On the Performance of SWORD in Detecting Zero-Day-Worm-Infected Hosts

Shad Stafford
University of Oregon
staffors@cs.uoregon.edu

Jun Li
University of Oregon
lijun@cs.uoregon.edu

Toby Ehrenkranz
University of Oregon
tehrenkr@cs.uoregon.edu

Keywords: Internet worms, worm detection, network security, host infection

Abstract

Once a host is infected by an Internet worm, prompt action must be taken before that host does more harm to its local network and the rest of the Internet. It is therefore critical to quickly detect that a worm has infected a host. In this paper, we enhance our SWORD system to allow for the detection of infected hosts and evaluate its performance. This enhanced version of SWORD inherits the advantages of the original SWORD—it does not rely on inspecting traffic payloads to search for worm byte patterns or setting up a honeypot to lure worm traffic. Furthermore, while acting as a host-level detection system, it runs at a network's gateway and stays transparent to individual hosts. We show that our enhanced SWORD system is able to quickly and accurately detect if a host is infected by a zero-day worm. Furthermore, the detection is shown to be effective against worms of different types and speeds, including polymorphic worms.

1 INTRODUCTION

The launching of a worm can have disastrous effects on millions of computers on the Internet in just a few short minutes [1, 2], potentially disrupting the operation of critical services such as emergency call centers [3]. The cost of disrupted service and repair from worms can also be extremely high; for example, it is estimated that the costs associated with the CodeRed and Sapphire/Slammer worms are over three billion dollars. Perhaps even more alarming, researchers have found that none of the worm attacks so far have come close to causing the amount of damage they are capable of [4].

With worms and their destructiveness gaining wide-spread recognition, it seems only a matter of time until new worms are created with even higher rates of spread. Worms that take advantage of a heretofore unknown vulnerability, so-called *zero-day* worms, are particularly dangerous because many existing security techniques require prior knowledge of the exploit in order to detect and defeat it. These worms could become even more dangerous by creating their own coordinated networks from the infected hosts [5]. In order to achieve effective containment, as shown in [6], reaction times must be on the order of a few minutes or less. Any worm defense mechanism which requires human action, therefore, is just not feasible.

Our recent research has devised an approach, named *SWORD* [7], to detect the occurrence of zero-day worms at an administrative domain level. SWORD has a number of desirable features in a worm-detection system: it requires deployment at only one place on the network (the gateway), it is capable of detecting many different types of worms, it does not rely on payload inspection, and it has a low false-positive rate. These features compare favorably with existing systems such as [8, 9, 10]. Solutions that depend on using worm signatures to identify the byte patterns sent from worm infected hosts may have difficulty detecting polymorphic worms or worms which encrypt their payload during propagation, not to mention the cost of inspecting the payload of every packet in transit. Honeypot based solutions typically capture connections to unused network addresses so will not detect a worm that only attempts to contact network addresses where a valid host exists. They are also vulnerable to spoofing attacks which deliberately send legitimate looking traffic to honeypots causing real traffic to be flagged as worm traffic.

However, it is still unknown how fast and accurate we can be in detecting the infection of individual hosts. While SWORD provides an elegant solution for detecting the occurrence of zero-day worms, it is often necessary to know which hosts are infected in order to take prompt, concrete actions. In this paper, we enhance SWORD to further detect worm infections at the host level without requiring intrusive monitoring systems installed on end-hosts. Furthermore, we will outline a new methodology for evaluating the performance of a worm detection system and then we will present a thorough evaluation of the speed and accuracy of the enhanced SWORD. We evaluate the effectiveness at identifying which hosts within a network are infected against a wide variety of worm propagation models, including random scanning, local-preference random scanning, and topological scanning. We also consider the effects of polymorphism and encryption of the worm payload.

The paper is organized as following. We will begin by reviewing some previous work related to our research in Section 2, followed by our design for enhancing SWORD to detect worm-infected hosts in Section 3. In Section 4, we then describe our methodology for evaluating the speed and accuracy of the enhanced SWORD. Section 5 presents our results and analysis of our experiments. More discussion is in Section 6, followed by our conclusion in Section 7.

This research is partially supported by a research grant from Intel Corporation.

2 RELATED WORK

Recently there have been studies on how worms may behave [11, 1, 12, 13, 5] and the general requirements for containing them [6]. However, worm defense still remains largely an open topic. Research on worm defense typically falls into three categories:

- Intrusion detection systems (IDS) that identify suspicious behavior as it happens [14, 15, 16, 10];
- Rate-limiting suspicious outbound connections [17, 18];
- Performing forensic analysis of worms [19, 2, 20].

Our approach is to study and evaluate the detection of infected hosts in real time, so research from the first category is closest to ours. Research from the second category is not aimed at detecting which hosts are infected by worms. Research from the third category is complementary to real-time worm detection approaches such as ours but is generally done *after* the fact. In the following, we focus on the first category.

Real-time worm detection generally can be divided into two different categories: host-based IDS and network-based IDS. While conventionally a host-based IDS detects whether or not a host is under attack and a network-based IDS detects whether or not a network is under attack [21], our work studies how well a *network*-based approach performs in detecting *host*-level worm infection events.

