Hotspots: The Root Causes of Non-Uniformity in Self-Propagating Malware

Evan Cooke, Z. Morley Mao, Farnam Jahanian
Department of Electrical Engineering and Computer Science
University of Michigan
{*emcooke, zmao, farnam*}*@umich.edu*

Abstract

Self-propagating malware like worms and bots can dramatically impact the availability and reliability of the Internet. Techniques for the detection and mitigation of Internet threats using content prevalence and scan detectors are based on assumptions of how threats propagate. Some of these assumptions have recently been called into question by observations of huge discrepancies in the quantity of specific threats detected at different points around the Internet. We call these deviations from uniform propagation "hotspots". This paper quantifies and explains these influences on malware propagation. We then propose that hotspots can be explained by two fundamental influences on propagation: algorithmic factors and environmental factors. We use measurement data from sensors deployed at 11 locations around the Internet to demonstrate the impact of these factors on worm and bot propagation. With this understanding, we simulate the outbreak of new threats with hotspots and show how algorithmic and environmental factors reduce the visibility of distributed detectors resulting in the inability to identify new threats.

1. Introduction

The construction of effective detection and mitigation systems to combat increasingly sophisticated Internet attacks requires a deep understanding of malicious Internet behavior. To date, the analysis of Internet threats like worms has focused largely on more salient features like the scanning algorithm [22, 27]. More recently, there has been increased attention on other features that impact threat propagation like properties of the vulnerable population [13, 19] and bot targeting behavior [6].

Previous studies often assumed that worms and other threats had mostly uniform targeting distributions [29, 23]. That is, there was an equal probability a threat would attempt to infect one address as any other address. However, we are learning that Internet threats have highly non-uniform propagation behavior. Several studies have provided empirical evidence suggesting that there are a set of internal and external processes that bias propagation of Internet threats [18, 5, 13].

This paper quantifies and explains these influences on propagation. We define *hotspots* as deviation from uniform malware propagation. We then hypothesize that hotspots can be explained by two fundamental influences on propagation. *Algorithmic factors* describe host-level and programmatic characteristics that impact threat propagation. *Environmental factors* describe external influences such as routing and filtering policy, network failures, misconfigurations, and network topology that impact threat propagation.

To validate the existence of hotspots and demonstrate their influence on propagation we use measurement data from Internet sensors deployed at 11 locations around the Internet. We illustrate the impact of algorithmic factors by showing how the use of hit-lists by bots and pseudo random number generator flaws in the Slammer and Blaster worms produce hotspots. We then demonstrate the impact of environmental factors by showing how the widespread use of private address space interacts with CodeRedII propagation and subsequently how the use of filtering in enterprises blocks outgoing worm probes and produces hotspots. These observations empirically demonstrate the existence and importance of hotspots in real-world threat propagation.

With an empirical understanding of hotspots, we present evidence that hotspots have serious implications on distributed detection systems. We argue that alerts from systems such as those based on prevalence [11, 12, 24] can be highly inaccurate in the face of hotspots. The core issue is that detection systems placed at different points around the Internet can report widely different observations of a particular threat. The result is that quorum-based detection algorithms can completely miss the outbreak of new threats. To demonstrate the impact of hotspots we simulate the outbreak of a new threat with hotspots caused by algorithmic and environmental influences. The results illustrate an inherent limitation in globally-scoped detection systems. We argue that the existence of hotspots demonstrates that a global detection system alone is not sufficient and local detection capabilities are still essential for detection.

This paper is organized as follows. §2 provides a background on malware propagation and discusses related work. In §3 we define hotspots and hypothesize that hotspots can be explained by algorithmic and environmental factors. Next, in §4 we provide empirical evidence of algorithmic and environmental factors. With this understanding, in §5 we simulate the outbreak of new threats with hotspots and discuss the implications on distributed detection.

2. Background and Related Work

There has been a significant effort in the research community to measure and model self-propagating Internet threats [3, 19, 27]. There are three types of host populations: a *vulnerable* host population, an *infected* host population, and an *immune* host population. A host moves from the vulnerable population to the infected population after a successful infection attempt, and a host can only be a member of one population at a time. *Propagation* describes this process in terms of how a host in the infected population chooses and infects a vulnerable host.

There are two basic steps in the propagation process, choosing an address to infect and attempting to infect the host at that address. Many threats do not know which hosts are vulnerable ahead of time so they will randomly scan through address space. This process of choosing the address of the next target has traditionally been modeled as a uniform process. For example, in the simple epidemic model every possible IPv4 address has an equal probability of being the next target for an infection attempt [27]. That is, a worm instance chooses the next target address from a uniform random distribution from 0 to 2^{32} in the IPv4 space. We use uniform random propagation as the baseline upon which our model of hotspots is constructed.

While many of the most significant worms to strike the Internet such as CodeRed [22], Slammer [16] and Blaster [2] use a random number generator in the target address generation process, that process is highly non-uniform. For example, CodeRedII and Nimba have been shown to scan nearby addresses with a higher probability [3, 19]. Moore *et al.* also pointed out how a flawed random number generator in the Slammer worm could lead to preference for certain addresses [16].

There is also significant empirical evidence of this non-uniform targeting behavior. Cooke *et al.* demonstrated that distinct darknet monitors (unused address space sensors) observed orders-of-magnitude different amounts of traffic and different numbers of unique source IPs [5]. These differences persisted even when local preference and specific propagation algorithms were accounted for. Pang *et al.* also showed that data collected at darknets at three locations belonging to three distinct networks differed significantly [18]. Specifically, they found that the number and type of packets per IP address across different ports and protocols, number of unique source IP addresses, and temporal characteristics of traffic patterns also differed. Kumar *et al.* recently illustrated how the Witty worm's random number generator produces non-uniform scanning [13].

