## **Designing a Framework for Active Worm Detection on Global Networks**

Vincent Berk vberk@ists.dartmouth.edu George Bakos gbakos@ists.dartmouth.edu

Robert Morris Robert.Morris.Sr@dartmouth.edu

Institute for Security Technology Studies Thayer School of Engineering Dartmouth College Hanover, New Hampshire

#### Abstract

Past active Internet worms have caused widespread damage. Knowing the connection characteristics of such a worm very early in its proliferation cycle might provide first responders an opportunity to intercept a global scale epidemic.

We are presenting a scalable framework for detecting, in near-realtime, active Internet worms on global networks, both public and private. By aggregating network error messages resulting from failed attempts at packet delivery, we are able to infer deviant connection behavior of hosts on interconnected networks. The Internet Control Message Protocol (ICMP) provides such error notification. Using a potentially unlimited number of collectors and analyzers, we identify 'blooms' of activity. The connection characteristics of these 'blooms' are then correlated to identify worm-like behavior, and an alert is raised.

Promising results have been produced with a simulated Internet worm, demonstrating that new worms can be detected within the first few minutes after release, depending on the level of participating router coverage.

## 1 Introduction

Active worms have caused major problems since the early days of the Internet (see [14], and [3]). As the Internet grows, there are more potential targets to infect when the next worm hits. In addition to an increasing number of active worms, the speed at which they propagate has also increased. This can again be attributed to the rapid growth

of the Internet; realizing that more systems on the Internet leads to higher connectivity resulting in greater propagation speeds. This makes early response to new events a key priority in limiting the damage a worm can do.

In our work we focus on detecting active Internet worms very early on in their life cycle through non-intrusive techniques. This has led us to investigate routing error messages, such as the ICMP destination unreachable message. The main consideration was to limit any privacy concerns that could arise from conventional Internet monitoring through traffic sniffing, while still providing the fastest detection time possible.

Because of the random scanning behavior of Internet worms, many vacant IP addresses will be probed. Routers have the option of generating error messages to notify the sender that an unused IP was addressed. Therefore, infected hosts will elicit vastly more destination unreachable messages than uninfected hosts do, making them stand out clearly. Thus only a small portion of these messages is needed to indicate malicious behavior.

We propose a system in which a portion of the Internet routers generate duplicate ICMP destination unreachable messages and forward those to a central collection point. We present how these messages can be analyzed and correlated to produce worm alerts. Promising results have been obtained in a simulated test environment and we discuss the challenges and difficulties faced by implementing this system on the Internet.

## 2 Background

In this section we introduce the ICMP destination unreachable message, and how and when it is produced. We also discuss active Internet worms in more depth, focusing on the properties that make it detectable using ICMP destination unreachable messages.

## 2.1 Internet Control Message Protocol

The purpose of the Internet Control Message Protocol (ICMP) is to provide feedback on problems in IP communication and routing. (See RFC 792 [13].) The message type that we are particularly interested in is the 'Destination Unreachable' message. In its various forms, it relays information on unreachable hosts, networks, ports, and services as well as various other codes that are of lesser importance to our work. Destination Unreachable messages are identified by the ICMP 'type' field value of 3; therefore, we will refer to these messages as ICMP-T3. Furthermore, the RFC states that routers *should be able* to generate ICMP-T3 messages, not that they *should* generate them. This means that these messages are not necessarily always returned. We will now explore the two destination unreachable messages most relevant to this project.

ICMP-T3 host unreachable messages are generated by local network routers (also called stub routers). When a packet arrives that is destined for an IP address on that local network, the router will try to contact a machine associated with that IP address. If, after repeated attempts, no host replies, that router responds to the original sender with an ICMP-T3 host unreachable. We will explore briefly how this is done.

In order for the router to communicate with a host on the local network it will have to match the IP address to an interface address on that local network medium (e.g. MAC address on Ethernet). This is done by sending an ARP (Address Resolution Protocol, see RFC 826 [11]) request out on that medium. The ARP request is received by all systems within that broadcast domain. If a system recognizes its own IP address it will respond to the router with its interface address. The router can now send the actual packet to that interface. If, after repeated attempts, no host on the local network responded to the routers requests, the router will generate the ICMP-T3 host unreachable.

