is shown in equation (1).

$$\delta_i = \frac{1}{p}\sum_{j=1}^{p}(x_{ij} - x_{ij}\text{`})^2 \tag{1}$$

If the input is x_i and the output is the same $x_i\text{`}$ as the input, the loss value δ_i is $i = 1, \cdots, n$ means the event log. Although the loss value δ_i can be used as an anomaly score, the proposed model used a statistical an anomaly score. Here, p means the number of features, which is the number of dimensions of input and output. In the experiment configuration, the input layer and output layer nodes are set to 22, which is the number of features, and the hidden layers are set in the order of 14, 6, and 14 to undergo the encoding and decoding process. Each layer's activation function uses the Relu function to return a value less than 0 as 0, and a value greater than 0 as it is.

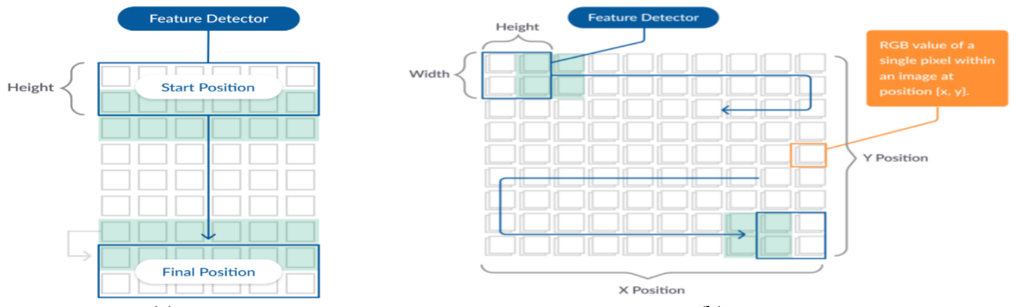
3.2.3 1D-CNN based unknown attack detection model

CNN is a kind of deep learning algorithm and was developed for classifying images. In the process of learning features, it uses 2D inputs representing the pixel and color channels of the image. CNN utilizes a number of filters that can be used as shared parameters to maintain spatial information of an image in 2D, effectively extract and learn features from adjacent images. CNN has the advantage of enabling easier learning with minimal parameters and pre-processing. The following is the formula of the output value according to the input data of CNN.

$$s(t) = (x * w)(t) = \sum x(a)w(t-a). \tag{2}$$

$$s(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{3}$$

Equation (2) is an output equation of the CNN layer for 1-dimensional input data. x is the input data, w is the kernel map and $s(t)$ is the output layer feature map. Equation (3) is an output equation of the CNN layer for 2D input data. I is the input data, K is the kernel map, and $s(i,j)$ is the output layer 2D feature map. CNN works the same way, whether in 1, 2 or 3 dimensions. The difference is the structure of the input data and how the filter, also called the convolution kernel or feature detector, moves the data. **Fig. 8** shows the difference between a 1D-CNN and a 2D-CNN.



(a) (b)

Fig. 8. Difference between 1D CNN and 2D CNN; (a) 1D CNN feature detector movement method; (b) 2D CNN feature detector movement method

We can apply 1D-CNN to sequence data. Extract features from sequence data and map internal features in the sequence. 1D-CNN is very effective at deriving features from fixed length segments of the entire dataset, and the location of the features in the segment is not critical. The 1D-CNN's final output is a one-dimensional vector because the feature detector moves in one direction and proceeds with convolution. We applied the 1D-CNN by learning the event log in sequence over time and predicting the next event log. We set the kernel size to 5 to reflect the sequence of the top 5 events. The filter size is set to 128 to create the feature map 1*128 vectors. The output is 22 features of the next event log. We ran a total of 5 epochs, and since 3 epochs the training evaluation has not changed noticeably.

3.2.4 Statistical Approach for Anomaly Score

We use statistically the loss values tested in the AutoEncoder and 1D-CNN models. Anomaly detection takes advantage of the fact that when testing normal data against a trained model, the loss that occurs is large. Statistical analysis of data by calculating z-score and CDF using loss values. **Fig. 9** shows the Example of cumulative distribution function.

$$\mathrm{P}(Z \le z') = \int_{-\infty}^{z'} f(z)dz$$

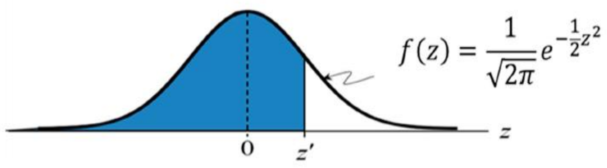


Fig. 9. Cumulative distribution function example

CDF indicates the probability that a random variable is less than or equal to a certain value for a particular probability distribution. Therefore, the CDF is calculated for statistical analysis using the loss value and the CDF value is used as the anomaly score. If the anomaly score is greater than the set threshold, it is judged as an anomaly data.