These observations clearly show that the targeting process of malware on the Internet today cannot be described by a uniform process. Furthermore, simple explanations like local preference and octet-based targeting don't account for the large observed differences between addresses [3, 5, 19]. This paper systematically explains the root causes of these non-uniform processes and then explores their impact on the structure of detection systems.

3. Defining Hotspots

In this section we develop a definition of *hotspots* encompassing internal, external, host, and network influences on threat propagation. Simply put, hotspots are deviations from uniform propagation behavior. For example, a worm that targets a subset of possible addresses (e.g., a /24) is said to exhibit hotspots while a worm that prefers every possible destination address with equal probability does not.

We can decompose the causes of hotspots into two major classes. *Algorithmic factors* are programmatic and host-centric features such as hit-lists, poorly constructed and seeded PRNGs, and local preferences that bias propagation. *Environmental factors* are external influences such as routing and filtering policy, network failures and misconfigurations, and network topology that impact reliability and reachability and thus threat propagation. It is important to highlight that there is no notion of intentionality in algorithmic or environmental factors. Certain hotspots may be intended and part of the threat specification, while other hotspots are unintended and are a consequence of ambiguous specification or incorrect implementation.

3.1. Algorithmic Factors

Algorithmic factors are host-centric, programmatic characteristics that impact threat propagation. They include the actual malware algorithms, the local operating system, and run-time services on vulnerable and infected hosts. Algorithmic factors are a superset of scanning strategies. Worm scanning strategies such as those investigated by Staniford *et al.* [27] including hit-list scanning, permutation scanning, topological scanning, and stealth scanning can cause hotspots, however, algorithmic factors go beyond scanning strategies and include host context such as sources of entropy and other run-time variables. We now describe three key classes of algorithmic factors:

Hit-lists are pre-programmed lists of target addresses stored in the threat payload or obtained remotely from a server. Hit-lists are used by worms and bots to target specific address ranges. For example, a hit-list can be used to speed up the propagation of worms [26] or help avoid

known detection systems. Hit-lists can have a very narrow focus (e.g., target a specific /24 subnet), or specify almost the entire address space (e.g., scan all the IPv4 space except for 127.0.0.0/8). Hit-lists are often used by bots to target specific address ranges known to contain live hosts such as academic networks [6].

Pseudo-Random Number Generators (PRNGs) are often used in the target address selection in self-propagating threats. Obtaining a uniform random number distribution from a PRNG is challenging and the output of many common PRNGs is biased. PRNGs require a source of entropy to seed the generation function, and a poor quality entropy source can severely bias the PRNG output. Thus, a bad PRNG, a bad PRNG implementation, or a poor source of entropy can cause non-uniform threat propagation.

Local Preference biases the targeting of a threat toward nearby addresses and is typically deliberately designed into a propagation algorithm. When the addresses of vulnerable systems are clustered, local preference can spread the infection faster [19]. Local context such as the network address, hardware address, system fingerprint, or global unique identifier can all be used to define locality.

3.2. Environmental Factors

Environmental factors are external influences that impact the propagation of a threat. Environmental factors derive primarily from network conditions along the end-to-end path between an infected host and its target. These factors impact the reachability of the target addresses due to failures, misconfiguration, and policy restrictions along the path. The structure and composition of network nodes such as link capacities, link latencies, and host addressing schemes are all considered environmental factors. We now describe three key classes of environmental factors:

Routing and filtering policy impact the reachability and path taken by infection packets. For example, a firewall at the egress router of a network can restrict the targets reachable by an infected host. A firewall in front of vulnerable but not-yet-infected hosts can also prevent those hosts from being infected by an external attacker but still leave them vulnerable to infected hosts within the same network.

Network failures and misconfigurations impact the reliability of infection packets reaching a destination. Dropped and mangled packets can significantly impact the probability of a successful infection. Failures can occur due to network equipment problems, link connectivity issues, or simply network congestion (which can be self-induced by the outbreak [16]). Misconfigured routers, switches, and policy devices can also cause unreliability.

Network topology impacts how nodes on the network are configured and connected. It determines message latency and bandwidth and thus the rate at which an infection can progress. The assigned addresses and the rules that govern the reachability between addresses can also significantly impact threat propagation. For example, private addresses [20] are widely used within networks and behind network address translation devices (NATs). Vulnerable hosts within private address space have more reachability constraints compared to hosts with globally advertised addresses.

4. Empirical Evidence of Hotspots

To demonstrate how algorithmic and environmental factors impact real-world malware propagation we now describe a series of case studies performed using Internet measurement data. We study the Blaster worm, the Slammer worm, the CodeRedII worm, and live botnets to empirically demonstrate the existence and severity of hotspots.

4.1. Measurement Methodology

The data analyzed in this paper is from two sources: darknets and live network capture. The darknet data was collected over two months in 2004 and 2005 as part of the Internet Motion Sensor distributed darknet monitoring project [1]. Darknets are blocks of unused address space. Any traffic observed at darknets must be the result of misconfiguration, backscatter from spoofed source addresses, or scanning from worms and other network probing. Darknets are also known by other names such as network telescopes [15], blackholes [1, 25]. These efforts have produced a new understanding of denial of service attacks [17], worms [1, 2, 16, 22], and malicious behavior [18].

The IMS darknet data used in this study was from 11 distinct address blocks at 9 organizations including ISPs, academic networks, and a large enterprise. The address blocks ranged in size from a /25 (CIDR notation, 128 addresses) to a /8 (16 million addresses). Throughout the paper we refer to these blocks by their anonymized label which also indicates the size of the block: (A/23, B/24, C/24, D/20, E/21, F/22, G/25, H/18, I/17, M/22, Z/8). The IMS sensor recording the traffic at each of these blocks actively responded to TCP SYN packets with a SYN-ACK packet to elicit the first data payload on all TCP streams [1]. This approach provided the necessary payload data to uniquely identify the threats studied in this paper.

