Journal of Information Processing Vol.21 No.3 527–538 (July 2013)

[DOI: 10.2197/ipsjjip.21.527]

Regular Paper

A Behavior-based Method for Detecting Distributed Scan Attacks in Darknets

Yaokai Feng[1,2,a)] Yoshiaki Hori[1,2,b)] Kouichi Sakurai[1,2,c)] Jun'ichi Takeuchi[1,2,d)]

Received: October 22, 2012, Accepted: March 1, 2013

Abstract: The technologies used by attackers in the Internet environment are becoming more and more sophisticated. Of the many kinds of attacks, distributed scan attacks have become one of the most serious problems. In this study, we propose a novel method based on normal behavior modes of traffic to detect distributed scan attacks in darknet environments. In our proposed method, all the possible destination TCP and UDP ports are monitored, and when a port is attacked by a distributed scan, an alert is given. Moreover, the alert can have several levels reflecting the relative scale of the attack. To accelerate learning and updating the normal behavior modes and to realize rapid detection, an index is introduced, which is proved to be very efficient. The efficiency of our proposal is verified using real darknet traffic data. Although our proposal focuses on darknets, the idea can also be applied to ordinary networks.

Keywords: distributed scan, anomaly detection, cyber-attack, collaborative attack, behavior-based methods, darknets

1. Introduction

1.1 Background

The frequency and extent of damages caused by network attacks have increased greatly in recent years, despite the many approaches for avoiding and detecting attacks proposed by the network security community. The basic reason for this is that the technologies used by attackers are also becoming more sophisticated.

A common difficulty of research in the network security field is the fact that real traffic data in many companies and other organizations are often not available to researchers because of the strict rules and laws (e.g., those for protecting personal or the company's privacy). Fortunately, it has been confirmed by many studies that global trends in network threats can be observed by monitoring darknets [1], [2], [3], [4]. A darknet is a set of unused IP addresses [5], in which no actual services (web, mail, and so on) exist, since these addresses have not been distributed to any legal users. Thus, except for misconfigurations in the sources, all traffic to darknets can be regarded as anomalies.

Collaborative attacks are well known among sophisticated attacks, and are often launched by multiple attackers (i.e., human attackers or criminal organizations). They are referred to as next generation cyber-attacks in Ref. [6], although the distributed denial-of-service (DDoS) attacks that already exist can be seen as simple examples of collaborative attacks, in that they involve many compromised computers. It has been stated that existing DDoS attacks from botnets have caused the most serious losses in the modern Internet environment. Thus, how to detect and avoid collaborative attacks has become one of the most important topics.

Scan attackers try to find servers or services with vulnerabilities [7]. Scan attacks are often classified into the following four distinct groups. 1) Vertical scan: a single source sequentially scans many or all ports of one destination IP address. The attacker is interested in a particular host, and wishes to characterize the services on it. 2) Horizontal scan: a single source sequentially scans a single destination port of many destination IP addresses. The attackers are interested in finding any hosts that expose some service. Thus, they scan the port of interest on all IP addresses within a certain range of interest. This has become one of the most common types of port scan [8]. Because such attacks often use a range of destination hosts instead of a single host, they often drop into darknets. 3) Distributed vertical scan: multiple sources sequentially scan many ports of one destination IP address. 4) Distributed horizontal scan: multiple sources sequentially scan a single destination port of many IP addresses. Note that in distributed scans, the actual IP addresses of attackers could change frequently to make detection difficult.

The last two kinds of scan attacks are related to collaborative attacks. The third one in darknets means that multiple attackers simultaneously "shoot" at a specific unused IP address, which, we think, only happens when attackers have misconfigured their systems. In other words, there are no actual distributed vertical scans in darknets. Thus, henceforth in this study, the term distributed scan in darknets refers to a distributed horizontal scan (i.e., the fourth kind of scan mentioned above).

1.2 Our Goal

In this study, we attempt to detect the above-mentioned dis-

1 Graduate School of Information Science and Electrical Engineering, Kyushu University, Fukuoka 819–0395, Japan
2 Institute of Systems, Information Technologies and Nanotechnologies (ISIT), Fukuoka 814–0001, Japan
a) fengyk@ait.kyushu-u.ac.jp
b) hori@csce.kyushu-u.ac.jp
c) sakurai@csce.kyushu-u.ac.jp
d) tak@inf.kyushu-u.ac.jp

© 2013 Information Processing Society of Japan

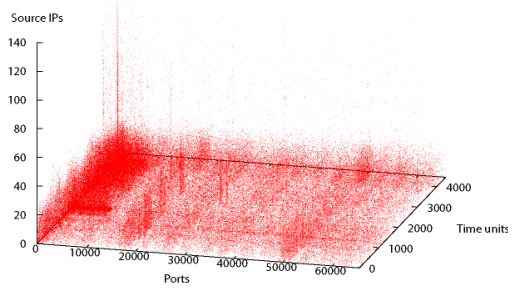


Fig. 1 Traffic data distribution in a darknet (2009.5.1 to 5.31, TCP).

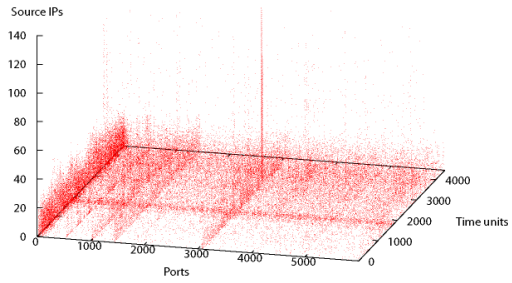


Fig. 2 Locally enlarged image of Fig. 1.

tributed scan attacks efficiently, i.e., detect the attacks in which multiple sources collaboratively attack one port, possibly on different destination IP addresses. The problem of multiple sources collaboratively attacking multiple ports on multiple destination IP addresses can be simply solved by monitoring all the ports to detect the distributed scan attacks. In fact, all possible destination ports are monitored in this study. Note that henceforth, ports refer to destination ports.

Visualization of an example can help to understand distributed scans. For this purpose, we investigated all the TCP packets sent to the /24 darknet of our campus network between May 1st and May 31st (one month) in 2009 (a total of 2,187,086 TCP packets). The investigation results are shown in **Fig. 1**, where the x-axis denotes all the possible destination TCP ports (0–65,535), the y-axis represents the time units, and the z-axis indicates the different source IP addresses. Note that in this investigation, 1) one time unit is 10 minutes, which means that the packets are accumulated every 10 minutes, and 2) all the IP addresses are locally numbered. Concretely, the source IP addresses are numbered from 0 according to the order they occur in each time unit. The different IP addresses are given different numbers in the same time unit. However, the numbering processes in different time units are independent. **Figure 2** is a locally enlarged image of Fig. 1.