ICMP-T3 network unreachables are generated by transit level routers. A router can be considered transit level when it is not responsible for directly delivering packets to a local network but rather passing on packets to other routers for further routing. Although there is often no clear distinction between stub routers and transit level routers (many routers perform both tasks), we will adopt the concept of dedicated transit level routers for now to explain how and when ICMP-T3 network unreachables are generated.

When a packet is delivered to a transit level router for further routing, that router will consult its routing tables to decide what to do next with the packet. On its way from source to destination, a packet traverses several (in some cases 10 to 20) transit level routers. If there is no corresponding entry in the routing table, the packet cannot be forwarded. The router will discard the packet and return an ICMP-T3 network unreachable. Again, the ICMP-T3 is sent to the originator of the failed packet.

What makes an ICMP-T3 message useful for analysis is that parts of the original packet will be encapsulated. According to RFC 792 [13], at least the original IP header and 8 bytes of the payload should be included. (RFC 1812 [1] dictates even more bytes should be included by routers.) This means that when a TCP/IP packet is embedded in an ICMP-T3, it should include (amongst others) the original source IP address, destination address, source port, and destination port as well as the original packet (and payload) size. This is enough to determine at least some of the packet originator's intent. Our experience is that many routers include more of the payload than the required first 8 bytes, possibly leading to even more information on the senders intentions. (See figure 2 for a graphical view of an ICMP-T3 message.)

With this in mind, the authors performed several random scans on the Internet, taking care to avoid reserved address space. We sent out connection requests (1 packet total) and awaited the response, which was either a connection response, a connection reset, a destination unreachable or no response at all (keeping in mind that routers are not *required* to respond). Table 1 is data captured during the propagation of the Nimda Internet worm. For tables 2 and 3 we selected IP addresses at random. All three tables suggest that in most cases no response will be returned at all, however, of all the responses, the ICMP-T3 was the most frequently observed. To make this clear we present table 4 showing what fraction of the responses were destination unreachables.

#### 2.2 Active Internet Worms

Active worms are different from viruses in that they are completely autonomous entities. A virus generally binds to executable code (both in system executables and scripts

SYN	18369	100 %
ACK/SYN	506	2.75 %
RST	519	2.83 %
ICMP t3	4858	26.45 %
no response	12486	67.97 %

Table 1. Target selection scan events - Nimdatarget selection

Table 2. ICMP ping requests (type 8) - randomtarget selection

ping requests	111656	100 %
ping replies	2158	1.93 %
ICMP t3	14351	12.85 %
no response	95147	85.21 %

Table 3. TCP port 80 connections - randomtarget selection

SYN	1981444	100 %
ACK/SYN	9374	0.47 %
RST	16369	0.83 %
ICMP t3	125946	6.36 %
no response	1829755	92.34 %

contained in email) and requires that code to be run to propagate. Even email worms are essentially viruses in that they piggy-back on email communications for propagation, although that statement has been debated (see [3]). Active worms, on the other hand, will autonomously select targets, attempt infection, and propagate their code to new infectors. We summarize the operational steps of an active Internet worms as follows, keeping in mind that these steps are iterated repeatedly:

- Target Selection
- Attack
- Code Propagation

Each infected host, in an active worm event, is itself an attacking system. It is solely responsible for target selection, target list maintenance (if any) and exploitation of some vulnerability in the targets' architecture. Individually, there is nothing to distinguish a worm-infected host from

# Table 4. Relative responses. What part of the responses were ICMP-T3?

Selection Algorithm	Total Responses	ICMP-T3
Nimda (table 1)	32.03 %	82.57 %
Random Ping (table 2)	14.79 %	86.92 %
Random Port 80 (table 3)	7.66 %	83.02 %

any other malicious process, thus we will have to correlate the behavior of multiple systems and identify similarities, within a given time window. The propagation efforts of a single hostile host will be referred to in this work as a "bloom" of activity.