In addition to darknet data, we also used data captured from a large live network to investigate bot propagation. We looked for the specific command signatures of Agobot/Phatbot [4], rBot/SDBot [14], and Ghost-Bot in the payload of traffic captured in a large academic network. The academic network was allocated a single /15 address block and included approximately 10,000 live hosts.

4.2. Algorithmic Factors

In this section we empirically demonstrate how algorithmic factors influence real-world malware propagation. We

Bot Propagation Command	Bot Propagation Command
ipscan i.i.i.i dcom2 -s	advscan wkssvcENG 100 0 0
ipscan s.s.s.s dcom2 -s	ipscan r.r.r.r dcom2 -s
advscan dcass 300 5 0 141.x.x.x	advscan lsass 100 5 999 -b
advscan dcass 300 5 0 140.142.x.x	ipscan 69.27.s.s dcom2 -s
ipscan s.s mssql2000 -s	ipscan s.s.s lsass -s
ipscan s.s webdav3 -s	ipscan r.r.r.r dcom2 -s
ipscan 194.s.s.s dcom2 -s	ipscan 194.116.s.s dcom2
ipscan 192.s.s.s dcom2 -s	ipscan 128.s.s.s dcom2 -s

Table 1. Botnet scan commands captured on a live /15 academic network.

show how hit-lists cause hotspots using data from botnets and show how bots target specific networks. Next, we illustrate how errors and bad parameters in random number generators cause hotspots in propagation using the Blaster and Slammer worms as examples.

4.2.1. Hit-lists: botnet targeted attacks

Hit-lists are a simple and prevalent cause of hotspots. To demonstrate how they impact real-world malware propagation we investigated the behavior of bots. Bots have recently gained attention for their flexible design and ability to perform targeted attacks against specific subnets [6, 28].

Bots typically wait for commands from a bot controller to initiate propagation. These commands can be intercepted and analyzed. To gather bot propagation commands we looked for the specific command signatures of Agobot/Phatbot [4], rBot/SDBot [14], and Ghost-Bot in the payloads of traffic captured in a large academic network. We then extracted the specific parts of the commands instructing bots to start propagating. For example, `advscan dcom135 300 5 9000 10.x.x.x -r -b -s` is a bot command we captured as it was sent through IRC.

Table 1 shows a sample of the commands from approximately 11 bots detected by the system during a month in 2005. Each command instructs the bot to begin scanning a range of IP addresses. The bot commands show that hit-lists are used by malware today to restrict propagation to certain subnets. It is hard to determine if the motivation for the targeted behavior is to avoid detection or increase the probability of infections, however, we can say that hit-lists are used by botnets today.

4.2.2. PRNG: Bad entropy in the Blaster worm

In this subsection we empirically show how a poorly seeded random number generator impacts malware propagation and produces hotspots. The Blaster worm uses the system clock on the infected computer to initialize its random number generator. The system clock is a bad source of entropy and we empirically demonstrate how hotspots in Blaster propagation can be directly correlated with system clocks and the time since the last reboot.

The Blaster worm uses the Windows *GetTickCount()* system call as a source of entropy for its pseudo-random number generator (PRNG) [2]. The system clock is a very

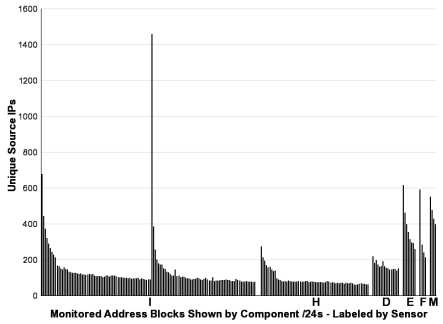


Figure 1. Observed unique source IPs of Blaster infection attempts by /24.

poor source of entropy because the count starts from 0 after each reboot. Because the Blaster executable is started automatically at boot time, the initial seed for the PRNG of a Blaster instance started as the result of a reboot will be restricted to a small subset of the possible 32-bit values of the clock.

To measure the number of clock ticks it takes to launch the worm after a reboot we wrote a simple program that called *GetTickCount()* and logged the result to a file. The program was launched at boot time using the same registry-based launching mechanism as the Blaster worm. The program then instructed the computer to reboot, and the process repeated.

We gathered tick count distributions from three generations of Intel-based systems: a Pentium II, a Pentium III, and a Pentium IV. The results revealed a mean boot time of about 30 seconds with a 1 second standard deviation. Thus, there was very restricted range of possible boot times for each generation of hardware. A more detailed analysis can be found in [7].

The restricted range of possible initial seeds means that the resulting Blaster scanning behavior should be biased. Figure 1 shows the distribution of persistently infected Blaster hosts observed at distributed IMS sensors over a month in August 2004. Hotspots are clearly visible in the middle of the I sensor block. The question is whether these hotspots can be correlated with probable initial seeds for the Blaster PRNG.

Using the decompiled Blaster source code [21] and a range of possible tick count values from *1000* to *10,000,000*, (i.e., boot times ranging from 1 second to 2.8 hours since 1 tick is a millisecond) we generated a mapping from seeds to IP addresses. Using this mapping we correlated the address ranges that observed the most Blaster sources in Figure 1 with initial seeds. The large spike in observed Blaster hosts at the I block maps back to a seed value corresponding to a *GetTickCount()* of 2.3 minutes.

Using the seed-to-target mapping, the other spikes in Figure 1 were mapped back to possible seeds. Resulting

seed values ranged from about approximately 1 minute to 20 minutes with the distribution centered around 4-5 minutes. To cross check the results we also mapped the addresses that observed very few Blaster hosts back to initial seeds and found they corresponded to improbable boot times of hours to days. These values confirm our earlier observations.

The evidence presented on Blaster propagation strongly suggests that the worm is heavily influenced by a poor source of entropy used to seed the PRNG. Because the Blaster worm continues to sequentially scan through the IPv4 address space after choosing an initial starting point (based on the PRNG), the effect of hotspots is not as pronounced as it could be. Nevertheless, the data suggests hotspots generated by algorithmic factors like errors in the PRNG implementation are real and can significantly impact propagation.

