Detecting Early Worm Propagation through Packet Matching

Xuan Chen and John Heidemann *

ISI-TR-2004-585
February 2004

Abstract

In this paper, we present DEWP, a router-based system designed to automatically detect and quarantine Internet worm propagation. DEWP detects worm probing traffic by matching destination port numbers between incoming and outgoing connections. This approach does not require knowledge of worm packet contents or profiles of normal traffic conditions; it can automatically detect and suppress worms due to their unusual traffic patterns. We describe how DEWP works and evaluate its performance with simulations. We study the speed of detection and the effectiveness of vulnerable host protection relative to factors including worm scanning techniques, DEWP deployment coverage and detection intervals. We also investigate false detections with network trace playback. We show that DEWP detects worm propagation within about 4 seconds. By blocking worm probing traffic automatically, DEWP can protect more than 99% hosts from random-scanning worms.

1 Introduction

Since the widespread outbreak of the Code-Red worm in July 2001 [1, 2] worm intrusion has become an increasingly severe threat to the Internet. Code-Red II [3], Nimda [4], Slammer [5], and SoBig [6] worms have all led to considerable cost to our society [7, 8]. Prior work [7, 9] has suggested that an automatic worm detection and containment system is important to protect the Internet from worm attacks. We identify three essential requirements for such a system.

First, this system must detect worm propagation at a very early stage to suppress the worm before it gets out of control [7, 8, 9]. For a router-based detection system, it needs to monitor and react to worm traffic within a small time interval (in seconds) in order to quarantine worm propagation quickly. Signature-based intrusion detection systems (IDS) can not respond to these 0-day attacks at this speed due to the time-consuming worm packet content analysis. Anomaly-based IDS improves detection speed by monitoring traffic changes but usually have high false alarm rate. In this work, we believe that routers should apply *simple but essential* worm detection techniques for fast reaction. These techniques should capture key characteristics of worm traffic but are not computational consuming.

Second, a worm defense system needs to create worm signatures without human interference in order to react and quarantine worm propagation rapidly. A signature identifies common characteristics of a specific worm suitable for detection and suppression. For example, it could be the destination port number in worm packet headers, or substrings of worm packet payload. Although signatures based on packet data contents can unambiguously detect a worm, it is challenging for an worm detection system to generate such signatures automatically because of the large computational overhead. On the other hand, we will show that simple signatures based on destination port numbers can effectively detect and contain worm traffic.

Third, it is important to reduce the false alarm rate of worm detection because false positives potentially lead to denial-of-service to legitimate traffic. Therefore, we need to carefully consider the trade-off between fast detection and low false-alarm rate when designing an automatic worm detection and quarantine system.

In this paper, we present a router-based worm detection and containment system called DEWP (Detector for Early Worm Propagation). As observed below, DEWP detects worm intrusions, creates worm signatures, and contains worm propagation automatically, without human interference. We believe it will be especially important against worms that propagate rapidly [9], such as the Slammer worm.

*Xuan Chen and John Heidemann are with University of Southern California, Information Sciences Institute. This material is based upon work supported by DARPA via the Space and Naval Warfare Systems Center San Diego under Contract No. N66001-00-C-8066 ("SAMAN"), and by the CONSER project supported by NSF.

This work has three major contributions. First, DEWP applies a novel worm detection algorithm by matching destination port numbers between incoming and outgoing connections. This idea is from the following two observations on worm traffic. First, a worm usually exploits particular security vulnerability corresponding to specific network port numbers. Second, the nature of worms is that an infected host will probe other vulnerable hosts exploiting the same vulnerability. Therefore, routers seeing unusually high levels of bi-directional probing traffic with the same destination port number can infer a new worm has arisen. In addition, since matching destination port numbers consumes low computational power, DEWP is more applicable to real networks than systems based on packet content analysis such as the Early Bird System [10].

Second, we evaluate DEWP performance by simulating an outbreak of the Slammer worm, which is known for the extremely fast spreading (infected about a total of 75,000 hosts in about 15 minutes). Our results show that DEWP detects worm propagation in about 4 seconds. With automatic destination port discovery and packet blocking, DEWP protects most hosts from infection: less than 1% hosts are compromised by random-scanning worm, and about 9% hosts are infected by local-scannning worm. We further investigate several important factors that affect DEWP performance including worm probing techniques, deployment coverage, and detection intervals. We also study issues on false detections with network trace playback.

The final contribution is that we introduce a new hybrid simulation model of worm propagation. This model allows detailed packet-level simulations within one particular network while representing the rest Internet analytically. We present the detailed description of our model in Section 5.

2 Related Work

In this section, we first briefly describe techniques to detect worm attacks. We also review different approaches to quarantine worm propagation.

Intrusion detection systems (IDS) are deployed to discover DDoS attacks and worm intrusions. There are two different kinds of IDS, namely signature- and anomaly-based. Signature-based IDS [11, 12] capture worm attacks based on pre-compiled signatures stored in database. Although they identify threats accurately, signature-based IDS have little effect on unknown worms. On the other hand, anomaly-based IDS [13] detect new threats by observing unusual traffic changes, that is the difference between current traffic measurement and its normal condition (usually in a profile). Since it is difficult to create a "proper" profile to cover all representative aspects of normal traffic condition, anomaly-based IDS have high false alarm rate.

DEWP does not need signatures with worm packet contents or traffic profiles. By matching destination port numbers between incoming and outgoing connections, it automatically detects worm intrusion within seconds. DEWP also has lower false alarm rate since it captures common characteristics of worm traffic directly.

Early Bird System (EBS) [10] is proposed to automatically detect worm propagation. Similar to our work, EBS also discovers worm by observing common patterns in network traffic. Unlike the simple technique of destination port matching, EBS needs per-packet content analysis and complicated hashing function to identify suspicious worm traffic aggregates. Therefore, its applicability to real networks is questionable.

Honey-pots [14, 15] and its variations (through aggregation and virtulization) monitor un-used address space (that is, dark space) and capture traffic interaction with dark space as worm intrusions. Honeypot is a flexible technique to detect unknown threats. However, it only detects worm propagation when being directly probed. Also, honey-pots by themselves could be compromised by worms.

