Network Anomaly Detection Based on NetFlow Time Series Data in a Healthcare Network Environment

Heng-Shuen Chen[1], Edgar Hsieh[2], Aaron Solomon[2*]

[1]Puli Christian Hospital, No. 1, Tieshan Road, Puli Township, Nantou County 545401, Taiwan
[2]Computer Science & Information Engineering Department, National Chi-Nan University, No. 1, Daxue Rd, Puli Township, Nantou County 545301, Taiwan
[*]*Corresponding Author: solomon@ncnu.edu.tw*

Abstract

In recent years, cyberattacks have become more frequent, and different attack methods have also evolved quickly. Many previous studies proposed various machine learning models for detecting abnormal network behaviors, but seldom apply time parameters to the design of feature data. In this study, the packet flow data are converted into time-series data of feature vectors, and the clustering algorithm is adjusted so that it can be clustered according to the time series similarity. The multi-factor clustering algorithm is used to stabilize the clustering results and define the correlation of data, and finally find the nodes with abnormal behavior.

Keywords: dynamic time warping (DTW), evidence accumulation clustering (EAC), revised fuzzy C-means (RFCM), unsupervised-learning.

1. Introduction

The advancement of technology has made people's lives more convenient, but it has also provided hackers with more convenient tools. For example, the global ransomware incident [1] that broke out in 2017 and kidnapped victims' data utilized encryption algorithms we rely on to protect data. The incident caused hundreds of thousands of devices to be compromised worldwide, and many services were shut down. After the incident, information security issues related to ransomware began to receive much attention. In addition to ransomware, other cybersecurity issues, such as phishing letters, phishing webpages, and files containing malicious code have also become the focus of information protection.

Just as technology advances, so does malware. In addition to the changes in attack methods, a different ransomware model has also been developed. Keepnet Labs analyzed various types of ransomware and compiled a list based on the threat to the organization [2], from which we can observe that the list has new kinds of ransomware different from the previous. Among which ransomware such as Nemty [3] is no longer a tool that encrypts and extorts data. Nemty is more like a ransomware service, which can be said to be an application of a cloud computing service model [4]. This form of ransomware is also known as Ransomware as a Service (RaaS), which will make the industry chain of ransomware well packaged, enabling intended people to use it with lower technical thresholds and cheaper costs to run a ransomware business. Sophos believes that RaaS will be a trend of ransomware in the future in the "2021 Ransomware Trend" article [5]. CrowdStrike also mentioned in the "2021 Cyber Threat" report that electronic crime programs are gradually transforming to provide RaaS services and concentrate on high-value targets [6] such as healthcare organizations.

The operation of RaaS is mainly responsible for the development of customized ransomware by the ransomware developer. The developer will license the software to his members and charge the members for use or take a share of the gouging revenue. Such a business model is similar to Software as a Service (SaaS) [4]. Members who purchase ransomware can set up the hosting server according to their preferences, distribute it, and extort the victims [7] (Figure 1). Technically RaaS has not changed much in attacking methods, but this business model will cause many problems: First, the difficulty of carrying out a ransomware business is significantly reduced. In the past, the ransomware developers handled all the technologies needed to develop ransomware. Now, members do not need to have superb skills. They only need to rent a server online to start their own ransomware business; Second, people who launched the ransomware attacks are members who purchased the ransomware service, and only these members are tracked and arrested, but the real developers can still freely provide services to other members to start their ransomware business.

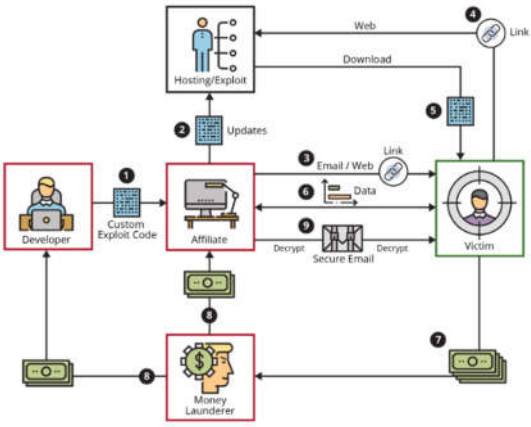

Figure 1. Schematic diagram of RaaS [7]

The botnets that ransomware relies on to hide its tracks also play an essential role in the process of extortion. Trickbot, which is dominated by servers of parasitic financial and critical administrative units, is one of the largest botnets in the world. At the end of 2020, when the US presidential election was imminent, Microsoft received an order from a federal court to exclude the largest global botnet – Trickbot. That move did significantly reduce the ability of hackers to attack vote-return systems or plant ransomware in them. According to the observation of Intel 471, Microsoft released system updates, modified Trickbot's configuration file, and changed the IP address of the Controller to localhost, so that the Trickbot node could no longer communicate with the domain controller and spread the virus [8], but this move only successfully paralyzed Trickbot for a few days [9].

Trickbot's main attack targets are servers. In addition to stealing data on infected machines, it also opens backdoors for malicious organizations to log in at will. The machine infected by Trickbot will spread the Trickbot loader to the target machine through the SMB vulnerability [10-15], and the machine receiving the loader will download a Trickbot client disguised as an image through HTTP after executing the program. In 2020, Trickbot was re-engineered to keep its code purely in memory without writing to disks. This makes detecting Trickbot through virus signatures more challenging. Since the server seldom shut down, this design allows Trickbot to persist in memory for a long time without being detected [16] (see Figure 2).

