SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks. (2008)
Published online in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/sec.51

Combating against Internet worms in large-scale networks: an autonomic signature-based solution

Kumar Simkhada[1‡], Tarik Taleb[2*,†], Yuji Waizumi[2], Abbas Jamalipour[3] and Yoshiaki Nemoto[2]
[1]*KDDI Corporation, Japan*
[2]*Graduate School of Information Sciences, Tohoku University, Japan*
[3]*Department of Electrical and Information Engineering, University of Sydney, Australia*

Summary

In this paper, we propose a signature-based hierarchical email worm detection (SHEWD) system to detect e-mail worms in large-scale networks. The proposed system detects novel worms and instantly generates their signatures. This feature helps to check the spread of any kind of worm—*known* or *unknown*.

We envision a two-layer hierarchical architecture comprising local security managers (LSMs), metropolitan security managers (MSM), and a global security manager (GSM). Local managers collect *suspicious* flows and hand them to metropolitan managers. Metropolitan managers then use cluster analysis to sort worms from the suspicious flows. The sorted worms are used to generate the worm signature which is relayed to the global manager and then to all the collaborating networks. A separate scheme is proposed to automatically select suitable values of the system parameters. This parameter selection procedure takes into account the current network state and the *threat level* of the ongoing attack. The performance of the whole system is investigated using real network traffic with traces of worms. Experimental results demonstrate that the proposed scheme is capable to accurately detect email worms during the early phase of their propagations. Copyright © 2008 John Wiley & Sons, Ltd.

KEY WORDS: Internet worms; network security; worm signature; clustering

1. Introduction

Internet worms have caused serious damage to network resources in recent years [1]. Worms such as Code Red, Slammer, Nimda, Witty, Beagle, NetSky, and MyDoom have infected millions of computers and have resulted in the loss of billions of dollars [2–5]. Worms can delete, steal, or modify files that belong to the infected hosts. They can also launch distributed denial of service (DDoS) attacks to specific targets. Once they infect a host, they search for other vulnerable targets by scanning, passive monitoring, and *hit-listing* [5–7]. Along with the growing popularity of e-mails, recent years have seen a significant rise in the number of email worms. Protection of networks against e-mail worms forms the focus of the research work outlined in this paper.

Current intrusion detection systems (IDSs) are based on approaches such as honeypot-based analysis, taint analysis, traffic analysis, and signature matching.

*Correspondence to: Tarik Taleb, Graduate School of Information Sciences, Tohoku University, 6-6-11-304 Aramaki-aza-Aoba, Aoba-ku, Sendai, Miyagi, Japan 980-8579.

†E-mail: taleb@aiet.ecei.tohoku.ac.jp

‡ He was with the Graduate School of Information Sciences (GSIS), Tohoku University, Japan when this research was carried out.

Signature matching remains is one of the most popular techniques on account of its scalability, accuracy, and cost-effectiveness. However, signature-based approaches are applicable to only *known* worms. Furthermore, they involve cumbersome work and require significant time until an *effective* and *accurate* signature is generated. An effective worm control is possible only when the detection and signature generation processes are carried out automatically.

In this paper, we present a signature-based hierarchical e-mail worm detection (SHEWD) scheme to detect and defend against e-mail worms. The proposed system, which is an extension of works presented in References [8,9], automatically detects novel worms through an anomaly analysis-based module. The newly detected worms are used to automatically generate the signature for the propagating worm. The generated signature is then added to a signature database to check the spread of the worm through a cost-effective and accurate signature matching mechanism.

In the proposed SHEWD system, the network topology is divided into a number of metropolitan areas. Each metropolitan area consists of a metropolitan security manager (MSM) and a number of local networks. Local networks are managed by their respective local security managers (LSMs). A global security manager (GSM) controls the whole network topology and directly communicates with MSMs. LSMs search for any *worm-like* or *suspicious* e-mails propagating in their networks, and report such e-mails to their corresponding metropolitan managers. A MSM uses cluster analysis to sort worms out of the suspicious contents transmitted by its local managers. The MSM then automatically generates signature from the sorted worms by using a differentiation-based approach presented in Reference [10]. It then sends the generated signature to the GSM. Upon receiving signatures from MSMs, the GSM relays the generated signature to all local networks via MSMs.

The proposed system dynamically moderates also the overall security of the whole network. High-hierarchy components define the level of threat posed by worms in their networks. Based on this information, local managers change the security level in their networks by adjusting three performance metrics, namely false negative avoidance (FNA), false positive avoidance (FPA), and performance overhead (PO). LSMs define the minimum levels of FPA and FNA, and the maximum level of PO they can tolerate under the corresponding threat level. They then search for the optimum parameter values that can satisfy the defined performance requirements. This enables SHEWD to automatically detect novel worms and to instantly generate their signatures.

The remainder of this paper is organized as follows. Section 2 surveys some previous research work related to worm detection. Section 3 portrays the envisioned architecture under which the proposed scheme is deployed. It also presents a detailed description of the *suspicious* or *worm-like* flow collection algorithm, the hierarchical agglomerative-clustering method for sorting worms, and the automatic signature generation process. Section 4 presents an approach to automatically select optimum values for the system parameters. The performance of the proposed scheme is evaluated in Section 5. Section 6 concludes the paper.

2. Background and Related Work

Detection of Internet worms is a hot topic in recent literature. Worm detection techniques are classified as either misuse-based or anomaly based. In a misuse detection system, properties of a *known* worm are studied and the corresponding signature is generated. IDSs match pre-defined worm signatures with the network traffic to check if any worm content is present in it. Misuse-based detection schemes are highly accurate in detecting *already known* worms. However, their inability to detect *novel* worms urges current IDSs to consider alternative approaches. Anomaly detection techniques are considered powerful as they can detect *novel* worms too. However, they are more error-prone and require significant resources compared to misuse-based approaches. An effective worm detection is possible with the integration of signature-based and anomaly based schemes in a single worm detection system [11]. We adopt a similar concept in the proposed SHEWD system whereby a cost-effective signature-based module detects known worms and an anomaly based module detects novel worms.

Anomaly based worm detection is possible through several approaches, such as honeypot-based detection [12,13], data mining [14], protocol framing, traffic pattern analysis, taint analysis [15–17], and content analysis [18–21]. HoneyStat [12] and honeycomb [13] use honeypots to detect worms. Honeypots are capable of checking the presence of worms in real traffic. Depending on the nature of the propagating worm, honeypots require a long period of time until they can detect the worm. Furthermore, there is a significant risk in using highly interactive honeypots as they leave their operating systems entirely exposed to attacks. Worms can exploit this weakness to attack other targets. Data mining is another worm detection technique [14]. One major drawback of data mining technique is the high cost they incur. Several worm detectors use taint analysis to detect worms that initiate buffer overflow attacks on their host computers [15–17]. These systems retrieve inputs from unreliable sources and track the data affected by such inputs. Throttling is also a possible approach to delay worm propagation [22]. This method limits the rate of the number of messages a computer can send to different destinations. This approach, however, incurs some delay to communication and needs to be deployed across all networks to be effective. The research work presented in Reference [24] matches inbound and outbound network packets destined to the same port to check if any worm packet is propagating from the network. This system is based on the fact that a specific worm tries to exploit a specific vulnerability and is thus more likely to communicate to a certain port. Malicious e-mail tracking system (MET) [25] filters worm attachments from e-mails. It computes MD5 hash for every binary attachment and checks if the same attachment is propagated in the network traffic. This system is, however, inefficient in case of polymorphic worms. In Reference [26], Zou *et al*. present an architecture design of a *feedback e-mail worm defense system* to protect e-mail users in enterprise networks. In this system, *suspicious* e-mails are stopped and sent to a server where their behaviors are statistically investigated. If an attachment is not cleared as innocuous, it is sent to a honeypot. The honeypot then verifies whether the attachment is contaminated.

A high traffic volume with identical contents is a strong indication of a worm propagation. This fact is used by several content-based worm detectors [18–20]. In Reference [18], Akritidis *et al*. present a scheme which generates Rabin fingerprints of packet payloads and tests if same content is carried by several flows. Autograph [19] and EarlyBird [20] also search for repetitive contents in worm payloads to generate worm signature by using Rabin fingerprints. These systems generate single substring signatures. However, single substring signatures are not effective in detecting polymorphic worms. Polygraph [21] addresses this problem by using multiple substrings as signature. In Reference [10], the authors also generate multiple substring signature by differentiating normal flows from worm flows.

Most of the above-mentioned techniques are deployable only at local networks. Information sharing amongst various IDSs is necessary in order to effectively control a worm spread at the early stage of its propagation. Vigilante [23] is an end-to-end approach to contain worms automatically. In this system, local hosts run a software to detect worms and broadcast *self-certifying alerts* to other hosts. In Reference [27], Chen and Tang suggest collecting Internet control message protocol (ICMP) host unreachable packets from edge routers of Internet service providers (ISPs) to detect scans. By blocking scans, scanning worms are prevented from penetrating into their target networks. Reference [28] proposes an approach where scan data are collected from several networks to monitor the global propagation of scanning worms. Distributed overlay for monitoring Internet outbreaks (DOMINO) [29] and intrusion detection and rapid action (INDRA) [30] are two architectures designed to detect intrusions in a distributed environment. INDRA is a peer-to-peer (P2P)-based distributed IDS. Interested and trusted peers share information about any intrusion attempts directed to them. Thus, peers that are not yet attacked have some time to equip themselves with appropriate defense strategies. On the other hand, DOMINO is a hybrid of hierarchical and P2P architecture. It comprises also active sinknodes that search for packets directed to unused IP addresses. Monitoring unused IP addresses enables detection of random scanning worms.

