Real-Time Detection of Malware Activities by Analyzing Darknet Traffic Using Graphical Lasso

Chansu Han[*,†], Jumpei Shimamura[‡], Takeshi Takahashi[*], Daisuke Inoue[*], Masanori Kawakita[†], Jun'ichi Takeuchi[†,*], and Koji Nakao[*]

[*]National Institute of Information and Communications Technology, Japan. {han, takeshi_takahashi, dai, ko-nakao}@nict.go.jp

[†]Kyushu University, Japan. {kawakita, tak}@inf.kyushu-u.ac.jp

[‡]clwit Inc., Japan. {shimamura}@clwit.co.jp

***Abstract*—Recent malware evolutions have rendered cyberspace less secure, and we are currently witnessing an increasing number of severe security incidents. To minimize the impact of malware activities, it is important to detect them promptly and precisely. We have been working on this issue by monitoring traffic coming into unused IP address spaces, i.e., the darknet. On our darknet, Internet-wide scans from malware are observed as if they are coordinated or working cooperatively. Based on this observation, our earlier method monitored network traffic arriving at our darknet, estimated the degree of cooperation between each pair of the source hosts, and detected significant changes in cooperation among source hosts as a sign of newly activated malware activities. However, this method does not work in real time, and thus, it is impractical. In this study, we extend our earlier work and propose an online processing algorithm, making it possible to detect malware activities in real time. In our evaluation, we measure the detection performance of the proposed method with our proof-of-concept implementation to demonstrate its feasibility and effectiveness in terms of detecting the rise of new malware activities in real time.**

***Index Terms*—Real-time detection, Malware, Network scan, Darknet, Cooperation, Outlier detection.**

I. Introduction

The number of cyberattacks has been growing recently. They often use malware as their tool, and their types have diversified in recent years. To minimize the impact of cyberattacks, it is important to ascertain the actual behaviors of malware on the Internet quickly and precisely.

Many intrusion detection systems and firewalls have already implemented signature-based and/or whitelist/blacklist-based access control functions, but these cannot cope with unknown malware variants and brand-new malware. Network security operators or analysts may conduct a heuristic and rule-based analysis, but this is costly, and human error may occur. Accordingly, a method that can detect the activities of malware of all types automatically, quickly and precisely is now needed.

We focus on the detection of indiscriminate attacks, which do not have any specific targets and attack arbitrary hosts on the Internet. Malware's network scanning activity is one type of such attacks, and malware often performs it to locate attack targets. Since the scans are sent to arbitrary IP addresses, some of them may arrive at unused IP address spaces, i.e., a darknet. To capture these scans, we set up receivers (sensors) in our darknet and captured all the traffic arriving at the sensors.

Through an analysis of the traffic, we intend to ascertain what kind of network service was being targeted and the trend of global cyberattacks. In this study, we work on a mechanism that automatically detects indiscriminate attacks reaching darknets, such as network scans from malware. This mechanism detects cyberattacks quickly and precisely, including variants and unknown attacks by using an unsupervised machine learning method, so that there is no human error.

On our darknet, Internet-wide malware scans are observed as if they are coordinated or working cooperatively. In reality, we believe that each infected host sends scans independently, although there are still minor cases where command-and-control (C2) servers coordinate bots. Here, the difficulty lies in how we should distinguish the ill-intentioned packets from others; darknet traffic includes not only ill-intentioned scans but also legitimate scans with research purposes and unintentionally sent packets due to device misconfiguration.

Based on this observation, our earlier method [7], called the *GLASSO* engine[1], estimates the cooperation of the source hosts by using a sparse structure learning algorithm called the graphical lasso [5]. It also provides alerts when the estimated cooperation changes significantly; it can detect a campaign or a group of related activities by estimating the cooperation among many pairs of source hosts for each unit time and by finding outliers of cooperation among different unit times. This approach is robust with regard to noise: packets that arrive accidentally in our darknet space due to misconfigurations will not show cooperation. Moreover, this approach ignores activities with weak cooperation.

However, since the *GLASSO* engine in [7] works in batch mode and requires three days' darknet traffic to process, the analysis result will not be provided in real time and will be delayed by more than three days in the worst-case. Accordingly, in this paper, we propose an online processing algorithm of the *GLASSO* engine that extends our earlier method [7], making it possible to detect the outliers sequentially and in real time. In our evaluation, we measure the detection performance of the proposed method with our proof-of-concept implementation

[†]The current affiliate of M. Kawakita is Nagoya University, Japan.

[1]The engine is named after the library of the R language "glasso" [6].

to demonstrate its feasibility and effectiveness in terms of detecting the rise of new malware activities in real time. For this evaluation, we manually prepared the list of malicious events and the port numbers they used; these events and port numbers should be identified by the proposed algorithm. With the list, we evaluate the detection performance of the *GLASSO* engine and demonstrate its feasibility and usefulness.

Contribution. This paper offers the following contributions:

1) We proposed an online processing algorithm of the *GLASSO* engine and undertook outlier detection sequentially. It was possible to detect outliers with a very short span compared to earlier work and obtain alert results in real time. (Discussed in Section V)
2) We tuned important parameters in the *GLASSO* engine. (Discussed in Section VI)
3) We operated the *GLASSO* engine in real time using live darknet traffic (raw packets) and obtained detection results without delay. We also analyzed the detection results by observing destination TCP ports (services) of the involved packets. (Discussed in Sections VII-B and VII-C)
4) We evaluated the detection performance of malware activities using the limited ground truth. This paper is the first and practical evaluation of the detection performance of malware activities on the darknet, and we explained the details of the detected malware activities. Finally, we considered some countermeasures against several malware activities that the *GLASSO* engine missed. (Discussed in Sections VII-D, VII-E, and VIII)
5) We published datasets used for performance evaluation in this paper on the web[2]: data matrices $\boldsymbol{D}$ introduced in Section IV-A and the alert results in Section VII-B.