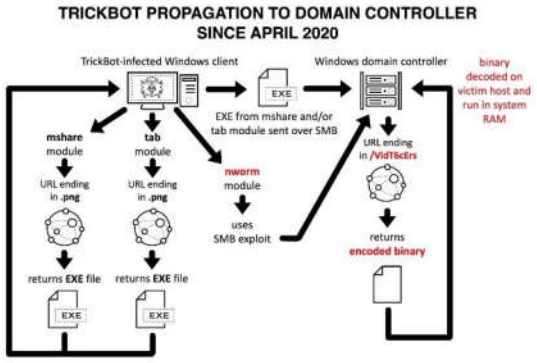

Figure 2. Schematic diagram of the propagation process of Trickbot [16]

Network anomaly detection is one of the methods that can effectively deal with these threats hidden in the network. The primary way to defend against improper network behavior is to block problematic network packets with information security functions on network devices. It is achieved to prevent the spread of viruses or network attacks. The network anomaly detection methods can be roughly divided into three categories: Signature-based, Supervised-learning-based, and Unsupervised-learning-based.

Signature-based detection systems can efficiently detect known abnormal behaviors through pre-written programs or rules. When new abnormal behaviors are discovered, information security experts will analyze the characteristics of abnormal behaviors and write programs or rules to deal with the abnormal behaviors. The detection system can detect these abnormal behaviors. This type of detection is potent in detecting anomalous behavior and, at the same time, easy to understand because it can directly correlate anomalous behaviors to features. However, this detection system cannot deal with unseen abnormal behaviors and requires analysis and research by information security experts to establish new abnormal behavior characteristics.

Supervised-learning-based is to use the labeled packet flow data to train a model that can distinguish the normal network packet flow. When the packet flow characteristics differ from the model's recognition, it is abnormal. Such anomaly detection models can detect new anomalous behaviors and unseen aggressive behaviors by comparing them with the model's cognitive of typical behavioral characteristics. However, supervised learning takes time to train and relies on packet flow data with no anomalous behavior. In practice, marking the packet flow data as "normal" is arduous, and even information security experts have a hard time guaranteeing that there is no abnormality hidden in the packet flow data. Furthermore, maintaining such a detection model to remain accurate and continuously updated is not an easy task, especially as new services and applications keep on emerging.

Unsupervised-learning-based compares packet flows with different behavioral characteristics by grouping the packet flow data, thereby finding abnormal behaviors. This anomaly detection method does not need experts to analyze packet flows to find the characteristic attributes of anomalous packets or provide data for model training, nor does it require long-term model training. We believe that with the rapid development of network technology, the methods of attack will not remain the same. Therefore, the detection system that needs to rely on the analysis of information security experts is likely to have a security gap due to the time difference. On the contrary, unsupervised learning models can automatically distinguish the characteristics of abnormal behavior, which will be the future trend of network security research. In this paper, our proposed method can cluster time series of packet flow features and compute correlations to find anomalous behaviors.

2. Related Works

[17] proposed a two-layer clustering structure, where each data sample is distributed in its subgroup, and the distributions of these subgroups are integrated by a Gaussian Mixture Model (GMM) [18]. For the streaming sample data, the paper proposes a structure *Rag Bag* to collect new samples. Rag Bag is responsible for clustering, and finally decide to fuse, form a new subgroup, or discard according to the center distance of

the subgroups of the old and new data. In detecting abnormal behavior, whether the new sample is abnormal will be determined by the similarity between the new sample and the existing cluster.

[19] cut out the data of a time window from the sample data of the stream with a length of time sufficient to form a feature, and organize the data of each period into a packet flow format. After that, feature vectors are extracted from the packet flow data of each period, and these vectors are integrated into a matrix to form a vector space of data features. For the model to be updated instantly, the detection and update of the model must use a smaller time interval than the time window. According to the research of [20], it takes at least 15 seconds of data to be sufficient to form the characteristics of the data. Therefore, this paper applies the concept of the time-sliding window, adding one micro-slot each time and discarding the oldest micro-slot. In order to further reduce the amount of calculation, the paper uses grid clustering to divide the feature vector space into disjoint blocks and group them according to the data density of the blocks. It is abnormal when the data features appear in the blocks without groups.

[21, 22] divide the feature set of sample data into subsets of different sizes and then use these subsets for clustering. After that, they use the clustering result to obtain the correlation among the packet flow features and use the correlation's strength to find abnormal behavior. Combining different clustering results of the same data set can reduce the situation that a single cluster triggers a false alarm. [22] also proposed the concept of abnormal behavior ranking. In abnormal behavior, flooding-based attacks and other attack methods that suddenly boost significant traffic quickly affect the operation of the entire network and need to be blocked as soon as possible. By calculating the proportion of packets and data, the risk factor obtained by the proportion of traffic can allow such abnormal behavior to be dealt with earlier.