The proposed SHEWD system is hierarchy based. The main rational beneath considering a hierarchical system consists in the associated simplicity in deploying security policy. Furthermore, the proposed system uses payload-content analysis which adds a constraint to co-operating entities for information sharing. Hence, we opt for a hierarchical information sharing approach among the IDSs of the same enterprises instead of a P2P approach. Some recent worms strategically bias their propagation strategy to nearby networks for a better chance of propagation [6]. With the growing use of mailing lists within the same enterprises, e-mail worms may also show similar tendency. It is envisioned that a hierarchical system is a better choice over a P2P system to combat against such worms.

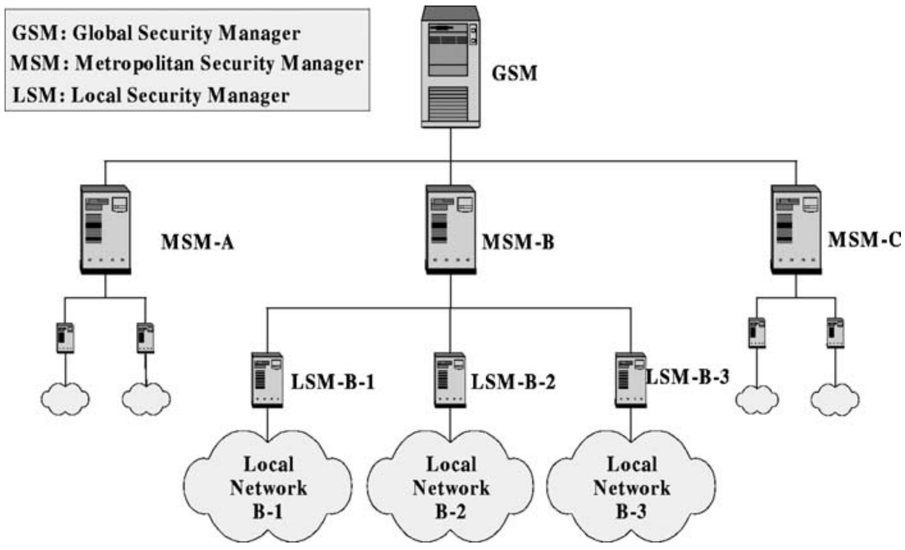

Fig. 1. The envisioned two-layer hierarchical worm detection architecture.

3. A Signature-based Hierarchical E-mail Worm Detection (SHEWD) System

3.1. Architecture

The overall worm detection architecture of the proposed SHEWD scheme is illustrated in Figure 1. The whole network topology is divided into a number of metropolitan areas. Each metropolitan area consists of a MSM and a number of local networks. Each local network is monitored by a LSM. A GSM, which lies on top of the topology, directly connects with metropolitan managers. LSMs sniff the traffic at their networks to check for any suspicious activity. When a LSM senses any anomalous content in an inbound e-mail, it reports the corresponding e-mail flow to its corresponding MSM as a suspicious flow. A MSM sorts worms from the pool of suspicious flows collected from its corresponding LSMs. It then generates a highly accurate signature for the propagating worm and reports the signature to the GSM. The GSM further refines the signature before relaying it to all local networks *via* metropolitan managers.

The architecture of a typical local manager is depicted in Figure 2. It consists of three units, namely signature update unit (SUU), anomaly detection unit (ADU), and multi-level automatic parameter selector (MAPS). The SUU unit verifies if the incoming network traffic contains any known worms or whether any known normal traffic is present. The detected worm is instantly blocked, and the *SUU filtrate* is

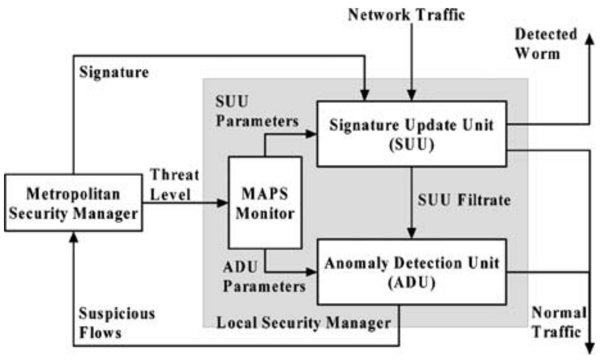

Fig. 2. Basic operations of worm detection procedure at local security managers.

subject to anomalous analysis at the ADU unit. ADU searches for flows that carry similar contents. Such flows are considered *suspicious* and are reported to high hierarchy metropolitan managers. The suspicious flow collection algorithm is discussed in detail in Section 3.3.

The MSM sorts worms from the reported suspicious flows. It then generates worm signature for the propagating worm. The SUU unit of local manager receives the generated signature from the GSM and the corresponding MSM. Further detection of the worm becomes then possible at the SUU unit of LSM through a simple signature matching. The proposed worm sorting algorithm and signature generation procedures are discussed in detail in Section 3.4 and Section 3.5, respectively.

The MAPS monitor selects optimum values for the parameters used by the SUU and ADU units. It receives from MSM the current level of threat posed by worms. Threat level, whose value ranges from 1 to 5, reflects the current worm activity in the network. Level 1 indicates that the network is under normal condition and no imminent worm activity is anticipated. On the other hand, level 5 suggests that the network is in a grave danger of a worm intrusion. Based on this information, the MAPS monitor adjusts the levels of three performance metrics, namely FNA, FPA, and PO. Under normal network conditions, the system gives priority to increasing FPA level and lowering PO level while maintaining a reasonable level of FNA. When the threat level is high, the system bears higher PO level in order to increase the FNA level. The parameters that best satisfy the required performance metrics are sent to the SUU and ADU units for deployment. Depending on the network conditions, the parameter value search is carried out in a dynamic way. The security of the network is thus automatically adjusted. Section 4 describes in detail the overall working of MAPS.

3.2. Initial Filtering of Pre-defined Traffic

The SUU unit stores all the signatures. It is updated with any new signature received from high hierarchy entities. The network administrator can also directly maintain the signature database. This unit discards any traffic that contains the signature patterns that are defined in the database. SUU can be also updated with normal signatures so that the normal traffic can be instantly forwarded to their destinations without having to go through the anomaly check in the ADU unit. Such an operation can considerably reduce false positives. In our experiment, we omit the operation of the normal-traffic definition to focus our investigation on the detection accuracy of the ADU unit in a challenging scenario where such assistance is not provided.

3.3. Collection of Suspicious Traffic

We next discuss the working of the ADU unit. ADU analyzes the SUU filtrate and collects worm-like or suspicious flows. It then sends the suspicious flows to the corresponding high-hierarchy metropolitan manager. The ADU unit exploits some intrinsic characteristics of worms. The ADU unit is provided also with a database of normal flows. These normal flows are used as reference to not mistakenly label normal activities as *suspicious*. When a worm is actively propagating in the Internet, similar contents are transmitted across various flows in the network traffic. The ADU unit extracts sample character strings from each flow in the SUU filtrate and uses them as analysis features. These strings are of constant length (L_S). A maximum number N_S of such strings are extracted from each flow. Every time a string is extracted from a flow, the ADU unit matches the extracted string with normal flows. If the string is normal, it is dropped and a new string is extracted to replace it. The ADU unit stores these sample strings and takes account of their occurrence frequencies in the traffic. Notations used in this paper are listed in Table I.

Let $\mathbf{A}$ be the set of these sample strings stored by the ADU unit at time t. If a sample string ($s \in \mathbf{A}$) exists in a flow F_j of the SUU filtrate, its *occurrence frequency* f_s is incremented by one. If only n ($0 \leq n < N_S$) sample strings exist in F_j, ($N_S - n$) sample strings are further extracted from F_j and are added to $\mathbf{A}$. String extraction from a flow is stopped if the ADU unit fails to find *non-normal* strings in it even after a certain number of trials (N_{TR}). Sample strings with high occurrence frequencies are likely to be part of worms. Alerts are thus generated for flows that carry strings whose occurrence frequencies exceed a pre-defined *repetitive occurrence threshold* (Δ_{TH}). In order to save buffer storage, relatively *old* sample strings are deleted from $\mathbf{A}$. Hence, only sample strings observed during a pre-defined *string caching time* (Θ_T) remain throughout the analysis. Alerted flows are considered suspicious and are sent to high-hierarchy MSM managers for further analysis.

3.4. Worm Sorting by Hierarchical Agglomerative Clustering

When there is an active propagation of a worm, a MSM is likely to receive similar reports from its corresponding LSMs. However, this does not rule out the possibility that LSMs may mistakenly report innocuous flows as suspicious. Hence, it is necessary to sort genuine worm flows in order to generate a highly accurate signature. We use cluster analysis to sort a reasonable number of worm flows from the pool of suspicious flows. These sorted worms are later used for signature generation.

We assume that there are N flows in the suspicious flow pool. The ith flow is expressed as a point in a 256-dimensions feature space as:

$$x_i = (x_{i,0}, x_{i,1}, \ldots, x_{i,255}) \quad (1)$$

Table I. Notations used in this paper.