4.2.3. PRNG: Error in Slammer worm PRNG

In this subsection we extend our understanding of the impact of algorithmic factors and the importance of the PRNG by showing how a poorly designed generator function can cause significant hotspots. To demonstrate the importance of PRNG flaws we use empirical data on Slammer worm propagation to definitively show a significant targeting bias by Slammer infected hosts. To our knowledge, this is the first real evidence that the flaw found in the Slammer [16] worm actually impacts real-world propagation.

The Slammer target generation algorithm is a simple linear congruent generator (LCG) [10] of the form $s(i+1) = a * s(i) + b \mod p$. The worm chooses an IPv4 address using this simple PRNG and then attempts to infect the targeted address with a single UDP packet. The generator contains a serious flaw that impacts the randomness of the resulting number stream.

In the case of Slammer, the value of a is `214013` and p is 2^{32} because the value is stored as a 32 bit integer. The b parameter however is not fixed. Although it may have been the intention of the worm author to fix the value of b at `0xffd9613c` (which is a commonly used value of b in many LCGs), it appears the author made an error and used the `OR` instruction instead of `XOR` to clear a register. The result is that `0xffd9613c` becomes `OR`'ed with a value left over in the `ebx` register. This leftover value is the `sqlsort.dll` Import Address Table entry which can vary with the version of the DLL. Three versions have been widely reported (`0x77f8313c`, `0x77e89b18`, and `0x77ea094c`) [8, 16].

In order to determine the value of b in the Slammer LCG we take the three possible leftover `ebx` register values and `XOR` them with `0xffd9613c`. Thus, the possible values of b (more if other DLL versions exist) are `0x88215000`, `0x8831fa24`, `0x88336870`. Unlike the value of b that appears to have been originally intended, these b values are not optimal for producing the greatest possible range of random numbers. The result is that certain choices of the initial seed may cause the LCG to loop over a small subset of the possible 32-bit values. The implications of these imperfections are enormous for Slammer targeting behavior. If the choice of initial seed forces Slammer into a small cycle with period of less than 1000, the resulting target distribution will be extremely restricted.

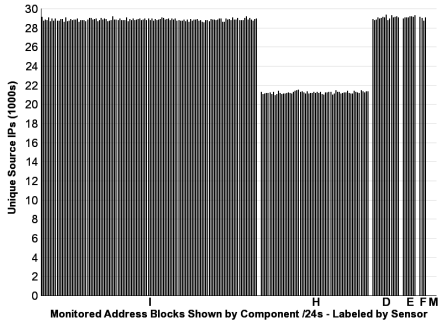
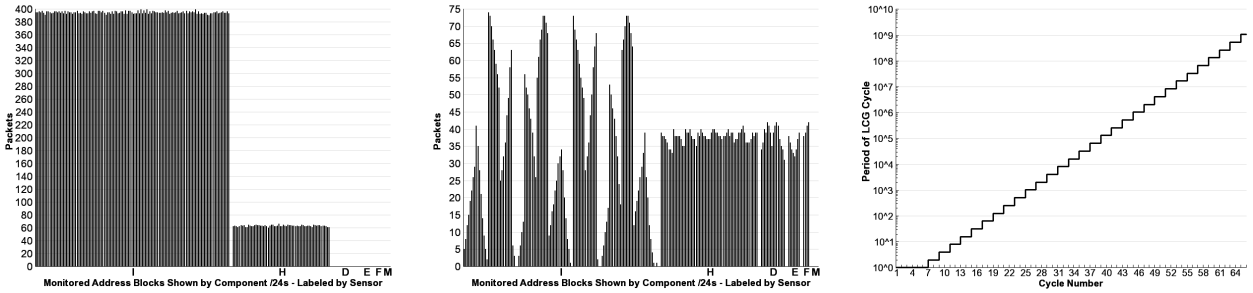


Figure 2. Observed unique Slammer infected source IPs by destination /24s.

Figure 2 shows the number of observed Slammer infection attempts from the IMS by /24. There are a few important observations. First, the M block did not see any Slammer infection attempts. This is due to policy blocking the worm deployed at its upstream provider. Second, the H block shows almost 8000 fewer Slammer sources than the other blocks. Recall that these observations were made over a period longer than one month so temporal effects (especially considering Slammer's fast propagation) are minimized.

Based on the flaws in the Slammer PRNG described above, it is possible to make certain testable predictions about Slammer scanning patterns. If a significant number of Slammer hosts chose a bad initial seed, they will scan a specific subset of the IPv4 space. To test this hypothesis we tracked individual Slammer infected IPs across IMS sensors. Host A in Figure 3 shows the number of Slammer infection attempts by /24 from a single Slammer source IP. Notice how block D observed no infection attempts from this particular source while block H observed some and block I received the most. Host B in Figure 3 shows another unique Slammer source. In this figure the intra-block variance is quite high and there is a distinct pattern. The important implication of Figure 3 is that there can exist huge non-uniformity in the scanning patterns of individual Slammer infected hosts.

The hotspots exhibited by the hosts in Figure 3 appear to indicate that Slammer infected hosts can get into PRNG cycles. To identify the PRNG cycles in Slammer we compute the length of all cycles of the LCG described above for each



(a) Slammer Host A (b) Slammer Host B (c) Possible LCG cycles

Figure 3. (a, b) Slammer infection attempts from two unique hosts by destination /24s. (c) Period of all possible cycles in the Slammer LCG for *b* of `0x88215000`.

value of *b*. We find that there are 64 cycles for each *b* value and the lengths are very similar in each case. Cycle lengths for *b* equal to `0x88215000` are shown in Figure 3(c). Notice that the log plot shows many small cycles and seven cycles having a period of only one. A Slammer instance stuck in one of these short cycles will repeatedly attempt to infected a handful of addresses appearing very much like a targeted denial of service attack.

