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. {tak}@inf.kyushu-u.ac.jp
‡clwit Inc., Japan. {shimamura}@clwit.co.jp

Abstract—**Recent malware evolutions have rendered cyber space less secure, and we have witnessed increasing amount of severe security incidents in these days. To minimize the impact of malware activities, it is important to promptly and precisely detect them. We have been working on this issue by monitoring traffic coming into unused IP address spaces, i.e., a darknet. On our darknet, Internet-wide malware scans are observed as if they are coordinated or cooperatively working. Based on this observation, our earlier method monitored network traffic arriving at our darknet, estimated cooperativeness between each pair of the source hosts, and detected drastic changes of the cooperativeness among source hosts as the signal of newly activated malware activities. However, the method cannot work in real-time and thus 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 malware activity detection, Darknet traffic analysis, Outlier detection, Cooperativeness.**

I. Introduction

The number of cyber attacks has been growing, and they often use malware as their tool. To minimize the impact of cyber attacks, it is important to quickly and precisely grasp actual behaviors of malware on the Internet. Many of 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 it is costly and human error may occur. Therefore, a method that can automatically, quickly, and precisely detect activities of malware of all types are needed in present days.

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 a type of such attacks, and malware often performs this 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., darknet. To capture these scans, we set up receivers (sensors) in our darknet and captured all the traffic arriving at the sensors. Through the analysis of the traffic, we intend grasp what kind of network service was being targeted and the trend of global cyber attack,

The difficulty lies in how we should distinguish the ill-intentioned packets from others; the darknet traffic includes not only ill-intentioned scans but also legitimate scans with research purposes and unintentionally sent packets due to device misconfiguration. In this study, we work on a mechanism that automatically detects indiscriminate attacks reaching darknets like network scans from malware. The mechanism detected cyber attacks 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 cooperatively working. In reality, we believe each infected host send scans independently, though there are still minor cases where command-and-control (C2) servers coordinate bots. Based on this observation, our earlier method [7], called "*GLASSO* engine[1]," estimates the cooperativeness of the source hosts by using a sparse structure learning algorithm, called "Graphical Lasso" [5]. It also provides alerts when the estimated cooperativeness drastically changes; it can detect a campaign or a group of related activities by estimating the cooperativeness among many pairs of source hosts for each unit time and by finding outlier of cooperativeness among different unit times. This approach is robust to noise: packets that accidentally arrive at our darknet space due to misconfigurations will not show cooperativeness. Moreover, this approach ignores activities with weak cooperativeness.

However, since the *GLASSO* engine in [7] works in a batch mode and requires 3-day darknet traffic to process, the analysis result will not be provided in real time and will be delayed by more than 3 days. Therefore, in this paper, we propose an online processing algorithm of *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 to demonstrate its feasibility and effectiveness in terms of detecting the rise of new malware activities in real-time. For this evaluation, we

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

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 *GLASSO* engine and demonstrate its feasibility and usefulness.

Contribution. This paper offers the following contributions:

1) We proposed an online processing algorithm of *GLASSO* engine and made an 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 got detection results without delay. Also, we analyzed detection results by destination TCP ports (by services). (discussed in Section VII-B, C)
4) We evaluated the detection accuracy of malware activities using the limited ground truth. This is the first and practical evaluation of the detection accuracy of malware activities on darknet, and we explained the details of detected malware activities. Finally, we considered some countermeasures against several malware activities that GLASSO engine missed. (discussed in Section VII-D, E and VIII)

II. Background

In this section, the overview of our fundamental tools, i.e., a darknet, graphical Gaussian model, and graphical lasso algorithm, are presented.

A. Darknet

The Internet can be categorized into two types of networks: livenet and darknet. A livenet is a network of a busy IP addresses, while a darknet is a network of unused IP addresses. We 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 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 indiscriminately tried attacks and intrusions by malware through vulnerable TCP ports is called a cyber attack, and a network scan that indiscriminately tried surveys and researches by organizations like Shodan and Censys is called a survey scan.

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 correlation between random variables is included. To cope with the issue, there is a method of obtaining a precision matrix Σ^{-1} from which conditional independence of a pair of random variables can be measured [10]. If and only if when $\Sigma^{-1}_{ij} = 0$, 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 a matrix element of Σ^{-1} is zero, there is no edge between the nodes, if a matrix element is non-zero, there is edge between the nodes. In other words, the graph in GGM using this precision matrix show conditional independence of all pairs of random variables.

