PAIDS: A Proximity-Assisted Intrusion Detection System for Unidentified Worms

†Zhenyun Zhuang, †Ying Li, ‡Zesheng Chen
† College of Computing, Georgia Institute of Technology, Atlanta, Georgia 30332
‡ Department of Electrical and Computer Engineering, Florida International University, Miami, Florida 33174
†{zhenyun, yingli}@cc.gatech.edu, ‡zchen@fiu.edu

***Abstract*—The wide spread of worms poses serious challenges to today's Internet. Various IDSes (Intrusion Detection Systems) have been proposed to identify or prevent such spread. These IDSes can be largely classified as signature-based or anomaly-based ones depending on what type of knowledge the system knows. Signature-based IDSes are unable to detect the outbreak of new and unidentified worms when the worms' characteristic patterns are unknown. In addition, new worms are often sufficiently intelligent to hide their activities and evade anomaly detection. Moreover, modern worms tend to spread more quickly, and the outbreak period lasts in the order of hours or even minutes. Such characteristics render existing detection mechanisms less effective.**

In this work, we consider the drawbacks of current detection approaches and propose PAIDS, a proximity-assisted IDS approach for identifying the outbreak of unknown worms. PAIDS does not rely on signatures. Instead, it takes advantage of the proximity information of compromised hosts. PAIDS operates on an *orthogonal* dimension with existing IDS approaches and can thus work *collaboratively* with existing IDSes to achieve better performance. We test the effectiveness of PAIDS with trace-driven simulations and show that PAIDS has a high detection rate and a low false positive rate.

***Index Terms*—Proximity; Intrusion Detection System; Worm**

I. Introduction

Self-propagating worms have been posing serious threats to today's Internet [29]. Such malicious programs are capable of propagating quickly and infecting a significant number of vulnerable machines in a short period of time (*e.g.*, a few hours). Victims compromised by worms can form botnets and be used to launch large-scale attacks (*e.g.,* DDoS and spamming).

To effectively prevent worms from propagating rapidly, an IDS (Intrusion Detection System) is needed. Over the last few decades, numerous intrusion detection mechanisms have been proposed. These various techniques can be largely classified into two categories based on the nature of the knowledge that an IDS has. If the characteristic patterns (*i.e.*, signatures) of attacks in an IDS are known, the IDS is referred to as a *signature*-based IDS. Otherwise, if the patterns of normal activities are known, the attacks can be identified by monitoring the differences between the normal and anomalous activities. An IDS relying on identifying anomaly features is referred to as an *anomaly*-based IDS.

Existing techniques can detect and prevent the propagation of many worms, in particular the known ones. Nevertheless, their operations often fail to work with new and more intelligent worms since the characteristics patterns of new worms have not been extracted and analyzed. Due to the unavailability of patterns of the worms, signature-based solutions simply do not work. Briefly, two limitations associated with signature-based IDSes prevent them from functioning effectively. First, it takes considerable time for detection entities (such as anti-virus software companies) to learn the attack pattern or the signature of the worms. Thus, before such signatures are available, the worms may have infected a significant number of machines. Second, it takes some time for a normal user to utilize such signatures in the form of updating or installing IDS software. On the other hand, modern worms tend to spread more quickly, and the outbreak period lasts in the order of hours or even minutes. In particular, flash worms are able to compromise a large number of hosts in several minutes [28]. Therefore, signature-based techniques fail to effectively detect the propagation of unknown worms.

Anomaly-based solutions also have their drawbacks. An anomaly-based IDS examines traffic, activities, or behaviors to find anomalies. The underlying principle is that "attack behaviors" may differ from "normal user behaviors." By cataloging and identifying the differences involved, IDSes can detect a worm in many circumstances. However, anomaly-based IDSes are far from being effective. First, since normal behaviors can change easily and readily, anomaly-based IDSes are prone to false positives, *i.e.,* many actually normal behaviors are falsely reported as attacks. Second, worms can potentially become increasingly intelligent to hide their abnormal activities and thus render most anomaly-based approaches ineffective. For example, if a worm applies a polymorphic blending attack [10], it can hide its activities in normal network activities. Therefore, anomaly-based IDSes may fail to effectively detect the propagation of intelligent worms.

In this work, our goal is to design an IDS that is capable of identifying new and intelligent fast-propagating worms and thwarting their spread, particularly during the worm's "start-up" stage. Since neither signature-based nor anomaly-based techniques can achieve such capabilities, we ask the question: *Given the unknown nature of an impeding worm, its high spreading speed, and its intelligence to hide its abnormal activities, is it still possible to efficiently thwart its propagation?*

We answer this question with a novel proximity-assisted

approach. Our approach is referred to as PAIDS (Proximity-Assisted Intrusion Detection System) and is mainly based on the observations of the clustered pattern of worm propagation and the typical long active time of a compromised host. Since vulnerable hosts are highly unevenly distributed in the Internet [5], compromised computers are usually centralized in certain outbreak areas, especially at the early stage of worm propagation. Thus, a host located in the area with more infected hosts is more likely to be compromised by the worm. The locations of the early infected hosts can be known in certain ways, for instance, using honeypots. In other words, although the nature and signatures of the upcoming worms cannot be identified at the early stage, the exact fact that they are infecting other machines can be detected. Thus, such information can be utilized to counteract the worms' infection process at their early stage when conventional signature-based methods fail to work. One important property about PAIDS is that it is fully *complementary* to existing approaches, particularly anomaly-based ones. That is, the mechanisms included in the proximity-assisted approach can be utilized in tandem with other approaches since PAIDS is working on a different dimension from existing IDSes. Moreover, our preliminary evaluation based on trace-driven simulations shows that PAIDS has a high detection rate and a low false positive rate.