From Fig. 1 and Fig. 2, we can clearly see that at a certain number of time units, a destination port was seemingly attacked by more than 100 sources. The details of the detection results are discussed in Section 4. In this study, it is not regarded as an anomaly if the number of sources sending packets to one port in a single time unit is not large. As mentioned above, all the packets coming to darknets are generally dubious. However, considering the possibility of misconfiguration of the sources and the huge scale of potential sources, it should be tolerated if only a few sources send packets to the monitored ports in a time unit. Note that the extent to which we can tolerate this can be adjusted

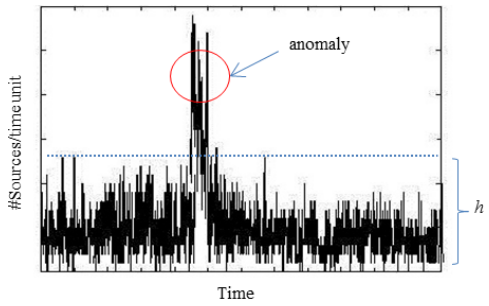


Fig. 3 Example of behavior-based methods.

by the user. Of course, zero tolerance can also be realized. It is certainly regarded as a distributed scan attack if the number of sources suddenly increases markedly in a certain time unit, such as the vertical "poles" in Fig. 1 and Fig. 2. Our goal in this study is merely to detect such distributed scans using a behavior-based method.

1.3 Behavior-based Method

Figure 3 shows an example of a behavior-based method. In this figure, the horizontal axis represents the time while the vertical axis denotes, for example, the number of sources sending packets to a port in each time unit. In Fig. 3, h can be regarded as the normal range of the number of sources sending packets to the monitored port in a single time unit. However, the traffic in the circle is seemingly an anomaly. If we can extract the normal behavior mode from the data over a relatively long time period, then this normal behavior mode can be used to detect anomalies.

Normal behavior modes can be obtained if we have sufficient clean learning data. In this study, because we do not have such clean data, we propose an algorithm for detecting outliers in the learning data, which will give our learning phase a self-cleaning ability to some extent.

1.4 Our Contributions

In this paper, a practical solution is presented for the first time based on normal behavior modes of traffic to detect distributed scan attacks in darknets and the performance of our proposal is verified using actual traffic data in the darknet of our campus network. Although our proposal focuses on darknets, this approach can, in fact, also be used in ordinary networks. Our approach mainly has the following advantages. 1) All the 65,536 possible TCP and 65,536 possible UDP ports are monitored and an alert is given when a port is attacked via a distributed scan attack. Moreover, in our proposal, it is easy to associate an anomaly score (called the alert level in this study) with a detection instance. The alerts may have different levels reflecting the relative scales of the attacks to the normal behavior modes of the corresponding ports. 2) For each port, a normal behavior mode learned from historical data is used to determine if a distributed port scan attack has occurred, unlike existing methods that simply use a predefined threshold. 3) An index structure (called the BH-index) is introduced to accelerate the learning and detection phases. All the necessary data for all the ports are managed through this index. When the information of one port is needed, only one leaf node has to be accessed and the others can be skipped, thereby accel-

© 2013 Information Processing Society of Japan

erating the learning and detection phases (see Section 3.6 for the details).

This paper is organized as follows. After the related work is introduced in Section 2, the details of our proposal are explained in Section 3. Then, in Section 4 we present our experimental results. Finally, in Section 5, we state our conclusions.

2. Related Work

To detect network attacks, many approaches have been proposed, including signature-based methods [9], volume-based methods [10], histogram-based methods [11], [12], and so on. However, as is known, signature-based methods are not efficient for new variants or new kinds of attacks, because they can only detect anomalies stored in the database in advance. Volume-based methods cannot detect low-rate attacks because such attacks can hide themselves in normal traffic [13]. Histogram-based methods often have very high false negative rates [13].

Several methods based on information theory have also been proposed [14], [15]. These, however, suffer from the following disadvantages: their performance is highly dependent on the choice of the information theoretic measure, they are efficient only when there are a significantly large number of anomalies present in the data, and it is difficult to associate an anomaly score with a test instance [16]. Stoecklin et al. [17] presented a novel technique for detecting traffic anomalies based on network flow normal behavior modes built into different traffic features. In addition, to detect low-rate attacks, Fukushima et al. [18] tried to find anomalies by classifying the traffic from various different viewpoints. Another study [19], [20] proposed a signature-based method for detecting low-rate attacks.

Kim et al. [21] proposed a method for finding scan attacks and TCP SYN flood attacks by detecting change-points based on the characteristics of the packets, IP addresses, and ports. The session information based on the actions of communication protocols such as TCP and UDP was also used in Ref. [22] to detect scan attacks, DoS, DDoS, and backscatter. Eto et al. [1] created a system called Nicter, which detects network attack threats by monitoring darknets. The number of IP addresses in the darknet monitored by Nicter has reached 140,000.

Behavior-based methods have two important advantages: 1) the false negative rate is low, and 2) detection is fast because the detection process is merely a simple value-comparison between the counting result in the latest time unit and the normal behavior mode of the current port. Despite the large number of methods already having been proposed, no behavior-based method exists for detecting distributed scans. In our previous work [13], a method based on normal behavior modes was proposed to detect outbreaks of low-rate attacks. The method determined whether the packets sent from each of the sources to the monitored port in a single time unit was a low-rate communication. Thereafter, if the number of sources sending low-rate packets was greater than normal (as learned from historical data), an outbreak of low-rate attacks was deemed to have occurred. In the current study, if the number of sources sending packets to the corresponding port is greater than the normal behavior mode of this port then an alarm is given, which is similar to the work [13]. However, in this study, all the possible ports are automatically monitored, distributed port scan attacks in darknets can be detected, and an index structure is introduced to implement fast learning and rapid detection. Thus, it can be said that the approach in Ref. [13] is extended in this study and adapted to detect distributed scans in darknet environments.

Generally, approaches based on normal behavior modes have several disadvantages: 1) much time is needed to construct the normal behavior modes for all the ports, and 2) the normal behavior modes need to be updated sometime, which is also time consuming. These two disadvantages are overcome in the current study by using an index, which proves to be very efficient. A further disadvantage is that normal behavior modes are often difficult to obtain if there are no clean learning data. To overcome this problem, we propose an algorithm, which has a self-cleaning ability to some extent, for extracting normal behavior modes.

Existing port scan detection approaches can be divided into three categories: threshold-based, algorithmic, and visual. Visual approaches focus on presenting the data to users in some visual manner so that they can recognize a scan by the pattern it generates. An algorithmic approach was proposed in Ref. [23] to detect port scanning activities. This approach generates graphs representing the communication patterns observed in a network, where the traffic added to a particular graph is determined based on similarity in time and network geography. Events can be identified based on the topography of the graphs. However, threshold-based approaches are still the mainstream methods. All three intrusion detection systems in the public domain, namely Snort, Bro and NSM, use threshold-based approaches for detecting port scans. For example, Snort observes connections to determine whether a scan is occurring. By default, Snort is configured to generate an alarm only when 20 different ports are observed within 60 seconds, although this can be adjusted manually. Note that, Snort does not detect distributed port scans [24], [25]. Another example is the study in Ref. [26], which defined distributed port scans as being "scans from multiple sources (5 or more) aimed at a particular port of destinations in the same /24 subnet within a one hour window." In the threshold-based methods, the thresholds have to be determined in advance and play a key role in detection efficiency. However, since there is no easy way of determining these thresholds, users have to consider the actual networks and rely on their own experiences.

3. Our Proposal

An overview of our proposal is given in **Table 1**. The first four steps constitute the general learning method, while step 5 considers how to speed up the learning phase using an index. This process is explained step by step in this section.