C. Graphical Lasso

The precision matrix Σ^{-1} is the inverse matrix of the sample covariance matrix S. We expect the precision matrix Σ^{-1} to be a sparse matrix such that variable pairs with essential dependencies take nonzero values and perhaps weakly related variable pairs with noise take zero. In general, however, the elements of the sample covariance matrix S cannot be strictly zero, and the precision matrix Σ^{-1} is also not generally sparse. Therefore, the "graphical lasso" which is a sparse structure learning algorithm optimizes a precision matrix by solving a maximum likelihood equation with ℓ_1 regularization term in one column (one row) and estimates a sparse precision matrix $\hat{\Sigma^{-1}}$ without explicit inverse matrix calculation [5]. The input arguments of the graphical lasso algorithm are the sample covariance matrix S and the ℓ_1 regularization coefficient $r \in \mathbb{R}$ (≥ 0). Here, r is a threshold to decide 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 express conditionally independent and more essential dependencies of all variable pairs.

III. Related Work

Many studies using darknet had been carried out, and showed its usefulness on 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 support census-like analyses of IP address space utilization [2]. Durumeric *et al.* analyzed a large-scale darknet to investigate scanning activities, and identified patterns in large horizontal scanning operations [3]. Also, they presented an analysis of the latest network scanning on the overall landscape, and 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 ample CPS protocols by correlating and analyzed 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. However, the dataset used in this abrupt-change detection algorithm processed traffic with only one destination TCP port. Since the *GLASSO* engine processed the entire live traffic without restriction, the range of the dataset was different from the abrupt-change detection algorithm. Actually, research using a 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, presented in [7], that uses GGM to analyze darknet traffic and discusses its shortcomings.

A. Applying GGM to Darknet Traffic

The *GLASSO* engine took a time pattern of the number of packets received from each source host as a variable and applied GGM to capture the dependency between variables (source hosts). Although this variable was never expressed as a Gaussian distribution, it could be assumed that there were many variables close to the form of a log-normal distribution, so it approximated to some extent a Gaussian distribution by log-transformation. Also, if the variables could not be completely complied with a Gaussian distribution but could be approximated to some extent, dependency relationships between the variables in the GGM could be grasped.

First, we considered how to process the dataset for darknet traffic. Darknet traffic for T seconds used for one model learning was called a time slot. We suppose there were N unique source hosts in a time slot t. A time series data was generated by counting the number of packets observed at a certain sampling interval for every source hosts. Here, if the number of time series samples was M, the sampling interval was T/M *(sec.)* Then, we converted from a time slot t to data matrix D.

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

Here, $\boldsymbol{x}^{(m)}$ meant N-dimensional variables of the number of samples of M, and $x_n^{(m)}$ represented the number of packets at the m-th point of the n-th source host. Since log-transformation could not be performed when $x_n^{(m)} = 0$, it was converted to an appropriate value $x_n^{(m)} = 0.1$. Also, $\mathbb{N}_0 = \{0, 1, 2, \cdots\}$.

Next, we obtain a precision matrix from the data matrix $\boldsymbol{D}_t$ using the graphical lasso algorithm and apply GGM. Then, a set of source host (variable) corresponds to a node set of a graph in the GGM, and a presence or absence of a dependence relationship of a source host pair (variable pair) corresponds to an edge set. The cooperativeness of host pairs meant that there was no cooperativeness with that host pair when the time pattern of the number of packets received from each source host was conditionally independent between two hosts.