Since worm is a threat to the Internet globally, some proposals [9] call for a nation-wide Internet worm control authority to coordinate worm detection and immunization efforts. Other intrusion detection system [16] also proposed to deploy sensors around the Internet, collect traffic statistics, and send them to a data processing center. We agree that global coordination is necessary to protect the Internet from worm intrusions. However, there are some difficulties for these proposals to realize in near future, such as resource constraint and administrative issues. So, we take another approach of designing a distributed system that detects worm probing traffic through local traffic observations.

NetBait [17] is a distributed system that provides detailed information about network intrusions. It collects data from geographically located machines, which use traditional intrusion detection systems (such as Snort [11]) to discover worm attacks. The goal of NetBait is to provide accurate information to identify infected hosts and expedite the process of worm containment and cleanup. It is complementary to DEWP which aims at fast detection of worm propagation.

Current schemes to contain worm propagation is by

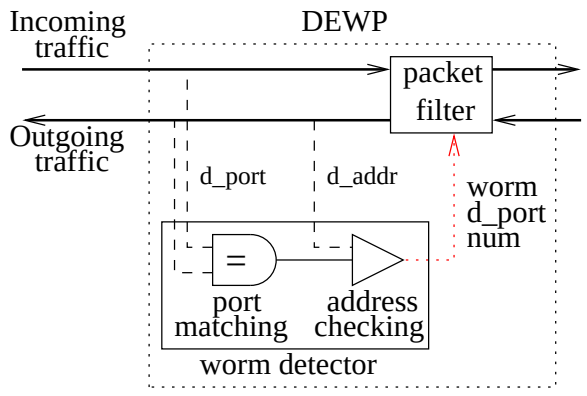


Figure 1: DEWP architecture.

packet filtering or address blocking [7, 8]. Researchers at Silicon Defense propose to partition enterprise network into small cells, and quarantine worm propagation coordinately [18]. In this work, DEWP applies packet filtering to suppress the spread of worms.

"Virus throttle" [19] is a scheme to slow down worm propagation by limiting the number of outgoing connections that one host can initiate simultaneously. To be effective, we need to modify current network implementation on most end-hosts.

3 Worm Detection and Containment

Internet worms can spread very rapidly [7, 9, 20]. For example, Code-Red II worm totally infected about 360,000 computers in 14 hours with a peak aggregate infection rate of 2,000 hosts per minute. The latest MyDoom worm compromised about 20,000 hosts in total within 2 hours after it was first discovered. In order to detect worm traffic promptly, routers need to automatically examine traffic changes and frequently conduct worm detections. As a result, when designing worm detection algorithm for routers, we need to choose *simple but essential* techniques so that worm detection does not interfere with routers' normal operations.

Figure 1 shows the two components that make up DEWP: the worm detector and packet filter. DEWP detects worm intrusions with two steps: destination port matching and destination address counting. After it discovers a worm attack and the corresponding destination port number, DEWP deploys packet filter to block worm probing traffic. We present detailed design considerations in following sections.

3.1 Detecting Worm Propagation with Destination Port Matching

DEWP applies a two-step detection algorithm, first port-matching, then address-counting. It first identifies suspicious traffic by matching destination port numbers between incoming and outgoing connections. This *simple* technique actually captures *important* characteristics of worm probing traffic: worms usually exploit a vulnerability in the same service, and so the same destination port number will be prominent in both incoming and outgoing traffic when a worm is spreading[1]. Also, since worms attempt to infect as many vulnerable hosts as possible, compromised hosts both receive and send worm probing traffic. As a result, corresponding access routers observe probing traffic in both directions.

DEWP uses port matching as a first line of filtering, since it can be done efficiently in routers, and since it can reduce computational overhead for per-port checking. We examine the effectiveness of port matching in Section 6.6. While one might want to explicitly monitor ports with source and destination addresses (watching a particular host become infected and then this infection spreading to another), we do not take this approach because a worm can often spoof its source address when spreading. So, this check is less robust.

Port-matching alone may identify some legitimate traffic as potential worm traffic. For example, an access router of one ISP network may observe normal web requests toward both inside and outside servers. So, port matching will capture web traffic as suspicious worm probings. To distinguish legitimate traffic from worm probings and avoid false positives, DEWP introduces a second step *address counting*. It counts distinct destination addresses of outgoing connections to suspicious ports (N). DEWP identifies worm traffic when observing large increase in N. The underline assumption is that worms probe many more unique addresses than normal traffic [19], and that worms result in a rapid increase in number of unique addresses seen. Through address-counting, DEWP is also able to capture worms that utilize popular network services such as sendmail (MyDoom and SoBig) and Web (Code-Red).

3.2 Containing Worm Propagation

There are two approaches to quarantine worm propagation: address blacklisting and traffic filtering. DEWP uses traffic filtering, asking routers to

[1]Even for polymorphic worms that attack multiple vulnerabilities, the ports used by each vulnerability will appear in traffic.

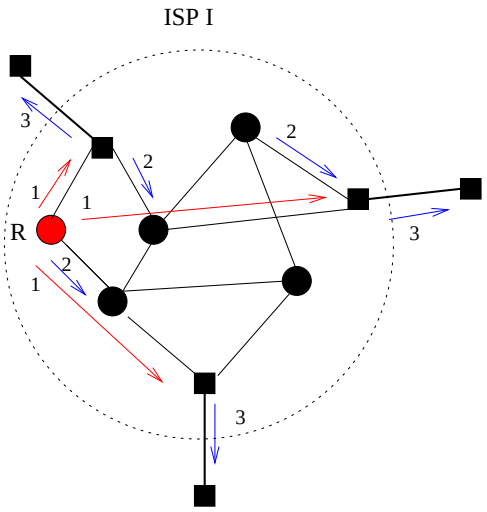


Figure 2: Worm containment in ISP I.

drop packets with the automatically discovered destination port. We have chosen this alternative because it protects more vulnerable hosts than address blacklisting given the same worm attack and reaction time [7].

There are two aspects for worm containment: protecting internal hosts from both internal and external threats, and notifying other networks about the ongoing attack. Theoretically, it is always beneficial to deploy DEWP to additional ISPs if some other ISPs have already done so. This is because that multiple systems can coordinate worm detection, and share the worm destination port number discovered. However, there are some difficulties in reality that prohibit such coordination including administrative issues and mutual trustiness among ISPs. So, we focus on deploying DEWP within one ISP in this work.

