

---

## Origins: an approach to trace fast spreading worms to their roots

---

### Andrew L. Burt

Techsoft,  
P.O. Box 16143,  
Golden, CO 80402, USA  
E-mail: aburt@tech-soft.com

### Michael Darschewski

University of Denver,  
Denver, CO 80210, USA  
E-mail: mdarsche@du.edu

### Indrajit Ray\*

Colorado State University,  
Fort Collins, CO 80523, USA  
E-mail: indrajit@cs.colostate.edu  
\*Corresponding author

### Ramakrishna Thurimella

University of Denver,  
Denver, CO 80210, USA  
E-mail: ramki@cs.du.edu

### Hailin Wu

Array Networks,  
Milpitas, CA 95035, USA  
E-mail: drhailin@gmail.com

**Abstract:** An automatic distributed mechanism is proposed to identify the propagation roots of fast spreading internet worms. The information obtained can be used to identify local worm outbreaks, identify network intrusion, identify internal network misuse, and help with the forensic trace-back after detection. It has been designed with simplicity, efficacy, and ease of deployment in mind. Two modes of operation are possible, yielding both real-time and post mortem propagation information. The proposed paradigm can work in unison with any intrusion detection, throttling and human-mediated responses. Simulation results show that even with only 20–30% deployment, worm origins can be pinpointed with great precision.

**Keywords:** computer worm attacks; attack trace-back; network security.

**Reference** to this paper should be made as follows: Burt, A., Darschewski, M., Ray, I., Thurimella, R. and Wu, H. (2008) 'Origins: an approach to trace fast spreading worms to their roots', *Int. J. Security and Networks*, Vol. 3, No. 1, pp.36–46.

**Biographical notes:** Andrew L. Burt, CEO, Techsoft, received his PhD in Mathematics and Computer Science from the University of Denver. He has an extensive background in the fields of security, networking, distributed processing and AI. He has taught courses in these subjects for over 20 years at the University of Denver and the Colorado School of Mines.

Michael Darschewski is currently working towards his PhD in Computer Science at the University of Denver. He was previously employed as Senior Network Engineer for GMACCH Capital Corp. His primary interests include network security, network infrastructure, routing, and peer-to-peer systems.

Indrajit Ray received his PhD in Information Technology from George Mason University. He is currently Assistant Professor of Computer Science at the Colorado State University. His main research interests are in the areas of security models, database and network security and computer forensics. He is the current chair of IFIP WG 11.9 on Digital Forensics and serves on the editorial board of Digital Investigations.

Ramakrishna Thurimella received his PhD in Computer Science from the University of Texas at Austin. He is currently Associate Professor of Computer Science at the University of Denver. His research interests include algorithms, application of randomisation techniques to computer security, and peer-to-peer systems. He is a member of ACM.

Hailin Wu received his PhD in Computer Science from the University of Denver. He is now a Senior Network Engineer at Array Networks, where he built the world's fastest and most scalable SSL VPN engine that can handle 64,000 concurrent users per system. His main interests are in networking and security.

---

## 1 Introduction

The information infrastructure has become indispensable to our economy, society, and security. The recent attacks on the internet by self-propagating code demonstrate the extent to which our information systems are vulnerable. The worms and viruses that we have witnessed so far could have been more invasive and destructive by using information destroying tactics such as erasing hard disks or modifying existing data without user knowledge. This has been by design – a deliberate choice made by worm authors. Future attacks may not be so kind. Doing nothing about securing the internet, in the hope that the malicious code in the future is also going to behave the same way, is certainly not our best defence (Reiss, 2003). Recognising the gravity of the situation, corporations such as Microsoft have announced huge sums of money to catch and prosecute virus writers (La Monica, 2003).

We propose here an automatic distributed monitoring system to trace-back to their origins, rapidly spreading worms, for the purposes of prosecution, deterrence and damage mitigation.

The existing practices to fend off attacks by malicious code are based on filtering using signatures – signatures derived from the worm code. As has been pointed out by other researchers (Staniford et al., 2002), worms can be designed to spread so rapidly that by the time a signature is developed and distributed to stop the spread of the worm, the damage is done, thus rendering any signature-based mediation futile. In fact, worms can be crafted to take over millions of hosts on the internet in a matter of minutes, making any human-mediated response impossible. In light of this frightening prospect, it is imperative to design automatic mechanisms to detect and counter such rapidly spreading contagions and provide authorities with information about the origin.

While there is extensive research on detecting intrusions into a local-area network, there is relatively little published work on automatically sensing worm outbreaks in a wide-area network. This paucity could be due to the technical difficulty of the problem. GrIDS (Cheung et al., 1999) and connection-history based intrusion detection (Toth and Kruegel, 2002) are two works in this sparsely

populated space. In addition, various strategies to be explored to combat this formidable problem are sketched in Staniford et al. (2002). We would like to mention that there is also a white paper (Arkin, 2002) that presents methods to analyse traffic logs from the internet in search of identifying the origins of a worm.

While this paper has been under review, Xie et al. (2005) presented random moonwalk algorithm to identify the entry point of epidemic spreading attacks. Moonwalk here refers to walking back in time along path of flows to trace the origin. Other recent works include Jung et al. (2004) and Weaver et al. (2004). Jung et al. show how to detect malicious port scanners using an online detection algorithm they call Threshold Random Walk. Weaver et al. in turn use this technique to develop an algorithm that contains fast spreading worms by detecting scans rapidly and suppressing them.

Our system is another tool in the white hat's arsenal. It works by correlating anomalous events across the internet and establishing a causal relationship between them. This system has been designed with simplicity, easy deployment, and efficiency in mind. We implement this system, conducting simulations with both a live network and NS-2, and argue that even with less than perfect adoption by network security administrators, it can very accurately narrow the worm origin to a small set of possibilities.

