On the Detection of Smart, Self-Propagating Internet Worms

Jun Li, Devkishen Sisodia, and Shad Stafford

Abstract—Self-propagating worms can infect millions of computers on the Internet in just several minutes. As witnessed by the recent Mirai and WannaCry worms, worm attacks are real, destructive, and continue to persist. Although many worm detectors exist, most that we studied suffer from three drawbacks: none systematically consider countermeasures from worm authors, potentially causing low effectiveness against evasive worms; all focus on outbound worms leaving a network, leaving their efficacy against inbound worms entering a network unanswered; and many require bi-directional traffic to detect worms, making their placement on the Internet inflexible. We therefore revisit worm detection in this paper, while avoiding the aforementioned drawbacks of existing work. We describe our design of SWORD, a new worm detector that focuses on the fundamental behavior of worms. It includes two complementary modules to monitor connections from and to a protected network, with one module monitoring burst durations and the other ensuring quiescent periods. Via extensive experiments using both simulated worm traffic and a real-world Mirai worm trace, we demonstrate that SWORD is superior to existing detectors at not only detecting both classic and evasive outbound worms, but also inbound worms, especially those that are superspreading or surreptitious.

Index Terms—Internet Worm; Smart Worm; Worm Detection; Behavior-Based Worm Detection; Mirai Worm

1 Introduction

Worms can propagate themselves rapidly and infect millions of hosts on the Internet in just several minutes and continue to pose a severe threat to the security of the Internet. In fact, the ground for worms to spread is potentially more fertile than ever. The number of Internet-capable devices continues to rise at a stunning rate [1], and each of these devices is capable of running a diverse range of software that can be vulnerable to malicious attacks.

While there was a relatively long lull without much worm activity between the Morris worm in 1988 and the big wave of many devastating worms in the late 1990s and early 2000s (*e.g.*, Trinoo, Tribe Flood Network, Code Red, Nimda, SQL Slammer) [2], worm activity has continued in the last two decades [3]. For example, in 2008, Conficker [3] infected over a million machines, and between 2010 and 2012, Stuxnet/Duqu/Flame [4] caused devastating damage to several industrial and energy-producing facilities in several countries. More recently, the world witnessed BASHLITE [5] in 2015, along with WannaCry and NotPetya in 2017 [4].

Worms we look at in this paper are self-propagating worms, i.e., worms whose code can self-execute whenever a logic condition for spreading is met and, therefore, can replicate completely autonomously. Even though some might need manual intervention for initial activation, they do not require manual intervention from a user for continuous spreading. One of the widest spreading worms in recent memory is the Mirai worm. In 2016, the Mirai worm infected over 300,000 Internet of Things (IoT) and embedded devices all over the world. Mirai continues to "thrive" on the Internet, especially in IoT-rich environments [6]. Furthermore, due to its propagation success, Mirai's scanning techniques have been copied by a plethora of newer worms such as Hajime, IoTroop, and Mozi, among others [7].

There are many existing worm detectors which we study in this paper [8], [9], [10], [11], [12]. These detectors usually assume that worms have no knowledge of a worm detector in place, much less their configurations, and do not observe legitimate traffic in place and adjust their scanning rate accordingly to stay undetected. We call such a worm a **classic worm**. Unfortunately, a worm can also be smart and continuously evolve in order to evade existing detectors. We call such a worm an **evasive worm**. A worm detector should continue to function effectively even if a worm is *not* a classic worm, but an evasive worm.

Furthermore, all of the existing worm detectors we studied focus solely on detecting outgoing worm traffic from a protected network to the Internet, leaving their efficacy at detecting incoming worm traffic from the Internet to a protected network unknown. Even if a detector is effective against outbound worms, we cannot assume it will be effective against inbound worms. Inbound worm detection is significantly more difficult than outbound worm detection. When detecting outbound worms, a detector can be placed to observe all of the outgoing scans from an infected network. However, when detecting inbound worms, because the scans from every infected host usually target victims all over Internet, no matter where a detector is placed, it may only observe a miniscule portion of all worm scans, making it significantly harder to detect the inbound worms.

We revisit worm detection in this paper. We treat worm detection as an arms race in which a worm can be smart and evasive, and propose a new worm detector called **SWORD** (Self-propagating Worm Observation and Rapid Detection).

- *J. Li, and D. Sisodia are with the Department of Computer and Information Science, University of Oregon, Eugene, OR, 97403. E-mail: {lijun, dsisodia}@cs.uoregon.edu.*
- *S. Stafford is with Palo Alto Software, Eugene, OR, 97401. E-mail: shad@techshadow.com.*

Unlike most existing detectors, SWORD is focused on the *fundamental* behavior of worms that is hard for any worm to evade. The only truly fundamental behavior of worms is that of connecting to new destinations. Behavior-based detection systems that do not focus on this fundamental behavior can be evaded successfully by sufficiently smart worms. Furthermore, SWORD is designed to detect both outbound worms from, and inbound worms toward a protected network. It only needs to observe a very small number of connections from an infected host to detect the presence of the worm, either outbound or inbound. In addition, unlike some detectors that require bi-directional traffic to detect worms in either direction, SWORD only requires outgoing traffic to detect outbound worms and incoming traffic to detect inbound worms.

SWORD's working mechanisms are novel. It includes two main modules in detecting violations that a worm will cause in connecting to new destinations, and these two modules complement each other: If a worm does not wish to violate one module when connecting to somewhere, it will inevitably violate the other, leaving little space for a worm to breathe and forcing it to slow down or freeze.

We have designed an experimental framework for evaluating various behavior-based worm detectors. We measured the performance of SWORD and compared it to six other state-of-the-art behavior-based worm detectors, DSC [8], MRW [9], PGD [10], RBS [11], TRW [12], and TRWRBS [11]. We first evaluate the detectors on outgoing worm traffic, where all of the detectors are given bi-directional traffic as if they were deployed at the protected network's border router. We found that SWORD can not only effectively detect classic worms, but also evasive worms, and significantly outperforms all other detectors.

We then evaluate SWORD and three detectors, MRW, PGD, and RBS, on incoming worm traffic, where all of the detectors are only given uni-directional incoming traffic as if they were deployed at an upstream Internet service provider (ISP) or exchange point (IXP) to protect multiple downstream networks. We exclude DSC, TRW, and TRWRBS from comparison because they require bi-directional traffic to detect worms. We leveraged a real-world, Mirai worm traffic trace collected at a major educational IXP [13], along with a trace of background traffic collected at the same IXP [14]. Not only does SWORD outperform every other detector in terms of false negative rates, but it also outperforms them at detecting *surreptitious* worm IPs with low total scans or low incoming scanning rates, and *superspreading* worm IPs with high total scans. Compared to its competitors, SWORD detects significantly more worm IPs that make as low as 5 incoming scans, and unlike any of its competitors, SWORD can detect worm IPs with incoming scanning rates as low as 0.002 scans/s. Furthermore, SWORD detects the first incoming Mirai worm scan the quickest, therefore allowing the fewest incoming worm scans out of all the detectors before alerting of the first worm scan.

2 Background and Related Work

2.1 Worm Traffic Detection

A worm running on a host actively scans the network (or the entire Internet) that the host is connected to and looks for new victims to infect. A worm can employ a variety of scanning mechanisms, including random, local preference, sequential, permutation, topological, and hitlist scanning [15]. It infects a remote host by gaining sufficient privileges to copy itself to, and then execute itself on, the remote host.

We categorize worm detection systems into two categories: host-based and network-based. Host-based detection uses information available at the end-host, and example techniques include buffer overflow detection, correlating network data to memory errors, and looking for patterns in system calls (*e.g.*, [16], [17], [18]). But since host-based detection requires deployment on every host to detect if a host is infected, network-based detection became more desirable with less overhead to install and maintain. Network-based systems usually only need a single deployment location, such as a network gateway, to protect an entire network. Network-based detection mainly includes content-based detection and behavior-based detection. Content-based detection observes the content of network traffic to look for byte patterns that match the signature of a worm. Early content-based detectors leveraged simple statistical methods (*e.g.*, [19], [20], [21]), while recent content-based detectors leverage deep learning to detect worms (*e.g.*, [22], [23]). Behavior-based detection observes the network behavior of end hosts and identifies patterns that are indicative of the presence of a worm. Because content-based detection is less capable against zero-day or polymorphic worms and can incur a high overhead to inspect traffic payload, we focus on behavior-based detection in this paper.

Existing behavior-based worm detection has focused on various types of traffic behaviors, including: how the outgoing connections from a host correlate to the incoming connections to that host, how the connection failure patterns of a host deviate from normal, and what a host's pattern of visiting destinations looks like. As we will need to compare SWORD against state-of-the-art behavior-based worm detectors, we now summarize these detectors below.