Operation	Notation	Definition
Collection of suspicious flows at LSM	L_S	Length of each sample string
	N_S	Number of sample strings extracted per flow
	N_TR	Maximum number of string-extraction trials
	Θ_T	String caching time
	Δ_TH	Repetitive occurrence threshold
Worm sorting at MSM	N	Total number of flows in the suspicious flow pool
	c_w	Worm concentration of the suspicious flow pool (ratio of total worms in the suspicious flow pool to N)
	d_{ij}	Euclid distance between flows i and j
	α	Final cluster factor
	n_n	Number of worms collected in the worm cluster
	n_w	Number of normal flows collected in the worm cluster
Signature generation at MSM	$\mathbf{S}_\mathrm{C}$	Set of strings common to worm flows
	$\mathbf{S}_\mathrm{N}$	The set of normal tokens
	$\mathbf{S}_\mathrm{S}$	Signature (the set of signature substrings)
	L_MIN	Minimum length of signature substrings
	T_SIG	Signature generation time
	S_AL	Minimum number of signature substrings that should exist in a flow to generate an alarm
	ATL	Attack tolerance level
Automatic parameter selection (at MAPS)	η	False negative avoidance (FNA)
	σ	False positive avoidance (FPA)
	θ	Performance overhead (PO)
	$E_{\eta,k}$	Partial FNA gain for quadruple k
	$E_{\sigma,k}$	Partial FPA gain for quadruple k
	$E_{\theta,k}$	Partial PO gain for quadruple k
	E_k	Total gain for quadruple k

where $x_{i,k}$ indicates the occurrence probability of the ASCII code k in flow. Euclid distance is used to evaluate the similarities between the suspicious flows. The Euclid distance between flows i and j, denoted by d_{ij}, can be obtained from the following equation:

$$d_{ij}^2 = |x_i - x_j|^2 = \sum_{k=0}^{255}(x_{i,k} - x_{j,k})^2 \qquad (2)$$

We use agglomerative hierarchical clustering technique to sort worms from the suspicious flow pool. In this clustering method, each point is initially considered as an individual cluster. The nearest clusters are joined to form a larger cluster, consequently reducing the number of clusters by one in each clustering step. In the proposed scheme, the initial N clusters are clustered until the number of clusters reaches (αN), where α denotes the final cluster factor. The largest cluster among the (αN) clusters is considered to be the worm cluster, as illustrated in Figure 3. The flows represented by the points in the worm cluster are used further to generate signature for the spreading worm. Three types of distances are used to compare the similarity between clusters.

(1) Nearest neighbor distance (D_N): The nearest neighbor distance between clusters R and S is defined as:

$$D_\mathrm{N}(\mathrm{R}, \mathrm{S}) = \mathrm{Min}\{d_\mathrm{rs}\} \quad (\mathrm{r} \in \mathrm{R}, \mathrm{s} \in \mathrm{S}) \qquad (3)$$

(2) Farthest neighbor distance (D_F): the farthest neighbor distance between R and S is defined as:

$$D_{\mathrm{F}(\mathrm{R},\mathrm{S})} = \mathrm{Max}\{d_\mathrm{rs}\} \quad (\mathrm{r} \in \mathrm{R}, \mathrm{s} \in \mathrm{S}) \qquad (4)$$

(3) Group average distance (D_A): the group average distance between R and S is defined as:

$$D_{\mathrm{A}(\mathrm{R},\mathrm{S})} = \frac{1}{n_\mathrm{R} n_\mathrm{S}} \sum d_\mathrm{rs} \quad (\mathrm{r} \in \mathrm{R}, \mathrm{s} \in \mathrm{S}) \qquad (5)$$

where n_R and n_S are the number of points in R and S, respectively.

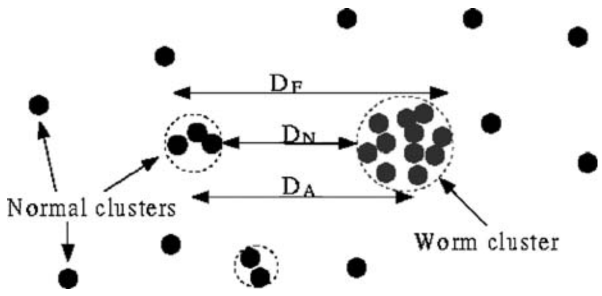

Fig. 3. Identifying the worm cluster from the suspicious flow pool: initial number of cluster ($N = 26$); final cluster factor ($\alpha = 1/2$); final number of clusters ($\alpha N = 13$).

Our preliminary experimental results show that all the aforementioned distances are practical for sorting worms at ($\alpha \geq 0.3$). As for values of α smaller than 0.3, accurate results can be obtained by using D_F. This is attributable to the fact that clusters formed by considering D_F are more spherical than the other two. Hence, we use D_F in the proposed system.

The following two parameters are used to evaluate the sorting algorithm:

(1) Sorting success rate (SSR): it reflects the percentage of worms collected in the worm cluster and is expressed as:

$$\text{SSR} = \frac{n_w}{c_w \times N} \tag{6}$$

where n_w denotes the number of worms in the worm cluster and c_w is the worm concentration (ratio of the number of worm flows to the total number of flows) in the whole suspicious flow pool.

(2) Sorting accuracy (SA): it defines the percentage of worms flows in the worm cluster. It is expressed as:

$$\text{SA} = \frac{n_w}{n_w + n_n} \tag{7}$$

where n_n is the number of normal flows in the worm cluster.

The final number of clusters is determined by α and is equal to (αN). Setting α to an appropriate value is vital. In order to generate appropriate worm signature, MSM needs to sort a reasonable number of worms. This is also governed by the worm concentration (c_w) in the suspicious flow pool. Here, we first investigate the accuracy of the proposed worm sorting algorithm with respect to α.

The sorting accuracy depends highly upon the worm concentration (c_w) in the suspicious flow pool. The value of c_w is directly governed by the accuracy of local managers. The higher the accuracy of local managers in collecting suspicious flows, the more concentrated the suspicious flow pool collected at the metropolitan manager becomes. Figure 4 depicts the SSR and the SA of the proposed clustering method for different values of α when ($c_w = 0.5$) and ($N = 30$). Results for four different worms—Beagle, NetSky, SWEN.A, and MyDoom are plotted. It is clear that false positives appear only for ($\alpha < 0.3$). Similarly, a large proportion of worms can be sorted for ($\alpha \leq 0.5$). Given that only worms are to be used to generate the signature, we set α to 0.5, a *very safe* value.

The proposed scheme identifies the biggest cluster as the worm cluster. However, other clusters may also be useful in correctly sorting worms. For instance, when there are several worms spreading at the same time, the clustering technique is likely to cluster separate worms in separate clusters. Also, if the number of worms sorted in the worm cluster is not sufficient for signature generation, the administrator can also search for other worm flows in other big clusters.

3.5. Signature Generation and Implementation

Having described the worm collection and sorting processes, we now explain the signature generation operation carried out by MSM. We also discuss how the signature is utilized by LSMs.

Generating appropriate signature for a worm is a challenging task. Most of the present worm detectors use a single substring as signature. This renders them inefficient against worms that exhibit polymorphism. Polymorphic worms change parts of their payloads by altering the byte sequence order or by adding dummy contents into their payloads. They can easily deceive signature-based detectors by modifying the *signatured* portion of their payloads. We address this issue by adopting multiple substrings as parts of the signature. Furthermore, we use only parts of core-worm codes as signature substrings.

Our signature generation approach is based on two facts: (a) worms carry some invariable portions in their payloads, and (b) normal payloads do not carry worm strings. Accordingly, signature is generated in two phases. In the first phase, the MSM extracts strings that commonly exist in worms. Any of these strings that are judged normal are excluded in the second phase.

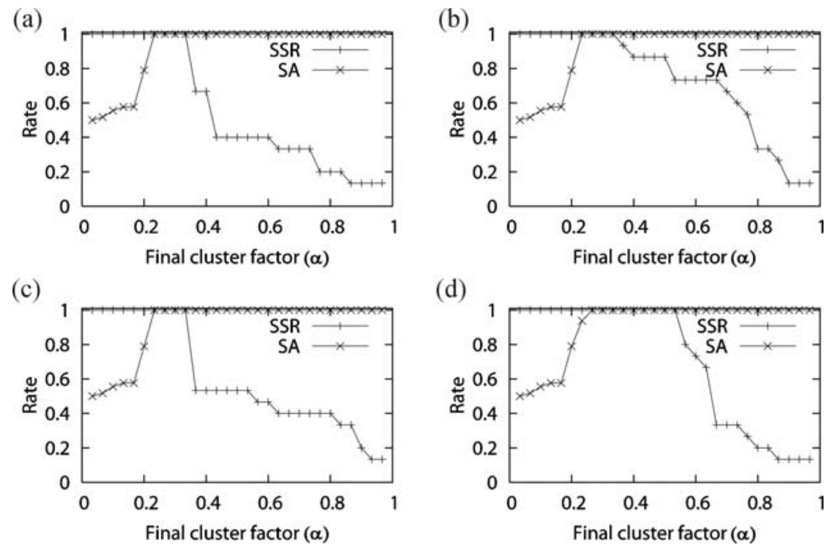

Fig. 4. Sorting success rate (SSR) and sorting accuracy (SA) as functions of the final cluster factor (α) for different worms ($c_{\mathrm{w}} = 0.5$). (a) Beagle (b) MyDoom (c) NetSky (d) SWEN.A.

Since too short strings may generate false alarms, a pre-defined minimum length threshold (L_{MIN}) is fixed to qualify these strings as parts of the signature.