The rest of the paper is organized as follows. We first motivate our design in Section II. We present the detailed design in Section III. We then perform trace-driven simulations in Section IV. Finally, we conclude our work in Section V.

II. Motivation

Our proposed approach, PAIDS, is motivated by three major observations: (i) limitations of existing IDSes; (ii) clustered pattern of worm spread; and (iii) long active time of compromised hosts.

A. Limitations of existing approaches

• **Signature-based or anomaly-based.** Intrusion detection efforts have been proposed for years and can be classified as signature-based and anomaly-based. Signature-based IDSes, such as anti-virus programs (*e.g.,* Norton Antivirous and Macfee), use the signatures to recognize and block infected files, programs, or active Web content from entering a computer system. Such IDSes are the most widely used approach in the commercial IDS technology today. Moreover, abundant research efforts such as [3], [6], [14], [16], [18], [24] also fall into this category.

Anomaly-based IDSes use rules or predefined concepts about "normal" and "abnormal" system activities (*i.e.*, heuristics) to distinguish anomalies from normal system behaviors and to monitor or block anomalies. Anomaly-based IDSes include [1], [7], [13], [17], [27], [33], [34]. For an anomaly-based IDS, a training process is typically required to extract normal patterns. Afterwards, when some activities are observed to be sufficiently different from such patterns, they are flagged as abnormal ones, and the corresponding actions may be taken. A wide variety of techniques have been explored to approach the anomaly detection problem, such as neural networks [13], statistical modeling [27], temporal sequence learning [17], n-grams [34], and states of web applications [7].

However, neither signature-based nor anomaly-based IDSes can effectively identify new and intelligent worms. On one hand, identifying a worm using signature-based IDSes requires the characteristic patterns of the worm. Such mechanisms will not work with new worms since the patterns of these worms are not known *a priori*. Furthermore, even if the signature of a new worm is extracted after a certain period of time, it takes considerable time for a normal user to adopt the signature in the form of updating his anti-worm software. On the other hand, modern worms are becoming increasingly intelligent, and many are capable of hiding their activities to avoid the detection of anomaly-based IDSes [10]. Worms achieve such stealthy goals either by learning the rules used by the IDS or by acting extremely less aggressively.

• **Network-based or host-based.** Various IDSes can also be classified into network-based [9], [11], [19], [22], [26], [31], [32] and host-based ones [8], [12], [20], [25], [30]. The difference between these two categories lies in where the intelligence used for detection resides. In host-based IDSes, the intelligence only resides on local hosts, whereas network-based IDSes rely on the information exchanged among many hosts. Both categories have their own advantages. For example, network-based IDSes can monitor an entire network with only a few well-situated nodes and impose little overhead on a network. Host-based IDSes can analyze activities on the host at a high level of details and determine which processes and/or users are involved in malicious activities.

Both types of approaches, however, have disadvantages and may fail to work under certain scenarios. For example, although network-based IDSes have the advantage of knowing aggregate traffic patterns, they may not know other crucial information such as the process environments and the exact protocol of a connection. Most of the time normal users are reluctant to give such information to other entities including IDSes due to privacy or security concerns. By contrast, host-based IDSes have the full information of the connections of a specific host, but they lack the aggregate view of the network. The disadvantages associated with either IDSes may greatly affect the effectiveness of detecting worms. Our designed PAIDS intends to incorporate the advantages of both IDSes.

B. Clustered pattern of worm spread

Once a worm is released to the Internet, it typically starts from a few hosts. It then guesses the addresses of targets and attempts to infect them. Typically, worms use random scanning or localized scanning to spread [29], [35]. Random scanning selects targets randomly and has been used by Code Red v2, Slammer, and Witty worms, whereas localized scanning preferentially searches for targets in the local network (*i.e.,* sharing the same prefix of IP addresses) and has been exploited by Code Red II and Nimda worms. Localized scanning has some advantages over random scanning. For example, hosts

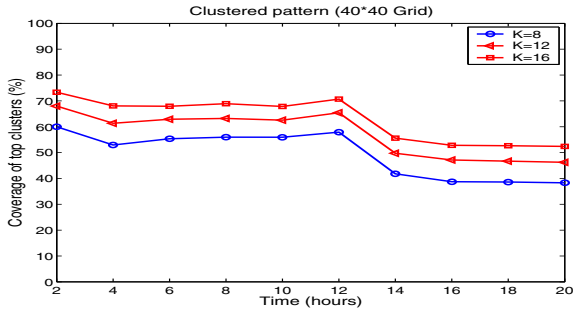


Fig. 1. Top K grids (out of 1600 grids) of U.S. Map

within the same subnet often expose similar vulnerabilities. In addition, in general infecting close-by hosts takes less time due to the smaller round trip time. Furthermore, many networks are protected by firewalls, and infecting hosts in the same network is relatively easier.

Worm spread demonstrates a highly clustered pattern, especially for localized scanning. The main reason is that vulnerable hosts are highly unevenly distributed in the Internet [4]. Therefore, at any time of the worm outbreak, the compromised hosts typically form clusters, *e.g.,* they reside in close-by geographical locations. To show this, we study the outbreak of the Code Red v2 worm in July 2001 based on the traces from CAIDA [2]. To examine the clustered pattern of geographical locations of infected hosts, we equally divide the US continent map into a number of grids and count the number of compromised hosts covered by each grid. We then choose the top K grids and show the percentage of compromised hosts covered by these grids. Intuitively, if the cluster degree is higher, more hosts would reside in the top K grids. Specifically, we divide the map into 40×40 grids (totally 1600) and measure the percentage of compromised hosts covered by the top 8, 12, and 16 grids (*i.e.*, $K = 8$, 12, 16). Figure 1 shows our measurements of the coverage of top K grids over time of up to 20 hours. We observe the highly clustered patterns. For example, as shown in the figure, with only 8 grids (0.5% of total grids), the coverage at the 2-hour time is about 60%, whereas 16 grids (1% of all grids) covers more than 72% of all compromised hosts. More interestingly, such clustered behaviors do not fade quickly over time. As shown in the figure, even after 20 hours, top 16 grids still account for more than 53% of all compromised hosts.