Our system is based on the observation that in order for any worm to spread rapidly, a necessary condition for flash worms, the moment it compromises a host, it must immediately identify a new set of victims. This is akin to the distributed spanning tree computation that uses flooding to extend its frontier. Thus, such a flash worm infection at a site A will almost surely be manifested by an explosive rate of outbound connection attempts from the site. Our monitors, deployed at gateways, work concurrently in reverse and identify an upstream neighbour with the eventual goal of identifying the root. Each monitor maintains sliding windows of incoming TCP and UDP connections. When an anomaly with respect to regular traffic patterns, such as an explosive rate of outbound connection attempts from one of the nodes at site A,

is detected by the Intrusion Detection System (IDS) at the same site, the monitor contacts its counterpart at hosts that have recently initiated a conversation, for example, site B. If site A can establish that an anomalous event at site B temporally preceded its own, site A concludes that it cannot be the root. By elimination, the hosts that have no anomalous upstream neighbour are candidates for worm originators. It is assumed that the IDS is capable of detecting anomalous traffic patterns such as this but not necessarily a worm or an infected host.

Several observations about our method are noteworthy. First, a stealth worm with a hit list can 'go under the radar' and might not register any anomalies with the IDS. Our method is ineffective against such attacks in real time; however an extension of our method involves logging and offline analysis to trace stealth attacks. Second, if a worm author deploys the worm and configures it to go off at a time in the future several months away (long enough that the log data is difficult to obtain), then identifying the origins of a worm is difficult and possibly of little value. Third, our algorithm assumes that the source of a virulent packet is not spoofed. If the source address is spoofed, the results are not 100% reliable. In such cases, the algorithm can give only an approximate origin of the worm. Note that TCP worms, because of the three-way handshake mechanism of TCP, cannot use spoofed addresses. In addition, almost none of the significant UDP worms discovered to date use spoofed source addresses, as they risk getting filtered by egress filters. Fourth, when we consider the case of less than 100% monitor installation, we assume monitors are distributed uniformly across the network. Studying other distributions is interesting, but beyond the scope of this paper.

In spite of these limitations, we believe the method proposed here is novel and valuable. As the first known research attempt to automatically trace-back the origin of a worm, it demonstrates that this problem is tractable in a practical manner. For forensic analysis of worm outbreaks time is of critical information and our technique is quite useful for this purpose. If it deters worm authors from writing easily detectable rapidly-spreading worms in favour of more challenging architectures (e.g., forcing worm authors to pre-identify vulnerable systems), it is likely this deterrence will result in fewer overall outbreaks. Moreover, in cases of worm outbreaks with spoofed addresses, knowing even the approximate origin of the worm is of considerable value for forensic analysis as it can provide a stepping stone for further investigation. It would probably not serve the best of interest of the prosecutor to rely solely on our technique to bring the worm writer to justice. It still needs to be backed up by other pieces of evidence. Our approach is deliberately kept simple in the hope that effective distributed systems built around the ideas sketched here may be incorporated into other network intrusion detection and mediation approaches. From this standpoint, our contribution can be seen as a way to spur further research and lure other researchers to focus on this growing epidemic.

There are two clear advantages to our method. From the ease of deployment standpoint, our method requires cooperation from only network security personnel, not individuals. In contrast, the throttling and connection-history based methods (Toth and Kruegel, 2002; Williamson, 2002) require cooperation from every node on the internet. Secondly, if a Centre for Disease Control for electronic viruses ever becomes a reality as envisioned by Staniford et al. (2002), the automated method presented here can work synergistically with human-mediated response by identifying the anomalous state of the internet in real time.

In the rest of the paper, we describe related work in Section 2, trace-back in Section 3, reporting in Section 4, simulation and performance in Section 5, discussion and future work in Section 6, and conclusion in Section 7.

## 2 Related work

There is vast amount of literature on worms and viruses, including defences against them. Without attempting to give a comprehensive summary on the topic, we will discuss briefly the papers that are directly related to our work and delineate our contribution over what is known.

The recent paper by Staniford et al. (2002), articulated clearly the havoc a well-crafted worm can wreak on the internet. In addition to presenting a mathematical model and analysing the magnitude of the threat posed, this paper presented various ways to defend the internet including a cyber version of the Centre for Disease Control. This paper suggested future research to focus on automated mechanisms for detecting worm outbreaks based on their traffic patterns and fostering the deployment of a widespread set of sensors. Our work can be seen as a first concrete implementation of a prototype along these lines.

Soon after the publication by Staniford et al. (2002) and Williamson (2002, 2003) presented a way to cope with fast spreading worms and viruses by throttling: an automatic method that restricts high-speed propagation. This paper relies on the fact that an infected machine will connect to many different machines as fast as possible. His solution limits the rate of connections to 'new' machines. It is worth noting that in order for this method to be effective, a large majority of hosts on the internet must have this defence installed on them, which may not be practical from the deployment point of view. However, if deployed at every host, throttling can be effective against worms as well as e-mail viruses.

Moore et al. (2003a) discussed ways of containing self-propagating code. They proposed reaction time, containment strategy, and blocking location as key aspects of any containment system directed toward worms. In order to be effective they also advocated automated methods to detect and react to worm epidemics. Although they proposed no specific systems to detect and react to an outbreak, automated methods of detection and reaction were inferred. They concluded that it would be very challenging to build internet containment systems capable of preventing widespread worm infections.

To the best of our knowledge, Cheung et al. (1999) were the first to use traffic analysis to detect worm outbreaks in wide-area networks. They proposed a network-wide IDS, GrIDS, which is based on matching activity graphs against known spread patterns. Toth and Kruegel (2002) discuss various shortcomings of GrIDS and present ways to address them based on anomalies in connection history. They claim automatic worm propagation determination, worm detection with very low false positive rate, effective countermeasures, including automatic responses in real time, and easy configurability. Neither of these papers attempts to trace the origins of a worm.

Arkin (2002) discussed ways to trace and profile malicious computer attackers. He described a method to locate exploits and bind them to originators using monitoring, intelligence gathering and electronic surveillance systems. In addition to looking for suspicious activity in traffic patterns, he suggested ways to reduce logging without losing any valuable information. His methods are performed manually and can complement our strategies.

