Vol.2016-MPS-107 No.3
2016/3/8

A Semi-Supervised Data Screening for Network Traffic Data using Graph Min-Cuts

Takayoshi Shoudai[1,3,a)] Hikaru Murai[2,3] Atsushi Okamoto[3]

Abstract: There are currently many projects aimed at devising efficient countermeasures against critical incidents occurring on the Internet through early detection. A nasty problem is hard-to-find accesses by well-analyzed malware whose packets make anomaly detection harder. In this paper, in order to find such accesses from raw data obtained by network monitoring, we propose an automatic data screening method using graph-based semi-supervised learning (Blum and Chawla, 2001) and show its effectiveness in experiments on darknet traffic.

Keywords: Semi-supervised learning, Minimum cut, Data screening, Incident detection, Darknet monitoring.

1. Introduction

Incidents caused by malicious users on the Internet, e.g. disclosure of personal information caused through computer viruses, have become nasty problems. In particular, malicious users called bot herders are causing serious problems. A bot herder scans specific network ranges and infects ordinary users' computers in order to control a large number of them remotely. A group of computers controlled in this manner is called a botnet and can be used to launch denial-of-service (DoS) attacks or spam e-mail. Many researchers have been developing technologies aimed at early detection and extermination of botnets.

Recently a number of Internet organizations have built observation networks for continuous monitoring to detect sudden network incidents. Here, darknet observation is a way of continuously monitoring for new attacks. A darknet is an IP address space that is available on the Internet but is not used, i.e., not assigned to any computer. As such, correct packets rarely reach a darknet. Nevertheless, many packets do indeed reach darknets; for example, in 2013, about 12.88 billion packets reached an anonymous darknet set up by a certain Japanese organization. It is considered that the amount will only increase in the future. Moreover, at present, it is very difficult and takes time to determine whether a packet reaching a darknet is part of a new malicious attack or not.

Many studies aim to detect new attacks by using statistics and/or visualization technology, but most of the measures proposed so far have a common problem. Network traffic contains huge amounts of simple malicious packets such as scans of IP address spaces and ports, as well as backscatter caused by old malware. In addition, there is a relatively small amount of hard-to-find packets from well-analyzed malware. Such packets obscure new attacks and potentially hide them from monitoring. So before trying to detect a new attack, we need to execute screenings to delete such malicious packets thereby make it easier to analyze the traffic. In fact, as the amount of traffic has been increasing rapidly, screening has become an important preprocessing for early detection of new attacks. In particular, for helping traffic analysis, screening methods need to have a high noise reduction capability and a short computational time for quick screening.

In this paper, we propose a screening method using semi-supervised learning (see [7] for a survey) and experimentally examine it operating on actual darknet traffic data. Here, an ordinary screening would be one that deletes all packets that have a certain port number or specified TCP flags, e.g., SYN, ACK, and RST. Such a method works well when there is no need to check a certain port number or TCP flag. However, packets should not be deleted if there is a chance to get clues from the number or flag; that means deleting all of the packets may be going too far and a more selective method should be used instead. Another method, called filtering, checks packets against a registry containing the characteristics of known worms or malware and deletes the packets matching them. This method of filtering is also problematic. For example, its registers may not include all subspecies of a virus and the amount of data in the registry can become too large. Because anyone can make subspecies easily, attempting to register them all is a dubious way to catch a quickly evolving subspecies.

As screenings that take advantage of machine learning, the ones devised by Tsuruta et al. [6] and Okamoto and Shoudai [4] use frequent patterns. These methods delete packets by considering the discovered pattern's coverage or size. Such screening methods enable us to automatically delete groups in which huge amounts of packets arrive in a short time. Such a group is called a spike. However, it is difficult to use these methods to detect and delete less frequent attacks such as slow scanning ones that con-

[1] Faculty of International Studies, Kyushu International University, 805-8512, Japan
[2] Department of Informatics, Kyushu University, Fukuoka, 819-0395, Japan
[3] Institute of Systems, Information Technologies and Nanotechnologies (ISIT), Fukuoka, 814-0001, Japan
[a)] E-mail:shoudai@isb.kiu.ac.jp

sist of one attack every ten or so minutes. Summarizing the above points, the conventional screenings have problems as regards (a) the quality of expression with increasing usable data, (b) accuracy of deleting only traffic of unrelated attacks, and (c) adaptability to new attacks or subspecies.

