Regular Paper

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

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

Received: February 9, 2016, Revised: March 30, 2016,
Accepted: May 16, 2016

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 the darknet set up by National Institute of Information and Communications Technology (NICT), Japan [6]. (The data are accessed as a part of MWS Dataset 2015 [4], [5].) 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 Ref. [13] 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. [11] and Okamoto and Shoudai [9] use frequent patterns. These methods delete packets by considering the discovered pattern's coverage or size. Such screening

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

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 consist 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 [10] 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 Yamauchi et al. [12] and Kawamura et al. [7]. NMF is a method to decompose a non-negative vector valued time series data into linearly independent non-negative components. Yamauchi et al. [12] proposed a method not only to decompose traffic data into some patterns by NMF but also to detect malware activities in those patterns. Their algorithm is an unsupervised learning method 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 $+$ or $-$. 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 [13] 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.

Table 1 Number of pairs of packets in NICT darknet data collected in the 13th period of 11th July 2011 (see the detail in Section 4.1).

		(1)	(2)	(3)	(4U)	(4L)	(EQ)
Source IP address ($X = a$)	Number of pairs of packets	286,139,041	10,348,698	503,606	14,041	2,017	131,549,323
		66.768%	2.415%	0.118%	0.003%	0.000%	30.696%
Destination IP address ($X = b$)	Number of pairs of packets	0	0	421,288,443	6,799,411	439,673	29,199
		0.000%	0.000%	98.304%	1.587%	0.103%	0.007%

3.1 Distance on Traffic Data Space

In this section, we give the definition of distance on traffic data in order to classify a given unlabeled data into positive and negative classes. The distance measure is defined based on our experiment (in Section 4.1) and common knowledge on the Internet.

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 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.

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

  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:

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

  In Section 4.1, we conducted the experiment on NICT darknet data collected in the 13th period of 11th July 2011. The number of packets contained in the data was 29,277. Thus, we had totally 428,556,726 pairs of packets in the data. The detail of the data and the experiment are stated in Section 4.1. In the experiment, we implicitly categorized all pairs of packets according to the octets of source and destination IP addresses. We give the result of the categorization in **Table 1**. In the table, for source IP address ($X = a$) and destination IP address ($X = b$), we divided all pairs of packets into six categories: (1) $X_{1,1} \neq X_{2,1}$, (2) $(X_{1,1} = X_{2,1}) \wedge (X_{1,2} \neq X_{2,2})$, (3) $\bigwedge_{i=1}^{2}(X_{1,i} = X_{2,i}) \wedge (X_{1,3} \neq X_{2,3})$, (4U) $\bigwedge_{i=1}^{3}(X_{1,i} = X_{2,i}) \wedge (X_{1,4} \,\&\, \mathrm{0xf0} \neq X_{2,4} \,\&\, \mathrm{0xf0})$, (4L) $\bigwedge_{i=1}^{3}(X_{1,i} = X_{2,i}) \wedge (X_{1,4} \,\&\, \mathrm{0xf0} = X_{2,4} \,\&\, \mathrm{0xf0}) \wedge (X_{1,4} \,\&\, \mathrm{0x0f} \neq X_{2,4} \,\&\, \mathrm{0x0f})$, (EQ) $\bigwedge_{i=1}^{4}(X_{1,i} = X_{2,i})$. The NICT darknet uses a class B IP address space [8]. Therefore, no pair of destination IP address ($X = b$) exists in (1) and (2). Except for the case, the distribution of the numbers in each category seems to follow standard rules for allocating an IP address to an individual computer. Especially, we consider that two distinct source IP addresses in (4U) worked together to perform a task.

Table 2 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

Table 3 Initial TTL estimated from received value (Eto et al. [3]).

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

- 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 2**. Eto et al. [3] described a way of estimating the initial TTL value of a received packet. We give a brief summary in **Table 3**. 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:

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

Most malware tends to attack to a specific operating system. We can know the operating system on which a packet was sent by checking the TTL value. In this observation, we consider that the difference of the receiving values of TTL is important.

- 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:

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

- 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:

$$\begin{aligned} d_{ip}(x_1, x_2) &= d_{sadr}(x_1, x_2) + d_{dadr}(x_1, x_2) \\ &\quad + d_{ttl}(x_1, x_2) + d_{id}(x_1, x_2). \end{aligned}$$

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
  It is well known that there are a number of ports that have security vulnerabilities. Many worms use them to propagate themselves from an Internet device to another device. 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. If two packets had distinct destination ports, it is probable that the packets were not sent by the same malware. 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