B. Algorithm of Batch-mode GLASSO Engine

We prepared darknet traffic observed over a long period of time (e.g. 3 days traffic) and divided traffic every T seconds to create multiple time slots. At that time, we consider only SYN packets, TCP. Next, a data matrix $\boldsymbol{D} \in \mathbb{R}^{M\times N}$ was created for each time slot, and a sample covariance matrix $S \in \mathbb{R}^{N\times N}$ was obtained. The sample covariance matrix S and a positive real number r were input to the graphical lasso algorithm to obtain a sparsely estimated precision matrix $(\hat{\Sigma^{-1}})^{(r)} \in \mathbb{R}^{N\times N}$. Here, we tried with some positive real numbers like $r \in R(= \{r_1, r_2, \cdots, r_s\} \in \mathbb{R}^s \; (\geq 0))$. From the estimated precision matrix, an undirected graph $G = \{V, E\}$ in GGM could be represented by node set $V = \{x_1, \cdots, x_N\}$ and edge set $E = \{(i, j)|\Sigma_{ij}^{-1} \neq 0\}$. Then, in order to express the degree of cooperativeness between all pairs of source host in each time slot with scalar values, we obtained a graph density value $d^{(r)} = |E|/N(N-1)$ for each time slot. The graph density value represented the ratio of the actual number of edges to the number of edges of the complete graph. Finally, time slots corresponding to graph densities showing anomalously high values compared to all graph density values were determined using outlier detection techniques.

C. Defects of Batch-mode GLASSO Engine

The batch mode *GLASSO* engine could detect anomalies of cooperativeness between source hosts and showed its usefulness, but it was necessary to prepare 3 days' darknet traffic data and processed at once. In other words, considering until the processing time, it would be delayed by 3 days or more until the result output, and the expected effect leading to prompt response would be low. Also, although there were multiple parameters in the *GLASSO* engine, it had not been evaluated on what basis the parameters were set. Finally, only a few results of outlier detection was mentioned as case studies and there was no evaluation using a ground truth; the verification of the method was insufficient. Therefore, in the rest of this paper, we would improve the defects mentioned above.

V. *GLASSO* Engine (Online-mode)

In this section, we propose a novel *GLASSO* engine that analyzes darknet traffic in real time. The engine contains online processing algorithm and alert judgment method. The input of the engine is 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 slots classified as outliers. The alert information includes the time stamp, targeted destination TCP port numbers, source IP addresses of the packets sent to the destination TCP ports, and the number of the 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.

A. Online Processing Algorithm

We consider a method to output alerts (outliers) sequentially by processing every time slot, instead of processing multiple time slots at once. First, the newest time slot t is processed in

Algorithm 1 The *GLASSO* Engine with Online Processing

```
Input: $t$ (a time slot), $M, r \in R(= \{r_1, r_2, \cdots, r_s\}), \boldsymbol{d}^{(r)}, K, \theta$
Output: alerts or none
 1: for a time slot $t$ is updated newly do
 2:     preprocess a time slot $t$
 3:     make $\boldsymbol{D}_t$ from a time slot $t$
 4:     compute $S$ from $\boldsymbol{D}_t$
 5:     for $r$ in $R$ do
 6:         compute $(\hat{\Sigma^{-1}})_t^{(r)}$ using the graphical lasso (input: $S, r$)
 7:         compute $d_t^{(r)}$ from $(\hat{\Sigma^{-1}})_t^{(r)}$
 8:         add $d_t^{(r)}$ to $\boldsymbol{d}^{(r)}$
 9:         if $length(\boldsymbol{d}^{(r)}) = K$ then
10:             run alert judgment method (Algorithm 2)
11:             if there are outliers then
12:                 collect alert information from outliers
13:                 output alerts
14:                 remove outliers from $\boldsymbol{d}$
15:             else
16:                 remove the most old time slot from $\boldsymbol{d}^{(r)}$
17:             end if
18:         end if
19:     end for
20: end for
```

the same manner as Section IV-B to obtain a graph density value $d_t^{(r)}$. Then, $d_t^{(r)}$ 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, wait until the next time slot updated without performing the alert judgment. Detailed alert judgment method is explained in a next section. The time slot judged as an alert is output as an alert and is deleted from $\boldsymbol{d}$ so that it will not be referred to in subsequent alert judgment. Also, if no time slot is determined as an alert, delete the oldest time slot from $\boldsymbol{d}$. When the time passes and a new time slot is updated, by repeating the above steps, it is possible to sequentially process *GLASSO* engine while keeping the number of time slots used for alert judgment. As a result, it obtains the result in real-time with a very short time compared to the batch-mode since only an updated time slot is processed. The pseudo code for *GLASSO* engine with online processing is described in the Algorithm 1. Here, *length*() function get the length of vectors.