Although there are many different target selection techniques (see [16]), some form of probing needs to be done at a certain point. This can be before the worm is launched so that it carries a target list, but in general worms use some method of scanning IP addresses across the Internet. Code Red II and Nimda (see [5] and [7]) both used target selection methods that gave targets IPs that were numerically closer to the infected host a higher chance of being selected for probing. This will invariably cause probes to IP addresses that cannot be reached, which might elicit a 'destination unreachable' response by a router. Keeping in mind that IP address selection will be more or less random, we can once again consult tables 1, 2, and 3, which were produced with random scanning, to get an idea of the reactions that a propagating worm provokes.

An important part of a virus is its hiding technique but for active worms this is not a priority since, in general, they do not piggy-back on other network protocols, meaning that worm propagation is clearly visible on the network medium. Although simple hiding techniques are often used, the main focus is on fast propagation. Another aspect of worms is their payload. If the payload is aimed at disabling the system, by wiping the disks for example, it should not be released before the worm has a chance to propagate sufficiently. If it is released too soon, the epidemic might die out due to lack of propagation. For more on computer-virus infection modeling see [8] and [2].

## 3 Implementation

We constructed an active worm to create a reliable test environment and restricted it to propagate within a finite domain. This ensures that test results can be reproduced and improved upon as well as being able to continuously test new detection algorithms. In this section, we explain how we implemented the active Internet worm detection system and how data is collected from the Internet.

#### 3.1 Simulating Worms

Being able to develop and test new algorithms and detection concepts is vital to our research; therefore, it has been important to have an isolated test environment in which results can be reproduced reliably. For this reason, we have built an active worm that is restricted in its propagation by only allowing it to produce copies of itself on the local machine.

Our target selection mechanism is based on random selection of IP addresses within a specified range. The random generator is seeded with the creation time of this instance of the worm, thus ensuring each new instance will exert unique random target selection behavior. In the near future, we expect to add several other well known target selection techniques, such as the target selection mechanism used by Code Red and NIMDA (see [16], [7], [6] and [5]).

The experiments were conducted on POSIX compliant operating systems. Worm simulations were run on a single processor i386 machine under LINUX. The analysis framework was tested on a 14 CPU Ultra SPARC III cluster, although a pair of Pentium 4 systems were also capable of handling test loads.

In our simulation, we replaced the attack step with a query to a data structure that holds information on all the vulnerable IP addresses. This list is compiled before the first instance of the worm is launched. Using such a list can be done efficiently with shared memory since all instances of the worm will remain on one physical computer. Since it is known up front which hosts are vulnerable, the need to simulate attacks is eliminated. Actual hosts with vulnerable services are not needed. When our worm targets an IP address that is both vulnerable and not yet infected, it will fork a copy (launch an independent copy) of itself using that vulnerable IP address as the new source address. This way the system will have a separate process for each instance of the worm and each process will keep track of the IP address that was compromised at its creation, thus simulating many infected systems on one physical host.

When an IP address is not reachable, the worm instance will generate a crafted network packet building the ICMP destination unreachable message that would have been generated if the attack probe had actually occurred. The creation of that packet simulates a routers response. This is done by first creating an embedded packet originating from the IP address that identifies this worm instance going to the target address that was selected (and was not reachable). Next, this packet is encapsulated in an ICMP destination unreachable message originating from the actual physical host's IP address going to our detection framework for processing. Note that no connection requests are generated for non-vulnerable machines that are actually reachable. This specific subset of IP addresses just causes a delay in the worm to simulate the successful connection attempt, but failure to infect. (See figure 2 for a graphical overview of the encapsulation of the original datagram in an ICMP-T3 response.)

To simulate network latency, we force each instance of the worm to sleep briefly (1 to 10 milliseconds) after sending out a packet. This way we can simulate up to one thousand vulnerable hosts in a search space of our choice. The size of the search space determines the fraction of vulnerable hosts, which directly influences propagation speed.

#### 3.2 Routers and Coverage

As mentioned before, arbitrary unsolicited traffic often elicits ICMP-T3 messages from routers across the Internet. Our active worm detection methods rely on receiving a 'blind carbon copy' of as many of these messages as possible. This means that the router has to forward a copy of such a message to a central collection point. To facilitate data collection, we distribute a slightly modified version of the popular LINUX kernel 1 that will upgrade the router to create a duplicate of every ICMP-T3 message that this router generates. That duplicate is forwarded to a central collection location for processing. Optionally, if site policy dictates, the original can be discarded while still sending a copy to the central collector. We will call these routers 'ICMP-BCC routers'. (See figures 1, 2 and 3 for, respectively the original connection attempt, the creation of the ICMP-T3 message, and that message being returned to both original sender as well as the analysis framework. The router in these pictures depicts an ICMP-BCC router.)