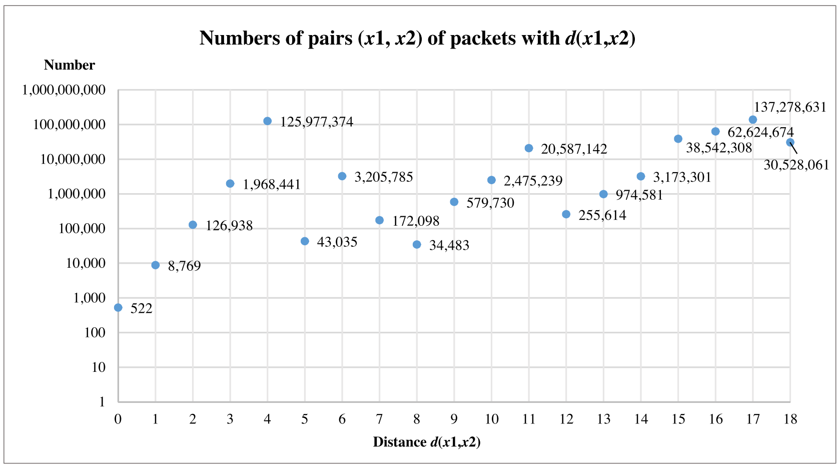


Fig. 1 Distribution of distances between all pairs of two packets in NICT darknet data collected in the 13th period on 11th July 2011 (TCP packets only, see the detail in Section 4.1).

```
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. 2 Algorithm MSSL (Min-cut Semi-Supervised Learner for Data Screening): Semi-supervised screening algorithm using minimum cut on graph.

by ICMP is defined as follows:

$$d_{icmp}(x_1, x_2) = d_{sport}(x_1, x_2) + d_{dport}(x_1, x_2) + d_{icmp\_code}(x_1, x_2).$$

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:

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

In **Fig. 1**, we give the distribution of the distances calculated on NICT darknet data collected in the 13th period on 11th July 2011 (TCP packets only). The data has 29,277 packets and thus 428,556,726 pairs of packets. As stated before, a NICT darknet exists in a class B network. Therefore the maximum distance between two packets in the data is 18, although the maximum distance in the definition is 20.

3.2 Semi-supervised Screening Algorithm Using Minimum Cut on Graph

Figure 2 shows the formal description of our algorithm, called Algorithm MSSL (Min-cut Semi-Supervised Learner for Data Screening), which 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 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

Table 4 The contents of the positive and negative data at the 13th period of 11th July 2011: These data were randomly chosen from the data detected by NMF-engine and the data not detected by NMF-engine, respectively.

	Number of packets	Numbers of packets to ports		
		22	23	3389
L_+ : whole data detected by NMF-engine	12,252	3,565	6,309	14
P : positive data (randomly chosen from L_+)	130	37	62	0
L_- : whole data not detected by NMF-engine	17,025	25	286	4
N : negative data (randomly chosen from L_-)	137	0	3	0

Table 5 The contents of the unlabeled and new positive data at the 14–48th periods of 11th July 2011: The new positive data was obtained from the unlabeled data by Algorithm MSSL.

	Number of packets	Numbers of packets to ports		
		22	23	3389
U : unlabeled data	355,495	35,242	8,657	16,893
U_+ : new positive data	222,327	29,013	8,513	777
Remaining rate (%)	37.46	17.64	1.66	95.40

negative. The labels indicate whether the packet is already known to be an attack or not. We used data labeled using the NMF-engine [7], [12]. 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 (06:00–06:29) on 11th July, where the 13th period 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 the malicious data detected by the NMF-engine to be positive data, which contained 12,252 packets. This whole positive data is denoted by L_+. We randomly chose 130 packets from L_+. This randomly chosen positive data is denoted by P. The second row of **Table 4** shows the contents of L_+ and P. P contained 37 and 62 packets bound for destination port number 22 and 23, respectively. However, P contained no packet bound for destination port number 3389, although L_+ contained 14 packets bound for 3389. We denote by L_- the set of all packets that were received during the 13th period of 11th July and not identified to be malicious by the NMF-engine. We randomly chose 137 packets from L_-. We set it to be N. The third row of Table 4 shows the contents of L_- and N. N contained 0, 3, and 0 packets bound for destination port number 22, 23, and 3389, respectively.

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.