There has been limited research work involved with identifying the origins of a worm (Dean et al., 2001). Current investigative technique typically involves looking at system logs, network usage patterns, and the code base after a worm outbreak is identified (Nazario, 2004). This rendezvous technique is labour intensive and is seldom successful in establishing the identity of the author of the malicious software or the source of infection.

Two other prominent methods to help identify and analyse internet worms are honeypots and dark network monitoring systems. Both listen for worm behaviour and log suspicious events to provide clues for future analysis (Honeynet Project Homepage, 2002; Moore et al., 2001). A honeypot is a functional system that fakes responses to malicious probes while dark network monitoring systems watch unused network segments for malicious traffic providing an overall attack picture. Because only a small number of honeypots are typically deployed on a network segment, they offer only a limited perspective on the wide-area network. The utility of dark network space monitoring is greatly diminished when a worm chooses to circumvent the unallocated network addresses, such as in the case of Slapper and SQL Snake worms. Our approach is different in that it is deployed in real network systems and can react to zero-day worms in real time.

An additional trace-back research study focuses on identifying the source of a particular IP packet (Snoeren et al., 2002; internet Engineering Task Force Homepage, 2003) or determining the route of a packet flow (Burch and Cheswick, 2000). These techniques help to reliably identify the originator of IP packets and are useful in thwarting attacks employing forged IP source addresses. With our goal of identifying worm origins, this work could interoperate with ours to identify the source of UDP-based attacks.

While this paper is under review, Xie et al. (2005) presented random moonwalk algorithm to identify the entry point of epidemic spreading attacks. Moonwalk here refers to walking back in time along path of flows to trace the origin. Other recent works include Jung et al. (2004) and Weaver et al. (2004). Jung et al. show how to detect malicious port scanners using an online detection algorithm they call Threshold Random Walk. Weaver et al. in turn use this technique to develop an algorithm that contains fast spreading worms by detecting scans rapidly and suppressing them.

### 3 Trace-back

The trace-back operation is an attempt to narrow the origin of a rapidly spreading worm to one or a small set of possible hosts. This is accomplished by recording and carefully correlating various events among Origins monitor agents.

#### 3.1 Outbreak detection

Many local-area networks currently employ an IDS to detect network infiltrations and worm/virus outbreaks. Our approach utilises the expertise of these systems in detecting anomalous traffic patterns indicating outbreak. For example, Snort can be configured to respond to anomalous traffic patterns. Configuration of the IDS would be simply entering the correct rules governing the patterns inherent to rapidly spreading worms. This configuration can include the protocols, the ports used, the success and failures of connections, the peers of the communications, the volume and pattern of traffic over time, and the number of scans occurring. In addition, all of these characteristics can be combined to develop a picture of the network under normal circumstances and also used to identify anomalous network activity possibly indicating the presence of a worm. It is important to note that the detection of the outbreak is dependent upon the resiliency of the IDS.

Network signature analysis, which focuses on the data communication patterns within network systems, is another, post-mortem, method for detecting worms. Worms typically have distinctive signatures as they attack other hosts on the network. By building up a library of known malicious signatures, an IDS can detect the presence of a worm quickly. Furthermore, distributed intrusion detection techniques can be utilised effectively to improve accuracy.

Additionally, systems which have not actually been compromised, but have detected an attempt made on them (via an IDS), could report this attempt. The monitor is not concerned whether a system was in fact compromised – just that an attempt was made and it may thus use that information to trace back to the node making the attempt.

Timely and accurate worm outbreak detection is a pre-condition that our algorithm relies on. For worms that are stealthy, not using built-in hit lists of known

vulnerable systems for instance, yet spread rapidly, an observable behaviour will need to be exhibited. Our goal is to track the origins of fast spreading worms, most of which so far operate in an observable fashion. At the very least, if we force worm writers to abandon the typical attack approaches for the more difficult approaches, such as reconnaissance-generated hit lists, we will still have achieved a measure of success.

### 3.2 Algorithm

Each monitor agent maintains incoming connection header information within a sliding observation window for every host in the LAN where the monitor agent is installed. The purpose of the sliding observation window is to allow the monitor agent to selectively record sufficient connection information for the trace-back operation. Therefore, the window size depends on the speed characteristics of the infection: a few seconds may be enough to do real-time trace-backs for fast-spreading worms; a few minutes may be required for worms which introduce random delays; and several months of data may be needed to do post-mortems on stealth worms. Testing our algorithm with dynamically configurable window sizes is an interesting research topic that we leave open for future research.

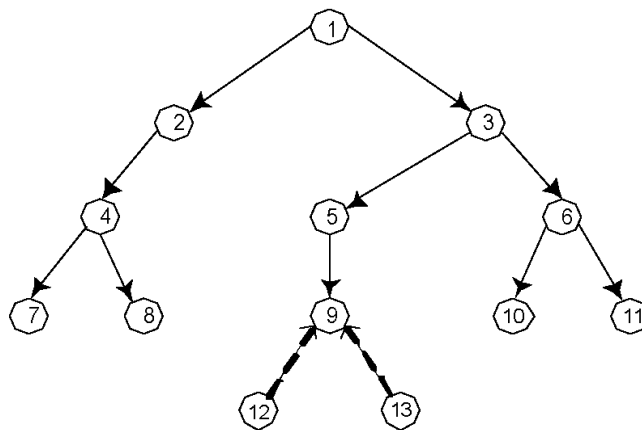
When the IDS detects an anomalous traffic pattern (supposedly signalling a worm outbreak), it directs the monitor to enter the trace-back state, and initiates the monitor's trace-back function. In the query phase, the monitor agent traverses its incoming connection list in real time, extracts the distinct set of source hosts, and sends a query message to each of them. The query message contains a specifically crafted TCP packet containing the monitor's IP address.

