**Abstract**

An increasing amount of malicious traffic by malware recently incurs severe security incidents, making cyberspace less secure. It is known that some malware often presents synchronized behavior, because specific actions are performed simultaneously by command & control servers or attackers. Our earlier work monitors traffic arriving of darknet and detects malicious cooperative activities of the traffic sources by using the graphical lasso, a well-known sparse learning algorithm. The method was effective but was unable to cope with real-time analysis because it relies on batch learning. In this study, we propose the method so as to capture anomalously cooperative behaviors in real-time. At the same time, program processing time is shortened. To demonstrate the feasibility of the proposed method, a prototype was implemented, and several security incidents were detected using real darknet traffic data. A performance evaluation was also conducted in terms of processing time and accuracy of security alerts to demonstrate the effectiveness and efficiency of the proposed method.

**Ⅰ. Introduction**

The threat on cyberspace has continued to grow and diverse recently, and the number of variants and unknown attacks are increasing. The Network Incident Analysis Center for Tactical Emergency Response (immediately ICTER) detects such detrimental incidents and threats on the Internet, and releases threat information and takes effective countermeasures. NICTER currently cooperates with organizations in about 40 countries, operates and monitors a large-scale black hole sensor darknet of around 300,000 IP addresses to grasp the trend of malicious activities occurring on the Internet. The darknet refers to a set of accessible but unused subspaces of the Internet. The black hole sensor is a receiver (sensor) that collects packets without responding to senders of packets at all. Almost all packets reaching darknets originate from malicious activities. Darknets reflect global trends in malicious activities of the Internet, and a lot of research analyzes darknet traffic on the Internet.

In recent years, the traffic volume of our darknets and cyber-attacks have increased dramatically every year. Our organization has detected abnormality from NICTER on a heuristic rule-based, but the number of alerts is increasing, and a diversification of attacks, consequently it has become more difficult to detect alerts from usual methods. Therefore, we worked on a method using machine learning which can automatically detect anomalies with high accuracy.

Akiyama et al. defined 3 metrics for detecting botnets through analyzing their behaviors: relationship, response, and synchronization (cooperativeness). Among these metrics, we focused on cooperative activities of malware that simultaneously operates on multiple hosts (devices) synchronously from darknet traffic. We devised a method to detect malicious activities automatically from darknet traffic by estimating cooperative relationships among hosts based on the unsupervised machine learning method. Specifically, among the types of malware that perform cooperative actions, hosts that collaborate with infected devices with the same malware such as worms, malicious scan activities and exploit codes to attack the vulnerability are our detection targets. Moreover, infected devices with botnets that spread like worms targeting IoT devices, such as Mirai, Bashlite and Hajime are also subjects of detection. In paper [13], it does not require communications between infected devices to detect them, so it can detect infected devices regardless of C&C server's structures and protocols used.

In other words, it is effective even if botmasters change their C&C communication protocol and structure.

Also, it does not require payload information. Furthermore, since our method captures the synchronous behaviors between hosts without supervisors and labeled data, irrespective of whether malware is known, even unknown attacks can be effectively detected if cooperative actions can be captured.

We briefly show the procedure of our previous study below. An input of this method is the source host IP addresses observed during a certain period of traffic and the number of packets received by those hosts. Then, Cooperative relationships among the hosts are estimated using the graphical lasso, which is well-known as a sparse structure learning algorithm. Finally, values obtained by quantifying the estimated cooperative relationship using a graph density is an output. A case where the graph density value during a certain period is abnormally higher than other periods is determined as an alert. During the certain period, it is judged that malicious activities have been made to the destination port which is the highest ratio of unique hosts among the period, and the hosts that sent packets to that destination port are regarded as malicious hosts. It helps to investigate causes of malicious activities.

However, this method requires 3 days of traffic data as a sample for detection, so it needs to store traffic data of 3 days before processing, hence the detection will be delayed by 3 days at most excluding the processing time. In other words, a real-time detection is impossible. Due to a characteristic of cybersecurity that must grasp cyberattacks as soon as possible to take prompt countermeasures, the impossibility of real-time processing is a great weak point. Accordingly, the main challenge of this paper is to enable this darknet traffic analysis method using machine learning to be detected in real-time.