DSC [8] detects a worm by correlating an incoming connection on a given port with subsequent outgoing connections on that port. If the outgoing connection rate exceeds a threshold established during training, the alarm is raised.

TRW [12] identifies a host as worm infected if its attempts to connect to new destinations result in a high rate of connection failures. The basic idea is that a worm-infected host that is scanning the network randomly will have a higher connection failure rate than a host engaged in legitimate operations.

The multi-resolution approach [9], which we refer to as MRW, supposes that when there is no worm, the growth curve of the number of distinct destinations over time is concave, but not so when a worm is present since worm scanning will lead to many destinations. This hypothesis can be leveraged by monitoring over multiple time windows with different thresholds for each window. If the number of new destinations for a host within a given window exceeds the threshold, an alarm is raised.

The Protocol Graph detector [10], which we refer to as PGD, is targeted at detecting slowly propagating hitlist or topologically aware worms. It works by building protocol-specific graphs where each node in the graph is a host,

and each edge represents a connection between two hosts over a specific protocol. It assumes that during legitimate operation over short time periods, the number of hosts in the graphs is normally distributed and the number of nodes in the largest connected component of each graph is also normally distributed. During a worm infection, however, both numbers will become abnormal, thus indicating the presence of a worm.

RBS [11] measures the rate of connections to new destinations, similar to MRW. It assumes that a worm-infected host contacts new destinations at a higher rate than a legitimate host does. It measures this rate by fitting the inter-arrival time of new destinations to an exponential distribution.

TRWRBS [11] combines the TRW and RBS detectors into a unified scheme, and observes both the connection failure rate and the first contact rate. It performs sequential hypothesis testing on the combined likelihood ratio to detect worms.

All aforementioned detectors are focused primarily on classic worms, without considering the countermeasures that a smart, evasive worm may employ. Research in [24] have also evaluated and compared their performance, but only against classic worms. Although a detector may appear to perform well by only considering classic worms, their performance against sophisticated evasive worms remains to be seen.

Furthermore, all aforementioned detectors are focused solely on detecting outbound worms from a network. While each detector may be applied to also detect inbound worms toward a network without significant changes, each detector was evaluated only against outbound worms.

Lastly, DSC, TRW, and TRWRBS require bi-directional traffic to detect worms. DSC needs to observe both incoming and outgoing traffic in order to correlate the two, TRW needs to observe incoming traffic in order to determine which outgoing connections led to connection failures, and TRWRBS leverages TRW and therefore also needs to observe incoming traffic. In some scenarios in the real world, the requirement of bi-directional traffic may be impossible to meet, due to the deployment location of a detector, hardware limitations, privacy considerations, and other constraints.

2.2 Content-Agnostic Traffic Analysis

The aforementioned behavioral-based worm detectors, including SWORD, are content-agnostic because they do not need to observe the content of the network traffic. There is a plethora of content-agnostic traffic anomaly detection approaches in literature, especially in two areas related to worm detection: bot detection and DDoS detection.

While somewhat similar, bot detection differs from worm detection in that typical bot detection approaches attempt to detect communication between bots and their command and control (C&C) servers, instead of detecting scanning behavior of the bots. Content-agnostic bot detection approaches leverage flow-level information, instead of deep packet inspection, to identify key features of C&C communication and develop detection frameworks (*e.g.*, [25], [26]). In many cases, worms are used to create botnets that ultimately launch DDoS attacks. Over the last couple decades, content-agnostic DDoS detection has been a fertile ground for network security research (*e.g.*, [27], [28]).

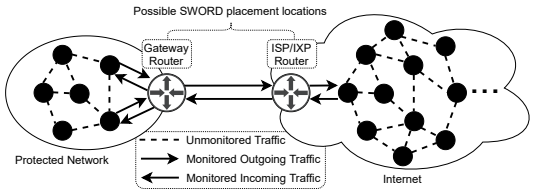


Fig. 1: Placement of the SWORD detector.

Some of the same shortcomings that apply to the previously investigated behavioral-based worm detectors also apply to the content-agnostic anomaly detection approaches in bot and DDoS detection. All of the investigated approaches require a relatively comprehensive view of the network in which they are deployed, which includes being able to observe bi-directional traffic. While content-agnostic approaches claim to be easier to deploy in the real-world due to not needing network traffic content, the requirement of comprehensive input data may render them infeasible for many networks on the Internet.

3 The SWORD Detector

3.1 Placement of SWORD

SWORD can detect both outgoing and incoming worm traffic from and toward a protected network. In other words, to detect outgoing worm traffic, SWORD needs to be placed where it can monitor the network's outgoing traffic; *no* incoming traffic is needed. To detect incoming worm traffic, it needs to be placed where it can monitor the network's incoming traffic; *no* outgoing traffic is needed.

A SWORD detector can be placed either at or away from a protected network. As depicted in Figure 1, a typical deployment position of SWORD is the gateway of a protected network where SWORD can monitor all of the outgoing and incoming traffic. Alternatively, SWORD could also run at an ISP/IXP that is *en route* of the outgoing traffic from and/or incoming traffic to the network. If the network is single-homed (*i.e.*, connected to the Internet with just one ISP/IXP) and SWORD is deployed at its direct ISP/IXP, SWORD can monitor all the outgoing and incoming traffic. However, if the network is multihomed (*i.e.*, connected to the Internet with more than one ISP/IXP), SWORD may not see all of the outgoing or incoming traffic and thus only detect outgoing or incoming worm traffic traversing the ISP/IXP where it is installed. This is also true if SWORD is deployed at an ISP/IXP multiple hops away from the protected network (regardless if the network is single-homed or multihomed). A distributed version of SWORD that runs multiple instances of SWORD at more than one location can also be deployed to collectively monitor all the outgoing/incoming traffic; in this work we focus on the single-instance version of SWORD.

To monitor outbound worms departing from a protected network, it is often preferable to deploy SWORD at a location where it can observe *all* the outgoing traffic, such as the network gateway or its direct ISP/IXP if the network is single-homed, so that the network can minimize its liability

of leaking worms to the Internet. Moreover, if SWORD is deployed at a location not able to monitor all the outgoing traffic, the worms that have infected the network may also learn the location of SWORD, and bypass it such that the protected network may not even know the presence of the worm.

To monitor inbound worms toward a protected network, SWORD can also be deployed at the network gateway or its direct ISP/IXP if the network is single-homed such that it can see all the incoming traffic to maximize the detection of all worm traffic. However, as inbound worms can travel toward the network along multiple incoming paths, even if SWORD is only deployed on one of the incoming paths, such as when SWORD is deployed at one of the protected network's ISP/IXPs, SWORD will still be able to detect the presence of the worm, for two reasons: First, even though in such a deployment scenario SWORD may only see incoming traffic to the network, SWORD only needs uni-directional traffic to detect worms. Second, although SWORD is not deployed on every incoming path, worm traffic usually appears on every incoming path toward the network. Running SWORD at an ISP/IXP can be more cost efficient compared to running SWORD at each network downstream, but far more difficult, as explained in Section 1, especially when only partial incoming traffic is observed.

3.2 BDD: Preventing Fast Scanning via the Burst Duration Detector

The behavior of contacting new destinations seeking new victims to infect simply cannot be avoided by a worm that is looking to propagate. So, to detect a worm one should look for anomalies in the rate at which a host contacts new destinations, *i.e.*, the rate of *first-contact connections*. The key is then to determine whether or not a host is making first-contact connections at a rate faster than usual.

Two previous detectors relied on heuristic of this flavor: the MRW detector and the RBS detector. However, they both have their deficiencies. The MRW detector counts the number of first-contact connections in a series of time windows of different length, but it only uses a relatively small set of windows, typically fewer than 10. An intermediate window size might produce a detection window that would detect a worm more quickly than the bigger or smaller sizes in use, but due to the limited number of windows, MRW cannot take advantage of this. RBS, on the other hand, computes a threshold for every different window size, and it uses the number of connections instead of time to describe the window. However, it suffers from sub-optimal thresholds, and thus a poor performance even against classic worms (Section 5.1). RBS attempts to fit a single curve to the distribution of inter-connection intervals and uses this curve to generate the thresholds, but in practice the distribution does not map well to a single curve.

BDD avoids the drawbacks in the MRW and RBS approaches. Rather than using a small number of time windows of different sizes like MRW, it uses RBS's method of creating a window for every different size of connection burst. Moreover, BDD derives a threshold for every burst size (from a two-connection burst size to a maximum-connection burst size). We introduce a training process, during which we measure multiple different durations observed for each burst size and use the minimum duration observed for each burst size to determine the threshold for a burst of that size. Different from RBS using a single curve to derive thresholds, this process allows for a more complex distribution of inter-connection interval times for connections in a burst, thus obtaining more accurate thresholds. As a result, BDD has the advantages of supporting a large number of window sizes and obtaining an accurate threshold for every burst of a different size.