On receiving the query message, the monitor agent at the source host establishes a TCP session with the querying monitor. The source host replies with the time of infection, if it notices an outbreak at its site. If there is no local worm outbreak, the source monitor agent replies no, because, except for IDS failure, it is impossible for it to have infected the querying host.

The querying monitor stores the header information of the incoming data from all sources, correlating events to determine the source network. If the local network's infection time is temporally first, the monitor considers itself as a possible worm source. If a remote network's infection time is temporally first, the remote network is considered the predecessor and the local network cannot be the source of the worm outbreak. It is possible that the source of worm outbreak cannot be narrowed down to a single host. In this case, all identified origins are recorded for further forensic evaluation.

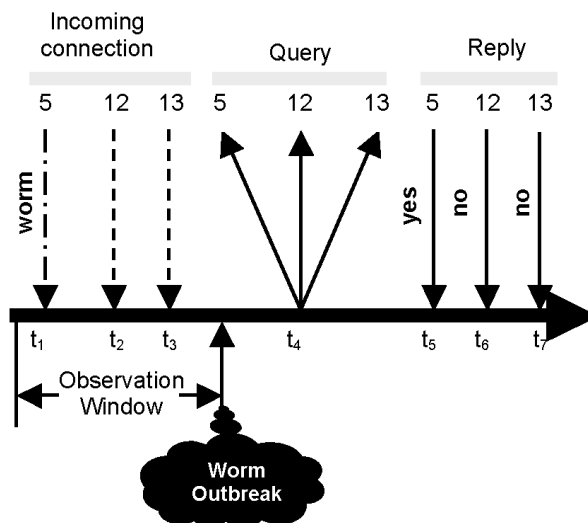
Figures 1 and 2 illustrates the case after a querying monitor has correlated data, and determined the source of the infection.

Figure 1 Dashed lines represent regular connection; solid lines represent worm-spread path



For any node, such as 9, once it detects worm outbreak, it initiates query phase then decides its predecessor in the reply phase. Each predecessor performs the same query, with the nodes detecting themselves as the source, notifying the local network administrator.

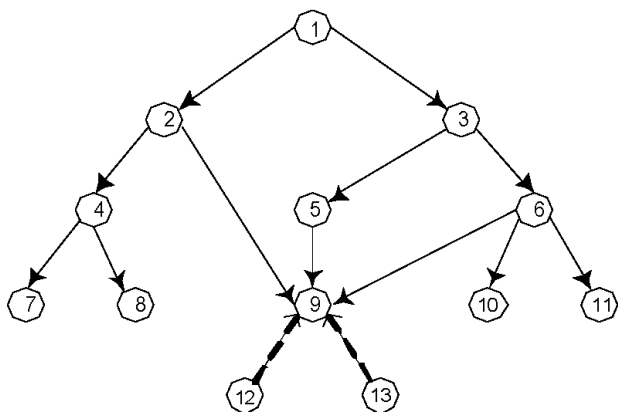
Figure 2 Timeline for node 9. Node 9 has regular connections from 5, 12 and 13, and is infected by 5 within observation window. Noticing worm outbreak, 9 queries every node in its incoming list at t4. 9 correlates events and determines 5 is its predecessor



It is possible for one monitor agent to receive yes reply from more than one predecessor. Arbitrarily picking the first yes reply suffices in the trace-back operation, because following any predecessor eventually leads back to the root of the infection tree. This scenario is illustrated in Figure 3.

During its observation window, node 9 receives malicious code from nodes 2, 5, and 6, in addition to regular connections from 12 and 13. Choosing any predecessor from {2, 5, 6} is fine here.

**Figure 3** Dashed lines represent regular connections; solid lines represent worm spread paths



### 3.3 Worm root report

The monitor agent's worm origin trace-back report contains the list of infected IP addresses and their respective time(s) of infection.

We leave it up to the individual enterprise whether or not to pursue legal actions after the infiltration.

Depending on the monitor agent installation ratio, it is possible that some of the source hosts to which the monitor sends query messages would not have a monitor agent in place. If so, those hosts won't participate in the trace-back computation, resulting in a different scenario.

Note that when an IDS fails to trigger, the scenario is similar to that node not having a monitor.

We analyse these scenarios in the following sections.

### 3.4 100% monitor agent installation case

Ideally, every site would have a monitor agent deployed at its gateway. This could be the case for example at corporate network or an Intranet. The monitor agent records connection information within the observation window, and queries source hosts to determine its worm predecessor or if itself is a worm origin. With ubiquitous monitor installation, every query message will be answered. In this ideal case, the worm origin will be the first host to be infected and thus the first (by NTP-based infection timestamp) to report itself as worm origin. In addition, paths from the worm origin's descendants will point back to itself.

This is confirmed in our extensive simulations (see Section 5). With universal monitor agent installation, we can always trace back to the very origin of the worm or all the origins if the worm is released at multiple locations.

### 3.5 Less than 100% monitor agent installation case

This is the more realistic scenario given the large number of distinct hosts and LANs on the internet. In this case, when a monitor agent queries hosts in its incoming list, some hosts which have monitor agents installed will reply, but others will not because there is simply no such reciprocal agent in place. As a result, the querying agent will be unable to

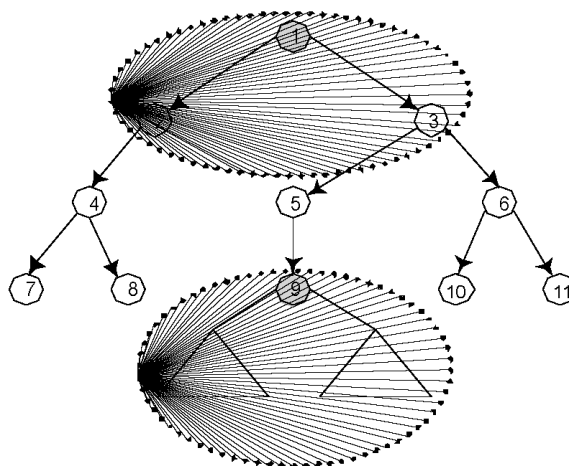
unambiguously conclude whether it has a predecessor or is a worm origin.