Originally, an ICMP-T3 message should contain the original IP header and at least 8 bytes of next layer protocol (see RFC 792 [13]). As per RFC 1812 [1] the ICMP-T3 message should contain as much of the original message as possible, with a maximum of 576 bytes for the entire packet. From this embedded information, we can retrieve

<sup>&</sup>lt;sup>1</sup>The 66 line kernel DIFF is available from our website at: http://www.ists.dartmouth.edu/IRIA/projects/dibs/

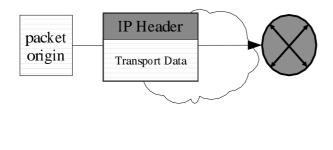
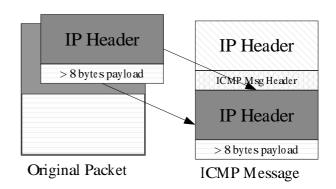


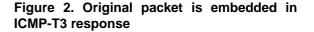
Figure 1. Original connection attempt

the original packet's sending IP address and can also infer much of the intent of the packet, i.e. port 80 connection request, DNS traffic, etc. Pseudo-random target selection results in a distribution of attack packets across a large portion of the Internet. To identify an infected host, it is important to gather ICMP-T3 messages resulting from its behavior from as many different routers as possible. As ICMP-T3 messages are generally produced by routers very close to either the source (network unreachables) or destination (host unreachables) of the packet, we need to instrument a significant portion of the Internet address space with ICMP-BCC enabled routers.

Any given host initiating arbitrarily targeted unsolicited traffic is likely to provoke ICMP-T3 responses from routers throughout the world. Since we aim to receive legitimate copies of ICMP-T3 messages, without sniffing for them, they will have to be forwarded by the router that generates them. Therefore, because ICMP-T3 messages are generated close to the targeted network, many routers need to be converted all over the Internet. Further research is needed to identify the appropriate scope of coverage.

Our introduction to ICMP-T3 messages included several tables (table 1, 2 and 3) that suggest that the actual number of ICMP-T3 messages is going to be relatively low. It should be noted that the lack of response to the majority of the connection attempts may indicate that routers have been configured to silently drop incoming packets, without returning any error message. An ICMP-BCC enabled router could similarly be configured to silently drop failed packets while still providing a properly constructed ICMP-T3 to a central collection point. Optimally, with 100% participation this would provide a nearly complete





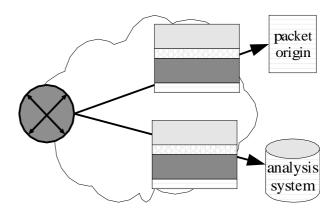


Figure 3. ICMP-T3 response is returned to sender and analysis system

view of all failed communication attempts. (See table 5.)

### Table 5. Relative responses to the central collector from a ICMP-BCC router

Selection Algorithm	No Responses + ICMP-T3
Nimda (table 1)	94.42 %
Random Ping (table 2)	98.06 %
Random Port 80 (table 3)	98.7 %

## 3.3 Detection Framework

The stream of data grows as more and more routers across the Internet forward copies of their ICMP-T3 messages to a central location. Since analysis is done on the fly, the system has to scale with the increase in ICMP-T3 messages. Here we explain how our collection, analysis, and detection systems are implemented and how we deal with the large volumes of data that flow into the system during the propagation of an active Internet worm.

#### The Collector

ICMP-BCC routers send the copies of their ICMP-T3 packets to a single process collector that buffers and stores them in a rotating packet store. This packet store is continuously rotated and reflects the messages of the last t seconds where t is based on available disk-space and bytes per second input flow:

## $t = \frac{BytesPerSecond}{DiskSpace}$

The collector then makes two copies of the packet and sends them off to two analyzers. The collector divides the entire IPv4 address space by the number of analyzers available. This means that each analyzer covers a fixed range of IP addresses. One of the copies of the packet is sent to the analyzer handling the IP range that the embedded source address falls in. Likewise, the other copy is sent to the analyzer that takes care of the IP address range that the embedded destination address falls in. This way the analysis load is divided over the available analyzers. The number of analyzers can be changed dynamically to suit the current load of the system and can be any number. This is what makes the framework scalable. Theoretically, an analyzer could be spawned for each new IP address (source and/or destination) that is encountered, up to  $2^{32}$  analyzers. (IPv4 uses 32 bit IP addresses.)