Another advantage of BDD is that even if a worm only makes a small number of connections, these connections will be verified against the thresholds for bursts of a small size, and if any threshold is violated, BDD can detect the worm. This advantage is especially useful in detecting inbound worm traffic from a worm-infected host toward a protected network, since the host, while scanning everywhere on the Internet, may only launch a small number of worm connections toward the network.

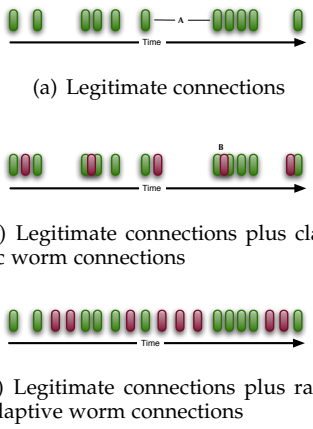
The potential drawback to this new method is greater overhead for storing different thresholds and greater computational requirements for examining a recent connection history to determine if it violates any of the thresholds. However, a truism is that computational power and storage space are constantly increasing, and this additional load is of a less concern.

3.3 QPD: Ensuring Quiescent Periods via the Quiescent Period Detector

A normal host will exhibit regular *quiescent* periods where it does not make any first-contact connections. In other words, legitimate traffic is typically bursty, with first-contact connections occurring in groups and quiet periods between them. Figure 2(a) shows an example pattern of legitimate connections. Point **A** in the figure shows a quiescent period with no worm traffic, followed by a burst of connections.

After a worm infects a host and tries to spread itself, if it scans at a fixed rate, it will make connections during the middle of a legitimate burst, which will raise the overall observed connection rate from the host. Figure 2(b) shows the legitimate traffic with the addition of classic worm traffic. Point **B** indicates a spot of increased connection rate due to the worm connections adding to the burst of legitimate traffic. If BDD is in place, it can detect the worm.

The worm, however, could be adaptive and avoid this additive effect. Specifically, the worm can dynamically adjust its first-contact rate so that it is always lower than the detection threshold. If the host makes bursts of legitimate first-contact connections, the worm can simply slow down to keep from adding too many of its connections to the legitimate connections, thus avoiding exceeding the detection threshold. When the host is otherwise idle, however, as long as the worm does not exceed the BDD thresholds, it is then free to make first-contact connections. Figure 2(c) shows the legitimate traffic with an adaptive worm overlaid. By scanning mostly when the host is in the middle of a quiescent period, the adaptive worm avoids having a scanning rate greater than the legitimate traffic, even at a higher scanning rate than the classic worm (with eight worm scans instead of five in Figure 2(b)).



(a) Legitimate connections

(b) Legitimate connections plus classic worm connections

(c) Legitimate connections plus rate-adaptive worm connections

Fig. 2: Examples of observed connections over time.

Preventing or limiting this adaptive behavior of worms would then help to reduce the achievable scan rate of a worm, and is the basis for QPD. Basically, if a host does not display quiescent periods as it typically does, and has been "active" for overly long, QPD then determines that the host is infected by a worm that is scanning the network.

QPD thus detects worms by measuring the duration of active periods during a training phase. An active period is defined as the duration of a period during which first-contact connections happen with *no more than* the specified quiescent period between them. QPD uses a series of different quiescent periods. For every quiescent period size, it measures the mean and standard deviation of all the active periods that are separated by a quiescent duration of at least that length. These values are used to generate a threshold duration for active periods, which is the mean plus β times the standard deviation. β can be tweaked for different environments to fix the false positives at a specific value. If a host has an active period exceeding the threshold duration for any of the quiescent period, it is likely infected with a worm. For example, we can apply QPD to Figure 2(c) where the host is active all the time and does not exhibit any quiescent period at all to detect the presence of the worm.

Note that, similar to BDD, QPD also has the advantage that it is sensitive to worms that only make a small number of connections, which, again, is particularly useful in detecting incoming worm traffic despite that there may be only a small number of scans from a worm IP. Among the different quiescent periods QPD uses, some of them can be extremely short, and the active period based on a short quiescent period will also be short and contain only a small number of connections. Therefore, even if a worm only makes a small number of connections, it could cause certain active periods to exceed their threshold values, causing QPD to detect the worm.

3.4 Clustering

Existing behavior-based detection systems employ the same threshold for all hosts in a protected network or on the Internet. This is a poor choice because hosts show widely divergent behaviors. As more devices (*e.g.*, IoT devices) connect to the Internet, they also come with even more divergent behaviors [29]. Desktop computers used primarily for web surfing make connections in a different pattern than a department email server would, for example. If a desktop computer started making connections at the same rate as the email server, it is likely an anomalous event and something strange must have happened to that computer. But if the desktop computer applies same thresholds as the email server does, its behavior would not appear to be anomalous because those thresholds must allow it as normal behavior to avoid constantly flagging the email server as infected.

We applied existing clustering techniques to automatically categorize the hosts such that different thresholds can be applied to different groups of hosts. We examined a range of clustering techniques, behavior characteristics to cluster against, and number of clusters to create. We have found that using k-means clustering to separate the hosts into groups allows us to improve overall performance. In our current design we cluster based on a single feature of the hosts, the number of destinations contacted during a training period.

3.5 Design of SWORD

We have combined the above principles into a new worm detector, *i.e.*, SWORD. It uses the BDD and QPD detectors outlined above, and declares a host to be infected with a worm when either BDD or QPD raises an alarm. SWORD observes legitimate network activity for a period of time to cluster hosts into groups and generate thresholds for each cluster.

The co-existence of BDD and QPD makes it extremely hard for a worm to avoid being caught. If a worm wants to escape BDD but still makes new connections, it cannot shorten the duration of a burst of any size; it will then have to lengthen active periods, but doing so will get it caught by QPD. On the other hand, if a worm wishes to escape QPD while still making new contacts, it then has to ensure the quiescent periods; it will then have to insert its connections into active periods, which however will cause certain connection bursts to have a shorter duration than permitted, thus triggering the alarm from BDD. Therefore, this combined, collective detection of SWORD captures the fundamental behavior of worm detection, preventing a worm from quickly spreading to many destinations.

4 Experiment Methodology for Outbound Worm Detection

4.1 Procedure Overview

Our objective is to evaluate and fairly compare the performance of different detectors, including SWORD, against outbound worms with various different parameters. We developed a custom testing framework that allows us to easily set up an evaluation environment with both background traffic and worm traffic, plug in any behavior-based detector, and run the same set of experiments for each detector against the same traffic to measure its performance. We implemented SWORD in this framework, as well as all detectors described in Section 2, *i.e.*, DSC, TRW, RBS, MRW, PGD, and TRWRBS. We place every detector, when evaluated, at the gateway of the protected network. This placement ensures every monitor to be able to observe all

the outgoing traffic, as desired for monitoring outbound worms (see Section 3.1); this placement also enables DSC, TRW, and TRWRBS to monitor all the incoming traffic, as they need complete bi-directional traffic.

For each worm detector, including SWORD, we first run it against classic worms. We measure its accuracy (*i.e.*, false positives and false negatives) and detection latency at different scanning rates. We then run every detector against evasive worms, and see how resilient a detector is against evasive worms. We describe evasive worms in Section 4.2, and detail the metrics and parameters for measuring against both classic and evasive worms in Section 4.3.

We run each detector in four distinct environments: *campus*, *enterprise*, *department*, and *wireless*. Every environment includes background traffic from a real source, and worm traffic with a variety of worm scanning strategies generated using the GLOWS worm simulator [30] that is tailored to that environment. Note, we ensure the worm traffic is unbiased toward SWORD. As the traffic behavior in each environment is different, such as the durations of traffic bursts and durations of active periods, SWORD may demonstrate different performance under different environments. We detail them in Section 4.4.

4.2 Evasive Worms

For a worm to evade detection, it must know the underlying details of the detector being used, then leverage its capabilities to adjust its behavior in order to avoid triggering detection. We define several terms to refer to capabilities of evasive worms. A worm with no knowledge of the legitimate network traffic on an infected host is said to be **blind**, whereas if it can observe the traffic it is **perceptive**. A worm that does not know the parameters of the detector deployed against it is described as **speculative**, whereas one that knows the actual deployed parameters is said to be **informed**. We consider all permutations of these capabilities.