3. Dataset

Past research has repeatedly addressed the challenge of benchmarking network packet testing data and provided researchers with reliable datasets to validate anomaly detection models. For example, the DARPA Intrusion Detection Evaluation Program [23] is dedicated to providing a stream of marked network packets to evaluate intrusion detection systems. DARPA has been widely used in related research, mainly through the 1999 KDD Cup Network Packet Dataset (KDD'99), and has provided strong support for researchers in the field of Intrusion Detection systems (IDS). The DARPA Intrusion Detection and Evaluation Program focuses on the evaluation of IDS, so the marked LAN packet stream is provided with the packet payload and the complete packet stream. However, KDD'99 was questioned on "the extent to which the experimental data are suitable for real-world tasks" and "the influence of the simulated environment architecture on the experimental data" [24]. Additionally, KDD'99's data is built with packet data from 1998, so it does not include recent applications or unusual traffic. Therefore, this dataset must be used with extreme caution as it does not represent real traffic [25] and lacks anomalous packet flows which are common in recent years.

Owezarski [26] presented a dataset in 2010 containing real backbone traffic with precisely labeled anomalies. This study collects network packets at different nodes in the RENATER network. RENATER is France's national research and education network, and the network environment is defined as non-anomalous. The researchers added two anomalous packet flows to this clean network traffic, Flash Crowd and DDoS attacks. The experiments in this study include anomalies of different strengths in different fields, so it is easier to analyze the susceptibility of anomaly detection models of DDoS and Flash Crowd attacks. However, this dataset contains only a few anomalies and is not representative of the anomalies found in natural network environments.

In addition to data sets that provide raw packet data, some studies provide packet flow data in NetFlow format. Sarhan et al. [27] designed a data set with essential features and provided four measurement benchmarks for machine learning-based network anomaly detection systems. In real-world scenarios, NetFlow features are relatively easy to derive from network traffic compared to the complex features used in raw packet data sets, as they are usually extracted from packet headers. In this study, the corresponding NetFlow data sets were generated from the four original data sets, respectively. The original data sets were compared with NetFlow data sets to have similar classification results on the four measurement benchmarks. However, the dataset does not contain timestamps, making it impossible to extract time series data from this dataset.

MAWILab [28] is committed to providing marked real backbone network traffic. The packet data used in this study are sampled from B and F observation points in the WIDE (Widely Integrated Distributed Environment) backbone network in Japan [29], including a 15-minute record between Japan and the United States over the transpacific submarine cables. This study uses four unsupervised anomaly detection models to predict labels for datasets and determine packet labels through trust scores. This dataset also includes timestamps, real-world network behavior, anomalies, and proportions.

3.1 Preprocessing

The collection of NetFlow data is more accessible than the raw packet data because they only need to extract the information of the packet header. In addition, compared with the original packet information, NetFlow data reduces the storage of packet payload data and integrates the packets of the same session into a piece of flow information, which in practice has the advantage of less storage space compared with the original packet data. Most organizations prefer to store NetFlow data, which is why we use NetFlow data.

MAWILab's packet dataset provides raw packet data,

so we need to convert the data into NetFlow format first. The most common tool for packet data format conversion is nfpcapd, which can convert captured raw packet data into NetFlow format data. However, the real-world packet data provided by MAWILab is incomplete and asymmetric, which causes nfpcapd to fail and crash during processing. In addition, we also tried to use tshark to convert the packet data format. Due to the excessive amount of packets on the backbone network, tshark was interrupted due to insufficient memory.

After trying many methods of packet data format conversion, we determined that there is no off-the-shelf tool to process the packet data of MAWILab, so we chose to "replay" packet data to the observation point and then form the NetFlow data from the observation point and send it to the flow data collection point (NetFlow collector). pmacct [30] is an open-source network analysis tool project that includes various network tool modules and is integrated with pmacctd. Among its many features, pmacctd can read a pcap file consisting of captured packets and use the nfprobe module to form NetFlow data and send it to the NetFlow collector. On the NetFlow collector, nfcapd will be responsible for storing the collected packet flow data and then using nfdump to export the data into plain text format for further processing (as shown in Figure 3).

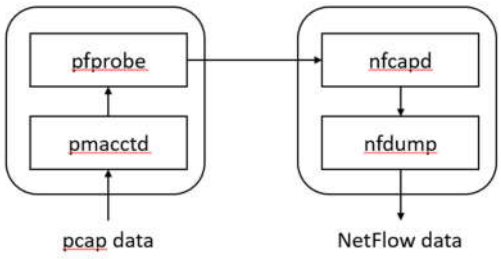

Figure 3. Schematic diagram of data preprocessing

3.2 Feature Extraction

Each node in the network plays two roles simultaneously: session initiator and session receiver. A session is a connection established between two nodes to transmit information. After the information transmission completes, the session ends. It is also the basic unit of NetFlow recording packet flow information. As the name suggests, the session initiator is the party that initiates the session, which is equivalent to the source IP address in NetFlow data; and the session receiver represents the destination IP address. We record the number of connections, the number of ports opened, the number of transmission packets, and the size of transmission packets for each node to form a data feature vector (see Table 1). The definition of characteristic parameters is detailed as follows:

- nSrcLink: The number of nodes connected to node x when node x is the session's initiator.
- nDstLink: The number of nodes connected to node x when node x is the session's receiver.
- nSrcPort: The number of ports used by node x to establish a connection when node x is the session's initiator.
- nDstPort: The number of ports used by node x to establish a connection when node x is the session's receiver.
- nInPkt: The number of packets received by node x when node x is the session's receiver.
- nInByte: The packet size is received by node x when node x is the session's receiver.
- nOutPkt: The number of packets sent by node x when node x is the session's initiator.
- nOutByte: The size of the packet sent by node x when node x is the session's initiator.

