

Real-time Cyber attack Detection Method based on Darknet Traffic Analysis by 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—Recently, as cyber attacks from malware are evolving more and more, serious security incidents occur frequently and making cyberspace less secure. Accordingly, studies on cyber attack detection methods that can cope with subspecies and unknown cyber attacks promptly and precisely are important. Our earlier method monitors network traffic arriving at a darknet, estimates cooperativeness of the source host pairs of this traffic, and detects outliers through anomaly cooperative traffic. One of the causes of anomaly source host-to-host cooperativeness on darknet traffic is a cyber attack using Internet-wide scanning. However, although the method is effective for cyber attack detection, it can not analyze in real-time. In this study, we propose an online processing algorithm, making it possible to detect cyber attacks in real-time. We also tune several important parameters, implement and operate the proposed method, and evaluate the performance of our method. As a result of evaluating cyber attack detection accuracy through a limited ground truth, we obtained results of 91.2% accuracy, 100% precision, 91.2% recall, 95.4% F-measure.

Index Terms—Real-time cyber attack detection, Darknet traffic analysis, Outlier detection, Cooperativeness.

I. INTRODUCTION

Cyber attacks have continued to grow and become diverse recently, and the number of variants and unknown cyber attacks is increasing. In any cyber attack, it is important to quickly and precisely grasp actual behaviors of cyber attacks on the Internet as a countermeasure. As a conventional method for dealing with actual cyber attacks quickly and precisely by network-based access control and analysis, there are signature based and whitelist/blacklist based access control method like a intrusion detection system and a firewall. In general, however, these conventional methods cannot expect to cope with subspecies and unknown cyber attacks. Also, there is a heuristic rule-based analysis and response method by experts such as network security operators, but human error can occur. For the above reasons, studies on a cyber attack detection method that can quickly and precisely deal with subspecies and unknown cyber attacks and without human error are important in recent years.

We focus on the detection of indiscriminate attacks like Internet-wide scan such as attacking and intruding to an unspecified number of Internet users. Here, we set up receivers (sensors) in our darknet—unused IP address blocks—and captured all the traffic arriving at the sensors to generate a

dataset. A lot of cyber attacks like indiscriminately scattered network scans reached the darknet. Therefore, by analyzing the darknet rather than using another network dataset, it was easy to grasp what kind of network service was being targeted and the trend of global cyber attack. However, since the darknet sensor we operate did not return a response, it was difficult for a human to manually distinguish whether the received packet was intended for cyber attack. Accordingly, in this study, we worked on a mechanism to mechanically detect indiscriminate cyber attacks reaching darknets like network scans. The mechanism detected cyber attacks quickly and precisely including subspecies and unknown cyber attacks by using an unsupervised machine learning method, so that there is no human error.

Since infected hosts (devices) forming a botnet behaved synchronously when they received commands from a command-and-control server [1], our earlier method [8] focused on cooperativeness between the source hosts. Since this method used the library of R language “glasso” [7], this method is called “GLASSO engine” from the following for convenience. The GLASSO engine estimated the cooperativeness of all pairs of source hosts from the time pattern of the number of packets received from each source host in a certain time period in darknet traffic using the sparse structure learning algorithm “Graphical Lasso” [6], [11]. The degree of cooperativeness in that time period was quantified into a numerical value and a time period in which the numerical value was anomaly high compared with other time periods was detected as an outlier [8]. It could be seen that a strongly cooperated campaign was performed from many source hosts in the time period judged as an outlier. One of the causes of anomaly source host-to-host cooperativeness on darknet traffic was a cyber attack using Internet-wide scanning, and the GLASSO engine could detect such cyber attacks. The cooperativeness of host pairs mentioned here 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. This cooperativeness could be estimated by the graphical lasso algorithm. In addition, the graphical lasso could be expected to scrape off weak cooperativeness of host pair that depend on coincidence. Therefore, this engine would not need to consider the communication due to misconfigu-

rations arriving at the darknet and the weak cooperativeness of the host pair, and it could be expected to estimate more essential cooperativeness.