Let $U^{(i)}$ be the unlabeled data consisting of packets received on the i-th period of the same day ($i = 14, 15, \ldots, 48$). Let U be the union of all the set $U^{(i)}$ for $i = 14, 15, \ldots, 48$. We applied Algorithm MSSL to labeled data $P \cup N$ and unlabeled data $U^{(i)}$ for every $i = 14, 15, \ldots, 48$. Let U_+ be the union of all the new positive data obtained from input $(P \cup N, U^{(i)})$ by Algorithm MSSL ($i = 14, 15, \ldots, 48$). To evaluate how many packets were still left in $U \setminus U_+$, 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 object of screening is to reduce the size of data that should be analyzed by early detection systems for network incidents. We consider that the smaller the size of remaining data after screening, the better the performance of early detections for network incidents. Therefore 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 5**. 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. We divided up the 24 hours worth of data into 48 periods. For $i = 3, 4, \ldots, 18$, let $U^{(i)}$ be the set of packets in the i-th period

Table 6 Summary of experiments.

Experiment	Input				Iteration
	Labeled data		Unlabeled data (X minutes' worth of data)		
1	$P^{(2)} \cup N^{(2)}$	fixed	$U^{(i)}$	30 minutes	for $i = 3$ to 18.
2	$P^{(i-1)} \cup N^{(i-1)}$	updated every iteration	$U^{(i)}$	30 minutes	for $i = 3$ to 18.
3 (a)	$P^{(2)} \cup N^{(2)}$	fixed	$U^{(i-1,i)} = U^{(i-1)} \cup U^{(i)}$	60 minutes	for $i = 5$ to 18.
3 (b)	$P^{(2)} \cup N^{(2)}$	fixed	$U^{(i-2,i-1,i)} = U^{(i-2)} \cup U^{(i-1)} \cup U^{(i)}$	90 minutes	for $i = 5$ to 18.

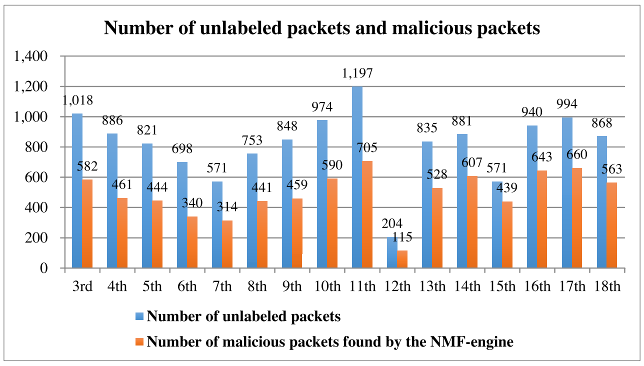


Fig. 3 International data collected in a darknet of the Maldives on 29th January 2014 (JST).

collected in this day. For example, $U^{(3)}$ is the set of packets collected in 01:00–01:29 of 29th January 2014. We used these sets as unlabeled data. We describe the detail of the numbers of unlabeled packets in the 3rd–18th periods in **Fig. 3**. These numbers are shown by blue bars in each bar graph. For example, 1,018 packets reached the darknet in the 3rd period. The total number of packets in all unlabeled data was 13,059. The maximum and minimum numbers of packets in each period were 1,197 and 204, respectively. We note that the 11th and the 12th period's data have the maximum and the minimum numbers of packets, respectively.

On 29th January 2014 (JST), the NMF-engine found malicious packets in all of the 2nd to 18th periods. For $i = 2, 3, \ldots, 18$, let $L_+^{(i)}$ be the set of packets in the i-th period of this day that were identified to be positive, i.e., malicious, by the NMF-engine, and let $L_-^{(i)}$ be the set of packets in the i-th period that were not identified to be malicious by the NMF-engine. We describe the detail of the numbers of those malicious packets in the 3rd–18th periods in Fig. 3. These numbers are shown by orange bars in the bar graph. For example, 582 packets in the 3rd period were identified to be malicious by the NMF-engine. The NMF-engine is an unsupervised method to decompose darknet traffic data into independent components for detecting malicious packets. That is, the target malware pattern of the NMF-engine is not fixed in advance. Thus any two labeled data $L_+^{(i)}$ and $L_+^{(j)}$ $(2 \le i < j \le 18)$ possibly consist of different kinds of malicious packets.

4.2.1 Experimental Details

We conducted the following three experiments 1–3.

Experiment 1 (Fixed labeled data):

At first, 149 positive labeled packets were selected randomly at a 25% rate from $L_+^{(2)}$. Let $P^{(2)}$ be the set of those packets. 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. Let $N^{(2)}$ be the set of those packets. In this experiment, the labeled data were fixed on $P^{(2)}$ and $N^{(2)}$ all the time. We applied Algorithm MSSL to the following 16 pairs of labeled and unlabeled data:

$$(P^{(2)} \cup N^{(2)}, U^{(i)})$$

for $i = 3, 4, \ldots, 18$, where $P^{(2)} \subset L_+^{(2)}$ and $N^{(2)} \subset L_-^{(2)}$.

Experiment 2 (Successively updated labeled data):

Let $P^{(i-1)}$ be a subset of the positive labeled packets selected randomly at a 25% rate from $L_+^{(i-1)}$, and let $N^{(i-1)}$ be a subset of the negative labeled packets selected randomly at a 25% rate from $L_-^{(i-1)}$. We applied Algorithm MSSL to the following 16 pairs of labeled and unlabeled data:

$$(P^{(i-1)} \cup N^{(i-1)}, U^{(i)})$$

where $P^{(i-1)} \subset L_+^{(i-1)}$ and $N^{(i-1)} \subset L_-^{(i-1)}$ for $i = 3, 4, \ldots, 18$. The labeled data were updated every period.

Experiment 3 (Unlabeled data collected in plural periods):

For any two positive integers i, j $(3 \le i < j \le 18)$, let $U^{(i,i+1,\ldots,j)} = U^{(i)} \cup U^{(i+1)} \cup \cdots \cup U^{(j)}$. Let $P^{(2)}$ be a subset of 149 positive labeled packets selected randomly at a 25% rate from $L_+^{(2)}$, and let $N^{(2)}$ be a subset of 170 negative labeled packets selected randomly at a 25% rate from $L_-^{(2)}$. These were the same positive and negative labeled data as the ones in Experiment 1. In this experiment, the labeled data were fixed on $P^{(2)}$ and $N^{(2)}$. We conducted the two experiments:

(a) Two period's worth of unlabeled data
We applied Algorithm MSSL to the following 14 pairs:
$$(P^{(2)} \cup N^{(2)}, U^{(i-1,i)}) \text{ for } i = 5, 6, \ldots, 18.$$

(b) Three period's worth of unlabeled data
We applied Algorithm MSSL to the following 14 pairs:
$$(P^{(2)} \cup N^{(2)}, U^{(i-2,i-1,i)}) \text{ for } i = 5, 6, \ldots, 18.$$

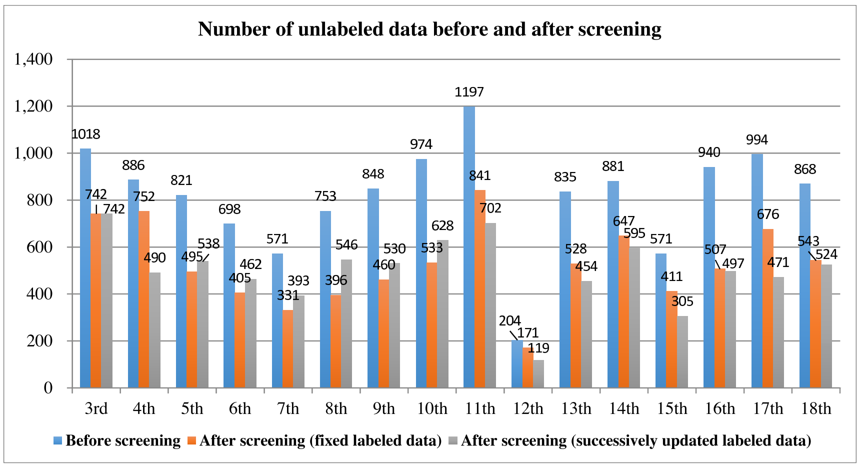
We summarize these experiments in **Table 6**. There is a significant difference between Experiment 2 and the other experiments in the point of view of target malware pattern data, i.e., positive labeled data. The main object of our screening method is to search the unlabeled data for packets that are near to given malware pattern data. In Experiment 2, the malware pattern data are not fixed. Therefore, it is probable that the new positive data output in Experiment 2 did not follow the target malware pattern detected in the 2nd period.

4.2.2 Evaluations

Experiment 1 (Fixed labeled data):

For a set S, we denote by $|S|$ the number of elements in S. We evaluated the precision, recall, and F-measure between the two kinds of the positive packets identified by the NMF-engine and Algorithm MSSL. Let $U_+^{(i)}$ be the set of packets that were identified to be positive by Algorithm MSSL on input $(P^{(2)} \cup N^{(2)}, U^{(i)})$. For $i = 3, 4, \ldots, 18$, the precision $p^{(i)}$, recall $r^{(i)}$, and F-measure $f^{(i)}$ are defined as follows:

$$p^{(i)} = |L_+^{(i)} \cap U_+^{(i)}| / |U_+^{(i)}|,$$
$$r^{(i)} = |L_+^{(i)} \cap U_+^{(i)}| / |L_+^{(i)}|,$$
$$f^{(i)} = 2 \cdot p^{(i)} \cdot r^{(i)} / (p^{(i)} + r^{(i)}).$$



	Minimum	Maximum	Whole data
Remaining rate (Experiment 1: fixed labeled data)	52.6%	84.9%	64.6%
Remaining rate (Experiment 2: successively updated labeled data)	47.4%	72.9%	61.2%

Fig. 4 Results of Experiments 1 and 2: This bar graph shows the numbers of unlabeled packets before and after screening.

In addition, we define the precision and recall of the whole unlabeled data:

$$p = |\bigcup_{i=3}^{18}(L_+^{(i)} \cap U_+^{(i)})|/|\bigcup_{i=3}^{18} U_+^{(i)}|,$$

$$r = |\bigcup_{i=3}^{18}(L_+^{(i)} \cap U_+^{(i)})|/|\bigcup_{i=3}^{18} L_+^{(i)}|.$$

Experiment 2 (Successively updated labeled data):

Let $W_+^{(i)}$ be the set of new labeled packets that were identified to be positive by Algorithm MSSL on input $(P^{(i-1)} \cup N^{(i-1)}, U^{(i)})$ for $i = 3, 4, \ldots, 18$. For $i = 3, 4, \ldots, 18$, the precision $p_1^{(i)}$, recall $r_1^{(i)}$, F-measure $f_1^{(i)}$ are defined as follows:

$$p_1^{(i)} = |L_+^{(i)} \cap W_+^{(i)}|/|W_+^{(i)}|,$$
$$r_1^{(i)} = |L_+^{(i)} \cap W_+^{(i)}|/|L_+^{(i)}|,$$
$$f_1^{(i)} = 2 \cdot p_1^{(i)} \cdot r_1^{(i)}/(p_1^{(i)} + r_1^{(i)}).$$

In order to compare this experiment with Experiment 1, we use the precision and recall of the whole unlabeled data:

$$p_1 = |\bigcup_{i=3}^{18}(L_+^{(i)} \cap W_+^{(i)})|/|\bigcup_{i=3}^{18} W_+^{(i)}|,$$

$$r_1 = |\bigcup_{i=3}^{18}(L_+^{(i)} \cap W_+^{(i)})|/|\bigcup_{i=3}^{18} L_+^{(i)}|.$$

Experiment 3 (Unlabeled data collected in plural periods):