II. Background

In this section, we present an overview of our fundamental tools, i.e., the darknet, the graphical Gaussian model, and the graphical lasso algorithm.

A. Darknet

The Internet can be categorized into two types of networks: livenet and darknet. A livenet is a network of busy IP addresses, while a darknet is a network of unused IP addresses. We have set up receivers (sensors) on our darknets and passively capture all the traffic arriving at these receivers. Most of the packets in the traffic are TCP packets with the SYN flag on, i.e., TCP-SYN packets. TCP-SYN packets reaching a darknet are non-ordinary packets; these are often scans or unintended packets sent by misconfigured network devices. The scans may be initiated by malware for indiscriminately locating the next victim candidates or by any other entities for investigating and researching the Internet. In this paper, a network scan that has indiscriminately attempted attacks and intrusions by malware through vulnerable TCP ports is called a cyberattack, and a network scan that has indiscriminately attempted surveys and research by organizations like Shodan and Censys is called a survey scan.

[2]https://csdataset.nict.go.jp/darknet

B. Graphical Gaussian Model

A graphical Gaussian model (hereinafter, GGM) is a probabilistic model for which a graph expresses the dependence structure between random variables given a multivariate Gaussian distribution. To measure the dependency structure between random variables, one may obtain correlation coefficients, but it may include spurious correlations between random variables. To address this issue, there is a method of obtaining a precision matrix Σ^{-1} (the inverse of the covariance matrix Σ) from which the conditional independence of a pair of random variables can be measured [10]. If and only if $(\Sigma^{-1})_{ij} = 0$, then x_i and x_j are independent, conditioned on all the other variables. The definition of the graph in the GGM using the precision matrix $\Sigma^{-1} \in \mathbb{R}^{N\times N}$ of a sequence of random variables following an N-dimensional multivariate Gaussian distribution is as follows: N random variables correspond to the nodes, and if an element $(\Sigma^{-1})_{ij}$ is zero, there is no edge between variables x_i and x_j, and if the element is nonzero, there is an edge between variables x_i and x_j. In other words, the graph in GGM using this precision matrix shows the conditional independence of all pairs of random variables.

C. Graphical Lasso

The maximum-likelihood estimate of a precision matrix Σ^{-1} is the inverse of the sample covariance matrix S. We expect the precision matrix Σ^{-1} to be a sparse matrix such that for the variable pairs with essential dependencies the corresponding elements are nonzero and for the weakly related variable pairs the corresponding elements are zero. In general, however, the elements of the inverse of the sample covariance matrix S cannot be strictly zero. Accordingly, the graphical lasso, which is a sparse structure learning algorithm, optimizes a precision matrix by solving a penalized maximum-likelihood equation with ℓ_1 regularization term $r\sum_{ij}|\Sigma_{ij}^{-1}|$ [5]. The input arguments of the graphical lasso algorithm are the sample covariance matrix S and the ℓ_1 regularization coefficient r (≥ 0). Here, r is a hyperparameter for deciding how much dependency is regarded as noise-derived, and it is possible to adjust the sparsity of the precision matrix to be estimated. From the above, the GGM graph using the precision matrix $\hat{\Sigma^{-1}}$ estimated by the graphical lasso algorithm expresses the more essential dependencies of all variable pairs than the maximum-likelihood estimate of the precision matrix.

III. Related Work

Many studies using darknets have been carried out and have shown their usefulness for analyzing Internet-wide scanning. Dainotti *et al.* developed and evaluated a methodology for removing spoofed traffic from both darknets and live networks and contributed to supporting census-like analyses of IP address space utilization [2]. Durumeric *et al.* analyzed a large-scale darknet to investigate scanning activities and identifying patterns in sizable horizontal scanning operations [3]. They also presented an analysis of the latest network scanning on the overall landscape, along with its influence, and countermeasures of the defender in detail. Fachkha *et al.* devised inference and characterization modules for extracting and analyzing cyber-physical systems (CPS) probing activities toward sufficient CPS protocols by correlating and analyzing the various dimensions of a large amount of darknet data [4].

Ban *et al.* [1] proposed an abrupt-change detection algorithm that detected botnet-probe campaigns with a high detection rate by exploring the temporal coincidence in botnet activities visible in darknet traffic. Although this method used the abrupt change of time series traffic volume addressed to one TCP port, in recent malware activities, there have been many cases where an abrupt change cannot be undertaken successfully due to multiple campaigns being overlapped or being continuously scanned. Moreover, since the *GLASSO* engine processed the entire live traffic without restrictions, the range of the dataset was different from the abrupt change detection algorithm. In fact, research using the darknet to detect malware activities of similar scale in the same dataset range as the *GLASSO* engine did not exist as far as we know, and it was difficult to compare and evaluate.

IV. *GLASSO* Engine (Batch-mode)

In this section, we introduce our preliminary algorithm briefly, presented in [7], which uses the GGM and the graphical lasso algorithm to analyze darknet traffic.

A. Applying GGM to Darknet Traffic

The *GLASSO* engine takes a time series of the number of packets received from each source host as a variable and applies the GGM to capture the dependency between variables (source hosts). Although this variable is never expressed as a Gaussian distribution, it can be assumed that there were many variables close to the form of a log-normal distribution, so it approximates a Gaussian distribution by log transformation to some extent. Besides, if the variables do not completely comply with a Gaussian distribution but can be approximated to some extent, dependency relationships between the variables in the GGM could be ascertained.

