Fast and Evasive Attacks:

Highlighting the challenges ahead

Moheeb Abu Rajab Fabian Monrose Andreas Terzis

Johns Hopkins University
Baltimore MD 21218, USA,
moheeb, fabian, terzis@cs.jhu.edu

Abstract. Passive network monitors, known as telescopes or darknets, have been invaluable in detecting and characterizing malware outbreaks. However, as the use of such monitors becomes commonplace, it is likely that malware will evolve to actively detect and evade them. This paper highlights the threat of simple, yet effective, evasive attacks that undermine the usefulness of passive monitors. Our results raise an alarm to the research and operational communities to take proactive countermeasures before we are forced to defend against similar attacks appearing in the wild. Specifically, we show how lightweight, coordinated sampling of the IP address space can be used to successfully detect and evade passive network monitors. Equally troubling is the fact that in doing so attackers can locate the "live" IP space clusters and divert malware scanning solely toward active networks. We show that evasive attacks exploiting this knowledge are also extremely fast, overtaking the entire vulnerable population within seconds.

***Index Terms*—Keywords: Network Monitoring, Network Worms, Invasive Software, Network Security.**

1 Introduction

Passive network monitors (or so-called *network telescopes* [17]) have provided a wealth of information in recent years about active scanning malware (*e.g.*, [18, 30]). The relative ease of deploying passive monitors has made them instrumental in a number of malware centric proposals, especially for early detection of global outbreaks (*e.g.*,[36]), containment and quarantine (*e.g.*, [15, 19, 22]), and forensic analysis [28, 30]. However, much of this work—including our own—has only been possible because the attacks observed thus far have been rather crude in nature. Arguably, the lack of sophistication in recent outbreaks has been justifiable, as thus far, there has been little reason to do otherwise.

It is clear however, that attackers will naturally become more savvy as more elaborate defenses are deployed. Indeed, since passive network monitors were introduced, a number of Internet malware outbreaks have applied non-uniform scanning to find victims and in doing so, limit the activity observed by centralized monitors (*e.g.*, CAIDA's network telescope [17]). In response, the research community has advocated the use of distributed telescopes (*e.g.*, [1, 36]) as a

way to increase detection speed by synthesizing multiple views of the infection as seen from various vantage points [27]. Furthermore, active responders (*e.g.*, honeynets) are now widely used to lure attackers, and capture the attacks' payloads to generate malware signatures [11, 16] in near real-time. Unfortunately, even the most sophisticated techniques for analyzing the data captured by network monitors are vulnerable to evasive tactics that refrain from scanning these monitors in the first place. In fact, recent evidence indicates that certain classes of malware already avoid well-known monitors (*e.g.*, Agobot [37]), and completely avoid prefixes of certain agencies [1].

In this paper we demonstrate the impending threat to network monitors by presenting a simple technique that can dynamically evade such monitors. The outlined approach raises a number of challenges for the research community because unlike previous techniques that use static "do-not-scan" lists, this approach can produce agile evasive malware that proceeds in an online fashion and completely undermines the utility of passive monitors. Specifically, the proposed technique employs lightweight sampling of the IP space to identify *live prefixes*, that is, prefixes that contain live networks, and isolates *empty prefixes* that are either unused or dedicated to passive monitoring. Sampling simply involves sending a small number of probes (*e.g.* TCP SYN packets) to random addresses within each target prefix. A single response from any address indicates that the prefix contains live hosts and is therefore classified as *live*. Otherwise, the prefix is identified as *empty*. Our results show that this technique successfully isolates large collections of distributed monitors and discovers 96% of the vulnerable population by probing less than 5% of the entire IP space. While the main focus of this paper is showing that sampling effectively evades passive network monitors, the proposed technique can be easily extended to evade active responders; if not designed correctly, active responders generate distinctive behaviors that are detectable by the same sampling mechanism.

To further illustrate this threat, we present malware infection strategies that use knowledge assembled from offline or online sampling to divert the infection towards live prefixes. We show that malware exploiting these strategies can infect more than 95% of the vulnerable population in tens of seconds while still successfully evading large collections of distributed passive monitors.

The rest of the paper is organized as follows. The sampling process, a core component of the proposed evasive techniques, is described in Section 2 and evaluated in Section 3. In Section 4 we provide examples of practical malware spreading strategies that employ such evasive techniques. Rather than providing guidelines for attackers, our goal is to highlight the challenges that network monitors must overcome. In that regard, in Section 5 we discuss a number of promising directions that can address these challenges. We highlight the differences between our work and related previous proposals in Section 6 and close in Section 7 with concluding remarks.

[1] For example, we have seen botmasters educating each other to avoid scanning prefixes listed at `http://professionalsecuritytester.com/modules.php?name=News&file=article&sid=70`.

2 Discovering the Live Population via Sampling

Our sampling technique takes advantage of the highly clustered nature of the allocated and live IP space [2, 13] in order to efficiently detect prefixes that contain live hosts. Specifically, we use a hierarchical sampling technique (shown pictorially in Figure 1) that follows a depth-first search strategy that, at first, probes addresses selected at random from each of the /8 prefixes. These probes can take many forms, and might be a TCP SYN packet, an ICMP packet, or an ACK packet with a popular source port (specially crafted to bypass stateless firewalls). If at least one response is received, the corresponding /8 prefix is then marked as live and the sampling process proceeds to send probes to the /16 prefixes within that /8. If no response is received, then the prefix is marked as empty and no further probes are sent to that prefix. For each live /16 prefix, the process continues in search of any live /24 prefixes.

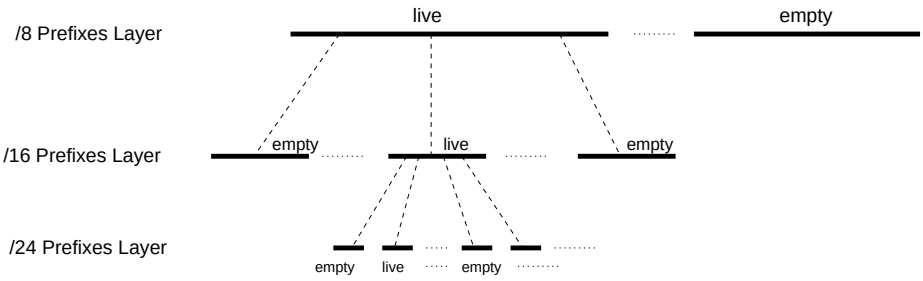

Fig. 1. Diagram illustrating the proposed hierarchical sampling process.