**Contribution.** This paper offers the following contributions:

1. The previous work was effective but restrictive because its process was a batch learning type, which means that it could not be used for real-time analysis. Therefore, we proposed a new algorithm and an alert judgment method for online processing.
2. We reduced computational cost through parameter tuning.
3. We present several instances among the alerts obtained from our proposed method's continuous operation.

**Ⅱ. Related Work**

Many techniques have been studied for detection of botnets via cooperative behaviors. BotSniffer uses statistical algorithms to detect botnets in a centralized topology on the basis of their multiple crowd-like behaviors. BotMiner uses a detection framework that extends BotSniffer to perform clustering with monitored C&C communication and malicious activities, respectively, and issue final detection results by cross-correlation. Garcia et al. applied a clustering algorithm to detect synchronization in bots and botnets behavior. This synchronization was studied as the relationship between IP addresses, ports, and time frames only. In this study, malware performing coordinated malignant activities is detected by capturing cooperative relationships between hosts using the graphical lasso.

Next, we investigated studies that used darknets as a data source. Many studies using darknets are ongoing.

Dainotti et al. proposed a methodology for removing spoofed traffic from both darknets and live networks, and contributed to utilization inference of IP address spaces. Durumeric et al. analyzed a large-scale darknet to investigate scanning activities, disclosed large horizontal scan operations and identified patterns in scanning operations. Fachkha et al. devised an inference and characterization modules for extracting and analyzing Cyber-Physical Systems (CPS) probing activities towards ample CPS protocols by correlating and analyzing various dimensions of a large amount of darknet data. Ban et al. proposed an abrupt-change detection algorithm that can detect botnet-probe campaigns with a high detection rate by exploring the temporal coincidence in botnet activities visible in darknets.

Finally, we devised another method to grasp the cooperative relationships of hosts from the darknet like the present method. Although our method estimates cooperative relations among hosts using the graphical lasso algorithm, the method of Yamauchi et al. decomposes the observed hosts into several host groups using the non-negative matrix factorization (NMF) and captures cooperative relationships. These two different methods differ in characteristics captured when capturing the cooperative relationship of the hosts, so the results may be different. In a past experience, it is known that the method of Yamauchi et al. is not suitable for small-scale data sources. However, our method can capture cooperative behaviors even for small-scale data sources.

**Ⅲ. Previous Studies**

In this section, we provide an overview of the graphical Gaussian model. Next, we present how to apply the model to darknet traffic data and present an outline of sparse structure learning using the graphical lasso algorithm. Finally, we describe the introduction of graph densities.

1. *Graphical Gaussian Model*

These studies used the graphical Gaussian model (hereinafter referred to as the GGM), a basic model that expresses linear dependencies between variables in a graph. In the GGM, it is assumed that a N-dimensional random variable sequence \bm{x}=(x\_{1},x\_{2},\cdots,x\_{N})^\mathrm{T} \in\mathbb{R}^N follows the N-dimensional multivariate Gaussian distribution. The multivariate Gaussian distribution can be expressed as

\mathcal{N}({\bm{x}}|\mu,\Sigma)=\frac{\mathrm{det} (\Sigma^{-1})^{1/2}}{(2\pi)^{N/2}}\exp\left(-\frac{1}{2}(\bm{x}-\mu)^\mathrm{T}{\Sigma^{-1}}(\bm{x}-\mu)\right),

where \mu\in\mathbb{R}^N is the mean of \bm{x}, \Sigma\in\mathbb{R}^{N \times N} is a covariance matrix, and \Sigma^{-1}\in\mathbb{R}^{N \times N} is a precision matrix. Both a covariance matrix \Sigma and a precision matrix \Sigma^{-1} are symmetric positive definite matrix. Also, the determinant of a matrices A, written \mathrm{det}(A).