Table 1. Description of data characteristics

Symbol	Description
nSrcLink	Number of source IP addresses connected
nDstLink	Number of destination IP addresses connected
nSrcPort	Number of port connected as source
nDstPort	Number of port connected as destination
nInPkt	Number of packets received
nInByte	Number of bytes received
nOutPkt	Number of packets sent
nOutByte	Number of bytes sent

We can observe that nodes with similar network behavior have similar changes in their characteristics over time, so we use time series data to describe the characteristics on the timeline. Next, we divide the NetFlow data set converted from MAWILab packet data into 60 time segments with a time window size of 15 seconds [20]. Finally, the eigenvectors of each node are calculated in each time segment, and each node's eigenvector time series data are formed.

4. Anomaly Detection Method

In the past, many researchers have worked hard in the field of network anomaly detection, resulting in a variety of detection methods and detection models. Among them, the detection methods using machine learning models can be subdivided into unsupervised, supervised, and semi-supervised learning. The supervised learning model can accurately judge similar abnormal network behaviors, but collecting marked sample data for training takes time. The abnormal traffic that behaves differently from the sample data will not be correctly judged. Unsupervised learning

models have the lowest dependence on professional knowledge in the application field, but because of the lack of expert guidance and allowing the mathematical model learn by itself, such network anomaly detection models have a low accuracy rate. Semi-supervised learning models combine the characteristics of the former two, which operate in much the same way as unsupervised learning, except that domain experts can influence the computations of such models through labeled data. We believe that the network environment will become more and more complex in the future, and the number of packets that need to be marked by domain experts will also increase. The application scenarios of supervised learning and semi-supervised learning methods will be limited to small network environments (such as Local area networks), so we choose to use unsupervised learning to detect abnormal network behavior and expect it to apply to various types of network environments.

4.1 Dynamic Time Warping (DTW)

In time series analysis, DTW [31] is an algorithm for measuring the similarity between the time series, which can find the data of corresponding time points from the time deviation. For example, if two people speak the same sentence at different rates, or if there is a pause during the observation process, DTW can detect the similarity between speeches of the two people.

DTW first establishes a distance matrix $C \in R^{N\times M}$ between time series (X, Y). This distance matrix can help aligning X and Y time series data:

$$C_l \in R^{N\times M}: c_{i,j} = \|x_i - y_j\|, i \in [1:N], j \in [1:M], (1)$$

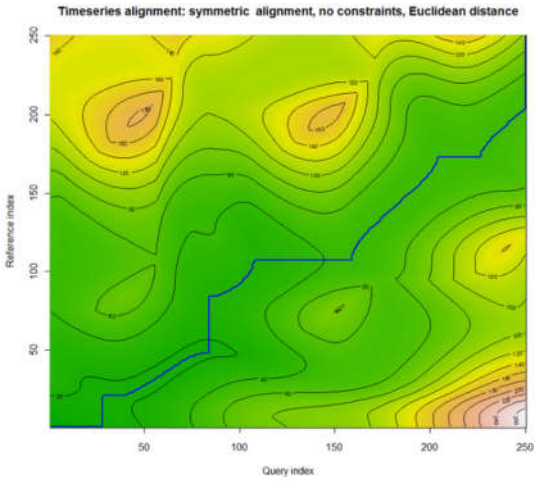

Figure 4. Time series distance matrix heatmap

DTW then finds the lowest-cost path with minimal variance from this distance matrix (see Figure 4 **[32]**). The similarity between time series can be obtained by summing the cost on the path, and the cost calculation formula is:

$$c_p(X, Y) = \sum_{l=1}^{L} c(x_{n_l}, y_{m_l}), \quad (2)$$

DTW replaces Euclidean distance, a method commonly used in clustering algorithms, with the distance between time series data of eigenvectors (see Figure 5). In the algorithm we designed, we also adopt the similarity calculated by DTW.

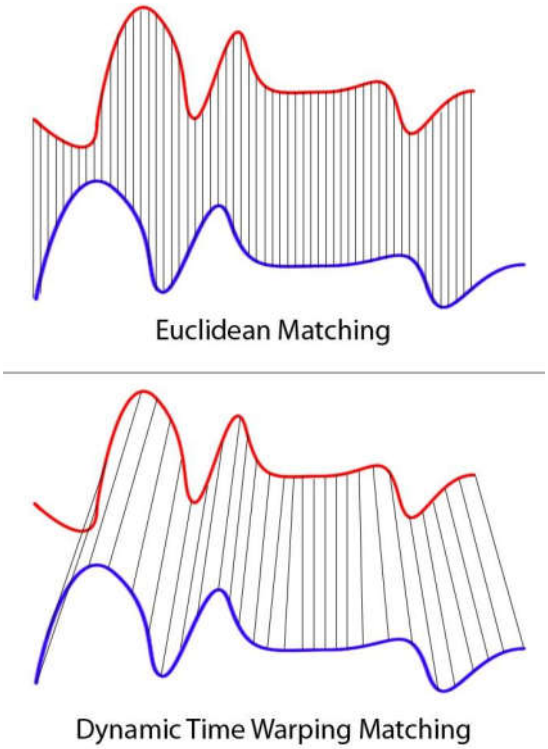

Figure 5. Compare Euclidean and DTW [33]