Worm IDS can also be divided into misuse-based detection or anomaly-based detection. In detecting host infections of zero-day worms, one popular approach is to quickly discover a byte pattern of a zero-day worm and use that as the signature for detection [9, 8], another is to set up a honeypot (which should not receive any traffic and thus any traffic it receives is probably malicious) and send out worm alerts upon the receipt of unexpected traffic [10, 22, 23]. The byte pattern approach can be used to detect if an individual host is infected or not, but because it needs to check the payload of the traffic, not only will it have a high amount of overhead, but it will also have difficulty detecting polymorphic worms when the payload changes. The honeypot approach cannot directly help detect which local hosts are infected by worms, as a honeypot can only tell for sure that itself is being attacked. Additionally, honeypots suffer from the fact that they can only detect a worm if it scans addresses that are not populated by a regular host. That is, a honeypot will not ever detect a worm that only scans addresses with valid hosts.

As our research is on evaluating the speed and accuracy of the enhanced SWORD in detecting worm-infected hosts, we note that there are several other recent host infection detection systems geared towards worm detection.

- EarlyBird [8] is a signature generation system which has been shown to be effective in discovering zero-day worms. However, the system suffers from having to make a trade-off of false positives for speed. Furthermore, the system is not able to detect polymorphic worms.
- Moonwalk [24] is a system which determines the host which originated a worm attack. It is more of a forensic system, more useful after an infection has been discovered than in detecting an attack. Furthermore, it requires complete connection information—not just a trace at a network's gateway—limiting its overall utility.
- The early warning system from Cliff Zou et al. [25] is also able to quickly detect Internet worms. The focus of their work, however, is to detect the presence of worms in the Internet at large, not to detect which hosts are infected.
- Snort [26] is probably the most widely used IDS, and it is entirely signature based. In order for it to be useful against zero-day worms, another system must be used to supply Snort with the proper signatures.

3 ENHANCING SWORD

In order to evaluate the performance of detecting the worm infection at host level we first enhance SWORD so that it is able to detect whether or not a particular host is infected with a zero-day worm. In this section, we first describe how SWORD works, then describe our enhancements.

3.1 SWORD: Self-propagating Worm Observation and Rapid Detection

The SWORD system detects worms by monitoring connection activity at a network's gateway. This scheme is minimally intrusive from a deployment standpoint, requiring far less administration than solutions that require installations on each host of the network. The monitor watches all connections into and out of the network, but does not look at internal network traffic. Because SWORD does not look at the contents of each packet, the packets can be easily coalesced into what we call **ConnectionDescriptors** which contain the basic information about the connection, such as the source and destination address and port numbers, and the TCP flags that have been seen. A *connection* here refers to an end-to-end communication using either the TCP or UDP protocol.

SWORD detects wormy connections by applying a pair of heuristics to each connection that is seen at the monitor. The two heuristics look for patterns in the connection traffic that are expressions of some of the essential characteristics of worm behavior: similarity between connections with a causal relationship by the *Causal Similarity Heuristic*, and non-power-law destination host distribution by the *Destination Distribution Heuristic*. These heuristics are completely agnostic to the content or payload of the connections, so are entirely robust to polymorphic worms. A connection is considered to be "wormy" if both heuristics agree that it is. Note that due to false positives in the detection process, a connection identified as "wormy" may or may not in fact be an actual "worm" connection.

SWORD then further discovers whether a TCP-based (or UDP-based) worm outbreak is indeed occurring at an administrative domain by determining whether the total number of outgoing TCP (or UDP) wormy connections from the domain during a sliding window is above the alert threshold. If so, an alert is then raised that a worm is active in the domain. The threshold is determined by observing normal traffic from a domain during a sliding window and measuring the total number of connections that would be considered to be wormy connections according to the two heuristics.

3.2 Enhancing SWORD to Detect Worm-Infected Hosts

Whereas SWORD is able to detect whether a domain has a zero-day worm, we enhance SWORD so that we can also detect whether an individual host is infected. We do so without requiring intrusive monitoring systems installed on end-hosts. Instead, we follow the same spirit of the original SWORD system. If the number of wormy TCP connections from a particular host during a sliding window is above the host-level threshold for detecting the infection of TCP-based worms, then a TCP-based worm has infected the host in question. The same is true for UDP-based worms; *i.e.*, if during a sliding window a host sends out more wormy UDP connections than the host-level threshold for detecting the infection of UDP-based worms, then the host has been infected by a UDP-based worm.

The process in obtaining the host-level thresholds is also similar to the original SWORD for domain-level thresholds. By observing normal TCP (or UDP) traffic from individual hosts during a sliding window, one can obtain the total number of TCP (or UDP) connections that the two heuristics would label as wormy connections according to the two heuristics, and use this number as the host-level threshold for detecting the infection of a TCP-based (or UDP-based) worm.