Let N_{W} and N_{N} denote the number of worm payloads used for signature generation and the number of normal flows at a MSM, respectively. If X_i $(1 \leq i \leq N_{\mathrm{W}})$ is the set of strings that exist in the ith worm flow, the set of common strings among these worms is expressed as:

$$\mathbf{S_C} = \{t | t \in X_1 \cap X_2 \cap \cdots \cap X_{N_W}, \mathrm{len}(t) \geq L_{\mathrm{MIN}}\} \tag{8}$$

where len(t) denotes the length of string t. Similarly, if Y_j $(1 \leq j \leq N_{\mathrm{N}})$ is the set of strings existing in the jth normal flow, the set of normal strings is expressed as:

$$\mathbf{S_N} = \{t | t \in Y_1 \cup Y_2 \cup \cdots \cup Y_{N_{\mathrm{N}}}, \mathrm{len}(t) \geq L_{\mathrm{MIN}}\} \tag{9}$$

By excluding normal strings from the common strings, we get the set of signature substrings as:

$$\mathbf{S_S} = \mathbf{S_C} - \mathbf{S_N} \tag{10}$$

The generated signature thus contains $|\mathbf{S_S}|$ substrings each with a length longer than L_{MIN}. The worm signature is generated from the element substrings of $\mathbf{S_S}$. Hence, only core-worm strings that remain unchanged across several worms are used as signature substrings. As worm flows undergo analyses at two different levels, LSM and MSM, the generated signature is expected to be highly accurate. The signature accuracy can be further enhanced at the GSM when the latter receives signatures generated by various MSMs.

We now explain how the signature is implemented for actual worm detection. As previously explained, the SUU units of LSMs are relayed multiple substrings. Several signatures can be formed out of these strings. Weighting each substring, and combining some or all of them can generate different signatures. However, in a real network scenario, decisions regarding the use of signature substrings should be made instantly. So, we define a new parameter called *attack tolerance level* (ATL). ATL is defined as:

$$\mathrm{ATL} = \frac{S_{\mathrm{AL}}}{|\mathbf{S_S}|} \tag{11}$$

where $S_{\mathrm{AL}} (0 < S_{\mathrm{AL}} \leq |\mathbf{S_C}|)$ is the minimum number of signature substrings that should be detected in a flow in order to generate an alarm for the flow. Use of multiple substrings in the signature and adjustment of ATL makes the proposed approach robust in detecting polymorphic worms which try to hide core worm parts in between normal sequences. Indeed, even if worms alter some portions of their payloads, there still remain unaltered portions that match some signature substrings. The ATL parameter can be adjusted according to the level of threat posed by worms to the network. If there is an active spread of a specific worm, the administrator or any high-hierarchy entities such as MSM or the GSM can warn LSMs to set their ATL to a lower value. In this context, it should be recalled that high values of ATL guarantee low false positive rates.

4. Automatic Parameter Selection

As previously discussed, the proposed system uses several parameters during suspicious flow collection. These parameters include length of sample strings (L_{S}), number of sample strings extracted per flow (N_{S}), repetitive occurrence threshold (Δ_{TH}), and string caching time (Θ_T). During the signature generation process, the proposed system needs to fix the minimum length of signature substrings (L_{MIN}). During signature implementation, the value of attack tolerance level (ATL) should be also adjusted. Considering an online implementation of the proposed scheme, values of these parameters should be automatically set. In this section, we discuss how these parameters can be dynamically adjusted by the MAPS monitor to the optimum values so as the performance suits the prevailing network state.

4.1. IDS Performance Metrics

Various metrics can be used to evaluate the performance of an IDS. In Reference [31], Fink *et al.* point out some of these metrics, such as accuracy, detection rate, flexibility, overhead, error reporting, router interaction, timeliness, etc. Considering a worm detection scenario, FPA rate, and PO rate need primary follow-ups. Apart from these three metrics, we also choose the *signature generation time* as another performance metric when the signature generation process is involved. We next discuss these metrics and explain how they should be adjusted for an autonomic security management of networks.

4.1.1. False negative avoidance (FNA)

The FNA indicates the detection rate of the IDS. For each threat level L ($1 \leq L \leq 5$), critical FNA rate (minimum detection rate) η_L ($0 < \eta_1 < \cdots < \eta_5 < 1$) is pre-defined by the administrator. A high FNA is required when there is an active propagation of Internet worms. During such adversaries, the IDS system may have to sacrifice other two metrics to some extent.

4.1.2. False positive avoidance (FPA)

FPA indicates how fairly the IDS system allows normal traffic to pass undisturbed. FPA is expressed in terms of false positive rate (FPR) as, FPA $= 1 -$ FPR. As there is a tradeoff between FPA and FNA, FPA rate may need relaxation if FNA rate is increased. The desired minimum FPA rate of the system at each threat level is also defined by the administrator as σ_L ($1 > \sigma_1 > \cdots > \sigma_5 > 0$).

4.1.3. Performance overhead (PO)

Cost is another prominent factor in determining the performance of an IDS. It is thus necessary to design the IDS in such a way that maximum security and accuracy can be achieved with a nominal overhead. We define PO rate as the percentage of available buffer size required for analysis. Similarly to FNA and FPA, the critical PO rate (maximum affordable PO rate) of an IDS at each threat level (L) is defined by the administrator as θ_L ($0 < \theta_1 < \cdots < \theta_5 < 1$).

There is a trade-off between FNA, FPA, and PO rates. It is always preferable to achieve high FNA and FPA at the cost a low PO. However, during emergency situations such as worm outbreaks, it is necessary to sacrifice some FPA and PO in favor of FNA. In contrast, during normal network states, FNA rate is slightly reduced in order to achieve higher FPA and lower PO rates.

4.1.4. Signature generation time

Signature generation time (T_{SIG}) is a vital performance metric for signature-based IDSs. During high threat levels, IDSs can afford relatively short time for signature generation. The value of T_{SIG} for e-mail worms such as Beagle, NetSky, and MyDoom can be fairly long (a few minutes). However, for fast spreading scanning worms such as Code Red, Blaster, and Slammer, signatures are required within a limited span of time. Signatures for scanning worms should be generated within 60 s in order to effectively control their propagation [5]. This indicates that an IDS system has to automatically generate signature within a significantly limited time after the worm has been detected. Indeed, we first ensure that the signature is generated within the defined T_{SIG}. FNA, FPA, and PO rates are then considered.

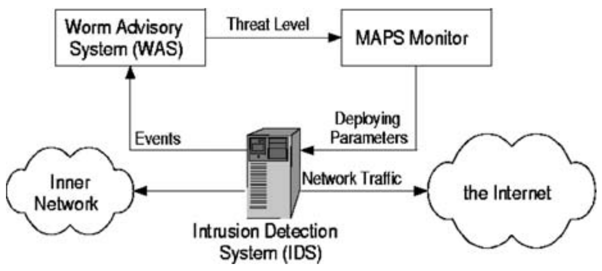

Fig. 5. A typical deployment of the proposed system.

4.2. Optimum Parameter Value Selection Algorithm

We next present the proposed multi-level security based autonomic parameter selection (MAPS) algorithm. A typical deployment of MAPS is illustrated in Figure 5. The IDS implemented at the gate of the network updates security events to a worm advisory system (WAS). WAS accordingly defines the *threat level* of the network. The MAPS monitor uses the defined threat level to find suitable parameters to advise the IDS. Note that MAPS does not propose a new worm detection scheme but *advises* the implemented IDS with *optimum* values of its parameters in order to efficiently defend against the spreading worm. The operations of WAS and MAPS are further explained below.

4.2.1. *Worm advisory system (WAS)*

The WAS defines the *threat level* of the network based on the events reported by the IDS or other entities, such as, firewalls, IDSs of other collaborating networks, and high-hierarchy components. It analyzes the events in timeslots. Threat level one corresponds to a normal network state when no malicious activities are reported. In contrast, threat level 5 implies that the system is either under attack or is in a grave danger of a worm propagation. The threat level of a given timeslot is decided by considering

(1) the total number of events observed in the timeslot with respect to a pre-defined *event threshold*, and
(2) the increasing or decreasing tendency of the number of events with respect to previous timeslots.

Given the fact that the WAS uses a simple statistical data of events to define the threat level, it can be directly handled also by the network administrator, if desired.

4.2.2. *MAPS monitor*

It is a common trend to investigate the performance of an IDS with test data before implementing it online. The MAPS monitor is provided with the results obtained when the IDS system is initially evaluated with such test data. Let $\mathbf{Q}$ be the set of these results. $\mathbf{Q}$ stores each scenario k as a quadruple $T_k = \{P_k, \eta_k, \sigma_k, \theta_k\}$, where P_k is the set of parameter values used and η_k, σ_k, and θ_k are the corresponding values of the three metrics in the test scenario k.

Figure 6 depicts the flow chart of the MAPS algorithm. The current threat level of the network is sent to MAPS by WAS. The critical rates of FNA, FPA, and PO, denoted by η_c, σ_c, and θ_c, respectively, are initialized to the critical values of the corresponding threat level, η_L, σ_L, and θ_L, respectively. The proposed algorithm takes a *greedy* approach in maximizing the FNA and FPA rates and minimizing the PO rate. Hence, the transitory critical values of these three metrics, are initialized as $\eta = 1$, $\sigma = 1$, and $\theta = 0$.