Consequently, instead of having one tree with a unique root (or multiple trees in case of multiple origins), we will have several forests within the network topology.

We observe that since all agents run the same distributed algorithm, the forest roots closer to the worm origin are more likely to report earlier than those farther away (and their local NTP-based infection timestamps will be earlier). In the case when the worm origin directly infects a host with a monitor agent installed, this infected host is likely to become a forest root. We can examine the union of forest roots and their timestamps to more accurately determine the worm origin.

For example, in Figure 4, suppose node 1 and 9 are non-monitor nodes, thus both 2 and 3 will report themselves as worm root networks or forest roots; another forest will likely report another forest root with source host 9. With the analysis of the union of all Origins monitors reporting worm root networks and their time stamps, we are likely to discover first monitored network outbreak, and its source IP.

**Figure 4** Grey nodes are non-monitor nodes and white nodes are monitor nodes



Identifying origins when there are multiple roots is a special case of the scenario under consideration; the special case being the root node not having a monitor.

In the case of most extant rapidly spreading worms, which exhibit unusual traffic patterns when they are infectious, this detection takes place in real time. In the case of a 'stealthy' worm that spreads using a list of known victims and has unobtrusive-looking connection patterns, this trace-back can be conducted as a post-mortem using knowledge of the infection vector and a database of connection history that goes back many months (or however long the infectious agent sat dormant on a given host). Clearly the 'stealthier' a worm is the harder it will be to detect, but if nothing else, deployment of a trace-back system such as this can deter worm creators from writing trivial worms, instead writing the harder, less-reliable stealthy kind of worm.

Our simulation results confirm that in the case of less than 100% monitor agents installation ratio, we can still pinpoint the worm origin to a small set of suspicious hosts. With a monitor agent installation ratio as low as 10%, we have a 45% chance to trace the worm back to its origin; when the ratio is above 30%, we have over a 95% chance of success, as described in Section 6.

## 4 Reporting

After a worm outbreak has been detected and the trace-back algorithm initiated, a simple text-based report is sent to the configured system administrator or security engineer. We rely upon the mitigation of the infiltrated network to take appropriate actions to the further prosecution of the offenders.

Further development could allow near real-time messages to be sent to preconfigured parties using instant messenger or e-mailed alerts.

The report contains the IP address and time of infection for all responding nodes. We envision the system administrators at infected sites collaborating to determine the vulnerability used to exploit the network as well as how to prevent the outbreak in the future.

## 5 Simulation and performance

Testing of the Origins algorithm was performed using two different methods. The first method used NS-2 simulation package to simulate the fast spreading worm; the second method used a live network, My Internet, coupled with real world worm to perform the simulation. These two independent methods yield results which agree with each other, confirming the validity of our algorithm.

### 5.1 NS-2 simulation

We first simulate a fast spreading worm using version 2.26 of NS-2 (NS-2 Homepage, 2003). NS-2 provides substantial support for simulation of TCP, UDP, routing, and multicast protocols over wired and wireless networks.

The requirements for simulating UDP and TCP worms, although different, are not radically so. Simulating TCP worms are more complex. UDP worms typically spread faster than TCP worms. This is because TCP worms have to wait for acknowledgments from the receiver before it can spread to the receiver of the worm message. While we can simulate UDP worms by using single packet transmission, we have to associate TCP worms with connection level semantics, such as three-way handshake, retransmission of lost packets, and acknowledgement of received data. Nevertheless, the focus of this paper is fast-spreading worms, regardless of their transport types. Our simulation environment allows us to accommodate the needed settings. For example, we used OC-12 links, which can handle high volume traffic, thus eliminating bandwidth as a differentiator between TCP and UDP worms. Furthermore,

NS-2 provides native support for TCP and UDP transport mechanisms. Our preliminary results for fast-spreading UDP worms confirm that our algorithm achieves similar results as those reported below.

We focus our simulation on Code Red worm, one of the fastest spreading TCP worm in history. The Code Red worm was spreading through IIS web servers on the internet via a buffer overflow exploit. The original worm would set up 100 threads on any infected host, use the first 99 threads to infect other web servers, and use the 100th thread to deface the system's website. In our simulation, we ignored the attacking part of the worm, and focused only on the spreading mechanism instead. In addition, we set the worm packet size for 4K bits, the size of Code Red worm.

We implement worm behaviour according to the classic SI epidemic model that describes the growth of an infectious pathogen spread through random contacts between susceptible and infected individuals. This model and its variance have been used to simulate fast spreading worms in many recent publications (Chen et al., 2003; Moore et al., 2003b). This model specifies that the number of new infections is determined by the product of the number of infective, contact rate, and the fraction of susceptible nodes. The following formula defines the SI model:

$$\frac{dI}{dt} = \beta \times I \times \frac{S}{N}$$

where

- $I(t)$ : infectives at time  $t$
- $\beta$ : contact rate
- $S(t)$ : susceptible nodes at time  $t$
- $N$ : the size of the total vulnerable population

In our simulation, we assume an infected node chooses its targets randomly from 32-bit IPv4 space. Consequently, we have:  $\beta = \text{Scan\_Rate} \times (N/2^{32})$  since one probe will reach a vulnerable host with probability  $(N/2^{32})$ .

We simulate 2000 vulnerable hosts because this is about the maximum number of nodes we can practically simulate given that we need to perform a large number of runs for any particular random configuration and average the results. However, when we vary the number of vulnerable hosts we find no noticeable inconsistency among the results.

To simulate the type of most aggressive worm whose spread rate is limited by bandwidth (Staniford et al. 2002; Moore et al., 2003a), we assume that the path between every pair of vulnerable hosts consists of 622 Mbps OC-12 links.