An evasive worm against SWORD must ensure that it has sufficient quiescent periods to evade QPD, while also limiting its bursts of connections to avoid triggering BDD. The combination of these two mechanisms puts significant constraints on the ability of the worm to scan. More specifically, the worm runs internal versions of both the QPD and BDD detectors. For every scan to initiate, it first checks to see whether the scan will violate any of the QPD constraints. If it will, the worm waits long enough to end the current active period for the QPD constraint in question. After eliminating QPD as a constraint, it checks the BDD durations to ensure that the BDD detector will not be triggered either. Note, while an evasive worm may be able to see legitimate connections **already** sent from its infected host, it cannot predict **future** legitimate connections from the same host. Based on the legitimate connections seen so far, the worm may decide it can make one or multiple worm connections that would not cause any threshold to be violated, but the subsequent legitimate connection(s) could.

We also implemented a different type of evasive worm against every other type of worm detector we evaluate. The evasive worm against DSC adds a delay between infecting a host and the beginning of scanning from that host to avoid any causality connection; the evasive worm against TRW contacts a list of known hosts to avoid connection failures; the evasive worms against MRW and PGD both scan at the maximum sustained rate that will not be detected; and the evasive worm against TRWRBS is a combination of the evasive worms against TRW and RBS (we do not consider evasive worms against RBS because RBS does not even perform well against classic worms, as shown in Section 5.1).

4.3 Metrics and Parameters

For each detector in each environment, we first run it against benign traffic with no injected worm activity. The **false positive rate** is the number of hosts misidentified as infected per hour. We then run 16 experiments for every permutation of the worm parameters (*e.g.*, we run 16 experiments to measure a random-scanning worm at every scanning rate). Each experiment consists of running the detector for 10 minutes of the experiment trace to warm up the connection histories, then injecting the simulated worm traffic into the trace, and running until either an hour has elapsed or the worm is detected. Each of the 16 experiments that we run for a given set of worm parameters has a different host in the protected network being infected first and uses a different random seed. The **false negative rate** is then the percentage of experiments where the worm is not detected, and the **detection latency** is the mean number of worm connections that have left the network at detection time.

For each evasive worm we vary a parameter ζ between zero and one that controls its aggressiveness of scanning. A value of zero means that the worm will generate traffic so that it has a 0% chance of being caught, which may mean that it may not scan at all. A value of one means that the worm's scanning traffic is the most aggressive possible without being detected (such as by not surpassing the worm detection thresholds) **when there is no legitimate traffic present**. Note, an infected host may also generate legitimate traffic; when the worm traffic from the host is not alone but interleaved with the legitimate traffic, a worm with a load factor of 1 or less may still be detected. Therefore, a worm's ability to evade detection is based not just on the load factor, but on a combination of the load factor, the worm's distribution of scanning traffic over time, the legitimate traffic generated by the infected host, and the worm's capabilities.

We run each evasive worm once in each environment for 1 hour for each of 16 different randomly selected first infected hosts, for 10 different values of ζ in the range [0.1–1.0], and with an upper bound of scanning rate at 10 scans per second.

We use three metrics to evaluate the success of an evasive worm. As we vary the load factor, we measure the worm's ability to evade detection: its **evasion rate**. This represents the percentage of experiments where the evasive worm is *not* detected by the worm detector in place. Clearly, the evasion rate of the worm is equivalent to the false negative rate of the corresponding worm detector.

The second metric is the **effective scanning rate**. This is the average number of worm scans per second the evasive worm is able to make during the one hour experiment for a given environment and value of ζ. The higher the value of ζ, the faster the evasive worm will scan, thus increasing its effective scan rate (but also reducing the evasion rate).

An evasive worm author's goal is to scan as quickly as possible while maintaining a high chance at evasion. As the load factor of a worm increases, its scanning rate also increases, but its evasion rate decreases. By choosing a minimum evasion rate, we then can find the **maximum effective scanning rate**. For this experiment we choose the minimum evasion rate to be 0.90, meaning the worm can survive if the false negative rate of detection is 90% or higher. The maximum effective scanning rate is the ultimate determination of a worm detector's effectiveness. The lower the maximum effective scanning rate allowed, the more effective a detector is. This single metric is the best metric for comparing detectors, as it reveals the damage that a worm can cause without being detected.

4.4 Evaluation Environments

The *campus* environment is built from a trace collected at the border of Auckland University [31]. It contains over a month of traffic from the entire university with two /16 and several /24 networks. We randomly select 200 hosts and construct an environment using traffic to and from those hosts, where the training and experiment segments each contain approximately 25,000 connections.

The *enterprise* environment is built from a trace collected at LBNL [32]. Heavy scanners were removed from the trace before it was released. It has 139 active hosts and the training and experiment segments each also contain roughly 25,000 connections.

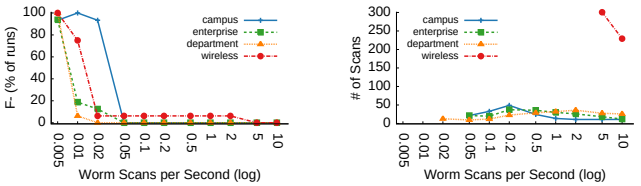
The *wireless* and *department* environments are built from traces collected at the University of Massachusetts [33]. The department environment is built from a trace capturing all traffic to and from the wired computers in the CS department. It has 92 active hosts and approximately 30,000 connections in each training or experiment segment. The wireless environment comes from a trace capturing all wireless network traffic from the university. It has 313 active hosts and approximately 120,000 connections in each segment.

5 Outbound Worm Detection Evaluation

5.1 Performance of SWORD vs. Classic Worms

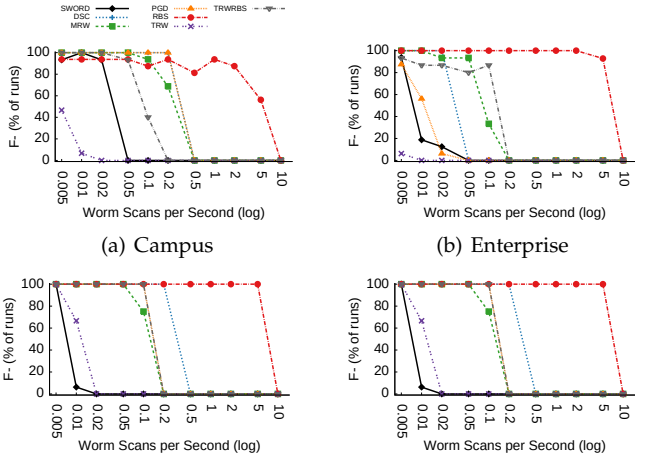
We report the following results using *random-scanning worms* as the classic worms. Our experiments show that in detecting other classic worms of different scanning types (such as local-preference worms, topological-scanning worms, or hitlist worms), SWORD's performance is similar to its performance against random-scanning worms, whereas the performance of other detectors is similar to or worse than their performance against random-scanning worms.

Figure 3(a) shows the false negative rate that SWORD achieved against a classic worm. The worm was detected at a scanning rate of 0.05 connections per second in every scenario except for a single host in the wireless environment. To make the direct head-to-head comparison between SWORD and other detectors easier, Figure 4 further plots SWORD and the other detectors all on the same graph. All these detectors are adjusted to have the same false positive rate (two falsely identified hosts per hour). In the campus environment (Figure 4(a)), we can see that compared to



(a) False negatives (b) Detection latency

Fig. 3: False negative and detection latency of SWORD when running against classic worms.



(a) Campus (b) Enterprise

(c) Department (d) Wireless

Fig. 4: False negatives for every detector when running against classic worms.

SWORD, the TRW detector is able to detect some worms at slightly slower scanning rates. However, it is the only detector that is able to do so, and it does not detect 100% of the infections with a slower scanning rate. The enterprise environment shows similar results (Figure 4(b)), again with TRW showing slightly better sensitivity and this time PGD just barely beating SWORD on worms at 0.005 and 0.02 scans per second. The other two environments, however, show SWORD with the best sensitivity, detecting worm infections at slower scanning rates than any other detector (Figures 4(c) and 4(d)).

Figure 3(b) shows the detection latency of SWORD. In the campus, enterprise, and department environments, the average detection latency is under 40 scans for all worm scanning rates but one, where the average detection latency is under 50 scans. We do not see much of the latency performance in the wireless environment, because we only plot detection latency for combinations of environment and scanning rate where the worm was detected in 100% of the experiments. However, if we were to relax our restriction and show the detection latency for those scenarios at each scanning rate where the worm was detected in the wireless environment, we would see that the latency is under 67 for all scan rates under 0.2, and under 327 for all worm scan rates. In Table 1, we present the average detection latency across all scanning rates where the worm was detected for each detector and environment (if a detector is faster than SWORD its latency is underlined). Clearly, in most cases SWORD is faster than the other detectors.

Overall, in detecting classic worms, the only two detec-

TABLE 1: Average detection latency for all detectors.