Since the main goal of the sampling process is to detect passive network monitors, it is important that the process itself evades detection. Therefore, to stay undetected, the sampling process must send as few probes as possible to each prefix and, at the same time, detect the live and empty prefixes with high accuracy. Additionally, the probing mechanism must be chosen in such a way that complicates the detection of probes and makes it difficult to correlate the probing activity itself. In this way, generating any useful signature for detecting this activity quickly, becomes a complex task. These evasive measures can be achieved, for example, by selecting popular target ports (*e.g.*, port 80) that easily "blend" the sampling probes within large volumes of innocuous traffic.

2.1 Sample Size Estimation

As stated earlier, the goal of the sampling process is to detect all live prefixes while sending as few probes as possible. In what follows, we derive the maximum number of samples n necessary to classify prefixes with high confidence. While applicable to all levels of the sampling hierarchy, the discussion that follows describes how to derive the number of samples n for the /16 prefix layer. We do so because the address allocation at the /8 prefixes is publicly available (*e.g.*,

$P(g)$	Distribution of live hosts at the /16 prefixes layer
$p^*(g)$	Marginal distribution of live hosts at the /8 prefixes layer
$p_{l,g}$	The probability of probing a live host in group g
β	Threshold probability of liveliness for a certain prefix
N	The total live population size
n	Maximum number of probes required to classify a prefix as live or empty
α	Confidence level of the sampling classification decision
V	Total number of vulnerable hosts
I_t	Number of infected hosts at time step t
s	Average scan rate (scans/time step) per infected host
p	Probability of contacting an address in the live IP space

Table 1. Notation used throughout the paper.

IANA [9] and ISC [10]) so one can easily preclude all unallocated /8 prefixes. Moreover, Zeiton *et. al.* [38] showed (in a study that applies only to /24 prefixes) that by exploiting common network administration practices [2] it is possible to use $n = 11$ probes and still detect more than 90% of the live /24 prefixes.

Sampling Model Let $p_{l,g}$ be the probability of probing a live host in prefix g. Then given n samples, the probability α of receiving at least one response from prefix g is:

$$\alpha = 1 - (1 - p_{l,g})^n \tag{1}$$

Our objective is to find the number of samples n necessary to contact at least one live host within a certain prefix with probability α. Therefore, n is given by:

$$n = \frac{\log(1-\alpha)}{\log(1-p_{l,g})} \tag{2}$$

Ideally, n should be large enough to detect live /16 prefixes containing a single live host. This, however, would require a prohibitively large number of probes (*e.g.*, approximately 301,000 probes for confidence $\alpha = 0.99$). Fortunately, it is unlikely that such /16 prefixes are in use today—in reality, live prefixes contain significantly more live hosts. Therefore, from a practical standpoint, the goal is to detect the /16 prefixes that contain the majority of live hosts and exclude the empty or sparsely populated prefixes. With this in mind, we amend the definition of an empty prefix to include prefixes with live host occupancy ($p_{l,g}$) below a certain threshold β. Accordingly, the maximum number of probes necessary to detect a non-empty prefix can be calculated by replacing $p_{l,g}$ with β in Equation 2.

Notice that as the threshold β increases in the denominator of Equation 2, the number of required samples, n, decreases. On the other hand, if β is too large, a significant number of live prefixes could be potentially misclassified as empty. Therefore, the goal is to determine a value for β such that the sampling

[2] Namely, probing addresses commonly assigned by network administrators (*e.g.*, `a.b.c.1`, `a.b.c.129`, etc.)

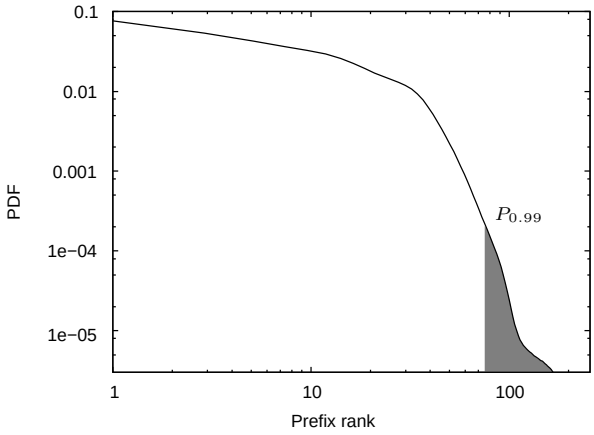

Fig. 2. Illustrative figure showing the marginal distribution of the Internet live hosts $p^*(g)$ that is used to estimate $P(g)$. The unshaded area represents the 99% of the live population contained in densely populated prefixes, each with a fraction of the total Internet population greater than the cutoff $P_{0.99}$.

process detects the live prefixes that contain the majority of the live population. In more specific terms, we define β as the threshold value of live host occupancy ($p_{l,g}$) at which the sampling process—with high probability—detects the set of live prefixes that contain 99% of the total Internet live population. Clearly, finding this set of live prefixes requires knowledge of the distribution of live hosts over the /16 address prefixes. We term this distribution as $P(g)$, which denotes the fraction of the overall Internet live population residing in the /16 prefix g. Assuming that $P(g)$ is known, $p_{l,g}$ can then be expressed as:

$$p_{l,g} = \frac{P(g) \cdot N}{2^{16}} \tag{3}$$

where N is the total number of live hosts in the Internet. The numerator in the expression above is the expected number of live hosts in /16 prefix g, while the denominator is the size of a /16 prefix.

As illustrated in Figure 2, 99% of the live population (the unshaded area) is contained in the most densely populated live prefixes with $P(g)$ greater than a cutoff probability termed $P_{0.99}$. Therefore to detect these live prefixes we set ($P(g) = P_{0.99}$) in Equation 3. Calculating $p_{l,g}$ in Equation 3 at the point corresponding to ($P(g) = P_{0.99}$) yields the minimum threshold occupancy β required to calculate the maximum number of samples n.

Unfortunately, $P_{0.99}$ cannot be directly determined since the distribution $P(g)$ is unknown. Indeed, if this distribution was known the entire sampling process would be superfluous. While $P(g)$ can be reliably estimated using a pilot Monte Carlo study, this would require a large sample size. Instead, we consider how $P(g)$ can be estimated using a small learning set of live IP addresses that can be easily obtained from various sources (*e.g.*, a pilot limited-scale random probing, historic logs, or traces of traffic arriving at unused IP space). Using

this dataset, we estimate $p^*(g)$, the marginal distribution of $P(g)$, defined as the distribution of live hosts at the /8 prefix level. We derive $p^*(g)$ by aggregating the IP addresses from the learning set to their common /8 prefixes. Due to space constraints, the discussion about the quality of this estimator is presented in the extended version of this paper [26] in which we derive a theoretical error bound for this estimator. Our results show that with a small learning dataset of only $20,000$ live IP addresses one can estimate distribution $p^*(g)$ with an empirical estimation error of $e = 4.3 \times 10^{-5}$.