This gives:

$$\begin{aligned} \text{scan\_rate} &= (622 \times 106)/4096 = 151855 \\ \beta &= (\text{scan\_rate} \times N)/(232) = (151855 \times 2000)/(232) \\ &= 0.0707. \end{aligned}$$

The observation window is set to 10 s, and NTP accuracy margin is set to 0.5 s. We further assume a vulnerable host can only be infected once. The time between a vulnerable

host's first infection and beginning to spread outward is set to one second.

To better model the real-world settings in which a worm actually spreads, we extract internet traffic pattern data from the data set maintained by Lawrence Berkeley National Laboratory (Internet Traffic Archive Homepage), and superimpose it in our NS-2 simulation. This allows us to more accurately simulate fast spreading worms by taking into consideration real-world network dynamics and usage characteristics.

We consider three metrics related to how effective our algorithm performs under varying conditions: *Success rate* is defined as the probability that our algorithm can successfully trace-back to the worm origin. It is calculated as the ratio of the number of simulation runs which resulted in successful trace-back to the total number of simulation runs. *# of sufficient forest roots* indicates the number of earliest forest roots we need to examine in order to find the worm origin. *Max distance* is the longest hop distance between worm origin and any of the forest roots.

Table 1 shows *success rate*, *# of sufficient forest roots* and *Max distance* vs. different monitor agent installation

ratios. The results are averaged over 100 rounds of simulation for every configuration. We can see that when the monitor agent installation ratio is very low, such as 10%, there is a 25% chance of successfully pinpointing the worm origin to a small set of suspicious hosts, which are the union of hosts in the incoming lists of around 35 forest roots. With the increase of the monitor agent installation ratio, the success rate also increases while the # of sufficient forest roots decreases. The success rate stabilises to above 95% when the monitor agent is installed on greater than 30% of networks. This is because when the monitor agent installation ratio is higher, it is more likely that the worm origin will hit a monitor agent directly within its first few infections, and as long as there is a direct hit, the worm origin is captured within the target's observation window. For example, when the monitor agent installation ratio equals 30%, the first worm packet from the worm origin will have a 30% chance to hit a monitored node; the first two will have 51% ( $1 - 0.7^2 = 0.51$ ); the first seven will have 92% ( $1 - 0.7^7 = 0.92$ ); This analysis agrees well with our simulation results in Table 1.

**Table 1** Performance results using NS-2

Monitor agent percentage (p)	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Success rate	25%	85%	85%	100%	95%	100%	100%	95%	100%	100%
Sufficient # of forest roots	35	17.1	16	14.7	11.5	10.6	8.7	7.7	7.4	1
Max distance	6.8	2.9	3.6	3.5	3.8	3.9	3.8	3.7	4	1

**Table 2** Performance results with My Internet

Monitor saturation	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Success rate	25.0%	37.5%	62.5%	75.0%	87.5%	100%	87.5%	87.5%	100%	100%

The set of actual hosts an investigator needs to examine for further trace-back is the union of hosts in the incoming list of sufficient forest roots. For example, when  $p = 60%$ , *# of sufficient forest roots* = 10.6, which implies that an investigator needs to examine all the distinct hosts contained in the incoming list of about 11 hosts within the observation window time frame.

It is confirmed in our simulation results that when every host has a monitor agent in place, we can always successfully trace back to the worm origin.

Table 1 also shows the maximum distance remains generally constant. This is because hosts farther from the root will spread the worm to non-monitored hosts, therefore reducing the chance for monitored but uninfected host to be directly infected by the worm origin.

Besides simulating the TCP worm, we also simulated Sapphire, a UDP worm. The packet size was changed to 404 bytes which is the size of the Sapphire worm with headers. This results in a scan rate of  $\text{scan\_rate} = ((622 \times 10^6)/(404 \times 8)) = 192450$  and a contact rate of  $\beta = \text{scan\_rate} \times (N/2^{32}) = 192450 \times (2000/2^{32}) = 0.0896$ . We found that the results are inline with what is reported here.

## 5.2 My Internet simulation

Additional simulation was performed using a live network, dubbed My Internet, with a handcrafted TCP worm whose size is roughly the size of the smallest allowed TCP packet. Table 2 summarises the results from this simulation. The live network consisted of five individual machines, each running a number of Origins monitors. Each monitor records the LAN's incoming IP addresses and performs the trace-back computation when an outbreak is detected.

The network topography of My Internet was a hub and spoke network with the 10.0.0.0/8 address space divided into separate subnets. Different subnets were used for different tests. One machine was used to monitor network traffic and store control data. Each of the remaining 4 host machines was configured with 25 network interfaces. Each network interface simulates a local network, allowing 100 different subnets to be configured for testing.

The experiments were conducted as follows. A fast spreading internet worm was created to give an accurate depiction of the worm propagation tree throughout My Internet. Origins monitor installation varied from 100% saturation (a monitor on every interface in My Internet) to

10% saturation (ten monitors distributed uniformly throughout My Internet). A random subnet was used for the start of worm propagation.

We also injected actual internet traffic data, obtained from Lawrence Berkeley National Laboratory (Internet Traffic Archive Homepage), into My Internet. The IP headers were analysed using a pcap-based (TCPDump/Libpcap Homepage) file analyser to determine source and destination addresses, which were further translated to live hosts on My Internet. RawIP (RawIP Networking FAQ Homepage, 1999) is used to transmit actual packets. With this method, we were able to accurately simulate traffic flowing among all nodes on My Internet.

A configurable sliding window time of 5 min is used for simulations. We analysed the data returned by the Origins monitors and compared the worm root result to the actual experimental data. We varied the saturation percentage of origins monitors, and conducted two tests with each network topologies at every saturation level. As each round took considerable time to run, we had to restrict the number of rounds and the total number of configurations. If the actual worm origin was reported as the worm root, a success was recorded. We also recorded a success if there were multiple forest roots found and one of those was the worm root, because collaboration between forest roots would lead to the actual origin. Failures were recorded when the worm root network was not identified as a forest root.