Such enhancement is straightforward, but it serves two important purposes: (1) As in the original SWORD, it continues to be a gateway-based approach and is transparent to individual hosts. (2) The performance of host infection detection can then be evaluated based on an approach that we believe is superior to payload-inspection-based or honeypot-based worm detection approaches.

4 METHODOLOGY

We adopt a trace-based simulation approach to evaluating the enhanced SWORD in detecting host infection. In the following, we first describe what metrics we use for evaluation, and then describe the traffic we choose in the evaluation.

4.1 Metrics

The metrics we use must be able to evaluate the following: the *latency*, or the average speed with which that any given host is detected to be infected with a worm, and the *accuracy* with which any infected host is identified.

The latency is simply the time it takes to detect that an infected host is infected, which we calculate by subtracting a host's time of infection from its time of detection. We report the value averaged across all of the hosts infected for a given run of the experiment.

The accuracy is slightly more complex. At a basic level we need to know *false positives* (how many non-infected hosts were flagged as being infected by the system) and *false negatives* (how many infected systems were *not* flagged as being infected). Moreover, we will need to know *adjusted false negatives* to find out how accurate we are in detecting those hosts that initiate more worm connections than the level defined by the threshold. This measure is specific to our experimental setup and filters out those false negative occurring from a host being infected immediately before the termination of the experiment. We feel that this adjusted measure more accurately represents the true performance of the enhanced SWORD because it does not penalize SWORD for the limitations of our experimental setup. We will use the false negative and adjusted false negative statistics to present *accuracy* and *adjusted accuracy* percentages which are, respectively, the percentage of worm infected hosts correctly detected and the percentage of worm infected hosts correctly detected which sent more than the threshold amount of worm connections (false positives are not addressed as they did not occur in our results).

Table 1. Auckland-IV Trace Details

Date	Active Hosts	Out Conns.	In Conns.
2001/03/06	2,344	2,459,281	979,366
2001/03/07	2,270	2,352,294	929,511
2001/03/08	2,296	2,263,636	1,074,695
2001/03/09	2,283	2,328,105	864,532

4.2 Background Traffic

We use a pre-recorded network trace from a real network as the background traffic in our experiment. We did not use a live network feed because we need to run controlled and repeatable experiments. Simulated traffic was not an option because it there simply isn't any way to simulate traffic with the realism that we require.

The real trace in our experiments is the Auckland-IV trace [27]. It is a continuous 45 days GPS-synchronized IP header trace recorded between February and April 2001 at the University of Auckland and Auckland University of Technology. Traffic was tapped from an OC3 ATM link that connects the Universities to the service provider. The inside networks contain two /16 and several /24 prefixes and all IP addresses are anonymized in the trace. The trace includes all the TCP and UDP header information necessary for our experiments, but no payload information. We redistributed the anonymized IP addresses from the trace into a fictionalized IP range that properly reflects the topology of the network inside the Auckland border router. On any given day there were approximately 2,250 hosts making roughly 2,300,000 total outbound connections and another 960,000 incoming connections (see Figure 1). The hosts which were active varied from day to day, and roughly 5,000 total internal hosts were active at some point in the trace.

We do not know, of course, whether there are any worms active in this trace, or which connections in the trace represent those initiated by an infected host, so there is no way to test the effectiveness of the enhanced SWORD against this trace alone. Instead, we make the assumption that this trace has no worm traffic in it whatsoever, and then we inject our own simulated worm traffic, which we discuss in the following section.

4.3 Worm Traffic

We have created our own worm simulator to model the spread of all of the different types of worms used in our experiments. It uses a network topology that matches that of the Auckland trace—two internal /16 networks separated from the Internet by a border router, and captures traffic that crosses the border router to a trace file.

Our worm simulator uses a high-fidelity [28] finite-state model to simulate the behavior of each vulnerable host in both the Internet and the internal network. This is feasible because the number of vulnerable hosts is only a small fraction of the total hosts in the Internet and because we do not model congestion effects or background traffic within the simulator. Addresses where a host is active in the internal network are derived from detected host activity in the real trace, and

Table 2. Network Topology Details

Internet Details	
Total Addresses	$\sim$4 billion (2^{32})
Hosts	$\sim$300 million (from isc.com)
Service Runners	3 Million
Vulnerable Hosts	300,000
Internal Network Details	
Total Addresses	$\sim$128,000 (2^{17})
Hosts	5,000 (from trace)
Service Runners	500
Vulnerable Hosts	500

external hosts are probabilistically allocated. Not all hosts run the service that the worm is attacking, we assign service runners probabilistically in both the internal and external network. See Table 2 for details.

The worm simulator accurately models the connection-level interaction between two hosts during an infection attempt down to setting appropriate TCP flags. The payload and target port number are configurable. We chose to use port 80 as the target port. This is one of the most numerous ports used in the Auckland trace (roughly 50% of connections go to port 80) which should make detection as hard as possible.

The simulator uses a pluggable propagation model, allowing us to simulate a number of different propagation schemes. For our experiments we simulated random, local preference, and topological scanning modes [1]. We also controlled the speed of the worm, allowing us to study relatively high speed (100 connections/second) and low speed (1 connection/second) worms.