In order to show the detailed procedure of worm containment in one ISP network, we present an example below. We assume that routers can communicate with each other using schemes such as multicast. For an ISP I with DEWP deployed (shown in Figure 2), suppose router R detects the worm. It first deploys packet filter locally. R may also be able to locate infected hosts and block their network connections individually.

Then, R notifies access routers of ISP I (labeled as 1 in Figure 2) so that internal hosts are protected from outside probes. Also, outside hosts are no longer scanned by infected hosts in I. Finally, routers (including access routers and R) that aware of worm propagation further alert their neighbors (labeled as 2). This process continues recursively within I until all DEWP-aware routers deploy traffic filter.

To protect hosts that are connected through other ISPs, access routers of I need to notify its peers about the worm attack (labeled as 3). Upon receiving these notifications, routers in other ISPs repeat the three steps described above.

In this work, we focus on early detection and fast containment of worm propagation Since DEWP quarantines worm propagation with port-based packet filtering, it may block legitimate traffic to the discovered port. However, we believe that this effect only exists when a worm is initially detected. Once we apply more sophisticated algorithms to further identify worms such as finding signatures based on worm packet contents, we should be able to resume service for legitimate traffic [8].

4 System Design

DEWP keeps track of both incoming and outgoing connections. It maintains one list for each direction (port-list) to record the number of connections to different destination ports. DEWP also keeps a timer for each entry in port-lists. If one port has not been accessed for certain time interval, DEWP resets the corresponding list entry. By this means, we reduce false positives.

DEWP matches non-zero entries in both port-lists, and further monitors the outgoing destination addresses of those connections. Every T seconds, DEWP checks the number of unique addresses observed within last time interval. DEWP detects worm traffic with the following condition:

$$N \quad > \quad \overline{N} \times (1 + \delta) \tag{1}$$

N is the number of unique addresses observed and $\overline{N}$ is its long-term average. For simplicity, we use the Exponentially Weighted Moving Average (EWMA) to compute $\overline{N}$: $\overline{N} = \alpha\overline{N} + (1 - \alpha)N$.

In the above condition, δ reflects system's sensitivity to changes. Small δ expedites worm detection with a potentially higher false alarm rate. On the other hand, large δ increases detection confidence with a cost of reduced sensitivity. Based on our experience, we choose $\delta = 1$ and $\alpha = 0.125$ in our simulations. We investigate the effect of detection interval T and sensitivity parameter δ on DEWP performance in Section 6.4 and 6.6.

For worm containment, we do not consider the actual procedure to notify neighbor routers in our current work. Hence, we ignore the notification delay among routers. We will investigate this issue in our future work.

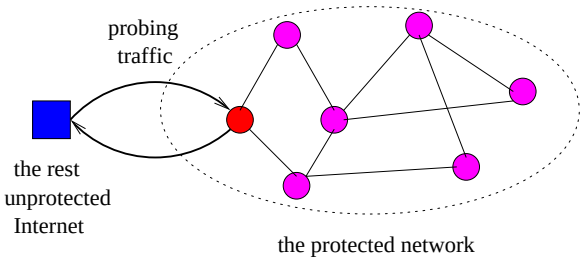


Figure 3: The worm propagation model.

5 Modeling Worm Propagation

In this work, we consider the scenario that one ISP network deploys DEWP to protect its hosts. This imposes two requirements for the worm propagation model in simulations. First, it should allow detailed packet-level simulations to evaluate DEWP performance in the protected network. Second, it also needs to reflect worm spreading throughout the whole Internet.

However, due to CPU and memory constraints, it is usually impossible to simulate millions of routers and hosts with packet-level details. Fortunately, since we are interested in deploying DEWP to a particular network, there is no need to model the whole Internet with such details. So, we apply an analytical model to represent the rest Internet without DEWP protection. Compared to the protected network, we only need to track several state variables in this abstract world such as the number of infected hosts.

As shown in Figure 3, we integrate SIR (susceptible-infectious-removal) model [7, 9, 20, 21]. with packet-level simulations in the protected network. The interaction of these two parts is through actual probing packet transmissions.

Our model has three major differences from prior work [20]. First, instead of applying the analytical model to the whole Internet, we partition the Internet into two parts and only model the unprotected network theoretically. Second, we model vulnerable and infected hosts in both parts of the Internet. Third, instead of using analytical model to compute traffic metrics for individual routers in detailed simulations, the protected network in our model interacts with the rest Internet through actual probing traffic.

5.1 Formal Representation of the Model

Formally, we present our worm propagation model with following equations. We describe model parameters in Table 1.

$$dS_U/dt = -\beta I_U S_U/S_U(0) - P_x, \quad (2)$$

Parameter	Meaning
S_U, I_U, R_U	number of vulnerable, infected, and removed hosts in the rest unprotected Internet
β	infection parameter
γ	removal parameter
N_U, N_P	total number of hosts and the
θ_U, θ_P	percentage of vulnerable hosts in both networks
p_r, p_x	number of probing packets received, sent by the protected network within one time unit
P_x	number of effective probing packets sent from the protected network within one time unit
C	worm scanning rate

Table 1: Parameters used in our worm propagation model.

$$dI_U/dt = \beta I_U S_U/S_U(0) - \gamma I_U + P_x, \quad (3)$$
$$dR_U/dt = -\gamma I_U, \quad (4)$$
$$P_x = p_x \times S_U/N_U, \quad (5)$$
$$p_r = C I_U N_D/N, \quad (6)$$

We also have the following equations for the rest Internet:

$$S_U + I_U + R_U = S_U(0)$$
$$S_U(0) = N_U \times \theta_U$$
$$N = 2^{32} - 1$$

$S_U(0)$ is the number of hosts in the rest Internet that are initially vulnerable. N is the address space of the Internet, which is used by worms to choose victims.

There is a rough relationship between worm scan rate (C) and infection parameter (β):

$$\beta = C S_U(0)/N$$

Equations (2, 3, 4) come directly from the mathematic representation of discrete SIR model. It updates states (I_U, S_U, R_U, and p_r, p_x, P_x) at each time unit.

We augment the SIR model by including two new variables p_x and p_r to represent probing traffic between the two networks. As shown in equations 5 and 6, the effective probing P_x is proportional to p_x and the percentage of vulnerable hosts in the unprotected Internet. Further, p_r represents the portion of aggregate probing traffic sent to the protected network. We describe probing traffic generation in Section 5.2.