Given $p^*(g)$, we can determine the cutoff probability $P_{0.99}$. Then, using an estimate N of the number of lives host in the Internet we find the number of live hosts in the /8 prefix corresponding to $P_{0.99}$. Assuming that hosts within that particular /8 prefix are uniformly distributed across all of its /16 siblings [3] (*i.e.*, $P(g) = P_{0.99}/256$), then from Equation 3, $\beta = \frac{P_{0.99} \cdot N}{256 \cdot 2^{16}}$. Finally, n can now be calculated by substituting $p_{l,g}$ with β in Equation 2.

3 Sampling Process Evaluation

First, we illustrate the sampling process by using two distributions of live hosts derived from two independent datasets. The first dataset was obtained from DShield [6] and consists of intrusion logs from over 1,600 intrusions detection systems distributed around the globe. The second dataset was collected at a local darknet covering a large number of /24 prefixes. Table 2 summarizes the statistics for both datasets. From each dataset, we independently derive two distributions: one for live hosts at the /16 level and another at the /8 level. Although collected from two distinct sources, the distributions at the /8 level are strikingly similar (see Figure 3) [4]. The reason behind this similarity is due to the fact that vulnerable hosts in both cases are selected from the same underlying distribution of live hosts which confirms a similar observation recently made in [2]. In what follows, we use these distributions as representatives of the live host distribution over the whole Internet.

DShield dataset	
Data Collection Period	three months (Oct. to Dec., 2004)
Total Unique sources	31,864,871
Sources attacking port 80	632,472
Darknet dataset	
Data Collection Period	one month (Oct., 2005)
Total Unique sources	1,153,599

Table 2. Summary of the data-set

[3] While this is not the case in practice, this assumption does not skew our calculations significantly since it is only applied to the sparsely populated /8 prefix with density $\leq P_{0.99}$.

[4] A similar relation was observed for the /16 distributions but is not shown due to space constraints.

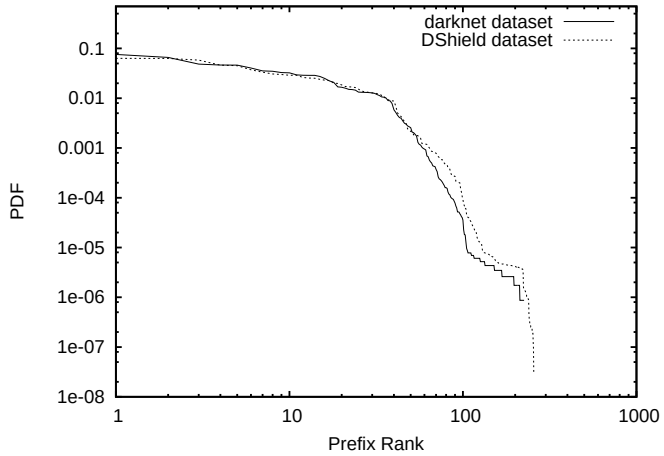

Fig. 3. Fraction of live addresses per /8 prefix for the DShield dataset compared to the darknet dataset.

Following the process described earlier, we first estimate the marginal live host distribution over the /8 prefixes ($p^*(g)$) using a small learning dataset of live IP addresses. A uniform random sample of 20,000 source IP addresses taken from the darknet dataset is used as the initial learning set. From $p^*(g)$ we find the cutoff live population density $P_{0.99}$. In this case $P_{0.99} = 0.00065$. Assuming a population of approximately 300 million live hosts [10], the estimated number of hosts in the /8 prefix with density $P_{0.99}$ is 180,000. Assuming that these hosts are uniformly distributed among its constituent /16 prefixes, we find that $\beta = 0.012$. This corresponds to /16 prefixes that contain as few as $(0.012 \times 2^{16} = 760)$ live hosts. Finally, substituting β in Equation 2 we find that $n = 400$ samples. This means that using a maximum of 400 probes per /16 prefix it is possible, with high probability ($\alpha = 0.99$), to detect the /16 prefixes that contain 99% of the overall Internet live population.

To validate the above result, we simulated the sampling process over a synthetic population of 300 million live host, distributed according to the DShield dataset. The simulated sampling process simply generates up to 400 random IP addresses from each /16 prefix. If at least one IP address exists in the hypothetical live host set we mark that prefix as live. Our results show that the sampling process successfully detected live prefixes containing 98% of the live population and isolated all empty and sparsely populated ones.

However, an attacker's ultimate purpose is to find the vulnerable population (*i.e.*, the subset of the live population that is susceptible to the attack). Therefore, a pertinent question is what percentage of the vulnerable population is contained in the live prefixes detected by the sampling process. To answer this question, we generated a hypothetical vulnerable population by extracting all the sources from the DShield dataset that contacted port 80 and mapped

each source to its corresponding /16 prefix. Our results show that 96% of the addresses from the vulnerable population are contained in detected live prefixes. This result shows that the sampling process accurately detects live prefixes without undue loss of the vulnerable population.

That said, the above trace-driven simulation implicitly assumes that all live networks are reachable and so responses (or lack thereof) to the sampling probes are indicative of network liveliness. In practice, perimeter security defenses such as firewalls that silently drop probes, can decrease the reachability of the live address space, negatively impacting the accuracy of the sampling process. In the next section, we evaluate the impact of such defenses on the effectiveness of the sampling process through a large scale IP space probing experiment.

3.1 Results from the wild

We further explored the effectiveness of this approach by conducting a large scale probing experiment based on the methodology presented in Section 2. The set of /8 prefixes sampled in this experiment was selected from publicly available information (*e.g.*, IANA [9] and ISC [10]) and excludes all unallocated or reserved /8 prefixes. We also excluded "sensitive" prefixes such as those used by certain government agencies. The outcome of this selection process was a list of 69 /8 prefixes. These prefixes were then sampled using 256 nodes of the PlanetLab distributed platform [21], each of which were assigned a set of /8 prefixes. Each node selected a /16 prefix from its assigned set and sent a maximum of ($n = 400$) SYN packets with destination port 80 to randomly generated IP addresses within that prefix. Probes were sent at a rate of one probe every 5 seconds [5]. Once the first response was received from an IP address within the probed prefix, the prefix was marked as live and outstanding probes to that prefix were terminated. If all the 400 samples received no response, the prefix was marked as empty.