Random scanning worms choose each new target address randomly. Local-preference worms choose randomly, but with a preference towards choosing randomly from the local subnet. Our local-preference worm chooses an address from its class B address space with a 50% probability, from its class A address space with a 25% probability and from the Internet as a whole with a 25% probability. The topological scanning worm starts with a list of roughly 500 addresses known to be running the target service. Once these have been contacted, it reverts to a pure random scanning worm. The worm simulator supports polymorphic payloads for any of the propagation models, but we do not run these simulations because the SWORD heuristics do not examine payload characteristics.

To create a worm trace, we begin with 3,000 hosts infected in the Internet and a worm connection crossing the border router which infects one of our internal hosts. This is meant to replicate a reasonable scenario for a newly introduced worm at the early stage of its development while also reducing the time required to run our simulations. We run each simulation until 100,000 worm connections have crossed the gateway, or 1 hour has passed in the simulation, whichever comes first. Every scenario is run 10 times to create 10 unique worm traces for each speed and propagation combination.

4.4 Merged Traffic

To run experiments against the enhanced SWORD, we create a merged trace to be the input. The merged trace consists of a single day's worth of connections just from the Auckland trace to warm up the heuristics, followed by connections interlaced from the desired worm trace and the next day's Auckland trace. These connections are interleaved based on the connection start-time, resulting in a single trace where we can identify which connections are worm connections and which hosts are infected at what time, allowing us to accurately measure the performance of our detection system.

5 RESULTS

In this section, we first present our results in selecting thresholds for detecting host infection, then report the accuracy and latency in detecting the host infection.

5.1 Threshold

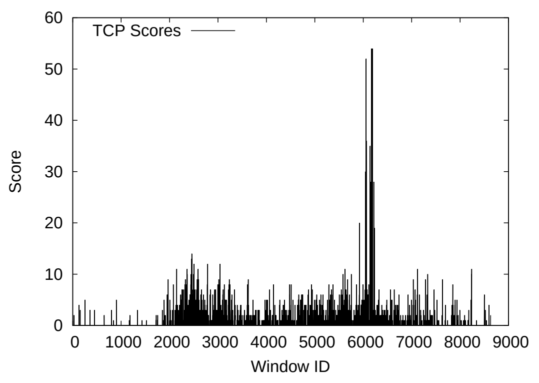
The *sliding window size* and the *threshold* are two key factors in the success of the enhanced SWORD system. The sliding window size determines the length of the period over which the wormy connections from a given host are counted. The threshold differentiates the number of wormy connections from a host's normal traffic—also called *background noise*—and that from a worm-infected host. The heuristics we utilize are expected to generate low levels of false positives at the connection level, perhaps labeling certain legitimate connections from a host wormy. It is important that the threshold is set in such a way that these false positives, *i.e.* background noise, do not trigger a report that a host is infected.

The selection of the sliding window size and the threshold is a compromise between detection sensitivity and latency. Our goal is to choose the smallest window size that still yields enough sensitivity to detect even slow-moving worms. Evaluation of the background (non-worm) traces reveals that the connection-level false positives tend to be quite bursty, as can be easily seen in Figures 1(a) and 1(b) and Tables 3 and 4.

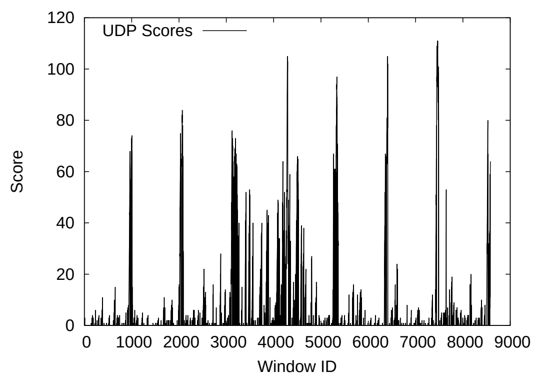
As we increase the size of the sliding window, it increases the absolute value of the threshold, but reduces the average number of connections per unit of time needed to exceed the threshold. A larger window is therefore desirable as it allows us to detect worms with slower scanning rates. The size of the window also impacts the speed with which we can detect worms, however. We must identify enough worm connections from a given host to exceed our chosen threshold, so as the threshold grows with the window size, the number of connections a worm must send to exceed it also grows. A smaller window is therefore desirable to maintain low detection latencies.

We chose our TCP and UDP thresholds and window sizes after examining four days of the Auckland trace. We ran the enhanced SWORD against the (assumed to be non-worm) connections from the Auckland trace and measured the average and maximum wormy connection counts for various window sizes. We selected a 2-minute window size for TCP connections with a threshold of 55 wormy connections, and a 4-minute window for UDP connections with a threshold of 112 wormy connections (Tables 3 and 4). These thresholds are higher than any measured bursts of false positives in the background traces, and substantially higher than the average observed value.