Contribution. Since the *GLASSO* engine in the earlier work [8] was in a batch mode requiring 3 days' darknet traffic data to process; when considering the processing time, the result was delayed by more than 3 days until output. In other words, it could not be said that the batch-mode *GLASSO* engine could analyze darknet traffic promptly. Therefore, in this paper, we proposed an online processing algorithm of *GLASSO* engine and made outlier detection sequentially and in real-time. Also, we selected and evaluated important parameters in the *GLASSO* engine. Next, in order to shorten processing time and to be able to process more scalable, we devised at the stage of preprocessing. Then, the *GLASSO* engine was operated in real-time using live darknet traffic (raw packets) without delay, and the detection result was evaluated by destination TCP ports (by services). In general, it was difficult to distinguish whether a packet arriving at the darknet was intended for a cyber attack. Therefore, it was difficult to give out the ground truth of cyber attacks in the darknet, but for the evaluation, we made a correct answer table in the range we understood and evaluated detection result using limited ground truth. As a result, it was found that the number of true positive TCP ports was 31, the number of false negative TCP ports was 3, and the number of false positive TCP ports was 0, and it was found that the *GLASSO* engine detected the following network scan in the darknet: A network scan that distributed scanning from infected botnet hosts like Mirai, Hajime, etc. or self-propagating worm-type malware searching for the next infected target. To sum up, the proposed *GLASSO* engine was possible to quickly and precisely grasp indiscriminate cyber attacks like network scans in real-time, automatically, and we thought it would lead to ease a burden of network operation.

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, backscatters caused by IP spoofing, 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 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. 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. If and only if when $\Sigma_{ij}^{-1} = 0$, 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 [11]. 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 [6]. The input arguments of the graphical lasso algorithm are the sample covariance matrix S and the ℓ_1 regularization coefficient $r \in \mathbb{R} (\geq 0)$. 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 [3]. Durumeric *et al.* analyzed a large-scale darknet to investigate scanning activities, and identified

patterns in large horizontal scanning operations [4]. 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 [5].

The *GLASSO* engine was an engine that detects cyber attacks by capturing the cooperativeness between source hosts reaching the darknet. Ban *et al.* [2] 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, and it was not a real-time detection system. 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 cyber attacks 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 [8], that uses GCM 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 .

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

Here, $x^{(m)}$ 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, $\mathbb{N}_0 = \{0, 1, 2, \dots\}$.

Next, we obtain a precision matrix from the data matrix 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 $D \in \mathbb{R}^{M \times N}$ 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 $r \in R(= \{r_1, r_2, \dots, r_s\} \in \mathbb{R}^s (\geq 0))$. From the estimated precision matrix, an undirected graph $G = \{V, E\}$ in GGM could be represented by node set $V = \{x_1, \dots, x_N\}$ 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 anomaly 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 $M, r \in R(=$

$\{r_1, r_2, \dots, r_s\}$, $\mathbf{d}^{(r)}$, K, θ , 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, targetted 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 considered a method of processing alerts (outliers) sequentially by processing every time slot, instead of processing multiple time slots at once. First, the newest time slot t was processed in the same manner as Section IV-B to obtain a graph density value $d_t^{(r)}$. Then, $d_t^{(r)}$ was added to a sequence of the past graph density value $\mathbf{d}^{(r)}$, and if the length of $\mathbf{d}^{(r)}$ was positive integer K , alert judgment (outlier detection) was performed. If the length of $\mathbf{d}^{(r)}$ was less than K , wait until the next time slot updated without performing the alert judgment. Detailed alert judgment method was explained in a next section. The time slot judged as an alert was output as an alert and was deleted from \mathbf{d} so that it would not be referred to in subsequent alert judgment. Also, if no time slot was determined as an alert, deleted the oldest time slot from \mathbf{d} . When the time passes and a new time slot was updated, by repeating the above steps, it was possible to sequentially process GLASSO engine while keeping the number of time slots used for alert judgment. As a result, it obtained the result in real-time with a very short time compared to the batch-mode since only an updated time slot was processed. The pseudo code for GLASSO engine with online processing was described in the Algorithm 1. Here, $\text{length}()$ function got the length of vectors.