B. Alert Judgment Method for Online Processing

In this section, we propose an outlier detection method for discriminating alerts (outliers) from a sequence of graph density values $\boldsymbol{d}$ during online processing. It is checked how much the largest element occupies a whole sample variance in the sequence of graph density values $\boldsymbol{d}$, and when a ratio is large, the largest element is judged to be an outlier. The pseudo code of the alert judgment method is indicated in the 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. Also, *order*() function returns a

Algorithm 2 Pseudo code for Alert Judgment Method

```
Input: $\boldsymbol{d} \in \mathbb{R}^K, K, \theta$
Output: outliers or none
 1: $i \leftarrow 0$
 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:         $outliers \leftarrow order(\boldsymbol{d}, decreasing = True)[1 : i]$
 8:         return outliers
 9:     end if
10: end while
```

permutation which rearranges its first argument into ascending or descending order, *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 how these five parameters are set in this section. These parameters are determined by an empirical heuristic method.

A. Length of Time Slot T

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. A computational complexity of the *GLASSO* engine depends greatly on the number of source hosts N. As T becomes longer, the number of source hosts generally also increases, a processing time increases exponentially, and a real-time processing becomes difficult. Next, we consider a lower bound. It is sufficient if we can make rich observations such that all one campaign of 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 in about 5 minutes (subnet /17). Therefore, we set to $T = 600$ *(sec.)* which can process in real-time without problems and make rich observations.

B. The Number of Time Series Samples M

The number of time series sample 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, as the sampling interval is severely divided, a time pattern of the number of packets is measured more finely, and learning with good precision is performed. However, if the sampling interval is too short, there is a risk of estimating that there is no dependence between source hosts that originally work in cooperation. On the contrary, if the sampling interval is too long, there is a fear that it may be estimated there is cooperativeness between any source hosts. Therefore, it is necessary to set M to an appropriate number, it is difficult to find the optimum value. But fortunately, the regularization coefficient r 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 pattern of the number of packets, so we set $M = 12$.

C. Regularization Coefficient r

The regularization coefficient $r \in \mathbb{R}\ (\geq 0)$ is an input parameter used in the graphical lasso algorithm. It is a threshold to decide how much dependence is considered to be derived from noise, and adjusts a sparsity of an estimated precision matrix Σ^{-1}. Here, it is similarly meaningful between scrape down weakly related dependencies and more severely measure a time pattern of the number of packets. Therefore, r and M have similar roles. As a feature of r, $d^{(r)}$ generally gets closer to 1 as r gets closer to 0, and $d^{(r)}$ gets closer to 0 as r gets bigger. Also, 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 a calculation time is shortened. Finally, r can be tried multiple times with $R(= \{r_1, r_2, \cdots, r_s\} \in \mathbb{R}^s)$, but a 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 the decimal point six digits. Even when trying with varying values from 6 decimal places to 2 digits, the graph density value did not change significantly. For example, our experience shows that time slots that are alerted with $r = 0.55$ are judged alert by r =0.5 or 0.6 in all cases. From this, we find out that it is sufficient to try while changing the value with one decimal place. Also, there are many zero matrices from $r \geq 1$, which often result in $d^{(r)} = 0$. And processing time takes longer for $r < 0.4$, that a sufficiently sparse precision matrix is not estimated, in most cases outliers do not come out. From the above, we finally decide 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 used for alert judgment, as well as the relationship between M and r, unless K is set to an extreme number, it can be covered with θ. The K means the number of data used when detecting an outlier. When K is too small, data that can be referred to by the outlier detection method is reduced, so the detection of outlier tends to be unstable. On the contrary, if K is too big, it would be too stable and hard 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 loosely judged, and as it is closer to 0, the outlier is strictly judged. We decided to use 3 days' data for outlier detection ($K = 432$), θ is now set to 0.98 after trial and error.

VII. Performance Evaluation

In this section, we described a performance evaluation of the *GLASSO* engine. First, in order to improve the performance of the *GLASSO* engine, we considered a method of preliminarily excluding packets which were not interested in at the preprocessing stage without estimating cooperativeness between source hosts. Next, we actually operated the *GLASSO* engine in real-time and analyzed detected alert results. Finally, we evaluated a malware activity detection accuracy and described the details of detected malware activities. Here, although there was a possibility that a slight deterioration in accuracy might have occurred compared with batch mode due to the online processing, the method that could not be detected in real-time had no practicality and there was no fundamental difference in the method, so that comparative evaluation was omitted.

