Journal of Information Processing Vol. 16 165–175 (Sep. 2008)

Regular Paper

Automated Port-scan Classification with Decision Tree and Distributed Sensors

Hiroaki Kikuchi,[†1] Naoya Fukuno,[†1]
Tomohiro Kobori,[†1] Masato Terada[†2]
and Tangtisanon Pikulkaew[†3]

Computer worms randomly perform port scans to find vulnerable hosts to intrude over the Internet. Malicious software varies its port-scan strategy, e.g., some hosts intensively perform scans on a particular target and some hosts scan uniformly over IP address blocks. In this paper, we propose a new automated worm classification scheme from distributed observations. Our proposed scheme can detect some statistics of behavior with a simple decision tree consisting of some nodes to classify source addresses with optimal threshold values. The choice of thresholds is automated to minimize the entropy gain of the classification. Once a tree has been constructed, the classification can be done very quickly and accurately. In this paper, we analyze a set of source addresses observed by the distributed 30 sensors in ISDAS for a year in order to clarify a primary statistics of worms. Based on the statistical characteristics, we present the proposed classification and show the performance of the proposed scheme[⋆1].

1. Introduction

There are several strategies for performing port scans. Some computer worms select random destination addresses uniformly distributed over the entire address space. Some types of worms intensively scan particular addresses, defined by "*black lists*". For instance, the *W32.Sasser* worm[1)] spreads by scanning the randomly selected IP addresses for vulnerable systems. It performs scans of fully randomly determined destinations with a probability of 0.52 and of randomly determined destinations leaving the two highest octets unchanged with a probability of 0.25, and one octet unchanged with a probability of 0.23. Many of the major worms have been well engineered and the common algorithms for performing port scans and for choosing random destinations are known, e.g., Slammer[2)], Witty[3)], and Code Red[4)]. On the Internet, the mixture of these complicated behaviors is a major source of complexity, which prevents one from predicting the exact impact of worms and distributed attacks, although new malicious codes are spreading every day.

There have been several attempts to observe the traffic of worms and intruders. A *honeypot* is a semi-passive sensor that tries to pretend a vulnerable host in a faked communication with intruders or worms. It identifies not only worms from the interaction with worms, but also risks being detected as a honeypot[5)]. On the other hand, a *non-interaction sensor*, is typically located at an unused IP address called a *dark net*, and can observe any packets in a passive manner, free from risk of detection by worms. Network Telescope[6)], Internet Storm Center[7)], DShield[8)] and ISDAS[9)] are examples of passive sensors. Note that as the passive sensor does not have any legitimate access, every packet sent to the sensor must be classified as malicious.

Given the traffic data of port scans, there are many studies toward fast and accurate detection of the spread of worms, Kumar uses the characteristics of the pseudo random number generation algorithm used in the witty worm to reconstruct the spread of infected hosts in Ref. 10). Ishiguro, et al. propose the Wavelet coefficients used as metric for anomaly detection in Ref. 11). Wang uses artificial neural networks for intrusion detection[12)], Laskov, et al. propose an application of support vector machine to information security in Ref. 13).

There are some simpler but light-weight detection methods. Jung, et al. presented an algorithm to detect malicious packets called the Sequential Hypothesis Testing based on Threshold of Random Walk (TRW)[14)]. Dunlop, et al. presented a simple statistical scheme called Simple Worm Detection Scheme (SWorD)[15)], where the number of connection attempts is tested with threshold values. These threshold based approaches allow us to detect the worms promptly before their damages gets significant. Thus, thresholds are widely deployed in many detection attempts. The thresholds, however, should be carefully chosen in terms of accu-

[†1] School of Information Technology, Tokai University
[†2] Hitachi Incident Response Team (HIRT), Hitachi, Ltd.
[†3] Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang
[⋆1] The original version of this paper is presented at the 2nd Joint Workshop on Information Security (JWIS 2007).

racy and performance. In the SWorD, source addresses of packets are examined in two-tired thresholds, determined via the linear regression, but the two tires are not proved as an optimal structure.

To address the issue of an optimal threshold, we introduce a machine learning technology, known as a *decision tree learning with continuous attribute* [16], in which the nearly optimal thresholds are chosen to maximize the entropy gain of classification. We stress that the decision tree classifies worms with a simple threshold as well as SWorD, but performs multiple-tired testing, which can make the classification more accurate than the two-tired structure used in SWorD. On the other hand, the difficult part of decision tree is the selection of learning data that should be considered as a representative of all categories. Unfortunately, the traffic data observed by distributed sensors can be skewed by the huge number of port-scanning packets generated by particular worms, which makes random sampling impossible. In order to deal with the distortion of the observation, we propose a notion of *visit* that is defined as the number of distinct destination addresses that a source address sends. With visit, we can divide the set of source addresses into several subsets that are good for choosing the learning data for a decision tree. We also clarify a new useful property of visit that a number of addresses dropped into a subset characterized with it can be approximated by the *Zipf's law* [17], which is a well-known empirical property that can be seen commonly in our daily life.

Our Contribution

There are some main aspects to our contributions.