Detector	Campus	Enterprise	Department	Wireless
SWORD	21.73	24.97	22.99	264.94
DSC	2.00	22.00	19.00	15.93
MRW	28.88	51.70	43.64	1014.16
PGD	93.80	28.11	25.81	621.76
RBS	17.36	4.25	26.44	349.53
TRW	4.23	11.13	24.75	49.93
TRWRBS	57.97	30.39	58.66	167.95

tors that *sometimes* beat SWORD are PGD (in one environment only) and TRW (in two environments). SWORD has a lower average detection latency than PGD in all environments here, including a latency of less than half in the wireless environment. The TRW detector has a lower detection latency in three of the four environments. However, a clever worm can evade the TRW detector by employing known neighbors to befuddle the detector. In the next section we show that the SWORD detector is dramatically better once evasive worms are taken into account. Against all other detectors, SWORD has either better sensitivity or detection latency, and in many cases both.

5.2 Performance of SWORD vs. Evasive Worms

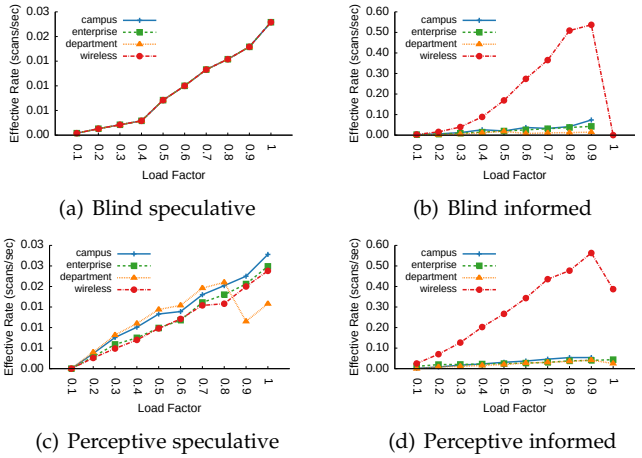
5.2.1 Effective Scan Rate and Evasion Rate of Evasive Worms against SWORD

The blind speculative version of the worm cannot achieve an effective scan rate of greater than 0.03 scans per second in any scenario (Figure 5(a)). In the department environment, when the load factor is 1, a scan rate even this low still gives an evasion rate of 0% (Figure 6(a)). The perceptive speculative version of the worm does not improve the effective rate at all (Figure 5(c)), but does improve the evasion rate in all but the department environment (Figure 6(c)). In the wireless environment, the informed versions of the worm are able to achieve an effective rate nearly 10x greater than the speculative worms were able to (Figures 5(b) and 5(d)). However, the speculative worms (Figures 6(a) and 6(c)) overall achieve better evasion rates than the informed worms (Figures 6(c) and 6(d)). This is because the speculative worms, while uninformed of the detection parameters, scan at a much lower rate to evade detection, as shown in Figure 5. Overall, SWORD works effectively against evasive worms and can limit both the effective scanning rate and evasion rate of evasive worms.

5.2.2 Maximum Effective Scan Rate of Evasive Worms Against SWORD and Existing Detectors

The best evaluation of a detector is the maximum effective rate achieved by the evasive worm while running less than a small chance (we use 10%) of being detected. In Figure 7, we plot the maximum effective rate achieved by the evasive worms against respective detectors.

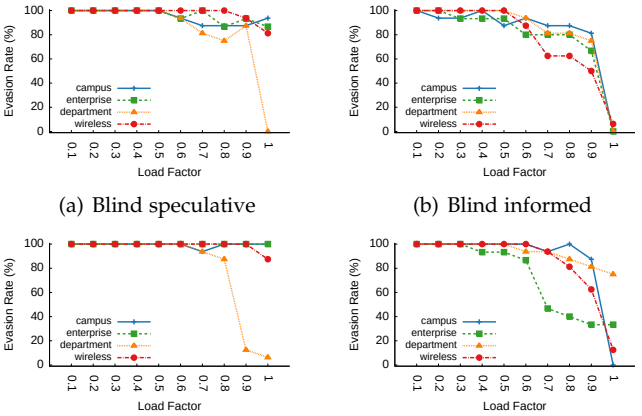
For the campus and department environments (Figures 7(a) and 7(c), respectively), the benefits of the SWORD detector are pronounced. In the campus environment, SWORD beats all other detectors by at least a factor of 2. In this environment, no other detector came close to limiting the maximum effective rate of the evasive worms as well as SWORD did. In the department environment, SWORD



(a) Blind speculative (b) Blind informed

(c) Perceptive speculative (d) Perceptive informed

Fig. 5: Effective scanning rate of evasive worms vs. SWORD.



(a) Blind speculative (b) Blind informed

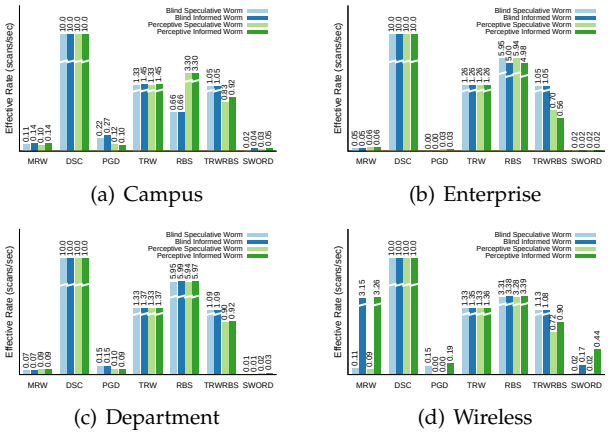
(c) Perceptive speculative (d) Perceptive informed

Fig. 6: Evasion rate of evasive worms vs. SWORD.

outperforms all other detectors by at least a factor of three for all evasive worm varieties.

For the enterprise and wireless environments (Figures 7(b) and 7(d), respectively), PGD is the only detector that outperforms SWORD in some scenarios, while all the other detectors continue to perform significantly worse than SWORD. And even when PGD outperforms SWORD, SWORD's performance is fairly good and stays very close to PGD's.

Therefore, among all four environments that totally include 16 evasive worm scenarios, other than PGD, SWORD outperforms every other detector in all 16 scenarios. This includes TRW that slightly beats SWORD against the classic worms in two out of four test environments; in 13 scenarios an evasive worm can scan 60+ times faster when it's against TRW rather than SWORD. For PGD, we can see SWORD outperforms PGD in 11 evasive worm scenarios while PGD outperforms SWORD in 5 scenarios. To further compare PGD vs. SWORD, we therefore introduce the "aggregated maximum effective rate" of evasive worms against a detector, which is the average maximum effective rate of evasive worms over all 16 scenarios. We found SWORD's and PGD's aggregated maximum effective rate are 0.06 and 0.10, respectively, indicating on average an evasive worm can scan about 67% faster when it is against PGD rather than SWORD.



(a) Campus (b) Enterprise

(c) Department (d) Wireless

Fig. 7: Maximum effective rate of evasive worms.

5.3 Summary

We have shown that in detecting outgoing worm traffic from a network, the SWORD detector significantly outperforms all other detectors. The TRW detector does perform slightly better than SWORD against classic worms in two out of the four environments, but against evasive worms it was outperformed in every scenario and outperformed by a factor of more than 60 in 13 scenarios. This significant superiority against evasive worms offsets any minor advantage TRW has over SWORD against classic worms. The PGD detector does outperform SWORD in 5 of the 16 evasive worm scenarios (four evasive worm types by four environments), but is dominated in the remaining 11 scenarios. Overall PGD is completely outperformed by SWORD against classic worms and it allows an evasive worm to scan about 67% faster than SWORD. The only other detector to come close is the MRW detector, which is consistently outperformed by SWORD against classic worms, and is soundly beaten in every scenario against evasive worms. None of the other detectors present even an appreciable level of competition.

5.4 Limitations and Open Issues

The evasive worm against every detector could be improved. For example, the current design of the evasive worm against TRW assumes that the worm can always find a list of known hosts to contact, but sometimes it may be impossible. Also, the experimentations assume that the training is reliable, but if the background traffic is infected by a worm, it can introduce noise into the results.

6 Experiment Methodology for Inbound Worm Detection

6.1 Procedure Overview

Our objective is to evaluate and fairly compare the performance of different detectors, including SWORD, against inbound worms. We chose to place each detector at a protected network's upstream ISP/IXP, a challenging deployment scenario where the detector is not guaranteed to be able to access all the incoming traffic, to study how effective SWORD and the other detectors are under such a realistic scenario. Moreover, such placement does not guarantee a detector to be able to access both incoming and outgoing traffic of a protected network either, a requirement for DSC, TRW, and TRWRBS to detect worms as explained in Section 2; we thus exclude them from comparison for inbound worm detection. The incoming traffic at the ISP/IXP is composed of (1) background traffic that we collected at a major IXP called FRGP [14] that does not contain worm traffic, and (2) the real-world Mirai worm traffic also collected at FRGP [13].