We can use the mathematical expression of worm propagation model to estimate some important properties. For example, in Appendix A, we compute the time when the protected network receives the first probing packet (probing time), and when the first host in the protected network is compromised (infection time). We verify our simulation results with this analysis.

5.2 Modeling the Interaction between the Protected network and the Rest Internet

One novel aspect of our worm propagation model is that the two networks interact through real probing traffic. With a certain scan rate C, an infected host selects a target to send out probing packets. If the target is vulnerable, it is compromised and starts to probe other hosts. Probing packets have no effect on invulnerable or infected hosts.

Worms have different options to send probing packets. For example, Code Red and Nimda first set up TCP connections with target hosts; while Slammer worm sends probing packets directly via UDP. With TCP connections, worms can send out large payload (for example, 50KB in Nimda's case). But, the scan rate is limited by the end-to-end latency to victims, and therefore can not be very high. On the other hand, UDP-based worms usually have extremely large scan rate, but only one small probing packet (400 bytes in Slammer's case). Their probing is not constrained by latency, but limited by available network bandwidth. As more probing traffic pumped into networks, it not only affects cross traffic, but also interferes among themselves.

We study two probing strategies in this work: random and local-preferred scanning. Worms (such as Code Red and Slammer) that apply random scanning choose an IP address from the whole Internet address space randomly. This address may not even be used. Other worms (like Nimda) prefer to choose local neighbors. In this work, we model local-preference with the probability that an infected host probes its neighbors in the protected network.

We validate the interaction between both networks through simulations (detailed methodology and model parameters in Section 6.1). In Figure 4, we show that the change of infection percentage in both networks follow the same trend. If we interpret the infection percentage as the probability that a vulnerable host will be compromised at time T, Figure 4 can be looked as the corresponding cumulative distribution function (CDF), that is $P(t < T)$. So, we can apply Kolmogorov-Smirnov goodness of fit test to formally determine if these two CDFs are significantly different.

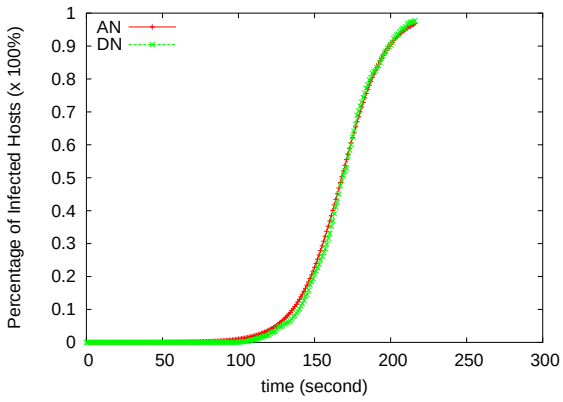


Figure 4: Worm infections in both the protected network and the rest Internet.

Parameter	Value
N_A	200M
θ_A	0.037%
S_{A0}	74,250
β	0.069
N_D	2550
θ_D	30%
C	4000 / second

Table 2: Parameters used to model Slammer worm.

Following the methodology described in prior work [22], we first compute the maximum divergence of these two curves (D value) and compare it to the critical value at 0.05 level significance ($\frac{c}{\sqrt{n}}$, $c = 0.874$ and $n = 256$). Since the D value (0.0515) is smaller than the critical value (0.054625), we accept the null hypothesis that the two curves are not significantly different from each other.

6 System Evaluation

We implement DEWP in the *network simulator (ns-2.27)* [23] and evaluate its performance through simulations. In this work, we only consider scenarios of deploying DEWP to one ISP network.

6.1 Methodology

Since we are particularly interested in the detection and quarantine of fast spreading worms, we evaluate DEWP performance with a simulated outbreak of the Slammer worm.

We choose model parameters (summarized in Table 2) based on previous measurement study on the Slammer worm [5]. For simplicity, we do not consider

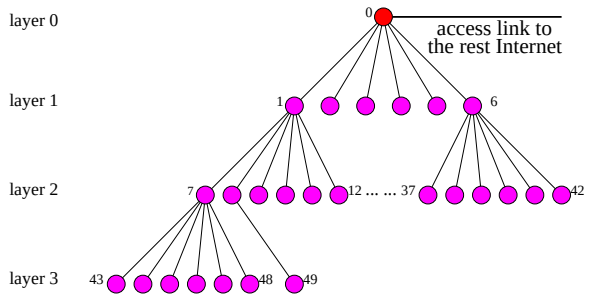


Figure 5: The the protected network topology of a 6-nary tree.

removal function (Function 4, representing hosts that die and are removed from service, or become patched and are "cured") in our current work. To investigate the effect of different worm scanning techniques on DEWP performance, we consider both random and local probings.

For the protected network, we randomly select and mark 30% of end hosts as vulnerable (θ_D). Intuitively, the higher θ_D, the more likely that hosts in the protected network are compromised.

In order to easily study the effect of deployment on DEWP performance, we choose a topology of a 6-nary tree with 50 routers (Figure 5). All connections have 100Mbps bandwidth and 50ms propagation delay. Each router connects 50 hosts with 100Mbps links (25ms propagation delay). We investigate DEWP performance in a random-generated topology in Section 6.5.

We deploy DEWP to routers in the protected network and quantify its performance with two metrics: detection delay and infection percentage. We define detection delay as the time interval from the first probing packet enters the protected network till DEWP detects worm propagation. The infection percentage represents the portion of vulnerable hosts in the protected network that are compromised.

The coverage of DEWP-aware routers in the protected network (deployment) and the DEWP detection interval both affect performance. We vary these parameters to investigate their effects.

Ideally, DEWP will be deployed on all routers. However, in practice, we may only be able to deploy DEWP to some routers due to the difficulty of adding detection algorithm in existing router software and controlling router's access policy. In Section 6.2 and 6.3, we incrementally deploy DEWP to routers on different layers (from 0 to 3 in Figure 5) and investigate how detection performance changes. We further study DEWP performance with different detection intervals in Section 6.4. We also investigate false alarms of DEWP detection algorithm in Section 6.6,

In all scenarios, we run simulations for 50 times and calculate statistics of detection delay and infection percentage in the protected network. Graphs that show mean values also indicate 90% confidence intervals; graphs that show medians instead depict 25% and 75% quartiles.