- We present a new category of worms in terms of the strategy of port scans: periodic, intensive, and random scans, which are based on the actual observation of the Internet from several distributed sensors.
- We propose a decision tree learning algorithm to determine the optimal structure consisting of an optimal threshold to quickly and accurately classify a large number of packets.
- We show a property of worms according to which the Zipf's law holds for a number of source addresses and the number of distinct destination addresses for the source address. The property is useful for randomly sampling representative data out of noisy observations containing many hazard packets.
- Our experiment based on the ISDAS (Internet Scan Data Acquisition System [9]) distributed observation data shows the performance of the proposed scheme and the ratio of worms for each category.

The rest of our paper is organized as follows. In Section 2, we give some fundamental definitions for worms and the passive sensors in the ISDAS. We also show typical behaviors of port scans in some worm binaries and define three classes of worms; random, periodic and intensive scans. Section 3 discusses the automated way to classify worms from distributed passive data and presents the experimental results of the classification and its accuracy.

2. Properties of Port Scans

2.1 Fundamental Definitions

We give fundamental definitions necessary to discuss the characteristics of worms.

Definition 2.1 A *scanner* is a host which performs port scans on other hosts, looking for targets to be attacked. A *sensor* is a host that passively observe all packets sent from scanners.

Typically, a scanner is a host which has some vulnerability and therefore is controlled by malicious codes, worms and viruses. Some scanners may be human operated, but we do not distinguish between malicious codes and malicious operators. In contrast with the honeypot that requires many interactions with scanners, a sensor is a passive device with few interaction and thus can be distributed at low cost without a risk of being detected by scanners. The global IP addresses assigned to sensors should never be exposed to scanners. Both scanners and sensors are assigned always-on static IP addresses.

Definition 2.2 Let S be a set of sensors, $\{s_1, s_2, \ldots, s_m\}$, where m is the number of sensors. Let n be a number of scanners and n_0 be the total number of global active addresses. Let h be a number of *unique hosts* observed by a sensor within a duration of observation. We denote by $\Delta h(x)$ the mean of unique hosts per day.

Suppose that a source address (scanner) is observed C times from k distinct sensors within the duration of observation. We refer to k as a *visit* and to C as a *count*.

Table 1 Statistics of ISDAS distributed sensors.

	sensor	count C	unique $h(x)$	$\Delta h(x)$[/day]
Average	–	146000	37820	104.9
Standard deviation	–	134900	29310	82.72
Max	s_1	450671	98840	270.79
Min	s_{15}	6475	1539	4.22

Scanning a randomly chosen destination address increases the visits, while an intensive scan just increases a count. Note that a visit and a count are independent of each other.

Definition 2.3 Let ω be the inter-arrival time of scans at the sensor. The mean of ω is given by C/t, where t is the duration of the observation. We denote an inter-arrival time of the j-th sensor's by ω_j. Let ω_S be an inter-arrival time of the whole set of sensors S.

2.2 ISDAS Distributed Sensors

The Internet Scan Data Acquisition System (ISDAS) is a distributed sensor, under the operation of JPCERT/CC[9], and estimates the scale of a current malicious event and its performance.

Table 1 shows the statistics of $m = 30$ sensors from April 1, 2006, through March 31, 2007. The most frequently scanned sensor is s_1 with about 451,000 counts, which is 70 times that of the least frequently scanned sensor s_{15}. In this sense, the destination addresses to scan are not uniformly distributed. In this data set, a visit k ranges from 1 to 30.

2.3 Black-box Analysis of Worms

There are several strategies to perform port scans. For vulnerable system, some worms randomly choose destination IP addresses from the entire address space, and some worms scan exhaustively every address in the local subnet in which the infected host exists. The former is called a *random* scan and the latter a *local* scan. The representative of local scans is the Blaster worm, which increments the destination of port scans one at a time. The random scans are widely applied by many major worms, e.g., Sasser, Witty, Netsky, and Slamer. Refer to Ref. 18) for more details on the behavior of worms.

We investigated the binary codes of some typical worms and observed the sequence of specified destination addresses. **Figure 1** describes each of the four octets of destination addresses sent by *W32.Sasser.Worm*, a worm that attempts to exploit the vulnerability described in Microsoft Security Bulletin MS04-011[19]. To monitor the behavior of the worm, we used the virtual machine, VMWare Workstation[20], with the specification in **Table 2**. According to the Symantec Security Response, the Sasser worm generates completely random addresses 52% of the time, the last two octets of the infected hosts are replaced at random 25%

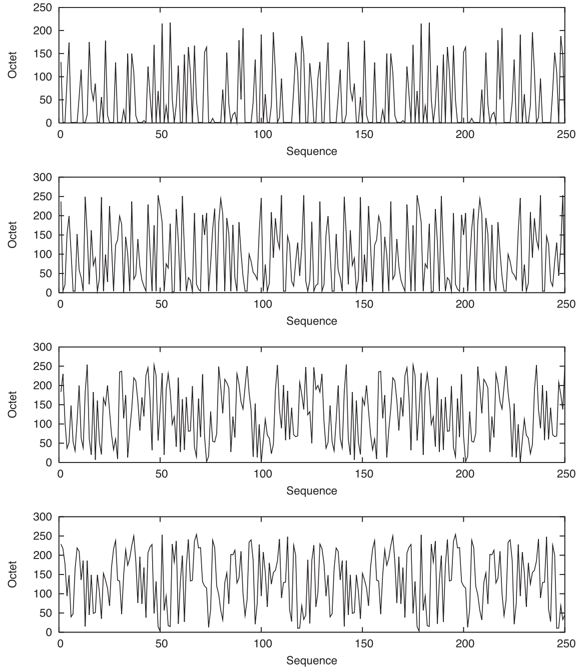

Fig. 1 Time series of destination IP addresses selected by `sasser`.