The best way to validate the accuracy of the sampling process is to compare the results to the actual address space usage. Unfortunately, that information is not readily available, and so in lieu of that we resort to a simple heuristic to indirectly assess the quality of the sampling results. Specifically, using BGP snapshots from RouteViews [29], we examine the reachability of the prefixes we probed. The intuition is that prefixes that were not advertised in the BGP snapshots are unreachable and therefore should appear as empty in the sampling results. Consequently, if the sampling process marks a non-advertised prefix as empty, then the sampling decision is indeed correct. Note that the converse is not true—prefixes that have no live hosts can still be reachable. For instance, address space monitored by a network telescope is practically empty space but it is advertised in order to receive the "unwanted" traffic.

Figure 4 visualizes the results of the probing experiment. The x-axis shows the probed /8 prefixes. To preserve the privacy of these networks, we anonymized the first octet of these prefixes and present them in a random order. The y-axis shows the /16 prefix index within each /8. Each block in the map is colored ac-

[5] The choice of the target port as well as the probing rate are specific to the conditions of our experiment. In practice, faster and more sophisticated techniques can be used.

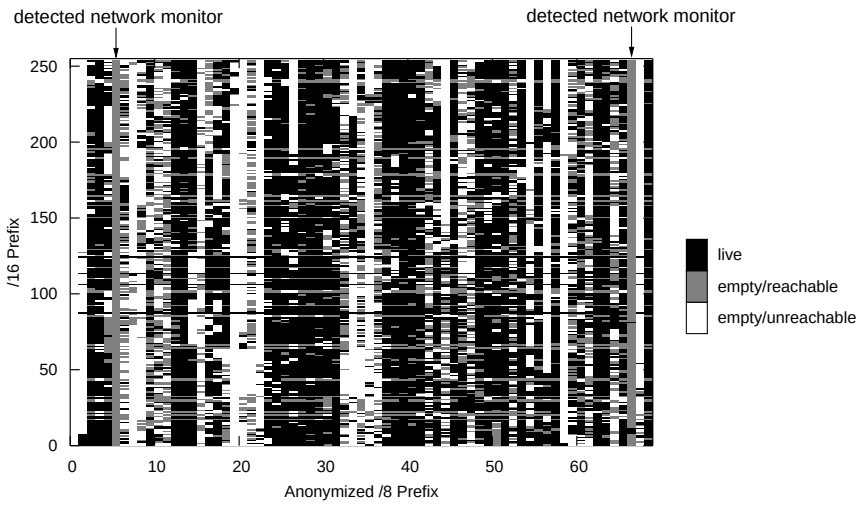

Fig. 4. Summary of results from the probing experiment.

cording to the sampling result of the corresponding /16 prefix. The white blocks in the figure show the empty and unreachable (non-advertised) prefixes. Overall, 63% of the empty prefixes were not advertised in the RouteViews dataset. The gray blocks show empty, but reachable, prefixes. These prefixes correspond to allocated but unused space or to passive network monitors. Interestingly, the sampling process successfully detected two large network monitors belonging to two different research institutions, which we verified using out-of-band information. All the detected live prefixes, shown by the black clusters on the map, were advertised in the RouteViews dataset. Finally, notice that the detected live prefixes within each /8 are highly clustered, which is a direct result of common prefix allocation practices.

The sampling process sent a total number of 3.3 million probes, which is significantly less than the 2.43 billion probes used by Bethencourt *et.al.* [3] [6]. Interestingly, the number of probes required to detect a live prefix follows a heavy tailed distribution with a mean of only 50 probes. This is due to the underlying live host distribution, and shows the effectiveness of the bound derived in Section 2.

Next, we examine the percentage of the live population that resides in the detected live prefixes. We do so using the two datasets mentioned earlier (see Table 2). Of all the sources that reside in the probed /8 prefixes, 86% of the DShield sources and 88% of the darknet sources belong to live prefixes detected by our probing experiment. This result further proves the effectiveness of the

[6] The total number of probes in [3] was actually $\sim$ 9 billion, but we scale it to the 69 /8 prefixes we targeted.

probing process in locating the majority of the live population. Moreover, it shows the minimal impact that current network perimeter defenses have in hiding the live address space.

While the previous experiment tested the accuracy of the sampling process at the /16 prefix level, we also examined its effectiveness at the /24 level. To do so, we selected two /8 prefixes belonging to two different major ISPs and applied the sampling process to detect live /16 and /24 prefixes. We found that the sampling process was equally effective in detecting these prefixes, requiring only 5 probes, on average, per /24 prefix.

4 Evasive Malware Attacks

Without question, the sampling mechanism presented in Section 2 can potentially be abused for nefarious purposes. For example, information about the location of live hosts could be exploited to launch targeted attacks against selected prefixes—a behavior widely exhibited by botnets. More importantly, malware strains that incorporate knowledge about the location of empty prefixes to guide their scans could potentially evade detection by passive network monitors. We demonstrate the practicality of this threat through two sample infection strategies outlined in the sections that follow. Later, we turn our attention to ways in which this threat can be mitigated.

4.1 Worm Spreading using off-line sampling knowledge

We first consider a scenario where the attacker samples the address space prior to launching the actual attack. The knowledge from the sampling process is encoded and shared as a hierarchical bitmap (similar to that shown in Figure 5.a) representing the live prefixes at each layer of the hierarchy.

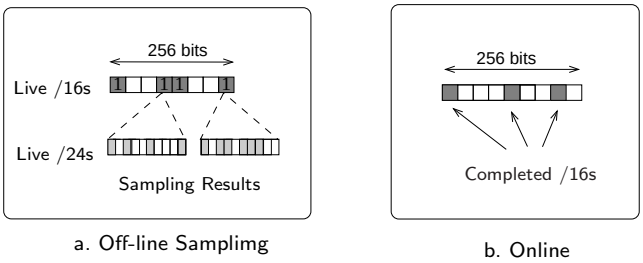

Fig. 5. Part (a) shows the information collected during offline sampling for a given /8; the index of the bitmap represents the prefix ID. Live prefixes are encoded as "1" in the bitmap and only live prefixes are expanded. Part (b) shows the online case where nodes only share progress information.

The infection phase begins by targeting an initial hit-list to which the attacker disseminates the constructed bitmap. Each infected node from the hit-list then starts scanning the IP space uniformly, but only sends scans to IP addresses

Number of Vulnerable hosts	630,000
Average scanning rate per infected host (s)	350 scans/sec
Size of initial Hit List	256
Scanning Algorithm	Uniform with evasion
Monitors configuration	256 /16 (randomly deployed)
Network Delay	$\mu = 50$ ms , $\sigma = 20$ ms
Sampling Interval per /16 prefix	3 sec
Sampling Interval per /24 prefix	1 sec
Number of delegated /8 prefixes per host (for the on-line case)	1

Table 3. Worm Simulation Parameters.

within live prefixes. Furthermore, each new victim receives a copy of the bitmap along with the malware payload.