First, we considered how to process the dataset of darknet traffic. The term of T seconds of which darknet traffic is an object of one model learning is called a time slot. Also, let X_t be a set of packet data included in the t-th time slot. We call X a time slot dataset. We assumed that there were N source hosts in the X_t. Time series data were generated by counting the number of packets observed at a certain sampling interval for each source host. Here, if the number of time series samples is M, the sampling interval is T/M *(sec.)*. We then convert from the X_t to a data matrix

$$D_t = [D_{mn}] \in \mathbb{R}^{M \times N}, \;\; D_{mn} := \log(x_n^{(m)}), \;\; \boldsymbol{x}^{(m)} \in \mathbb{N}_0^N.$$

Here, $\boldsymbol{x}^{(m)}$ denotes N-dimensional variables of the number of samples M, and $x_n^{(m)}$ denotes the number of packets at the m-th point of the n-th source host. In addition, $\mathbb{N}_0 = \{0, 1, 2, \cdots\}$. Since log-transformation cannot be performed when $x_n^{(m)} = 0$; we add a small real number 0.1.

Next, we obtain a precision matrix from the data matrix D_t using the graphical lasso algorithm and apply the GGM. A set of source hosts (variable) then corresponds to a node set of a graph in the GGM, and the presence or absence of a dependence relationship (cooperativeness) of a source host pair (variable pair) corresponds to an edge set.

B. Algorithm of Batch-mode GLASSO Engine

We prepare darknet traffic observed over a long period (e.g., three days' traffic) and divided traffic every T seconds to make datasets of multiple time slots $\boldsymbol{X}$. At that time, we consider only SYN packets, TCP. Next, data matrices $\boldsymbol{D}$ are created for $\boldsymbol{X}$, and sample covariance matrices $\boldsymbol{S}$ is obtained. Sample covariance matrices $\boldsymbol{S}$ and a positive real number r are input to the graphical lasso algorithm to obtain a sparsely estimated precision matrices $\hat{(\boldsymbol{\Sigma}^{-1})}^{(r)}$. Here, we try some positive real numbers like $r \in R(= \{r_1, r_2, \cdots, r_s\} \subset [0, \infty))$. From each estimated precision matrix, an undirected graph $G = \{V, E\}$ in the GGM could be represented by node set $V = \{x_1, \cdots, x_N\}$ and edge set $E = \{(i, j) | \hat{(\Sigma^{-1})}_{ij} \neq 0\}$. Then, in order to express the degree of cooperation between all pairs of source hosts in each time slot dataset with scalar values, we obtain a graph density value $d^{(r)} = |E|/N(N-1)$ for each time slot dataset. The graph density value represents the ratio of the actual number of edges to the number of edges of the complete graph. Finally, time slot datasets corresponding to graph densities showing anomalously high values compared to all graph density values are determined using outlier detection techniques.

V. *GLASSO* Engine (Online-mode)

In this section, we propose a novel *GLASSO* engine that analyzes darknet traffic in real time. The engine contains an online processing algorithm and an alert judgment method. The input of the engine the packet capture (PCAP) file of darknet traffic for T seconds and other parameters, while the output is alert information generated from the PCAP files in the time slot datasets classified as outliers. The alert information includes the timestamp, the targeted destination TCP port number, the source IP addresses, and the number of addresses. The targeted TCP port refers to the TCP port with the largest number of source hosts that sent packets to a destination TCP port in that time slot dataset.

A. Online Processing Algorithm

We consider a method of outputting alerts (outliers) sequentially by processing every time slot dataset, rather than by processing multiple time slot datasets at once. First, the newest time slot dataset X is processed in the same manner as in Section IV-B to obtain a graph density value $d^{(r)}$. Then, $d^{(r)}$ is added to a sequence of the past graph density value $\boldsymbol{d}^{(r)}$, and if the length of $\boldsymbol{d}^{(r)}$ is an arbitrary positive integer K, alert judgment (outlier detection) is performed. If the length of $\boldsymbol{d}^{(r)}$ is less than K, we wait until the next time slot dataset is updated without performing the alert judgment. The detailed alert judgment method is explained in the next section. The

Algorithm 1 *GLASSO* Engine with Online Processing

```
Input: $X$, $M, r \in R(= \{r_1, r_2, \cdots, r_s\}), \boldsymbol{d}^{(r)}, K, \theta$
Output: alerts or none (per while-loop)
 1: while $X$ is updated newly do
 2:   preprocess $X$
 3:   make $D$ from $X$
 4:   compute $S$ from $D$
 5:   for $r$ in $R$ do
 6:     compute $(\hat{\Sigma^{-1}})^{(r)}$ using the glasso($S, r$)
 7:     compute $d^{(r)}$ from $(\hat{\Sigma^{-1}})^{(r)}$
 8:     add $d^{(r)}$ to $\boldsymbol{d}^{(r)}$
 9:     if $length(\boldsymbol{d}^{(r)}) = K$ then
10:       call alert-judgment-method($\boldsymbol{d}^{(r)}, K, \theta$)
11:       if there are $outliers$ then
12:         collect alert information from $outliers$
13:         return alerts
14:         delete $outliers$ from $\boldsymbol{d}$
15:       else
16:         delete the oldest time slot dataset from $\boldsymbol{d}^{(r)}$
17:       end if
18:     end if
19:   end for
20: end while
```