To solve these problems, we propose a traffic screening method using semi-supervised learning with using graph minimum cut (or min-cut, for short) methods [1]. Learning with min-cut is a semi-supervised method based on graphs (see [5] for a review of recent results). In this study, packets are considered to be vertices that are partitioned into three labels, i.e. positive, negative, and unlabeled. We construct a weighted directed graph and apply the semi-supervised learning with min-cut to the weighted directed graph. As a result, we obtain new labels so that almost all vertices are labeled as either positive or negative. The labeled packets are checked and some of them are deleted. We show the effectiveness of this method in experiments on actual darknet traffic.

2. Preliminaries

2.1 Darknet and Its Traffic

A darknet is a space of IP addresses that are not used by active computers but are available on the Internet. As such, most packets sent to darknets are not proper accesses, but the result of some of hacking activity. Some legitimate organizations have attempted to catch a tendency of a hacking by monitoring packets sent to darknets. Packets sent to darknets are considered to contain scans made by malware, rebounding packets, called backscatter, from hosts that are targets of distributed denial of service (DDoS) attack, configuration error, preparatory action for reflection attacks, and so on.

Each packet of darknet traffic contains the following information: Source IP address and port, Destination port, TTL (Time To Live), Identification, Sequence number, and Acknowledge (ACK) number. By analyzing such information, we can clarify the past situation of incidents, as well as detect attacks early that may cause serious damage. The amount of traffic data reaching a darknet may be huge, and ordinarily, it is impossible to analyze all of the data manually. Thus, many researchers have developed systems and techniques to analyze darknet traffic.

2.2 Semi-supervised learning

Semi-supervised learning is expected to output better results than supervised or unsupervised learning. Semi-supervised learning uses relatively small amounts of labeled data together with large amounts of unlabeled data. The labeled data is classified into the two classes (positive and negative). In Section 4, we obtain labeled data by executing another learning algorithm based on non-negative matrix factorization (NMF) proposed by Kawamura et al. [3]. Their algorithm is an unsupervised learning methods for early detection of network incidents. In this paper, we call it the NMF-engine.

We propose a screening method based on a semi-supervised learning on a weighted directed graph constructed from data processed using the method of Blum and Chawla [1]. Their method uses the maximum flow/minimum cut algorithm to find the minimum cut in a weighted directed graph. It classifies the unlabeled data into two classes according to the discovered minimum cut. Here, let us briefly explain semi-supervised learning. Let X be the whole data, and let Y be a set of labels, i.e., $Y = \{+, -\}$. The elements $+$ and $-$ in Y represent positive and negative, respectively. The inputs of a semi-supervised method are two subsets of X, say L and U. Every element in L is labeled with 0 or 1. That is, a function $f_L : L \to Y$ is given in advance. Every element of the set U is unlabeled. The output of the method is a function $f : L \cup U \to Y$, i.e., a mapping the elements in Y to the elements in $L \cup U$, such that $f(x) = f_L(x)$ for all $x \in L$.

There are several kinds of semi-supervised learning. Methods based on classification start from an initial classification and repeatedly update (refines) it. Self-training and co-training [7] are examples of this kind. The most important point about this method is how to compute or choose the initial classification. If the initial classification is not good enough, the updates might increase the number of errors. This means that the reliability of the labeled data might determine the overall reliability. Subspecies of various viruses are possibly included in traffic data, and new subspecies are frequently generated. Therefore, if the identity of the malware does not reflect these changes, it would not be advisable to use it to label unlabeled data. For this reason, a labeling method that repeatedly uses a particular classification may be considered inaccurate. On the other hand, in this paper, we construct a labeling function f by using the maximum flow/minimum cut algorithm, assuming that similar data tend to have the same label. At first, we define the similarity between two packets in detail. Next, we define the concept of similarity and construct a weighted directed graph using similarity. After that, we describe the screening method using the semi-supervised learning of Blum and Chawla [1].

3. Graph-Based Screening

First let us explain how to construct a weighted directed graph for graph-based learning. One packet in traffic data is represented by one vertex of a weighted directed graph. In order to create weighted directed edges, we define the similarity between two vertices (see in Section 3.1). The similarity is a distance that is calculated from traffic data described in Section 2.1. For any two vertices, we decide on whether or not to create a weighted directed edge between the vertices according to the similarity between them and the similarities among the k-nearest neighbors ($k \geq 1$) of them, where k is a constant positive integer that is given in advance. The details on how to create a weighted directed graph are described in Section 3.2.