Table 1 Overview of the proposed method.

Steps		Descriptions
Learning	1	Collect and arrange the traffic data
	2	Extract a source number vector for each port
	3	Create the frequency distribution for each port
	4	Learn the normal behavior mode for each port
	5	Use an index to speed up the learning phase
Detection		Count and compare (see Section 3.5 for the details)

© 2013 Information Processing Society of Japan

3.1 Collect and Arrange the Traffic Data

Two parameters, namely, time unit and observation period must be determined in advance. The time unit denotes a short time period (e.g., 10 minutes), in which packet data are accumulated for each port. The observation period indicates the length of time for collecting data from which the normal behavior modes are extracted.

Then, the number of sources sending packets to each of the monitored ports is counted separately in each time unit during a single observation period. That is, in each time unit, one pair of (destination port, number of sources) for each of the monitored ports is obtained. **Figure 4** illustrates the process of data collection.

3.2 Extract a Source Number Vector for Each Port

As discussed in Section 3.1, for each port, the number of sources sending packets to the port is counted in every time unit. Thus, for each of the monitored ports, a vector reflecting the number of sources in every time unit can be obtained. In other words, an element in this vector denotes the number of sources that sent packets to this port in the corresponding time unit. For example, a source number vector like (25, 69, ···, 300) indicates that 25, 69, and 300 sources sent packets to this port in the 1st, 2nd, and last time unit, respectively. Note that the number of elements in this vector is equal to the number of time units in the entire observation period.

3.3 Create Frequency Distribution for Each Port

For each of the monitored ports, a distribution of the number of its sources in a time unit can be obtained from its source number vector. **Figure 5** shows an example. The horizontal axis represents the quantized number of sources, while the vertical axis denotes the number of time units, that is, in how many time units the number of sources for this port corresponds to the current bin. Note that the frequency distribution of each port is expressed as a vector, which is called the FD vector in this study. Each element in the FD vector denotes the value of the corresponding bin.

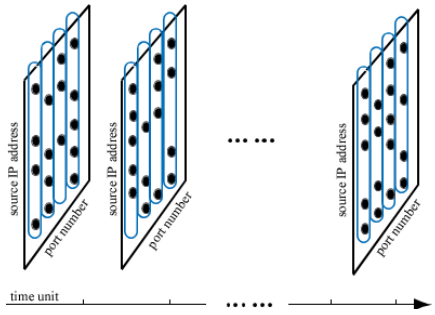


Fig. 4 Number of sources for each port is counted in each time unit.

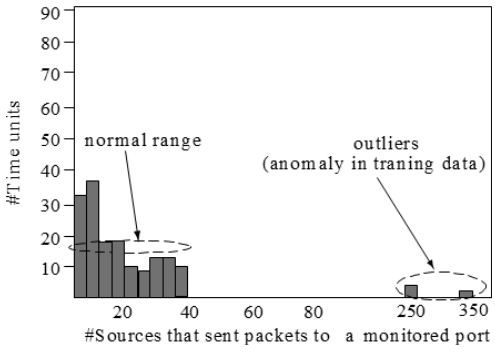


Fig. 5 Frequency distribution and behavior mode of one port.

3.4 Learn Normal Behavior Mode for Each Port

In this study, the normal behavior mode of a port denotes the maximum number of sources sending packets to this port in one time unit in normal time. An example is shown in Fig. 5, where the normal behavior mode of this port is the upper bound of the left group. In this distribution, the small groups located on the right far from the largest group are regarded as outliers (the anomalies in the learning data), which obviously should be discarded. After all the outliers have been discarded, the range of the remaining bins on the x-axis is regarded as the normal range of the number of sources in one time unit. In this study, only the upper limit of the normal range is used. That is, if the number of sources in the current time unit exceeds this upper limit, an alert is given, denoting that a distributed scan has possibly occurred. Note that for each of the monitored ports, one frequency distribution is created, from which one normal behavior mode is extracted. In the case that several normal behavior modes need to be built for one port from different learning data, one frequency distribution is necessary for building each normal behavior mode. Of course, if the normal behavior modes need to be updated, the learning phase is repeated. See Section 3.7. An algorithm for finding outliers from the frequency distribution is given in **Table 2**.

In the process of determining the normal behavior mode for a port after the outliers have been discarded, the following special cases must be considered.

1) If no packets were sent to this port during the whole observation period, that is, no normal behavior mode can be constructed for this port, a default value is used as the normal behavior mode of this port. This default value should be decided according to the actual network and how strictly we wish to detect attacks.

Table 2 Algorithm for finding outliers in a frequency distribution.

Steps	Descriptions
1	· The bins are checked one by one starting from the rightmost bin in the frequency distribution (see Fig. 5). · The checked bins are placed in Ω. · Let d_n be the distance from the bin that was just checked, to the next bin. If there is no next bin, use the distance from the current bin to the y-axis as d_n.

2	Check the next bin if it exists.
	If (($d_n > \alpha$[1])) and (the area[2]) in Ω is less than $\beta\%$[3] of the total area)) then
	the bins in Ω are regarded as outliers and are discarded
	go to step 1 // to find other outliers
	else
	put the current bin in Ω
	go to step 2 // this group is not finished

1) Here α is a threshold.
2) The area denotes the number of time units.
3) β is another threshold, used to avoid the case where most of the bins are regarded as outliers.
Note that: if d_n is large enough (e.g., larger than 2α), β is ignored, which means that if a bin cluster is far enough from the y-axis, it will be regarded as an anomaly even if this cluster has a large enough area.

© 2013 Information Processing Society of Japan

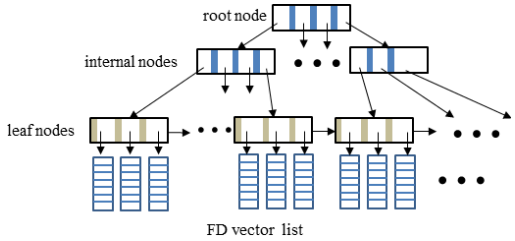


Fig. 6 General structure of the BH-tree.

2) If the learned normal behavior mode is smaller than the above-mentioned default value, then the default value is used.

3.5 Detection

After all the normal behavior modes for the possible ports have been prepared, detection is simple. Concretely speaking, all the possible ports are monitored and, in the current time unit, the numbers of sources that sent packets to each port are counted separately. Once a time unit has finished, the counted number for each port is compared with its normal behavior mode obtained in the learning phase. If the former is larger, then an alert is given indicating that a distributed scan has possibly occurred for this port. Moreover, according to the comparison result, the alert can be associated with a level, indicating the relative scale of the attack to the normal behavior mode of the port.

3.6 Indexing

As mentioned above, in order to monitor all the possible 65,535 TCP ports and all the possible 65,535 UDP ports, accelerating the learning and detection phases is necessary. For this purpose, we introduce an index, referred to as the BH-index in this study.

3.6.1 General Structure of the BH-index

The BH-index is illustrated in **Fig. 6**, and consists of two parts: the upper part is like a B^+-tree [27], while the lower part comprises a list of FD vectors. This index manages all the monitored ports including their FD vectors and their normal behavior modes. The information of all the ports is stored in the leaf nodes of the upper part in ascending order of ports.