Algorithm 2 Pseudo code for *alert-judgment-method*

```
Input: $\boldsymbol{d} \in \mathbb{R}^K, K, \theta$
Output: $outliers$ or none
 1: $i, j \leftarrow 0$
 2: while TRUE do
 3:   $i \leftarrow i + 1$
 4:   $\boldsymbol{d}_{(i)} \leftarrow order(\boldsymbol{d}, decreasing = True)[i : K]$
 5:   $\sigma^2_{(i)} \leftarrow var(\boldsymbol{d}_{(i)})$
 6:   if $\sigma^2_{(i+1)}/\sigma^2_{(i)} < \theta$ then
 7:     $j \leftarrow j + 1$
 8:   else if $i = 1$ then
 9:     break
10:   else
11:     $outliers \leftarrow order(\boldsymbol{d}, decreasing = True)[1 : j]$
12:     return $outliers$
13:   end if
14: end while
```

time slot dataset judged as an outlier is output as an alert and is deleted from $\boldsymbol{d}$ so that it will not be referred to in a subsequent alert judgment. In addition, if no time slot dataset is determined as an outlier, the oldest time slot dataset is deleted from $\boldsymbol{d}^{(r)}$. When the time has passed and a new time slot dataset is updated, by repeating the above steps, it is possible to sequentially process the *GLASSO* engine while maintaining the number of time slot datasets used for alert judgment. As a result, it obtains the result in real time after a very short time compared to the batch mode, since only an updated time slot dataset is processed. The pseudo code for the *GLASSO* engine with online processing is described in Algorithm 1. Here, the $length()$ function obtains the length of the vectors.

B. Alert Judgment Method for Online Processing

In this section, we propose an outlier detection method for differentiating alerts (outliers) from a sequence of graph density values $\boldsymbol{d}$ during online processing. It is checked how much the largest element occupies a total sample variance in the sequence of graph density values $\boldsymbol{d}$, and when the ratio is large, the largest element is judged to be an outlier. The pseudo code of the alert judgment method is indicated in Algorithm 2. Here, $\sigma^2_{(i+1)}/\sigma^2_{(i)} < \theta$ is an outlier judgment formula, and $\theta\,(0 \leq \theta \leq 1)$ is a threshold value of an outlier judgment formula. In addition, the $order()$ function returns a permutation that rearranges its first argument into ascending or descending order, and the $var()$ function returns a sample variance. From the above, we understand how to process the *GLASSO* engine and how to output alerts.

VI. Parameter Tuning

The *GLASSO* engine has five important parameters, and we introduce the way these five parameters are set in this section. These parameters are determined using an empirical heuristic method.

A. Length T of Time Slot

The length of time slot T *(sec.)* means the length of the entire observation time series of the data used for one model learning. First of all, we think from an upper bound of T. The computational complexity of the *GLASSO* engine depends significantly on the number of source hosts N. As T takes longer, the number of source hosts generally also increases, the processing time increases polynomially in N, and real-time processing becomes difficult. Next, we consider a lower bound. It is sufficient if we can make in-depth observations such that all of one campaign of a network scan is observed in T seconds. From our experience, one campaign of many network scans observed at the maximum observation scale of the darknet sensor used in the *GLASSO* engine ends within 5 minutes (subnet /17). Accordingly, we set it to $T = 600$ *(sec.)*, which can be processed in real time without problems, and make in-depth observations.

B. The Number of Time Series Samples M

The number of time series samples M means the number of time series samples of data used for one model learning by dividing the time slot length T into the number M. In other words, it has a meaning of adjusting the length of the sampling interval T/M *(sec.)*. Generally, since the sampling interval is strictly divided, the time series of the number of packets is measured more finely, and learning with good precision is performed. If the sampling interval is too short, however, there is a risk of estimating that there is no dependence between the source hosts that originally work in cooperation (false negatives). In contrast, if the sampling interval is too long, the number of false positives may increase. Accordingly, it is necessary to set M to an appropriate number, but it is difficult to find the optimum value. Fortunately, however, the regularization coefficient r that is introduced in the next section has the same function as this M. That is, unless M is set to an extreme number, it can be covered with the regularization coefficient r. Based on our experience, if the sampling interval T/M is about 50 seconds, we can measure a sufficient time series of the number of packets, so we set $M = 12$.

C. Regularization Coefficient r

The regularization coefficient r (≥ 0) is an input parameter used in the graphical lasso algorithm. It is a hyperparameter to decide how much dependence is considered to be derived from the noise and adjusts the sparsity of the estimate of precision matrix Σ^{-1}. In other words, by adjusting r, it is possible to scrape-down weakly related dependencies and reduce the number of false positives of dependency estimation. Based on this perspective, r and M have similar roles. As a feature of r, $d^{(r)}$ generally becomes closer to 1 as r becomes closer to 0, and $d^{(r)}$ becomes closer to 0 as r becomes larger. In addition, as the value of r is increased, the number of zero elements of a precision matrix estimated by the graphical lasso algorithm increases, so that the calculation time is shortened. Finally, r can be tried multiple times with $R(= \{r_1, r_2, \cdots, r_s\} \subset [0, \infty))$, but the processing time of the *GLASSO* engine increases by the number of trials.

We set the *GLASSO* engine to $T = 600$ *(sec.)*, $M = 12$ and fine-tune the value of r to six decimal places. Even when trying with varying values from six decimal places to two decimal places, the graph density value did not change significantly. For example, our experience shows that time slot datasets that are alerted with $r = 0.55$ are judged alert by r =0.5 or 0.6 in all cases. From this, we discover that it is sufficient to try while changing the value to one decimal place. In addition, there are many zero matrices from $r \geq 1$, which often result in $d^{(r)} = 0$. The processing time also takes longer for $r < 0.4$, where a sufficiently sparse precision matrix is not estimated, and outliers do not emerge in most cases. From the above, we finally decided to set $r \in R(= \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\})$.