3.3 Analysis and Verification of Anomaly Data

VirusTotal is used to verify whether the anomaly detected data is actually a malicious process. VirusTotal is a website that provides file scanning for free. The site scans with a variety of antivirus engines such as FireEye, Kaspersky, Microsoft, Trend Micro, which are well known to us, and shows the results transparently. VirusTotal determines whether it is malicious/normal through various anti-virus engines through file upload, URL inspection, and hash value inspection. It also shows various hash values for files (MD5, SHA-1, SHA-256, Vhash, etc.) and file information. By verifying the hash value of the anomaly data, it extracts the malicious/normal status of each of more than 70 anti-virus engines and the name of the detected malicious code. However, because anti-virus may be misdiagnosed, if more than 5 anti-virus engines determine that the verification result is malicious, it is judged as malicious data.

4. Unknown Attack Detection Results

4.1 Dataset

The dataset used in the experiment was collected in a real commercial endpoint environment.

Table 4 shows the original dataset. The original dataset consists of 13 fields. Each event log has a specific process name. The *EventType* field can distinguish whether the event log is a network action or a system action. *EventSubType* is detailed in the *EventType* field behavior. In addition, there is a time field where the event log occurred and a corresponding endpoint *IP* address field. The *RemoteIP* field is present only when the *EventType* is network. This means the destination IP. The *ProPath* field means where the process of the event exists. *FilePath* means the location of the file, so it does not exist in network behavior. *FileType*, likewise, doesn't exist in network behavior, it means the file is a Portable Executable (PE) file or a file type like a document file. Since the original dataset is data collected in a real commercial environment, the file name and file path are encrypted. Therefore, other fields such as file name and file path cannot be used. The dataset collected at the endpoint was provided by G* Corporation. We used 8,748,392 event logs collected in January 2020 for training, and 9,142,782 event logs collected in February 2020 for testing. Because this data is provided by the enterprise, the file name and file path are encrypted. Therefore, other fields such as file name and file path cannot be used. **Table 5** shows the dataset configuration used.

Table 4. Original dataset collected from a real commercial endpoint environment

_index	_id	ProcName	FileName	EventType	EventSubType	EventTime	IP	RemoteIP	LocalIP	ProcPath	FilePath	FileType
endpoi	01D25	motty.exe	78121e9be	process	ChildProcessCreate	1.57846E+12	172.29.*.*			C:₩Users₩ja	44f50c742	PE
endpoi	1B2A9	NaverAgent.exe		network	NetworkConnect	1.57792E+12	172.29.*.*	210.89.*.*	172.29.*.*	C:₩Program		
endpoi	1B2A9	NaverAgent.exe		network	NetworkConnect	1.57792E+12	172.29.*.*	184.28.*.*	172.29.*.*	C:₩Program		
endpoi	2B504	Code.exe		network	NetworkConnect	1.57897E+12	172.29.*.*	65.55.*.*	172.29.*.*	C:₩Users₩ja		
endpoi	2B504	Code.exe		network	NetworkConnect	1.57897E+12	172.29.*.*	111.221.*.*	172.29.*.*	C:₩Users₩ja		
endpoi	37F1B	Code.exe	b2fd9c9e2f	process	ChildProcessCreate	1.57896E+12	172.29.*.*			C:₩Users₩ja	8e1080c90	PE
endpoi	37F1B	Code.exe	b2fd9c9e2f	process	ChildProcessCreate	1.57896E+12	172.29.*.*			C:₩Users₩ja	8e1080c90	PE
endpoi	3AB9C	Hwp.exe	f88155d117	file	DocOpen	1.57792E+12	172.29.*.*			C:₩Program	b14e4ed0f	DOC
endpoi	3BFEEF	motty.exe		network	NetworkConnect	1.57846E+12	172.29.*.*	172.29.*.*	172.29.*.*	C:₩Users₩ja		
endpoi	3C393	Code.exe	b2fd9c9e2f	process	ChildProcessCreate	1.57897E+12	172.29.*.*			C:₩Users₩ja	8e1080c90	PE
endpoi	598BA	motty.exe	78121e9be	process	ChildProcessCreate	1.57846E+12	172.29.*.*			C:₩Users₩ja	44f50c742	PE

Table 5. Dataset configuration

Data	Collection	Collection Period	Event Count
Training	G*	January 1, 2020 – January 31, 2020	8,748,392
Test		February 1, 2020 – February 29, 2020	9,142,782