3.1 Distance on traffic data space

3.1.1 IP header

The IP header is a prefix to an IP packet, and it determines the destinations and routes. It contains a source IP address, destination IP address, TTL (Time To Live), and so on, which is common information among all protocols. Below, we define distances in terms of the source IP address, TTL, and identification of the IP header.

- Distance determined by source and destination IP addresses
  Here, we divide the source IP address of packet x_1 into

Table 1 The initial values of typical operating systems

OS	initial value of TTL
Windows 95	32
Mac OS 2.0.x	60
Mac OS X	64
Windows 98	128
Windows XP	128
MPE/IX (HP)	200
OpenBSD	255

octets. Let $a_{1,1}, a_{1,2}, a_{1,3}$, and $a_{1,4}$ be the 1st, 2nd, 3rd, and 4th octets of the source IP address of x_1. Similarly, let $a_{2,1}, a_{2,2}, a_{2,3}$, and $a_{2,4}$ be the octets of the source IP address of packet x_2. We can use the standard of allocating IP addresses to define a source address distance $d_{sadr}(x_1, x_2)$ between x_1 and x_2 such that the upper octet is weighted more heavily than the lower octet.

$$d_{sadr}(x_1, x_2) = \begin{cases} 4 & \text{if } a_{1,1} \neq a_{2,1}, \\ 3 & \text{if } (a_{1,1} = a_{2,1}) \wedge (a_{1,2} \neq a_{2,2}), \\ 2 & \text{if } (a_{1,1} = a_{2,1}) \wedge (a_{1,2} = a_{2,2}) \\ & \quad \wedge (a_{1,3} \neq a_{2,3}), \\ 1 & (a_{1,1} = a_{2,1}) \wedge (a_{2,1} = a_{2,2}) \\ & \quad \wedge (a_{1,3} = a_{2,3}) \\ & \quad \wedge (a_{1,4} \,\&\, \text{0xf0} \neq a_{2,4} \,\&\, \text{0xf0}), \\ 0 & \text{otherwise,} \end{cases}$$

where "&" is the bitwise AND operator.

In a similar way, let $b_{1,1}, b_{1,2}, b_{1,3}$, and $b_{1,4}$ be the 1st, 2nd, 3rd, and 4th octets of the destination IP address of x_1, and let $b_{2,1}, b_{2,2}, b_{2,3}$, and $b_{2,4}$ be the octets of the destination IP address of packet x_2. We define a destination address distance $d_{dadr}(x_1, x_2)$ between x_1 and x_2 as follows:

$$d_{dadr}(x_1, x_2) = \begin{cases} 4 & \text{if } b_{1,1} \neq b_{2,1}, \\ 3 & \text{if } (b_{1,1} = b_{2,1}) \wedge (b_{1,2} \neq b_{2,2}), \\ 2 & \text{if } (b_{1,1} = b_{2,1}) \wedge (b_{1,2} = b_{2,2}) \\ & \quad \wedge (b_{1,3} \neq b_{2,3}), \\ 1 & (b_{1,1} = b_{2,1}) \wedge (b_{2,1} = b_{2,2}) \\ & \quad \wedge (b_{1,3} = b_{2,3}) \\ & \quad \wedge (b_{1,4} \,\&\, \text{0xf0} \neq b_{2,4} \,\&\, \text{0xf0}), \\ 0 & \text{otherwise.} \end{cases}$$

- Distance determined by TTL
  TTL (Time to Live) describes how long a packet survives, i.e., the maximum number of times that a packet can go through routers. The initial TTL value depend on the OS and its version. The initial values of typical OSs are given in Table 1. Eto et al. [2] described a way of estimating the initial TTL value of a received packet. We give a brief summary in Table 2. The numbers $a_{ttl}(x_1)$ and $a_{ttl}(x_2)$ denote the initial TTL values estimated from the received ones. The TTL distance $d_{ttl}(x_1, x_2)$ is defined as follows:

$$d_{ttl}(x_1, x_2) = \begin{cases} 2 & \text{if } a_{ttl}(x_1) \neq a_{ttl}(x_2), \\ 0 & \text{otherwise.} \end{cases}$$