Propagation Model: This infection strategy can be modeled by extending the worm spreading models presented in [4, 27]. The worm search space in this case is reduced from the entire 2^{32} IP address space to the sum of the space covered by all the detected live prefixes. Therefore, the probability of contacting a certain host is equal to the probability (p) of contacting a host in the live space. Given p, the expected number of infected hosts I_{t+1} at time $t+1$ is given by:

$$I_{t+1} = I_t + (V_t - I_t)\left[1 - (1-p)^{sI_t}\right] \tag{4}$$

where V_t is the total number of vulnerable hosts and sI_t is the total number of scans generated by all currently infected hosts I_t, each scanning at a rate of s scans/time step. Since $p > \frac{1}{2^{32}}$ the infection speed is higher than that of a uniform scanning worm. Moreover, since p increases as the live portion of the address space decreases, worm speed increases proportionally to the size of the un-scanned (empty) space. Therefore, sampling not only improves the stealthiness of the worm but increase its spreading speed as well.

We evaluate malware spreading in this case via simulation and compare it to a conventional uniform scanning worm outbreak. The simulation parameters we used are shown in Table 3. Network monitors in our simulation are abstracted as IP prefixes that record the source IP address of each connection attempt.

The sampling process is simulated first, as described in Section 3. In addition to detecting the live prefixes that contain 96% of the target vulnerable population, sampling classified *all* the 256 /16 monitor prefixes as empty after sending only 400 probes to each monitor. In practice, this small number of scans has a very low likelihood of triggering alarms given the sheer amount of background "radiation" that is continuously received at network telescopes [20]. Once the sampling process ends, we simulate the worm spreading over the detected live prefixes. Figure 6 illustrates the infected fraction of vulnerable hosts versus time compared to a uniform scanning worm. First, notice that since the uniform scanning worm scans the entire IP space each infected host will send at least one scan to the distributed network monitors at some point in the infection cycle. These

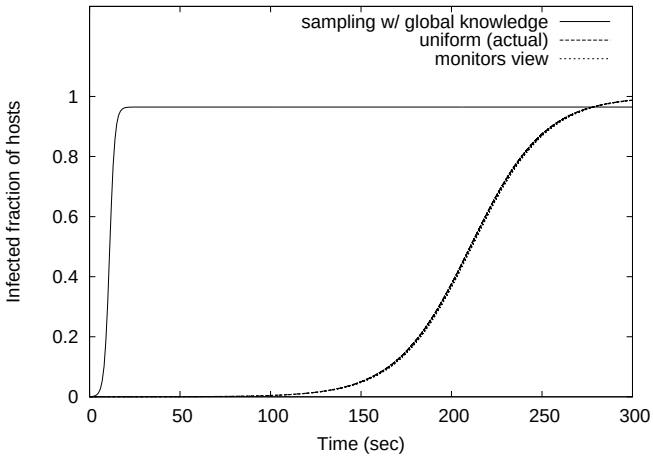

Fig. 6. Fraction of hosts infected over time for an evasive worm compared to ordinary uniform scanning worm with the same parameters.

contacts are recorded by each network monitor generating the combined monitors' view also shown in Figure 6. While the 256 /16 monitors accurately track the evolution of the uniform worm (in fact, the two lines overlap in the figure), the worm that exploits the knowledge from sampling remains invisible. Moreover, since the sampling worm targets only live prefixes it spreads significantly faster than its uniform counterpart.

Including the hierarchical bitmap in the worm payload results in a relatively large footprint; nearly 568 KB based on the evaluation in Section 3. This shortcoming can be easily alleviated by applying other mechanisms to disseminate sampling information among the infected hosts. A simple alternative is to incrementally cover the address space by exchanging bitmaps that cover a single /8 prefix bitmap at a time. In the next section, we illustrate a strategy that incorporates the sampling process in the actual infection. Unlike the offline case, this strategy is immune to short term changes in the address space usage. Moreover, as we show next, the worm payload is significantly reduced in this case.

4.2 Online Worm Spreading Strategy

The online worm variant incorporates the sampling process into the actual infection. As before, we assume that the attacker starts with an initial hit-list of vulnerable hosts. Each host in the hit-list is delegated a number of /8 prefixes. Once infected, the infectee selects a random /8 prefix from the group of delegated prefixes and starts sampling it using the hierarchical sampling process to detect the live /16 and /24 prefixes. Once the first response from a live /24 prefix is received, the worm activates its scanning vector and attempts to infect any vulnerable hosts in that /24 prefix.

To avoid re-sampling, each worm instance maintains a bitmap (see Figure 5.b) that tracks the already sampled /16 prefixes within the delegated /8 prefixes.

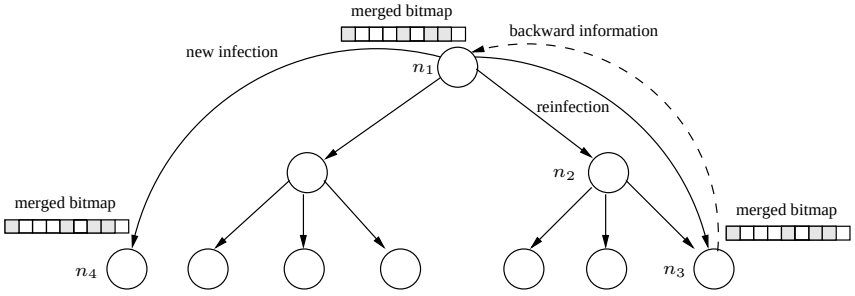

Fig. 7. Example of backward information sharing as a result of re-infection. Here, n_1 re-infects n_3 which in turn transmits its progress to n_1 when both nodes merge their bitmaps. n_1 can now disseminate this updated information to subsequent infectees (*e.g.*, n_4).

Nonetheless, this mechanism by itself cannot eliminate re-sampling across different worm instances. Doing so requires some form of continuous information exchange among worm instances. However, this can be easily accomplished by taking advantage of the inherent communication channel provided by the infection process. In particular, the worm instance can simply transfer a bitmap that represents its current progress to each new infectee. In this way, the infectee does not re-sample or re-scan prefixes already scanned by its infector. Additionally, as Figure 7 illustrates, infected hosts can exploit the re-infection process to continuously update their bitmaps. Notice that in this case the information exchanged, as well as the size of the worm payload, is significantly reduced compared to the offline case—now, only 256 bits are required to track the sampling progress within an entire /8 prefix.