Because certain addresses can be part of longer PRNG cycles and newly infected hosts have a higher probability of being a member of longer cycles, certain addresses should observe more unique Slammer source addresses. This means we can predict the relative number of Slammer observations at different addresses based on the length of the PRNG cycles that traverse each address.

Using the logic above, the H block should have fewer long cycles than the other blocks because the distribution of unique Slammer hosts depicted in Figure 2 shows a clear bias away from the H block. We can check this by computing the total length of all cycles that traverse each block. The sum of the lengths of the cycles for the D, H, I blocks are 42.67, 29.33, 42.67, (divided by 16^{10}) respectively. This shows that the H block is traversed by far fewer long PRNG cycles than the D or I block confirming the prediction.

This analysis has demonstrated that there are two forms of hotspots in the Slammer worm. First, each individual Slammer instance enters a PRNG cycle that is a subset of 32-bit space and thus propagation from individual Slammer infected hosts can have a significant bias. The second class of hotspots emerges when observing Slammer infected hosts in aggregate. Because there is a higher probability to enter longer cycles, address ranges that are not part of as many longer cycles do not observe as many Slammer hosts. These results have demonstrated that flaws and problems with PRNGs are algorithmic factors that can cause significant hotspots in malware propagation.

4.3. Environmental Factors

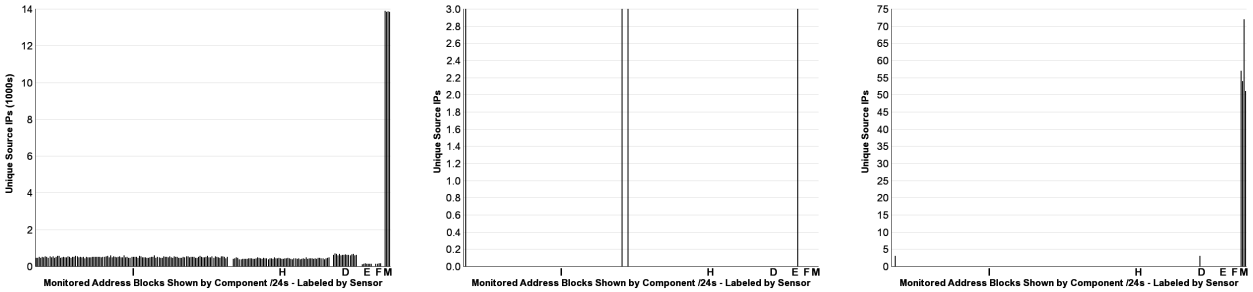
In this section we empirically show how environmental factors influence real-world malware propagation. We show how Internet topology and the widespread use of private address space is an environmental factor that causes huge hotspots in CodeRedII propagation. Subsequently, we show how Internet filtering impacts the propagation of the CodeRedII, Slammer, and Blaster worms.

4.3.1. Network topology: NATs & CodeRedII

Network topology is an environment factor that has a large influence on propagation. One important topological feature is the wide adoption of private address space inside homes and enterprises spurred by the use of NATs [9]. An implication of this feature is the significant loss of bidirectional reachability for hosts on the Internet. This has implications on how vulnerable hosts become infected and how infected hosts propagate.

To investigate how private address space influences malware propagation we analyzed the targeting behavior of the CodeRedII worm. CodeRedII has a large local preference in its propagation algorithm which means that infections can progress much more quickly in clusters of hosts in nearby address space. This also means that when a CodeRedII infected host is assigned a private address like 192.168.1.2 behind a NAT device, the infected host will continue to prefer the local /8 (192.0.0.0/8) and the local /16 (192.168.0.0/16). Since 192.168.0.0/16 is the only private /16 in 192.0.0.0/8, CodeRedII infection packets to other /16 networks in 192.0.0.0/8 will leak out into the Internet and produce a large hotspot.

To investigate CodeRedII hotspots we measured CodeRedII infection attempts at IMS sensors for more than a month. Figure 4(a) shows the number of unique CodeRedII source IP addresses detected by /24. The distribution is clearly not uniform and there is a large hotspot in the M block. The propagation distributions from individual CodeRedII infected hosts reveal two classes of behavior: a



(a) Observed CodeRedII (b) Quarantine *non*-192/8 Address (c) Quarantine 192/8 Address

Figure 4. (a) Observed unique CodeRedII source IPs by destination /24s. (b, c) Infection attempts by destination /24s from two quarantined CodeRedII hosts.

uniform scanning behavior, and a scanning behavior with a large bias for the M block as illustrated in Figure 4(a).

A hint about the origin of this abnormal host behavior is the address of the M block. The M block is located inside 192.0.0.0/8 which supports the hypothesis that many CodeRedII infected hosts are behind NATs in 192.168.0.0/16 private address space and thus prefer the 192.0.0.0/8 address block. The challenge is that the hypothesis is difficult to test without tracking down each actual infected machine. Instead, we attempted to replicate the conditions of a CodeRedII host behind a NAT and then monitored the scanning preference. Using a honeypot running VMWare, the CodeRedII worm was captured and placed in a controlled environment with sensors in the same addresses as the sensors in Figure 4(a).

Using the captured worm we performed two experiments. First, we configured the infected VMWare host with an address outside 192.0.0.0/8. Figure 4(b) shows the scan targets plotted over the same /24s as in Figure 4(a). Even though a total of 7,567,093 infection attempts were recorded from the quarantined host, only a small number of attempts reach the monitored blocks. The CodeRedII worm has a very large local preference and a completely random target address is chosen only 12.5% of the time so packets to the sensors were very rare.

In the second experiment, the vulnerable host running under VMWare was configured with the IP address 192.168.1.50. Figure 4(c) shows the scan targets plotted over the same /24s as in Figure 4(a). During this run, a total of 7,567,361 infection attempts were recorded from the infected host. However, this time, the graph shows a distinct spike at the M block just like the distribution observed on the IMS darknets. While the number of samples is not huge, there is clearly a significant effect due to local preference.