A. Enhance Preprocessing

We pre-excluded a destination TCP port that constantly observed packets from a large number of packets or source hosts for a long time. Such destination TCP ports could easily be noticed by anyone, and generally, those were kept observing for a while without a detection method. For the same reason, packets addressed to TCP ports intensively observed as alerts were excluded in advance. Furthermore, when such packets were included in model learning of the *GLASSO* engine, these occupied a majority of cooperativeness among the source hosts to be estimated, and there was a fear that smaller but essential cooperativeness might be overlooked. Such packets that had an adverse effect on model learning and were not interested should be regarded as noise and excluded. In addition, the number of source hosts was reduced to some extent, which led to shortening of processing time. We regularly updated such work automatically.

B. Real-time Operation of GLASSO Engine

In this section, the results of the *GLASSO* engine which operated for 1 month in October 2018 were shown. 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 SYN packets, TCP and TCP ports 22, 23, 80, 81, 445, 2323, 3389, 5555, 8080, 50382, 50390, 52869 were excluded in advance as preprocessing. These TCP ports were constantly observed a large number of packets or source hosts before October 1, 2018. Also, a source host IP address usually had one IP address up to the fourth octet, but in order to reduce the number of N, in this experiment, the IP addresses up to the second octet were taken as one source host IP address. We used 8 different darknet sensors and operated the *GLASSO* engine in real-time for each sensor. This darknet sensor observed 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 observation IP address size and the number of alerts for each of the eight sensors.

TABLE I
IP Address Size and the Number of Alerts of Each Darknet Sensor

Sensor	Size	# of Alerts	Sensor	Size	# of Alerts
A	29,182	122	E	8,188	198
B	14,593	199	F	16,384	115
C	4,098	146	G	2,044	118
D	4,096	460	H	2,045	276

C. Analysis of Alert Result

By analyzing 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) cyber attack: a network scan that indiscriminately tried attacks and intrusions through vulnerable TCP ports in order for multiple hosts already infected with malware to search for a next infection target.
2) Survey scan: a network scan that indiscriminately tried surveys and researches to TCP ports using multiple hosts by organizations such as Shodan, Censys, etc.
3) *One-dst-centralized*: a phenomenon in which packets are concentrated from multiple source hosts to one darknet destination IP address and a TCP port suddenly, due to some cause.

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

In an *one-dst-centralized* type, packets were concentrated from a number of source hosts to specific destination IP address, clearly differed from the other two types which scan a wide range of destination IP address. In other words, there was no inclusive relationship clearly between an *one-dst-centralized* type and the other two types. When both of the following two ratios were greater than 70% in the alerts, the alert was judged an *one-dst-centralized*: (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 observed and the total number of source hosts. As a result of applying this criterion to all alerts, 95 alerts out of a total of 1,634 alerts were found to be an *one-dst-centralized*, 81 out of a total of 128 alerts were divided into an *one-dst-centralized*. In this time, we could not grasp exactly with our darknet what an *one-dst-centralized* was intended traffic, only weak guesses that there was a possibility of a routing mistake. However, since this paper covered the detection of malware activities, and *one-dst-centralized* did not seem to be a campaign originating from malware; *one-dst-centralized* 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 address cooperatively. There were two kinds of network scans of SYN packet reaching the darknet: cyber attacks and survey scans. In the case of an alert by survey scanners, since a size of the source hosts tended to be smaller than an alert by cyber attacks, we focused on a size of the source hosts and considered dividing it into cyber attacks and survey scans. In many cases, a survey scanner could be found from HTTP connection and a host name obtained by 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 survey scanners ratio exceeded 50%, we divided that an alert whose size of source host was smaller than 20 was a survey scan, otherwise it was a cyber attack. Here, in the case of cyber attacks, the ratio of a survey scanner was about 20% at most. Finally, we were divided into alerts with 57 survey scans (16 TCP ports) and alerts with 1,482 cyber attacks (31 TCP ports). Table II shows a result of dividing all alerts obtained during the month of October 2018 into three types for each TCP port.