B. Alert Judgment Method for Online Processing

In this section, we proposed an outlier detection method for discriminating alerts (outliers) from a sequence of graph density values \mathbf{d} during online processing. It was checked how much the largest element occupies a whole sample variance in the sequence of graph density values \mathbf{d} , and when a ratio was large, the largest element was judged to be an outlier. The pseudo code of the alert judgment method was indicated in the Algorithm 2. Here, $\sigma_{(i+1)}^2 / \sigma_{(i)}^2 < \theta$ was an outlier judgment formula, and $\theta (0 \leq \theta \leq 1)$ was a threshold value of an outlier judgment formula. Also, $\text{order}()$ function returned a permutation which rearranges its first argument into ascending or descending order, $\text{var}()$ function returned a sample variance. From the above, we understood how to process the GLASSO engine and how to output alerts.

VI. PARAMETER TUNING

GLASSO エンジンにはタイムスロットの長さ T , 時系列サンプルの数 M , 正則化項係数 r , そしてアラート判定に用いられるタイムスロット (グラフ密度値) の数 K とそのしきい値 θ のように、複数パラメータが存在する。本節では、どのようにこれら 5 つのパラメータを設定したか紹介する。

Algorithm 1 The GLASSO Engine with Online Processing

Input: t (a time slot), $M, r \in R (= \{r_1, r_2, \dots, r_s\})$, $\mathbf{d}^{(r)}, K, \theta$
Output: alerts or none

```

1: for a time slot  $t$  is updated newly do
2:   preprocess a time slot  $t$ 
3:   make  $\mathbf{D}_t$  from a time slot  $t$ 
4:   compute  $S$  from  $\mathbf{D}_t$ 
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  $\mathbf{d}^{(r)}$ 
9:     if  $\text{length}(\mathbf{d}^{(r)}) = K$  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  $\mathbf{d}$ 
15:      else
16:        remove the most old time slot from  $\mathbf{d}^{(r)}$ 
17:      end if
18:    end if
19:  end for
20: end for

```

Algorithm 2 Pseudo code for Alert Judgment Method

Input: $\mathbf{d} \in \mathbb{R}^K, K, \theta$
Output: *outliers* or none

```

1:  $i \leftarrow 0$ 
2: while TRUE do
3:    $i \leftarrow i + 1$ 
4:    $\mathbf{d}_{(i)} \leftarrow \text{order}(\mathbf{d}, \text{decreasing} = \text{True})[i : K]$ 
5:    $\sigma_{(i)}^2 \leftarrow \text{var}(\mathbf{d}_{(i)})$ 
6:   if  $\sigma_{(i+1)}^2 / \sigma_{(i)}^2 < \theta$  then
7:      $\text{outliers} \leftarrow \text{order}(\mathbf{d}, \text{decreasing} = \text{True})[1 : i]$ 
8:   return outliers
9:   end if
10: end while

```

これらパラメータは経験的ヒューリスティックな方法で決めている。

A. Length of Time Slot T

このタイムスロットの長さ $T(\text{sec.})$ は 1 モデル学習に用いるデータの全観測時系列の長さを意味する。まず、上界 (upper bound) から考える。GLASSO エンジンの計算量は送信元ホストの数に大きく依存する。 T が長くなると、一般に送信元ホストの数も増え、処理時間が指数的に増えてしまい、リアルタイム処理が困難になる。次に、下界 (lower bound) を考える。 T 秒間でネットワークスキャンの 1 キャンペーンが全て観測されるような、豊富な観測ができていれば十分である。経験上、GLASSO エンジンで用いるダークネットで観測される多くのネットワークスキャンの 1 キャンペーンは、観測規模が最大 29,182IP アドレス (約/17) でも 5 分ほどで TCP ポートスキャンが終わる。従って、GLASSO