6.2 Effectiveness of Worm Detection and Quarantine

We first consider a random-scanning worm. In this section, we present numbers of infected hosts instead of showing the small infection percentages (less than 1%). Our simulation results show that DEWP detects worm traffic in 4.8 seconds when fully deployed with a 1 second detection interval. The median number of infected hosts is 1, which is the minimal requirement for DEWP to detect worm probing traffic. Therefore, DEWP quickly detects the worm attack and effectively protects almost all vulnerable hosts from infection.

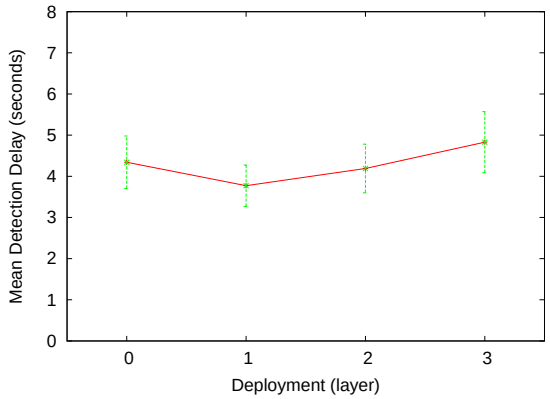
We further investigate the effect of deployment on DEWP performance. As shown in Figure 6(a), DEWP always detects worm probing traffic in 4–5 seconds when deployed to different layers. So, DEWP detection delay is not sensitive to deployment in the scenario of random-scanning worms.

We also observe that the median number of infected hosts remains as 1 when DEWP deployment covers different layers (Figure 6(b)). This is because that infected hosts in the protected network almost always probe outside networks rather than their neighbors: the possibility that an infected host in the protected network probes its neighbors is very small (about 2 out of 10 million probes). So, the number of infected hosts in the protected network is primarily determined by the number of probing packets received from outside. Therefore, DEWP is still able to protect almost all hosts from infection even when only deployed on the access router.

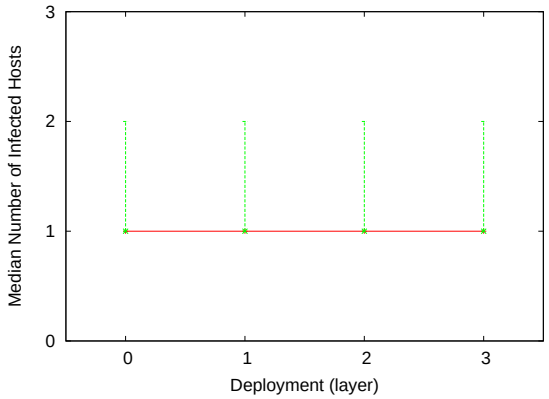
Based on our results, we conclude that deployment does not have significant effect on DEWP performance to detect and quarantine random-scanning worms. Further, it is sufficient to deploy DEWP on the access router to protect vulnerable hosts from random-scanning worms.

6.3 Effect of Worm Scanning Techniques

Worms could apply other scanning techniques. For example, the Nimda worm chooses an address with the same first two octets with a probability of 50% and an address with the same first octet with a probability of 25%. Only 25% of the time, it picks a random address. As shown below, local-scanning worms



(a) Mean detection delay.



(b) Median number of infected hosts.

Figure 6: Detect and quarantine random-scanning worm with different layers of deployment.

spread rapidly within the protected network after the first host is compromised. This property imposes great challenge to worm detection and containment. In this section, we investigate its effect on DEWP performance by simulating a local-scanning worm. With other parameters unchanged, we configure the probability that an infected host within the protected network probes its neighbors as 50%.

With full deployment and 1 second detection interval, we find that DEWP detects worm probing traffic in 3.87 seconds. But, almost all vulnerable hosts in the protected network are compromised before DEWP blocks worm traffic. Also, deployment does not have significant impact on either detection delay or infection percentage. Therefore, with 1 second detection interval, even though DEWP quickly detects worm traffic, it can not effectively quarantine local-scanning worm.

To improve DEWP's effectiveness in worm containment, we reduce the detection interval to 0.0625 second. In Section 6.4, we investigate how different detection intervals affect DEWP performance in details; however here we observe that this more frequent detection reduces vulnerability to local-scanning worms.

Our simulation results show that DEWP detects worm probing traffic in 4.63 seconds with full deployment. We also find that deployment does not considerably affect DEWP detection delay. Given the similar detection delays observed in all scenarios, we conclude that DEWP is able to quickly detect worm attacks regardless probing techniques.

On the other hand, even with full deployment, we still observe about 9% vulnerable hosts compromised in the protected network. The infection percentage increases as we reduce the number of layers with DEWP deployed. As an extreme case, when we only deploy DEWP on the access router, all vulnerable hosts in the protected network are compromised within 10 seconds. Given the difficulty to effectively quarantine local-scanning worms, we conclude that a very small detection interval and wide deployment is critical to protect vulnerable hosts.

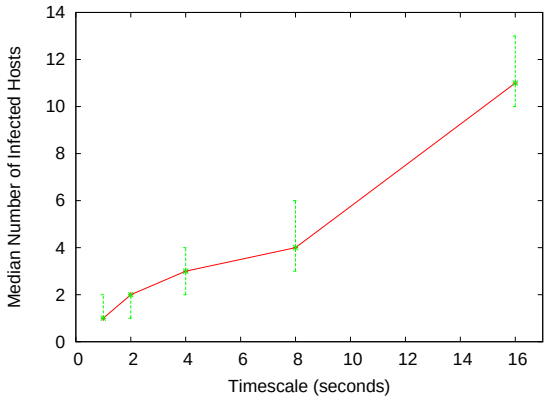
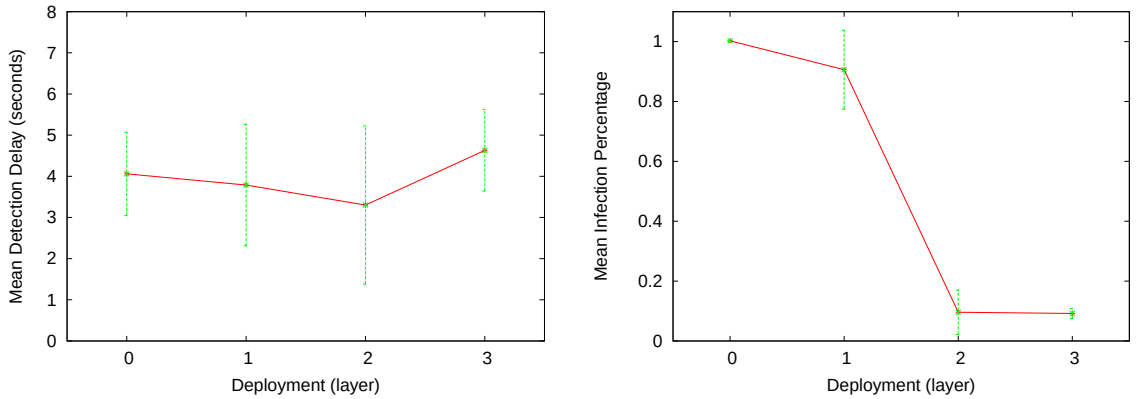


Figure 9: Median Number of infected hosts under different detection intervals.