The optimum quadruple search is carried out in the order of PO, FPA, and FNA. At first, MAPS searches for quadruples in $\mathbf{Q}$ whose corresponding PO rates are within the transitory critical PO rate (θ). Thus, the set of quadruples ($\mathbf{Q}_1$), which fulfill the PO requirement, can be expressed as:

$$\mathbf{Q}_1 = \{T_k : T_k \in \mathbf{Q}; \theta_k \leq \theta\} \tag{12}$$

$\mathbf{Q}_1$ contains only the quadruples whose parameter can perform within the overhead θ. $\mathbf{Q}_1 = \phi$ implies that the IDS cannot function within the given transitory PO rate (θ). Thus, the MAPS monitor increases θ by $\Delta\theta$. If θ is within the critical PO rate (θ_c), it carries out another search from $\mathbf{Q}$. Otherwise, θ_c is incremented to $(\theta_c + \Delta\theta_c)$ and the whole search is repeated, but with a higher θ_c.

From the quadruples in $\mathbf{Q}_1$, MAPS then searches for the quadruples whose accuracy is more than the transitory critical FPA rate (σ). Hence, the set of the selected quadruples ($\mathbf{Q}_2$), which fulfill the PO and FPA requirements, is expressed as:

$$\mathbf{Q}_2 = \{T_k : T_k \in \mathbf{Q}_1; \sigma_k \geq \sigma\} \tag{13}$$

In case $\mathbf{Q}_2 = \phi$, the MAPS monitor decreases σ by a fixed value. If this decrease does not affect the relation ($\sigma > \sigma_c$), MAPS searches for appropriate quadruples from $\mathbf{Q}_1$ again. Otherwise, σ_c is relaxed and θ_c is

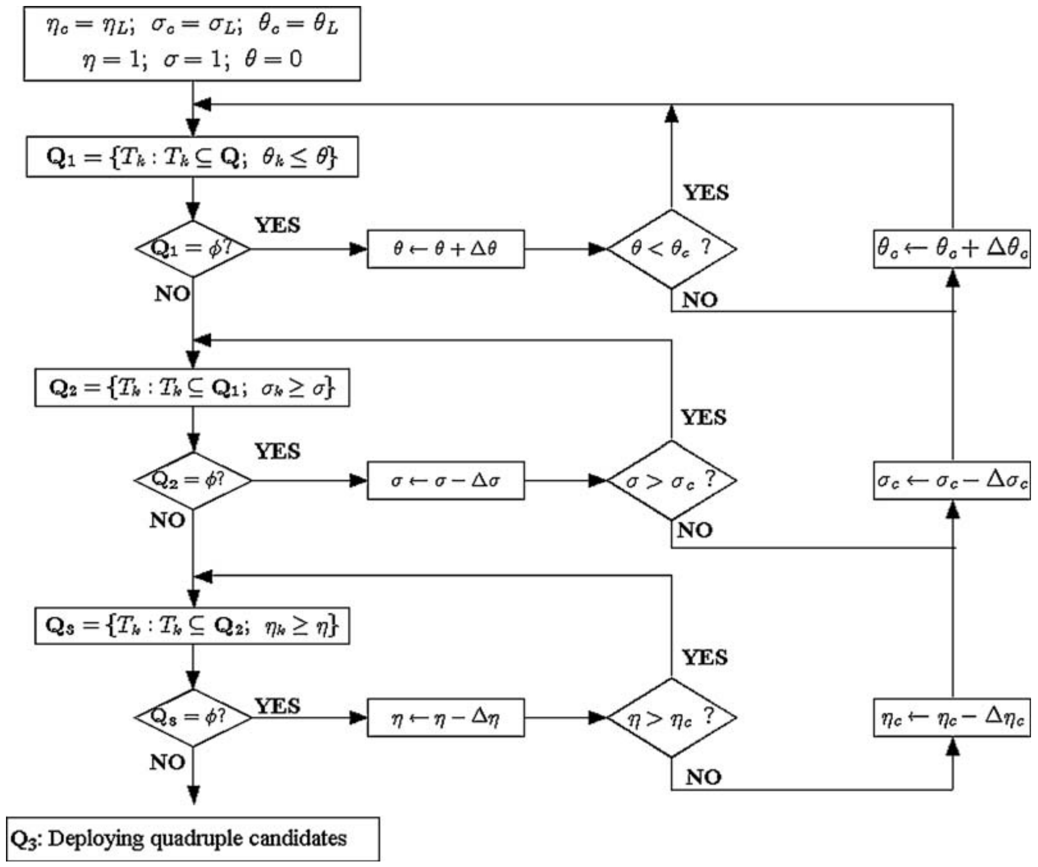
$\eta_c = \eta_L$; $\sigma_c = \sigma_L$; $\theta_c = \theta_L$
$\eta = 1$; $\sigma = 1$; $\theta = 0$
$\mathbf{Q}_1 = \{T_k : T_k \subseteq \mathbf{Q};\ \theta_k \leq \theta\}$
$\mathbf{Q}_1 = \phi$?
YES
NO
$\theta \leftarrow \theta + \Delta\theta$
$\theta < \theta_c$?
YES
NO
$\theta_c \leftarrow \theta_c + \Delta\theta_c$
$\mathbf{Q}_2 = \{T_k : T_k \subseteq \mathbf{Q}_1;\ \sigma_k \geq \sigma\}$
$\mathbf{Q}_2 = \phi$?
YES
NO
$\sigma \leftarrow \sigma - \Delta\sigma$
$\sigma > \sigma_c$?
YES
NO
$\sigma_c \leftarrow \sigma_c - \Delta\sigma_c$
$\mathbf{Q}_3 = \{T_k : T_k \subseteq \mathbf{Q}_2;\ \eta_k \geq \eta\}$
$\mathbf{Q}_3 = \phi$?
YES
NO
$\eta \leftarrow \eta - \Delta\eta$
$\eta > \eta_c$?
YES
NO
$\eta_c \leftarrow \eta_c - \Delta\eta_c$
$\mathbf{Q}_3$: Deploying quadruple candidates

Fig. 6. MAPS algorithm.

increased. The whole search is repeated with a lower σ_c and a higher θ_c. This process is repeated until ($\mathbf{Q}_2 \neq \phi$). Similar adjustment is carried out for FNA. MAPS searches for quadruples in $\mathbf{Q}_2$ that satisfy the FNA transitory critical rate (η). The set of the quadruples that are now chosen ($\mathbf{Q}_3$) is expressed as:

$$\mathbf{Q}_3 = \{T_k : T_k \in \mathbf{Q}_2; \eta_k \geq \eta\} \tag{14}$$

Similarly to the PO and FPA checks, MAPS confirms whether transitory critical FNA rate is satisfied. If $\mathbf{Q}_3 = \phi$, MAPS relaxes η to $\eta - \Delta\eta$. If the new transitory critical FNA rate is more than η_c, it carries out next search from $\mathbf{Q}_2$. Otherwise, η_c and σ_c are further reduced while θ_c is increased to start the search from the beginning. This is repeated until $\mathbf{Q}_3 \neq \phi$. MAPS uses $\mathbf{Q}_3$ to select the quadruple that is optimum for the current network state.

As $\mathbf{Q}_3$ may contain multiple quadruples, a separate parameter needs to be defined to select the best quadruple from $\mathbf{Q}_3$. We define partial gains of FNA, FPA, and PO for the quadruple T_k, denoted by $E_{\eta k}$, $E_{\sigma k}$, and $E_{\theta k}$, respectively, as:

$$\begin{aligned} E_{\eta k} &= \eta_k - \eta_L \\ E_{\sigma k} &= \sigma_k - \sigma_L \\ E_{\theta k} &= \theta_L - \theta_k \end{aligned} \tag{15}$$

where η_L, σ_L, and θ_L are the critical FNA, FPA, and PO rates, respectively, for the corresponding threat level L. Using the partial gains of the three metrics, the total gain of T_k is calculated as:

$$E_k = E_{\eta k} + E_{\sigma k} + E_{\theta k} \tag{16}$$

From $\mathbf{Q}_3$, the quadruple with the maximum value of E is selected. The parameter values of the selected quadruple are sent to the IDS for deployment.

5. Performance Evaluation

We now move on to evaluate the performance of the proposed scheme. We use real network data that contains traces of worms to investigate the accuracy of the proposed SHWD system to in collecting suspicious flows, signature generation time and its accuracy, and the efficiency of the automatic parameter selection algorithm.

5.1. Suspicious Flow Collection

We first investigate the accuracy of ADU unit of LSM in collecting suspicious flows. By setting L_S, N_S, and Δ_{TH} to different values, a number of scenarios can be considered. An off-line real network traffic is used as test data. The traffic is captured at a network which consists of 85 computer hosts and 160 actively used e-mail addresses. The test data consists of a total of 3054 flows on port 25 directed to the monitored network. Among these inbound flows, 34 flows are contaminated with the Beagle worm.

The length of token-caching time (Θ_T) depends upon the worm type. For a fast spreading worm such as Slammer and Code Red, setting Θ_T to a low value (in the order of seconds or minutes) is sufficient to collect enough worm flows. However, for e-mail worms such as Beagle, a reasonably longer time is required to gather worms. Hence, we fix Θ_T to 60 min.