As before, we evaluate the online infection strategy using the simulation parameters from Table 3. The left line in Figure 8 represents the evolution of the worm over time. Again, notice that the worm successfully evades detection with fewer than 400 sources sending probes to any of the distributed monitors (out of a total of 600,000 infected hosts). In addition, the worm's infection speed is not severely reduced by the overhead of the sampling process—it still reaches saturation in under 500 seconds. It is also noteworthy that even without having to continuously estimate the fraction of infected hosts (as required by Ma *et.al.* [16]), the worm self-terminates its scans upon saturation after ∼1050 seconds.

Finally, one would expect that infected nodes that fail or are immunized during the worm outbreak would result in losing the parts of the IP space delegated to the failed nodes. However, as the right line in Figure 8 illustrates, even with a node failure rate [7] of 2% the worm still infects all the vulnerable population. This is because we deliberately chose a sub-optimal redundancy reduction scheme in which certain prefixes were scanned by more than one host—a tactic that can be easily used by attackers.

[7] We define the failure rate as the percentage of infected nodes per second that simply stop scanning either because they are treated or because they fail.

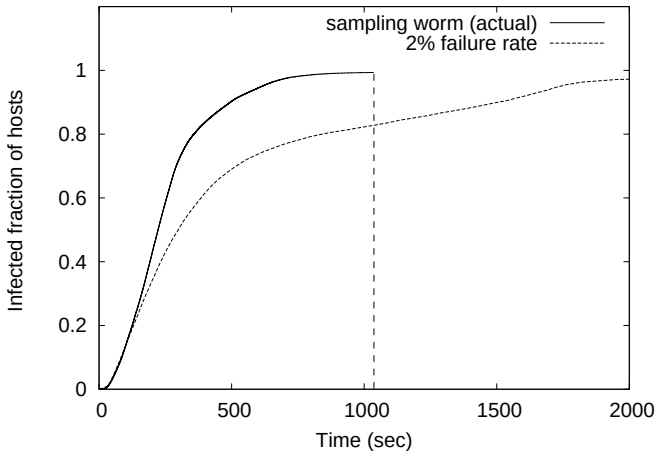

Fig. 8. Fraction of hosts infected over time for an online sample-and-spread strategy.

5 Countermeasures and Challenges

To maintain their future value, current passive malware monitoring practices must evolve to face the threats posed by evasive attacks. In what follows, we discuss a number of promising research directions and the pertinent challenges that need to be addressed in order to counter this emerging threat.

Increased Network Surveillance. Given the proliferation of malware on the Internet, it is fair to assume that more resources will be allocated to build early warning systems. To better understand the ability of such distributed warning systems to detect an on-line evasive worm, we consider the case in which a distributed monitoring system comprises a heterogeneous mix of a single /8 monitor, 256 /16 monitors, and a collection of 1024 /24 monitors. The /24 monitors are deployed within heavily populated prefixes as recommended in [27], while the rest are deployed randomly over the IP space.

Figure 9 depicts the actual onset of the worm compared to the collective view of all monitors in the distributed system. Although the monitors' view is slightly enhanced compared to the results from the previous section, it is still severely limited; the monitors only received probe traffic from 1% of the infected population. To make things worse, these results assume an idealistic condition where receiving a single probe from an infected host is enough for a monitor to deduce that the host is indeed infected and instantaneously notify all other monitors in the distributed system.

A promising protection against such evasion techniques is to use smaller monitors. For example, Pouget *et. al.*[23] recently established a distributed monitoring system using monitors of only three IP addresses deployed in more than 25 different countries. Such monitors are not easily detected by the proposed sampling process as this would require extensive probing. However, in order to be useful as an early warning system, coordination among a substantial number

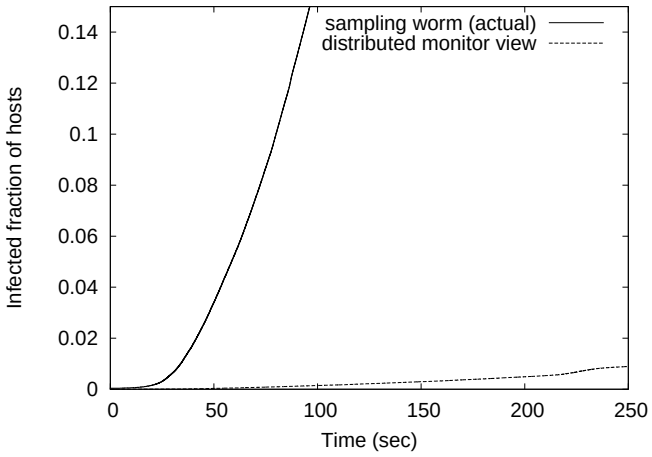

Fig. 9. Onset of an evasive worm showing the actual evolution and the collective view of a (/8 + 256 /16 + 1024 /24) distributed monitoring system.

of small (geographically dispersed) monitors will be required [27]. Implementing such a distributed monitoring systems in a scalable manner is non-trivial and remains an open research direction which deserves further investigation.

Active Responders. Another avenue that can offer pragmatic value lies in the widespread adoption of virtual active responders (*e.g.*, [1, 25, 37]). Of late, active responders have become popular for automatically generating attack signatures [11, 12, 14, 34]. If successful, these approaches would also mislead the sampling process into marking the monitored space as live and subsequently scanning it, thereby exposing the attack. That said, implementing deep-interaction active responders (also known as honeynets) in a scalable and inconspicuous manner is non-trivial [7, 37]. Recently, Vrable *et. al.* [35] proposed techniques based on aggressive multiplexing of virtual machines (VMs) to achieve scalable honeynets that improve the state-of-the-art by up to six orders of magnitude. However, it is well known that several malware actively detect VM-based execution environments [8] and alter the malware behavior accordingly. Therefore, limiting the detectable effects of VM-based honeynets remains an ongoing challenge.

In the context of evasive malware attacks, an equally difficult challenge is masking any *external* side effects of the monitor's presence. For example, a virtual responder can not respond to all incoming connection attempts since this would appear suspicious from the attackers' stand-point given the low likelihood of observing such a dense mass of live hosts. On the other hand, selectively responding to incoming probes is not ideal either. For one, probabilistically responding to a fraction of the incoming traffic degrades the monitor's fidelity and can lead to loss of "interesting" malicious flows. Moreover, even probabilistic responding is not immune to well crafted sampling probes. For example, a monitor responding with an ACK to a probe sent to a non-popular or unknown destination port will be equally suspicious as not responding at all.