エンジンでは問題なくリアルタイムに処理を行えて、豊富な観測ができると思われる、 $T = 600(sec.)$ に設定している。

B. The Number of Time Series Samples M

この時系列サンプル数 M は、タイムスロットの長さ T を M 個に分割し、1 モデル学習に用いるデータの時系列サンプルの数を意味する。この意味は、データ行列 D_t から送信元ホストから受信したパケット数の時間傾向を測るときの1区間を M 個に設定し、1区間の長さであるサンプリング間隔 $T/M(sec.)$ をどれぐらいの長さに設定するかを意味する。一般に厳しく区切るほど、パケット数の時間傾向をより厳し目に測ることとなり、良い精度の学習が行われる。しかし、あまりにも区切り過ぎると、本来は協調して動く送信元ホスト間に依存関係はないと推定する恐れがある。逆に区切りが少な過ぎると、どんな送信元ホスト間にも協調性があると推定される恐れがある。従って、 M を適当な数に設定する必要があり、最適な値を探すことは難しいことだが、幸いなことに次の節で紹介する正則化項係数 r はこの M と同様な働きができる。つまり、 M を極端な数に設定しなければ、どのように設定しても正則化項係数 r でカバーすることができる。経験上、サンプリング間隔 $T/M = 50(sec.)$ 程度あれば、十分なパケット数の時間傾向を測れることから、 $M = 12$ と設定している。

C. Regularization Coefficient r

正則化項係数 $r \in \mathbb{R} (\geq 0)$ は graphical lasso アルゴリズムで用いられる入力パラメータであり、どの程度の依存関係までノイズ由来のもののみならずか決めるしきい値であり、推定する精度行列 Σ^{-1} のスパースリティを調整する。時系列サンプル数 M との関係と述べると、 r を調整することで弱く関係し合った依存関係を削ぎ落とすことができるため、 M を調節してパケット数の時間傾向をより厳し目に測ることと同様な意味を持つ。従って、我々は M を12に固定して、 r の値を調整することにした。

r の特徴として、一般に r が0に近いほど $d^{(r)}$ は1に近づく、 r が大きくなるほど $d^{(r)}$ は0に近づく。また、graphical lasso アルゴリズムは r の値を大きくするほど、ゼロ要素が多くなるため、計算時間が短くなる特徴がある。最後に $r \in R(= \{r_1, r_2, \dots, r_s\} \in \mathbb{R}^s)$ のように設定し、何度でも試行可能だが、その試行回数分だけ GLASSO エンジンの処理時間は伸びる。

我々は GLASSO エンジンを $T = 600(sec.)$, $M = 12$ に設定し、 r の値を少数点6桁まで微調整を行ってみた。少数点6桁から2桁までいろいろ値を変えながら試してみても、グラフ密度値は大きく変わることはなかった。例えば、 $r = 0.55$ でアラート判定されるタイムスロットは、ほとんどの場合 $r = 0.5$ か 0.6 でもアラートに判定されることが経験から分かった。このことから少数点1桁で値を変えながら試行すれば良いことが分かった。また、 $r \geq 1$ からはゼロ行列が多くなり、 $d^{(r)} = 0$ となる場合が多かった。そして、 $r < 0.4$ では処理時間が長くなること、十分にスパースな精度行列が推定されないこと、ほとんどの場合外れ値が出てこないことから、最終的に $r \in R(= \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\})$ に設定することにした。

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

アラート判定に用いるグラフ密度値の数 K としきい値 θ は、時系列サンプル数 M と正則化項係数 r の関係と同様に、 K を極端な数に設定しなければ、 θ でカバーすることができる。 K は外れ値検知するときに用いるデータの数を意味する。このデータの数が少な過ぎると、参照できるデータが少なくなるため、外れ値検知が不安定になりやすい。逆に多すぎると、安定し過ぎて外れ値が検知されにくくなる。しかし、この安定さは外れ値判定式 $\sigma_{(i+1)}^2 / \sigma_{(i)}^2 < \theta$ のしきい値 $\theta (0 \leq \theta \leq 1)$ でも調整できる。 θ は1に近いほど外れ値を緩く判定し、0に近いほど外れ値を厳しく判定する。我々は3日分のデータを外れ値検知に用いることにし ($K = 432$)、 θ は試行錯誤の上、現在 0.98 に設定している。

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. Then we evaluated a cyber attack detection accuracy and described the details of detected cyber attacks.