6.4 Effect of Detection Intervals

DEWP conducts address-counting with an interval of T seconds. Different detection intervals affects DEWP performance such as detection delay and infection percentage. We investigate its effect below. In this section, we fully deploy DEWP on all routers



(a) Mean detection delay.

(b) Mean Infection Percentage.

Figure 7: Detect and quarantine local-scanning worm with different layers of deployment.

in the protected network.

We first consider a random-scanning worm. As shown in Figures 8 and 9, both the detection delay and the number of infected hosts increases with detection intervals. Therefore, we believe that an automatic system should detect worm traffic with small intervals.

For local-scanning worms, small detection interval is even more critical. As shown in Figure 10, although we do not observe significant difference in detection delay (always about 4–5 seconds), the infection percentage increases dramatically at larger intervals: from 9% ($T = 0.0625 second$) to 100% ($T = 1 second$). So, we conclude that an automatic system needs to react to worm traffic within small time intervals in order to quickly detect and effectively quarantine worm propagation.

6.5 Effect of the Protected Network Topology

So far, we have investigated DEWP's effectiveness with a simple tree topology in the protected network. Although this setup has simplified our analysis, it does not reflect real network connections. To evaluate DEWP performance in a more realistic environment, we replace the tree topology with a randomly generated one (using GT-ITM [24, 25]) as shown in Figure 11. Link properties and the access connection of the protected network remain unchanged. We also fully deploy DEWP to all routers.

We first investigate scenarios with random-scanning worms. Our results show that DEWP detects worm probing traffic within about 4 seconds

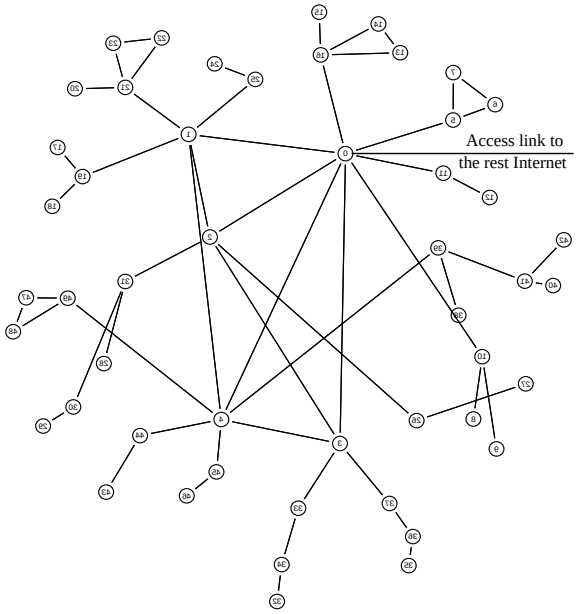
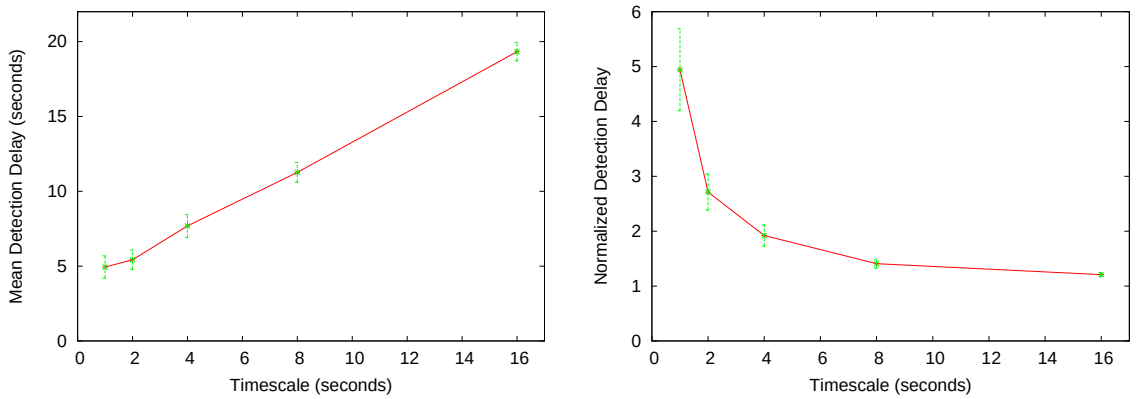


Figure 11: Randomly generated topology for the protected network.

with the detection interval of 1 second. The median number of infected hosts in the protected network is 1. So, we conclude that for these topologies, DEWP is not sensitive to the protected network topology when facing random-scanning worms.

For local-scanning worms, the detection delay is still about 4 seconds with detection interval as 0.0625 second. But, we observe smaller mean infection percentage (about 5%) than our previous results. This is because some nodes in the random generated topol-



(a) Mean Detection delay.

(b) Normalized Detection delay.

Figure 8: Detection interval affects DEWP performance.

ogy has larger out-degree and deeper hierarchies, hence some probing packets are dropped due to congestion caused by more traffic aggregation. As a result, the mean infection percentage reduces.

6.6 Understanding False Detections

False detection is a serious concern of automatic worm detection systems. There are two kinds of false detections: false positives, when DEWP incorrectly identifies legitimate traffic as worms, and false negatives, when DEWP fails to identify worm traffic. We investigate both cases by playing back network trace through DEWP algorithm.

Our trace contains two parts: background and worm probing traffic. For background traffic, we use three one-hour packet trace sets collected from a 100Mbps link connecting USC/ISI to the Internet. These traces were taken on August 21, 2002. We choose different time (9am, 3pm, and 6pm) to reflect variety in traffic. ISI has a B-class network providing services mainly for computer science researchers. It also hosts Web, FTP, sendmail servers and one b.root DNS server.