In researching the source of the background noise in the traces, we discovered that many of UDP false positives were triggered by DNS activities. This is not surprising, as DNS lookups may generate many similar connections to many different hosts, matching the criteria of both of our heuristics.



(a) TCP



(b) UDP

Figure 1. Background Trace Scores vs Time. Score is number of wormy connections from a single host during sliding window ending at this time.

Table 3. Background Trace TCP Score Details

Date	Mean Window Score	Max Window Score
2001/03/06	2.88 ± 2.29	20
2001/03/07	2.94 ± 2.26	12
2001/03/08	3.27 ± 4.40	54
2001/03/09	2.66 ± 2.02	13
Window: 2 minutes, Threshold Chosen: 55		

Table 4. Background Trace UDP Score Details

Date	Mean Window Score	Max Window Score
2001/03/06	22.23 ± 22.05	109
2001/03/07	24.66 ± 24.78	95
2001/03/08	31.03 ± 27.61	111
2001/03/09	28.23 ± 27.54	99
Window: 4 minutes, Threshold Chosen: 112		

This implies that white-listing the DNS connections would improve our results by lowering our UDP window size and threshold (see Table 5). However, this would also create a hole in our coverage that could be exploited by a worm targeting a DNS vulnerability. The decision of whether the improved performance is worth the reduced protection is a matter of judgment and a case-by-case decision. In the remaining results sections, we will present our results *without* DNS white-listing save for a brief discussion in Section 5.4.

5.2 Accuracy

Having established a suitable threshold, we can now examine the accuracy of the enhanced SWORD at identifying worm infected hosts.

Table 5. Background Trace UDP Score Details w/DNS white-list

Date	Mean Window Score	Max Window Score
2001/03/06	4.38 ± 6.61	49
2001/03/07	4.31 ± 6.11	49
2001/03/08	2.44 ± 1.87	16
2001/03/09	3.53 ± 3.51	36
Window: 2 minutes, Threshold Chosen: 50		

Table 6. Accuracy: High Speed TCP Worms

Type	Accuracy	Adjusted Accuracy
Random	100.0 ± 0.0	100.0 ± 0.0
Topological	98.49 ± 2.30	100.0 ± 0.0
Local Preference	88.08 ± 2.26	95.69 ± 1.30

Table 7. Accuracy: High Speed UDP Worms

Type	Accuracy	Adjusted Accuracy
Random	100.0 ± 0.0	100.0 ± 0.0
Topological	97.22 ± 2.91	98.72 ± 2.65
Local Preference	79.55 ± 2.42	96.92 ± 0.75

We begin by evaluating the enhanced SWORD's performance against high speed worms. The results for both the TCP and UDP experiments are presented in Tables 6 and 7 and show the enhanced SWORD's exceptional performance.

The enhanced SWORD correctly identified 100% of the infected hosts with zero false positives in all of our experiments for the TCP-based random scanning and topological scanning worms (Table 6). These worms are more likely to initiate a worm connection to an address across the gateway than the local preference worm is, so they are easier for our system to detect. The local preference worm sends fewer connections across the gateway, which allows for normal traffic from the infected host to have a better chance of interfering with our detection mechanisms. Even so, we correctly identified more than 95% of the infected hosts in the local preference experiments.

The performance against UDP worms is not quite as good (Table 7), due to the increased threshold and window size. Each infected host must produce more identified worm connections to exceed the threshold, and the longer window allows more time for normal traffic to interfere with correct connection categorization. Even so, we correctly identify 100% of the infected hosts for the random propagation model and more than 96% of the infected hosts for the topological and local preference worms.

The enhanced SWORD's performance against low-speed worms is similar, but suffers somewhat at detecting TCP local preference worms (Tables 8 and 9). One might expect that detecting the UDP local preference worm would be more difficult than the TCP-based version due to the higher UDP

Table 8. Accuracy: Low Speed TCP Worms

Type	Accuracy	Adjusted Accuracy
Random	100.0 ± 0.0	100.0 ± 0.0
Topological	99.17 ± 2.50	100.0 ± 0.0
Local Preference	80.56 ± 1.84	87.54 ± 1.89

Table 9. Accuracy: Low Speed UDP Worms

Type	Accuracy	Adjusted Accuracy
Random	100.0 ± 0.0	100.0 ± 0.0
Topological	96.81 ± 4.97	100.0 ± 0.0
Local Preference	77.21 ± 2.00	95.01 ± 1.34

threshold, but instead it seems to be the shorter TCP window that limits detection efficacy.

5.3 Latency

We have seen that the enhanced SWORD can accurately identify worm infected hosts, but the question remains: can this be done in a timely fashion? The Sapphire/Slammer worm was able to infect most of its vulnerable population in under 10 minutes [2], setting a benchmark we must be able to beat to field an effective worm detector. In fact, our results show that we are able to detect worm activity substantially faster than 10 minutes for both high-speed and low-speed worms.