A promising trade-off is to have active responders intelligently mimic the surrounding IP space in terms of the live host distribution and active services. Using this persona, the monitor can decide which probes should be answered, thereby increasing the difficulty of distinguish the monitored space from its immediate surroundings. Again, to be useful, the probability of response must be significantly greater than that of contacting a real vulnerable host in the operational network. This is certainly feasible if the monitor space is substantially larger than the network being protected. If not, the responder will offer little value in protecting operational end-hosts. Exploring the potential of designing camouflaged responders in this manner is an area of research that requires further investigation.

Finally, even if the above measures are implemented, the attackers will eventually locate monitors bound to static locations. Therefore, for these techniques to be of long term value, they should be combined with periodic "rotation" of the monitored address space. This can be achieved, for example, by having organizations actively rotate the operational portion of their address space used for DHCP leases and deploy active responders within the remaining unused space. This approach raises a number of practical challenges, but is a direction that can have valuable impact in mitigating these threats and so warrants further examination.

6 Related Work

The research community has only recently shown interest in techniques that detect network monitors. Bethencourt *et.al.* [3] and Shinoda *et.al.* [31] showed how so-called probe and response attacks can locate network monitors that issue periodic reports of suspicious connection attempts. Although effective, these approaches are slow and heavyweight since they require low-rate, exhaustive probing of the address space. Furthermore, they are limited to network monitors that issue public reports—a requirement that can be easily invalidated by eliminating or anonymizing these reports. Moreover, these probe-response techniques target predetermined list of locations in which the reports are published (*e.g.*, web-pages of certain repositories [6]) and therefore cannot detect monitors that publish reports in public, yet unknown or untargeted locations. On the other hand, the techniques in this paper detect network monitors through faster, stealthy and lightweight sampling that does not require a pre-established set of publishing locations.

Zeiton *et.al.* showed that it is possible to estimate the liveliness of /24 address prefixes by selectively probing IP addresses based on common network administration practices (*e.g.*, selecting addresses commonly used by router interfaces such as `a.b.c.1` and `a.b.c.129`). While the technique successfully detected live prefixes in their study with more than 90% accuracy, it is only applicable to /24 prefixes. By contrast, the methodology we use provides an upper-bound on the number of probes and is applicable to prefixes of any size.

Over the past few years a number of proposals have highlighted the threat from fast worms that employ novel scanning techniques. At a high level, these

techniques boost worm spreading using various forms of collaboration among worm instances. For example, Staniford *et.al.* [33] outline a number of collaborative scanning strategies, including permutation scanning in which the worm maps the IP space into a large permutation and diversifies the starting point of scanners to reduce redundant scanning. While this strategy allows worms to spread much faster, the scanning activity is still visible to network monitors because the worm still scans the entire IP space. Flash and topological worms can be even faster, reaching saturation in a few seconds [32]. However, although inherently evasive, these worms assume a priori knowledge of the vulnerable population through an already existing large hit-list.

More recently, Chen *et.al.* presented an alternative strategy to disseminate information about the vulnerable population distribution and divert worm scans toward populated address groups [5]. However, as evasion was not a core objective, the proposed approach suffers various limitations from that perspective. For one, the worm initially scans the IP space uniformly at random to find enough vulnerable hosts to derive an accurate estimate of the vulnerable population distribution. This activity is easily detected by distributed network monitors. Second, the vulnerable population distribution is only estimated at the /8 prefix level. Hence, the technique can limit worm scans toward monitors occupying entire /8 address prefixes, but the worm is still detectable by distributed collections of small monitors deployed in heavily populated prefixes (as recommended in [27]). More problematic is the fact that in order to learn the vulnerable population distribution, each infected host must contact a centralized "worm-server". Such a server is both a single point of failure and an unnecessary bottleneck.

The coordination mechanism we propose exploits re-infections to disseminate updated knowledge about the vulnerable population across worm instances. This idea was independently suggested by Ma *et.al.* for designing *self-stopping* worms [16]. In that case, re-infection is used to share an estimate of the infected population which is then used to decide when to stop scanning. Although such worms can hide the infected population past worm saturation, they can still be detected (and contained) during the spreading phase. In contrast, the examples we present conceals the worm activity during its spreading phase and is inherently self-stopping.

Lastly, a number of measurement studies based on packet traces collected from network monitors have already speculated that persistent port scanning activities is being used to fingerprint vulnerable hosts (*e.g.*, Pang *et.al.* [20], Pouget *et.al.* [24]). In this paper, we show how such reconnaissance can be performed in a dynamic, fast, and evasive manner.

7 Summary

The use of passive network monitors has played an important role in a myriad of malware detection and containment studies to date. However, with the increased use of passive monitoring techniques, it is prudent to expect that attacks will soon evolve to minimize the practical benefits gained from such techniques. In this paper, we highlight the challenges posed by evasive techniques

that severely limit the view of the infection as recorded by collections of distributed network monitors. The techniques we present use lightweight sampling to detect passive network monitors as well as clusters of live network prefixes. We show the effectiveness of these evasive techniques through trace-based analysis and actual probing experiments conducted in the wild. Our experimental results verify our assertion that with a reasonably small number of probes, it is possible to accurately detect the locations of passive network monitors and to identify live address clusters containing the majority of the vulnerable population. We substantiate the threat from these techniques by outlining the design of evasive malware capable of evading extensive collections of network monitors, while saturating the vulnerable population in a matter of seconds. We hope that our results will stimulate the research community to develop monitoring infrastructures capable of countering these impending threats.

Acknowledgments

This work is supported in part by National Science Foundation grant SCI-0334108. We thank DShield for graciously providing access to their IDS logs. We also extend our gratitude to the reviewers for their insightful comments and feedback.