These two experiments demonstrate a strong correlation between the scanning behavior of a CodeRedII infected host behind a NAT device and the empirical observations of CodeRedII propagation. The implication is that network topology is an environmental factor that can dramatically influence propagation and result in large hotspots.

FT100-Corp	Total IPs	CRII IPs	Slammer IPs	Blaster IPs
Corp-Banking	6129672	1	68	16
Corp-Media	1273256	9	2	67
Corp-Logistics	8985520	0	5	44
Broadband ISP	**Total IPs**	**CRII IPs**	**Slammer IPs**	**Blaster IPs**
ISP-A	5336880	40251	88165	20517
ISP-B	19501064	2277	26261	3285
ISP-C	19448864	1340	5798	28

Table 2. The top 3 Fortune 100 enterprises and top 3 broadband ISPs with worm infections detected by IMS.

4.3.2. Network filtering: Fortune 100 filtering

Filtering on the Internet has become pervasive. The use of routing policy and filtering devices like firewalls are common on most networks. We now demonstrate how filtering is an important environmental factor that causes hotspots.

Large companies typically have hundreds of thousands of hosts in their networks. The size and complexity of these networks means stamping out all infections is nearly impossible. The result is that there will inevitably be a number of infected hosts inside large enterprise networks that will make outgoing infection attempts. We can use the outgoing infection attempts made by these infected hosts to estimate the prevalence of filtering.

To measure the number of infections from large enterprises, we can compare the list of worm infected IPs detected by the IMS to the addresses managed by Fortune 100 companies. To find the addresses of large enterprises we took the companies in the Fortune 100 during 2004 and looked up the IP addreses allocated to each enterprise by ARIN. We removed Tier 1 ISPs and broadband providers from the list because their addresses can be reallocated to other organizations. We then tested to see if the persistently infected IPs from the CodeRedII, Slammer, and Blaster worms were present in the IP ranges allocated to the Fortune 100 companies.

The results are shown in Table 2. Despite the size of the companies and the huge number of addresses they man-

age, there were almost no external indication of infections. To check these results we also studied the top 3 broadband providers and looked at the number of infections leaking from their networks. The results are also shown in Table 2 and reveal 10's of thousands of infections from the broadband providers. Since there is little outgoing filtering from broadband providers today, these results support the idea that there is outgoing filtering at enterprises. While not conclusive because enterprises could be exceptionally good at patching, it does highlight how filtering can result in hotspots when threats such as CodeRedII use local context to bias propagation.

5. Impact of Hotspots on Distributed Detection

The existence of significant hotspots in malware propagation – as evidenced by the Slammer, CodeRedII, and Blaster worms – has significant implications on distributed detection systems. One important issue is that alerts from systems such as those based on content prevalence [12, 24, 11] can be inaccurate due to hotspots. The problem is that one IP address may observe a large number of infection attempts while another address observes few or no infection attempts. The result is that detection systems placed at different points around the Internet may report widely different observations of a particular threat.

We now investigate how hotspots in malware propagation impact detection systems. We analyze two processes that could significantly increase the prevalence of hotspots: (1) the move to more targeted attacks as illustrated by the rise of botnets, and, (2) the continuing loss of bi-directional connectivity due to greater use of NATs, private address space, and filtering at the customer edge.

In this section we demonstrate how hotspots generated by algorithmic and environmental factors consistent with these two trends reduce the effectiveness of distributed malware detection. To illustrate the influence of algorithmic factors we simulate the outbreak of a new threat that uses hit-lists and investigate the impact on distributed detection systems. Next, we simulate the outbreak of a threat that uses local preference inside private address space and demonstrate the impact of environmental factors on distributed detection. Because hotspots are inherently difficult to predict, these results illustrate the danger hotspots pose to distributed detection systems.

5.1. Simulation Platform

To model hotspots we constructed a simulation environment to reproduce the outbreak of a CodeRedII-type worm. The platform was designed to model a uniform scanning worm, a hit-list worm, and the internal propagation algorithm of the CodeRedII worm. To ensure that the CodeRedII algorithm was accurate, we used the disassembled CodeRedII code to construct the propagation code. This includes how the random generators were seeded and how the target addresses were chosen. To provide comparable results to [12], we fixed the scanning rate at 10 probes/second and used a randomly chosen seed population of 25 hosts for each worm. The vulnerable population for each worm was set to the actual infected IPs as discussed in Section 4. The vulnerable population for CodeRedII worm included 134,586 unique addresses that were clustered in 47 /8 networks.

5.2. Algorithmic Factors and Detection

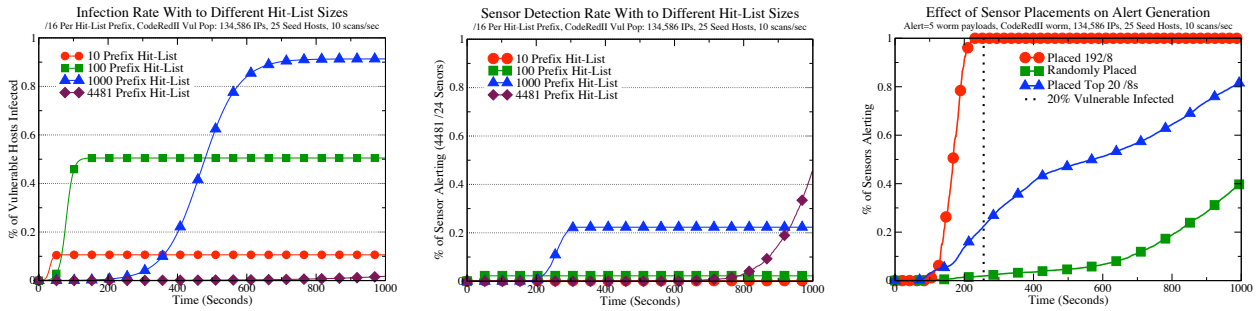
In this subsection we demonstrate how hotspots due to algorithmic factors impact distributed detection systems. To illustrate the point we investigate how hit-lists, an algorithmic factor increasingly used by bots, produce hotspots. We use the observation that bots will often target specific /24 and /16 networks described in Section 4.2 as a basis for a simulation. Using this information we built a worm that uses a list of prefixes that specify /16 IPv4 networks as targets. Each newly infected host may only propagate to the addresses covered by the prefixes in the hit-list.