Concretely, each entry in a leaf node is a tuple containing the (port, normal behavior mode, pointer), where the pointer points to the FD vector of this port. In addition, to access the information of all the ports in a quick succession (which is necessary in the learning phase) without following the BH-index from the root node each time, all the leaf nodes are linked from left to right.

3.6.2 Use in the Learning Phase

In step 1 of the learning phase (see Table 1), in every time unit, a pair comprising (port, number) is obtained for each of the monitored ports. Here number is the number of sources that sent packets to the port in the current time unit. Then, using the algorithm given in **Table 3**, all these pairs are inserted into the BH-index at the end of the time unit. Note that the index is empty at the start and it grows as the (port, number) pairs are inserted one by one. This algorithm is explained step by step below.

Step 1: Determine the leaf node. First the root node is checked. The branch that should be followed is determined by comparing the port and the key values in the root node. Then the corresponding child node is checked. This process is repeated until a leaf node is reached.

Table 3 Index insertion algorithm.

Objective: inserting (*port*, *number*)	
Steps	Descriptions
	Start
1	Determine the leaf node.
2	Check if the current port exists in this leaf node. If so, go to step 3. If not, go to step 4.
3	Update the FD vector and go to step 5.
4	Insert one new entry into this leaf node.
5	End.

Step 2: Check if the current port, *port*, exists in this leaf node. If it does, go to step 3, otherwise go to step 4.

Step 3: Update the FD vector. Since *port* already exists in this leaf node, this port has occurred previously. In this case, we can find its FD vector. The bin position corresponding to the number is then fixed in the FD vector. Thereafter, the value of that bin is increased by one, which means that one more time unit is located in that bin.

Step 4: Insert one new entry for *port*. Because no same port exists in the leaf node, this is the first time *port* has occurred, and therefore, a new entry should be added in the leaf node. Adding a new entry includes the following three actions. First, *port* is inserted as a new key in the appropriate location of the current leaf node, retaining the order of keys in the node. Second, an empty FD vector (in which all the elements are zero) is created for *port*. Finally, the bin corresponding to *number* is set to one. Importantly, if the number of entries in this leaf node overflows (exceeds a certain limit), this leaf node is split into two leaf nodes and a new entry is inserted in the parent node. Note that the split may be recursively applied upwards, with even the root node possibly being split (in this case the index height increases by one). The index grows in this way.

Once all the (*port*, *number*) pairs in every time unit in the learning data have been inserted into the BH-index, one FD vector has been created automatically for each port. Then, the normal behavior modes of all the ports can be extracted using the method in Section 3.4. In this process, the FD vectors of all the ports can be accessed from the leftmost leaf node to the rightmost leaf node by making use of the horizontal pointers in the leaf nodes (see Fig. 6).

After the learning phase has terminated, the learned normal behavior mode for each port is also stored in the leaf nodes of the BH-index. These are used in the detection phase. Again, every entry in the leaf nodes has three values: a port number, normal behavior mode, and a pointer pointing to the relevant FD vector.

3.6.3 Use in the Detection Phase

After all the normal behavior modes of the monitored ports have been extracted and stored in the leaf nodes of the BH-index, detection is easy.

In the current time unit, the number of sources is counted for each of the monitored ports. At the end of each time unit, a (*port*, *number*) pair is obtained for each port. We can find the normal behavior mode of *port* in the BH-index, by repeating steps 1 and 2 in Table 3. Then, a simple comparison is made between *number* and the normal behavior mode of port. If the former is

© 2013 Information Processing Society of Japan

greater, then this port was seemingly attacked in this time unit by a distributed scan. The BH-index is helpful in finding the corresponding normal behavior modes quickly. If the current port does not exist in the BH-index, which means that this port did not appear in the learning data, a default value is used. See Section 3.4 for the details.

3.7 Update of Normal Behavior Modes

Obviously, the normal behavior modes need to be updated sometimes. Because the learning phase is so fast, the normal behavior modes can be updated using new learning data at any time. In addition, if necessary, several normal behavior modes for different cases can be prepared for one port.

3.8 Alert Level

In our proposal, it is easy to associate an anomaly score (called the alert level in this study) to a test instance. The alerts may have different levels reflecting the relative scales of the attacks to the normal behavior modes of the corresponding ports. In this study, the alerts are divided into three levels. Specifically, in the current time unit, if the actual number of sources for one port is greater than that of its normal behavior mode, but less than twice that of the normal behavior mode, the level is set to 1. If this number is between two and three times that of the normal behavior mode, the level is set to 2. If this number is greater than three times that of the normal behavior mode, the level is set to 3. Note that the criteria for setting the alert levels can be adjusted freely. For example, we could use 4 or more levels. In fact, the alert levels are merely different expressions of the scales of suspicious attacks.

3.9 Advantages of Our Proposal

Our approach has three main advantages over existing related works. 1) All the possible TCP and UDP ports are monitored and it is easy to associate an anomaly score (i.e., the alert level) with a test instance. 2) The normal behavior modes of all the ports are obtained by a learning phase using historical data of each port. Thus, the normal behavior modes can reflect the individual features of the ports. In fact, it is generally impossible to manually decide thresholds for each of all the possible ports. If we use the same thresholds determined in advance for all ports, problems related to high false positive or false negative rates are likely to occur. Moreover, the identical thresholds could cause high false positive rates for some ports and high false negative rates for others. 3) An index is applied to accelerate the learning phase and to realize rapid detection.

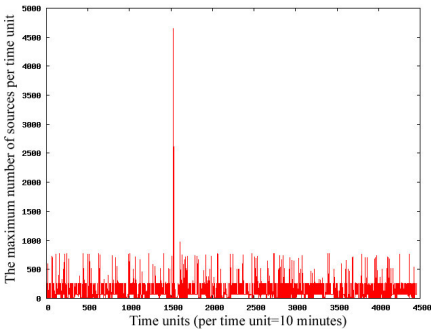
4. Experiments

4.1 Data Description

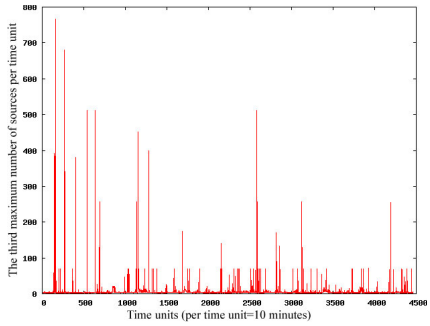
The traffic data of the /24 darknet of our campus network were used in this study. Concretely, three pairs of datasets shown in **Table 4** were used. One of the two datasets in each pair was for learning and the other for testing.

Table 4 Datasets used in our experiments.

		Data	Number of packets
Pair 1	Learning	2009.4.1–30	1,322,270
(TCP)	Testing	2009.5.1–31	2,187,086
Pair 2	Learning	2010.4.1–30	143,518
(UDP)	Testing	2010.5.1–31	113,682
Pair 3	Learning	2010.6.1–30	2,431,333
(TCP)	Testing	2010.7.1–31	2,404,247



(a) First maximum number of sources in each time unit.



(b) Third maximum number of sources in each time unit.

Fig. 7 General structure of the BH-tree.