Preliminary results agree with those obtained with NS-2 simulations. With 100% deployment, the worm origin is accurately identified 100% of the time. With 90% saturation, the worm root was accurately identified 100% of the time. As monitor saturation decreased, the accuracy of the worm root also decreased, but at a rate greater than that of NS-2 simulations. We attribute this small discrepancy to the limited number of tests performed for each saturation level.

We used NTP (Network Time Protocol Project Homepage, 2006; Mills, 1992) to correlate events between Origins monitors. Each monitor sets an NTP offset on start-up and uses this offset to calculate timestamp information. Synchronisation of NTP between Origins monitors has not been an issue in simulations, even when different NTP sources are used. We attribute this to the accuracy of stratum 1 time servers with reference to each other.

## 6 Discussion and future work

### 6.1 False negatives and positives

A false negative is the case when there is a worm root, but Origins fails to detect it. There are two possible causes for this: failure to detect worm outbreak at a node or probabilistic failure due to inadequate monitor installation ratio. Failing to detect at a given link is treated the same as the upstream node incident on that link having no monitor. When the monitor installation ratio is less than 100%,

although the probability is very low as demonstrated in Section 5, it can happen that all the immediate infection targets of the root can be non-monitors, causing Origins to fail to report worm root. However, even in this case, Origins is still of value because it reports forest roots which are *closest* to the true worm root. This set can be used to narrow down the scope of further investigation.

A false positive is the case when there is not a worm root, but Origins reports one. This happens only when the IDS falsely triggers the query and reply processes. Therefore, it depends on the accuracy and resiliency of the existent IDS. At worst, the number of forest roots to be examined and correlated to determine the true source increases by a small amount.

Additionally, monitor distribution is also a factor in false negatives and positives rate. Since the current work is intended to be a proof-of-concept, we assume uniform distribution in our simulations. When susceptible hosts or monitor agents are not uniformly distributed, the success rates may differ. We plan to research the efficacy of our algorithm under distributions other than uniform in the future.

### 6.2 Privacy concerns

Given the need to protect against corporate network intrusion, many companies have already deployed IDSs. These systems receive a mirror copy of all incoming and outgoing connection data. By allowing the IDS to detect the worm and transfer the payload to Origins, Origins is only required to monitor the incoming source IP addresses. Thus, the invasion of privacy is very limited.

### 6.3 Deployment issues

While the current implementation of Origins is geared toward deployment on a site's LAN by system or corporate administrators, an additional avenue could be ISPs, monitoring all of their customers. Deployment at an ISP would lack the ability to determine which host on a site is infected when Network Address Translation is used, but since it is likely that the ISP has access to all data packets entering or leaving the customer site, the IDS and Origins monitor can be treated and contained as a single host for detection and trace-back. Mixing ISP-based and LAN-based monitoring would allow a greater effective deployment ratio with fewer actual deployments.

At first glance, Origins seems to require relatively high adoption ratios in order to be effective (above 10%). However, considering the fact that Cisco controlled 63% of the core router market and 80% of the IP aggregation router market in North America, or the quick rise in market penetration of software such as Apache and Sendmail, we believe satisfactory monitoring can realistically be achieved. In addition, Origins can be easily ported to run on a network gateway router and could be included in new versions of a router's IOS.

#### 6.4 NTP issues

Origins relies on NTP for accurate time correlation. On start-up, Origins calculates a time offset using NTP. This offset is used in all timestamp computations. The NTP offsets are updated at a predetermined time interval, or if there is a change in the system clock.

It may be possible for an adversary to design a specially crafted worm that will infect the Origins monitors and change the system time functions, in effect causing incorrect or invalid timestamps. This case is very unlikely as IDS servers are required to have a high level of security set due to the sensitive information they receive.

Even an adversary directly attacking NTP servers with a DDoS would not prevent Origins from functioning correctly.

First, NTP is a robust distributed system consisting of 300 or more primary and secondary timer servers geographically dispersed in many countries (Public NTP Primary (stratum 1) Time Servers Homepage). By design, NTP is resilient against large-scale DDoS attacks.

Second, while constant availability of NTP is important, it is not imperative to maintain because once monitors are synchronous with NTP time servers, the drift of a given local clock is negligible within a relatively short time period. In other words, if the local clocks obtain the time some time before the worm outbreak, the time drifts should be negligible during the lifetime of a fast-spreading worm.

#### 6.5 Origins traffic

Another valid concern is whether traffic generated by Origins will compound the effect of worms. We contend that the amount of such traffic is insignificant.

The Origins packets are designed to be very small. A query packet can be as small as the payloads of currently active worms plus the IP header information. Reply packets are simply the INFECTED or NOT\_INFECTED response with timestamp and worm infection identification.

Queries from a particular monitor are sent only during the query phase to a small number of hosts. In addition, only an infected host with monitor installed will generate query messages. Origins monitors only produce traffic during the query phase.

In addition, since queries and replies happen only between nodes within immediate levels in the infection tree, the number of overall messages is greatly reduced.

#### 6.6 Other issues

Deliberate attempts to subvert the detection algorithm are possible. For example, if a worm operated in a step-stone approach, infecting one node which acts harmless but slowly attempts to infect a second who acts rapidly, (i.e., every second level of the tree operating rapidly), then this would make each second level of the tree appear to be a root. However, this would have two effects, one of slowing down the worm's spread and secondarily of generating a large number of reported roots.

Notwithstanding the high volume of data, the NTP-based time stamps would still serve to identify the earliest roots.

## 7 Conclusion

Rapidly spreading worms bring a new level of threat to the internet. Because of their speed in spreading across the globe, automated response mechanisms to detect and contain them are vital to the security of networks.

In this paper, we describe the design of Origins, an automated, distributed mechanism to identify the roots of propagation of fast spreading worms.

Ultimately, because of the increased threat and actual economic damage posed from worm infections, automated tools such as this are required. While the Origins system described above is merely an initial foray into the waters of automated trace-back, it does exemplify that key aspects of this problem are not only possible, but tractable in real time.