4.2 Unknown Attack Detection Results

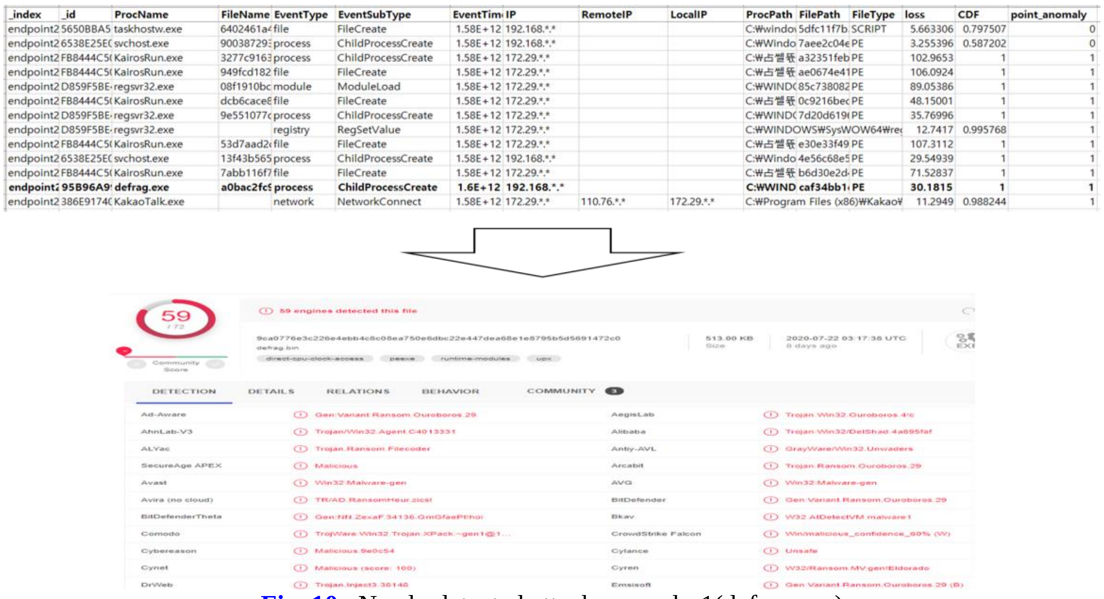
We proposed a model to detect unknown attacks based on Autoencoder and 1D CNN. As a result of the experiment, all of the proposed models are converted to the number of 0 and 1 by applying the CDF from the loss value for the test data. We set the threshold to 0.98, and if it is 0.98 or more, it is judged as a suspicious attack. However, since the detected suspected attack behavior cannot be judged as malicious, it is verified through *VirusTotal* to determine if it is actually malicious. **Table 6** shows the proven unknown attacks. It shows the list of malicious processes verified through VirusTotal. For each verified malicious process, it shows the maximum anomaly score generated from the proposed model. Also, it indicates the number of engines detected as malicious by VirusTotal's anti-virus engine. AVclass stands for detection name. AVClass refers to the malware family, the name of the detected malware. Accordingly, it shows the malicious behavior to be attempted. As a result, you can use the event log to detect an unknown attack that does not detect an existing initial attack as an antivirus.

Table 6. Unknown attack detection results in two proposed model

Process Name	**Maximum Anomaly Score**		**Virus Total**	**AVclass**	**Malicious Behavior**
	1D CNN	**Auto-Encoder**			
RPRTSetup.exe	1.0000	1.0000	12/71	buzus	Attempt to steal PC information and modify the registry using Keylogger function
controller.exe	1.0000	0.9999	10/72	high	A type of malicious process that steals user information
svchost.exe	1.0000	0.9988	27/72	swrort	For easy malicious behavior, registry manipulation related to services and servers and drop of malicious files
WinClientService.exe	1.0000	0.9998	18/72	kraddare	Automatically launches the program and displays an advertising window without use consent
NaverAgent.exe	1.0000	0.9987	51/70	zegost	DLL injection into normal svchost.exe file works, and malicious DLL file performs eavesdropping and carious malicious function
defrag.exe	1.0000	-	59/72	delshad	Read data from binary image and create own copy and system file
dasHost.exe	1.0000	-	45/72	johnnie	Task manager list may show unwanted processes, Collects information to fingerprint the system
reg.exe	1.0000	-	41/67	hupigon	As a kind of backdoor, it creates a self-replicated copy in a specific folder and modifies the registry. Also, it receives encrypted commands from specific sites and performs them
regsvr32.exe	1.0000	-	20/72	agen	Trojan horse that collects user's keyboard input information
csrss.exe	1.0000	-	15/72	midie	create and set registry keys with a series of long bytes to store malware configuration
dumpchk.exe	1.0000	-	9/72	driverpack	PUP-like malware that can install unwanted programs and steal system and user information
wlanext.exe	0.9999	-	51/72	tiggre	Malware that can be remotely controlled by and attacker's command
sh.exe	0.9999	-	45/70	razy	Malicious code that installs malicious extensions in web browsers to provide phishing links to infected PCs or to perform mining activites
SetupImgBurn_2.5.8.0.exe	0.9998	0.9996	38/71	installcore	PUP-like malware that installs unwanted programs and communicates with malicious networks
ose.exe	0.9998	-	44/72	mimikatz	Tools for stealing and decrypting information related to various accounts on Windows

keygen.exe	0.9998	-	35/68	high	A type of malicious process that steals user information
NGM.exe	0.9993	-	25/65	razy	Malicious code that installs malicious extensions in web browsers to provide phishing links to infected PCs or to perform mining activities
Nox_unload.exe	0.9993	-	19/72	fusioncore	As kind of Adware.Malwarebytes' detection name for a large family of adware bundlers targeting Windows systems. Users of affected systems may find that they have installed more than they expected.
HncUpdate.exe	0.9992	-	36/70	agen	Trojan horse that collects user's keyboard input information
install2.exe	0.9946	-	45/72	agen	Trojan horse that collects user's keyboard input information
DropboxOEM.exe	-	0.9904	53/69	fareit	Trojan horse that performs malicious behavior by dropping a malicious file similar to the normal file name
DeviceManager.exe	0.9901	0.9993	51/65	phorpiex	Distribute spam campaigns, including many malware and massive sex torsion email campaigns
FileZilla_3.46.3_win64 _sponsored-setup.exe	-	0.9885	16/71	fusioncore	As kind of Adware.Malwarebytes' detection name for a large family of adware bundlers targeting Windows systems. Users of affected systems may find that they have installed more than they expected.
taskhost.exe	-	0.9833	49/72	autoit	A dropper that collects user information by creating a malicious file disguised as a normal name
scrt726-x64.exe	-	0.9938	46/72	installcore	PUP-like malware that installs unwanted programs and communicates with malicious networks
ReaderUpdater.exe	0.9804	-	62/69	neshta	It is a kind of virus and spreads through its own propagation function. Encrypt the normal PE file code in the Windows system and insert the virus code. It also creates malicious files in the Windows system folder.