4.2 Parameters

4.2.1 Parameters of the BH-index

When the BH-index is created, the capacity of a single node (i.e., the maximum number of entries in a node, known as the fanout in this study) needs to be determined in advance. For a disk-resident index, the node size should be the page size. However, in this study, the BH-index resides in main memory. The merit of using the BH-tree is that, when accessing the information of one port, only one leaf node and a very few of index nodes (non-leaf nodes) need to be accessed and all the other nodes are skipped. Thus, the advantage of using the BH-tree is obvious. If we were to choose a larger fanout, the tree possibly becomes shorter, which may lead to a smaller number of index nodes that have to be accessed. However, more comparisons would be needed when the nodes were accessed. We set the fanout to 5, 8, and 15. All the three BH-trees make the learning and detection phases very fast without noticeable difference among them (less than 5%). We also found that, when the fanout is 8, the response time is slightly better than the other two cases. The fanout is set to 8 in this study.

4.2.2 Parameters for Constructing the FD Vectors

When the FD vectors are constructed in Section 3.3, the number of bins and the width of a bin must be determined. Based on a large number of experiments, the number of sources sending packets to a single port in one time unit is not large (this is confirmed by the investigation shown in **Fig. 7**). Thus, the width of a bin should be small to differentiate different ports. In this study, it

© 2013 Information Processing Society of Japan

is set to 2 because we wish to observe the details of the frequency distributions of the ports. The number of bins is set to 21 making the starting number of the rightmost bin 40. In other words, all the time units with more than 40 sources will be allocated to the rightmost bin. The starting number of the rightmost bin must be much larger than (greater than twice in our experiments) the default value that is used in Section 3.4 and discussed in Section 4.2.3. This is because there must be enough space between the rightmost bin and the normal behavior mode to efficiently find the outliers.

Figure 7 shows an investigation result of the number of sources for a single port in one time unit between 2009.5.1 and 2009.5.31 in the darknet of our campus network. Figure 7 (a) shows the first maximum number and Fig. 7 (b) the third maximum number of sources among all the ports in each time unit. From Fig. 7, we can observe that, in the majority of the time units, the first maximum number of sources among all the ports is several hundreds. However, if we see the third maximum number, it drops to a small number, which means that, in most time units, most of the ports have a small number of sources.

4.2.3 Parameters for the Learning Phase

In Section 3.4, we refer to three parameters: α, β, and a default value. α is the upper bound of the distance whereby two subgroups with a distance less than α can be thought of as one group. By intuition, 20–30% of the total number of bins in the frequency distributions is appropriate for α. Thus, we set α to 5. β is used to avoid the case where too many bins are regarded as outliers and is set to 40% in the experiments. This means that if the area of a bin group is large enough (greater than 40% of the total area) and is not very far from the y-axis, this group will not be regarded as an outlier. The default value is used for ports for which no normal behavior modes have been obtained in the learning phase (i.e., ports that did not occur in the learning data) or ports whose normal behavior modes are too small. Generally, the default value should be determined according to how strictly we wish to detect attacks. It is set to 15 in this study. Obviously, it can be adjusted easily. If a smaller default value is used, more suspicious attacks may be detected and the false positive rate possibly increases. However, a greater default value may lead to a high false negative rate.

4.2.4 Time Unit and Observation Period

In the experiments the time unit was set to 5 minutes and 10 minutes, respectively. Only the detection results for a time unit of 10 minutes are presented in this paper because the results are similar for both. The observation period was set as one month, which means that the normal behavior modes were built for one month.

4.3 Performance Verification

4.3.1 Response Time for Learning

The repsonse time for building normal behavior modes of all the ports using the learning data are shown in **Table 5**, where the response time for learning includes reading the learning data from hard disk, counting the packets for each port in every time unit, constructing the BH-index, and extracting the normal behavior modes for all the ports. The computer we used was a Dell Precision T3500 with 16 GB main memory and an Intel Xeon W3520 (2.66 GHz) CPU. From Table 5, we can see that all the processes terminated in a few seconds.

Table 5 Response time for constructing normal behavior modes.

Datasets	Role	Response time (seconds)
Pair 1	Learning	4.516
Pair 2	Learning	2.148
Pair 3	Learning	6.433

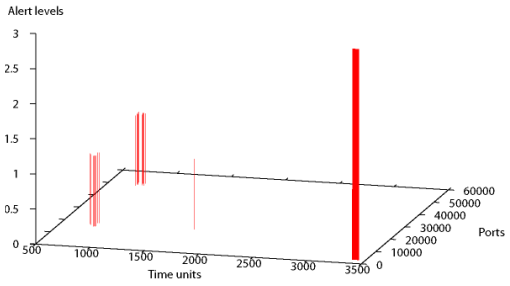


Fig. 8 Detection results for May 2009 (TCP traffic).

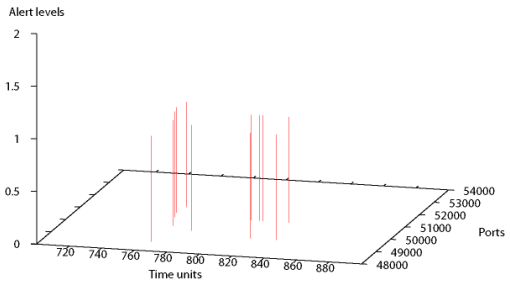


Fig. 9 Locally enlarged part of Fig. 8.

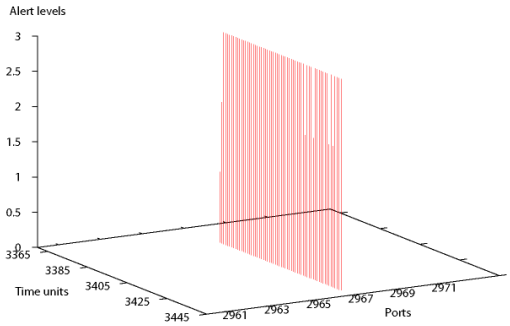


Fig. 10 Locally enlarged part of Fig. 8.

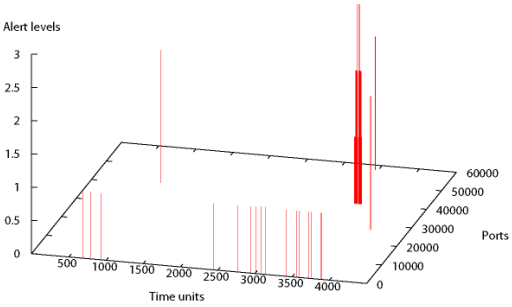


Fig. 11 Detection results for May 2010 (UDP traffic).

4.3.2 Detection Results

The detection results are illustrated in **Fig. 8**–**Fig. 14**. All the detected attacks were also confirmed manually from the original traffic data. That is, for each of the detected attacks, we checked the original traffic data and confirmed the number of

© 2013 Information Processing Society of Japan

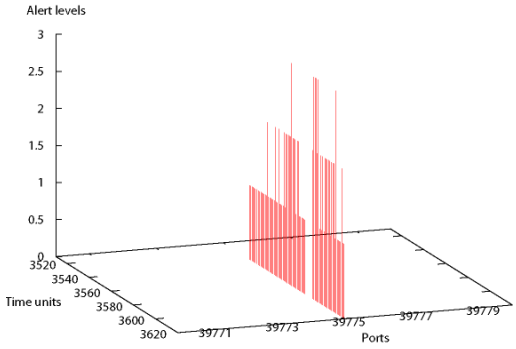


Fig. 12 Locally enlarged part of Fig. 11.