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 $T = 600sec.$, $M = 12$, $K = 432$, $\theta = 0.98$, $r \in R(= \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\})$. 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. We used 8 different darknet sensors and operated the GLASSO engine in real-time for each sensor. This darknet

TABLE I
IPs SIZE AND THE NUMBER OF ALERTS OF EACH DARKNET SENSOR

Sensor	IPs Size	# of Alerts	Sensor	IPs 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

sensor observed with different IPs blocks, the IPs 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.

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 IPs and a TCP port suddenly, due to some cause.

Please note that multiple source hosts in an alert cooperated with each other anomaly, 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 IPs, clearly differed from the other two types which scan a wide range of destination IPs. 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 cyber attacks, an *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 IPs 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 IPs. 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 Cyber attack 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 (timestamp, source IPs, 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 cyber attack detection in October 2018 of the *GLASSO* engine was 91.2% accuracy, 100% precision, 91.2% recall, 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 cyber attacks will be clarified in the future and false negatives may increase.

E. Details of the Detected Cyber attacks

In this section, we introduced the details of detected cyber attacks according to three attack types: IoT malware, router

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 25th), 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 CYBER ATTACK 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%

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 command-and-control server and executes 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, HNS (Hide and Seek). First of all, 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 [12].

2) *Router Vulnerability*: The true positive 11 TCP ports of router vulnerability in Table III were divided into cyber attacks against vulnerabilities that presented in the router products of five manufacturers [9], [13], [14]. First, vulnerabilities existed in the router product of manufacturer A, and as a result of infecting many routers with malware, network scanings 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*: 表 III のその他脆弱性の正解 TCP ポート計 6 個は大きく 4 つのサービスに対する脆弱性に対するサイバー攻撃に分かれる [10]. ある NoSQL データベースに脆弱性が存在し, そのデータベースを探索するためのスキャン活動が観測された. (TCP ポート 5379, 6379, 7379) その他にも L2TP VPN(Layer 2 Tunneling Protocol Virtual Private Network), VNC(Virtual Network Computing), Supermicro BMC(Baseboard Management Controller) といったサービスを探索するためのスキャン活動が観測された. (TCP ポート 1701, 5900, 49152)

2018 年 10 月におけるサイバー攻撃の多くは, それより以前から恒常的 or 定期的に観測されていた既知のマルウェアもしくは既知の脆弱性を狙ったサイバー攻撃である. そ

のような場合、2018 年 10 月の中でサイバー攻撃を検知した時期が適切かどうか語るのには大きな意味を持たない。ただし、2018 年 10 月において新たな傾向を示すサイバー攻撃に対しては検知時期を評価するのは意味がある。そのような意味では、A 社のルータ製品がいくつかの TCP ポートにスキャンを仕掛けたのは 2018 年 10 月において新たな傾向のサイバー攻撃であり、GLASSO エンジンは送信元ホスト数がピークを迎えたときを、全て適切な時期に検知している。

VIII. 見逃しに対する考察

本節では今回見逃した 3 個の TCP ポートに関連するサイバー攻撃に対して考察を行う。TCP ポート 444, 8010 番に対しては Mirai の特徴を持ったスキャン活動が観測され、TCP ポート 2004 番に対してはある CMS(Content Management System) サービスを探索するためのスキャン活動が観測された。10 分間のユニーク送信元ホスト数をダークネットセンサ別に見てみると、8010 番は多いときは 161 送信元ホストも観測されている反面、444,2004 番はそれぞれ高々 19,23 個に達しない。