Because we did not have traces that contain actual worms, we generate synthetic worm traffic using our worm propagation model and add this to our traces. We record packet trace at the access link between the protected network and the rest Internet.

During trace playback, we first start with background traffic. At the simulation time of 50 seconds, we inject worm traffic.

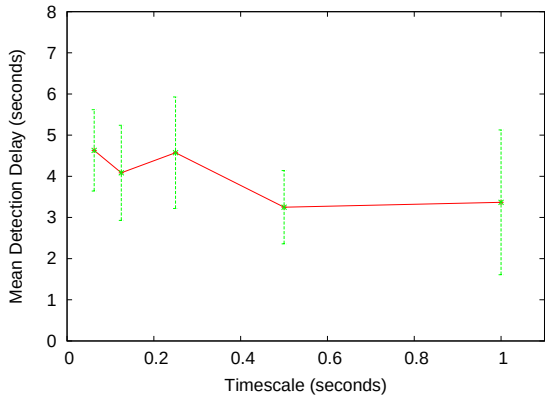
In the experiment, we do not observe any false positives—when both stages of DEWP are used, it never classifies non-worm traffic as a worm. On the other hand, DEWP does discover about 10 suspicious destination ports including 21 (FTP), 53 (DNS), and 80 (Web).

This finding has two implications. First, address-counting is important to reduce false positives. Without this procedure, DEWP will block FTP and Web traffic due to false detections.

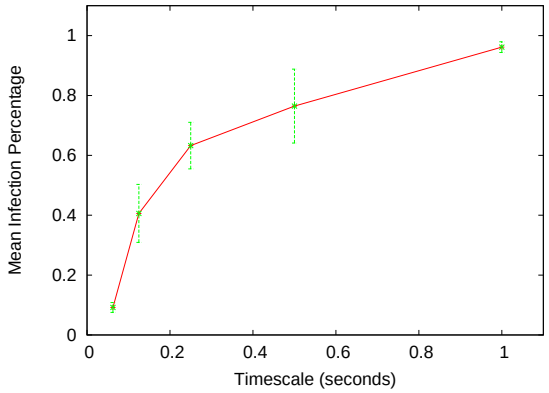
Second, port-matching identifies about 10 different ports out of the total of 542 active ones observed in our trace. So, DEWP only needs to count address to these 10 ports. Since address-counting runs with very small intervals (less than 1 second), port-matching saves the computational power on routers greatly.

In the procedure of address-counting, the sensitive parameter β in condition 1 reflects the trade-off between detection and false alarms. We study DEWP false detections with different values of β. We observe that with $\beta = 1$, there's no legitimate traffic being falsely detected as worm propagation. But, with smaller β, DEWP wrongly identifies other traffic such as Web and DNS ($\beta = 0.5$), and FTP ($\beta = 0.25$) as worms. So, $\beta = 1$ seems to be a plausible configuration of DEWP detection algorithm. We need to further verify this suggestion with traces taken at different networks.

Another important issue affecting false negatives is worm scan rate C. When a worm scan at low rate, its probing traffic has less effect on overall traffic. So, DEWP routers have more difficulty distinguishing them from normal traffic. We investigate different scan rates from 4000 down to 500 per second. We observe that even with $C = 500$, worm traffic still stands out compared to regular traffic: 728 packets per sec-



(a) Mean Detection delay.



(b) Mean Infection Percentage.

Figure 10: Detection interval affects DEWP performance.

ond versus 34 (web traffic). So, we conclude that due to its high spreading speed, UDP-based worm traffic has dramatic larger packet rate than normal traffic.

On the other hand, DEWP is not able to detect worms with scanning rate lower than $C = 25$ because the packet rate of worm traffic is low enough to escape from DEWP detection algorithm. So, low-speed worm is more difficult to detect by only observing traffic changes.

In future work we hope to consider TCP-based worms where scan rate is constrained by the end-to-end latency between infected host and victims.

An important area of future work will be to examine traces that include real worm propagation mixed with live Internet traffic. Unfortunately we were unable to find such a trace for this study. However, we believe that the methodology described above will be applicable to evaluations with real network traces. We also hope to prototype and deploy DEWP to a real network environment to further investigate the issues of false detection.

7 Future Work

As a future direction, we want to extend our worm propagation model for more complicated scenarios, such as multi-homing, and asymmetric routing. We also want to consider removal function in our future work.

Currently, we only investigate worms that send probing traffic via UDP. We will extend our work to TCP-based worms in future. Another interesting direction is to study the interference among worm traffic.

Finally, in order to evaluate DEWP in real networks, implementation is one of our future directions.

8 Conclusion

In this work, we propose DEWP to detect and quarantine the propagation of Internet worms at an early stage. DEWP identifies worm traffic through port-matching and address-counting. We evaluate DEWP performance with simulations. We find that DEWP detects worm attack within 4–5 seconds. By automatically blocking worm traffic, it protects most vulnerable hosts from random-scanning worms. We also study the effect of deployment and detection intervals on DEWP performance. We believe that an automatic worm detection and containment system should be widely deployed and have very small detection intervals. We further investigate issues of false detection through traffic trace playback.

Acknowledgments

We are graceful to Professor Christos Papadopoulos for his insightful comments on related work and deployment issues.

References

[1] Code-Red: a case study on the spread and victims of an Internet worm. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, pages 273–284, Marseille, France, November 2002. ACM.

http://www.caida.org/outreach/papers/2002/codered/codered.pdf.

[2] D. Moore and C. Shannon. The spread of the code-red worm (CRv2). Technical report, CAIDA, the Cooperative Association for Internet Data Analysis, 2002. http://www.caida.org/analysis/security/code-red/coderedv2_analysis.xml.

[3] R. Russell and A. Mackie. Code red II worm. Incident analysis report, SecurityFocus, August 2001. http://aris.securityfocus.com/alerts/codered2/.

[4] A. Mackie, J. Roculan, R. Russell, and M. V. Velzen. Nimda worm analysis. Incident analysis report, SecurityFocus, September 2001. http://aris.securityfocus.com/alerts/nimda/010919-Analysis-Nimda.pdf.