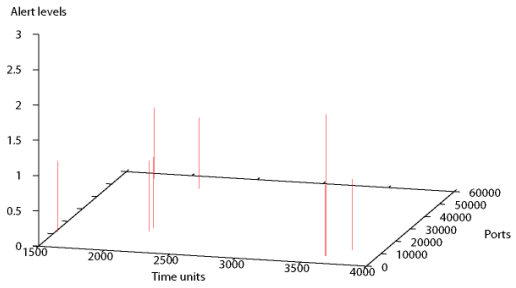


Fig. 13 Detection results for July 2010 (TCP).

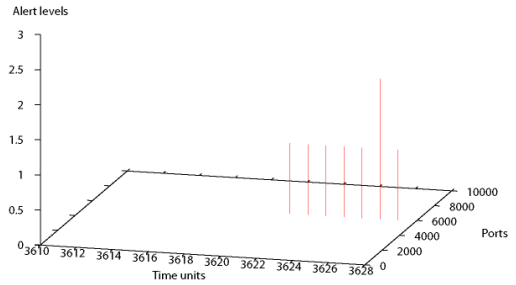


Fig. 14 Locally enlarged part of Fig. 13.

sources sending packets to the corresponding port in the corresponding time units. Moreover, all the detailed detection results are presented in Appendix A.1, A.2, and A.3.

a) Detection results using the datasets of Pair 1

The TCP traffic data for April and May 2009 were used as learning data to build the behavior nodes and as test data, respectively. The detection result is illustrated in Fig. 8, where the x-axis denotes the time units, the y-axis denotes the ports, and the z-axis indicates the alert levels of the detected distributed scan attacks. To see the attacks more clearly, two locally enlarged parts are shown in Fig. 9 and Fig. 10. Note that the detection results for TCP port 445 are not included in Fig. 8; the reason for this is given in Section 4.3.3.

From these results, we can see that TCP port 2967 was attacked by distributed scans for more than 10 hours and that the alert level was 3 for most of the time during this period. The detailed detection result is presented as Appendix A.1.

b) Detection results using the datasets of Pair 2

The UDP traffic data for April and May 2010 were used as learning data and test data, respectively. The detection result is illustrated in Fig. 11. A locally enlarged part of Fig. 11 is shown in Fig. 12, from which we can see that the UDP port 39775 was attacked by a distributed scan for more than 13 hours with a break

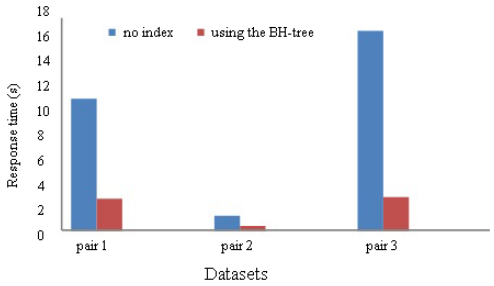


Fig. 15 Comparison on response times: using the BH-tree vs. no index.

of about 1 hour. The detailed detection result is presented as Appendix A.2.

c) Detection results using the datasets of Pair 3

In this dataset pair, the TCP traffic data for June 2010 were used as learning data and the TCP traffic data for the next month as test data. The detection result is shown in Fig. 13, with a locally enlarged part shown in Fig. 14. The TCP port 5900 was attacked by a distributed scan in seven consecutive time units (70 minutes). The detailed detection result is presented as Appendix A.3.

4.3.3 Confirmation of the Self-cleaning Ability in the Learning Phase

From the process of extracting the normal behavior modes shown in Section 3.4, we know that our algorithm can, at least to some extent, automatically discard anomalies in the learning data, which is also confirmed in our experiments. For example, in the learning data of April 2009, in 1,484 time units of the total 4,464 time units, the TCP port 445 was attacked by more than 100 sources. Thus, the rightmost bin in the frequency distribution of this port (see Fig. 5) is high, However, since it was automatically discarded by the learning algorithm, the attacks to TCP port 445 were not included in a normal behavior mode. Thus, the distributed scan attacks to this port in the test data were still detected. This example clearly indicates that our learning algorithm possesses a self-cleaning ability. Note that, the detection results for TCP port 445 in May 2009 were intentionally not included in Fig. 8. This is because the port was attacked by more than 100 sources in almost all the time units. Thus, if this port was included, the other attacks would not be clearly seen.

4.3.4 Response Time for Detection

The response time includes reading all the test data from hard disk, counting the packets for all the ports, and detecting the attacks in the entire test datasets (a total of 4,464 time units in each test dataset). We tested the response time for detection with and without using the BH-tree. The efficiency of the BH-tree can be verified from the results shown in **Fig. 15**.

4.4 Observations

From the above experiments, we can confirm that the proposed approach can detect distributed scan attacks efficiently and quickly. The learning phase is also quick with the aid of the BH-index. In addition, it is easy to associate an alert level with a test instance. Moreover, our learning algorithm has a certain degree of self-cleaning ability.

5. Conclusion

In this paper, we focused on one kind of the most important

© 2013 Information Processing Society of Japan

network attacks, distributed scan attacks, which are referred to as next generation cyber-attacks. A novel proposal based on normal behavior modes for detecting distributed scan attacks in darknet environments was presented and discussed. In our approach, all the possible TCP and UDP ports can be monitored and the alerts may have different levels reflecting the relative scales of the attacks to the normal behavior modes of the corresponding ports. An index (called the BH-tree) was introduced to speed up the learning and detection process and proved to be very efficient. This index is very useful in learning and updating the normal behavior modes and realizing rapid detection. The discussion and experiments using real darknet traffic data showed that our proposal is efficient and fast. Moreover, in the proposed approach, anomalies in the learning data can be cleaned to some extent in the learning phase, as confirmed in our experiments. Note that although this paper focused on distributed scan attacks in which multiple sources attack one port of multiple destination IP addresses, this approach can easily be adapted to distributed scan attacks in which multiple sources attack multiple ports of multiple destination IP addresses because all possible ports are monitored. Moreover, this idea can also be applied to ordinary networks, although our discussion and experiments focused on darknet traffic. In the near future, we will extend our proposal to ordinary networks.

Acknowledgments This work was partially supported by Proactive Response Against Cyber-attacks Through International Collaborative Exchange (PRACTICE), Ministry of Internal Affairs and Communications, Japan.

This work was also partly supported by Grant-in-aid for Scientific Research (C) No. 22500093, Japan Society for the Promotion of Science.

The first author also would like to express his gratitude to Emeritus Professor Akifumi Makinouchi of Kyushu University, Japan, for his helpful comments in this study.

References

[1] Eto, M., Inoue, D., Song, J., Ohtaka, K. and Nakao, K.: Nicter: A Large-Scale Network Incident Analysis System, *Proc. 1st Workshop on Building Analysis Datasets and Gathering Experience Returns for Security* (*BADGERS*), pp.37–45 (2011).