4.2 Revised Fuzzy C-Means (RFCM)

Since real-world data is full of noise and outliers, these data can affect the clustering algorithm in marking cluster centers. According to the research of Salar Askari [34], the location of clustering centers of the clustering algorithms, except for DBSCAN [35], are all affected when faced with a large number of noisy data sets. DBSCAN needs to define the adjacent value ε and the critical value ρ_{min} of the density first, both of which need to find the point with the most significant change through the k-distance method. By doing so, we can define the adjacent value ε and compare different k adjacent nodes to obtain the most suitable ε and a combination of ρ_{min}. Taking the test data of [34] as an example, $\varepsilon = 0.03$ and $\rho_{min} = 8$ were obtained through the k-distance algorithm proposed by [35], but the clustering result obtained by this value is not ideal (as shown in Figure 6). Network behavior changes over time, and it would be challenging to implement DBSCAN in a network environment to update the model automatically.

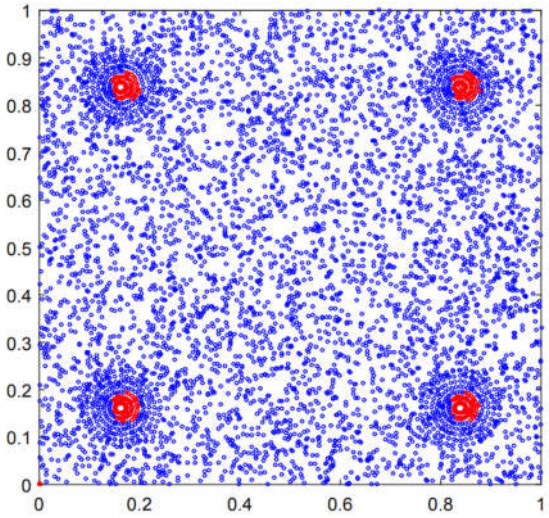

Figure 6. DBSCAN clustering results ($\varepsilon = 0.03$, $\rho_{min} = 8$) [34]

RFCM [34] divides the clustering algorithm into two parts to calculate the objective function: one is used to prevent larger clusters from pulling smaller cluster centers (Size-insensitive), and the other is used to exclude noise and outliers for cluster center interference (Noise-resistant). The cluster centers are obtained by minimizing the size-insensitive objective function and then adjusted with the noise-resistant objective function. Here is the target function for the size-insensitive purpose:

$$J = \sum_{j=1}^{n} \sum_{i=1}^{c} u_{ij}^{m} \left\| x_j - v_i \right\|_A^2, \quad (3a)$$

$$\sum_{k=1}^{c} u_{kj} = \rho_j(X, U), \quad (3b)$$

n is the number of profiles, c is the number of target clusters, u is membership, x is the profile, v is the cluster center, and ρ is used to reduce the influence between clusters. We rewrite Equation 3b with the Lagrange multiplier method and combine it with Equation 3a, and rewrite the objective function as:

$$J = \sum_{j=1}^{n} \sum_{i=1}^{c} u_{ij}^{m} \left\| x_j - v_i \right\|_A^2 + \sum_{j=1}^{n} \lambda_j \left(\sum_{k=1}^{c} u_{kj} - \rho_j \right), \quad (4)$$

We perform partial differentiation of u and v separately for the size-insensitive objective function (Equation 4) and minimize the objective function. After a simple formula tidying up, we can get the updated equations for u and v parameters (such as Equation 5a and Equation 5b).

$$v_i = \frac{\sum_{j=1}^{n} u_{ij}^{m} x_j}{\sum_{j=1}^{n} u_{ij}^{m}}, \quad (5a)$$

$$u_{ij} = \rho_j \left[\sum_{k=1}^{c} \left(\frac{\left(1 - \frac{\partial \rho_j}{\partial u_{kj}}\right) \left\| x_j - v_i \right\|_A^2}{\left(1 - \frac{\partial \rho_j}{\partial u_{ij}}\right) \left\| x_j - v_k \right\|_A^2} \right)^{\frac{1}{m-1}} \right]^{-1}, \quad (5b)$$

From this, we can generate the algorithm pseudocode for Size-insensitive RFCM (Algorithm 1).

Algorithm 1 – Size-insensitive RFCM

Inputs: X, c, ϵ, m, τ, p
Outputs: U, V

U = rand(c, n)
for t = 1 : τ

$$v_i = \frac{\sum_{j=1}^{n} u_{ij}^{m} x_j}{\sum_{j=1}^{n} u_{ij}^{m}} \; \forall i \in [1, c]$$

$$S_i = \frac{1}{|X|} \sum_{x_j \in A_i} \left(1 + \frac{u_{ij}}{|X|^p}\right) \; \forall i \in [1, c]$$

$$i = arg\ arg\ (u_{rj}) \quad \forall j \in [1, n]$$

$$\rho_j = 1 - S_i \; \forall j \in [1, n]$$

$$\frac{\partial \rho_j}{\partial u_{ij}} = \begin{cases} -\frac{1}{|X|^{p+1}}, if\ i = arg\ arg\ (u_{ij}) \\ 0, Otherwise \end{cases} \forall i \in [1, c], \forall j \in [1, n]$$