#### The Analyzers

In the analyzer, one copy of the message will be analyzed based on source IP-address and the other copy will be analyzed based on destination IP-address. The packet will be stored for a pre-determined amount of time  $\Delta t$ . If for a certain IP-address the number of stored ICMP-T3 messages exceeds a given threshold (say 30) further investigation is initiated.

Further investigation begins by counting the number of recurring source and destination port numbers. Any of several tracks can be chosen if the port count for any port p exceeds a given threshold N for any protocol P, with P being either TCP, UDP or ICMP: (port can be replaced by type/code-pair for ICMP).

- 1. One IP address has contacted at least N different other IP addresses on exactly the same port p using the same protocol P in the last  $\Delta t$  seconds.
- 2. One IP address was contacted by at least N different other IP addresses on the same port p using the same protocol P in the last  $\Delta t$  seconds.
- 3. One IP address has contacted one other IP address at least N times on the same port p using the same protocol P in the last  $\Delta t$  seconds.
- 4. One IP address was contacted by one other IP address at least N times on the same port p using the same protocol P in the last  $\Delta t$  seconds.

Two other cases can be detected:

- 5. One IP address has contacted one other IP address on at least N different ports in the last  $\Delta t$  seconds.
- 6. One IP address was contacted by one other IP address on at least N different ports in the last  $\Delta t$  seconds.

Any protocol other than TCP, UDP, or ICMP will generate an Alert whenever the total number of different IP destination addresses for the failed connection attempts exceeded threshold N.

In any of the first four cases, an Alert is sent to the correlation engine. The correlation engine collects alerts from all the analyzers and tries to identify similarities between them. Case 1 is the most relevant case since it could be a direct indication of actual active worm propagation. We will explore case 1 shortly. Case 2 can also be a sign of active worm propagation but is not expected to be seen until much later in the life cycle of the worm, although that is highly unlikely. This is because the chances are very small that in a given  $\Delta t$  there are enough (N) infected hosts that select the same target for attack. More likely, case 2 is the sign of a server being disconnected from the network. Imagine a web-server with a severed connection, the number of failed connections will initially be large but will decline over time as current sessions time out and people start to realize that the service is no longer available. It could also indicate the activation of a payload in a D-DOS attack, after the target was successfully disabled or simply taken off the wire, although chances are that the worm had already been detected through a flood of Case 1 Alerts.

Case 3 and 4 most likely indicate a disconnected machine that is part of an autonomous communication. The ICMP-T3 messages are a sign of the service trying to reconnect. Case 5 and 6 are probably the result of a vertical port-scan of a single host, thus attempting to connect to various ports on one machine. (These messages will be host unreachables if the actual system is unreachable, but could also be port, or service unreachable messages, depending on the filtering rules of the router.)

Case 1 is most likely a sign of active probing by one host of many different hosts on the same service port. This could be a horizontal port-scan looking for a single vulnerability. This is what was previously defined as a 'bloom'. If multiple hosts start showing a bloom of similar activity (scanning for the same service) then we might be observing an Internet worm propagating. If the number of detected blooms (with similar parameters) increases exponentially, we may assume we have detected a new active Internet worm and captured its basic characteristics.

#### The Correlator

The analyzers produce Alerts based on IP addresses and one of the six possible behavioral patterns, observed over a small time span. None of these Alerts by themselves directly indicate a propagating Internet worm. For further analysis the behavior of many systems needs to be compared and evaluated. Therefore, all the Alerts produced by the analyzers are forwarded to the Correlator.

The Correlator compares all the Alerts received in the previous  $\Delta t$  time span and identifies all the similarities between those Alerts. If a certain number (currently 4) of IP addresses exhibit identical behavior a Worm Warning is issued. This final stage reports to the user and gives a list of the IP addresses, their scanning behavior, protocol and port numbers, and time stamps. Each new instance (IP address) with similar behavior will now also be reported directly and connected to the matching Worm Warning. When a Worm

Warning is issued, the user can decide what action to take.