Different from the evaluation against outbound worms where we evaluated both classic worms and evasive worms, we do not assume inbound worms can be evasive. While an outbound worm, as it originates from a network already compromised, may have the knowledge of the legitimate traffic of the network and/or the parameters of the detector in place to become evasive, we do not expect the same knowledge for inbound worms.

For each worm detector, we first train it on 1-hour worth of background traffic. Specifically, we set the threshold values of each detector as aggressive as possible *without* triggering a false positive during the 1-hour period. Then on a different 1-hour period of background traffic that is mixed with worm traffic, we test each detector and measure its accuracy (total number of detected worm IPs, false positive rate, and false negative rates) and latency.

6.2 The Mirai Worm

We use the Mirai worm as a case study for this evaluation. As Mirai's code will continue to be the basis of future worms [7], the detection of Mirai traffic should be considered a baseline for worm detectors striving to achieve success in today's Internet. Hosts infected by the Mirai worm are diverse: Some only make a very small number of scans during a period or scan with an extremely slow scanning rate, which we call running a **surreptitious** worm; some scan a large number of targets, which we call running a **superspreading** worm. The ability to detect surreptitious worms potentially can alert a protected network of a worm infection otherwise unnoticed, while superspreading worms are clearly too dangerous not to detect.

6.3 Metrics and Parameters

The key metrics we will focus on in this evaluation are **number of Mirai IPs detected**, **false positive rate**, **false negative rate**, and **detection latency**. We denote TP as the number of Mirai IPs correctly detected as Mirai, FP as the number of legitimate IPs incorrectly detected as Mirai, TN as the number of legitimate IPs correctly detected as legitimate, and FN as the number of Mirai IPs incorrectly detected as legitimate. Then the number of Mirai IPs detected is $TP + FP$ and we define the false positive rate as $\frac{FP}{FP+TN}$ and false negative rate as $\frac{FN}{FN+TP}$. The detection latency is in terms of the number of worm scans allowed before detection. A Mirai IP's detection latency is thus the number of incoming scans the Mirai IP made before it is detected. On the other hand, the Mirai worm is detected when detecting the first Mirai IP, so the detection latency of the Mirai worm is the total number of all incoming worm scans made by all the Mirai IPs before detecting the first Mirai IP.

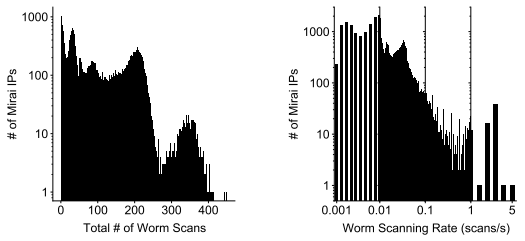
We also define two parameters associated with each Mirai IP: W that is the number of incoming worm scans made by the Mirai IP during the testing period and R that is

the worm scanning rate of the Mirai IP, which is particularly useful when we inspect slowly scanning, surreptitious Mirai IPs. If a Mirai IP is detected, say after making m worm scans, assuming its first scan is at time t_1, the m-th scan is at time t_m, $R = \frac{m}{t_m - t_1}$; otherwise $R = \frac{W}{t_W - t_1}$ where t_W is the time of the last scan from the Mirai IP.

6.4 Evaluation Environment

The Mirai traffic trace are Argus flow records and was collected at FRGP during four days in September of 2016 (8th, 9th, 10th, 12th) [13]. These days coincide with Mirai's first major growth phase in early September 2016 [34]. The background traffic was also collected at FRGP in September 2016, across the entire month [14]. Both are uni-directional traffic collected at a single router at FRGP towards its downstream customers; as such traffic is not guaranteed to be *all* the incoming traffic toward a customer, it creates a more challenging deployment scenario.

We trained the detectors on 1-hour worth of FRGP background traffic (∼3 GB of records) collected on 9/8/2016 at 6:20 PM—7:20 PM (MST). We verified that no incoming Mirai connections are present during this time window (there were no incoming connections to destination ports 23/2323 from random source ports). Once trained, the detectors were tested on 1-hour worth of Mirai scanning traffic mixed with FRGP background traffic (totaling ∼8 GB), both collected at FRGP on 9/9/2016 at 6:20 PM—7:20 PM (MST).



(a) W.r.t. total number of worm scans (W)

(b) W.r.t. worm scanning rate (R)

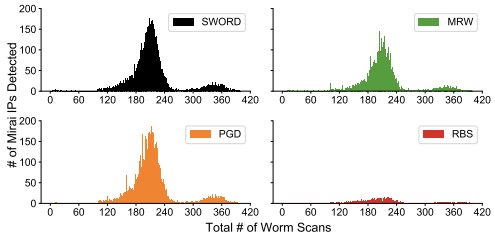
Fig. 8: Number of Mirai IPs in the 1-hour testing period.

There was a total of 45,291 unique Mirai IPs present in the 1-hour testing period, along with 45,903 unique legitimate source IPs making incoming connections to FRGP customer IPs. Figures 8(a) and 8(b) show the total number of Mirai IPs (log scale) for different W and R values (defined in Section 6.3) in the 1-hour testing period, respectively. From Figure 8(a) we can see that a large portion of the Mirai IPs made fewer than 50 connections to FRGP customer IPs. Detecting such IPs is almost impossible for any detector. Figure 8(b) shows a large number of worm IPs had extremely slow scanning rates between 0.001 and 0.030 incoming scans per second, also almost impossible to detect.

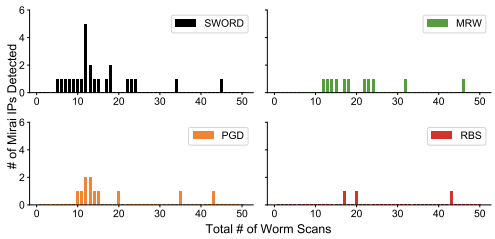
7 Inbound Worm Detection Evaluation

7.1 Total Number of Mirai IPs Detected

We list the total number of Mirai IPs detected by each detector in the 1-hour testing period: SWORD: **8882**, MRW: **5797**, PGD: **9100**, and RBS: **1041**. By setting the threshold values of each detector *without* triggering a false positive during training, we found during testing that every detector's false positive rate is also 0, and therefore, all detections are true positives. PGD detected slightly more Mirai IPs than SWORD, while MRW and RBS detected around 35% and 88% less Mirai IPs than SWORD, respectively.



(a) # of Mirai IPs detected w.r.t. W



(b) # of Mirai IPs detected w.r.t. to $W \leq 50$

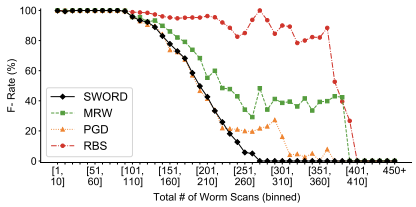
Fig. 9: # of Mirai IPs detected.

Figure 9(a) provides us with a more detailed view of the number of Mirai IPs detected by grouping the Mirai IPs based on their number of scans, *i.e.*, their W values as defined in Section 6.3. First, for all of the detectors, most Mirai IPs that were detected made anywhere from around 120 to 250 scans (120≤W≤250). Second, there is a dip in detections between around 250 and 300 scans (250≤W≤300); this is because there was only a small number of Mirai IPs (less than 10) that fell within this range, as shown in Figure 8(a). Third, although there were many Mirai IPs whose number of scans were relatively low with $W \leq 120$, it was difficult to detect them; without enough incoming Mirai traffic, a detector has difficulty in distinguishing between incoming legitimate and malicious connections.

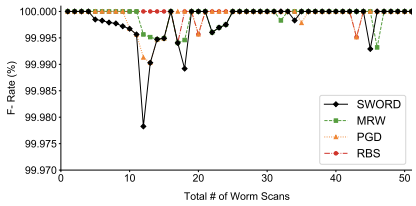
Nonetheless, if we take a closer look at surreptitious Mirai IPs with a low number of incoming scans (e.g., no more than 50 scans or $W \leq 50$), as shown in Figure 9(b), we see that SWORD significantly outperformed the other three detectors, and shows that it has the ability to detect some worm IPs with very low number of scans. While other detectors did not detect any Mirai IPs making less than 10 scans in the entire 1-hour period, SWORD was even able to detect a Mirai IP that only made 5 scans in the period. The main reason for SWORD's superior performance here is that even with an extremely low number of scans, some surreptitious worm IPs will exhibit bursty behavior, and will therefore be caught by SWORD's BDD module.