- Distance determined by identification
  In order to send and receive huge amount of data completely, that data has to be divided into packets. Identifications, i.e., IDs, are used to identify to which data a packet belongs. Packets belonging to the same data will have the same identifications. Let $a_{id}(x_1)$ and $a_{id}(x_2)$ be the identifications of x_1 and x_2. The identification distance $d_{id}(x_1, x_2)$ is defined as follows:

$$d_{id}(x_1, x_2) = \begin{cases} 1 & \text{if } a_{id}(x_1) \neq a_{id}(x_2), \\ 0 & \text{otherwise.} \end{cases}$$

Table 2 Initial TTL estimated from received value (Eto et al. [2]).

receiving value of TTL d	estimated initial value of TTL
$0 \leq d \leq 21$	32
$22 \leq d \leq 39$	48
$40 \leq d \leq 59$	64
$60 \leq d \leq 89$	100
$90 \leq d \leq 120$	128
$120 \leq d \leq 159$	168
$160 \leq d \leq 189$	200
$190 \leq d \leq 255$	255

- From the above, the distance $d_{ip}(x_1, x_2)$ as determined by the IP header between x_1 and x_2 is defined as follows:

$$d_{ip}(x_1, x_2) = d_{sadr}(x_1, x_2) + d_{dadr}(x_1, x_2) + d_{ttl}(x_1, x_2) + d_{id}(x_1, x_2).$$

3.1.2 TCP header

A TCP (Transmission Control Protocol) packet contains information identifying the source port, destination port, sequence number, and acknowledgment number in its header.

- Distance determined by source and destination ports
  Most malware tends to attack to a specific destination port. For example, one piece of malware called "Morto" spreads by misusing a remote desktop connection in Windows, and it aims for the destination port number 3389. In view of this, a difference in destination port number is regarded as more important than one in source port number. Let sp_i and dp_i $(i = 1, 2)$ be the source and destination ports of packets x_i $(i = 1, 2)$. The distances $d_{sport}(x_1, x_2)$ and $d_{dport}(x_1, x_2)$ between x_1 and x_2 as determined by the source ports and destination ports are

$$d_{sport}(x_1, x_2) = \begin{cases} 2 & \text{if } sp_1 \neq sp_2, \\ 0 & \text{otherwise.} \end{cases}$$

$$d_{dport}(x_1, x_2) = \begin{cases} 4 & \text{if } dp_1 \neq dp_2, \\ 0 & \text{otherwise.} \end{cases}$$

- Distance determined by sequence numbers
  The sequence number is a serial number for TCP communications. It is used for verifying lists of orders and detecting midstream losses of a packets. This number determines how much data is sent. It increases by 1 every time 1 byte is sent. Its initial value must not be 0 and is chosen randomly. Let seq_1 and seq_2 be the sequence numbers of x_1 and x_2. The distance $d_{seq}(x_1, x_2)$ between x_1 and x_2 as determined by the sequence number is defined as follows:

$$d_{seq}(x_1, x_2) = \begin{cases} 2 & \text{if } |seq_1 - seq_2| > 4, \\ 0 & \text{otherwise.} \end{cases}$$

- Distance determined by acknowledge numbers
  The acknowledge number indicates how much data is received. It is observable at the receiving side. It corresponds to a sequence number. The receiving side adds 1 to a sequence number and returns its value to the sending side. Let ack_1 and ack_2 be the acknowledged numbers of x_1 and x_2. The distance $d_{ack}(x_1, x_2)$ between x_1 and x_2 as determined by the acknowledge number is defined as follows:

$$d_{ack}(x_1, x_2) = \begin{cases} 1 & \text{if } ack_1 \neq ack_2, \\ 0 & \text{otherwise.} \end{cases}$$

- From the above, the distance $d_{tcp}(x_1, x_2)$ between x_1 and x_2 as determined by TCP is defined as follows:

$$\begin{aligned} d_{tcp}(x_1, x_2) &= d_{sport}(x_1, x_2) + d_{dport}(x_1, x_2) \\ &\quad + d_{seq}(x_1, x_2) + d_{ack}(x_1, x_2). \end{aligned}$$

3.1.3 UDP header

UDP (User Datagram Protocol) only has information on the source and destination ports, and the length of UDP data. The length of UDP data is the value in bytes of the whole datagram including its header and data.