Let $U_+^{(i-1,i)}$ (resp. $U_+^{(i-2,i-1,i)}$) be the set of new labeled packets that were identified to be positive by Algorithm MSSL on input $(P^{(2)} \cup N^{(2)}, U^{(i-1,i)})$ (resp. $(P^{(2)} \cup N^{(2)}, U^{(i-2,i-1,i)})$) for $i = 5, 6, \ldots, 18$. Let $L_+^{(i,i+1,\ldots,j)} = L_+^{(i)} \cup L_+^{(i+1)} \cup \cdots \cup L_+^{(j)}$.

(a) Two period's worth of unlabeled data

For $i = 5, \ldots, 18$, the precision $p_2^{(i)}$, recall $r_2^{(i)}$, and F-measure $f_2^{(i)}$ of two period's worth of unlabeled data are defined as follows:

$$p_2^{(i)} = |L_+^{(i-1,i)} \cap U_+^{(i-1,i)}|/|U_+^{(i-1,i)}|,$$
$$r_2^{(i)} = |L_+^{(i-1,i)} \cap U_+^{(i-1,i)}|/|L_+^{(i-1,i)}|,$$
$$f_2^{(i)} = 2 \cdot p_2^{(i)} \cdot r_2^{(i)}/(p_2^{(i)} + r_2^{(i)}).$$

(b) Three period's worth of unlabeled data

For $i = 5, \ldots, 18$, the precision $p_3^{(i)}$, recall $r_3^{(i)}$, and F-measure $f_3^{(i)}$ of three period's worth of unlabeled data are defined as follows:

$$p_3^{(i)} = |L_+^{(i-2,i-1,i)} \cap U_+^{(i-2,i-1,i)}|/|U_+^{(i-2,i-1,i)}|,$$
$$r_3^{(i)} = |L_+^{(i-2,i-1,i)} \cap U_+^{(i-2,i-1,i)}|/|L_+^{(i-2,i-1,i)}|,$$
$$f_3^{(i)} = 2 \cdot p_3^{(i)} \cdot r_3^{(i)}/(p_3^{(i)} + r_3^{(i)}).$$

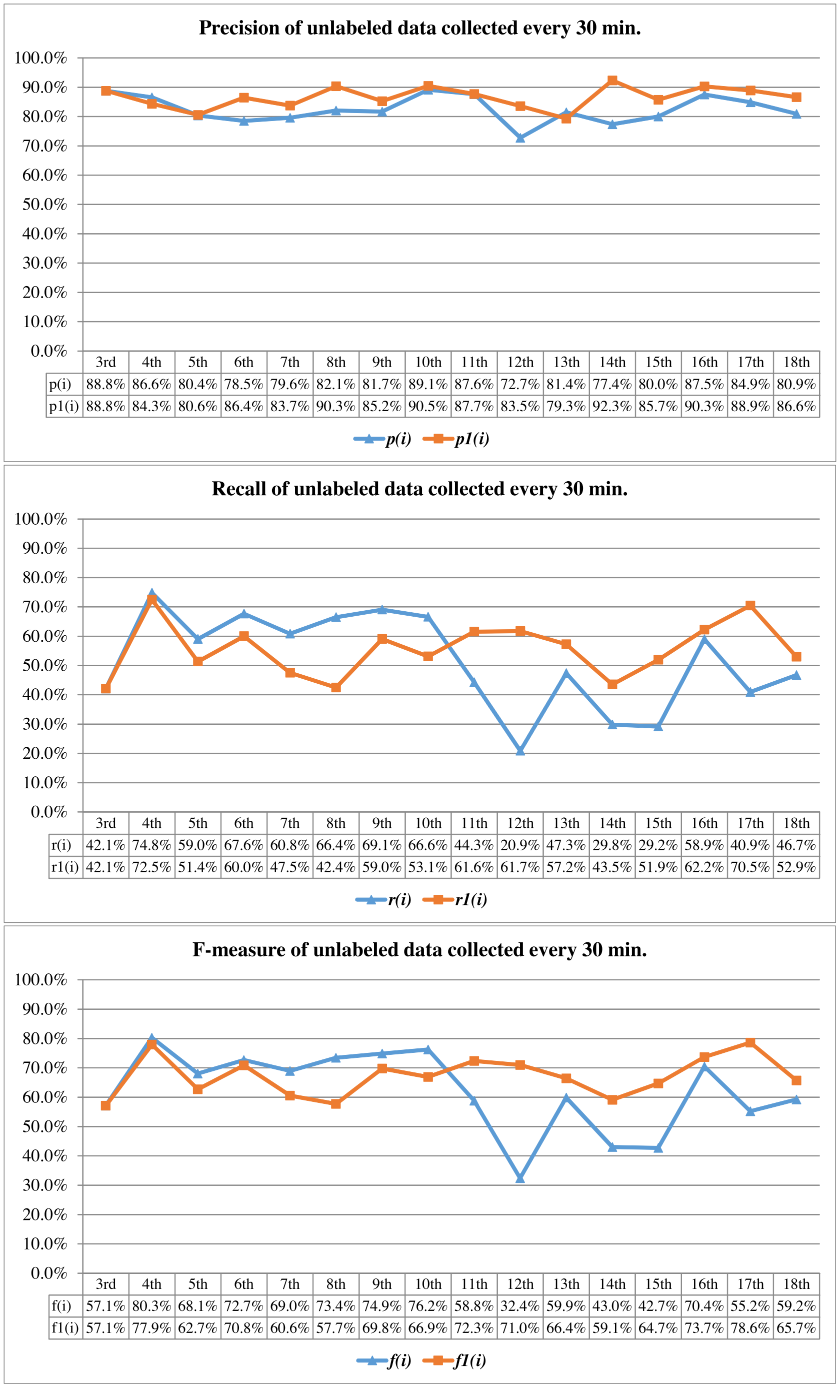
4.2.3 Results and Discussion

Figure 4 shows the numbers of unlabeled packets before and after screening in Experiments 1 and 2. The bar graph shows the number of unlabeled packets before screening (blue bars) and the number of unlabeled packets after Experiment 1 (orange bars) and Experiment 2 (gray bars). From this figure, we can observe that after the 11th period, every remaining rate of Experiment 2 is better than that of Experiment 1. In Experiment 2, we updated labeled data successively. Therefore, we consider that after the 11th period, the packets in $U^{(i)}$ $(i = 11, 12, \ldots, 18)$ are closer to the packets in $L_+^{(11)}$ than $L_+^{(2)}$ with respect to the distance defined in Section 3.1.

Figure 5 shows the transitions of precision, recall, and F-measure in Experiments 1 and 2. In the top line graph in Fig. 5, the blue line represents the precision $p^{(i)}$, and the orange represents the precision $p_1^{(i)}$ $(i = 3, 4, \ldots, 18)$. We also give the two line graphs in the middle and bottom of Fig. 5 that represent the recall $r^{(i)}$, $r_1^{(i)}$ and F-measure $f^{(i)}$, $f_1^{(i)}$ $(i = 3, 4, \ldots, 18)$. The detailed values of the precision, recall, and F-measure are indicated in the data table of each line graph. The precision and recall of the whole unlabeled data in Experiment 1, i.e., p and r, can be calculated in the following way:

$$p = |\bigcup_{i=3}^{18}(L_+^{(i)} \cap U_+^{(i)})|/|\bigcup_{i=3}^{18} U_+^{(i)}| = \sum_{i=3}^{18}|L_+^{(i)} \cap U_+^{(i)}|/\sum_{i=3}^{18}|U_+^{(i)}|$$
$$= \sum_{i=3}^{18}(|U_+^{(i)}| \cdot p^{(i)})/\sum_{i=3}^{18}|U_+^{(i)}|.$$

The values $|U_+^{(i)}|$ and $p^{(i)}$ $(i = 3, 4, \ldots, 18)$ can be found in Fig. 3



	3rd	4th	5th	6th	7th	8th	9th	10th	11th	12th	13th	14th	15th	16th	17th	18th
p(i)	88.8%	86.6%	80.4%	78.5%	79.6%	82.1%	81.7%	89.1%	87.6%	72.7%	81.4%	77.4%	80.0%	87.5%	84.9%	80.9%
p1(i)	88.8%	84.3%	80.6%	86.4%	83.7%	90.3%	85.2%	90.5%	87.7%	83.5%	79.3%	92.3%	85.7%	90.3%	88.9%	86.6%

	3rd	4th	5th	6th	7th	8th	9th	10th	11th	12th	13th	14th	15th	16th	17th	18th
r(i)	42.1%	74.8%	59.0%	67.6%	60.8%	66.4%	69.1%	66.6%	44.3%	20.9%	47.3%	29.8%	29.2%	58.9%	40.9%	46.7%
r1(i)	42.1%	72.5%	51.4%	60.0%	47.5%	42.4%	59.0%	53.1%	61.6%	61.7%	57.2%	43.5%	51.9%	62.2%	70.5%	52.9%

	3rd	4th	5th	6th	7th	8th	9th	10th	11th	12th	13th	14th	15th	16th	17th	18th
f(i)	57.1%	80.3%	68.1%	72.7%	69.0%	73.4%	74.9%	76.2%	58.8%	32.4%	59.9%	43.0%	42.7%	70.4%	55.2%	59.2%
f1(i)	57.1%	77.9%	62.7%	70.8%	60.6%	57.7%	69.8%	66.9%	72.3%	71.0%	66.4%	59.1%	64.7%	73.7%	78.6%	65.7%

Fig. 5 Results of Experiments 1 and 2: In the line graphs, we use notations $p(i)$, $p1(i)$, $r(i)$, $r1(i)$, $f(i)$, $f1(i)$ instead of $p^{(i)}$, $p_1^{(i)}$, $r^{(i)}$, $r_1^{(i)}$, $f^{(i)}$, $f_1^{(i)}$, respectively.

and the top data table of **Fig. 6**. Finally we have $p = 83.4\%$ as a result of this expression. In a similar way, we have $r = 50.8\%$.