[5] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. The spread of the Sapphire/Slammer worm. Technical report, CAIDA, the Cooperative Association for Internet Data Analysis, January 2003. http://www.caida.org/outreach/papers/2003/sapphire/.

[6] CERT. W32/Sobig.F worm. Incident note, CERT-Coordination Center, August 2003. IN-2003-03.

[7] David Moore, Colleen Shannon, Geoffrey M. Voelker, and Stefan Savage. Internet quarantine: Requirements for containing self-propagating code. In *Proceedings of the IEEE Infocom*, pages –, San Francisco, CA, USA, March 2003. IEEE. http://www.caida.org/outreach/papers/2003/quarantine/.

[8] C. C. Zou, L. Gao, W. Gong, and D. Towsley. Monitoring and early warning for internet worms. In *Proceedings of the ACM conference on Computer and Communication Security*, pages 190–199, Washington D.C., USA, October 2003. ACM.

[9] S. Staniford, V. Paxson, and N. Weaver. How to 0wn the Internet in your spare time. In *Proceedings of the USENIX Security Symposium*, pages –, San Francisco, CA, USA, August 2002. USENIX.

[10] S. Singh, C. Estan, G. Varghese, and S. Savage. The earlybird system for real-time detection of unknown worms. Technical Report CS2003-0761, University of California, San Diego, August 2003. http://ial.ucsd.edu/earlybird/.

[11] Martin Roesch. Snort - lightweight intrusion detection for networks. In *USENIX Large Installation Systems Administration Conference*, Seattle, WA, USA, November 1999.

[12] Vern Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(23–24):2435–2463, 1999.

[13] T. M. Gil and M. Poletto. MULTOPS: A data-structure for bandwidth attack detection. In *Proceedings of the USENIX Security Symposium*, pages 23–38, Washington, D.C., USA, August 2001. USENIX.

[14] Intrusion detection systems, honeypots and incident response. http://www.honeypots.net/.

[15] Lance Spitzner. Honeypots, definitions and value of honeypots. http://www.spitzner.net/honeypots.html, May 2003.

[16] V. Berk, G. Bakos, and R. Morris. Designing a framework for active worm detection on global networks. In *Proceedings of the IEEE International Workshop on Information Assurance*, pages 13–23, Darmstadt, Germany, March 2003. IEEE.

[17] Brent N. Chun, Jason Lee, and Hakim Weatherspoon. Netbait: a distributed worm detection service. http://netbait.planet-lab.org/, February 2003.

[18] S. Staniford. Containment of scanning worms in enterprise networks. White paper, Silicon Defense, October 2003. http://www.silicondefense.com/research/researchpapers/scanContainment.

[19] J. Twycross and M. M. Williamson. Implementing and testing a virus throttle. In *Proceedings of the USENIX Security Symposium*, pages –, Washington, D.C., USA, August 2003. USENIX.

[20] M. Liljenstam, Y. Yuan, BJ Premore, and David Nicol. A mixed abstraction level simulation model of large-scale internet worm infestations. In *Proceedings of the International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages –, Fort Worth, TX, USA, October 2002. IEEE.

[21] H. W. Hethcote. The mathematics of infectious diseases. *SIAM Review*, 42(4):599–653, October 2000.

[22] Kunchan Lan and John Heidemann. Rapid model parameterization from traffic measurement. *ACM Transactions on Modeling and Computer Simulation*, 12(3):201–229, July 2002.

[23] VINT group. UCB/LBNL/VINT network simulator—ns (version 2), 1997. http://www.isi.edu/nsnam/ns/.

[24] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee. How to model an internetwork. In *Proceedings of the IEEE Infocom*, volume 2, pages 594–602, San Francisco, CA, USA, March 1996. IEEE.

[25] E. W. Zegura, K. L. Calvert, and M. J. Donahoo. A quantitative comparison of graph-based models for Internet topology. *ACM/IEEE Transactions on Networking*, 5(6):770–783, 1997.

A Expected Time of Probing and Infection

Using the analytical representation of our worm propagation model (details in Section 5.1), we can compute the time when the protected network receives the first probing packet (probing time), and the expected time when the first vulnerable host in the protected network is compromised (infection time). The probing time is determined by the ratio of hosts in both networks. The larger the protected network is, the earlier it receives probing packets. Since DEWP detects worm propagation after at least one host infected, the difference between probing time and the expected infection time is actually the lower bound of the expected detection delay.

For simplicity, we do not consider removal function (4) below. Since there's no probing packet sent by the protected network before its first vulnerable host is compromised, we ignore function 5. So, we have a simplified model below:

$$\frac{dS_U}{dt} = \frac{-\beta I_U S_U}{S_U(0)}, \quad (7)$$

$$\frac{dI_U}{dt} = \frac{\beta I_U S_U}{S_U(0)}, \quad (8)$$

$$p_r = \frac{C I_U N_P}{N}, \quad (9)$$

The first probing packet is sent to the protected network when $p_r(t) \geq 1$. From equation 8, we have:

$$\begin{aligned} dI_U/dt &= \beta I_U S_U / S_U(0) \\ &= \beta I_U (S_U(0) - I_U)/S_U(0) \end{aligned} \quad (10)$$

So, in our discrete-time model, we have:

$$\begin{aligned} I_U(t) &= \beta I_U(t-1)(S_U(0) - I_U(t-1))/S_U(0) \\ &\quad + I_U(t-1) \\ &= (1+\beta) I_U(t-1) \\ &\quad - \beta I_U^2(t-1)/S_U(0) \end{aligned} \quad (11)$$

Combining equation 9 and 11, we get:

$$\begin{aligned} I_U(t) C N_P / N &\geq 1 \\ \frac{N_P}{N} C((1+\beta) I_U(t-1) & \\ - \beta I_U^2(t-1)/S_U(0)) &\geq 1 \end{aligned} \quad (12)$$

where $I_U(0) = 1$. Therefore, we can solve the above inequation to get probing time. Similarly, we have the expression for expected infection time:

$$\begin{aligned} \frac{N_P}{N} C((1+\beta) I_U(t-1) & \\ - \beta I_U^2(t-1)/S_U(0)) &\geq 1/\theta_P \end{aligned} \quad (13)$$

Given the configuration in our simulations (details in Section 6.1) we find that the probing time is 93.20 seconds, and the expected infection time is 98 seconds. Assuming fully deployment the expected detection delay is about 4 seconds. These results are consistent with our simulations.