Interestingly, the clustering behavior of worm spreading occurs not only in the geographical dimension, but in the network address dimension. Works such as [5] have identified the clustered pattern in the network address space. Specifically, work [5] finds that more than 80% of malicious sources are clustered in the same 20% of the IPv4 address space over time based on DShield data. Our designed PAIDS can work on both geographical and network address dimensions. In this paper, we mainly present the results on the geographical dimension.

C. Long active time of compromised hosts

Another observation is that compromised hosts are typically engaged in infecting other hosts for a considerable amount of time. Taking the Code Red v2 worm as an example, we plot the distribution of active time of all compromised hosts (more than 359,000) in Figure 2. It can be seen that 80% of compromised hosts have active time of longer than 30 minutes, while only 12% of these hosts have active time shorter than 5 minutes. In other words, the figure shows that most compromised hosts attempt to infect other hosts for a considerably long time (*e.g.*, more than dozens of minutes) before they are stopped. This characteristic of compromised hosts can potentially be utilized to assist detection. For example, if a compromised host can be identified as a suspicious one, then hosts that are communicating with this host should be alerted.

III. Design

We now present the design of PAIDS. After giving an overview, we will present the deployment model, the software architecture, and three major components of PAIDS.

A. Overview

The key idea of PAIDS is to take advantage of the clustered pattern of worm outbreak and the long active time of compromised hosts. Briefly, PAIDS works as follows. PAIDS runs on a local host as a daemon and keeps monitoring the ongoing connections. It records the IP addresses and port numbers of the hosts with which the local host is communicating. The recorded information is sent to a processing center that determines the danger level of the corresponding hosts. If the processing center determines that a recorded host is suspicious, then it reports back to the local host so that PAIDS can raise alerts and notify the user. The processing center makes the decision by collecting the information of suspicious hosts on the Internet and performing clustering operations to model the danger levels of each area. The clustering is based on proximity information associated with the suspicious hosts. Note that proximity is not necessarily limited to geographical and network address dimensions and can also occur in other dimensions such as DNS. In this work, we present the design of PAIDS only in the context of the geographical dimension for the purpose of simplicity. However, PAIDS can be easily changed to utilize other dimensions.

The operations of PAIDS involve the cooperation of several entities (shown in Figure 3). *Firstly*, it requires the local host

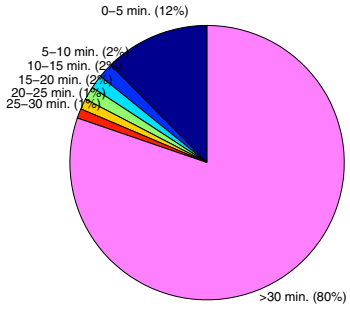


Fig. 2. Active time distribution of the Code Red v2 worm

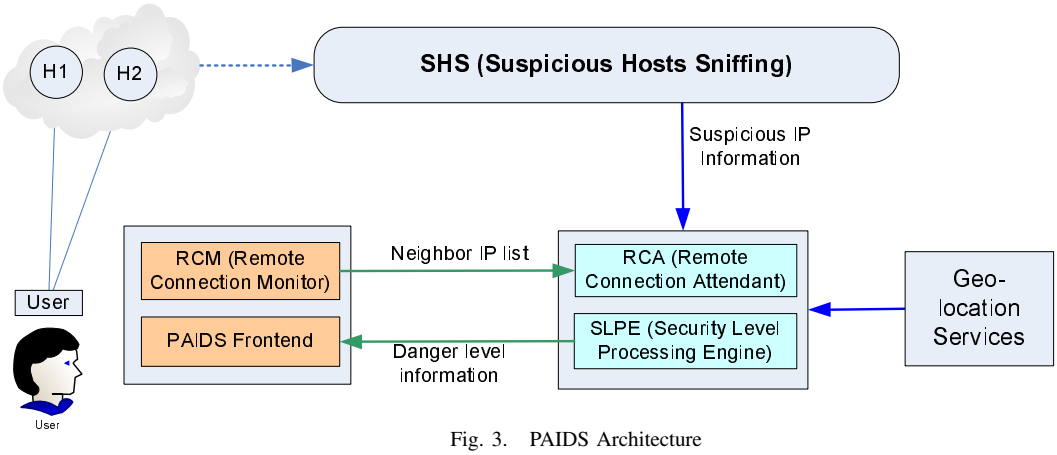


Fig. 3. PAIDS Architecture

to report its communicating hosts (referred to as "communication neighbors") to a processing center. The communication components of this reporting service are referred to as Remote Connection Monitor (RCM) and Remote Connection Attendant (RCA). *Secondly*, the processing center is referred to as Security-Level Processing Engine (SLPE). SLPE is the entity that determines the danger level of each of the reported neighbors from the local host. The determination is based on proximity correlations, *i.e.*, the closer the communicating host is to the outbreak area, the greater chance it might conduct malicious activities. The proximity correlations can be computed using various geographic mapping techniques [21]. One of such techniques is the IP-to-location service [15]. The SLPE can be implemented at a place convenient to the local hosts, *e.g.*, a dedicated machine of a group or an institute. *Thirdly*, it requires a suspicious-activity monitoring service that typically runs honeynet to attract worms and collects related information. In this paper we refer to the monitoring service as Suspicious Hosts Sniffing (SHS). SHS serves no other purpose than simply recording suspicious connections reaching the honeynet. The recorded information at least includes the IP addresses and the port numbers of the suspicious connections. Such information will expose the outbreak information of new worms for they have to propagate to other hosts after they are released.