We detected and verified an unknown attack. Referring to **Table 6**, which shows the results verified by the attack, it describes the process that VirusTotal's antivirus was detected as an attack on two processes, the most detected number. We test the generated feature for each event log in a sequence format as much as the window size. For example, if the window size is 10, the top 10 including the corresponding event log is input, and the event log that occurs next is trained as the output. **Fig. 10** shows the defrag.exe detection process and verification screen. In the above **Fig. 10**, to detect defrag.exe, including the defrage.exe event log, the top 10 KairoRun.exe event logs are input, and the next KakaoTalk.exe event log is trained as output. If the training model again tests the defrag.exe event log, a field called Loss is created and a Loss value of 30.1815. It is used statistically and the anomaly score is calculated by applying the CDF. At this time, the anomaly score is 1. This study detects this as anomalous data because a specific threshold is set to 0.98. Therefore, when defrag.exe, which has detected abnormal data, is verified through a total of viruses, it is created as shown in the figure below in **Fig. 10**. It detected 59 anti-viruses out of 72. AVclass gives the name of the malware family, which is delshad, the most common word for detection in 59 antiviruses. **Fig. 11** goes through the detection and verification process in the same way as in **Fig. 10**.



_index	_id	ProcName	FileName	EventType	EventSubType	EventTim	IP	RemoteIP	LocalIP	ProcPath	FilePath	FileType	loss	CDF	point_anomaly
endpoint2	5650BBA5	taskhostw.exe	6402461a4	file	FileCreate	1.58E+12	192.168.*.*			C:₩windo	5dfc11f7b	SCRIPT	5.663306	0.797507	0
endpoint2	6538E25E(	svchost.exe	90038729	process	ChildProcessCreate	1.58E+12	192.168.*.*			C:₩Windo	7aee2c04e	PE	3.255396	0.587202	0
endpoint2	FB8444C5(	KairosRun.exe	3277c9163	process	ChildProcessCreate	1.58E+12	172.29.*.*			C:₩占썥뚯	a32351feb	PE	102.9653	1	1
endpoint2	FB8444C5(	KairosRun.exe	949fcd182	file	FileCreate	1.58E+12	172.29.*.*			C:₩占썥뚯	ae0674e41	PE	106.0924	1	1
endpoint2	D859F5BE	regsvr32.exe	08f1910bc	module	ModuleLoad	1.58E+12	172.29.*.*			C:₩WIND(	85c738082	PE	89.05386	1	1
endpoint2	FB8444C5(	KairosRun.exe	dcb6cace8	file	FileCreate	1.58E+12	172.29.*.*			C:₩占썥뚯	0c9216bec	PE	48.15001	1	1
endpoint2	D859F5BE	regsvr32.exe	9e551077c	process	ChildProcessCreate	1.58E+12	172.29.*.*			C:₩WIND(	7d20d619(	PE	35.76996	1	1
endpoint2	D859F5BE	regsvr32.exe		registry	RegSetValue	1.58E+12	172.29.*.*			C:₩WINDOWS₩SysWOW64₩re(			12.7417	0.995768	1
endpoint2	FB8444C5(	KairosRun.exe	53d7aad2(	file	FileCreate	1.58E+12	172.29.*.*			C:₩占썥뚯	e30e33f49	PE	107.3112	1	1
endpoint2	6538E25E(	svchost.exe	13f43b565	process	ChildProcessCreate	1.58E+12	192.168.*.*			C:₩Windo	4e56c68e5	PE	29.54939	1	1
endpoint2	FB8444C5(	KairosRun.exe	7abb116f7	file	FileCreate	1.58E+12	172.29.*.*			C:₩占썥뚯	b6d30e2d	PE	71.52837	1	1
endpoint2	**95B96A9**	**defrag.exe**	**a0bac2fc**	**process**	**ChildProcessCreate**	**1.6E+12**	**192.168.*.***			**C:₩WIND**	**caf34bb1**	**PE**	**30.1815**	**1**	**1**
endpoint2	386E9174(	KakaoTalk.exe		network	NetworkConnect	1.58E+12	172.29.*.*	110.76.*.*	172.29.*.*	C:₩Program Files (x86)₩Kakao₩			11.2949	0.988244	1