Under the assumption of the multivariate Gaussian distribution, only when \Sigma^{-1}\_{ij} is 0, then x\_ {i} and x\_ {j} are statistically independent, given other random variables. In other words, it can be expressed as follows.

\Sigma^{-1}\_{ij}=0\Leftrightarrow x\_i \perp x\_j | \bm{x} \setminus \{ x\_i, x\_j \}.

For the Gaussian distribution, independence implies uncorrelatedness.

That is, \Sigma^{-1}\_{ij}=0 indicates that there is no cooperative relationship between variables x\_{i} and x\_{j}.

1. *Applying the GGM to Darknet Traffic Data*

In this section, we present how to apply the GGM to darknet traffic data. When the data is applied to the GGM, the length of observation period used for an one model learning is set to T. For example, when 10 minutes of observed darknet traffic data is used for one-model learning, T refers to 10 minutes. Data observed in an interval of T is called a time slot. Fig. 1 shows an example of how the darknet traffic dataset is divided into time slots.

Next, packet data from N number of source hosts are recorded from darknet traffic data in a time slot.

At this time, the number of packets is counted for each source host, and the length of the time series is assumed to be M. The time series data of the number of packets per unit time (sampling interval) is then created. That is, in this case, M is the number of samples in a time slot and the sampling interval is T/M.

The number of packets at the time of m of the i-th source host is represented by y\_i^{(m)}. Furthermore, since the number of packets is a nonnegative value, we take a logarithm to the variable x\_i^{(m)}=\log(y\_i^{(m)}) in order to make it similar to the Gaussian distribution. However as an exception, if y\_i^{(m)}=0, x\_i^{(m)} is set to \log0.1. That is, a time slot is converted to the following matrix.

D=[D\_{im}]\in\mathbb{R}^{N \times M},\;\;D\_{im} := x\_i^{(m)}.

1. *Sparse Learning Using Graphical Lasso Algorithm*

In this section, we introduce how to estimate the sparse precision matrix by the graphical lasso. A sparse matrix refers to a matrix with few nonzero elements, and in sparse structure learning, a weak cooperative relationship is not an essential cooperative relationship. It is expected that essential cooperative relationships can be extracted by scraping them.

First, a sample covariance matrix S is obtained from input data (2), D. The sample covariance matrix S is substituted into the follow penalized maximize log-likelihood function with a \ell\_1-regularization term based on the multivariate Gaussian distribution (1).

{\rm arg}\mathop{\rm max}\limits\_{\Sigma^{-1} \succ 0} \left({\ln\mathrm{det}(\Sigma^{-1})}-\mathrm{Tr}(S\Sigma^{-1})-r||\Sigma^{-1}||\_{1}\right),

where r is a positive real number.

The graphical lasso is an algorithm to solve (3) at high speed and with a high accuracy using the block coordinate descent method. The graphical lasso is implemented as an R language package ``glasso.'' Input arguments of ``glasso'' are the sample covariance matrix S and the regularization coefficient r, the output is the sparsely estimated precision matrix \hat{{\Sigma}^{-1}}. In the graphical lasso, by adjusting the value of the regularization coefficient r, it is possible to adjust how much the precision matrix is estimated to sparse. As the value of r increases, the precision matrix is estimated to be more sparse. From the above, it is possible to estimate the sparse precision matrix.

1. *Usage of Graph Densities*

Ide et al. expressed cooperative relationships between source hosts as a graph by applying a GGM to traffic data. An undirected graph G = (V, E) has a node set V=\{x\_{1}, \cdots, x\_{N}\} and an edge set E=\{(i,j) \, | \, \Sigma^{-1}\_{ij}\neq0\}. Each node corresponds to a random variable, and the existence of an edge between nodes indicates a presence or absence of a cooperative relationship between nodes (variables). In addition, in the analysis of darknet data, if a variable for the number of packets from a source host is made to correspond to a node, an undirected graph of the precision matrix shows the cooperative relationships between the source hosts.