## Acknowledgements

The authors would like to thank the anonymous referees for their insightful comments, which improved the presentation and focus of this work.

Dr. Thurimella's research was funded in part by the NSF under Grant No. DUE – 0416969 and an IBM Faculty Development Award. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect those of the NSF.

## References

- Arkin, O. (2002) *Trace-Back: A Concept for Tracing and Profiling Malicious Computer Attackers*, Whitepaper, Atstake Limited, <http://www.sys-security.com/archive/papers/traceback.pdf>.
- Burch, H. and Cheswick, B. (2000) 'Tracing anonymous packets to their approximate source', *Proceedings of USENIX Large Installation System Administration Conference 2000*, New Orleans, LA, pp.319–327.
- Chen, Z., Gao, L. and Kwiat, K. (2003) 'Modeling the spread of active worms', *Proceedings of IEEE INFOCOM 2003*, San Francisco, CA, Vol. 3, pp.1890–1900.
- Cheung, S., Crawford, R., Dilger, M., Frank, J., Hoagland, J., Levitt, K., Rowe, J., Staniford-Chen, S., Yip, R. and Zerkle, D. (1999) *The Design of GrIDS: A Graph-Based Intrusion Detection System*, Technical Report CSE-99-2, University of California Davis – Computer Science Department, <http://citeseer.nj.nec.com/cheung99design.html>.
- Dean, D., Franklin, M. and Stubblefield, A. (2001) 'An algebraic approach to IP traceback', *Proceedings of the 2001 Network and Distributed System Security Symposium*, San Diego, CA, pp.1–10.
- Honeynet Project Homepage (2002) *Know your Enemy: Passive Fingerprint, Identifying Remote Hosts, Without Knowing* [www], <http://project.honeynet.org/papers/finger>.
- Internet Engineering Task Force Homepage (2003) *ICMP Traceback (itrace)* [www], <http://www.ietf.org/html.charters/OLD/itrace-charter.html>.

- Jung, J., Paxson, V., Berger, A.W. and Balakrishnan, H. (2004) 'Fast portscan detection using sequential hypothesis testing', *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, pp.211–225.
- La Monica, P.R. (2003) *Microsoft: Bounty Hunter. The World's No. 1 Software Company Announces a \$5M Reward Program to Help Catch Virus Authors* [www], <http://money.cnn.com/2003/11/05/technology/microsoftbounty>.
- Mills, D.L. (1992) 'Network time protocol (version 3) specification, implementation and analysis', *The Network Time Protocol Project Documentation Homepage*, <http://ntp.isc.org/bin/view/Support/Rfc1305>.
- Moore, D., Paxson, V., Savage, S., Shannon, C., Staniford, S. and Weaver, N. (2003a) *The Spread of the Sapphire/Slammer Worm*, <http://www.caida.org/publications/papers/2003/sapphire/sapphire.html>.
- Moore, D., Shannon, C., Voelker, G.M. and Savage, S. (2003b) 'Internet quarantine: requirements for containing self propagating code', *Proceedings of IEEE INFOCOM 2003*, San Francisco, CA, Vol. 3, pp.1901–1910.
- Moore, D., Voelker, G. and Savage, S. (2001) 'Inferring internet denial-of-service activity', *Proceedings of the 10th USENIX Security Symposium*, Washington DC, pp.9–22.
- Nazario, J. (2004) 'Defense and detection strategies against internet worms', *Artech House*, Boston, MA, ISBN 1-58053-537-2.
- Network Time Protocol Project Homepage (2006) *The NTP FAQ and HOWTO: Understanding and using the Network Time Protocol* [www], <http://www.ntp.org/ntpfaq/NTP-s-algo.htm>.
- NS-2 Homepage (2003) *The Network Simulator – NS-2* [www], <http://www.isi.edu/nsnam/ns/>.
- RawIP Networking FAQ Homepage (1999) *RawIP Networking FAQ* [www], <http://www.ntua.gr/rin/rawfaq.html>.
- Reiss, S. (2003) 'Hope is a lousy defense – interview with bill joy', *Wired Magazine*, No. 11.12, pp.8–11.
- Snoeren, A., Partridge, C., Sanchez, L., Jones, C., Tchakountio, F., Schwartz, B., Kent, S. and Strayer, W. (2002) 'Single-packet IP traceback', *IEEE/ACM Transactions on Networking (ToN)*, Vol. 10, No. 6, pp.721–734.
- Staniford, S., Paxson, V. and Weaver, N. (2002) 'How to own the internet in your spare time', *Proceedings of the 11th USENIX Security Symposium*, San Francisco, CA, pp.149–167.
- Toth, T. and Kruegel, C. (2002) 'Connection-history based anomaly detection', *Proceedings of the 2002 IEEE Workshop on Information Assurance and Security*, West Point, NY.
- Weaver, N., Staniford, S. and Paxson, V. (2004) 'Very fast containment of scanning worms', *Proceedings of the 13th USENIX Security Symposium*, San Diego, CA, pp.29–44.
- Williamson, M.M. (2002) 'Throttling viruses: restricting propagation to defeat malicious mobile code', *Proceedings of the 18th Annual Computer Security Applications Conference*, Las Vegas, NV, pp.61–68.
- Williamson, M.M. (2003) 'Design, implementation and test of an email virus throttle', *Proceedings of the 19th Annual Computer Security Applications Conference*, Las Vegas, NV, pp.76–85.
- Xie, Y., Sekar, V., Maltz, D., Reiter, M. and Zhang, H. (2005) 'Worm origin identification using random moonwalks', *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, pp.242–256.

## Websites

- Internet Traffic Archive Homepage, <http://ita.ee.lbl.gov/>.
- Network Time Protocol Project Homepage, <http://www.ntp.org/>.
- Public NTP Primary (stratum 1) Time Servers Homepage, <http://www.eecis.udel.edu/~mills/ntp/clock1a.html>.
- TCPDump/Libpcap Homepage, <http://www.tcpdump.org/>.