The ADU unit searches for flows propagating similar contents. Thus, it is necessary to firmly study the nature of each flow. A flow is better scrutinized if a large number of sample strings are extracted from it. Hence, a high detection rate is expected when N_S is set to a high value. However, setting N_S to a high value also increases the probability of selecting normal strings as samples, thus increasing the risk of false positives. This is depicted in Figure 7(a) where the number of true positives increases as N_S takes higher values. In addition, increasing N_S results also in a higher number of false positives. We have shown in Section 3.4 that the proposed clustering scheme carried out at MSM can accurately sort worms even if the collaborating LSMs generate a certain level of false positives. In order to reduce the risk of false positives generated during a high N_S (in Figure 7(a)), increasing the value of the repetitive occurrence threshold (Δ_{TH}) is an option. Considering the fact that the proposed method searches suspicious flows from the inbound traffic where worms are scarce compared to outbound traffic, an alternative approach becomes necessary. Increasing L_S is such

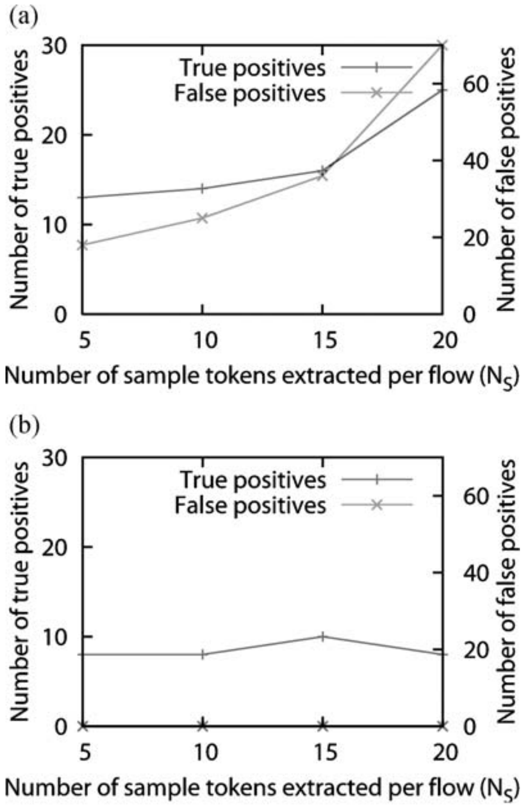

Fig. 7. Number of true positives and false positives for different numbers of sample tokens extracted per flow ($\Theta_T = 60$ min; number of worms = 34; number of normal flows = 3024). (a) $\Delta_{TH} = 4$, $L_S = 10$. (b) $\Delta_{TH} = 6$, $L_S = 30$.

an alternative approach. The probability of a string pattern appearing in a text decreases with an increase in its length. However, this rate of decrease is low for worm strings in worm texts than that for normal strings in normal texts. Although increasing L_S may reduce some true positives, we expect it to be advantageous because it is effective in reducing false positives. This is illustrated in Figure 7(b) which shows the numbers of true positives and false positives obtained when Δ_{TH} and L_S are increased from their previous values in Figure 7(a). False positives are completely suppressed while a reasonable number of true positives is maintained.

Adjusting the repetitive occurrence threshold (Δ_{TH}) may be a possible solution when false positives need to be oppressed. However, this approach also affects the suspicious flow collection. Figure 8(a) depicts such a case where false positives are completely suppressed for high values of Δ_{TH}. However, this has also led to a nominal number of true positives. This

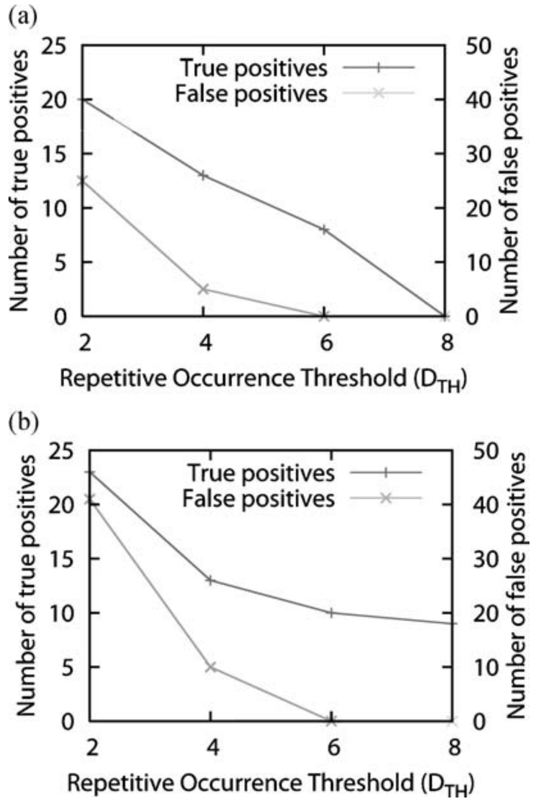

Fig. 8. Number of true positives and false positives for different repetitive occurrence thresholds ($\Theta_T = 60$ min; number of worms = 34; number of normal flows = 3024). (a) $L_S = 30$, $N_S = 5$. (b) $L_S = 30$, $N_S = 15$.

problem can be overcome by increasing N_S as shown in Figure 8(b). Indeed, by increasing N_S, the number of true positives increases while false positives remain suppressed.

Our experimental results for NetSky show that N_S and L_S should be set to values close to those for Beagle. Traces of NetSky and Beagle payloads are about 30–40 kb long when encoded by base64. In addition, both of these worms are mass-mailers and were in fact involved in a fierce battle [32]. These similarities account for the similar results obtained for these two worms. However, how to tune these parameters to best performing values during a novel-worm propagation is still an issue. As discussed in Section 4.1, it is necessary to loosen the PO and the false avoidance requirements in order to collect enough worm samples under such scenarios. After MSMs generate worm signatures by using these collected suspicious samples, these parameters can be switched to values that yield a fewer number of false positives.

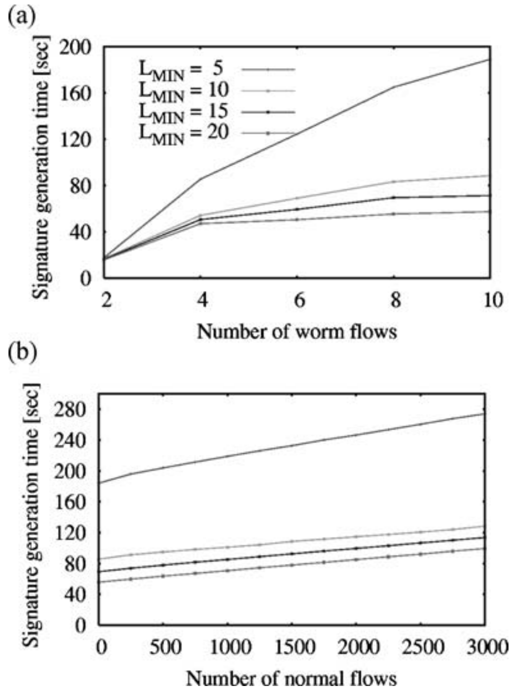

Fig. 9. Signature generation time (T_{SIG}) for various numbers of worms and normal flows used during signature generation. (a) T_{SIG} for various values of N_W ($N_N = 100$). (b) T_{SIG} for various values of N_N ($N_W = 10$).

5.2. Signature Generation

The signature generation is carried out at MSM. Worms sorted from the suspicious flow pool are used for this operation. The proposed signature generation scheme first extracts common substrings from worms. It then differentiates the set of these common substrings with the set of substrings present in normal flows. Here, we study the time required to generate worm signature. The signature generation time (T_{SIG}) depends more on the payload length than on the worm type. We use ten NetSky flows (with 30–40 kb payloads) for common substring extraction and 3000 normal flows for differentiation. Various scenarios are considered by changing the minimum length of substrings (L_{MIN}).

Figure 9(a) depicts the signature generation time for different numbers of worm flows when 100 normal flows are used for differentiation (i.e., $N_N = 100$). Signature generation time (T_{SIG}) increases with respect to the number of worm flows. However, this increase gradually becomes insignificant. This is because only a few strings are common among a large number of worm flows. Also, longer strings break into shorter strings as more worms are considered. For example, let '`aabbbcccc`' be a string common to worm flows 1–*i*. And let the $(i+1)$th flow contain a portion '`xbbbccdxer`'. Hence, the string common to flows 1–$(i+1)$ is '`bbbcc`', a relatively shorter string. With an increase in L_{MIN}, T_{SIG} shows a distinctly decreasing property. This is also attributable to the fact that fewer strings are considered when L_{MIN} is increased. This is depicted in Figure 9(b) which plots T_{SIG} for various numbers of normal flows under different settings of L_{MIN}. The number of worms used for signature generation is set to 10 (i.e., $N_{\mathrm{W}} = 10$). Figure 9(b) illustrates that T_{SIG} decreases by a significant margin as L_{MIN} is increased from 5 to 10. The signature generation time when the number of normal flows is zero is the common string extraction time. It is evident that extraction of common strings takes most of the signature generation time. From the experiment results shown in Figure 9, it is clear that signature for NetSky, a fairly long worm, can be generated within 2 min. For a shorter worm, signature can be generated relatively faster.

5.3. Detection Accuracy of the Generated Signature

In order to evaluate the accuracy of the generated signature, we used the signature generated for Beagle worm. The test data contain a total of 45 193 e-mail flows. Two hundred seventy-one of these e-mail flows are Beagle. As discussed in Section 3.5, different values of attack tolerance level (ATL) reflect different scenarios. We study the detection accuracy of the generated signature with respect to L_{MIN} for two scenarios. In Scenario 1, ATL = 0, a flow is considered worm if the flow contains at least one signature substring. We set ATL = 0.5 in Scenario 2, an alert is generated only if a flow contains more than 50% of the signature substrings.