Table 2 Specification of experiment in VM-Ware[20].

attribute	value
host OS	Windows XP SP2 (Pentium 4, 3.0 GHz, 1 GB)
guest OS	Windows XP SP1, Windows 2000 SP2
duration	24 hours
log size	100 MB per worm

of the time, and the last three octets changed at random 23% of the time. We can observe the random behavior from Fig. 1.

As an instance of local scans, we show the behavior of *W32.Blaster.Worm* in **Fig. 2**. The worm connects on TCP port 135 and sends a large amount of data sufficient to overrun the buffer. The Security Response reported that the worm picked a random IP address and performed scanning, incrementing the last octet to scan the entire subnet. Our experiment demonstrates the incrementing of addresses, as depicted in the figure.

2.4 Scan Strategies

The strategy for determining the destination to scan varies for worms. The typical strategies are classified into three categories:

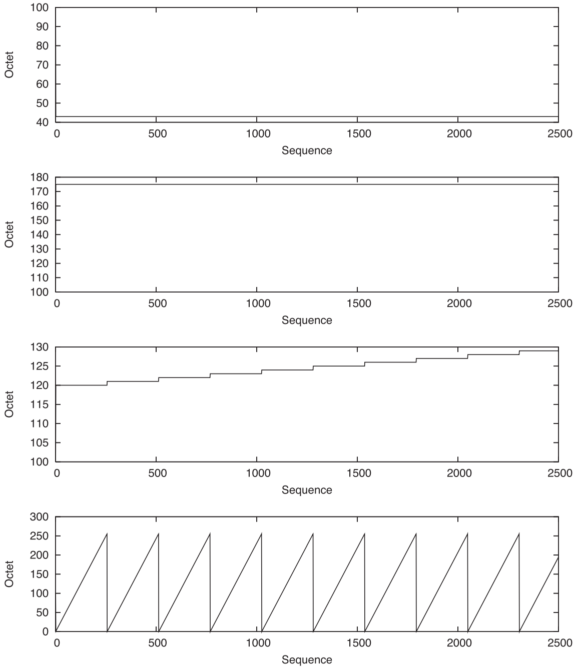

Fig. 2 Time series of destination IP addresses selected by `blaster`.

(1) **Periodic scan**
uses an algorithm to choose every destination exhaustively over the whole address space in Periodic rounds,

(2) **Intensive scan**
focuses on a particular (typically smaller) set of (probably vulnerable) ad-

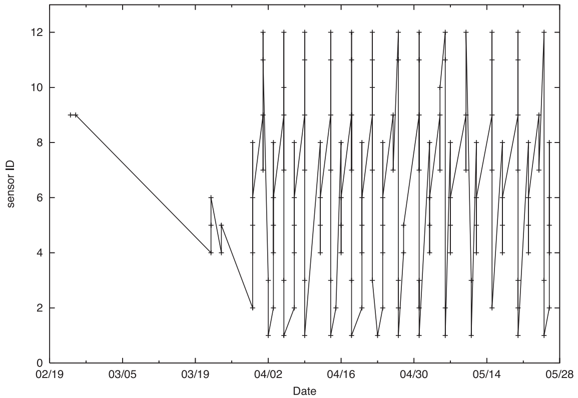

Fig. 3 Periodic scan – Time diagram of destination to scan from a source address lying in 218.0.0.0/8 (V_1).

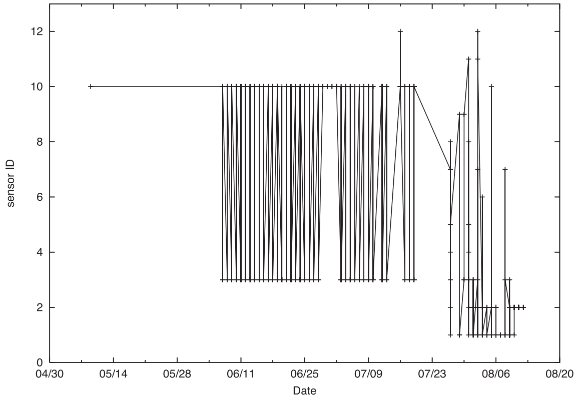

Fig. 4 Intensive scan – Time diagram of destination to scan from a source address lying in 61.0.0.0/8 (V_2).

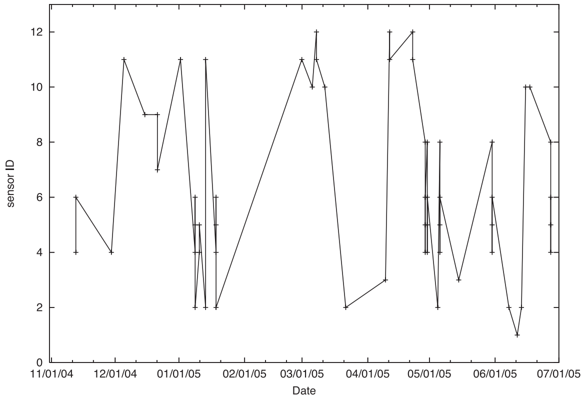

Fig. 5 Random scan – Time diagram of destination to scan from a source address lying in 218.0.0.0/8 (V_3).

dresses, and

(3) **Random scan**
is the most common way to find vulnerable hosts and the destination address is uniformly chosen.