$$u_{ij} = \rho_j \left[\sum_{k=1}^{c} \left(\frac{\left(1 - \frac{\partial \rho_j}{\partial u_{kj}}\right) \left\| x_j - v_i \right\|_A^2}{\left(1 - \frac{\partial \rho_j}{\partial u_{ij}}\right) \left\| x_j - v_k \right\|_A^2} \right)^{\frac{1}{m-1}} \right]^{-1} \forall i \in [1, c], \forall j \in [1, n]$$

if $\left\| V^{t+1} - V^{t} \right\| \leq \epsilon$
Stop
else
end
end

The optimized u and v are then used as starting values for the following noise-resistant objective function:

$$J = \sum_{j=1}^{n} \sum_{i=1}^{c} u_{ij}^{m} f\left(\left\| x_j - v_i \right\|_A^2 \right), \quad (6a)$$

$$\sum_{k=1}^{c} u_{kj} = \phi_j(X, U), \quad (6b)$$

$$f\left(\left\| x_j - v_i \right\|_A^2 \right) = 1 - exp\ exp\left(-\frac{\left\| x_j - v_i \right\|_A^2}{\omega_i^2} \right), \quad (7)$$

The distance function $f\left(\left\| x_j - v_i \right\|_A^2 \right)$ includes an exponential function (as in Equation 7) to reduce the effect of noise and outliers on cluster centers. Like the objective function in the previous section, we rewrite Equation 6b with the Lagrange multiplier method and combine it with Equation 6a, and rewrite the objective function as:

$$J = \sum_{j=1}^{n} \sum_{i=1}^{c} u_{ij}^{m} f\left(\left\| x_j - v_i \right\|_A^2 \right) + \sum_{j=1}^{n} \lambda_j \left(\sum_{k=1}^{c} u_{kj} - \phi_j \right), \quad (8)$$

Among them, ϕ is a user-defined value, and the default is 0. Again, a partial differentiation is separately applied to u and v for the Noise-resistant objective function (Equation 8) and minimizes the objective function. After a simple formula arrangement, we can get the updated equations of u and v parameters (such as Equation 9a and Equation 9b) and calculate the final clustering result.

$$v_i = \frac{\sum_{j=1}^{n} u_{ij}^{m} f'\left(\left\| x_j - v_i \right\|_A^2 \right) x_j}{\sum_{j=1}^{n} u_{ij}^{m} f'\left(\left\| x_j - v_i \right\|_A^2 \right)}, \quad (9a)$$

$$u_{ij} = \phi_j \left[\sum_{k=1}^{c} \left(\frac{\left(1 - \frac{\partial \rho_j}{\partial u_{kj}}\right) f\left(\|x_j - v_i\|_A^2\right)}{\left(1 - \frac{\partial \rho_j}{\partial u_{ij}}\right) f\left(\|x_j - v_i\|_A^2\right)} \right)^{\frac{1}{m-1}} \right]^{-1}, \quad (9b)$$

We can sort and generate pseudocodes for Noise-resistant RFCM algorithms (Algorithm 2).

Algorithm 2 – Noise-resistant RFCM

Inputs: X, c, ϵ, m, α, τ, p
Outputs: U, V

[U, V] = Size-insensitive RFCM (X, c, ϵ, m, τ, p)
for t = 1 : τ

$$S_{ij} = \left[\sum_{k=1}^{c} \left(\frac{\|x_j - v_i\|_A^2}{\|x_j - v_k\|_A^2} \right)^{\frac{1}{m-1}} \right]^{-1} \quad \forall i \in [1, c], \forall j \in [1, n]$$

$$\omega_i^2 = \frac{\sum_{j=1}^{n} s_{ij}^m \|x_j - v_i\|_A^2}{\alpha \sum_{j=1}^{n} s_{ij}^m} \quad \forall i \in [1, c]$$

$$u_{ij} = \phi_j \left[\sum_{k=1}^{c} \left(\frac{\left(1 - \frac{\partial \rho_j}{\partial u_{kj}}\right) f\left(\|x_j - v_i\|_A^2\right)}{\left(1 - \frac{\partial \rho_j}{\partial u_{ij}}\right) f\left(\|x_j - v_k\|_A^2\right)} \right)^{\frac{1}{m-1}} \right]^{-1} \quad \forall i \in [1, c], \forall j \in [1, n]$$

$$v_i = \frac{\sum_{j=1}^{n} u_{ij}^m f'\left(\|x_j - v_i\|_A^2\right) x_j}{\sum_{j=1}^{n} u_{ij}^m f'\left(\|x_j - v_i\|_A^2\right)} \quad \forall i \in [1, c]$$

if $\|V^{t+1} - V^t\| \leq \epsilon$
Stop
else
end
end

Since the data we use is time series data with eigenvectors, we replace the Euclidean distance used by the RFCM algorithm to calculate the distance to DTW distance so that the algorithm can calculate the distance between time series data.

4.3 Evidence Accumulation Clustering (EAC)

EAC can accumulate the data correlation of the results of multiple clustering algorithms. In addition to stabilizing the output of the clustering results and approaching the actual clustering results, it can also explain the correlation between the data. We use four kinds of IP netmasks (*/8, */16, */24, */32) for feature fusion to prepare multiple copies of the data and use the Elbow method to predict the optimal number of clusters and the ones that are one unit greater and smaller. Three kinds of target grouping numbers use RFCM for grouping to get the grouping result.