D. Evaluation of Malware Activity Detection Accuracy

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 a detection accuracy of how many of the 31 TCP ports were correct. In order to confirm the correct/incorrect answer, we made an answer table of attacked TCP ports in our darknet in October 2018, as far as we knew at the beginning of December 2018. We comprehensively looked at various information such as information contained in the SYN packet (time stamp, source IP address, source port, destination port, sequence number, window size), vulnerability report, threat intelligence service, security report, and judged the correct answer.

We used the limited answer table to check the answer of attacked TCP ports detected in real-time from the *GLASSO* engine. The results of the answers were shown in Table III. Table III shows TCP port information of true positive, false negative, false positive according to three types of cyber attack. Since this limited answer table recorded only the attacked TCP port, that is, correct answer, there was no true negative in such evaluation. As a result, we got 31 true positives, 3 false negatives and 0 false positives of TCP ports. As shown in Table IV, the accuracy of malware activity detection in October 2018 of the *GLASSO* engine was 91.2% accuracy, 100% precision, 91.2% recall, and 95.4% F-measure. Also, there was one attacked TCP port detected only by the *GLASSO* engine in true positives (TCP port number 1701). It was a cyber attack event that we missed due to human error when making the answer table. Finally, please note that unknown attacks will be clarified in the future and false negatives may increase.

E. Details of the Detected Malware Activity

In this section, we discuss the details of detected malware activities according to three attack types: IoT malware, router vulnerabilities, and other vulnerabilities. There were two types of malware used for cyber attacks: (1) malware that forms a botnet and receives a command from a C2 server and executes

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 (The Number of Alerts, First Detected Date)
cyber attack (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)
One-dst Centralized (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 CYBER ATTACK (TCP PORT)

Attack Types	True Positive	False Negative	False Positive
IoT Malware	82,83,84,85,88,2480, 5358,5984,7547,8000, 8088,8443,8888,9000 (14 TCP port)	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	31	3	0

TABLE IV
ACCURACY OF MALWARE ACTIVITY DETECTION OF THE *GLASSO* ENGINE

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

a network scan, (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 port 80 and 8000 series and were spreading more and more to scan new web-based TCP ports. Next, the Hajime regularly scanned TCP ports 5358, 9000. Finally, it is reported that the HNS scanned TCP port 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 that 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 scannings by a large number of routers were performed in a short time in several TCP ports 21, 110, 443, 8291, 23023, 65000. Next, other network scans with features of Mirai were observed from infected routers using the vulnerability of manufacturer B, C, D's router products (TCP ports 8001, 8081, 8181, 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 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), 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 constantly or regularly observed before that time. In such a case, it is not meaningful to discuss whether the timing of detecting malware activities in October 2018 was appropriate or not. 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 malware activities in October 2018 that manufacturer A's infected router product launched scanning to several TCP ports. At this point, the *GLASSO* engine detected when the number of source hosts peaked in this scan campaign, all at an appropriate time.

VIII. Consideration on False Negative

In this section, we considerate about the false negative in this time. Scans with the features of Mirai were observed to TCP port numbers 444 and 8010, and scans for searching CMS (Content Management System) service to 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 is a peak, while TCP ports 444 and 2004 were observed at most 19, 23 source hosts.

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

Next, if the number of source hosts is small, its cooperativeness will become thin and difficult to capture campaigns like TCP ports 444, 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 cooperativeness between a small number of source hosts, the following three points are conceivable.

1) Do not reduce the number of source hosts and make the IP addresses up to the fourth octet as one source host IP address.
2) Use a larger IP address size of darknet sensors.
3) Frequently perform TCP port exclusion in the preprocessing.

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 situations more quickly. Various evaluation have been conducted to demonstrate the effectiveness of the engine. We clarified that the engine produced alerts for three types of activities, i.e., cyber attack, survey scan, and *one-dst centralized*, and they can be properly distinguished by preparing 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 by the conventional method. Although the engine suffers from some false negatives, we believe it could be alleviated by properly preparing the environment. Therefore, we conclude that the proposed *GLASSO* engine can automatically and precisely grasp indiscriminate cyber attacks like network scans in real-time.

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

Acknowledgment

The authors thank associate Prof. K. Yoshioka from the Yokohama National University and Prof. N. Murata from the Waseda University for their valuable comments.

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].