PAIDS addresses the shortcomings of signature-based and anomaly-based IDSes with a mechanism based on proximity information. It deals with the unavailability of worm signatures by using the notion of suspicious hosts, *i.e.*, the hosts that are captured by honeynet when they attempt to connect to honeynet. Specifically, if a host is compromised, it will attempt to infect other hosts. If it falls into the view of the monitoring service, it is recorded. Since compromised hosts typically keep active for a considerably long time, such a "black-list" style treatment will bring benefits. Moreover, since PAIDS is not based on the known patterns of normal behaviors, it does not have the drawbacks of the anomaly-based approach.

PAIDS addresses the drawbacks of network-based and host-based approaches with a processing center, which is actually the third layer between a global information center and a local host. Introducing such a layer can gain benefits by striking the balance between the advantages and disadvantages associated with these two types of IDSes. Since hosts are more likely to trust a local processing center and present detailed information to the latter, such undertaking can bridge the trust gap between these two types of IDSes. In other words, the processing center can serve as the intermediate proxy for the information exchanges between the global information center and local hosts. The functioning of the processing center works in two directions. On one hand, it collects the information from local hosts and reports filtered information to the global center. On the other hand, it obtains information from the global center and determines the danger level of reported connections on behalf of local hosts. Other than the trust benefit, introducing the processing center can also help relieve the scalability concern embedded in alternative two-layer approaches.

B. Deployment model of PAIDS

The design of PAIDS is not intended to replace any other type of IDSes. Thus, it is inappropriate to compare PAIDS directly with other IDSes. Instead, it *complements* other IDSes by working orthogonally with them. For instance, PAIDS can be combined with an anomaly-based IDS. One way for such a combination is to let PAIDS adjust the threshold values usually used in anomaly-based IDSes. An anomaly-based IDS typically sets certain "threshold" values to determine whether a connection is suspicious or not. These values are difficult to determine in many cases for the determination has to strike a balance between false positives and false negatives. With PAIDS, the threshold values can be adjusted according to the danger levels measured by PAIDS. For example, if a connection involves a host that is within an outbreak area and thus has a higher danger level, the threshold values can be adjusted to reflect such information. Such treatments are possible since PAIDS utilizes the proximity information that other IDSes typically do not consider.

We now use one example scenario to elaborate how PAIDS can work cooperatively with other existing IDSes. We choose an anomaly-based IDS simply because of easy explanation. Note that such a scenario should not be treated as the only way in which PAIDS can work with other IDSes. The anomaly-based IDS is Swaddler [7]. Swaddler supports two modes:

training and detection modes. During the training mode, suitable thresholds are derived to represent the anomalous scores. The system then switches to the detection mode, and anomalous states can be reported based on the derived thresholds. PAIDS can assist the setting of the threshold values by replacing each threshold by a pair of threshold values, one for *safe* neighbors and the other for *suspicious* neighbors. The suspicious neighbors are the hosts causing PAIDS to raise alerts, whereas the safe neighbors are those not triggering alerts. Intuitively, the threshold value for safe neighbors should be higher than the value for suspicious neighbors since the suspicious ones require stricter screening. Alternatively, PAIDS may help the threshold setting in a finer way by assigning a range of values. In other words, PAIDS may define a threshold value function $Th = g(x)$, where x is the danger level outputted by PAIDS. With such a function, the threshold value is not fixed across all hosts, it is adjusted according to the danger level of the corresponding host.

C. Software Architecture and Components

The software architecture of PAIDS is shown in Figure 3. PAIDS mainly consists of two main entities: a PAIDS client and a PAIDS server. It also assumes the existence of a third entity, namely the SHS (Suspicious Hosts Sniffing). The PAIDS client runs two software components: a RCM (Remote Connection Monitor) and a Danger Level Frontend. The first component, RCM, keeps track of on-going connections on the local host and reports the connection list to the PAIDS server. The second component can be as simple as a typical browser such as Internet Explorer or Firefox, but may also include more functionalities such as providing the user ways to disconnect certain connections.

The PAIDS server consists of two parts: a RCA (Remote Connection Attendant) and a SLPE (Security-Level Processing Engine). The purpose of RCA is simply to receive the reported connection list from RCM. SLPE is the core part of PAIDS. It collects information from both RCA and SHS, determines the danger level of each reported connection using geographic mapping techniques, and then sends the relevant information back to the PAIDS client.

We now elaborate on the three major components of PAIDS:

• **RCM and RCA.** RCM and RCA are used to report the local host's communicating neighbors to the SLPE. Specifically, RCM runs on the local host side, whereas RCA runs on the server side. The basic functionalities of RCM are to monitor and report. RCM keeps track of all the ongoing connections of the local host and is capable of reporting to RCA periodically (*i.e.*, pushing) or in the on-demand fashion (*i.e.*, pulling). The major advantage of pushing information to RCA is the timeliness. However, it incurs higher communication and processing overhead. In contrast, pulling information from RCM requires less overhead, but it inflates the information gathering time. RCA collects connection information from RCM. Under the pushing model, RCA is simply a server daemon collecting the reports of RCM. Under the pulling model, RCA pro-actively requests the information from RCM.