まず、8010 番を見逃した原因を考えると、現在はアラートが発行されたタイムスロットに対して送信元ホストの数が 1 番多い TCP ポートに対してのみアラートとして扱っていることが考えられる。1 番目以降に送信元ホストの数が多いい TCP ポートが 1 番目の TCP ポートと比べて送信元ホスト数に差があまりないのであれば、1 番目以降の TCP ポートに対してもアラートとして扱うようにすると、TCP ポート 8010 番は簡単に検知できる。しかし、この対策は誤検知を増やす可能性がある。

送信元ホスト数が少ないと協調性は薄くなり、GLASSO エンジンで 444, 2004 番のような事象を捉えることは難しくなる。このような見逃しの対策として、送信元ホストの数を増やせるような方法や少ない数の送信元ホスト間の協調性も捉えられるような方法を考えると、以下の 3 点が考えられる。

- 1) 送信元ホストの数が 1 番多い TCP ポート以降の 2 番目
- 2) 送信元ホストの数を縮小せず、第 4 オクテットまでの IP アドレスを 1 つの送信元ホストとする。
- 3) より大きい規模のダークネットセンサを用いる。
- 4) TCP ポート除外をよりこまめに行う。

上記の対策を適用し、今回見逃した 3 個の TCP ポートに関するサイバー攻撃が GLASSO エンジンで捉えられるのか確かめることは今後の課題とする。

IX. CONCLUSION

先行研究での GLASSO エンジンではできなかったリアルタイム処理を可能にし、より迅速な対応が可能なネットワークスキャンサイバー攻撃検知エンジンを提案した。GLASSO エンジンから得られるアラートは 3 タイプ(サイバー攻撃, survey scan, 一点集中型)あることが分かり、適当な基準を立て分別することで、サイバー攻撃だけのアラートを判別することができた。我々は限定的な ground truth を作成し、GLASSO エンジンのサイバー攻撃検知精度を評価し、そのサイバー攻撃の詳細を紹介した。また、従来の方法では検知できなかったものを検知することもできた。GLASSO エ

ンジンはいくつかのサイバー攻撃を見逃したが、考察を行い対策を考えた。

今後の予定として、見逃しを減らすように GLASSO エンジンを調整し、長期間に渡る運用を行うことで、検知時期の適切さを適当な基準で評価を行いたい。そして、一点集中型がどういうものかダークネット以外の別のデータセット(ハニーポットなど)を用いて真相を調べたい。最後に、GLASSO エンジンはサイバー攻撃検知のみならず、Survey scanner の検知ができることが分かったため、拡張研究として survey scanner の検知を行いたい。

ACKNOWLEDGEMENT

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] M. Akiyama, T. Kawamoto, M. Shimamura, T. Yokoyama, Y. Kadobayashi, and S. Yamaguchi. A Proposal of Metrics for Botnet Detection based on Its Cooperative Behavior. In *Proceedings of the 2007 International Symposium on Applications and the Internet Workshops(SAINT-W'07)*, pp.82-85, 2007.
- [2] 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.
- [3] 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.
- [4] Z. Durumeric, M. Bailey, and J.A. Halderman. An Internet-wide view of Internet-wide scanning. *23rd USENIX Security Symposium*, pp.65-78, 2014.
- [5] 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.
- [6] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3), 2008.
- [7] J. Friedman, T. Hastie, and R. Tibshirani. Graphical Lasso: Estimation of Gaussian Graphical Models. <https://cran.r-project.org/web/packages/lasso/lasso.pdf>, [Accessed Dec. 2018].
- [8] 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.
- [9] Huawei, Security Notice - Statement on Remote Code Execution Vulnerability in Huawei HG532 Product. <https://www.huawei.com/en/psirt/security-notice/huawei-sn-20171130-01-hg532-en>, [Accessed Dec. 2018].
- [10] 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].
- [11] 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.
- [12] Netlab 360, HNS Botnet Recent Activities. <https://blog.netlab.360.com/hns-botnet-recent-activities-en/>, [Accessed Dec. 2018].
- [13] 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].
- [14] 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].