In the first experiment we investigate the impact of different lengths of hit-lists on the threat propagation. We used 4 hit-list sizes: 10 /16's, 100 /16 networks, 1000 /16's, and 4481 /16's. Each /16 was chosen to cover as many remaining vulnerable hosts as possible. The 10-item list covered 10.60% of the vulnerable population, the 100-item list covered 50.49%, the 1000-item list covered 91.33%, and the 4481-item list covered 100%.

Figure 5(a) shows the result of the first experiment. The threat using the smallest hit-list infected all the possible hosts in its hit-list quickest. This is due to the higher vulnerable population density. However, the threat using the larger hit-list reached a larger percentage of the total vulnerable population but did so more slowly. The result demonstrates the importance of a high ratio of vulnerable addresses to total addresses for speed [27, 26].

With an understanding of the propagation rates, the next step is to model the impact of hit-lists on detection. In this experiment we randomly placed a /24 detector (i.e., each detector monitored 256 contiguous addresses) in each of the 4481 /16 networks with at least one vulnerable host. Each sensor was set to generate an alert after observing n worm infection attempts (similar to many network-based detection systems [12, 24]). Our detector had no false positives and was set to generate an alert after observing 5 threat payloads. We then released the same hit-list-based threat and observed the alert rate across all the sensors.

Figure 5(b) shows the result of the hit-list detection experiment. It shows that even if all the sensors had no false positives, and were able to instantaneously communicate with each other, a quorum-based alerting approach would likely never alert as very few of the sensors ever alert. When



(a) Hit-list Infection (b) Hit-list Detection (c) Detection & NATs

Figure 5. (a, b) Infection and detection rate during the simulated outbreak of a new threat using hit-lists of different lengths. (c) Detection rate with vulnerable hosts placed in private address space using different sensor placement strategies.

more than 90% of the vulnerable population has been infected, only slightly more than 20% of the detectors have alerted. This is a important result. It means that even with pre-knowledge of the vulnerable population, and the ubiquitous placement of detectors, hotspots caused by algorithmic factors reduce visibility of distributed detectors.

5.3. Environmental Factors and Detection

In this subsection we demonstrate how hotspots caused by environmental factors impact distributed detection systems. To illustrate the point we investigate how the increasingly complex topology of the Internet contributes to hotspots and creates new challenges.

The large increase in broadband subscribers has helped drive fundamental changes in the structure of the Internet. In particular, the wide adoption of NAT devices and the increasing use of private addresses has reduced the ability to make bidirectional connections. These environmental factors have serious implications on hotspots and detection.

To illustrate how changes in Internet topology create hotspots and detection challenges we simulated the outbreak of a new threat in an Internet environment with private address space. The threat was based on the decompiled CodeRedII code with the same local bias.

To model the private address usage among vulnerable hosts on the Internet we estimated the number of hosts inside 192.168.0.0/16 in the real CodeRedII vulnerable population. Using the data plotted in Figure 4(a), we were able to compute an estimate. By comparing the number of hosts observed with source addresses in 192.0.0.0/8 with the total number of hosts observed with any address we can estimate the number of hosts with 192.168.0.0/16 private addresses at 15%. This is a crude estimate, as it may significantly underestimate the real percentage of vulnerable hosts inside private address space. Using the same simulation environment as before, we configured 15% of vulnerable hosts as if they were NAT'ed with 192.168.0.0/16 addresses.

In the first run of the experiment we placed 10,000 /24 sensors randomly throughout the IPv4 space and released the CodeRedII-type worm. Figure 5(c) shows the resulting alert rate as a function of time. The figure shows that it takes more than 11 minutes for just 10% of the sensors to generate an alert. After 11 minutes the worm has already infected more than 50% of the vulnerable population making global containment difficult or impossible.

For the second run of the experiment we assumed that organizations around the Internet were able to collaboratively determine where potentially vulnerable hosts were located and we placed 10,000 sensors randomly inside the top 20 /8 networks with vulnerable hosts. Because the vulnerable population was clustered inside certain networks (the top 20 /8 networks include 94% of the vulnerable population), there was a better chance of detecting the outbreak because the sensors were nearby and the threat had a local preference. Figure 5(c) shows the resulting alert rate when the sensor were placed in the top 20 /8 networks. The figure illustrates that placement closer to the vulnerable population results in faster detection, however, only 20% of the sensor have alerted when 20% of the vulnerable population has been infected.

In the third experiment we tested whether the empirically measured hotspot detected in 192.0.0.0/8 could be leveraged for detection. We placed 255 sensors in each of the /16 networks inside 192.0.0.0/8 avoiding 192.168.0.0/16. The result is plotted in Figure 5(c). Every single sensor generated an alert before the worm has infected 20% of the vulnerable population. Thus, even with only 15% of the vulnerable population within 192.168.0.0/16 private address space, a single detector can be hugely effective at providing early warning. However, these results again demonstrate that even with pre-knowledge of the vulnerable population, and the ubiquitous placement of detectors, hotspots – in this case produced by environmental factors – reduce the visibility of distributed detectors.

6. Conclusion

This paper has introduced the idea of hotspots and demonstrated how hotspots impact malware propagation on the Internet. We showed how hotspots are caused by algorithmic factors inherent to the specification and implementation of a threat and by environmental factors in the network. We empirically demonstrated how algorithmic factors cause hotspots in bots and in the Blaster and Slammer worms and how environmental factors cause hotspots in the CodeRedII worm. We then demonstrated the impact of hotspots by showing how they reduce the visibility of distributed detectors raising the possibility that new outbreaks may be missed.