When run with our high-speed worm variants, the enhanced SWORD was able to detect infected hosts in under 10 seconds in all cases (see Figure 2(a)). This shows that the enhanced SWORD is indeed capable of countering a worm like Sapphire/Slammer. Note also that the connection rate of this high-speed worm is reasonable in comparison to the Sapphire/Slammer. The Sapphire/Slammer worm is bandwidth limited, and a host on a 100 Mbit/s connection would be capable of sending approximately 300 connections/second. This is greater than the 100 connections/second of our experiment and would therefore be even easier for the enhanced SWORD to detect.

There is little variation in the high-speed results between the different worm types because these high speed worms produce enough wormy connections to overwhelm our thresholds. In fact, for worms with speeds as high as these, the fact that we only check the host scores every 10 seconds is actually the limiting factor in detection speed. The latency could be improved for these high-speed worms by checking host scores after ever wormy connection, rather than waiting for a fixed time.

There is substantially more variation between detection latencies of the various worm propagation models in the low-speed experiments (Figure 2(b)). The enhanced SWORD is substantially slower at detecting low speed worms due to the lower volume of worm connections, but even so it detects all worm varieties in under five minutes, and all but the Local Preference UDP worm in under two and a half minutes. These times are again faster than the benchmark set by the Sapphire/Slammer worm, and in any case slow scanning worms will be much slower at infecting the entire Internet.

The average latency for detecting slow speed worms which use the local preference scanning method is substantially higher than that of worms which use other scanning methods. This is a direct result of the fact that only 50% of the connections from the local preference worm pass through the gateway and the monitor. These slow speed worms only make one connection per second, so every connection that does not pass through the monitor slows detection by a second.

Table 10. Accuracy: High Speed UDP Worms with DNS White-list

Type	Accuracy	Adjusted Accuracy
Random	100.0 ± 0.0	100.0 ± 0.0
Topological	98.50 ± 2.30	100.0 ± 0.0
Local Preference	90.00 ± 2.03	96.72 ± 1.06

Table 11. Accuracy: Low Speed UDP Worms with DNS White-list

Type	Accuracy	Adjusted Accuracy
Random	100.0 ± 0.0	100.0 ± 0.0
Topological	99.17 ± 2.50	100.0 ± 0.0
Local Preference	89.90 ± 1.52	96.69 ± 1.16

There is a clear difference in the TCP latencies versus UDP latencies because of the different windows and thresholds used, as described in Section 5.1.

5.4 DNS White Listing

The high threshold and long window size used for UDP worms negatively impacts the enhanced SWORD's detection latency. The threshold and window size choices were influenced substantially by false positives stemming from DNS traffic (as was mentioned in Section 5.1), but if DNS traffic were exempted from consideration by the enhanced SWORD, we could reduce both the window size and the threshold which would allow for superior performance.

Table 10 shows that our detection accuracy actually increases marginally for high-speed worms when we employ the DNS white-list, with the topological worm detection rate increasing from 98.72% to 100%. Low-speed worms see a similar improvement (see Table 11) though in no case is there a dramatic improvement.

More substantial improvements can be seen in the latency results, particularly for low-speed worms (see Figure 3(b)). The average detection latency for random and topological scanning worms was reduced by half from near 150 seconds to under 75 seconds. The local preference worm saw an even more substantial improvement being reduced from over 275 seconds to under 140.

The improvement for high-speed worms (Figure 3(a)) was not as impressive, with the local preference results the only ones showing real improvement changing from 7.5 to 6.4 seconds.

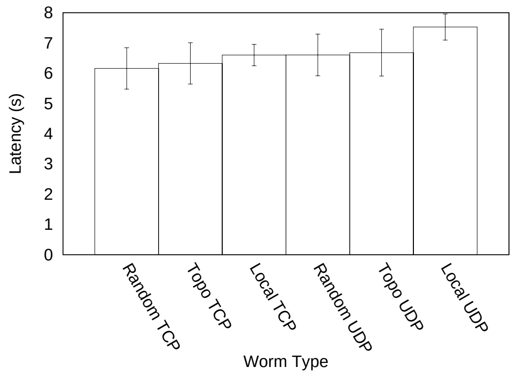
Clearly, exempting DNS traffic from our monitor improves performance. Reducing the detection latency of slowly scanning worms reduces the damage done before they are detected and could be extremely important in limiting the overall impact of a worm outbreak. On the other hand, failing to monitor DNS traffic would leave the network wide open to a worm exploiting a DNS vulnerability.

6 DISCUSSION

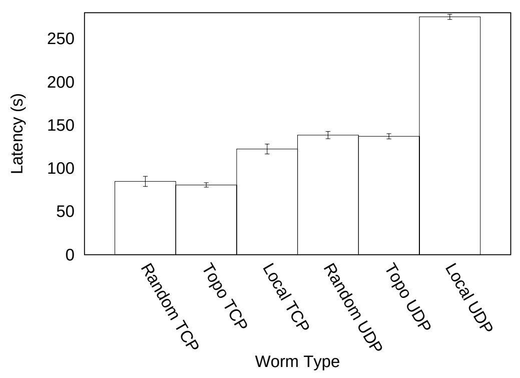
In addition to the above discussion regarding the results of our evaluation, there are some other topics which deserve attention.