7.2 False Negative Rate

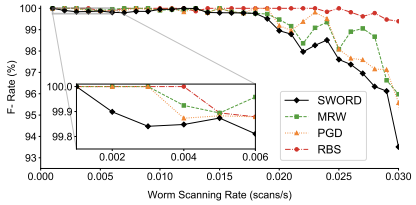
Figure 10(a) shows a granular view of the false negative rates for each detector. We first binned Mirai IPs based on their total number of scans, *i.e.*, their W values, at 10-scan increments and then calculated every detector's false



(a) False negative rates w.r.t. W



(b) False negative rates w.r.t. $W \leq 50$



(c) False negative rates w.r.t. $R \leq 0.03$

Fig. 10: False negative rates.

negative rate for each bin. Overall, SWORD outperformed MRW, PGD, and RBS. Specifically, while SWORD's false negative rates were lower than MRW and RBS over all the Mirai IPs, PGD had a slightly lower false negative rate than SWORD for Mirai IPs that made from 121 to 240 total scans, which further led PGD to detect slightly more Mirai IPs than SWORD over all Mirai IPs. However, PGD was not as effective as SWORD against Mirai IPs that were superspreading or surreptitious, which are of particular importance for worm detection.

For superspreading Mirai IPs, which are clearly damaging when left undetected, we can observe at which point a detector reached 0% false negative rate to gauge its capability in detecting them. From Figure 10(a), we can see for Mirai IPs with more than 240 scans, PGD remained at about a 20% false negative rate, while SWORD approaches 0%. SWORD reached a 0% false negative rate when Mirai IPs made more than 270 scans. PGD, MRW, and RBS, on the other hand, did not reach a 0% false negative rate until 371, 391, and 401 scans, respectively, allowing Mirai IPs 100, 120, and 130 more scans than SWORD, respectively, before they were guaranteed to be detected.

For surreptitious Mirai IPs, we look at every detector's performance against them in two complementary measurements. We first view every detector's false negative rates against Mirai IPs that made no more than 50 scans during the entire 1-hour testing period. Figure 10(b) shows for the most part SWORD's false negative rates were clearly lower than those of every other detector compared. Note that although the false negative rates here were fairly high, so long as they were not 100%, given the false positive rates were 0%, even if only one Mirai IP was detected, a reliable early warning could be issued against the worm. On the other hand, we also notice the false negative rate curves oscillated as the number of scans increased. This is because a surreptitious worm IP that made less scans than the other surreptitious worm IPs could have just scanned within a smaller time window, thus achieving a higher scanning rate and potentially a lower false negative rate. We therefore also look at every detector's performance against Mirai IPs using every worm's scanning rate, as defined in Section 6.3. Figure 10(c) shows every detector's false negative rates against surreptitious Mirai IPs at different scanning rates between 0.001 and 0.030 scans/s. Now, the false negative rate curve decreases as the worm scanning rates increase. Again, clearly SWORD outperformed all of the other detectors at detecting surreptitious worms. For example, SWORD was even able to detect at least some Mirai IPs with scanning rates as low as 0.002 scans/s, while MRW, PGD, and RBS did not detect any worms with scanning rates less than 0.004, 0.004, and 0.005 scans/s, respectively.

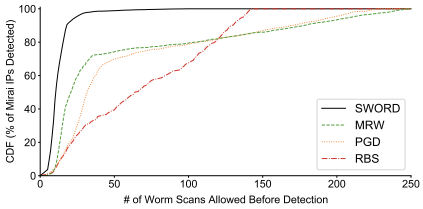
TABLE 2: Mirai worm detection latency of each worm detector.

Detector	First Detection	Allowed IPs	Allowed Scans
SWORD	7.87s	102 IPs	5380 scans
MRW	19.77s	272 IPs	13728 scans
PGD	18.20s	235 IPs	12641 scans
RBS	12.50s	169 IPs	8674 scans

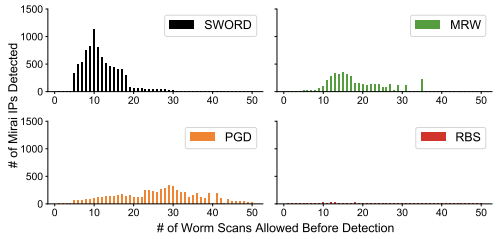
7.3 Detection Latency

Table 2 shows for each detector when it detects the first worm scan and the number of unique worm IPs allowed to scan the protected network before the detection. It also shows the total number of worm scans that occurred before the detection, which is the detection latency of a worm detector, as defined in Section 6.3. For the given 1-hour testing period, SWORD detected the first worm scan more than twice as fast as PGD and MRW, and 1.5 times faster than RBS. Out of a total of 45,291 unique Mirai IPs present in this 1-hour period, SWORD only allowed 102 unique Mirai IPs undetected to scan the protected network before the first worm scan was detected, which accounted for 5,380 total worm scans in a 7.87-second period. SWORD therefore allowed around 57% less worm scans than PGD, 61% less worm scans than MRW, and 38% less worm scans than RBS before detecting the occurrence of the Mirai worm. While 5,380 allowed incoming scans may seem high, this accounts for only 0.23% of all of the incoming Mirai scans in the 1-hour testing period (in other words, SWORD detected the presence of Mirai before 99.77% of the Mirai scans reached the protected network). While PGD detected slightly more Mirai IPs overall, SWORD's ability to detect Mirai IPs faster could be a more valuable attribute to a network operator who may want to perform mitigative and preventative steps as soon as possible to limit the damage of the worm.

We further measured the detection latency of every Mirai IP detected, as shown in Figure 11(a). Recall the latency is measured in terms of the number of worm scans allowed before detection, as defined in Section 6.3. Among all Mirai IPs detected by SWORD (8,882), 98.87% of them were all detected before they could make more than 50 scans. The next best detector in terms of latency was RBS, none of the Mirai IPs detected (1,041) made more than 140 scans. For all the Mirai IPs that PGD and MRW detected (9,100 and



(a) Detection latency (*i.e.*, # of scans before detection) CDF of the Mirai IPs detected



(b) # of Mirai IPs detected for low latency values

Fig. 11: Detection latency of Mirai IPs detected.

5,797, respectively), they made no more than 230 and 250 scans, respectively. In fact, SWORD was able to detect 80% of all of its detected Mirai IPs each within 20 scans. When compared to MRW, which detected less than 40%, and PGD along with RBS, which detected less than 10% of all of their detected Mirai IPs each within 20 scans, clearly SWORD can detect individual Mirai IPs faster than the other detectors.

Finally, Figure 11(b) details how many Mirai IPs each detector detected with very low latency (*i.e.*, 50 scans or less). For SWORD, the majority of low latency detections occurred within 10 scans, and a large portion of detections occurred even within just 5 scans. MRW's low latency detections were spread across 10 to 30 scans, and PGD's low latency detections were spread fairly evenly across the 5 to 50 scans range. Both MRW and PGD detected a significantly lower number of Mirai IPs with 5 scans, as compared to SWORD. RBS had such a low number of detected Mirai IPs for each latency that it is difficult to eyeball, but it too follows an even distribution similar to PGD, with its detections spread across the 10 to 30 scans range like MRW; it does not detect any Mirai IPs until 8 scans. Clearly, compared to other detectors, SWORD can detect many more Mirai IPs within a very small number of scans (e.g., 5 allowed scans); this fact is consistent with SWORD's lowest detection latency of the Mirai worm among all detectors evaluated.

7.4 Summary

As with outbound worm detection, SWORD again significantly outperformed all of the other detectors in inbound worm detection. While PGD detected slightly more total Mirai IPs than SWORD, SWORD outperformed PGD in terms of false negative rates and detection latency. SWORD also outperformed MRW and RBS in all three metrics. Furthermore, SWORD detected far more superspreading Mirai IPs with relatively high total scans, and surreptitious Mirai IPs with relatively low total scans and relatively low scanning rates than the other three detectors. In fact, SWORD was even able to detect some surreptitious Mirai IPs with extremely low scanning rates, unlike MRW, PGD, and RBS.

7.5 Limitations and Open Issues

It is possible that worm traffic may be present in the 1-hour training dataset. While we verified that no incoming Mirai scans were present, it may be possible that other strands of worms are present. Also, we only tested the detectors on one 1-hour period. While the results may vary across different 1-hour periods, we doubt that we will arrive at vastly different conclusions as the ones presented in this section.

8 Future Work

There are several open issues that warrant future work. First, as described in Section 3.1, a distributed version of SWORD can be studied, where multiple SWORD instances are distributed throughout the network to collectively monitor traffic. Further, in evaluating SWORD against outbound worms, rather than placing SWORD at the gateway of a protected network, one may place SWORD upstream where it can only observe a fraction of the outgoing traffic from the protected network. Also, in evaluating SWORD against inbound worms, there are several Mirai worm variants [7], and SWORD can be evaluated against these variants. Finally, SWORD could be deployed and evaluated in a real-world network on the Internet to verify the findings in this paper.