D. The Number of Graph Densities K and Threshold θ for Alert Judgment

The number of graph density values K and the threshold θ are used for alert judgment, as well as the relationship between M and r, and unless K is set to an extreme number, it can be covered with θ. K means the amount of data used when detecting an outlier. When K is too small, the data that can be referred to by the outlier detection method are reduced, so the detection of outliers tends to be unstable. In contrast, if K is too large, it is too stable, making it difficult to detect outliers. However, this stability can also be adjusted with the threshold $\theta\,(0 \leq \theta \leq 1)$ of the outlier judgment formula $\sigma^2_{(i+1)}/\sigma^2_{(i)} < \theta$. As θ approaches 1, the outlier is judged loosely, and as it is closer to 0, the outlier is judged strictly. We decided to use three days' data for outlier detection ($K = 432$), and θ is now set to 0.98 after trial and error.

VII. Performance Evaluation

In this section, we evaluate the performance of our *GLASSO* engine. First, to improve the performance of the *GLASSO* engine, we exclude packets that are outside our interests at the preprocessing stage without estimating the cooperation between source hosts. Next, we actually operated the *GLASSO* engine in real time and analyzed the detected alert results. Finally, we evaluated the performance of malware activity detection and described the details of the detected malware activities. Here, although there is a possibility that a slight deterioration in performance may have occurred compared with the batch mode due to online processing, the method that cannot be detected in real time has no practicality and there is no fundamental difference in the method, so the comparative evaluation is omitted.

A. Data Preprocessing

We pre-exclude destination TCP ports that constantly receive a large number of packets and source hosts for a long time (e.g., more than one week). In addition, packets addressed to TCP ports that are intensively observed as alerts are excluded. If such packets are included in the model learning of the *GLASSO* engine, they will massively increase the indicator of cooperation among the source hosts to be estimated, and there is a concern that smaller but essential cooperation may be overlooked. Such packets that have an adverse effect on model learning and that are not interested should be regarded as noise and excluded. In addition, the number of source hosts is reduced to some extent, which leads to the shortening of processing time. This preprocessing should take place on a regular basis.

B. Real-time Operation of GLASSO Engine

In this section, the results of the *GLASSO* engine, which operated for one month in October 2018, are shown. The input parameters of the *GLASSO* engine used for the operation were set to $T = 600$ *(sec.)*, $M = 12, K = 432, \theta = 0.98, r \in R(= \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\})$. We considered only TCP-SYN packets, and TCP ports 22, 23, 80, 81, 445, 2323, 3389, 5431, 5555, 8080, 50382, 50390, and 52869 were excluded in the preprocessing phase. These TCP ports constantly received a large number of packets from various source hosts before September to October 1, 2018. In addition, although the IPv4 address of a source host is 32 bits, in this experiment, the upper 16 bits of the IPv4 address is regarded as one IP address, and the lower 16 bits are combined into one in order to reduce the number of source hosts N. We used eight different darknet sensors and operated the *GLASSO* engine in real time for each sensor. This darknet sensor observed that with different IP address blocks, the IP address observation size and source country were also different. As a result of the operation, a total of 1,634 alerts were obtained in real time without delay. Table I shows the number of observed IP addresses and alerts for each of the eight darknet sensors.

TABLE I
Number of Observed IP Addresses and Alerts for Each Darknet Sensor

Sensor	#Observed IP Address	#Alerts	Sensor	#Observed IP Address	#Alerts
A	29,182 (/17)	122	E	8,188 (/19)	198
B	14,593 (/18)	199	F	16,384 (/18)	115
C	4,098 (/20)	146	G	2,044 (/21)	118
D	4,096 (/20)	460	H	2,045 (/21)	276

C. *Analysis of Alert Result*

By analyzing the alerts obtained from the *GLASSO* engine for one month in October 2018 for each TCP port, a total of 128 TCP ports were obtained among 1,634 alerts, and these were broadly classified into the following three types.

1) Cyberattack: a network scan that indiscriminately attempted attacks and intrusions through vulnerable TCP ports in order for multiple hosts already infected with malware to search for the next infection target.
2) Survey scan: a network scan that indiscriminately attempted surveys and research on TCP ports using multiple hosts by organizations such as Shodan, Censys, etc.
3) sporadically-focused traffic: a phenomenon in which packets were suddenly concentrated from multiple source hosts to one darknet destination IP address and a TCP port for some reason.

Please note that multiple source hosts in an alert cooperated or coordinated with each other anomalously, and the following describes how the alerts were divided into three types as above.

A sporadically-focused traffic type means packets were concentrated from many source hosts to a specific destination IP address, clearly differing from the other two types that scan a wide range of destination IP addresses. In other words, there was no inclusive relationship between the sporadically-focused traffic type and the other two types. When both of the following two ratios were greater than 70% in the alerts, the alert was judged as sporadically-focused traffic: (1) The ratio of the number of packets of the destination most received and the total number of packets, (2) The ratio of the number of source hosts of destinations where the source host was most frequently observed and the total number of source hosts. As a result of applying this criterion to all alerts, 95 out of a total of 1,634 alerts were found to be sporadically-focused traffic, and 81 out of a total of 128 alerts were divided into sporadically-focused traffic.

At this time, we could not ascertain with our darknet precisely what sporadically-focused traffic was intended traffic, and we could only make weak guesses that there was a possibility of a routing mistake. However, since this paper covered the detection of malware activities, and sporadically-focused traffic did not seem to be a campaign originating from malware, sporadically-focused traffic was not considered.