6.1 Worm Speed

In our evaluation we considered worms which had both high speed (100 connections/second) and low speed (1 connec-

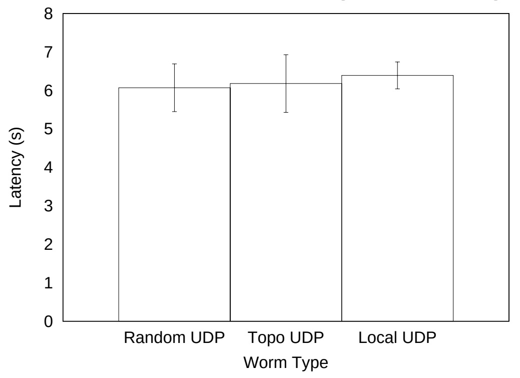


(a) High Speed

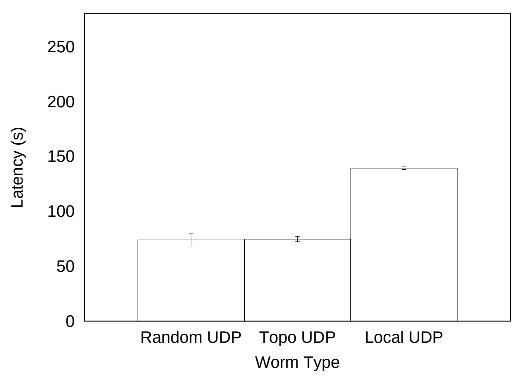


(b) Low Speed

Figure 2. Average Detection Latency for Worms.



(a) High Speed



(b) Low Speed

Figure 3. Average Detection Latency for Worms with DNS White-list.

tion/second) propagation speeds. Of course there are other speeds that worm authors could use, including connection rates of even less than 1 per second. If a worm is slow enough that the traffic it generates is interspersed throughout a large amount of normal traffic, it becomes much more difficult to detect. It becomes more difficult for the Causal Similarity Heuristic to find similar connections, and there are too few worm connections to measurably affect the destination address distribution, thwarting the Destination Distribution Heuristic as well. On the bright side, if a worm propagates so slow as to be undetectable it is also likely too slow to be a real danger.

6.2 Modern Traffic Traces

For our background traffic, we would have preferred a more modern trace including modern peer-to-peer (P2P) traffic, but were unable to find any which met all of our needs. Some readers may worry that P2P traffic would alter the destination address distribution of legitimate traffic, causing false positives or increasing the required detection threshold. However, we are confident that P2P traffic would not seriously affect our results. Consider, for example, Gnutella. Leaf Gnutella nodes usually connect to a small (around 3) number of peers directly and even the core, or ultrapeer, nodes generally only connect to around 30 peers. BitTorrent clients similarly receive only a short list of peers to contact, and DHT (distributed hash table) nodes generally contact only those few nodes in their neighbor lists. These numbers are relatively small and so should not noticeably affect the accuracy of the Destination Distribution Heuristic.

6.3 Possible Improvements

Since the enhanced SWORD detects worm connections passing through a gateway, it can not detect worm connections which are between internal hosts. Therefore, one area of improvement could be related to the detection of internal worm connections. One solution would be to essentially leave the enhanced SWORD as it is and add on a second system such as an ARP-based detection method [29]. Another solution would be to have the system observe all of the internal traffic, in addition to the traffic which crosses the gateway. By observing the internal traffic of the network, we may be able to detect a worm infected host faster than if we only observe traffic going through the gateway. Researching the advantages and trade-offs of each solution is an area of our future work.

Performance also may be improved by using sliding windows with a finer grain—for instance by having separate windows for each port number. This may decrease the threshold required to differentiate worm connections from normal connections, which would in turn decrease the time required for detection.

7 CONCLUSION

Detecting whether or not an individual host is infected by zero-day worms is critical and must be accurate and fast. Our research shows how the SWORD system can be enhanced to successfully meet this goal.

The SWORD system is designed to detect zero-day worms at an administrative domain. By enhancing SWORD to also operate at the host level, our research demonstrates that a host infection detection solution does not have to be rooted at individual hosts themselves, but can be cleanly and transparently deployed at the gateway point of those hosts.

Our contribution in this work mainly lies in the performance evaluation of the enhanced SWORD. By creating synergistic traffic traces through merging real traces with simulated worm traces, we are able to evaluate the efficacy and efficiency of the enhanced SWORD in detecting whether individual hosts are infected by zero-day worms. The enhanced SWORD can not only detect hosts infected by different types of worms, but can also detect both high-speed and low-speed worms, all successfully with high accuracy and low latency. Given that the payload is not relevant in the detection process, the results also apply to polymorphic worms that dynamically change or encrypt their payload.

8 ACKNOWLEDGMENTS

We thank Eric Anderson, Paul Knickerbocker, Eric Purpus, and Zhen Wu for their earlier involvement in this work.