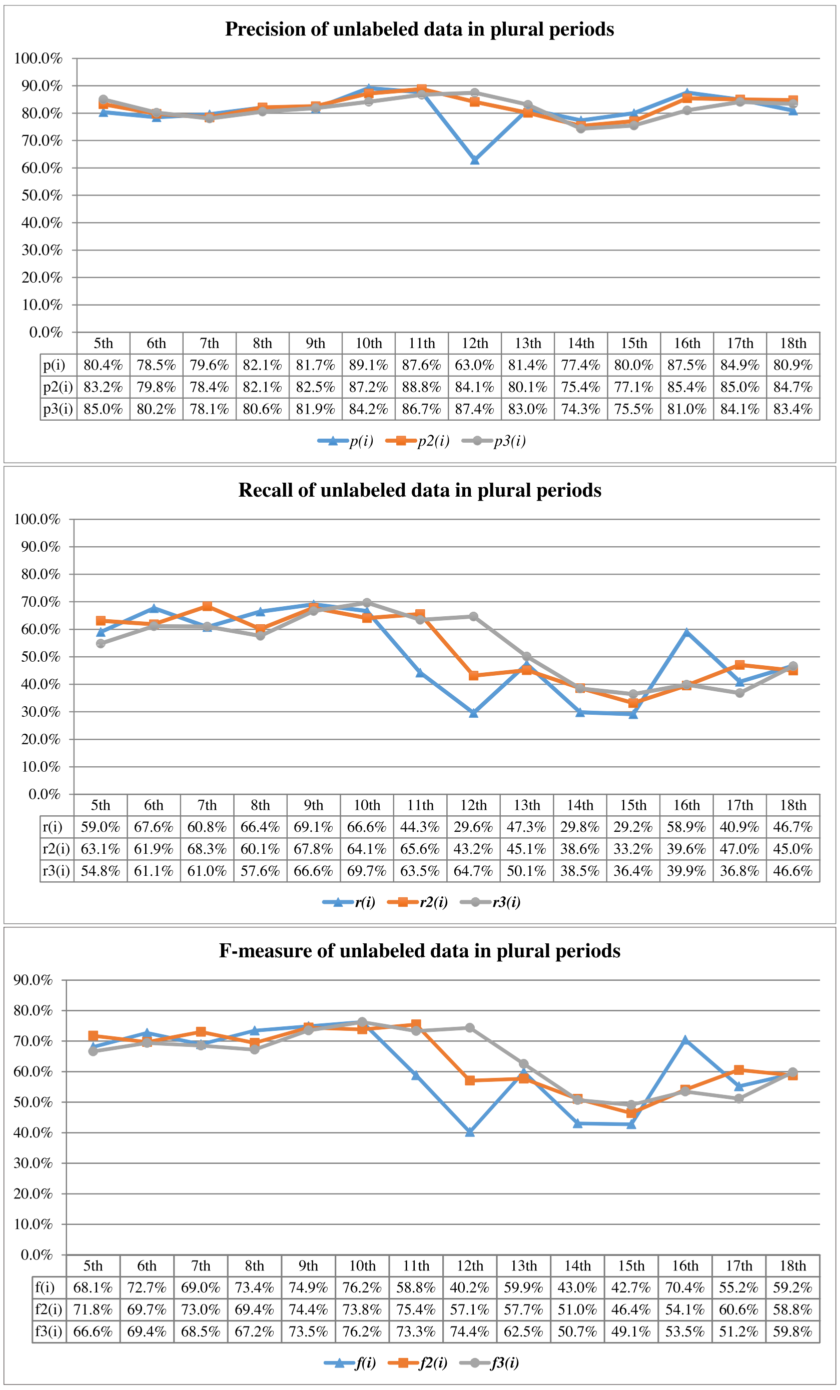
Generally speaking, a screening traffic data method will not remove packets originating from unknown malware. Therefore, we organized the parameters so as to keep the precision high. As a result of Experiment 1, we consider that the precision is good enough. However, the recall is far worse than the precision. We note that the the number of packets contained in $U^{(12)}$ radically decreased from the number of packets in $U^{(11)}$. This large decrease was not unrelated to the change of property (information on packets, e.g., source IP address, TTL, etc) of the data. We observe that the change of property of the data affects the recall of unlabeled data (shown by the blue line in the middle line graph in Fig. 5). Actually, the recall of unlabeled data decreases extremely along the 10–12th periods. The following two reasons are considered as the cause that the recall decreased:

(1) We used the same labeled data $L_+ \subseteq L_+^{(2)}$ all the time.

(2) The semi-supervised learning has less efficiency on small unlabeled data.

Since malware attacks are being replaced at faster rates, accuracy with fixed labeled data is considered bad.

In Experiment 2, we updated the labeled data every period.



	5th	6th	7th	8th	9th	10th	11th	12th	13th	14th	15th	16th	17th	18th
p(i)	80.4%	78.5%	79.6%	82.1%	81.7%	89.1%	87.6%	63.0%	81.4%	77.4%	80.0%	87.5%	84.9%	80.9%
p2(i)	83.2%	79.8%	78.4%	82.1%	82.5%	87.2%	88.8%	84.1%	80.1%	75.4%	77.1%	85.4%	85.0%	84.7%
p3(i)	85.0%	80.2%	78.1%	80.6%	81.9%	84.2%	86.7%	87.4%	83.0%	74.3%	75.5%	81.0%	84.1%	83.4%

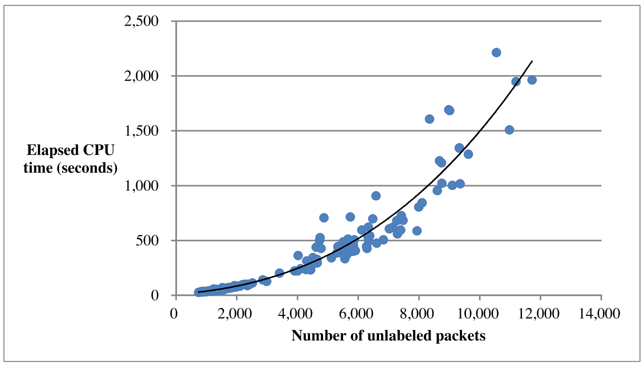
	5th	6th	7th	8th	9th	10th	11th	12th	13th	14th	15th	16th	17th	18th
r(i)	59.0%	67.6%	60.8%	66.4%	69.1%	66.6%	44.3%	29.6%	47.3%	29.8%	29.2%	58.9%	40.9%	46.7%
r2(i)	63.1%	61.9%	68.3%	60.1%	67.8%	64.1%	65.6%	43.2%	45.1%	38.6%	33.2%	39.6%	47.0%	45.0%
r3(i)	54.8%	61.1%	61.0%	57.6%	66.6%	69.7%	63.5%	64.7%	50.1%	38.5%	36.4%	39.9%	36.8%	46.6%

	5th	6th	7th	8th	9th	10th	11th	12th	13th	14th	15th	16th	17th	18th
f(i)	68.1%	72.7%	69.0%	73.4%	74.9%	76.2%	58.8%	40.2%	59.9%	43.0%	42.7%	70.4%	55.2%	59.2%
f2(i)	71.8%	69.7%	73.0%	69.4%	74.4%	73.8%	75.4%	57.1%	57.7%	51.0%	46.4%	54.1%	60.6%	58.8%
f3(i)	66.6%	69.4%	68.5%	67.2%	73.5%	76.2%	73.3%	74.4%	62.5%	50.7%	49.1%	53.5%	51.2%	59.8%

Fig. 6 Results of Experiments 1, 3(a) and 3(b): Transitions of precision, recall, and F-measure using data labeled by MSSL in plural periods. (In these line graphs, we use notations $p(i)$, $p2(i)$, $p3(i)$, $r(i)$, $r2(i)$, $r3(i)$, $f(i)$, $f2(i)$, $f3(i)$ instead of $p^{(i)}$, $p_2^{(i)}$, $p_3^{(i)}$, $r^{(i)}$, $r_2^{(i)}$, $r_3^{(i)}$, $f^{(i)}$, $f_2^{(i)}$, $f_3^{(i)}$, respectively.)

The precision of the whole unlabeled data in Experiment 2 was $p_1 = 86.8\%$. This rate is better than the precision of whole unlabeled data in Experiment 1 ($p = 83.4\%$). According to Fig. 5, in the latter half of the periods, the recall of Experiment 2 were better than that of Experiment 1. Actually, the recall of the whole unlabeled data was $r_1 = 55.7\%$. That is, the recall of the whole unlabeled data improved from that of Experiment 1. Thus we conclude that updating the labeled data improved learning accuracy. Particularly, at the 11th and 12th periods of the graph of the recall (the middle graph) in Fig. 5, we see that $r_1^{(11)}$ and $r_1^{(12)}$ are better than $r^{(11)}$ and $r^{(12)}$, respectively. This fact shows that the distribution of the distances between any two packets in $L_+^{(11)}$ and $U^{(12)}$ was different from that of $L_+^{(2)}$ and $U^{(12)}$. We consider that most of the packets in $U^{(12)}$ are closer to the packets in $L_+^{(11)}$ than $L_+^{(2)}$.