[2] Bailey, M., Cooke, E., Jahanian, F., Nazario, J. and Watson, D.: The Internet Motion Sensor: A Distributed Blackhole Monitoring System, *Proc. 12th ISOC Symposium on Network and Distributed Systems Security* (*NDSS*), pp.167–179 (2005).

[3] SANS Internet Storm Center, available from ⟨http://isc.sans.org/⟩.

[4] Pouget, F., Dacier, M. and Pham, V.H.: Leurre.com: On the Advantages of Deploying a Large Scale Distributed Honeypot Platform, *E-Crime and Computer Conference* (*ECCE*), pp.1–13 (2005).

[5] Cooke, E., Bailey, M., Mao, Z.M., Watson, D., Jahanian, F. and McPherson, D.: Toward Understanding Distributed Blackhole Placement, *Proc. ACM CCS Workshop on Rapid Malcode*, pp.54–64 (2004).

[6] Xu, S.: Collaborative Attack vs. Collaborative Defense, *Proc. 4th International Conference on Collaborative Computing* (*CollaborateCom2008*), *LNICST 10*, pp.217–228 (2009).

[7] Kanlayasiri, U., Sanguanpoing, S. and Jaratmanachot, W.: A Rule-based Approach for Port Scanning Detection, *Proc. 23rd Electrical Engineering Conference*, pp.148–153 (2000).

[8] Staniford, S., Hoagland, J. and McAlerney, J.: Practical Automated Detection of Stealthy Portscans, *Journal of Computer Security*, Vol.10, No.1, pp.105–136 (2002).

[9] Tang, Y.: Defending against Internet Worms: A Signature-based Approach, *Proc. 24th IEEE Annual Joint Conference of the Computer and Communications Societies* (*INFOCOM*), pp.1384–1394 (2005).

[10] Yazid, I., Hanan, A. and Aizaini, M.: Volume-based Network Intrusion Attacks Detection, *Advanced Computer Network and Security*, UTM Press, pp.147–162 (2008).

[11] Kind, A., Stoecklin, M.P. and Dimitropoulos, X.: Histogram-Based Traffic Anomaly Detection, *IEEE Trans. Network Service Management*, Vol.6, No.2, pp.1–12 (2009).

[12] Eskin, E. and Lee, W.: Modeling System Call for Intrusion Detection with Dynamic Window Sizes, *Proc. DARPA Information Survivability Conference and Exposition* (*DISCEX*), pp.165–175 (2001).

[13] Feng, Y., Hori, Y., Sakurai, K. and Takeuchi, J.: A Behavior-based Method for Detecting Outbreaks of Low-rate Attacks, *Proc. 3rd Workshop on Network Technologies for Security, Administration and Protection* (*NETSAP*), *SAINT2012*, pp.267–272 (2012).

[14] Xiang, Y., Li, K. and Zhou, W.: Low-Rate DDoS Attacks Detection and Traceback by Using New Information Metrics, *IEEE Trans. Information Forensics and Security*, Vol.6, No.2, pp.426–437 (2011).

[15] Lee, W. and Xiang, D.: Information-theoretic Measures for Anomaly Detection, *Proc. IEEE Symposium on Security and Privacy*, pp.130–143 (2001).

[16] Chandola, V., Banerjee, A. and Kumar, V.: Anomaly Detection: A Survey, *ACM Computing Survey*, Vol.41, No.3, pp.1–72 (2009).

[17] Stoecklin, M.P., Boudec, J.Y. and Andreas, K.: A Two-Layered Anomaly Detection Technique Based on Multi-modal Flow Normal behavior models, *Proc. PAM '08*, pp.212–221 (2008).

[18] Fukushima, Y., Hori, Y. and Sakurai, K.: A Study of Detecting Low Rate Attack Traffic, *Proc. Multimedia, Distributed, Cooperative and Mobile Symposium* (*DICOMO '10*), pp.492–500 (2010) (in Japanese).

[19] Sun, H., John, C.S., Lui, D. and Yau, K.Y.: Defending Against Low-rate TCP Attacks: Dynamic Detection and Protection, *Proc. 12th IEEE International Conference on Network Protocols* (*ICNP*), pp.196–205 (2004).

[20] Sun, H., John, C.S., Lui, D. and Yau, K.Y.: Distributed Mechanism in Detecting and Defending Against Low-rate TCP Attack, *International Journal of Computer and Telecommunications Networking*, Vol.50, No.13, pp.2312–2330 (2006).

[21] Kim, M.S., Kang, H.J. and Hong, S.C.: A Flow-based Method for Abnormal Network Traffic Detection, *Proc. IEEE/IPIP Network Operations and Management Symposium*, pp.599–612 (2004).

[22] Treurniet, J.: A Network Activity Classfication Schema and Its Application to Scan Detection, *IEEE/ACM Trans. Networking*, Vol.19, No.5, pp.1396–1404 (2011).

[23] Staniford, S. et al.: GrIDS – A Graph Based Intrusion Detection System for Large Networks, *Proc. 19th USA National Information Systems Secutiry Conference*, pp.361–370 (1996).

[24] Snort Users Manual, available from ⟨http://www.snort.org/docs⟩.

[25] Gates, C.: The Modeling and Detection of Distributed Port Scans: A Thesis Proposal, Technical Report CS-2003-01, Dalhousie University (2003).

[26] Yegneswaran, V., Barford, P. and Ullrich, J.: Internet Intrusions: Global Characteristics and Prevalence, *Proc. 2003 ACM Joint International Conference on Measurement and Modeling of Computer Systems*, pp.138–147 (2003).

[27] Bayer, R. and McCreight, E.M.: Organization and Maintenance of Large Ordered Indices, *Acta Informatica*, Vol.1, No.3, pp.173–189 (1972).

Appendix

A.1 Original Detection Result I

Learning data: TCP 2009.4 Test data: TCP 2009.5			
Alert Levels	Port No.	TimeUnit No.	#Sources
1	51030	759	23
1	51491	761	24
1	50697	761	24
1	48750	764	23
1	17750	765	24
1	50031	766	24
1	17161	777	24
1	49749	780	23
1	50812	807	23
1	50812	812	23
1	50812	830	24
1	19018	833	23
1	16149	833	22
1	49429	835	24
1	16482	839	24

© 2013 Information Processing Society of Japan

Alert Levels	Port No.	TimeUnit No.	#Sources
1	2967	3378	21
2	2967	3379	34
3	2967	3380	47
3	2967	3381	61
3	2967	3382	70
3	2967	3383	77
3	2967	3384	84
3	2967	3385	84
3	2967	3386	94
3	2967	3387	91
3	2967	3388	98
3	2967	3389	95
3	2967	3390	82
3	2967	3391	96
3	2967	3392	89
3	2967	3393	100
3	2967	3394	85
3	2967	3395	86
3	2967	3396	90
3	2967	3397	84
3	2967	3398	87
3	2967	3399	83
3	2967	3400	69
3	2967	3401	75
3	2967	3402	54
3	2967	3403	65
3	2967	3404	62
3	2967	3405	70
3	2967	3406	65
3	2967	3407	67
3	2967	3408	50
3	2967	3409	61
3	2967	3410	51
3	2967	3411	48
3	2967	3412	60
3	2967	3413	56
3	2967	3414	56
3	2967	3415	47
3	2967	3416	51
3	2967	3417	51
3	2967	3418	51
3	2967	3419	54
3	2967	3420	46
2	2967	3421	39
3	2967	3422	48
3	2967	3423	45
3	2967	3424	46
2	2967	3425	35
3	2967	3426	44
3	2967	3427	52
3	2967	3428	42
3	2967	3429	42
3	2967	3430	46
3	2967	3431	46
3	2967	3432	49
2	2967	3433	37
3	2967	3434	41
2	2967	3435	40
3	2967	3436	42
3	2967	3437	49
3	2967	3438	57
3	2967	3439	51