Next, it could be assumed that 1,539 remaining alerts (47 remaining TCP ports) were network scans since many source hosts sent packets to a specific TCP port of many destination IP addresses cooperatively. There were two kinds of network scans of SYN packet reaching the darknet: cyberattacks and survey scans. In the case of an alert by survey scanners, since the size of the source hosts tended to be smaller than an alert by cyberattacks, we focused on the size of the source hosts and considered dividing it into cyberattacks and survey scans. In many cases, a survey scanner could be found from the HTTP connection and a hostname obtained by a reverse lookup of a source host IP address. If there was a high proportion of survey scanners among the alert's source hosts, that alert was considered to be an alert by survey scanners.

As a result of examining alerts in which the survey scanners ratio exceeded 50%, we decided that an alert whose size of source host was smaller than 20 was a survey scan; otherwise, it was a cyberattack. Here, in the case of cyberattacks, the ratio of a survey scanner was about 20% at most. Finally, we divided remaining alerts into 57 survey scans (16 TCP ports) and 1,482 cyberattacks (31 TCP ports). To facilitate analysis, in this evaluation, all alerts for the ports that were obtained as cyberattacks even once were all counted as cyberattacks. It means some of the alerts classified as cyberattacks included survey scans. Table II shows the result of dividing all alerts obtained during October 2018 into three types for each TCP port.

D. *Performance Evaluation of Malware Activity Detection*

From the previous section, it was possible to divide 31 attacked TCP ports out of all *GLASSO* engine alerts in October 2018. In this section, we measured the detection performance of how many of the 31 TCP ports were correct. In order to confirm the correct/incorrect answer, we manually created an answer table of attacked TCP ports in our darknet in October 2018 as far as we knew in December 2018. We comprehensively looked at a range of information such as information contained in SYN packets, vulnerability reports, threat intelligence services, security reports, and we determined the correct answer.

The results of the answers are shown in Table III. Table III shows the TCP port information of true positive, false negative, and false positive according to three types of cyberattacks. This limited answer table recorded only definitely attacked TCP ports, that is, unknown attacked TCP ports may remain. Accordingly, the true negative cannot be evaluated. As a result, we obtained 31 true positives, 3 false negatives, and 0 false positives of TCP ports. There was one attacked TCP port detected only by the *GLASSO* engine in true positives (TCP port number 1701). It was a cyberattack event that we missed due to human error when creating the answer table. As shown in Table IV, the performance of malware activity detection in October 2018 of the *GLASSO* engine had 91.2% accuracy, 100% precision, 91.2% recall, and 95.4% F-measure. Please note that unknown attacks will be clarified in the future, and false negatives may increase. In addition, although the usual formula for accuracy is (#TP+#TN)/(#TP+#FP+#FN+#TN), we do not know the true negative, so it matches the formula for usual accuracy when #TN is 0.

TABLE II
Result of Dividing All Alerts of the *GLASSO* Engine in October 2018 into Three Types for Each TCP Port

Alert Type	TCP ports (Number of Alerts, First Detected Date)
Cyberattack (1,482 Alerts) (31 Ports)	21(13, 13:40 20th), 82(56, 10:10 7th), 83(9, 20:00 11th), 84(8, 00:30 12th), 85(25, 11:40 6th), 88(100, 18:20 1st), 110(3, 14:50 17th), 443(143, 19:30 12th), 1701(1, 04:30 9th), 2480(4, 20:10 14th), 5358(309, 21:30 24th), 5379(27, 13:50 31st), 5431(26, 10:20 3rd), 5900(2, 21:40 31st), 5984(3, 03:20 20th), 6379(4, 18:20 26th), 7379(25, 13:30 31st), 7547(27, 09:50 20th), 8000(78, 21:40 5th), 8001(47, 22:40 5th), 8081(267, 21:00 10th), 8088(7, 06:10 2nd), 8181(69, 01:30 1st), 8291(17, 14:40 5th), 8443(47, 02:40 20th), 8888(31, 20:30 5th), 9000(11, 01:30 2nd), 23023(5, 07:20 14th), 37215(100, 01:30 1st), 49152(11, 01:10 14th), 65000(7, 03:00 14th)
Survey Scan (57 Alerts) (16 Ports)	17(1, 21:00 31st), 53(12, 01:10 20th), 102(6, 18:12 12th), 111(6, 00:00 27th), 990(1, 21:00 28th), 1900(4, 16:00 28th), 3128(1, 18:20 26th), 3780(2, 21:00 25th), 4567(1, 05:40 28th), 5000(2, 01:30 31st), 5357(1, 17:30 31st), 5560(1, 10:30 30th), 7657(1, 16:00 25th), 9200(2, 22:40 28th), 9981(1, 02:30 26th), 11211(15, 00:50 25th)
Sporadically Focused Traffic (95 Alerts) (81 Ports)	99(1, 21:00 25th), 139(3, 14:00 28th), 321(1, 17:20 26th), 792(1, 20:40 25th), 1678(1, 20:10 28th), 1859(1, 19:20 25th), 3227(1, 23:00 24th), 3407(1, 19:30 27th), 4466(5, 19:40 31st), 5601(1, 17:00 27th), 5777(1, 20:00 28th), 6821(1, 04:40 27th), 7199(1, 00:10 27th), 8096(1, 22:30 31st), 8185(1, 19:20 28th), 8983(1, 04:10 31st), 10994(1, 14:10 30th), 11647(1, 01:11 31st), 11876(1, 10:00 25th), 12385(1, 18:40 28th), 13750(1, 20:20 31st), 13804(1, 06:20 26th), 14401(1, 17:20 31st), 16964(1, 13:20 25th), 17396(1, 15:00 28th), 17502(1, 06:50 20th), 19533(1, 05:10 26th), 20340(1, 23:30 26th), 20382(1, 19:30 31st), 20405(1, 21:40 28th), 21221(1, 03:30 27th), 21490(1, 21:00 27th), 22063(1, 01:10 26th), 24357(1, 14:40 26th), 25024(1, 17:50 25th), 25476(1, 05:20 28th), 26137(1, 01:30 29th), 26644(1, 22:40 27th), 26934(1, 23:40 24th), 27200(1, 14:50 14th), 27910(1, 14:20 20th), 29217(1, 17:10 25th), 31632(1, 19:20 29th), 34149(1, 01:00 30th), 35669(1, 17:10 30th), 35927(1, 23:10 25th), 36064(1, 16:20 26th), 36678(1, 06:10 25th), 37822(1, 20:40 25th), 38718(1, 05:30 29th), 39420(1, 00:40 28th), 40500(4, 14:40 27th), 41939(1, 19:30 26th), 43160(6, 12:00 28th), 43361(1, 21:40 29th), 43566(1, 08:30 30th), 46928(1, 17:30 30th), 47149(1, 18:40 20th), 48449(1, 14:40 25th), 49328(1, 05:40 25th), 49516(1, 02:00 25th), 52204(1, 20:20 27th), 52854(1, 15:30 30th), 53518(1, 19:10 29th), 53557(1, 02:40 20th), 55186(1, 08:20 26th), 56011(1, 00:50 21st), 56409(1, 20:50 27th), 56499(1, 21:10 24th), 57343(1, 20:50 29th), 57762(1, 07:10 26th), 59751(1, 00:50 28th), 60850(1, 22:30 24th), 60917(1, 10:20 20th), 60928(1, 05:10 31st), 62627(1, 14:40 29th), 63591(1, 13:50 26th), 63918(1, 01:00 26th), 65032(1, 18:00 29th), 65165(1, 01:20 28th), 65238(1, 14:10 28th)