For instance, we demonstrate the typical behavior for three strategies with a time diagram indicating sensor Identifiers (random order) to be scanned by a particular source address. **Figures 3**, **4** and **5** show instances of (1) periodic, (2) intensive, and (3) random scans, respectively.

3. Automated Classification of Scanners

3.1 Zipf Behavior of Scanners

In order to roughly classify scanners, we partition a set of unique hosts into m subsets specified by k. **Figure 6** shows a distribution of unique hosts $h(k)$ with respect to *visit* k in a duration of one year. The vertical axis is logarithmic. The figure illustrates that 105 million scanners (source addresses), which correspond to about 86% of all scanners, are observed by just one sensor, i.e., $k = 1$. Followed by 0.12 million of $h(2)$, the smallest set with $k = 12$ contains 293 unique scanners. The scanners in $h(12)$ can be considered to perform an exhaustive scan over all the addresses.

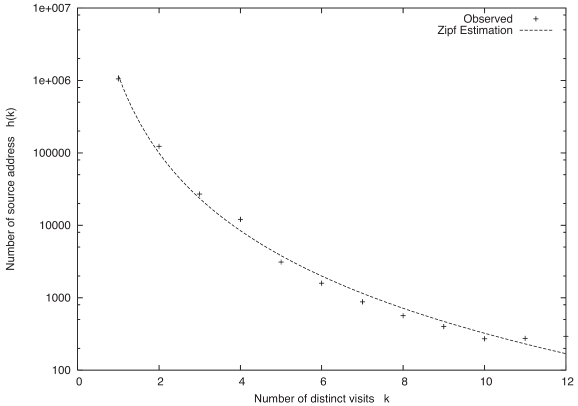

Fig. 6 Unique hosts $h(k)$ with respects to visit k.

Let us now consider a mathematical model of the behavior of scanners from a macroscopic point of view. We find the scanners' behavior follows the well-known Zipf's law [17]. The Zipf's law states that the most frequent words will occur approximately twice as often as the second frequent word in a natural language. It is an empirical law without any theoretical background but can be observed commonly in our daily life. More formally, the fraction of k-th ranked subset is linear to $1/k$, that is, a number of unique hosts with visit k is approximated by

$$h(k) = \frac{h_0}{k^s}$$

where s is a constant characterizing the distribution, which is 1 in the original Zipf's law. Note that k is not the rank of a subset but can be identical to the rank when we have a large enough set divided into a small number of subsets $k \leq 12$, i.e., no subset violates the rank and the visits.

By taking the log of both sides of the equation and minimizing the mean squared error according to the fitting algorithm, we identify the optimal constant

$$s = 3.5616$$

for the initial (the whole unique hosts) h_0 is 1174433. The estimated value in the model looks a good approximation of the observed data in Fig. 6.

3.2 Overview of the Proposed Scheme

Classifying strategies from observed scans is, however, not so easy. First, there are more than 300 million packets observed, which is too many to be analyzed one-by-one. Second, worm behaviors are not deterministic, i.e., destination addresses are often selected by probabilistic methods. Even though a single kind of worm has an uncertainty in its distribution of target addresses, we need to deal with several behaviors of many variants of worms. Third, the source address sent from the infected host may be spoofed. Our observation has potentially the same noise as the one caused by the faked addresses for disturbing our analysis.

To overcome these difficulties, we attempted the following analysis.

(1) A manual analysis with random sampling
To deal with the uncertainty of classification, we have a human examiner classify the source address while looking at the time series of destination addresses such as depicted in Figs. 3, 4 and 5. The logic for determining the target may vary considerably with the frequency of the random scans. For example, source addresses with $k < 5$ must not be random scans. To avoid the skew in sampling, we chose 20 source addresses for each of the 30 subsets that were partitioned by *visit k*. Taking 20 random samples from each of 30 subsets, we have a total of about 300 addresses. In the end, we have a data set classified into three classes, *random*, *intensive* and *periodic*, which will be used as a learning data.

(2) Machine learning for the sample log data
Before we present the machine learning technique for automated classification of malicious intrusions, we extract some features from each of the classes, using the statistics of frequency of scans, the number of sensors, the inter-arrival time of scans, etc, as described in Section 3.3. From among the many learning algorithms proposed so far, such as neural networks, clustering algorithms, and support vector machines, we chose the decision tree algorithm mentioned in Section 3.4. The reasons include (1) the availability of the numerical attribute, (2) a lightweight decision after training, and (3) the threshold based classification, which is good for grasping the distribution of the worms.

(3) Automated Classification for the entire data
We investigate the whole data set with the trained classifier, i.e., the decision tree. Once the tree is constructed, we can apply it to any given data set. This feature allows us to detect any change in the scanning target by classifying the monthly or weekly observations. We clarify the difference in the ratio of scanning classes in *visit k*.