The number of similar Alerts that will trigger a Worm Warning within the  $\Delta t$  time span is still unclear. Good results were obtained with values between 4 and 6, keeping in mind detection time will increase, and false-positive rate will decrease as this number increases. However, we suspect that as the participating router coverage on the Internet grows, we will be forced to increase the value to suppress false positives.

## 4 Results

In this section we look at the test results that we have collected so far with the system described in the previous section. We use real-world data to calibrate the many parameters of the detection system, as well as to improve on our worm model. The most important calibration is that of the baseline. Many ICMP-T3 messages get generated in normal Internet traffic, but rarely as many as an actively propagating worm does. Even if normal traffic generates an alert or even a bloom, the relevance of that event will be small since it is never matched by other hosts showing similar behavior.

#### 4.1 Worm Simulation

In our worm model we do not account for a dynamically changing Internet. That includes ignoring human patch rate. The main reason for doing this is that only the initial propagation of the worm is of relevance to us. The primary goal is to detect active Internet worms in their earliest stages of propagation. The most tell-tale signature is the exponential increase of blooms with similar characteristics.

The primary parameters are the number of vulnerable hosts and the number of reachable hosts. Figure 4 depicts our worm model propagating over time. Time is in seconds and each point in the graph indicates a newly infected host. This is done in an environment with 800 vulnerable hosts out of a total addressable space of 1 million IP addresses. Note that reachable machines (regardless of vulnerability) will not provoke ICMP-T3 responses.

Figure 4 depicts the simulated worm propagating over the time-span of about 450 seconds that it took to infect all 800 vulnerable hosts. The dotted line next to it is the Sigmoid Curve given by the formula:

$$y = \frac{1}{1 + e^{-g(x)}}$$

Where g(x) represents a linear function to scale and translate the curve to match the test data. Finding the proper parameters for g(x) is the task of the Correlator. Initial comparison with exponential curves was less successful since it cannot give a prediction of the epidemic. (Exponential curves grow infinitely and never slope down.) Although the Kermack and McKendrick model (see [2]) gives a far more accurate representation of the life cycle of an epidemic, it proved significantly harder to implement in the Correlator. Furthermore, it increased processing load without a significant increase in accuracy over the Sigmoid approximation. The goal is to provide an estimation of the course of the epidemic very early on, updating the estimate as more Alerts flow into the Correlator. Most notably, expected duration and total infected systems is estimated.

Although there is a striking similarity between both graphs, it needs to be remarked that the simulation curve is slightly steeper and overshoots the Sigmoid Curve at the end. Looking at the data for Code Red (fig: 5, taken from: http://www.caida.org/analysis/security/code-red/gifs/compare-cumulative-ts.gif) it is clear that the curve flattens out in the end. This has been attributed to infected systems being taken off-line as well as infected systems being cleaned and patched (see [4]). Since our simulation does not account for these factors, our curve continues and flattens out only at the very end where nearly all vulnerable systems have been infected. For our work, only the initial, near-exponential growth is relevant for detection.

Since each worm instance crafts its own ICMP-T3 response packets, our simulation assumes a 100% ICMP-BCC router coverage in their address space. Every connection attempt to an unreachable host will result in an ICMP-T3, as if it was generated by an ICMP-BCC router. Determining the effects of router coverage on detection speed and accuracy of detection will be the focus of future research.

#### 4.2 Detection Results From Worm Simulation

With the given 100% ICMP-BCC router coverage, the collection engine gets an ICMP-T3 packet for every infection attempt that any of the worm instances make. See table 6 for the current set of system parameters. The worm uses the same protocol and destination port for all its attempts within a propagation run. This makes destination port p and protocol P invariant over the duration of a single experiment and are thus left out of the graphs. Figure 6 shows the results of an experiment with 100 vulnerable hosts.