Figure 10(a) depicts the numbers of true positives and false positives for various values of L_{MIN} in Scenario 1. A 100% detection rate is achieved for L_{MIN} below 175 byte. The number of false positives decreases as L_{MIN} increases. There are two reasons behind this phenomenon: (1) the probability of a string's occurrence probability in a text decreases with respect to its length, (2) the number of signature substrings decreases as L_{MIN} becomes longer. Figure 10(b) depicts the results for Scenario 2, where ATL = 0.5. Compared to the first scenario, false positives have significantly decreased. This is most clearly seen at $(5 \le L_{\mathrm{MIN}} \le 15)$. This implies

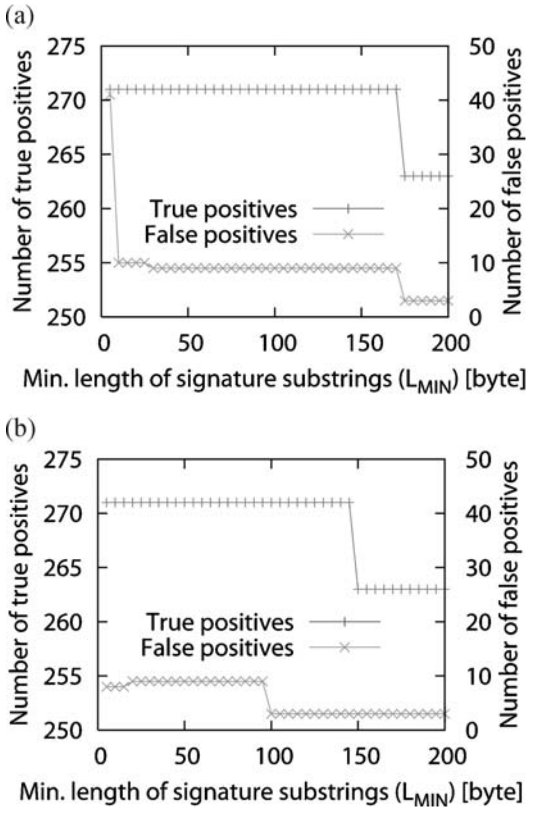

Fig. 10. Number of true positives and false positives for different values of minimum length of signature strings for a real network data that contain 45,193 flows out of which 271 are worms. (a) Scenario 1: ATL = 0. (b) Scenario 2: ATL = 0.5.

that the false positives generated when $(L_{\mathrm{MIN}} \le 15)$ in Scenario 1 were because of the *short* substrings. The fact that signature substrings are parts of worm codes, along with the requirement of half of signature substrings in order to generate an alarm, are the factors behind better results obtained in Scenario 2 than in Scenario 1.

The e-mails for which false positives were generated during $100 \le L_{\mathrm{MIN}} \le 175$ were normal e-mails with more than 2 MB of payloads. As they are significantly long, they are more likely to contain signature substrings. Besides, setting L_{MIN} to a high value results in discarding a large number of shorter worm substrings from the signature. As a result, many normal substrings which contained continuous ...`AA`... sequences in them were left in the signature. These substrings were the main reasons for the false positives generated against these normal e-mails. During low threat levels when FPA requirements are high, manually refining the signature files by deleting these *obviously* *normal* substrings can significantly improve the system performance.

5.4. Automatic Parameter Selection

In Section 5.1, we demonstrated that the proposed suspicious flow collection algorithm can successfully collect worm-like flows. However, considering the dynamics of the network state, different parameter sets should be chosen for different network conditions. We now investigate the MAPS monitor's ability to dynamically adjust parameter values according to the prevailing network state. The results obtained for various scenarios (under different values of N_S, L_S, and Δ_{TH}) in Section 5.1 are stored in a database. The FNA rate is expressed as the ratio of detected worms to the total number of detectable worms. Similarly, the FPA rate is the ratio of the number of normal flows that have been correctly judged as *normal* to the total number of normal flows. The IDS is supposed to set aside a maximum of 1000 Byte of buffer for each flow. The overhead rate per each flow is thus

$$\theta = \frac{L_S \times N_S}{1000} \tag{17}$$

The critical FNA rates from level 1–5 are set to $\{0.70, 0.80, 0.85, 0.90, 0.95\}$ while the critical FPA rates are fixed to $\{0.95, 0.90, 0.85, 0.80, 0.0\}$. Similarly, the critical PO rates (maximum buffer storage) are set to $\{0.2, 0.4, 0.5, 0.6, 1.0\}$.

Figure 11 depicts the FNA, FPA, and PO rates for different threat levels. In a normal network state (corresponding to a low threat level) the IDS has to ensure minimal false positives and to incur low cost. The FNA is, however, comparatively low. With an increase in the threat level, FNA also increases. Because there is a trade-off between these three metrics, an increase in FNA is achieved at the price of some additional overhead. However, the sacrificed metrics are maintained within their critical values.

We next consider a signature generation scenario. The signature generation time (T_{SIG}) basically depends on the worm payload length. For a worm with a short payload, a short span of time is sufficient to generate signature. In this experiment, we consider e-mail worms with about 30–40 kb of payloads. Thus, a relatively long time is allocated for signature generation. We set T_{SIG} to 105, 90, 75, 60, and 45 s for threat levels 1–5, respectively. We investigate the performance of the signatures generated within

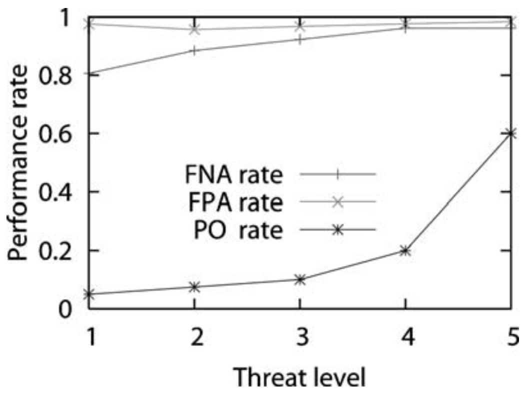

Fig. 11. FNA, FPA, and PO rates for different threat levels.

these allocated lengths of T_{SIG} using the data set explained in Section 5.3. The FNA and FPA rates are obtained directly from the experiment. We define PO rate as the percentage of the worm payload used as signature.

Table II depicts the performance of the optimum signature generated for different threat levels. T_{SIG} is the primary metric that is first ensured. During a high threat level state, the IDS can allocate only a limited time for signature generation. In such state, the IDS tolerates a slight relaxation in FNA rate. As indicated by Table II, the FNA rate is slightly affected as T_{SIG} is lowered. Similarly, the PO rate also becomes larger as the threat level increases. When T_{SIG} is short, there is less time to exclude normal tokens. This decreases the density of worm substrings in the signature. This causes FNA rate to get lower and PO rate to become higher. However, figures in Table II indicate that MAPS monitor is capable of selecting optimum parameter values to fit the prevailing network condition.

5.5. Implementation Issues and Discussion

The main performance requirements of SHEWD are (i) response time and (ii) accuracy of generated signatures. The overall response time of SHEWD is equal to the

Table II. Performance of the worm signature generated during different threat levels.

Threat level	T_G (s)	FNA rate	FPA rate	PO rate
1	105	1.000	1.000	0.013
2	90	1.000	1.000	0.013
3	75	0.998	1.000	0.018
4	60	0.970	1.000	0.048
5	45	0.970	1.000	0.048

total time that elapses since collecting suspicious flows to matching signatures with network traffic at the SUU unit. The time taken by ADU to collect suspicious flows depends directly upon the aggressiveness of the propagating worm. Details on the communication delay between the various entities of SHEWD and the signature updating procedure in SUU are also outside the scope of this paper. So, we evaluate the response of SHEWD by the time taken (i) to sort worms, (ii) to generate worm signatures, and (iii) to match the retrieved signatures with network traffic. Our results show that the worm sorting time at MSM is less than 5 s when the suspicious flow pool size is 50. In Section 5.2, we showed that signature for NetSky, a fairly fast-spreading worm, can be generated within 2 min. In Reference [33], the authors present the signature matching time for NetSky signature with 10 substrings. The results indicate that about 300 worm flows can be detected in 3 s; a satisfactory performance. From above, it is clear that the total response time of SHEWD is about 2 min. Enhancing the signature generation procedure is expected to improve the overall system response. We leave this for future research work.

The lack of a common worm database restricts us from directly comparing the performance of various analysis steps of SHEWD with those of existing worm detectors. The worm detection of SHEWD is a three step procedure—suspicious flow collection at ADU, worm sorting and signature generation at MSM, and signature matching at SUU. While enhancing the suspicious flow collection accuracy at ADU and sorting all worms at MSM are expected to ensure a high performance, SHEWD does not require these to be fulfilled. The main objective here is to sort worms (and worms only) at MSM and generate a highly accurate signature to ensure a high detection rate against the same kind of worms that try to penetrate into the network. Our results, illustrated in Reference [10] and in Section 5.2 demonstrate that the accuracy of these signatures for NetSky and Beagle traces used in our experiments is over 99.5%.

While SHEWD is robust against both known and novel worms, its scope is limited to fast spreading e-mail worms. We next list some of its weaknesses and discuss some possible ways to overcome them.