• **Security Level Processing Engine (SLPE).** The SLPE is the key component of PAIDS, determining the danger level of the reported neighbors using the IP-to-location service. The determination is based on the distance between a neighbor and the outbreak areas. Formally, given a set of outbreak areas $S = \{S_1, S_2, ..., S_I\}$ and a list of neighbor hosts $H = \{H_1, H_2, ..., H_J\}$, SLPE will output a danger level vector of $L = \{L_1, L_2, ..., L_J\}$, where L_i is the danger level of neighbor host H_i. Specifically, let $d_{i,j}$ denote the distance between H_i and S_j. The danger level of H_i is given by a function $f(d_{i,j}, S)$. The design of $f(d_{i,j}, S)$ by itself is an important and interesting problem. Due to space limitations, we only provide two simple forms. The first form is the average value of all $d_{i,j}$ on S_j:

$$L_i = \frac{1}{I}\sum_{j=1}^{I} d_{i,j}. \quad (1)$$

The second form simply chooses the minimum value of all $d_{i,j}$, *i.e.,* $L_i = \min_j \{d_{i,j}\}$. Both forms have advantages and disadvantages. Without explicit notes, in the following we only use the first form. If SHS also reports the intensities of each outbreak areas, the above equation is given as

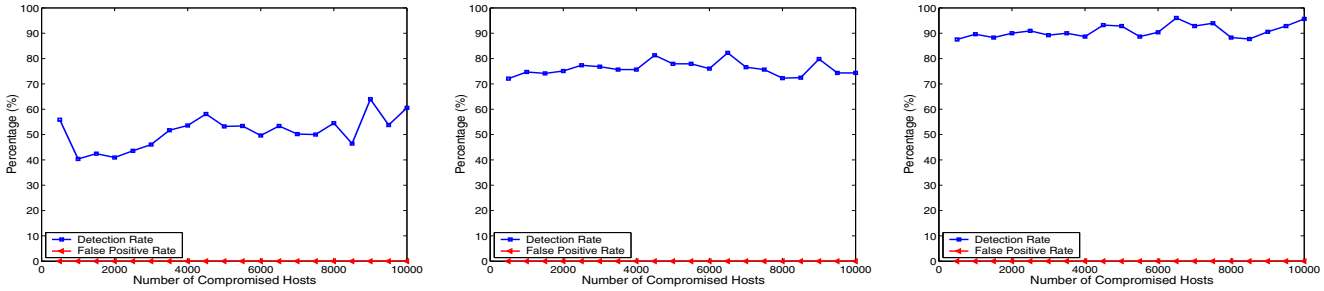
$$L_i = \frac{1}{I}\sum_{j=1}^{I} d_{i,j}t_j, \quad (2)$$

where t_j is the intensity of S_j.

After computing the danger level of a neighboring host, SLPE compares its danger level value to a pre-defined threshold value T_d. If the danger level value is below the threshold, which implies that the host is sufficiently close to the outbreak area, the host will be flagged as suspicious.

One important design issue of SLPE is the scalability. As the number of compromised hosts increases, the processing overhead on SLPE also increases. Thus, the SLPE should appropriately limit the number of outbreak areas when the scalability is a concern. There are many ways to achieve this, and one popular way is to use K-means. With K-means, the number of outbreak areas is fixed to K irrespective of the number of compromised hosts.

• **Suspicious Hosts Sniffing (SHS).** SHS is responsible for collecting the activities of suspicious hosts and reporting to SLPE. The implementation of SHS can be as simple as an individual honeypot or as a more complicated form such as honeynet [23]. Since typically SHS does not have legitimate hosts, SHS records suspicious hosts whenever it receives connection requests from other hosts. The information that SHS records can be simply a list of IP addresses or both IP addresses and port numbers. To take full advantage of the suspicious activities, SHS may also record the intensity of the threat by counting the occurrence frequencies of specific connections. Since there might be a significant number of suspicious activities, SHS may aggregate individual hosts into suspicious outbreak areas. In other words, each outbreak area may contain many hosts and can be represented in the format of $< C, r, t >$, namely, the center C, the radium r, and the outbreak intensity t.



(a) Threshold = 10 (b) Threshold = 15 (c) Threshold = 20

Fig. 4. Effectiveness of PAIDS (Sniffing Latency = 100 seconds)

IV. Evaluation

A. Trace-Driven Simulations

We evaluate the effectiveness of PAIDS by performing trace-driven simulations. The simulation uses the outbreak trace of the Code Red v2 worm from CAIDA [2]. We focus on two performance metrics: detection rate and false positive rate.

The simulation is performed as follows. First, for all the recorded compromised hosts, we sort them according to their starting time, *i.e.*, the time they are observed to begin to infect other hosts. The method is illustrated in Figure 5, where all hosts are ranked along the time line. For each of the hosts, we assume that their infectious activities can be sniffed by SHS and their information are sent to SLPE after a certain latency. This latency is referred to as the Sniffing Latency, denoted by SL. Let C_i denote the set of all compromised hosts when host H_i is compromised, and we have $C_i = \{H_j, j \leq i\}$. Let H_m denote the last host that is at least SL time ahead of the infection of H_i. We use R_i to denote the set of all recorded hosts before host H_i, *i.e.*, $R_i = \{H_j, j \leq m\}$. We assume that honeynet can provide R_i immediately, but cannot provide the signatures and the patterns in a short time. With these notations, C_i contains all hosts that are *actually compromised* when H_i is compromised, whereas R_i contains all hosts in C_i that are *sniffed* or recorded by SHS. In other words, R_i is only a subset of C_i.