References

1. Michael Bailey, Evan Cooke, Farnam Jahanian, Jose Nazario, and David Watson. Internet motion sensor: A distributed blackhole monitoring system. In *Proceedings of the ISOC Network and Distributed System Security Symposium (NDSS)*, 2005.
2. Paul Barford, Rob Nowak, Rebecca Willet, and Vinod Yagneswaran. Toward a Model for Source Address of Internet Background Radiation. In *Proceedings of Passive and Active Measurement Conference (PAM 2006)*, March 2006.
3. John Bethencourt, Jason Franklin, , and Mary Vernon. Mapping Internet Sensors with Probe Response Attacks. In *Proceedings of the* 14^{th} *USENIX Security Symposium*, pages 193–212, August 2005.
4. Zesheng Chen, Lixin Gao, and Kevin Kwiat. Modeling the Spread of Active Worms. In *Proceedings of IEEE INFOCOMM*, volume 3, pages 1890 – 1900, 2003.
5. Zesheng Chen and Chuanyi Ji. A Self-Learning Worm Using Importance Scanning. In *Proceedings of ACM Workshop On Rapid Malcode (WORM)*, November 2005.
6. The Distributed Intrusion Detection System (DShield). See `http://www.dshield.org/`.
7. Xinwen Fu, Bryan Graham, Dan Cheng, Riccardo Bettati, and Wei Zhao. Camouflaging Virtual Honeypots. In *Texas A&M University technical report #2005-7-3*, 2005.
8. Thorsten Holz and Frederic Raynal. Defeating Honeypots. Online article, see `http://www.securityfocus.com/infocus/1826#ref3`.
9. Internet Assigned Numbers Authority (IANA). See `http://www.iana.org/`.
10. Internet Systems Consortium (ISC). See `http://www.isc.org`.
11. Vinod Yegneswaran Jonathon T. Giffin Paul Barford Somesh Jha. An architecture for generating semantic-aware signatures. In *Proceedings of the* 14^{th} *USENIX Security Symposium*, August 2005.
12. Hyang-Ah Kim and Brad Karp. Autograph: Toward automated, distributed worm signature detection. In *Proceedings of* 13^{th} *USENIX Security Symposium*, 2004.

13. Eddie Kohler, Jinyang Li, Vern Paxson, and Scott Shenker. Observed Structure of Addresses in IP Traffic. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, November 2002.
14. Christian Kreibich and Jon Crowcroft. Honeycomb—creating intrusion detection signatures using honeypots. In *Proceedings of* 2^{nd} *Workshop on Hot Topics in Networks (Hotnets-II)*, 2003.
15. Tom Liston, LaBrea Tarpit Project. See `http://labrea.sourceforge.net/`.
16. Justin Ma, Geoffrey Voelker, and Stefan Savage. Self-stopping worms. In *Proceedings of ACM Workshop On Rapid Malcode (WORM)*, pages 12–21, November 2005.
17. David Moore. Network Telescopes: Observing Small or Distant Security Events. In 11^{th} *USENIX Security Symposium, Invited Talk*, August 2002.
18. David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford, and Nicholas Weaver. Inside the Slammer Worm. *IEEE Magazine of Security and Privacy Magazine*, pages 33–39, July 2003.
19. David Moore, Colleen Shannon, Geoffrey M. Voelker, and Stefan Savage. Internet Quarantine: Requirements for Containing Self-Propagating Code. In *Proceedings of IEEE INFOCOM*, 2003.
20. Ruoming Pang, Vinod Yegneswaran, Paul Barford, Vern Paxson, and Larry Peterson. Characteristics of Internet Background Radiation. In *Proceedings of ACM SIGCOMM/USENIX Internet Measurement Conference (IMC)*, October 2004.
21. Larry Peterson, Tom Anderson, David Culler, and Timothy Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. *SIGCOMM Comput. Commun. Rev.*, 33(1):59–64, 2003.
22. Phillip Porras, Linda Briesemeister, Keith Skinner, Karl Levitt, Jeff Rowe, and Yu-Cheng Allen Ting. A hybrid quarantine defense. In *Proceedings of the Second ACM Workshop on Rapid Malcode (WORM)*, November 2004.
23. Fabien Pouget, Marc Dacier, and Van Hau Pham. Lurre.com: On the Advantages of Deploying a Large Scale Distributed Honeypot Platform. In *Proceeding of the E-Crime and Computer Conference ECCE*, March 2005.
24. Fabien Pouget, Marc Dacier, Van Hau Pham, and Herve Deber. Honeynets: Foundations for the development of early warning systems. In *NATO Advanced Research Workshop*, 2004.
25. Neil Provos. A virtual honeypot framework. In *Proceedings of the* 13^{th} *USENIX Security Symposium*, August 2004.
26. Moheeb Abu Rajab, Fabian Monrose, and Andreas Terzis. Fast and Evasive Attacks: Highlighting the challenges ahead. In *JHU Computer Science Technical Report HiNRG-RMT-112205*, November 2005.
27. Moheeb Abu Rajab, Fabian Monrose, and Andreas Terzis. On the Effectiveness of Distributed Worm Monitoring. In *Proceedings of the* 14^{th} *USENIX Security Symposium*, pages 225–237, August 2005.
28. Moheeb Abu Rajab, Fabian Monrose, and Andreas Terzis. Worm Evolution Tracking via Timing Analysis. In *Proceedings of ACM Workshop on Rapid Malware (WORM)*, pages 52–59, November 2005.
29. David Meyer, University of Oregon RouteViews Project. `http://www.routeviews.org/`.
30. Colleen Shannon and David Moore. The Spread of the Witty Worm. *IEEE Security and Privacy Magazine*, 2(4):46–50, July 2004.
31. Yoichi Shinoda, Ko Ikai, and Motomu Itoh. Vulnerabilities of Passive Internet Threat Monitors. In *Proceedings of the* 14^{th} *USENIX Security Symposium*, pages 209–224, August 2005.

32. Stuart Staniford, David Moore, Vern Paxson, and Nick Weaver. The Top Speed of Flash Worms. In *Proceedings of the ACM Workshop on Rapid Malcode (WORM)*, pages 33–42, October 2004.
33. Stuart Staniford, Vern Paxson, and Nicholas Weaver. How to 0wn the internet in your spare time. In *Proceedings of the* 11^{th} *USENIX Security Symposium*, August 2002.
34. George Varghese Sumeet Singh, Cristian Estan and Stefan Savage. Automated worm fingerprinting. In *Proceedings of* 6^{th} *Symposium on Operating System Design and Implmentation (OSDI)*, 2004.
35. Michael Vrable, Justin Ma, Jay Chen, David Moore, Erik Vandekieft, Alex C. Snoeren, Geoffrey M. Voelker, and Stefan Savage. Scalability, Fidelity and Containment in the Potemkin Virtual Honeyfarm. *Proceedings of ACM SIGOPS Operating System Review*, 39(5):148–162, 2005.
36. Vinod Yegneswaran, Paul Barford, and Somesh Jha. Global intrusion detection in the domino overlay system. In *Proceedings of the ISOC Network and Distributed Systems Security Symposium (NDSS)*, 2004.
37. Vinod Yegneswaran, Paul Barford, and David Plonka. On the Design and Use of Internet Sinks for Network Abuse Monitoring. In *Proceedings of the Symposium on Recent Advances in Intrusion Detection (RAID)*, Sept. 2004.
38. Amgad Zeitoun and Sugih Jamin. Rapid Exploration of Internet Live Address Space Using Optimal Discovery Path. In *Proceedings of Globecomm*, 2003.