Developing upon this idea, in our previous research proposed a new criterion for judging abnormality using graph density. Given G=(V,E), where the number of nodes in the graph is N(=|V|) and the number of edges is |E|, the graph density is defined as d=|E| \, / \, N(N-1). The graph density is the ratio of the actual number of edges to the number of edges of the complete graph. If the graph density value is higher than those in the other time slots, it is considered that there are more cooperative relationships between source hosts than in the other time slots. In our previous research, a statistical criterion for determining an abnormally high graph density value was determined and abnormalities were detected effectively. At that time, a criterion for selecting the optimum regularization coefficient r was proposed.

**Ⅳ. Proposed Method**

The method of our earlier work is effective but restrictive because its process is a batch learning type, and it may get alerts to be delayed for 3 days at maximum excluding the processing time. It cannot be used for real-time analysis. In this research, we extend our method to enable an online processing and create an anomaly detection system to issue alerts in real-time. Fig. 2 shows the overview of the proposed method. Our approach was divided into 4 stages: preprocessing, model selection, quantification, online processing & alert judgment. In this section, we introduce an alert judgment method, describe the online processing of anomaly-based detection using the alert judgment method. Finally we discuss the evaluation of processing time reduction.

1. *Alert Judgment Method for Online Processing*

We considered an anomaly detection-based method of discriminating alerts during online processing. We could quantitatively quantize cooperative relationships among hosts in one time slot as graph density. We desired to automatically identify time slots with abnormally high graph densities compared with the graph densities of other time slots by online processing. For this purpose, we prepared graph densities of past K time slots from the current time slot as a comparison object of the anomaly-based detection and considered a method of judging whether the current calculated graph density of a time slot is abnormal.

The method is described as follows.

Let \bm{d} = (d\_{1}, d\_{2}, \cdots, d\_{K}) be a sequence in which graph densities of obtained time slots are arranged in descending order. Next, let \bm{d}\_{(i)} = (d\_{i}, d\_{i+1},\cdots,d\_{K}), where i is a natural number. Let \sigma^{2}\_{(i)} be a variance of elements of \bm{d}\_{(i)}.

At this time, assume that the maximum i that satisfies \sigma^{2}\_{(i+1)}/\sigma^{2}\_{(i)}<\theta \, (0\leq \theta \leq 1) is i\_{max} and set time slots corresponding to (d\_{1},d\_{2}, \cdots,d\_{i\_{max}}) as an alert. \theta is usually set to 0.95. Finally, we look for a destination port with the highest ratio of unique hosts among the time slots judged as alerts and examine the port protocol and the hosts that sent the packets to the destination port.

Also, the sparsity of the estimated precision matrix from the graphical lasso differs depending on the value of regularization coefficient r. That is, graph density values change in accordance with the value of r. In the batch processing method, only one optimum value of r was selected on the basis of input data. However, we believed that was is important to issue an alert rather than deciding the optimal value of r for online processing this time. In this method, instead of selecting only one value of r, we calculate the graphical lasso by setting several values of r in advance and issue an alert when a condition of an alert is satisfied for at least one value of r.

1. *Algorithm for Online Processing*

In this section, we outline the procedure for online processing. Suppose that there is a sequence \mathcal{D}=\{D\_1, D\_2, \cdots, D\_K\} of data matrices obtained from time series of time slots and a set d\_{\mathcal{D}}^{(r)}=\{d^{(r)}\_{D\_1}, d^{(r)}\_{D\_2}, \cdots, d^{(r)}\_{D\_K}\} of K graph densities calculated from \mathcal{D} in advance. d^{(r)}\_{D\_i} (i \in \{1, 2, \cdots, K\}) is the graph density for each predetermined regularization coefficient r in the data matrix D\_i.

It is assumed that the value of r is preset like r \in R ( =\{r\_1, r\_2, \cdots, r\_s\} ). For example, preset R=\{0.2, 0.4, \cdots, 1.8\}. For online processing, the matrix of an updated time slot is given by D\_t.

The pseudocode for online processing is described in Algorithm 1.