- Distance determined by the length of UDP data
  Let len_1 and len_2 be the lengths of UDP data of x_1 and x_2. The distance $d_{udp\_len}(x_1, x_2)$ by the length of UDP data is defined as follows:

$$d_{udp\_len}(x_1, x_2) = \begin{cases} 1 & \text{if } len_1 \neq len_2, \\ 0 & \text{otherwise.} \end{cases}$$

- The distance $d_{udp}(x_1, x_2)$ between x_1 and x_2 as determined by UDP is defined as follows:

$$\begin{aligned} d_{udp}(x_1, x_2) &= d_{sport}(x_1, x_2) + d_{dport}(x_1, x_2) \\ &\quad + d_{udp\_len}(x_1, x_2). \end{aligned}$$

3.1.4 ICMP header

ICMP (Internet Control Message Protocol) handles error notification and transports control messages. ICMP is used to diagnose communication lines connecting computers. ICMP packets contain type and code information.

- Distance determined by the code of ICMP data
  Let $code_1$ and $code_2$ be the codes of the ICMP data of x_1 and x_2, and the distance $d_{icmp\_code}(x_1, x_2)$ as determined by the code of the UDP data is defined as follows:

$$d_{icmp\_code}(x_1, x_2) = \begin{cases} 1 & \text{if } code_1 \neq code_2, \\ 0 & \text{otherwise.} \end{cases}$$

- The distance $d_{icmp}(x_1, x_2)$ between x_1 and x_2 as determined by ICMP is defined as follows:

$$\begin{aligned} d_{icmp}(x_1, x_2) &= d_{sport}(x_1, x_2) + d_{dport}(x_1, x_2) \\ &\quad + d_{icmp\_code}(x_1, x_2). \end{aligned}$$

```
Algorithm MSSL
Input: a set $L$ of packets each of which is labeled with + or −, and a set $U$ of packets with no label.
Output: a partition $\{U_+, U_-, U_0\}$ of $U$.
begin
(1) For any 2 packets $x_1, x_2 \in L \cup U$, compute the distance $d(x_1, x_2)$.
(2) Construct a weighted directed graph $G = (V, E)$ as follows:
    (a) Let $V = L \cup U \cup \{v_+, v_-\}$ be a set of vertices, where $v_+$ and $v_-$ are new vertices.
    (b) For any vertex $x \in L \cup U$, find the $k$ nearest vertices in the ascending order of the distances. Let the distances be $D_1, \ldots, D_k$. For each $y \in L \cup U$ that has distances $D_j$ $(1 \leq j \leq k)$ from $x$, make the weighted directed edges $(x, y)$ and assign the integer $k - j + 1$ as their weights.
    (c) For any positive labeled vertex $x \in L$, make directed edges $(v_+, x)$ and $(x, v_+)$, and assign infinity as their weights.
    (d) For any negative labeled vertex $x \in L$, make directed edges $(v_-, x)$ and $(x, v_-)$, and assign infinity as their weights.
    (e) For two vertices having the same label if they have no edge between them, make weighted directed edges and assign infinity as their weights.
(3) For the weighted directed graph constructed in the above way, execute the maximum flow/minimum cut algorithm to compute the maximum flow from $v_+$ to $v_-$.
(4) By removing the minimum cut-set obtained from the maximum flow, divide $V$ into the following three sets.
    • $V_+$ : the set of vertices that are reachable from $v_+$,
    • $V_-$ : the set of vertices reachable to $v_-$,
    • $V_0$ : the set of vertices that are not reachable from $v_+$ or to $v_-$.
(5) Label the vertices in $V_+$ with + and the vertices in $V_-$ with −.
(6) Let $U_+ = U \cap V_+$, $U_- = U \cap V_-$, and $U_0 = U \cap V_0$.
end.
```

Fig. 1 Algorithm MSSL: Semi-supervised screening algorithm using minimum cut on graph.