Since the trace from CAIDA does not provide the information of "who infects whom", we make several assumptions to evaluate the performance of PAIDS. For a host H_v, we assume that PAIDS is running on this host and reporting suspicious hosts with which the host is communicating. Moreover, we assume that H_v is communicating with H_i. We then measure the effectiveness of PAIDS by monitoring how well PAIDS can detect the infected host H_i. That is, if PAIDS was running on host H_v when the host H_i was infected and begin infecting H_v, then PAIDS running on H_v may be able to determine the danger level of H_i based on the information of R_i. If PAIDS is able to raise the alert, then it is considered to be effective. Otherwise, it is considered to be ineffective. Since R_i is only a subset of C_i (*i.e.*, $R_i \subset C_i$), PAIDS running on H_v may or may not be able to raise a alert depending on the current threshold value. Specifically, we compute the averaged distance of H_i to R_i, *i.e.*,

$$L_i = \frac{1}{|R_i|} \sum_j d_{H_i,H_j}, \text{ where } H_j \in R_i.$$

Note that such a scenario provides a worst case for PAIDS since we consider the time slot right after host H_i is infected when the size of SHS set R_i is smallest.

After L_i is obtained, it is then compared to T_d, a pre-configured threshold value. If L_i is smaller than or equal to T_d, then host H_i is considered to be in danger, and the corresponding effectiveness value is set to 1. Otherwise, if L_i is larger, then the effectiveness value is set to 0, as illustrated below: $Alert_i = 1$, if $L_i \leq T_d$; $Alert_i = 0$, otherwise.

We measure the effectiveness of PAIDS for the first 10,000 infected hosts. For every 500 hosts, we output a detection rate that is calculated based on the number of alerts raised. Specifically, for each of the 500 hosts, if PAIDS can raise the alert, we count 1; otherwise, 0. Then we obtain the detection rate by dividing the sum of alerts by 500. Hence, denoting the number of alerts as $Alert$, the effectiveness of PAIDS is calculated as $\frac{Alert}{500}$.

The false positive rate is obtained by considering a randomized scenario. Specifically, we generate a scenario by randomly

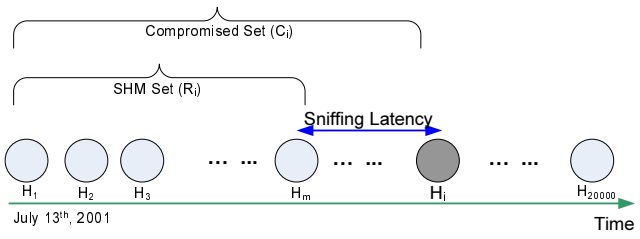


Fig. 5. Illustration of Trace-driven Evaluation

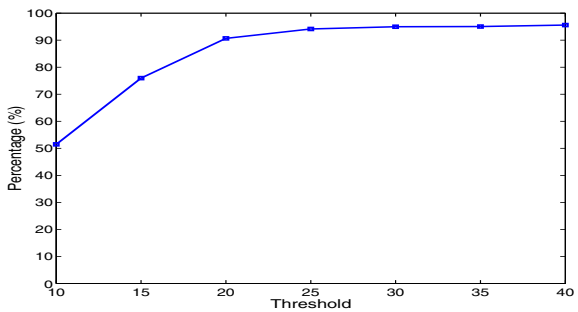
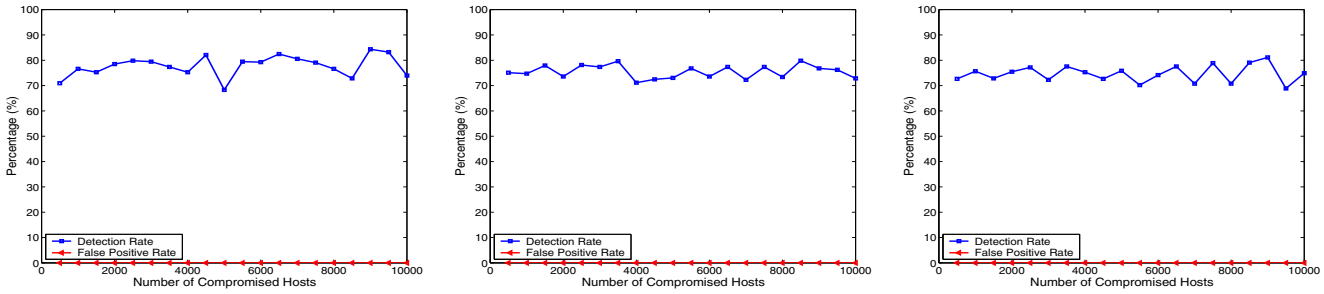


Fig. 6. Impact of Thresholds



(a) Sniffing Latency = 25 seconds (b) Sniffing Latency = 50 seconds (c) Sniffing Latency = 150 seconds

Fig. 7. Effectiveness of PAIDS (Threshold = 15)

choosing a host that is not compromised and is communicating with host H_v. That is, we introduce a host with a random IP address and use this host to replace the host H_i in each step. As a result, the randomly selected neighbor represents the false positive. Using the same method as described in calculating the detection rate, we also obtain the false positive rate.

B. Results

• **Impact of T_d.** We study the impact of T_d by varying its values. We set the $SL = 100$ and test three cases: $T_d = 10$, 15, and 20. We consider totally 10,000 hosts, and Figure 4 shows the results. In the figure, we observe two trends. Firstly, for randomly introduced hosts, only very few false positives are raised, whereas for compromised hosts, PAIDS can raise a very high percentage of alerts. For example, with $T_d = 15$, about 75% of compromised hosts are detected. Secondly, the effectiveness increases with higher values of T_d. These results are expected since higher T_d means that more neighbor hosts are treated as suspicious, thus reducing the false negatives. To more closely study the trend, we vary T_d from 10 to 40 and show the results in Figure 6. We can see that $T_d = 10$ gives the detection rate only 51%, and the rate quickly reaches 91% when $T_d = 20$. When $T_d = 40$, it can detect up to 96% compromised hosts. The false positive rates for all results shown in the figure are almost zero.