Although the impact of hotspots in Internet threats can often be explained after the fact, hotspots are nearly impossible to model in general. The core problem is that hotspots are very difficult to predict. The result is that it is not possible to pre-position detection systems and many or all sensors in a distributed detection system may never alert. The main message of this study is that while global distributed detection systems have an important function, it is critical to invest in local detection systems to protect networks from the targeted impact of hotspots.

Acknowledgments

This work was supported by the Department of Homeland Security (DHS) under contract number NBCHC040146, and by corporate gifts from Intel Corporation and Cisco Corporation.

References

[1] M. Bailey, E. Cooke, F. Jahanian, J. Nazario, and D. Watson. The Internet Motion Sensor: A distributed blackhole monitoring system. In *Proceedings of Network and Distributed System Security Symposium (NDSS '05)*, San Diego, CA, February 2005.

[2] M. Bailey, E. Cooke, D. Watson, F. Jahanian, and J. Nazario. The Blaster Worm: Then and Now. *IEEE Security & Privacy*, 3(4):26–31, 2005.

[3] Z. Chen, L. Gao, and K. Kwiat. Modeling the spread of active worms. In *IEEE INFOCOMM*, 2003.

[4] Computer Associates. Win32.Agobot. `http://www3.ca.com/securityadvisor/virusinfo/virus.aspx?id=37776`, July 2004.

[5] E. Cooke, M. Bailey, Z. M. Mao, D. Watson, and F. Jahanian. Toward understanding distributed blackhole placement. In *Proceedings of the 2004 ACM Workshop on Rapid Malcode (WORM-04)*, New York, Oct 2004. ACM Press.

[6] E. Cooke, F. Jahanian, and D. McPherson. The Zombie roundup: Understanding, detecting, and disrupting botnets. In *Proceedings of the Steps to Reducing Unwanted Traffic on the Internet (SRUTI 2005 Workshop)*, Cambridge, MA, July 2005.

[7] E. Cooke, Z. M. Mao, and F. Jahanian. Worm hotspots: Explaining non-uniformity in worm targeting behavior. Technical Report CSE-TR-503-04, University of Michigan, 2004.

[8] eEye Digital Security. ANALYSIS: Microsoft SQL Server Sapphire Worm. January 2003.

[9] K. Egevang and P. Francis. RFC 1631: The IP Network Address Translator (NAT). 1994. http://www.ietf.org/rfc/rfc1631.txt.

[10] P. B. Garrett. *Making, Breaking Codes: an Introduction to Cryptology*. Prentice-Hall, Inc., Upper Saddle River, NJ 07458, USA, 2001.

[11] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishna. Fast portscan detection using sequential hypothesis testing. *Proceedings 2004 IEEE Symposium on Security and Privacy, Oakland, CA, USA, May 9–12, 2004*, 2004.

[12] H.-A. Kim and B. Karp. Autograph: Toward Automated, Distributed Worm Signature Detection. In *Proceedings of the 2004 USENIX Security Symposium*, San Diego, CA, USA, August 2004.

[13] A. Kumar, V. Paxson, and N. Weaver. Exploiting underlying structure for detailed reconstruction of an internet-scale event. *Proceedings of the USENIX/ACM Internet Measurement Conference*, Oct. 2005.

[14] McAfee. W32/Sdbot.worm. `http://vil.nai.com/vil/content/v_100454.htm`, April 2003.

[15] D. Moore. Network telescopes: Observing small or distant security events. In *11th USENIX Security Symposium, Invited talk*, San Francisco, CA, Aug. 5–9 2002. Unpublished.

[16] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the Slammer worm. *IEEE Security & Privacy*, 1(4):33–39, 2003.

[17] D. Moore, G. M. Voelker, and S. Savage. Inferring Internet denial-of-service activity. In *Proceedings of the Tenth USENIX Security Symposium*, pages 9–22, Washington, D.C., Aug. 2001.

[18] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson. Characteristics of Internet background radiation. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 27–40. ACM Press, 2004.

[19] M. A. Rajab, F. Monrose, and A. Terzis. On the effectiveness of distributed worm monitoring. In *Proceedings of the 14th USENIX Security Symposium*, Baltimore, MD, August 2005.

[20] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. RFC 1918: Address allocation for private internets, 1996. http://www.ietf.org/rfc/rfc1918.txt.

[21] Robert Graham. Decompiled Source for MS RPC DCOM Blaster Worm. `http://robertgraham.com/journal/030815-blaster.c`, 2003.

[22] C. Shannon, D. Moore, and J. Brown. Code-Red: a case study on the spread and victims of an Internet worm. In *Proceedings of the Internet Measurement Workshop (IMW)*, Dec. 2002.

[23] C. Shannon, D. Moore, G. M. Voelker, and S. Savage. Internet quarantine: Requirements for containing self-propagating code. In *INFOCOM*, Dec. 21 2003.

[24] S. Singh, C. Estan, G. Varghese, and S. Savage. Automated worm fingerprinting. In *6th Symposium on Operating Systems Design and Implementation (OSDI '04)*, pages 45–60, San Francisco, CA, Dec. 6–8 2004.

[25] D. Song, R. Malan, and R. Stone. A snapshot of global Internet worm activity. FIRST Conference on Computer Security Incident Handling and Response, June 2002.

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

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

[28] The Honeynet Project. Know your enemy: Tracking botnets. `http://www.honeynet.org/papers/bots/`, March 2005.

[29] C. C. Zou, L. Gao, W. Gong, and D. Towsley. Monitoring and early warning for Internet worms. In *Proceedings of the 10th ACM conference on Computer and communication security*, pages 190–199. ACM Press, 2003.