3.3 Characteristics of the Infected Address

From a careful observation of the scans, we focus on the variance of the scan frequency, which plays a significant role in the classification of strategies. The periodic scans mostly run routinely and therefore the interval between scans tends to converge to a particular value; correspondingly, the frequencies of the scans are likely to be even and the variance gets smaller. On the other hand, the intensive scan could yield a greater variance of the scan counts.

Let us define some statistical parameters that are useful for classifying worms. The mean and the variance of the *scan counts* are defined by:

$$\mu(C_*) = \frac{1}{m}\sum_{i=1}^{m} C_i,$$

$$\sigma(C_*) = \frac{1}{m}\sum_{i=1}^{m}(C_i - \mu(C_*))^2$$

where $C_1, \ldots C_m$ is the number of scans observed in m distributed sensors. In the same way, we define the mean *inter-arrival time* of scans as:

$$\mu(\omega_*) = \frac{1}{m}\sum_{i=1}^{m} \mu(\omega_i),$$

$$\sigma(\omega_*) = \frac{1}{m}\sum_{i=1}^{m}(\mu(\omega_i) - \mu(\omega_*))^2$$

where $\mu(\omega_i)$ is the *mean inter-arrival time of scans* observed at the i-th sensor. Additionally, we evaluate the overall frequency of scans by the *integrated inter-arrival time* captured by the whole set of sensors S as follows,

$$\mu(\omega_S) = \frac{1}{mC_*}\sum_{j}^{mC_*} \omega_{S,j},$$

$$\sigma(\omega_S) = \frac{1}{mC_*}\sum_{j=1}^{mC_*} (\omega_{S,j} - \mu(\omega_S))^2,$$

where $\omega_{S,j}$ is j-th *inter-arrival time of all scans* is S.

3.4 Decision Tree Learning Algorithm

To automate the classification process we propose a decision tree learning algorithm such as ID3 [21] and C4.5 [16]. The ID3 is a decision tree algorithm that operates by greedily choosing the best attribute to classify the set of pages, resulting in a tree of words (attributes) with leaves (target class). The ID3 chooses the best attribute t which maximizes the information gain [⋆1] defined by

$$I(f;t) = H(f) - E[H(f|t)],$$

where $H(f)$ is the entropy of the class f, defined as $H(f) = -p_{true}\log p_{true} - p_{false}\log p_{true}$, and the probabilities $p_{yes} = |D_{true}|/m$ and $p_{false} = 1 - p_{true}$. The expected entropy is

$$E[H(f|t)] = \sum_{x=0,1} P(t = x)H(f|_{t=x}).$$

A decision tree algorithm is appropriate for our purpose to classify scan-strategies with numerical (continuous) attributes (C4.5 allows dealing with numerical data). Moreover, the learning results are simple enough to grasp the distribution of scanners over several orthogonal characteristics. We will show the graphical analysis in the following sections.

3.5 Analysis Results

Table 3 shows the number of unique source addresses, the total number of scans, and the average number of scans per host. The average number of scans increases as k increases, i.e., random scans take place more frequently. Let A_k be a subset of unique source addresses such that the address is observed by k distinct sensors. For example, the smallest subset, A_{24}, has 23 unique addresses chosen out of 4,079,162 addresses. Note that $h(k) = |A_k|$.

Table 4 shows the result of the manual classification of port scans for randomly sampling data. **Table 5** shows the result of the automated classification made by the decision tree, illustrated in **Fig. 7**. The tree classifies addresses by the statistical values indicated at nodes. For instance, addresses V_1, V_2 and V_3 with statistical values shown in **Table 6** are classified into three types correctly.

In comparing the two results, the random class is the most common type of

Table 3 Number of Scans wrt. visit k.

k	# of address	# of scans	average [scan/host]
A_1	858997	2649627	3.08
A_2	71573	667945	9.33
A_3	14068	143080	10.17
A_4	5619	127607	22.71
A_5	2735	71912	26.89
A_6	1659	71912	43.35
A_7	1023	48604	47.51
A_8	669	29079	43.47
A_9	474	19736	41.64
A_{10}	339	17197	50.73
A_{11}	263	12705	48.31
A_{12}	183	6362	34.77
A_{13}	172	9064	52.70
A_{14}	138	12017	87.08
A_{15}	116	11199	96.54
A_{16}	105	24081	229.34
A_{17}	98	14001	142.87
A_{18}	80	10986	137.33
A_{19}	77	10550	137.01
A_{20}	63	7879	125.06
A_{21}	54	5796	107.33
A_{22}	52	9995	192.21
A_{23}	42	15906	378.71
A_{24}	31	22699	732.23
A_{25}	26	17793	684.35
A_{26}	16	18174	1135.88
A_{27}	6	13786	2297.67
A_{28}	2	456	228.00
A_{29}	2	7383	3691.50
A_{30}	0	0	0

Table 4 Manual classification of scans in random sampling.

	Random	Periodic	Intensive	Total
Frequency	223	101	22	346
Ratio	0.64	0.3	0.06	1

Table 5 Automated classification of scans for the whole data.

	Random	Periodic	Intensive	Total
Frequency	783028	164404	11250	958682
Raito	0.82	0.17	0.01	1

[⋆1] It is equivalently referred as the *mutual information* of two random variables f and t.