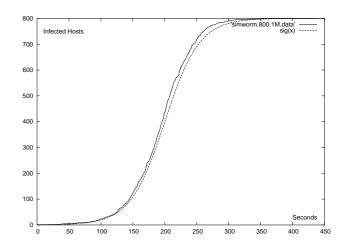


Figure 4. Solid curve is the simulated worm. Dotted line is the Sigmoid Curve.

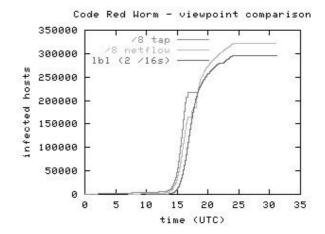


Figure 5. Code red propagation, infected hosts, from [4]

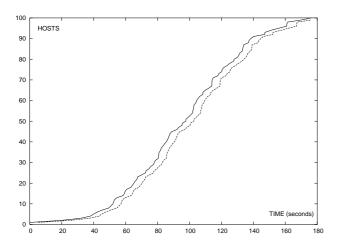


Figure 6. Propagating Worm. Dotted line is detection curve.

Table 6. Parameters for detection framework for the presented results The N threshold indicates number of packets with matching characteristics.

Parameter	Value
$\Delta t$ (received packet timeout)	600 seconds
N (threshold)	25
P (protocol)	TCP
p (destination port)	80

The first curve (figure 6) represents accumulatively each new instance of the worm. Every point indicates a newly infected machine as reported by the worm processes themselves. The curve closely following the first is the accumulated result from the detection system. When a host has been classified as 'infected', it will remain infected to prevent reporting of one infected host twice. However, if that host starts to exert scan behavior with different characteristics than for its last detection, it will be reported again for this separate incident. This ensures that if a single system is hit by two different active worms, the detection of the first does not obscure the detection of the second infection.

The time between infection and detection with a 100% ICMP-BCC router coverage is about 5 seconds. At least 4 individual hosts showing blooms with similar characteristics will need to be identified before a sequence of events is

classified as a potential active worm. This point occurs 43 seconds after the initial host has been infected. Note that it took 19.7 seconds before a second, vulnerable host was found and infected by the first.

As the router coverage decreases, the number of ICMP-T3 messages that the collector receives will decrease with it in a linear fashion. Since blooms are not identified until threshold N is exceeded, it will take proportionally longer for the alert to be raised. One way to offset this is by lowering threshold N, while increasing time window  $\Delta t$ (to avoid purging messages prematurely). Invariably this will produce more false-positives, as well as increasing the load on the analysis system. Future experiments need to be done to determine the exact relationship between these factors.

To simulate regular network background ICMP-T3 traffic, the experiments were done while injecting a steady stream of recorded background noise. This noise consists of random ICMP-T3 messages directed at our collector. The injection speed is 2M-bit per second and does not significantly change the average latency between infection and detection. However, it does increase processing load and memory usage of the detection system, specifically that of the analyzers. Even if a random system would produce bloom-like behavior, connecting to the same destination port on many different hosts, it is unlikely that a Worm Warning would be raised. The Correlator will not start tracking blooms until at least four hosts have exerted exactly the same behavior.

## 5 Conclusion

Although the current results look promising, there are still many ways in which we intend to extend and improve the system. Until we have sufficient participating routers across the Internet, we are forced to conduct our tests in an experimental environment. Finally, in addition to early detection, rapid response should not be ignored. Novel ways of fast response, with or without the intervention of the administrator, still need to be researched in-depth.

Furthermore, the framework that we propose has many parameters and characteristics. As of now we do not have any significant information on the values and behavior of these parameters. Most notably, we have very little quantitative information on the critical number of participating routers to adequately detect active worms on the Internet, in a reasonable amount of time to allow any response to make a difference. The behavior of the false-positive rate compared to router coverage is also largely unknown. Each of the parameters governing the detection in itself warrants further investigation.

## 5.1 Research Direction

As participating router coverage of the Internet is relatively low, we are looking for way to improve the detection algorithm. Much of the background noise might actually be valuable data. Tests were performed with very high thresholds to filter most of the background noise. In the real-world version of the system the thresholds are significantly lower to improve detection speed at the cost of an increased false-positive rate. This leads us to believe that the critical ICMP-BCC router coverage on the Internet should be relatively high.