5.5.1. Detecting worms with constant character variations

The ADU unit of SHEWD uses strings of length L_S as samples and checks if these sample strings are present in other inbound e-mails. If a worm intentionally embeds dummy characters in regular intervals of less than L_S, it would degrade the accuracy of the ADU in detecting novel worms. In such cases, it is necessary to adopt alternative approaches, such as, statistical studies, behavior studies, and worm-dissection.

5.5.2. Detecting slow worms

SHEWD is designed to detect worms with a fast propagation rate. Its effectiveness considerably degrades in case of slow and stealthy worms, such as trojan horses, which hide inside the inner network for a certain period of time before propagating to next target. Although the distributed architecture of SHEWD makes it possible to detect such worms, it would be necessary to increase the string caching time Θ_T and decrease the repetitive occurrence threshold Δ_{TH}. Increasing Θ_T and reducing Δ_{TH} results, in turn, in higher PO with poor accuracy.

5.5.3. Detecting encrypted worms

Encrypted worm payloads have very little in common. Only the parts which carry decryption keys would be meaningful for SHEWD. This greatly limits the number of sample strings N_S ADU can extract from each inbound e-mail. Furthermore, ADU has to rely on the portion of payload with decryption keys. As such portions do not necessarily carry core worm codes, the system is likely to generate a high number of false positives.

5.5.4. Defining normal files

The ADU analysis and the signature generation processes require defining normal flows. However, defining normal flows is a non-trivial task. While too many normal flows consume a large amount of resources and retard the analysis speed, too few normal flows degrade the accuracy. Thus, the administrator needs to decide on the priority of resource and response *versus* accuracy. The false positive e-mails, however, can be very useful in enhancing the performance of the detecting system. By adding these false positive e-mails to the normal-flow database, similar false alarms can be suppressed. Also, the system can have better response with less resource by deleting *old* and *unused* normal flows and signatures from the signature, a method supported by Li *et al.* [11]. This implementation on SHEWD deserves further investigation and defines one direction for our future research work in this particular area of research.

6. Concluding Remarks

In this paper, we proposed a signature-based worm detection system. The overall network topology consists of hierarchically placed LSMs, MSMs, and a GSM. LSMs are constantly updated with worm signatures. Known worms are detected by signature matching. To detect any novel worms included in the traffic, an anomaly based module searches for worm-like or suspicious flows. At metropolitan managers, suspicious flows from corresponding local managers are collected. Cluster analysis is carried out to sort worms from the pool of suspicious flows. The sorted worms are used for signature generation. The worm signature consists of multiple substrings. The signature is relayed to all local managers from the global manager and metropolitan managers. Local managers contain also a module that automatically searches for ideal values of parameters used in the analysis. We evaluated the performance of the proposed system by using real network traffic containing traces of real worms. The experiment results elucidate that the proposed SHWD scheme is capable of detecting worms in a nearly real-time fashion with high detection rates and low false positive rates.

References

1. CERT/CC Statistics 1988-2006. Available at: http://www.cert.org/stats/cert_stats.html
2. Moore D, Paxson V, Savage S, Shannon C, Staniford S, Weaver N. Inside the slammer worm. *IEEE Security and Privacy* 2003; **1**(4): 33–39.
3. Shannon C, Moore D. The spread of the witty worm. *IEEE Security and Privacy* 2004; **2**(4): 46–50.
4. Yaneza JLA, Mantes C, Avena E. The trend of malware today: annual virus round-up and 2005 forecast. Annual Virus Roundup, Trend Micro. 2004.
5. Staniford S, Paxson V, Weaver N. How to Own the Internet in your spare time. In *Proceedings of the 11th USENIX Security Symposium*, San Francisco, CA, USA, August 2002.
6. Nazario J. *Defense and detection strategies against Internet worms*. Artech House, 2004.
7. Weaver N, Paxson V, Staniford S, Cunningham R. A taxonomy of computer worms. In *Proceedings of the First Workshop on Rapid Malcode (WORM 2003)*, Washington DC, USA, October 2003.
8. Simkhada K, Taleb T, Waizumi Y, Jamalipour A, Kato N, Nemoto Y. An efficient signature-based approach for automatic detection of Internet worms over large-scale networks. In *Proceedings of IEEE International Conference on Communications (ICC 2006)*, Istanbul, Turkey, June 2006.
9. Simkhada K, Taleb T, Waizumi Y, Jamalipour A, Hashimoto K, Kato N, Nemoto Y. A multi-level security based autonomic parameter selection approach for an effective and early detection of Internet worms. In *Proceedings of the IEEE Globecom 2006*, San Francisco, CA, USA, November 2006.
10. Simkhada K, Tsunoda H, Waizumi Y, Nemoto Y. Differencing worm flows and normal flows for automatic generation of worm signatures. In *Proceedings of the 7th IEEE International Symposium on Multimedia (ISM2005)*, Irvine, CA, USA, December 2005.
11. Li P, Salour M, Su X. A survey of Internet worm detection and containment. *IEEE Communications Surveys & Tutorials*, 1st Quarter 2008.
12. Dagon D, Qin X, Gu G, Lee W, Grizzard J, Levine J, Owen H. HoneyStat: local worm detection using honeypots. *Recent Advances in Intrusion Detection (RAID)*, Sophia Antipolis, France, September 2004.
13. Kreibich C, Crowcroft J. Honeycomb—creating intrusion detection signatures using honeypots. In *Proceedings of the 2nd Workshop on Hot Topics in Networks (HotNets-II)*, Cambridge, MA, USA, November 2003.
14. Schultz MG, Eskin E, Zadok E. Data mining methods for detection of new malicious executables. *IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2001.
15. Newsome J, Song D. Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. In *Proceedings of the 12th Annual Network and Distributed System Security Symposium*, Alexandria, VA, USA, February 2005.
16. Evans D, Larochelle D. Improving security using extensible lightweight static analysis. *IEEE Software* January/February 2002, pp. 42–51.
17. Cowan C, Beattie S, Johansen J, Wagle P. PointGuard: protecting pointers from buffer overflow vulnerabilities. In *Proceedings of 12th USENIX Security Symposium*, Washington DC, USA, August 2003.
18. Akritidis P, Anagnostakis K, Markatos EP. Efficient content-based detection of zero-day worms. In *Proceedings of IEEE International Conference on Communications (ICC 2005)*, Seoul, Korea, May 2005.
19. Kim H, Karp B. Autograph: toward automated, distributed worm signature detection. In *Proceedings of 13th USENIX Security Symposium*, San Diego, CA, August 2004.
20. Singh S, Estan C, Varghese G, Savage S. The Early-Bird system for real-time detection of unknown worms. *Technical Report CS2003-0761*, University of California, San Diego, August 2003.
21. Newsome J, Karp B, Song D. Polygraph: automatically generating signatures for polymorphic worms. In *IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2005.
22. Williamson MM. Design, implementation and test of an email virus throttle. In *Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC)*, Las Vegas, NV, USA, December 2003.
23. Costa M, Crowcroft J, Castro M, Rowstron A, Zhou L, Zhang L, Barham P. Vigilante: end-to-end containment of Internet worms. In *Proceedings of the 20th ACM Special Interest Group on Data Communications (SIGCOMM'05)*, Philadelphia, PA, USA, August 2005.
24. Chen X, Heidemann J. Detecting early worm propagation through packet matching. *Technical Report ISI-TR-2004-585*, USC/Information Sciences Institute, Marina del Rey, CA, USA, February 2004.
25. Bhattacharya M, Hershkop S, Eskin E. MET: an experimental system for malicious email tracking. In *Proceedings of the 2002 New Security Paradigms Workshop*, Virginia Beach, VA, USA, September 2002.

26. Zou CC, Gong W, Towsley D. Feedback email worm defense system for enterprise networks. *Technical Report TR-04-CSE-05*, University of Massachusetts, April 2004.
27. Chen S, Tang Y. Slowing down Internet worms. In *Proceedings of 24th International Conference on Distributed Computing Systems (ICDCS'04)*, Tokyo, Japan, March 2004.
28. Zou CC, Gong W, Towsley D, Gao L. The monitoring and early detection of Internet worms. *IEEE/ACM Transactions on Networking* 2005; **13**(5): 961–974.
29. Yegneswaran V, Barford P, Jha S. Global intrusion detection in the DOMINO overlay system. In *Proceedings of 11th Annual Network and Distributed System Security Symposium*, San Diego, CA, USA, February 2004.
30. Janakiraman R, Waldvogel M, Zhang Q. Indra: a peer-to-peer approach to network intrusion detection and prevention. In *Proceedings of 2003 IEEE WETICE Workshop on Enterprise Security*, Linz, Austria, June 2003.
31. Fink GA, Chappell BL, Turner TG, O'Donoghue KF. A metrics-based approach to intrusion detection system evaluation for distributed real-time systems. In *Proceedings of 10th International Workshop on Parallel and Distributed Real-Time Systems*, FL, USA, April 2002.
32. Tanachaiwiwat S, Helmy A. VACCINE: war of the worms in wired and wireless networks. In *Proceedings of IEEE INFOCOM 2006*, Barcelona, Spain, April 2006.
33. Suzuki Y, Waizumi Y, Tsunoda H, Nemoto Y. A low cost worm detection technique based on flow payload similarity. In *Proceedings of Web Information Systems and Technologies*, Barcelona, Spain, March 2007.