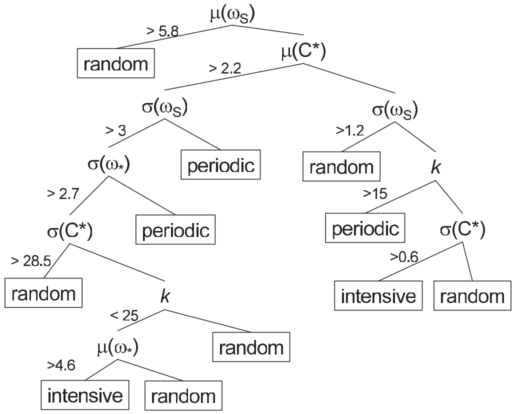

Fig. 7 Decision Tree (C4.5) classifying scanning strategies.

Table 6 Statistical value for typical scanning strategies.

	IP address	$\mu(C_*)$	$\sigma(C_*)$	$\sigma(\omega_*)$	type
V_1	61.0.0.0	8.033	36.1	15466528	Intensive
V_2	125.0.0.0	9.3	2508.21	0.85	Periodic
V_3	121.0.0.0	5.1	365.49	2.86	Random

scans in both, while the fraction of the random class in the entire data is greater than that of the sampling data. The difference between the two classifications is not significant, taking into consideration the sampling size, i.e., 300 samples out of 100,000 addresses.

3.6 Effect of *Visit*

How much does a *visit* affect the scanning types?

In **Fig. 8**, we illustrate the ratio of three classes for every subset, $A_1, \ldots, A_{30}$. From the observation, the fraction of random scans decreases slightly as the *visit* k increases, except for the last subset A_{29}, which is occupied by random scans. Note that the last subset A_{30} has no element in our experimental data for some reason. The curve of the periodic scans has two peaks $k = 24$ and 25. The intensive class exhibits less common behavior for any k. We conclude that port scans consist of 80% random, 18% periodic, and 2% intensive classes.

The quantity of k plays an important role in the decision tree of Fig. 7, where nodes represent statistical values and the edges indicate the threshold value to classify scans into two subsequent nodes. The leaves are final classified classes. For instance, an address with $\mu(\omega_S) = 6.0 > 5.8$ is classified as random (top left leaf), while the address $\mu(\omega_S) < 5.8$, $\mu(C_*) < 2.2$ and $\mu(\omega_S) > 1.2$ belongs to the random scan class. The tree has nodes of k on both sides, which means that the classification logic may be changed depending on k. The behavior makes sense in the fact that the fraction of periodic scans gets larger, with k more than 15 in Fig. 8.

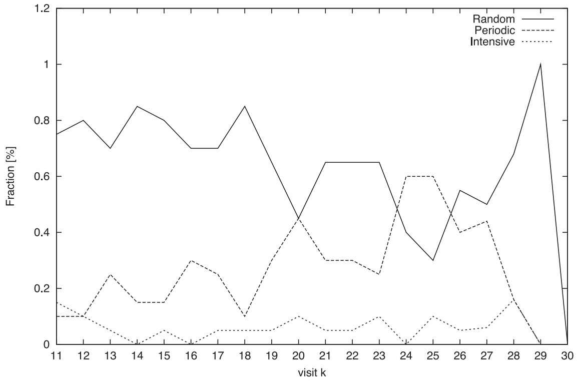

Fig. 8 Ratio of each Scan Classes.

Table 7 Accuracy of classification in decision tree.

Classified as \	Random	Periodic	Intensive	Total	Precision
Random	168	68	3	239	70%
Periodic	59	19	0	78	78%
Intensive	24	2	2	28	28%
Total	251	89	5	345	
Recall	67%	21%	40%		

3.7 Accuracy of Classification

Table 7 summarizes the accuracy of classification made by the decision tree. The accuracy of our proposed scheme is evaluated with a precision P and a recall R, defined ordinarily as

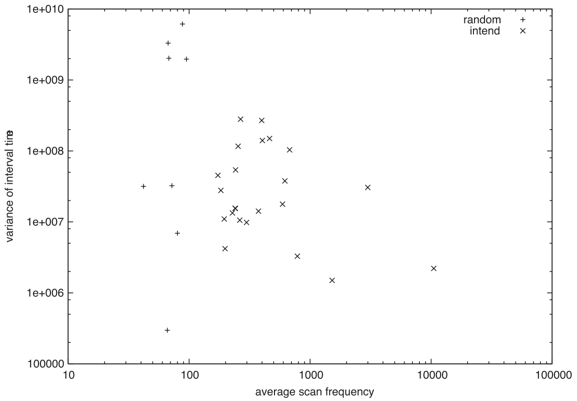

Fig. 9 Scatter diagram between mean scan counts C_* and variance of inter-arrival time $\sigma(\omega_*)$.

$$P = \frac{\text{\# of correct classifications}}{\text{\# of classified scans}},$$
$$R = \frac{\text{\# of correct classifications}}{\text{size of the true class}}.$$

In the evaluation, we regard the manually determined classification as the true class, although it may contain some unstable and inconsistent decisions. The random classes are mostly correctly classified with a precision of 70% and a recall of 67%. However, 60% of intensive scans are wrongly classified into other classes. The misclassifications may happen especially in the smaller subset.