Note: Updating \mathcal{D} indicates that D\_t is linked to \mathcal{D} and the oldest time slot in \mathcal{D} is deleted.Similarly, updating d\_{\mathcal{D}}^{(r)} indicates that d\_{D\_t}^{(r)} is linked to d\_{\mathcal{D}}^{(r)} and the oldest graph density in d\_{\mathcal{D}}^{(r)} is deleted.

When updating \mathcal{D} and d\_{\mathcal{D}}^{(r)}, each size is kept at K. In this way, the number of time slots used for an alert determination is fixed like a sliding window. With this procedure, an alert determination can be made sequentially from the updated data, and the computation cost has been reduced because only the graphical lasso needs to be calculated for the updated data.

1. *Processing Time*

Since December 2017, we have been trial operating our proposed method continuously with darknet sensors located in 7 different places operated by NICTER. Table Ⅰ shows details of IP addresses scales of each darknet sensors we used this time. In this section, we evaluate the processing time of the prototype. The specifications the computer used for the evaluation at the Cybersecurity Laboratory, NICT include 16GB of memory and an Intel Xeon E3-1230 v3 3.30 GHz CPU. To perform real-time processing with limited resources, it is necessary to finish a current processing before updating the next time slot. The largest computational complexity of the program is the graphical lasso algorithm O(N^3). N is the number of source hosts in the processing target time slot.

Table Ⅱ shows the average and maximum value of the processing time and the number of source hosts when operated from December 2017 to March 2018. Parameter are set to T=10min.,\, M=6,\, K=432,\, \theta=0.95. As shown in Table Ⅱ, as the scale of the darknet sensor increases, N also increases, so we need to shorten the processing time. Since sensors F and G are actually large in scale, we decided to shorten the processing time by considering the following measures.

･ Restrict the regularization coefficient, r.

･ Ignore the third and fourth octets of IP addresses of source hosts.

･ Select source hosts uniformly at random and limit the number of nodes, N.

Restring r refers to reducing the number of its elements. Ignoring octets means that IP addresses on the same subnet are regarded as one host. Limiting the number of source hosts N indicates that when N exceeds the limit number. Since characteristics of darknet traffic vary depending on its scale and the country in which the sensor is located, it is necessary to periodically adjust the above-mentioned parameters to be limited for each darknet. Actually, in Table Ⅱ, we set sensors A to E at r=\{0.2, 0.4, \cdots, 1.8\}, \, octet=4 and sensors F and G at r=\{0.6, 0.8, \cdots, 1.8\}, \, octet=2\sim3, \, limitation\\_N = 1000. We regularly adjusted the parameters so that online processing would be on time. In addition, If a processing time of a time slot exceeds 900 seconds, the processing is forcibly terminated.

**Ⅴ. Case Studies**

1. *Preprocessing of Data*

In this study, we collect only TCP packets to which a SYN flag was attached. As a feature of the black hole sensor darknet, packets except for SYN such as SYN-ACK, ACK, and RST are often backscatters, and we ignored those packets since they are not our detection target. And we pre-exclude packets arriving from known scanners which are for research and investigation purposes. When a large number of packets constantly arrive at a specific TCP port, or when packets are constantly arriving from many hosts, that TCP ports are excluded. In addition, we expect to get an unknown vulnerable TCP port as an alert by excluding known fragile TCP ports that we have frequently obtain as alerts. We intentionally exclude those ports considering the possibility that other cooperative activities would be buried. Our goal is not only to detect known attacks but also to detect unknown attacks. Detection of unknown attacks increases by excluding constantly targeted or known fragile ports in advance. In operation from December 2017 to March 2018, TCP ports 22, 23, 80, 443, 445, 1433, 2323, 3389, 5358, and 7547 were excluded from the preprocessing.