3.1.5 Distance between two packets

Finally the distance $d(x_1, x_2)$ between two packets x_1 and x_2 is defined as follows:

$$d(x_1, x_2) = \begin{cases} d_{ip}(x_1, x_2) + d_{tcp}(x_1, x_2) \\ \quad \text{if both } x_1 \text{ and } x_2 \text{ are } \textit{TCPs}, \\ d_{ip}(x_1, x_2) + d_{udp}(x_1, x_2) \\ \quad \text{if both } x_1 \text{ and } x_2 \text{ are } \textit{UDPs}, \\ d_{ip}(x_1, x_2) + d_{icmp}(x_1, x_2) \\ \quad \text{if both } x_1 \text{ and } x_2 \text{ are } \textit{ICMPs}, \\ \infty \quad \text{otherwise.} \end{cases}$$

3.2 Semi-supervised screening algorithm using minimum cut on graph

Figure 1 shows the algorithm that creates a weighted directed graph and labels the vertices with either positive or negative labels. The algorithm is based on the semi-supervised algorithm proposed by Blum and Chawla [1]. Let k be a constant positive number that is given in advance. Blum and Chawla showed that when $k = 1$, minimizing the value of the minimum cut corresponds to minimizing the *LOOCV (leave-one-out cross-validation) error*.

4. Experiment on Real Darknet Data

Many researchers are engaged in developing early detection

Table 3 Labeled data that was randomly selected from the 13th period of 11th July 2011.

	Number of packets	Numbers of packets to ports		
		22	23	3389
Positive data L_+	130	37	62	0
Negative data L_-	137	0	3	0

Table 4 Remaining rate of unlabeled packets as determined by MSSL.

		Total amount	New positive data U_+	Remaining rate
Unlabeled data U		355,495	222,327	37.46 %
Content:	Packets to port No.22	35,242	29,013	17.64 %
	Packets to port No.23	8,657	8,513	1.66 %
	Packets to port No.3389	16,893	777	95.40 %

systems for network incidents. Their main purpose is to detect new cyber attack patterns and to issue early warnings, not to detect known patterns. It can be said that known patterns have been dealt with. So when we search for new patterns of attacks, the known ones make it difficult to detect new patterns. Our purpose is to remove these known malware patterns from packets in order to detect new patterns of malicious attacks.

The screening experiment needs data labeled with positive or negative. The labels indicate whether the packet is already known to be an attack or not. We used data labeled using the NMF-engine of Kawamura et al. [3]. We applied Algorithm MSSL to the labeled and unlabeled darknet traffic data. We screened out darknet traffic data that had initially no labels other than new positive ones, which would then be sent for further tests; i.e., for a given set U of unlabeled packets, we computed a partition $\{U_+, U_-, U_0\}$ of U by Algorithm MSSL and left $U \setminus U_+$, i.e., $U_- \cup U_0$. Two experiments are discussed below.

4.1 NICT darknet data collected in July 2011

The National Institute of Information and Communications Technology, Japan (NICT) has set up a darknet and monitors it in order to examine and understand the behavior of Internet traffic data. First, we have experimented on NICT darknet data collected in July 2011. As preprocessing, packets with the TCP flag RST were deleted because they are unrelated to any malware attacks. After that, we divided the 24 hour data in amounts received in the corresponding 30 minutes; i.e. we divided up the 24 hours worth of data into 48 periods of which there were tens of thousands packets in each.

The "Morto" malware was first recognized on 11th July 2011 (JST), when it attacked destination port number 3389. There are 12,252 packets identified to be malicious by the NMF-engine in the 13th period of 11th July, which includes 29,277 packets. During this period, the NMF-engine detected packets bound for destination port numbers 22 and 23 and issued alerts on them. We considered that the malicious data detected by the NMF-engine to be positive data. We randomly chose 131 packets from positive data and set the set of these packets to be L_+. We also randomly chose 137 packets from negative data that consisted of packets during the 13th period of 11th July that were not identified to be malicious by the NMF-engine (Table 3). We set the set of these packets to be L_-.

The purpose of this experiment was to remove only packets of known malware found by existing malware detection systems. We expected to obtain new data that contained attacks to destination port number 3389, like "Morto", at a higher rate.

Evaluation 1: We have applied Algorithm MSSL to every data of the 14th to 48th periods, which were used as unlabeled data U. To evaluate how many packets were still left in $U_- \cup U_0$, we calculated the remaining rate of packets attacking destination port numbers 22, 23, and 3389. More precisely, let N be the number of packets sent to destination port number A before screening, and n the number of packets sent to the same port after screening. The remaining rate of A is defined as n/N.