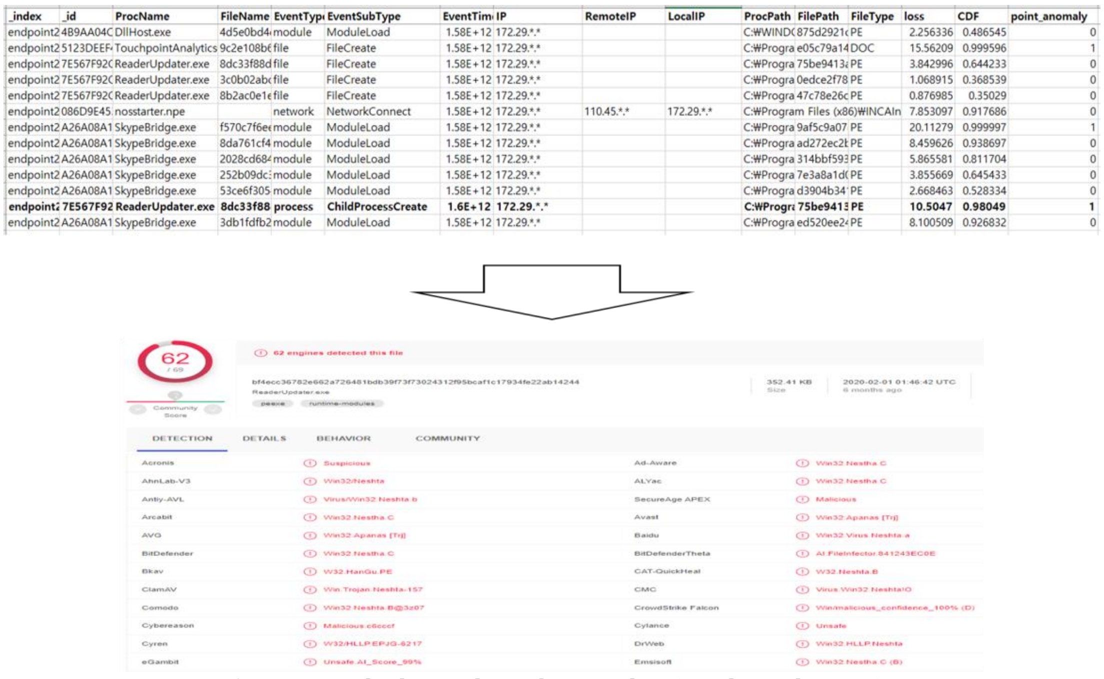
59 engines detected this file

9ca0776e3c226e4ebb4c8c08ea750e6dbc22e447dea68e1e8795b5d5691472c0
defrag.bin
direct-cpu-clock-access peexe runtime-modules upx

513.00 KB Size
2020-07-22 03:17:38 UTC 8 days ago

Ad-Aware	Gen:Variant.Ransom.Ouroboros.29	AegisLab	Trojan.Win32.Ouroboros.4!c
AhnLab-V3	Trojan/Win32.Agent.C4013331	Alibaba	Trojan:Win32/DelShad.4a695faf
ALYac	Trojan.Ransom.Filecoder	Antiy-AVL	GrayWare/Win32.Unwaders
SecureAge APEX	Malicious	Arcabit	Trojan.Ransom.Ouroboros.29
Avast	Win32:Malware-gen	AVG	Win32:Malware-gen
Avira (no cloud)	TR/AD.RansomHeur.zicst	BitDefender	Gen:Variant.Ransom.Ouroboros.29
BitDefenderTheta	Gen:NN.ZexaF.34136.GmGfaePf!hoi	Bkav	W32.AIDetectVM.malware1
Comodo	TrojWare.Win32.Trojan.XPack.~gen1@1...	CrowdStrike Falcon	Win/malicious_confidence_60% (W)
Cybereason	Malicious.9e0c54	Cylance	Unsafe
Cynet	Malicious (score: 100)	Cyren	W32/Ransom.MV.gen!Eldorado
DrWeb	Trojan.Inject3.38146	Emsisoft	Gen:Variant.Ransom.Ouroboros.29 (B)

Fig. 10. Newly detected attack example-1(defrag.exe)