1. *Summary of Alert Results*

Table Ⅲ shows the monthly alert summary. We ignored alerts with a ratio of unique hosts number less than 10\% and the number of unique hosts less than 50. In addition, we divide priority into 10\% and 40\%. In the case of alerts obtained only about once or twice, it is often the case of a temporary host increase, and many cases were difficult to investigate the cause. Table Ⅳ show the alert summary by each sensor. We did not get alerts for A and B with a small sensor scale. Sensors C, D, and E had important events, but temporary events were also acquired as alerts. In sensors F and G, alert port types are relatively small compared with other sensors, but they were often an important event. In the following, we introduce instances of actual new trends attacks which could be known from these alerts.

1. *Detection Instances Related to ADB Miner (5555/TCP)*

We present detection instances of activities related to a mining botnet utilizing android ADB (Android Debug Bridge). Fig. 3 shows a chronological chart of the number of unique hosts observed across our all darknets per 1 day. On February 4, 2018, a TCP port 5555 was seen to have a rapidly increasing number of unique hosts. We detected it timely. Details of the alerts obtained are described below.

The following items are related to 5555/TCP. ADB.Miner a mining botnet utilizing android ADB (Android Debug Bridge) is spreading via 5555/TCP. It is seen in large quantity from February 4 and about 16,000 hosts could be seen at the peak from NICTER's darknets. Since then, the attack continues while maintaining the scale. According to Netlab 360 report, they detected at 15:00 on February 3 (GMT +8) by their system and the earliest time of the infection could be traced back to January 31. It infects when 5555/TCP (ADB debugging interface) is open and mainly its target is Android devices.

･ Worm infection: Infected devices will initiate a network scan on 5555/TCP, and attempt to execute ADB commands for newly infecting worm like.

･ XMR Mining: After infection, it will dig XMR tokens.

In addition, this worm borrows code from Mirai's SYN scanning module for efficiency. This is the first case where the Mirai's code was seen on Android bots. In the case of Japan, there were many infections when using Android devices from overseas manufacturers and a specific mobile SIM card.

1. *Detection Instances Related to Hajime Variants (8291, 2000/TCP)*

We introduce detection instances of activities related to Hajime botnet variants. Fig. 4 shows a chronological chart of the number of unique hosts observed across our all darknets per 1 day. On March 25, 2018, a TCP port 8291 was seen to have a rapidly increasing number of unique hosts and also on March 31, 2018, a TCP port 2000 was seen to have a rapidly increasing in the same way. We detected these timely. Details of the alerts related to TCP ports 8291 and 2000 are described below.

TCP ports 8291 and 2000 scanning events are caused by a Hajime botnet variant targeting vulnerabilities of router products manufacturer ``X''. 8291/TCP increased sharply from March 25, but attacks with the same behaviors moved from March 31 to 2000/TCP. As a feature of Hajime, C&C communication using BitTorrent's P2P protocol is performed and it supports only ARM and MIPS architecture. Also, the source code is not published and there are few variants. If the TCP port 8291 or 2000 is open, then it tries to infect by accessing following common web ports: 80, 81, 82, 8080, 8081, 8082, 8089, 8181, 8880. It exploits known vulnerabilities of an operating system of router (ChimayRed HTTP exploit, SMB buffer-overflow vulnerability (CVE-2018-7445)) as well as password brute-forcing.

**Ⅵ. Conclusion**

We proposed a method to detect coordinated malignant activities from darknets in real-time using the graphical lasso and an alert judgment method by graph density. The methods were put to practical uses using real darknet traffic, and the processing time was shortened and parameter tunings were evaluated. We were able to show alert summaries and introduce instances of attacks related to ADB Miner and Hajime variants.

Since almost all packets reaching darknets originate from malicious activities, there is no correct or incorrect answer of attacks. Moreover, we do not know the cause of most of it. In other words, even if we catch cooperative relationships from darknet traffic by this method, we can not evaluate accuracy unless we know the cause. For future works, we would like to quantitatively evaluate the accuracy. An alert that knows the cause would be judged as a correct answer, and unknown the cause would be judged as a false-positive. A global attack that could not be detected with this method is judged to be false-negative. We will determine the optimal parameters automatically for each darknet to obtain alerts regardless of its scale in our next step. In addition, filtering ports in preprocessing were done manually until now, but automatic preprocessing is also one of our future works.