The rate of remaining packets considered to be part of known attacks should be smaller than that of packets considered to be unrelated to any other attack. These "remaining rates" after screening are shown in Table 4. We can see that the remaining rate for destination port number 3389 is much bigger than the others. Therefore, it can be said that Algorithm MSSL successfully reduced the number of packets of each data. However, the remaining rate of the whole unlabeled data was small. This is because the positive labeled data contains packets destined not only to destination port numbers 22 and 23 but also to the other ports. The remaining rates of packets sent to each port tend to depend on the original labeled data.

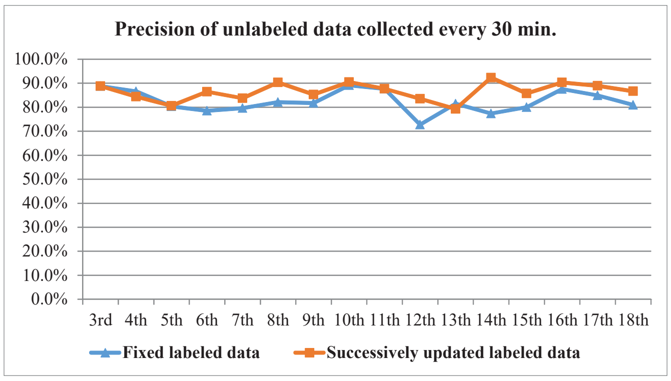
4.2 International data collected in January 2014

Next, we ran Algorithm MSSL on data obtained by monitoring a darknet in the Maldives on 29th January 2014 (JST). Packets with SYN and ACK, or RST TCP flags were all removed because we decided that those packets were unrelated to any attack.

4.2.1 Fixed labeled data

On 29th January 2014 (JST), the NMF-engine found malicious packets in all of the 2nd to 18th periods. 149 positive labeled packets were selected randomly at a 25% rate from the malicious packets of the 2nd period on 29th January 2014. We set the set of these packets to be L_+. Moreover, 170 negative labeled packets were selected randomly at a 25% rate from packets of the same period that were not identified to be malicious by the NMF-engine. We set the set of these packets to be L_-. We used unlabeled data of the 3rd to 18th periods on the same day. We appied Algorithm MSSL to the 16 sets of packets in the 3rd to 18th periods. We refered these sets as $U^{(i)}$ for $i = 3, 4, \dots, 18$. The total number of packets in each unlabeled data was 13,059. The maximum and minimum numbers of packets in each period were 1,197 and 204, respectively. We evaluated the accuracy of learning as follows:

Evaluation 2: We evaluated the precision, recall, and F-measure between the two kinds of the positive packets identified



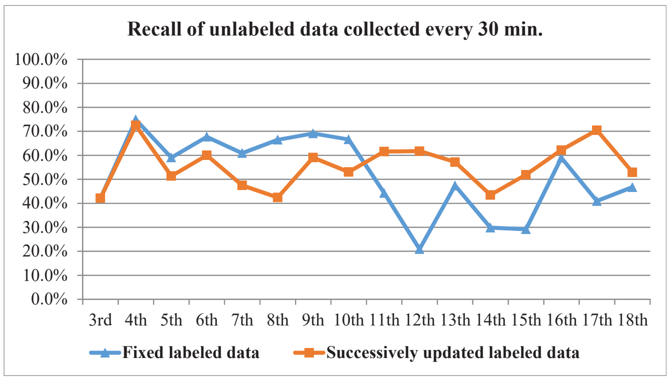


Fig. 2 Transitions of Precision (on left) and Recall (on right) using data labeled by MSSL fixed or updated. Horizontal axises means start times of divided by 30 minutes.

by the NMF-engine and Algorithm MSSL. Let $L_+^{(i)}$ be the set of packets in the i-th period for $i = 3, 4, \dots, 18$ that were identified to be positive, i.e., malicious, by the NMF-engine. Let $U_+^{(i)}$ be the set of packets that were identified to be positive by Algorithm MSSL on inputs L_+, L_-, and $U^{(i)}$. For $i = 3, 4, \dots, 18$, the precision $p^{(i)}$, the recall $r^{(i)}$, and F-measure $f^{(i)}$ are defined as $p^{(i)} = |L_+^{(i)} \cap U_+^{(i)}|/|U_+^{(i)}|$, $r^{(i)} = |L_+^{(i)} \cap U_+^{(i)}|/|L_+^{(i)}|$, and $f^{(i)} = 2 \cdot p^{(i)} \cdot r^{(i)}/(p^{(i)} + r^{(i)})$, respectively.