_index	_id	ProcName	FileName	EventTyp	EventSubType	EventTim	IP	RemoteIP	LocalIP	ProcPath	FilePath	FileType	loss	CDF	point_anomaly
endpoint2	4B9AA04C	DllHost.exe	4d5e0bd4	module	ModuleLoad	1.58E+12	172.29.*.*			C:₩WIND(	875d2921(	PE	2.256336	0.486545	0
endpoint2	5123DEEF	TouchpointAnalytics	9c2e108b6	file	FileCreate	1.58E+12	172.29.*.*			C:₩Progra	e05c79a14	DOC	15.56209	0.999596	1
endpoint2	7E567F92(	ReaderUpdater.exe	8dc33f88d	file	FileCreate	1.58E+12	172.29.*.*			C:₩Progra	75be9413	PE	3.842996	0.644233	0
endpoint2	7E567F92(	ReaderUpdater.exe	3c0b02ab(	file	FileCreate	1.58E+12	172.29.*.*			C:₩Progra	0edce2f78	PE	1.068915	0.368539	0
endpoint2	7E567F92(	ReaderUpdater.exe	8b2ac0e1	file	FileCreate	1.58E+12	172.29.*.*			C:₩Progra	47c78e26c	PE	0.876985	0.35029	0
endpoint2	086D9E45	nosstarter.npe		network	NetworkConnect	1.58E+12	172.29.*.*	110.45.*.*	172.29.*.*	C:₩Program Files (x86)₩INCAIn			7.853097	0.917686	0
endpoint2	A26A08A1	SkypeBridge.exe	f570c7f6e	module	ModuleLoad	1.58E+12	172.29.*.*			C:₩Progra	9af5c9a07	PE	20.11279	0.999997	1
endpoint2	A26A08A1	SkypeBridge.exe	8da761cf4	module	ModuleLoad	1.58E+12	172.29.*.*			C:₩Progra	ad272ec2	PE	8.459626	0.938697	0
endpoint2	A26A08A1	SkypeBridge.exe	2028cd684	module	ModuleLoad	1.58E+12	172.29.*.*			C:₩Progra	314bbf593	PE	5.865581	0.811704	0
endpoint2	A26A08A1	SkypeBridge.exe	252b09dc	module	ModuleLoad	1.58E+12	172.29.*.*			C:₩Progra	7e3a8a1d(	PE	3.855669	0.645433	0
endpoint2	A26A08A1	SkypeBridge.exe	53ce6f305	module	ModuleLoad	1.58E+12	172.29.*.*			C:₩Progra	d3904b34	PE	2.668463	0.528334	0
endpoint2	**7E567F92**	**ReaderUpdater.exe**	**8dc33f88**	**process**	**ChildProcessCreate**	**1.6E+12**	**172.29.*.***			**C:₩Progra**	**75be9413**	**PE**	**10.5047**	**0.98049**	**1**
endpoint2	A26A08A1	SkypeBridge.exe	3db1fdfb2	module	ModuleLoad	1.58E+12	172.29.*.*			C:₩Progra	ed520ee2	PE	8.100509	0.926832	0

62 engines detected this file

bf4ecc36782e662a726481bdb39f73f73024312f95bcaf1c17934fe22ab14244
ReaderUpdater.exe
peexe runtime-modules

352.41 KB Size
2020-02-01 01:46:42 UTC 6 months ago

Acronis	Suspicious	Ad-Aware	Win32.Nestha.C
AhnLab-V3	Win32/Neshta	ALYac	Win32.Nestha.C
Antiy-AVL	Virus/Win32.Neshta.b	SecureAge APEX	Malicious
Arcabit	Win32.Nestha.C	Avast	Win32:Apanas [Trj]
AVG	Win32:Apanas [Trj]	Baidu	Win32.Virus.Neshta.a
BitDefender	Win32.Nestha.C	BitDefenderTheta	AI:FileInfector.841243EC0E
Bkav	W32.HanGu.PE	CAT-QuickHeal	W32.Neshta.B
ClamAV	Win.Trojan.Neshta-157	CMC	Virus.Win32.Neshta!O
Comodo	Win32.Neshta.B@3z07	CrowdStrike Falcon	Win/malicious_confidence_100% (D)
Cybereason	Malicious.c6cccf	Cylance	Unsafe
Cyren	W32/HLLP.EPJG-6217	DrWeb	Win32.HLLP.Neshta
eGambit	Unsafe.AI_Score_99%	Emsisoft	Win32.Nestha.C (B)

Fig. 11. Newly detected attack example-2(ReaderUpdate.exe)

4.3 Operation policy

The proposed model operating policy can set the learning period in real time. The anomaly detection method of the existing study has the disadvantage of having to label each log from an operational point of view, because security administrators analyzed and detected using labels. In this study, you can learn without a label as an unsupervised learning method. This has the advantage of analyze the anomaly and not having to label each log. Normal processes validated at a particular time can be retrained after a certain time. In other words, the performance of the anomaly detection model is improved by using the results of the model trained during a specific training period to provide normal behavioral information to the next training model. Fig. 12 shows the real-time operational policy flow.

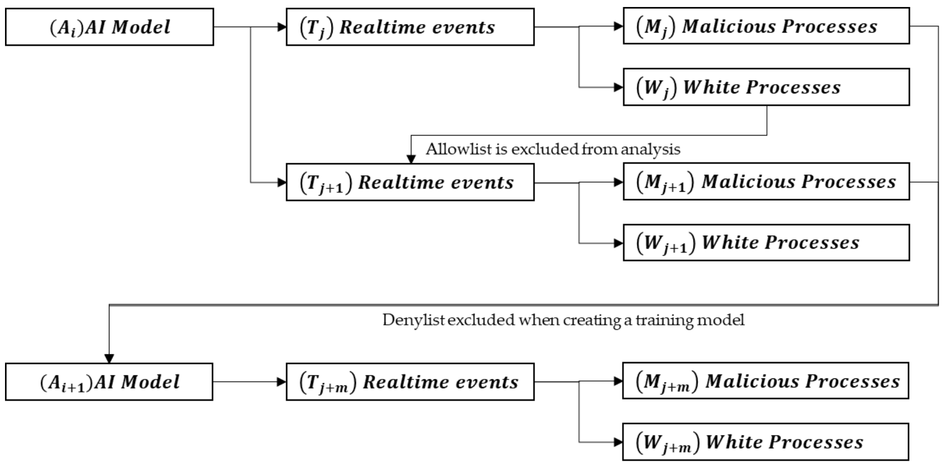

(A_i)AI Model
(T_j) Realtime events
(M_j) Malicious Processes
(W_j) White Processes
Allowlist is excluded from analysis
(T_{j+1}) Realtime events
(M_{j+1}) Malicious Processes
(W_{j+1}) White Processes
Denylist excluded when creating a training model
(A_{i+1})AI Model
(T_{j+m}) Realtime events
(M_{j+m}) Malicious Processes
(W_{j+m}) White Processes