As a result of Experiment 3, we give a line graph in the top of Fig. 6, which represents the precision $p^{(i)}$, $p_2^{(i)}$, and $p_3^{(i)}$ ($i = 5, \dots, 18$). We also give two line graphs in the middle and bottom of Fig. 6, which represent the recall $r^{(i)}$, $r_2^{(i)}$, and $r_3^{(i)}$, and F-measure $f^{(i)}$, $f_2^{(i)}$, and $f_3^{(i)}$ ($i = 5, \dots, 18$). We also give the detailed values of the precision, recall, and F-measure in the data



	Minimum	Maximum	Average
Number of packets	750	11,724	4,527.47
Execution time (s)	27.44	2,212.25	434.02

Fig. 7 Distribution of the execution time (in elapsed CPU seconds) for Algorithm MSSL with respect to 120 unlabeled data obtained from 5 international darknets on 1st June 2015 (GMT).

table of each line graph in Fig. 6. We see that $r_2^{(11)}$ and $r_3^{(11)}$ are better than $r^{(11)}$. We also see that $r_3^{(12)}$ is improved well comparing with $r^{(12)}$. In order to observe it more precisely, we calculate the recall of the union of the unlabeled data in the 10th–12th periods that were processed in every period independently by Algorithm MSSL, which is represented by the following expression:

$$|\bigcup_{i=10}^{12}(L_+^{(i)} \cap U_+^{(i)})|/|\bigcup_{i=10}^{12} L_+^{(i)}| = \sum_{i=10}^{12}|L_+^{(i)} \cap U_+^{(i)}|/\sum_{i=10}^{12}|L_+^{(i)}|$$

$$= \sum_{i=10}^{12}(|L_+^{(i)}| \cdot r^{(i)})/\sum_{i=10}^{12}|L_+^{(i)}|.$$

Since the values $|L_+^{(i)}|$ and $r^{(i)}$ $(i = 10, 11, 12)$ can be found in Fig. 3 and the middle data table of Fig. 6, we have 52.4% as the value of this expression. The recall of the 12th period of Experiment 3 (b), i.e., $r_3^{(12)} = 64.7\%$, is really better than the value 52.4%. Thus, we consider that Algorithm MSSL utilizes well the distances between two packets in the 10–12th periods so as to connect 10th period's packets with 12th ones.

We conclude that if we can successively update positive labeled data that have the same (or similar) malware pattern, we can obtain new labeled data with good precision and recall. However, in a real situation like darknet traffic data, it is often hard to obtain new labeled data of the same malware pattern constantly. Even in such a case, Algorithm MSSL utilizes well the information of unlabeled data if it uses unlabeled data that is larger than labeled data.

4.3 Computational Time

In this section, we show the execution time (in elapsed CPU seconds) for Algorithm MSSL. We implemented Algorithm MSSL with Dinic's $O(n^2m)$ algorithm [2], where n and m are the numbers of vertices and edges in a given graph, respectively. Algorithm MSSL also runs in cubic time with respect to the number of packets. In Section 4.2, we carried out Experiments 1, 2, 3(a), and 3(b) and in them, we suggested three methods how to operate Algorithm MSSL on real darknet data. From the theoretical viewpoint, the asymptotic complexities of the three methods are the same.

We ran our algorithm on unlabeled data obtained by monitoring darknets in 5 countries all over the world on 1st June 2015 (GMT). The experiments were run on the following computer environment: CPU: Intel Xeon E5-2407 2.20 GHz (4 cores 4 threads) × 2, Memory: 64 GB, HDD: 1.7 TB, OS: CentOS 6.5. We used 120 unlabeled data, each of which had 1 hour's worth of packets. The scatter plot in **Fig. 7** shows the relationship between the number of packets contained in unlabeled data and its execution time. In Fig. 7, we give the cubic curve that approximates the points in the scatter plot. We consider that the performance of this implementation is fast enough, because the algorithm processed 1 hour's worth of packets within 1 hour (maximum execution time: 2,212.25 s).

5. Conclusion

We proposed a screening method using semi-supervised learning based on graphs proposed by Blum and Chawla [1]. Screening experiments were conducted on real darknet traffic data using positive and negative data labeled by the NMF-engine. We mainly ran Algorithm MSSL on the following three cases: (1) We fixed labeled data all the time (Experiment 1). (2) We updated labeled data successively (Experiment 2). (3) We fixed labeled data and used unlabeled data that was relatively large (Experiments 3 (a) and (b)).