A.2 Original Detection Result I

Learning data: UDP 2010.4 Test data: UDP 2010.5			
Aleart Levels	Port No.	TimeUnit No.	#Sources
1	53	2948	20
1	53	3010	23
1	607	3072	26
1	607	3124	24
1	53	3424	22
1	57695	3455	24
2	57695	3456	37
2	57695	3457	37
2	57695	3458	38
1	57695	3459	26
1	39775	3534	27
1	39775	3535	28
1	39775	3536	24
1	39775	3537	29
1	39775	3538	26
1	39775	3539	26
1	39775	3540	26
1	39775	3542	26
1	39775	3543	26
1	39775	3544	25
1	39775	3545	24
1	39775	3546	29
1	39775	3547	29
1	39775	3548	28
1	39775	3549	26
2	39775	3550	43
1	39775	3551	27
1	39775	3552	30
1	39775	3553	26
1	39775	3554	22
1	39775	3555	23
1	53	3556	22
1	39775	3556	26
2	39775	3557	44
1	39775	3558	28
1	39775	3559	30
2	39775	3560	42
1	39775	3561	25
1	39775	3562	28
1	39775	3563	27
1	39775	3564	27
2	39775	3565	44
1	39775	3566	29
2	39775	3567	42
2	39775	3568	44
2	39775	3569	43
2	39775	3570	41
2	39775	3571	45
3	39775	3572	68
2	39775	3573	46
2	39775	3574	43
2	39775	3575	48
1	39775	3576	28
2	39775	3577	45
2	39775	3578	41
1	39775	3579	29
1	39775	3580	27
1	39775	3581	26
1	39775	3582	23
1	39775	3583	23
1	39775	3584	22
2	39775	3591	43
3	39775	3592	61
1	53	3592	22
3	39775	3593	65
3	39775	3594	62
2	39775	3595	46
3	39775	3596	62
1	39775	3597	25
2	39775	3598	43

© 2013 Information Processing Society of Japan

Aleart Levels	Port No.	TimeUnit No.	#Sources
1	39775	3599	28
2	39775	3600	45
1	39775	3601	24
2	39775	3602	41
2	39775	3603	43
2	39775	3604	44
2	39775	3605	41
2	39775	3606	45
1	39775	3607	29
2	39775	3608	49
2	39775	3609	45
2	39775	3610	48
1	39775	3611	30
3	39775	3612	62
1	39775	3613	30
1	39775	3614	28
1	39775	3615	25
1	39775	3616	24
2	39775	3617	41
1	39775	3618	30
1	39775	3619	30
1	39775	3623	29
1	39775	3624	27
2	39775	3625	43
2	39775	3626	40
1	39775	3627	25
1	39775	3628	21
1	39775	3629	27
1	39775	3630	30
1	39775	3631	25
1	39775	3632	25
1	39775	3633	24
1	53	3725	22
1	53	3753	22
1	53	3883	22
1	53	3895	22
1	53	3898	22
2	27487	3999	38
2	27487	4000	37
2	27487	4001	40
2	27487	4002	37
1	27487	4003	30

A.3 Original Detection Result III

Learning data:	TCP 2010.6		
Test data:	TCP 2010.7		
Aleart Levels	Port No.	TimeUnit No.	#Sources
1	11740	1514	21
1	11740	1516	17
1	56331	1751	18
1	64095	1808	16
1	50812	2160	16
1	19018	2163	22
1	16149	2164	16
1	5900	3621	21
1	5900	3622	30
1	5900	3623	25
1	5900	3624	27
1	5900	3625	26
2	5900	3626	34
1	5900	3627	23
1	11740	3765	18

Yaokai Feng received his B.E. and M.E. degrees in computer science from Tianjin University, China, in 1986 and 1992, respectively. He received his Ph.D. degree in Information Science from Kyushu University, Japan, in 2004. Now, he is an Assistant Professor at Kyushu University, Fukuoka, Japan. Since 2011, he has been a Researcher (part-time) at Institute of Systems, Information Technologies and Nanotechnologies (ISIT), Fukuoka, Japan. His current research interests include network security, pattern recognition, and database. He received MIRU Excellent Paper Award in 2011. He is a member of IPSJ and IEEE.

Yoshiaki Hori received his B.E., M.E., and D.E. degrees from Kyushu Institute of Technology, Iizuka, Japan in 1992, 1994, and 2002, respectively. From 1994 to 2003, he was a Research Associate in Common Technical Courses, Kyushu Institute of Design, Fukuoka. From 2003 to 2004, he was a Research Associate in the Department of Art and Information Design, Kyushu University, Fukuoka. Since March 2004, he has been an Associate Professor in the Department of Computer Science and Communication Engineering, Kyushu University. He has been an Associate Professor in the Department of Informatics, Kyushu University. His research interests include network security, network architecture, and performance evaluation of network protocols on various networks. He is a member of IEEE, ACM, and IPSJ.

Kouichi Sakurai received his B.S. degree in mathematics from the Faculty of Science, Kyushu University and M.S. degree in applied science from the Faculty of Engineering, Kyushu University in 1986 and 1988 respectively. He had been engaged in the research and development on cryptography and information security at the Computer and Information Systems Laboratory at Mitsubishi Electric Corporation from 1988 to 1994. He received his D.E. degree from the Faculty of Engineering, Kyushu University in 1993. Since 1994 he has been working for the Department of Computer Science of Kyushu University as an Associate Professor, and now he is a Full Professor from 2002. His current research interests are in cryptography and information security. Dr. Sakurai is a member of IEICE, IPSJ, Mathematical Society of Japan, IEEE, ACM and the International Association for Cryptologic Research.

© 2013 Information Processing Society of Japan

Jun'ichi Takeuchi was born in Tokyo, Japan in 1964. He graduated from the University of Tokyo in majoring physics in 1989. He received his D.E. degree in mathematical engineering from the University of Tokyo in 1996. In 1989, he joined NEC Corporation, Japan, where he worked on learning theory and its applications including network security. In 2006, he joined Kyushu University, Fukuoka, Japan, where he is a Professor of Mathematical Engineering. From 1996 to 1997 he was a Visiting Research Scholar at Department of Statistics, Yale University, New Haven, CT, USA. From 2005 to 2006 he was an Expert Researcher (part-time) at Security Advancement Group, National Institute of Information and Communications Technology (NICT), Japan, where he was involved in development of the nicter system. Since 2006, he has been a Researcher (part-time) at Institute of Systems, Information Technologies and Nanotechnologies (ISIT), Fukuoka, Japan. His research interest includes mathematical statistics, information geometry, information theory, machine learning and its applications. He is a member of IEEE, IEICE, IPSJ, and JSIAM.

© 2013 Information Processing Society of Japan