TABLE III
Checking Answers Using Limited Answer Table According to Types of Cyberattack (TCP port)

Attack Type	True Positive	False Negative	False Positive
IoT Malware	82,83,84,85,88,2480, 5358,5984,7547,8000, 8088,8443,8888,9000 (14 TCP ports)	444,8010 (2 TCP ports)	None
Router Vulnerability	21,110,443,5431, 8001,8081,8181,8291, 23023,37215,65000 (11 TCP ports)	None	None
Other Vulnerability	1701,5379,5900, 6379,7379,49152 (6 TCP ports)	2004 (1 TCP port)	None
Total	#TP=31	#FN=3	#FP=0

TABLE IV
Performance of Malware Activity Detection of *GLASSO* Engine

Accuracy $(= \frac{\#TP}{\#TP+\#FP+\#FN})$	Precision $(= \frac{\#TP}{\#TP+\#FP})$	Recall $(= \frac{\#TP}{\#TP+\#FN})$	F-measure $(= \frac{2\#TP}{2\#TP+\#FP+\#FN})$
91.2%	100%	91.2%	95.4%

E. Details of Detected Malware Activity

We discuss the details of detected malware activities according to three attack types: IoT malware, router vulnerabilities, and other vulnerabilities. There are two types of malware used for cyberattacks: (1) malware that forms a botnet and receives a command from a C2 server then executes a network scan, and (2) malware such as worms, computer viruses, etc. that spreads network scanning on a wide area to self-propagate.

1) IoT Malware: The true positive 14 TCP ports of IoT malware type in Table III were roughly divided into three types of malware: Mirai, Hajime, and HNS (Hide and Seek). First, the Mirai scanned many web-based TCP ports of 7547, and a series of 80, 8000 and spread more and more to scan new web-based TCP ports. Next, the Hajime regularly scanned TCP ports 5358 and 9000. Finally, the HNS scanned TCP ports 23, 80, 8080, 2480, 5984, and random port [11].

2) Router Vulnerability: The true positive 11 TCP ports of router vulnerability in Table III were divided into malware activities against vulnerabilities presented in the router products of five manufacturers [8], [12], [13]. First, vulnerabilities existed in the router product of manufacturer A, and as a result of infecting many routers with malware, network scans were performed by many routers within a short period (TCP ports 21, 110, 443, 8291, 23023, and 65000). Next, other network scans with the features of Mirai were observed from the infected routers using the vulnerability of manufacturers B, C, and D's router products (TCP ports 8001, 8081, 8181, and 37215). Finally, using the vulnerability of manufacturer E's UPnP (Universal Plug and Play), many router products using manufacturer E's UPnP were hijacked, and network scans were observed from the infected routers (TCP port 5431).

3) Other Vulnerability: The true positive 6 TCP ports of other vulnerability in Table III were roughly divided into malware activities against vulnerabilities against four application services [9]. First, a vulnerability existed in a NoSQL database service, and network scans for searching that database were observed (TCP ports 5379, 6379, 7379). In addition, network scans for searching services such as L2TP VPN (Layer 2 Tunneling Protocol Virtual Private Network), VNC (Virtual Network Computing), and Supermicro BMC (Baseboard Management Controller) were observed (TCP ports 1701, 5900, 49152).

Many of the malware activities in October 2018 were attacks using known malware or known vulnerabilities that were observed continuously or regularly before that time. In such a case, it is not meaningful to discuss whether or not the timing of detecting malware activities in October 2018 was appropriate. However, it was meaningful to evaluate the detection timing for malware activities that showed a new trend in October 2018. In that sense, it was a new trend of malware activities in October 2018 that manufacturer A's infected router product launched scannings of several TCP ports. At this point, the *GLASSO* engine detected these scan campaigns all at an appropriate time when the number of source hosts peaked in.