This relationship between coverage, thresholds, and infection-detection latency is as yet not well known. In addition to researching this relationship, we are experimenting with various other detection and classification algorithms to improve accuracy and detection speed while reducing the need of wide-spread ICMP-BCC router coverage across the Internet. In order to do this it is important to identify networks that have a high likelihood of being targeted by active worms. ICMP-BCC routers will be most effective at these networks.

Essential to continued viability of this system under heavy load is the ability to dynamically pre-filter messages corresponding to a known worm event. This will free up critical resources to enable continued monitoring and alerting of unrelated, yet concurrent events. This dynamic filtering mechanism is not yet developed.

In order to direct the resources of first responders appropriately it is important to give an early warning of a possible Internet worm. However, to avoid tying those valuable resources up unnecessarily, we find it important to give a prediction of the epidemic, very early on and adjust the prediction as more and more messages come in. This includes estimated machines that will be infected and the duration of the epidemic. Current estimation is done by fitting a Sigmoid curve. In the future we aim to justify that this is usable as an approximation over the far more complex Kermack and McKendrick model, for our purposes.

#### 5.2 Summary

We have presented a scalable method of detecting active Internet worms in near-realtime on a global scale.

Our framework collects ICMP destination unreachable messages from across the Internet and sorts them by source and destination IP address. These connection statistics are monitored over time and alerts will be raised when certain thresholds are exceeded. An active worm can be identified if multiple machines exert similar bloom-like connection behavior with the same protocol characteristics.

For the collection of the ICMP destination unreachable messages, we rely on Internet routers to forward copies of those messages that they generate to a central collector. From there, they are distributed to an array of analyzers that all report back to a Correlator system. The analyzers generate reports of significant behavior and create a set of identifying characteristics. Based on those characteristics the Correlator determines whether an active worm is propagating by comparing reports received from other analyzers.

Further studies are needed to show what the effects of coverage are on the latency between initial infection and detection. Also, increased background noise has a significant impact on performance, which can only be mended by dedicating more analyzers to the system. We hope to address both these issues by developing better classification, detection techniques and dynamic pre-filtering.

## References

- [1] F. Baker. Rfc 1812: Requirements for ip version 4 routers. volume 1812 of *Request for Comments*. 1995.
- [2] D. Daley and J. Gani. *Epidemic Modelling*. Cambridge University Press, 1999.
- [3] M. W. Eichin and J. A. Rochlis. With microscope and tweezers: An analysis of the internet virus of novermber 1988. In *Proceedings of the 1989 IEEE Computer Society Symposium on Security and Privacy*, 1988. eichin89with.ps: morris worm.
- [4] http://www.caida.org/analysis/security/code red/coderedv2\_analysis.xml. The spread of the codered worm (crv2).
- [5] http://www.caida.org/analysis/security/code red/index.xml. Caida analysis of code-red. 2002.
- [6] http://www.incidents.org/react/code\_red.html. Code red. 2001.
- [7] http://www.incidents.org/react/nimda.pdf. Nimda worm/virus report.
- [8] J. O. Kephart and S. R. White. Directed-graph epidemiological models of computer viruses. In *IEEE Computer Society Symposium on Research in Security and Privacy*, 1991.
- [9] G. Mansfield, K. Ohta, Y. Takei, N. Kato, and Y. Nemoto. Towards trapping wily intruders in the large. 1999.
- [10] V. Paxson. End-to-end routing behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(5):601–615, 1997.

- [11] D. C. Plummer. Rfc 826: An ethernet address resolution protocol. volume 826 of *Request for Comments*. 1982.
- [12] P. Porras and A. Valdes. Live traffic analysis of tcp/ip gateways. In *ISOC Symposium on Network and Distributed Systems Security*, 1998.
- [13] J. Postel. Rfc 792: Internet control message protocol. volume 792 of *Request for Comments*. 1981.
- [14] E. Spafford. The internet worm incident. In European Software Engineering Conference, volume 87 of Lecture Notes in Computer Science, pages 446–468. Springer-Verlag, 1989.
- [15] S. Staniford. Analysis of spread of july infestation of the code red worm. Technical report, Silicon Defense, 2001.
- [16] S. Staniford, V. Paxson, and N. Weaver. How to 0wn the internet in your spare time. *Usenix*, 2002.