Fig. 12. Real-time operation policy flow

In addition, the proposed technology based on the existing motion log supports stable anomaly detection. Fig. 13 is a flow chart showing that the results of the analysis can be efficiently run with respect to legacy systems such as allowlist, denylist and pattern-based policies. The proposed model is the process of detecting anomaly in the endpoint. Anomaly detection results are displayed for each event, and each event is checked for abnormal behavior.

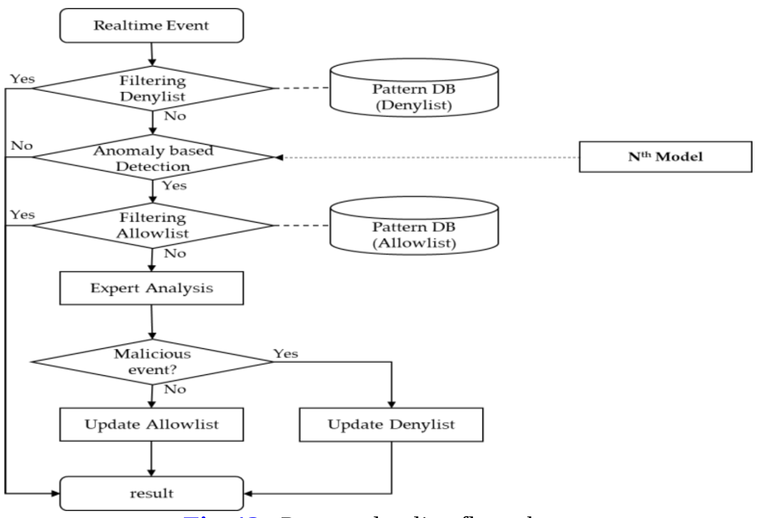


Fig. 13. Proposed policy flow chart

If the event is not on the denylist, **Fig. 13** proceeds to detect anomaly. If it turns out to be abnormal, check the allowlist database. Events not in the allowlist database are manually analyzed by experts. If you check for malicious behavior, you can update the denylist database and, if it is a normal event, the allowlist database. Therefore, the allowlist or denylist policy is expected to work effectively as many events are fired on the endpoint.

5. Conclusion

So far, we have focused on network security. We believed that endpoint security was sufficient for vaccines and patches. Unfortunately, security incidents are not diminishing and the damage is growing day by day. Therefore, as the number of advanced threats increases, we are rapidly moving from past file-based prediction and defense to user behavior-based detection and response. Recently, due to the advancement of 5G/IoT, various devices constitute an endpoint environment, collecting and analyzing all the actions on the endpoint, and the EDR solution for responding to the threat has been in the spotlight. EDR currently collects various events at the endpoint, but detecting suspicious events is a difficult reality. In this paper, anti-virus does not respond to the initial attack, so deep learning AutoEncoder and 1D-CNN are used to detect unknown attacks. The detected attacks are verified using VirusTotal. In addition, various policies can be applied for stable and effective operation in the endpoint environment. As an example of model operation, we also proposed the operation policy of legacy systems using allowlist and denylist. This can greatly improve performance by minimizing false positives. The proposed model can be applied to various environments such as IoT, ICS, and cloud as well as endpoints. In the future, we plan to verify data and improve models in various environments as well as endpoints to ensure continuous operation and practicality.

References

[1] CISCO, Cisco Annual Internet Report (2018-2023) White Paper. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internetreport/white-paper-c11-741490.html

[2] H. H. Pajouh, R. Javidan, R. Khayami, A. Dehghantanha, and K. R. Choo, "A Two-Layer Dimension Reduction and Two-Tier Classification Model for Anomaly-Based Intrusion Detection in IoT Backbone Networks," *IEEE Transactions on Emerging Topics in Computing*, vol. 7, no. 2, pp. 314-323, 2019. Article (CrossRef Link)

[3] T. Li, Y. Jiang, C. Zeng, B. Xia, Z. Liu, W. Zhou, X. Zhu, W. Wang, L. Zhang, J. Wu, L. Xue, and D. Bao, “FLAP: An end-to-end event log analysis platform for system management,” in *Proc. of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1547-1556, 2017. Article (CrossRef Link)

[4] M. Zaman, T. Siddiqui, M. R. Amin, and M. S. Hossain, "Malware detection in Android by network traffic analysis," in *Proc. of 2015 International Conference on Networking Systems and Security (NSysS)*, pp. 1-5, 2015. Article (CrossRef Link)

[5] T. Isohara, K. Takemori, and A. Kubota, "Kernel-based Behavior Analysis for Android Malware Detection," in *Proc. of 2011 Seventh International Conference on Computational Intelligence and Security*, pp. 1011-1015, 2011. Article (CrossRef Link)