• **Impact of SL.** We also vary the Sniffing Latency value SL from 25 seconds to 150 seconds and evaluate the impact. The results are shown in Figure 7. Note that in these experiments, the threshold is 15 (*i.e.*, $T_d = 15$). We observe that as SL increases, the detection rate of PAIDS decreases. For instance, with $SL = 25$, the rate is about 79%, while with $SL = 150$, the detection rate drops to about 74%. This is because a larger SL value means that it takes a longer time for SHS to learn the information about compromised hosts. We further test the impact of SL with varying values and show the results in Figure 8. We can see that as SL increases from 25 to 200, the effectiveness only drops from about 79% to 74%. These results imply that the effectiveness of PAIDS is relatively insensitive to the changes of SL, which is actually an advantage of PAIDS.

• **Impact of $f(d_{i,j}, S)$.** As we elaborated in Section III, the choice of $f(d_{i,j}, S)$, the danger level function for a host H_i, also needs the careful design. We have proposed two forms of this function. In all above evaluations we use the *average* form, which calculates the danger level by averaging on all $d_{i,j}$. We also preliminarily evaluate the impact of the other form, which takes the minimum value of all $d_{i,j}$.

Since the second form of $f(d_{i,j}, S)$ takes the *minimum* value of all $d_{i,j}$, the value of $f(d_{i,j}, S)$ is much smaller than the first form. We choose three threshold values for T_d, namely, 0.01, 0.05, and 0.1. The results are shown in Figure 9. We see that as T_d increases, the detection rates also gradually increase, and with $T_d = 0.1$ the rate almost reaches 100%. The false positive rates are kept very low. Specifically, our results indicate that the highest false rate is only 1.3%. The interesting observation regarding the false positive rate is that the rate increases as more hosts are compromised. Part of the reason for such a result is that with increasing number of hosts, the randomly selected hosts have a higher probability of being close to outbreak areas.

It would be interesting to study the question of applying which form of $f(d_{i,j}, S)$ under what conditions. Although

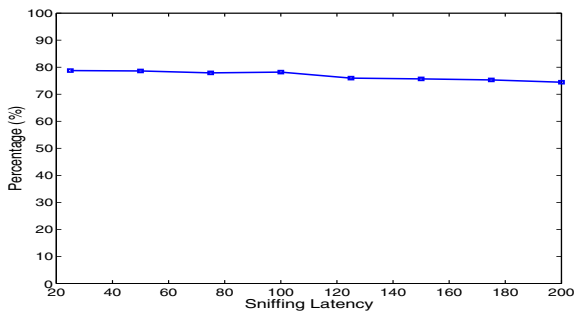


Fig. 8. Impact of Sniffing Latency

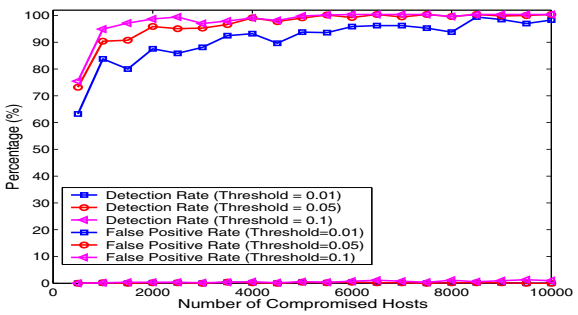


Fig. 9. Impact of Danger Level Function

the page limit does not allow us to present the details, in short, our results suggest that the average form has the better false positive rate (all are zero), but with the relatively high detection rate; whereas the minimum form has the non-zero false positive rate (though very small), but with the higher detection rate.

V. Conclusions

In this work, we have studied the problem of intrusion detection and focused on the detection of *unidentified* worms. We have observed the failures and the drawbacks of existing IDSes and approached the problem in a novel way. Our design is based on the typical behaviors of worm outbreaks. By utilizing the proximity and activity information of worm outbreaks, we have proposed a detection system, called PAIDS, that is *complementary* to existing IDSes. Our trace-driven simulation show that PAIDS can detect an unidentified worm with a high detection rate and a low false positive rate.

VI. Acknowledgements

We thank Dr. Jonathon Giffin of Georgia Tech and the anonymous reviewers for their insightful comments that helped us improve the quality of this research.

References

[1] S. Bhatkar, A. Chaturvedi, and R. Sekar. Dataflow anomaly detection. In *SP '06: Proceedings of the 2006 IEEE Symposium on Security and Privacy*, pages 48–62, Washington, DC, USA, 2006. IEEE Computer Society.

[2] CAIDA. The cooperative association for internet data analysis. In *http://www.caida.org/home/*.

[3] R. Cathey, L. Ma, N. Goharian, and D. Grossman. Misuse detection for information retrieval systems. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, 2003.

[4] Z. Chen and C. Ji. Measuring network-aware worm spreading ability. In *Proceedings of IEEE INFOCOM*, pages 116–124, Anchorage, AK, USA, 2007.

[5] Z. Chen, C. Ji, and B. Paul. Spatial-temporal characteristics of internet malicious sources. In *Proceedings of IEEE INFOCOM (Mini-Conference)*, Phoenix, AZ, USA, 2008.

[6] C. Y. Chung, M. Gertz, and K. Levitt. Demids: A misuse detection system for database systems. In *In Third International IFIP TC-11 WG11.5 Working Conference on Integrity and Internal Control in Information Systems*, pages 159–178. Kluwer Academic Publishers, 1999.

[7] M. Cova, D. Balzarotti, V. Felmetsger, and G. Vigna. Swaddler: An approach for the anomaly-based detection of state violations in web applications. In *Proceedings of the 10th International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 63–86, Queensland, Australia, September 5–7, 2007.

[8] H. Debar, M. Dacier, and A. Wespi. A revised taxonomy for intrusion detection systems. Technical report, IBM Research, October 1999.