To investigate the reason for the misclassification, we have a sample of the demonstration about the distribution of A_{24} as a scatter diagram in **Fig. 9**, where horizontal and vertical axes correspond to the classification attributes in the decision tree. For example, the X-axis indicates values of the root attribute, $\mu(C_*)$ and Y gives the 2nd-highest attribute, $\sigma(\omega_*)$ in Fig. 7. In the diagram, we can see two subsets for the random (left) and intensive scans (right).

Figure 10 demonstrates a more complicated example of the sample data that are divided into three classes with threshold $k = 15$ and $\sigma(C_*) = 0.6$. The threshold values correspond to the two nodes from the right bottom of Fig. 7, where any scans with k more than 15 are classified as *Periodic*, corresponding to the upper half of Fig. 10. The lower half is divided into *Intensive* (left) and *Random* (right) by the condition whether $\sigma(C_*)$ is greater than 0.6 or not. We see that the thresholds are so properly specified that most scans are classified correctly.

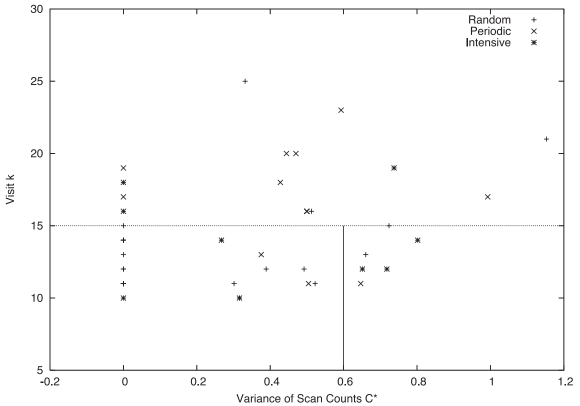

Fig. 10 Scatter diagram between $\sigma(C_*)$ and visit k.

3.8 Performance of Classification

According to Ref. 15), *SWorD* detects 51% of the infected hosts within the first 10 seconds, and spends 30 seconds to detect 88% of all the infected hosts. Although *SWorD* focuses on identifying infected hosts from a worm outbreak rather than classifying scanners, the detection algorithm based on statistical testing is close to ours.

Both *SWorD* and ours use a combination of logical expressions to identify/classify cases based on the statistics of traffic, e.g., a unique destination counts, and an average inter-arrival time. The proposed decision tree allows to express a more complicated logic function than does *SWorD*, whose detection algorithm was carefully ran manually. For example, the decision tree in Fig. 7 contains 10 nodes to classify scanners, while Figs. 2 and 3 in Ref. 15) have 4 **if-then** rules. We summarize the comparison of these two logical expressions in **Table 8**.

A decision tree is a light-weight data structure that can be evaluated quickly. Based on an experiment with the tree in Fig. 7 in a platform (Xeon 2.3 GHz,

Table 8 Comparison in terms of logic to classify (detect).

	SWorD[15]	Ours (Fig. 7)
number of tests	4	10
tier (height of tree)	2	6
rule generation	heuristics	automated

1 GB), the tree classifies 47,800 addresses per second. Although it is impossible to compare *SWorD* and ours with exactly the same environment and traffic flow, we estimate that the proposed tree is able to investigate 470 k addresses for 10 seconds while *SWorD* detects 51 infections. We believe the performance of classification is fast enough to classify the scanners.

4. Conclusions

Based on observations of distributed sensors, we have presented an automated classification of port scans into three classes, namely periodic, intensive, and random scans. Our main results are: (1) port scans are classified into three classes with distinct statistics, (2) a decision tree has good performance in classifying the worms with some precision from 28% to 70% and recall from 21% to 67%; (3) the number of distinct destination addresses for a source address, called *visit k*, determines how to perform port scans, that can be approximated by the simple equation $h(k) = h_0/k^{3.5616}$ according to the Zipf's law, and (4) the most common scans are the random (80%), followed by the periodic (18%) and the intensive (2%).

Acknowledgments We thank Mr. Masato Jingu of Tokai University for his contribution to the experiment. We thank Mr. Taichi Sugiyama and Prof. Norihisa Doi of Chuo University for the discussion, and the JPCERT/CC for providing the ISDAS observation data in order that we could evaluate the proposed model. We thank anonymous reviewers and proofreaders.