The precision, recall, and F-measure of the whole data were 83.4%, 50.8%, and 63.1%, respectively. We consider that the precision is high enough. Generally speaking, a screening traffic data method will not remove packets originating from unknown malware. Therefore, we organized the parameters so as to keep the precision high. Consequently, the recall became relatively low. One of the reasons is that semi-supervised learning has less efficiency on little unlabeled data. Another reason is the fixed labeled data. Since malware attacks are being replaced at faster rates, accuracy with fixed labeled data is considered bad. Therefore, we conducted an experiment in which the labeled data were successively updated.

4.2.2 Successively updated labeled data

In the above experiment, we took positive data to be labeled at a fixed point in time. Consequently, accuracy deteriorated over time (Figure 2). For that reason, we experimented on our algorithm using successively updated labeled data. In doing so, we expected to improve accuracy. We applied Algorithm MSSL in the following way ($i = 3, 4, \dots, 18$):

(1) Let L_+ be a subset of the positive labeled packets selected randomly at a 25% rate from $L_+^{(i-1)}$, and let L_- be a subset of the negative labeled packets selected randomly at a 25% rate from packets of the $(i-1)$-th period that were not identified to be malicious by the NMF-engine.

(2) Apply Algorithm **MSSL** to unlabeled data $U^{(i)}$ in the i-th period with L_+ and get new positive labeled data $W_+^{(i)} \subseteq U^{(i)}$.

(3) Evaluate $W_+^{(i)}$ and $L_+^{(i)}$ by calculating its accuracy, i.e., its precision, recall, and F-measure, which are defined in the same way as the previous experiment.

When the labeled data were updated every period, the precision of the whole unlabeled data was 86.8%. This rate is better than the 83.40% of when the labeled data is fixed. Moreover, the recall and F-measure improved on average. Figure 2 shows the precision and recall when labeled data were fixed or updated successively. We can conclude that updating the labeled data improved learning accuracy.

5. Conclusion

We proposed a screening method using semi-supervised learning based on graphs. Screening experiments were conducted on real darknet traffic data using positive and negative data labeled by the NMF-engine. We are currently developing a screening method that achieves high recall while keeping precision high.

Acknowledgments This work was partially supported by Proactive Response Against Cyber-Attacks Through International Collaborative Exchange (PRACTICE), Ministry of Internal Affairs and Communications, Japan, and partially supported by Grant-in-Aid for Scientific Research (B) (Grant Numbers 26280087) from Japan Society for the Promotion of Science (JSPS), and Grant-in-Aid for Scientific Research on Innovative Areas (Grant Number 24106010) from the Ministry of Education, Culture, Sports, Science and Technology (MEXT), Japan.

References

[1] Blum, A. and Chawla, S.: Learning from Labeled and Unlabeled Data using Graph Mincuts, *Proceedings of the 18th International Conference on Machine Learning (ICML2001)*, pp.19–26 (2001).

[2] Eto, M., Inoue, D., Suzuki, M., and Nakao, K.: A Statistical Packet Inspection for Extraction of Spoofed IP Packets on Darknet,*Proceedings of the 4th Joint Workshop on Information Security (JWIS 2009)*, (2009).

[3] Kawamura, Y., Shimamura, J., Nakazato, J., Yoshioka, K., Eto, M., Inoue, D., Takeuchi, J., and Nakao, K.: Experimental Evaluation of A Botnet Detection Method based on Non-negative Matrix Factorization (in Japanese), *The Institute of Electronics, Information and Communication Engineers, Japan, IEICE Technical Report*, 113(288), ICSS2013-61, pp.23–28 (2013).

[4] Okamoto, A. and Shoudai, T.: Mining First-Come-First-Served Frequent Time Sequence Patterns in Streaming Data, *Proceedings of the IADIS International Conference on e-Society (ES2013)*, pp.283–290 (2013).

[5] Subramanya, A. and Talukdar, P.P.: *Graph-Based Semi-Supervised Learning*, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers (2014).

[6] Tsuruta, H., Shoudai, T., and Takeuchi, J.: Network Traffic Screening Using Frequent Sequential Patterns, *Intelligent Control and Innovative Computing, Springer, Lecture Notes in Electrical Engineering*, Vol.110, pp.363–375 (2012).

[7] Zhu, X. and Goldberg, A.B.: *Introduction to Semi-Supervised Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers (2009).