If we can successively update positive labeled data that have the same (or similar) malware pattern, we can obtain new labeled data with good precision and recall from unlabeled data. However, in a real situation like darknet traffic data, it is often hard to obtain new labeled data of the same malware pattern constantly. Even in such a case, Algorithm MSSL utilizes well the information of unlabeled data if it uses unlabeled data that is larger than labeled data.

At this moment, we consider that the performance of our implementation is fast enough. However, in order to process huge amount of data fast and efficiently, we are now planing to implement Algorithm MSSL again with a practically efficient maximum flow/minimum cut algorithm. We are also developing an efficient and effective screening method for darknet traffic data by using the other semi-supervised learning algorithms.

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, *Proc. 18th International Conference on Machine Learning* (*ICML2001*), pp.19–26 (2001).

[2] Dinitz, Y.: Algorithm for solution of a problem of maximum flow in a network with power estimation, *Soviet Mathematics Doklady*, Vol.11, pp.1277–1280 (1970).

[3] Eto, M., Inoue, D., Suzuki, M. and Nakao, K.: A Statistical Packet Inspection for Extraction of Spoofed IP Packets on Darknet, *Proc. 4th*

Joint Workshop on Information Security (*JWIS 2009*) (2009).

[4] Hatada, M., Akiyama, M., Matsuki, T. and Kasama, T.: Empowering Anti-malware Research in Japan by Sharing the MWS Datasets, *IPSJ Journal of Information Processing*, Vol.23, No.5, pp.579–588 (2015).

[5] Kamizono, M., Akiyama, M., Kasama, T., Murakami, J., Hatada, M. and Terada, M.: Datasets for Anti-Malware Research – MWS Datasets 2015– (in Japanese), *The Information Processing Society of Japan, IPSJ SIG Technical Report*, Vol.2015-CSEC-70, No.6, pp.1–8 (2015).

[6] Kasama, T. and Kamizono, M.: NICTER Darknet Dataset 2014/NONSTOP, *anti Malware engineering WorkShop 2014* (*MWS 2014*) (2014), available from ⟨http://www.iwsec.org/mws/2014/files/NICTER_Darknet_Dataset_2014.pdf⟩ (in Japanese).

[7] 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*, Vol.113, No.288, ICSS2013-61, pp.23–28 (2013).

[8] Nakazato, J. and Ohtaka, K.: nicter Report – Transition Analysis of Cyber Attacks Based on Long-term Observation, *Journal of the National Institute of Information and Communications Technology*, Vol.58, No.3/4, pp.27–34 (2011).

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

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

[11] 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).

[12] Yamauchi, S., Kawakita, K. and Takeuchi, J.: Botnet detection based on non-negative matrix factorization and the MDL principle, *Proc. 19th International Conference on Neural Information Processing* (*ICONIP2012*), *Part V, Springer, Lecture Notes in Computer Science*, Vol.7667, pp.400–409 (2012).

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

Takayoshi Shoudai received his B.S. in 1986, M.S. degrees in 1988 in Mathematics and Dr. Sci. in 1993 in Information Science from Kyushu University. Currently, he is a professor in the Faculty of International Studies, Kyushu International University. His research interests include algorithmic learning theory, knowledge discovery, and machine learning. He is a member of the IEICE, ACM, JSIAM, and MSJ.

Hikaru Murai received his B.S. degree in Physics and M.S. degree in Informatics from Kyushu University in 2013 and 2015. He is currently with Treasury & Capital Markets System Solutions Division, Financial System Solutions Bureau, NS Solutions Corporation.

Atsushi Okamoto received his B.S. degree in Physics and M.S. degree in Informatics from Kyushu University in 2011 and 2013. He was a research assistant of Information Security Laboratory, Institute of Systems, Information Technologies and Nanotechnologies (ISIT), Japan until March 2016.