[6] J. Sun, T. Jeng, C. Chen, H. Huang, and K. Chou, "MD-Miner: Behavior-Based Tracking of Network Traffic for Malware-Control Domain Detection," in *Proc. of 2017 IEEE Third International Conference on Big Data Computing Service and Applications*, pp. 96-105, 2017. Article (CrossRef Link)

[7] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, “Long Short Term Memory Networks for Anomaly Detection in Time Series,” in *Proc. of 23rd European Symposium on Artificial Neural Networks,* p. 89, 2015. Article (CrossRef Link)

[8] M, Toledano, I. Cohen, Y. Ben-Simhon, and I. Tadeski, “Real-time anomaly detection system for time series at scale,” *Proceedings of Machine Learning Research,* vol. 71, pp. 56-65, 2017. Article (CrossRef Link)

[9] S. He, J. Zhu, P. He, and M. R. Lyu, “Experience report: System log analysis for anomaly detection,” in *Proc. of the 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 207-218, 2016. Article (CrossRef Link)

[10] R. J. Gutierrez, B. C. Boehmke, K. W. Bauer, C. M Saie, and T. J Bihl, “anomalyDetection: Implementation of augmented network log anomaly detection procedures,” *The R Journal*, vol. 9, no. 2, pp. 354-365, 2017. Article (CrossRef Link)

[11] S. Garg, K. Kaur, N. Kumar, G. Kaddoum, A. Y. Zomaya, and R. Ranjan, "A Hybrid Deep Learning-Based Model for Anomaly Detection in Cloud Datacenter Networks*," IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 924-935, 2019. Article (CrossRef Link)

[12] M. Ahmed, A. N. Mahmood, and J. Hu, “A survey of network anomaly detection techniques,” *Journal of Network and Computer Applications*, vol. 60, pp. 19-31, 2016. Article (CrossRef Link)

[13] B. I. Kwak, M. R. Han, A. R. Kang, and H. K. Kim, “A study on detection methodology of threat on cars from the viewpoint of IoT,” *Journal of the Korea Institute of Information Security & Cryptology*, vol. 25, no. 2, pp. 441-421, 2015. Article (CrossRef Link)

[14] K. Kim, “Status of abnormal sign detection technology in smart manufacturing environment," *Review of Korea Institute of Information Security and Cryptology*, vol. 29, no. 2, pp. 36-47, 2019. Article (CrossRef Link)

[15] I. Alrashdi, A. Alqazzaz, E. Aloufi, R. Alharthi, M. Zohdy, and H. Ming, "AD-IoT: Anomaly Detection of IoT Cyberattacks in Smart City Using Machine Learning," in *Proc. of 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0305-0310, 2019. Article (CrossRef Link)

[16] M. Kravchik and A. Shabtai, “Detecting cyberattacks in industrial control systems using convolutional neural networks,” in *Proc. of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*, pp. 72-83, 2018. Article (CrossRef Link)

[17] S. Kim, C. Hwang, and T. Lee, "Anomaly Based Unknown Intrusion Detection in Endpoint Environments," *Electronics*, vol. 9, no. 6, 2020. Article (CrossRef Link)

[18] T. Andrade, J. Gama, and P. Ribeiro, “W. Sousa and A. Carvalho, Anomaly Detection in Sequential Data: Principles and Case Studies,” *Widly Online Library*, 2019. Article (CrossRef Link)

[19] L. Bontemps, V. L. Cao, J. McDermott, and L. K. L. Na, “Collective Anomaly Detection Based on Long Short-Term Memory Recurrent Neural Networks,” *Future Data and Security Engineering*, vol. 10018, pp 141-152, 2016. Article (CrossRef Link)

[20] M. Ahmed and A. N. Mahmood, "Network traffic analysis based on collective anomaly detection," in *Proc. of 2014 9th IEEE Conference on Industrial Electronics and Applications*, pp. 1141-1146, 2014. Article (CrossRef Link)

[21] VirusTotal. [Online]. Available: https://www.virustotal.com/gui/home/search

[22] J. Chen, S. Sathe, C. Aggarwal, and D. Turaga, “Outlier detection with autoencoder ensembles,” in *Proc. of the 2017 SIAM International Conference on Data Mining*, pp. 90-98, April 2017. Article (CrossRef Link)

[23] C. Zhou and R. C. Paffenroth, “Anomaly detection with robust deep autoencoders,” in *Proc. of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 665–674, 2017. Article (CrossRef Link)

Chan Woong Hwang is studying for a master`s degree in information security at Hoseo University. He received a bachelor`s degree in information security from Hoseo University. His research interests include malware analysis, machine learning and anomaly detection.

Do Yeon Kim is studying for a master`s degree in information security at Hoseo University. She received a bachelor`s degree in information security from Konyang University. Her research is interest in deep learning, malware analysis and malware detection.

Tae Jin Lee graduated from Postech Computer Engineering Department in 2003 and graduated from Yonsei University in 2008 and Ajou University in 2017. He worked at Korea Internet Security Agency from 2003 to 2017 and he has been worked in Hoseo University since 2017. His research area are intrusion tolerance technology, VoIP/Wibro security, malware distribution detection/analysis, email security, cyber black box, and malware profiling and mobile payment fraud detection. His current main interests are artificial intelligence, malicious code analysis, intrusion detection.