References

1) Symantec Security Response (2007). http://www.symantec.com/security_response/
2) Moore, D., Paxson, V., Savage, S., Shannon, C., Staniford, S. and Weaver, N.: Inside the Slammer Worm, *IEEE Security & Privacy*, pp.33–39 (July 2003).
3) Shannon, C. and Moore, D.: The spread of the Witty worm, *IEEE Security & Privacy*, Vol.2, No.4, pp.46–50 (Aug. 2004).
4) Changchun Zou, C., Gong, W. and Towsley, D.: Code Red Worm Propagation Modeling and Analysis, *ACM Computer and Communication Security Symposium* (*CCS*) *2002* (Nov. 2002).
5) The Distributed Honeypot Project: Tools for Honeynets. http://www.lucidic.net
6) Moore, D., Shannon, C., Voelker, G. and Savage, S.: Network telescopes: Technical report, *Cooperative Association for Internet Data Analysis* (*CAIDA*) (July 2004).
7) SANS Institute: Internet Storm Center. http://isc.sans.org
8) DShield.org: Distributed Intrusion Detection System. http://www.dshield.org
9) JPCERT/CC: Internet Scan Data Acquisition System (ISDAS). http://www.jpcert.or.jp/isdas
10) Kumar, A., Paxson, V. and Weaver, N.: Exploiting Underlying Structure for Detailed Reconstruction of an Internet-scale Event, *ACM Internet Measurement Conference 2005* (2005).
11) Ishiguro, M., Suzuki, H., Murase, I. and Shinoda, Y.: Internet Threat Analysis Methods Based on Spatial and Temporal Features, *IPSJ Journal*, Vol.48, No.9, pp.3148–3162 (2007).
12) Wang, J., Wang, Z. and Dai, K.: IDS, content filtering, Java, etc.: A network intrusion detection system based on the artificial neural networks, *2004 Proc. 3rd International Conference on Information Security* (*InfoSecu '04*), pp.166–170, ACM (2004).
13) Laskov, P., Gehl, C., Kruger, S. and Muller, K.-R.: Incremental Support Vector Learning: Analysis, Implementation and Applications, *The Journal of Machine Learning Research*, Vol.7, pp.1909–1936 (2006).
14) Jung, J., Paxson, V., Berger, A.W. and Balakrishnan, H.: Fast Portscan Detection Using Sequential Hypothesis Testing, *Proc. 2004 IEEE Symposium on Security and Privacy* (*S&P'04*) (2004).
15) Dunlop, M., Gates, C., Wong, C. and Wang, C.: SWorD — A Simple Worm Detection Scheme, *OTM Confederated International Conferences: Information Security* (*IS 2007*), LNCS 4804, pp.17520–1769 (2007).
16) Quinlan, J.R.: *C4.5 Programs for Machine Learning*, Morgan Kaufmann, San Mateo, California (1992).
17) Zipf, G.K.: *Human Behavior and the Principle of Least-Effort*, Addison-Wesley, Cambridge MA (1949).
18) Nakakoji, H., Terada, M. and Susaki, S.: Proposal for the Visualizing and Quantitative Method of Searching Characteristics of Node, *IPSJ Journal*, Vol.48, No.9, pp.3163–3173 (2007).
19) Microsoft Security Bulletin MS04-011, Security Update for Microsoft Windows (2004). http://www.microsoft.com/technet/security/bulletin/MS04-011.mspx
20) VMWare Workstation (2007). http://www.vmware.com/products/ws/
21) Mitchell, T.: Machine Learning, McGraw Hill (1997).

22) Kikuchi, H., Fukuno, N., Kobori, T. and Terada, M.: Classification of Port-scans via Distributed Observation, *The 2nd Joint Workshop on Information Security* (*JWIS 2007*) (Aug. 2007).

(Received November 30, 2007)
(Accepted June 3, 2008)
(Released September 10, 2008)

Hiroaki Kikuchi was born in Japan. He received B.E., M.E. and Ph.D. degrees from Meiji University in 1988, 1990 and 1994. After he working in Fujitsu Laboratories Ltd. from 1990 through 1993, he joined Tokai University in 1994. He is currently a Professor in the department of communication and network Engineering, school of information and telecommunication Engineering, Tokai University. He was a visiting researcher of the school of computer science, Carnegie Mellon University in 1997. His main research interests are fuzzy logic, cryptographical protocol, and network security. He is a member of the Institute of Electronics, Information and Communication Engineers of Japan (IEICE), the Information Processing Society of Japan (IPSJ), the Japan Society for Fuzzy Theory and Systems (SOFT), IEEE and ACM.

Naoya Fukuno was born in Japan. He received the B.E. and M.E. in Tokai University in 2005 and 2007, respectively. He joined the Internet Security Systems K.K. in 2007 and is currently a researcher of IBM Japan Ltd.

Tomohiro Kobori was born in Japan. He received B.E. from Tokai University in 2007. He is currently a graduate student of Tokai University.

Masato Terada was born in Japan. He received the M.E. in Information and Image Sciences from Chiba University, Japan, in 1986. He joined Hitachi, Ltd. in 1986. He is currently the Chief Researcher at the Security Systems Research Dept., Systems Development Lab., Hitachi. Since 2002, he has been studying at the Graduate School of Science and Technology, Keio University and received the Ph.D in 2006. Since 2004, he has been with the Hitachi Incident Response Team. Also, he is a visiting researcher at the Security Center, Information - Technology Promotion Agency, Japan (ipa.go.jp), JVN associate staff at JPCERT/CC (jpcert.or.jp) and a visiting researcher at the Research and Development Initiative Chuo University as well.

Tangtisanon Pikulkaew received B.E., and M.E. degrees from King Mongkut's Institute of Technology Ladkrabang in 2003 and 2005. After she worked in ITONE Company from 2003 through 2004, she has joined King Mongkut's Institute of Technology Ladkrabang in 2004 as a research assistant. Since 2005, she has been with Information Engineering department at King Mongkut's Institute of Technology Ladkrabang, Thailand, where she is presently a lecturer. Her recent research interests include network security, image processing and web application.