9 REFERENCES

[1] S. Staniford, V. Paxson, and N. Weaver, "How to 0wn the Internet in your spare time," in *Proceedings of the 11th USENIX Security Symposium*, 2002.

[2] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "The spread of the Sapphire/Slammer SQL worm," http://www.caida.org/analysis/security/sapphire/, CAIDA, Tech. Rep., 2003.

[3] ——, "Inside the Slammer worm," *IEEE Security and Privacy*, vol. 4, pp. 33–39, July 2003.

[4] N. Weaver and V. Paxson, "A worst-case worm," in *Workshop on Economics and Information Security*, 2004.

[5] J. Li, T. Ehrenkranz, G. Kuenning, and P. Reiher, "Simulation and analysis on the resiliency and efficiency of malnets," in *Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation PADS '05*, Monterey, CA, June 2005, pp. 262–269.

[6] D. Moore, C. Shannon, G. M. Voelker, and S. Savage, "Internet quarantine: Requirements for containing self-propagating code," in *INFOCOM*, 2003.

[7] J. Li, S. Stafford, and T. Ehrenkranz, "SWORD: Self-propagating worm observation and rapid detection," University of Oregon, Tech. Rep. CIS-TR-2006-03, 2006.

[8] S. Singh, C. Estan, G. Varghese, and S. Savage, "Automated worm fingerprinting," in *Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI)*, 2004.

[9] H.-A. Kim and B. Karp, "Autograph: Toward automated, distributed worm signature detection," in *USENIX Security Symposium*, August 2004, pp. 271–286.

[10] C. Kreibich and J. Crowcroft, "Honeycomb: Creating intrusion detection signatures using honeypots," *SIGCOMM Computer Communication Review*, vol. 34, no. 1, pp. 51–56, 2004.

[11] J. Nazario, J. Anderson, R. Wash, and C. Connelly, "The future of Internet worms," http://www.crimelabs.net/docs/worms/worm.pdf, July 2001.

[12] Z. Chen, L. Gao, and K. Kwiat, "Modeling the spread of active worms," in *INFOCOM*, 2003.

[13] M. Garetto and W. Gong, "Modeling malware spreading dynamics," in *INFOCOM*, 2003.

[14] T. Toth and C. Kruegel, "Connection-history based anomaly detection," in *Proceedings of the 2002 IEEE Workshop on Information Assurance and Security*, June 2002, pp. 30–25.

[15] C. Kruegel and T. Toth, "Distributed pattern detection for intrusion detection," in *Network and Distributed System Security Symposium*. Internet Society, 2002.

[16] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle, "GrIDS: A graph based intrusion detection system for large networks," in *National Information Systems Security Conference*, 1996.

[17] J. Twycross and M. Williamson, "Implementing and testing a virus throttle," in *12th Usenix Security Symposium*, August 2003.

[18] S. Staniford, "Containment of scanning worms in enterprise networks," *Journal of Computer Security*, 2004.

[19] D. Moore, C. Shannon, and K. C. Claffy, "Code-Red: A case study on the spread and victims of an Internet worm," in *SIGCOMM Internet Measurement Workshop*, 2002.

[20] B. N. Chun, J. Lee, and H. Weatherspoon, "Netbait: A distributed worm detection service," http://netbait.planet-lab.org, Intel Research Berkeley, Tech. Rep. IRB-TR-03-033, September 2003.

[21] B. Mukherjee, L. Heberlein, and K. Levitt, "Network intrusion detection," *Network*, vol. 8, no. 3, pp. 26–41, May/June 1994.

[22] D. Dagon, X. Qin, G. Gu, W. Lee, J. Grizzard, J. Levin, and H. Owen, "HoneyStat: Local worm detection using honeypots," in *Proc. of the 7th International Symposium on Recent Advances in Intrusion Detection*, Sophia Antipolis, France, September 2004.

[23] J. Kloet, "A honeypot based worm alerting system," http://www.sans.org/rr/whitepapers/detection/1563.php, January 2005.

[24] Y. Xie, V. Sekar, D. A. Maltz, M. K. Reiter, and H. Zhang, "Worm origin identification using random moonwalks," in *IEEE Symposium on Security and Privacy*, 2005, pp. 242–256.

[25] C. C. Zou, L. Gao, W. Gong, and D. Towsley, "Monitoring and early warning for Internet worms," in *10th ACM Conference on Computer and Communications Security*, October 2003, pp. 190–199.

[26] M. Roesch, "Snort - Lightweight intrusion detection for networks," in *Proceedings of the 13th Systems Administration Conference - LISA '99*. USENIX, November 1999.

[27] WAND Network Research Group, "WAND WITS: Auckland-IV trace data," http://wand.cs.waikato.ac.nz/wand/wits/auck/4/, April 2001.

[28] K. Perumalla and S. Sundaragopalan, "High-fidelity modeling of computer network worms," in *Annual Computer Security Applications Conference*, December 2004.

[29] D. Whyte, E. Kranakis, and P. C. van Oorschot, "ARP-based detection of scanning worms in an enterprise network," in *Annual Computer Security Applications Conference*, December 2005.