[9] H. Dreger, A. Feldmann, M. Mai, V. Paxson, and R. Sommer. Dynamic application-layer protocol analysis for network intrusion detection. In *USENIX-SS'06*, Berkeley, CA, USA, 2006. USENIX Association.

[10] P. Fogla and W. Lee. Evading network anomaly detection systems: formal reasoning and practical techniques. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 59–68, New York, NY, USA, 2006. ACM.

[11] J. M. González and V. Paxson. Enhancing network intrusion detection with integrated sampling and filtering. In *Proceedings of RAID*, pages 272–289, 2006.

[12] R. Gopalakrishna, E. H. Spafford, and J. Vitek. Efficient intrusion detection using automaton inlining. In *SP '05: Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 18–31, Washington, DC, USA, 2005. IEEE Computer Society.

[13] A. K. Gosh, J. Wanken, and F. Charron. Detecting anomalous and unknown intrusions against programs. In *ACSAC '98: Proceedings of the 14th Annual Computer Security Applications Conference*, page 259, Washington, DC, USA, 1998. IEEE Computer Society.

[14] H.Bos and K. Huang. Towards software-based signature detection for intrusion prevention on the network card. In *Proc. of Eighth International Symposium on Recent Advances in Intrusion Detection (RAID2005)*, Seattle, WA, USA, 2005. ACM.

[15] IP2Location. Geolocation ip address to country city region latitude longitude. In *http://www.ip2location.com/*.

[16] C. Krügel and T. Toth. Using decision trees to improve signature-based intrusion detection. In *Proceedings of RAID*, pages 173–191, 2003.

[17] T. Lane and C. E. Brodley. Temporal sequence learning and data reduction for anomaly detection. *ACM Trans. Inf. Syst. Secur.*, 2(3):295–331, 1999.

[18] U. Lindqvist and P. A. Porras. Detecting computer and network misuse through the production-based expert system toolset (p-BEST). In *IEEE Symposium on Security and Privacy*, pages 146–161, 1999.

[19] C. Muelder, K.-L. Ma, and T. Bartoletti. Interactive visualization for network and port scan detection. In *Proceedings of RAID*, pages 265–283, 2005.

[20] D. Mutz, W. K. Robertson, G. Vigna, and R. A. Kemmerer. Exploiting execution context for the detection of anomalous system calls. In *Proceedings of RAID*, pages 1–20, 2007.

[21] V. N. Padmanabhan and L. Subramanian. An investigation of geographic mapping techniques for internet hosts. In *SIGCOMM '01*, pages 173–185, New York, NY, USA, 2001. ACM.

[22] M. Polychronakis, K. G. Anagnostakis, and E. P. Markatos. Emulation-based detection of non-self-contained polymorphic shellcode. In *Proceedings of RAID*, volume 4637, pages 87–106. Springer, 2007.

[23] M. A. Rajab, F. Monrose, and A. Terzis. On the effectiveness of distributed worm monitoring. In *SSYM'05: Proceedings of the 14th conference on USENIX Security Symposium*, pages 15–15, Baltimore, MD, USA, 2005. USENIX Association.

[24] S. Rubin, S. Jha, and B. P. Miller. Language-based generation and evaluation of nids signatures. In *SP '05: Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 3–17, Washington, DC, USA, 2005. IEEE Computer Society.

[25] M. I. Sharif, K. Singh, J. T. Giffin, and W. Lee. Understanding precision in host based intrusion detection. In *Proceedings of RAID*, pages 21–41, 2007.

[26] S. Sinha, F. Jahanian, and J. M. Patel. Wind: Workload-aware intrusion detection. In *Proceedings of RAID*, pages 290–310, 2006.

[27] A. Soule, K. Salamatian, and N. Taft. Combining filtering and statistical methods for anomaly detection. In *IMC'05: Proceedings of the Internet Measurement Conference 2005 on Internet Measurement Conference*, pages 31–31, Berkeley, CA, USA, 2005. USENIX Association.

[28] S. Staniford, D. Moore, V. Paxson, and N. Weaver. The top speed of flash worms. In *WORM '04: Proceedings of the 2004 ACM workshop on Rapid malcode*, pages 33–42, New York, NY, USA, 2004. ACM.

[29] S. Staniford, V. Paxson, and N. Weaver. How to 0wn the internet in your spare time. In *Proceedings of the 11th USENIX Security*, San Francisco, CA, USA, 2002.

[30] Sufatrio and R. H. C. Yap. Improving host-based ids with argument abstraction to prevent mimicry attacks. In *Proceedings of RAID*, pages 146–164, 2005.

[31] M. Vallentin, R. Sommer, J. Lee, C. Leres, V. Paxson, and B. Tierney. The nids cluster: Scalable, stateful network intrusion detection on commodity hardware. In *Proceedings of RAID*, pages 107–126, 2007.

[32] G. Vasiliadis, S. Antonatos, M. Polychronakis, E. P. Markatos, and S. Ioannidis. Gnort: High performance network intrusion detection using graphics processors. In R. Lippmann, E. Kirda, and A. Trachtenberg, editors, *Proceedings of RAID*, volume 5230 of *Lecture Notes in Computer Science*, pages 116–134. Springer, 2008.

[33] K. Wang, G. F. Cretu, and S. J. Stolfo. Anomalous payload-based worm detection and signature generation. In *Proceedings of RAID*, pages 227–246, Seattle, Washington, USA, 2005.

[34] K. Wang, J. J. Parekh, and S. J. Stolfo. Anagram: A content anomaly detector resistant to mimicry attack. In *In Proceedings of the 9 th International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 226–248, Hamburg, Germany, 2006.

[35] C. C. Zou, D. Towsley, and W. Gong. On the performance of internet worm scanning strategies. *Perform. Eval.*, 63(7):700–723, 2006.