9 Conclusions

We identify two principles that an effective worm detection solution must follow: (1) Worm propagation and worm detection are in an arms race, and a detector must consider potential countermeasures from worm authors; and (2) Behavior-based worm detection must focus on the fundamental behavior of worm propagation that worms cannot avoid. Although there are many existing worm detectors, they are inadequate in following these two principles.

In this paper, we revisited behavior-based worm detection. We identified that the fundamental behavior of worm propagation is that of connecting to new destinations, and designed SWORD, a detector that encompasses two complementary modules that can detect violations that a worm will cause in connecting to new destinations. With one module monitoring burst duration and the other ensuring quiescent periods, SWORD is extremely hard for a worm to evade.

Furthermore, unlike previous behavior-based worm detectors which only focus on outbound worm detection, we designed SWORD such that it not only can detect outbound worms, but also inbound worms, even though detecting inbound worms is difficult given that a detector usually observes only a fraction of scans from worm-infected hosts. Additionally, unlike some worm detectors that require bi-directional traffic to detect worms, SWORD can detect worms solely from uni-directional traffic.

As demonstrated in our evaluation, the SWORD detector significantly outperforms all other detectors. For outbound worm detection, SWORD is not only as competitive as the best detector for detecting classic worms, but is also highly resilient against evasive worms. Furthermore, we evaluated SWORD and its competitors against the inbound Mirai worm using only uni-directional real-world incoming traffic, and demonstrated that SWORD is most effective at inbound worm detection, especially in detecting superspreading and surreptitious worm IPs.

ACKNOWLEDGMENTS

This material is partially based upon work supported by the National Science Foundation under Grant No. 0644434. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] S. Sinha, "State of IoT 2021," https://iot-analytics.com/number-connected-iot-devices/, 2021.
[2] P. Li, M. Salour, and X. Su, "A Survey of Internet Worm Detection & Containment," in *IEEE Communications Surveys & Tutorials*, 2008.
[3] M. Ward, "Why some computer viruses refuse to die," https://www.bbc.com/news/technology-44564709, 2018.
[4] C. Cimpanu, "A decade of hacking," https://www.zdnet.com/article/a-decade-of-hacking-the-most-notable-cyber-security-events-of-the-2010s/, 2019.
[5] ——, "A decade of malware," https://www.zdnet.com/article/a-decade-of-malware-top-botnets-of-the-2010s/, 2019.
[6] R. Hummel, C. Hildebrand, H. Modi *et al.*, "NETSCOUT Threat Intelligence Report: DDoS in a Time of Pandemic," https://www.netscout.com/threatreport/, 2021.
[7] O. Alrawi, C. Lever, K. Valakuzhy *et al.*, "The Circle Of Life: A Large-Scale Study of The IoT Malware Lifecycle," in *USENIX Security Symposium*, 2021.
[8] G. Gu, M. Sharif, X. Qin *et al.*, "Worm Detection, Early Warning and Response Based on Local Victim Information," in *Annual Computer Security Applications Conference*, 2004.
[9] V. Sekar, Y. Xie, M. K. Reiter *et al.*, "A Multi-Resolution Approach for Worm Detection and Containment," in *International Conference on Dependable Systems and Networks*, 2006.
[10] M. P. Collins and M. K. Reiter, "Hit-List Worm Detection and Bot Identification in Large Networks Using Protocol Graphs," in *Symposium on Recent Advances in Intrusion Detection*, 2007.
[11] J. Jung, R. Milito, and V. Paxson, "On the Adaptive Real-Time Detection of Fast-Propagating Network Worms," in *Conference on Detection of Intrusions & Malware & Vulnerability Assessment*, 2007.
[12] S. E. Schechter, J. Jung, and A. W. Berger, "Fast Detection of Scanning Worm Infections," in *Symposium on Recent Advances in Intrusion Detection*, 2004.
[13] "FRGP Continuous Flow Dataset, IMPACT ID: USC-LANDER/Mirai-FRGP-scanning-20160908/rev10326," USC/LANDER. http://www.isi.edu/ant/lander, 2016.
[14] "FRGP Continuous Flow Dataset, IMPACT ID: USC-LANDER/FRGPContinuousFlowData-20090729/rev3998," USC/LANDER. http://www.isi.edu/ant/lander, 2016.
[15] S. Staniford, V. Paxson, and N. Weaver, "How to 0wn the Internet in Your Spare Time," in *USENIX Security Symposium*, 2002.
[16] Z. Liang and R. Sekar, "Fast and Automated Generation of Attack Signatures: A Basis for Building Self-Protecting Servers," in *Conference on Computer & Communications Security*, 2005.
[17] J. Crandall, Z. Su *et al.*, "On Deriving Unknown Vulnerabilities from Zero-Day Polymorphic & Metamorphic Worm Exploits," in *Conference on Computer & Communications Security*, 2005.
[18] J. Newsome and D. Song, "Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploits on Commodity Software," in *Network and Distributed System Security Symposium*, 2005.
[19] H.-A. Kim and B. Karp, "Autograph: Toward Automated, Distributed Worm Signature Detection," in *USENIX Security Symposium*, 2004.
[20] S. Singh, C. Estan *et al.*, "Automated Worm Fingerprinting," in *Symposium on Operating System Design and Implementation*, 2004.
[21] Z. Li, L. Wang, Y. Chen, and Z. Fu, "Network-based and Attack-resilient Length Signature Generation for Zero-day Polymorphic Worms," in *International Conference on Network Protocols*, 2007.
[22] H. Zhou, Y. Hu, X. Yang, H. Pan, W. Guo, and C. C. Zou, "A Worm Detection System Based on Deep Learning," *IEEE Access*, 2020.
[23] S. M. Sohi, J.-P. Seifert, and F. Ganji, "RNNIDS: Enhancing Network Intrusion Detection Systems through Deep Learning," *Computers & Security*, 2021.
[24] S. Stafford and J. Li, "Behavior-based Worm Detectors Compared," in *Symposium on Recent Advances in Intrusion Detection*, 2010.
[25] G. Gu, R. Perdisci, J. Zhang *et al.*, "BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection," in *USENIX Security Symposium*, 2008.
[26] L. Bilge, D. Balzarotti *et al.*, "Disclosure: Detecting Botnet Command and Control Servers through Large-scale Netflow Analysis," in *Annual Computer Security Applications Conference*, 2012.
[27] R. Doshi, N. Apthorpe, and N. Feamster, "Machine Learning DDoS Detection for Consumer IoT Devices," in *Security and Privacy Workshops*, 2018.
[28] A. Procopiou, N. Komninos, and C. Douligeris, "ForChaos: Real Time Application DDoS Detection using Forecasting and Chaos Theory in Smart Home IoT Network," *Wireless Communications and Mobile Computing*, 2019.
[29] O. Alrawi, C. Lever *et al.*, "SOK: Security Evaluation of Home-Based IoT Deployments," in *Symposium on Security & Privacy*, 2019.
[30] S. Stafford, J. Li, T. Ehrenkranz, and P. Knickerbocker, "GLOWS: A High Fidelity Worm Simulator," Tech. Rep. CIS-TR-2006-11, 2006.
[31] WAND Group, "WAND WITS: Auckland-IV trace data," http://wand.cs.waikato.ac.nz/wand/wits/auck/4/, 2001.
[32] Lawrence Berkely National Laboratory, "LBNL/ICSI enterprise tracing project," http://www.icir.org/enterprise-tracing/, 2005.
[33] University of Massachusetts Amherst, "Umass trace repository," http://traces.cs.umass.edu/, 2008.
[34] M. Antonakakis, T. April, M. Bailey *et al.*, "Understanding the Mirai Botnet," in *USENIX Security Symposium*, 2017.

Jun Li is a Professor in the Department of Computer and Information Science and founding director of the Center for Cyber Security and Privacy at the University of Oregon. He received his Ph.D. from UCLA in 2002 (with Outstanding Doctor of Philosophy honor), M.E. from Chinese Academy of Sciences in 1995 (with Presidential Scholarship), and B.S. from Peking University in 1992, all in computer science. His research is focused on networking, distributed systems, and network security, with about 100 peer-reviewed publications. He has served on US National Science Foundation research panels and more than 70 international technical program committees, including chairing six of them. He is a senior member of ACM and IEEE and an NSF CAREER awardee in 2007.

Devkishen Sisodia is a Ph.D. student in the Department of Computer and Information Science at the University of Oregon (UO), and conducts research at the Center for Cyber Security and Privacy (CCSP). He received his B.S. degree from the University of Texas at Arlington (UTA) in Computer Science. His research interests include distributed denial-of-service (DDoS) attacks and defenses, Internet of Things (IoT) security and privacy, and network measurement.

Shad Stafford is a Principal Engineer at Palo Alto Software, where he's been building software to help small businesses for more than 10 years. He earned his Ph.D. in Computer Science from the University of Oregon in 2012.