The correlation $S(p, q)$ matrix and the outlier $D(o)$ vector are responsible for recording data status between different clustering results in EAC. The correlation $S(p, q)$ matrix is an $n \times n$ matrix, the purpose is to record the correlation strength of the data in the cluster, and the value recorded in the matrix increases only if the data p and the data q are clustered into the same cluster. The outlier $D(o)$ vector is a length n vector whose purpose is to record the distance between the outlier data and the normal data. Each clustering will record the distance of the data o to the center of the adjacent cluster in the vector. Finally, we can find clusters of abnormal behavior from the correlation $S(p, q)$ matrix and find outliers from values in $D(o)$ vectors above the critical threshold by an agglomerative clustering algorithm (Figure 7).

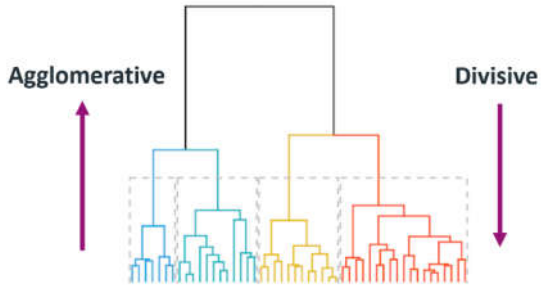

Figure 7. Schematic diagram of agglomerative clustering and divisive clustering

5. Experiment

In this section, we use the downloadable dataset from MAWI Lab as the validation of the proposed method. We used the network packet data captured on 03/01/21 and the anomaly detection results obtained by MAWI Lab, integrating four different algorithms as our ground truth. The accuracy of the proposed method is then calculated.

5.1 Confusion Matrix & Accuracy

We aim to detect packet flow anomalies, using this as the presence of a condition to form a confusion matrix. The confusion matrix consists of four quadrants:

- True positive (TP): a truth in binary classification in which a test result correctly indicates the presence of a condition.
- False positive (FP): an error in binary classification in which a test result incorrectly indicates the presence of a condition.
- False negative (FN): an opposite error where the test result incorrectly indicates the absence of a condition when it is actually present.
- True negative (TN): an opposite truth where the test result correctly indicates the absence of a condition.

Accuracy is used to measure how well a binary classification test correctly identifies or excludes a condition. That is, the accuracy is the proportion of correct predictions (both true positives and true negatives) among the total number of cases examined. The formula for quantifying binary accuracy is:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

5.2 Result

We use the agglomerative clustering method to aggregate the results of the similarity matrix and calculate the accuracy of the detection of various attack techniques (see Table 2). The result shows that the accuracies of each kind of anomaly are around 92%.

Table 2. Detection accuracy

Attack tech.	Accuracy
Alpha Flow	92.1057%
Heavy Hitter	92.1602%
Multipoint to Multipoint	91.9031%
Multipoint to Point	92.0511%
Multipoint to Point Low Activity	92.0200%
Point to Multipoint	92.0979%
Point to Multipoint HTTP	92.1602%
Network Scan UDP Other	92.1057%
Network Scan UDP UDP ICMP Response	92.1602%
Small Alpha Flow	92.0433%
Small Network Scan SYN	92.0433%
Small Point to Point Denial of Service SYN	92.1524%

6. Discussion

Although our method can detect different types of attacks, not all can be detected effectively (there are still cases of false positives), and there are still undetectable attacks.

We believe that the occurrence of false positives may be affected by how the similarity matrix is calculated, in addition to the fact that the selected feature parameters may not provide enough information to distinguish abnormal behaviors. When we record the similarity of feature time series between data, we only calculate simple accumulation. Although this is enough to represent the similarity between data, it may cause the similarity matrix to lose the benchmark for comparison.

Most of the undetected attacks are network scans using TCP. We speculate that attacks such as scanning last for a short time, and we use a sequence of time as a feature, which may cause our method to be insensitive to scanning behaviors. This problem may be solved by replacing the calculation of time series distances. Alternatively, it can be jointly defended by two different anomaly detection methods, targeting short-term and long-term abnormal behaviors.

7. Conclusion

The proposed method uses the data of a time series instead of a single time point. Furthermore, we take advantage of the time series of the feature vector as the node's behavior so that the time parameters can be recorded as the time series changes during the comparison process. At the same time, the DTW we use can allow temporal dislocations between time series data, which helps us to find similar behaviors at similar times. Our method also combines EAC to make the clustering results of time series more stable and can be used to judge abnormal behavior.

Acknowledgment

This research was partially supported by the joint funding of Puli Christian Hospital and National Chi Nan University under the grant number 111-PuChi-AIR-006.

References

[1] "WannaCry ransomware attack." [Online]. Available: https://en.wikipedia.org/wiki/WannaCry_ransomware_attack.

[2] "Top 11 Ransomware Attacks in 2020-2021." [Online]. Available: https://www.keepnetlabs.com/top-11-ransomware-attacks-in-2020-2021/.

[3] A. Mundo, "Nemty Ransomware – Learning by Doing," 2020. [Online]. Available: https://www.mcafee.com/blogs/other-blogs/mcafee-labs/nemty-ransomware-learning-by-doing/.

[4] J. Barabas, "IaaS, PaaS and SaaS – IBM Cloud service models." [Online]. Available: https://www.ibm.com/cloud/learn/iaas-paas-saas.

[5] Sophos, "The 3 Trends Defining Ransomware in 2021," 2020. [Online]. Available: https://www.msspalert.com/cybersecurity-guests/the-3-trends-defining-ransomware-in-2021/.