VIII. Consideration of False Negative

In this section, we consider the false negative this time. Scans with the features of Mirai were observed of TCP port numbers 444 and 8010, and scans for searching the CMS (Content Management System) service of TCP port number 2004 were observed. Looking at the number of source hosts for 10 minutes by each darknet sensor, 161 source hosts were observed when TCP port 8010 was at the peak, while TCP ports 444 and 2004 were observed with 19 and 23 source hosts at most.

First of all, the reason why we missed TCP port 8010 is because of handling only TCP ports with the largest number of source hosts in outlier time slot datasets as alerts. TCP port 8010 can be easily detected if it is considered as an alert, even for the TCP port that has the second and the subsequent largest number of source hosts. However, this measure may increase the number of false positives.

Next, if the number of source hosts is small, its cooperation will become thin, and it will be difficult to capture campaigns like TCP ports 444 and 2004 by the *GLASSO* engine. As a countermeasure for such false negatives, considering a method that can increase the number of source hosts and a method that can capture the cooperation between a small number of source hosts, the following three points are conceivable.

1) Do not combine the lower 16 bits of IPv4 address.
2) Use a larger IP address size of darknet sensors.
3) Perform TCP port exclusion in the preprocessing often.

IX. Conclusion

The proposed *GLASSO* engine, unlike our earlier work [7], makes it possible to detect new malware activities in real time and thus enables us to respond to situations more quickly. Various evaluations have been conducted to demonstrate the effectiveness of the engine. We clarified that the engine produced alerts for three types of activities, i.e., cyberattack, survey scan, and sporadically-focused traffic, and that they can be properly distinguished by preparing the proper criteria. We made limited ground truth by manual analysis, with which we evaluated the malware activity detection performance of the engine and analyzed the details of the detected activities. Moreover, the engine detected anomalous events that could not have been detected using the conventional method. Although the engine suffers from some false negatives, we believe that these could be alleviated by preparing the environment properly. Accordingly, we conclude that the proposed *GLASSO* engine can automatically and precisely ascertain indiscriminate cyberattacks like network scans in real time. Datasets which we used in this paper are available online at https://csdataset.nict.go.jp/darknet

This work is to be improved further. First, the *GLASSO* engine could be optimized further to reduce false negatives. Second, pilot operation with the engine should also be conducted to evaluate its practical effectiveness over a long period. Third, the phenomenon we call sporadically-focused traffic remains unclear; further investigation of this phenomenon should be conducted using another dataset generated from the honeypot in addition to the darknet. Finally, since it was found that the *GLASSO* engine can detect not only cyberattacks but also survey scanners, we would like to detect survey scanners as an extension study. Through this work, we hope to streamline security operations and reduce the burden of network security operators and incident response teams.

Acknowledgment

The authors wish to thank Associate Prof. K. Yoshioka from Yokohama National University and Prof. N. Murata from Waseda University for their valuable comments. This research was partially supported by JSPS KAKENHI Grant Number 18H03291.

References

[1] T. Ban, L. Zhu, J. Shimamura, S. Pang, D. Inoue, and K. Nakao. Detection of botnet activities through the lens of a large-scale darknet. In *International Conference on Neural Information Processing, Springer*, 2017.

[2] A. Dainotti, K. Benson, A. King, K. Claffy, M. Kallitsis, E. Glatz, and X. Dimitropoulos. Estimating Internet address space usage through passive measurements. *ACM SIGCOMM Computer Communication Review*, 44(1):42-49, 2013.

[3] Z. Durumeric, M. Bailey, and J.A. Halderman. An Internet-wide view of Internet-wide scanning. *23rd USENIX Security Symposium*, pp.65-78, 2014.

[4] C. Fachkha, E. Bou-Harb, A. Keliris, N. Memon, and M. Ahamad. Internet-scale probing of CPS: inference, characterization and orchestration analysis. In *Proceedings of NDSS*, 2017.

[5] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3), 2008.

[6] J. Friedman, T. Hastie, and R. Tibshirani. Graphical Lasso: Estimation of Gaussian Graphical Models. https://cran.r-project.org/web/packages/glasso/glasso.pdf, [Accessed Dec. 2018].

[7] C. Han, K. Kono, S. Tanaka, M. Kawakita, and J. Takeuchi. Botnet detection using graphical lasso with graph density. In *International Conference on Neural Information Processing, Springer*, 2016.

[8] Huawei, Security Notice - Statement on Remote Code Execution Vulnerability in Huawei HG532 Product. https://www.huawei.com/en/psirt/security-notices/huawei-sn-20171130-01-hg532-en, [Accessed Dec. 2018].

[9] Imperva, RedisWannaMine Unveiled: New Cryptojacking Attack Powered by Redis and NSA Exploits. https://www.imperva.com/blog/rediswannamine-new-redis-nsa-powered-cryptojacking-attack/, [Accessed Dec. 2018].

[10] T. Ide, A.C. Lozano, N. Abe, and Y. Liu. Proximity-Based Anomaly Detection Using Sparse Structure Learning. In *Proceedings of 2009 SIAM International Conference on Data Mining*, 2009.

[11] Netlab 360, HNS Botnet Recent Activities. https://blog.netlab.360.com/hns-botnet-recent-activities-en/, [Accessed Dec. 2018].

[12] Netlab 360, BCMPUPnP_Hunter: A 100k Botnet Turns Home Routers to Email Spammers. https://blog.netlab.360.com/bcmpupnp_hunter-a-100k-botnet-turns-home-routers-to-email-spammers-en/, [Accessed Dec. 2018].

[13] Netlab 360, 7,500+ MikroTik Routers Are Forwarding Owners' Traffic to the Attackers, How is Yours?. https://blog.netlab.360.com/7500-mikrotik-routers-are-forwarding-owners-traffic-to-the-attackers-how-is-yours-en/, [Accessed Dec. 2018].