[6] CrowdStrike, "2021 Globla Threat Report," 2021. [Online]. Available: https://www.crowdstrike.com/resources/reports/global-threat-report/.

[7] M. Midler, "Ransomware as a Service (RaaS) Threats," 2020. [Online]. Available: https://insights.sei.cmu.edu/sei_blog/2020/10/ransomware-as-a-service-raas-threats.html.

[8] B. Krebs, "Attacks Aimed at Disrupting the Trickbot Botnet," 2020. [Online]. Available: https://krebsonsecurity.com/2020/10/attacks-aimed-at-disrupting-the-trickbot-botnet/.

[9] "Security firms call Microsoft's effort to disrupt botnet to protect against election interference ineffective," 2020. [Online]. Available: https://www.washingtonpost.com/technology/2020/10/16/microsoft-trickbot-intel-471/.

[10] "CVE-2017-0143," 2016. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0143.

[11] "CVE-2017-0144," 2016. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2017-0144.

[12] "CVE-2017-0145," 2016. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0145.

[13] "CVE-2017-0146," 2016. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2017-0146.

[14] "CVE-2017-0148," 2016. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0148.

[15] "CVE-2017-0147," 2016. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0147.

[16] B. Duncan, "Goodbye Mworm, Hello Nworm: TrickBot Updates Propagation Module," 2020. [Online]. Available: https://unit42.paloaltonetworks.com/goodbye-mworm-hello-nworm-trickbot-updates-propagation-module/.

[17] E. Bigdeli, M. Mohammadi, B. Raahemi, and S. Matwin, "Incremental anomaly detection using two-layer cluster-based structure," *Information Sciences,* vol. 429, pp. 315-331, 2018/03/01/ 2018, doi: https://doi.org/10.1016/j.ins.2017.11.023.

[18] D. Reynolds, "Gaussian Mixture Models," in *Encyclopedia of Biometrics*, S. Z. Li and A. Jain Eds. Boston, MA: Springer US, 2009, pp. 659-663.

[19] J. Dromard, G. Roudière, and P. Owezarski, "Online and Scalable Unsupervised Network Anomaly Detection Method," *IEEE Transactions on Network and Service Management,* vol. 14, no. 1, pp. 34-47, 2017, doi: 10.1109/TNSM.2016.2627340.

[20] J. Mazel, "Unsupervised network anomaly detection," 12/19 2011.

[21] J. Mazel, P. Casas, Y. Labit, and P. Owezarski, *Sub-Space clustering, Inter-Clustering Results Association & anomaly correlation for unsupervised network anomaly detection*. 2011, pp. 1-8.

[22] J. Mazel, P. Casas, R. Fontugne, K. Fukuda, and P. Owezarski, "Hunting attacks in the dark: clustering and correlation analysis for unsupervised anomaly detection," *International Journal of Network Management,* vol. 25, no. 5, pp. 283-305, 2015, doi: https://doi.org/10.1002/nem.1903.

[23] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 DARPA off-line intrusion detection evaluation," *Computer Networks,* vol. 34, no. 4, pp. 579-595, 2000/10/01/ 2000, doi: https://doi.org/10.1016/S1389-1286(00)00139-0.

[24] J. McHugh, "Testing Intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory," *ACM Trans. Inf. Syst. Secur.,* vol. 3, no. 4, pp. 262–294, 2000, doi: 10.1145/382912.382923.

[25] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 8-10 July 2009 2009, pp. 1-6, doi: 10.1109/CISDA.2009.5356528. [Online]. Available: https://ieeexplore.ieee.org/document/5356528/

[26] P. Owezarski, "A database of anomalous traffic for assessing profile based IDS," presented at the Proceedings of the Second international conference on Traffic Monitoring and Analysis, Zurich, Switzerland, 2010. [Online]. Available: https://doi.org/10.1007/978-3-642-12365-8_5.

[27] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann, *NetFlow Datasets for Machine Learning-based Network Intrusion Detection Systems*. 2020.

[28] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, "MAWILab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking," presented at the Proceedings of the 6th International COnference, Philadelphia, Pennsylvania, 2010. [Online]. Available: https://doi.org/10.1145/1921168.1921179.

[29] K. Cho, K. Mitsuya, and A. Kato, "Traffic data repository at the WIDE Project," 01/07 2002.

[30] "pmacct project." http://www.pmacct.net/ (accessed.

[31] M. Müller, "Dynamic time warping," *Information Retrieval for Music and Motion,* vol. 2, pp. 69-84, 01/01 2007, doi: 10.1007/978-3-540-74048-3_4.

[32] P. Senin, "Dynamic Time Warping Algorithm Review," 01/01 2009.

[33] D. Iskandaryan, F. Ramos, and S. Trilles Oliver, "Air Quality Prediction in Smart Cities Using Machine Learning Technologies Based on Sensor Data: A Review," *Applied Sciences,* vol. 10, p. 2401, 04/01 2020, doi: 10.3390/app10072401.

[34] S. Askari, "Fuzzy C-Means clustering algorithm for data with unequal cluster sizes and contaminated with noise and outliers: Review and development," *Expert Systems with Applications,* vol. 165, p. 113856, 2021/03/01/ 2021, doi: https://doi.org/10.1016/j.eswa.2020.113856.

[35] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," presented at the Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Portland, Oregon, 1996.