Automated detection and containment of worms and viruses into heterogeneous networks: a simple network immune system

Francesco Palmieri* and Ugo Fiore

Centro Servizi Didattico Scientifico,
Università degli Studi di Napoli Federico II, Napoli, Italy
E-mail: Francesco.Palmieri@unina.it
E-mail: Ugo.Fiore@unina.it
*Corresponding author

Abstract: While much recent research concentrates on propagation models, the defence against worms is largely an open problem. Classical containment strategies, based on manual application of traffic filters, will be almost totally ineffective in the wide area since the worms are able to spread at rates that effectively preclude any human-directed reaction. Consequently, developing an automated, flexible and adaptive containment strategy is the most viable way to defeat worm propagation in an acceptable time. As a case in point, we look to natural immune systems, which solve a similar problem, but in a radically different way. Accordingly, we present a cooperative immunisation system inspired in principles and structure by the natural immune system that helps in defending against these types of attacks. Our system automatically detects pathologic traffic conditions due to an infection and informs, according to a cooperative communication principle, all the reachable networked nodes about the ongoing attack, triggering the actions required to their defence. To evaluate our proposal, we formulated a simple worm propagation and containment model, and evaluated our system using numerical solution and sensitivity analysis. Our measurements show that our reaction strategy is sufficiently robust against all the most common malicious agents. We envision that the above solution will be an effective line of defence against more aggressive worms.

Keywords: worms; viruses; automatic detection/containment; network immune systems.

Reference to this paper should be made as follows: Palmieri, F. and Fiore, U. (2007) 'Automated detection and containment of worms and viruses into heterogeneous networks: a simple network immune system', *Int. J. Wireless and Mobile Computing*, Vol. 2, No. 1, pp.47–58.

Biographical notes: Francesco Palmieri received two Computer Science degrees from Salerno University, Italy. Since 1997 he leads the network management/operation center of the Federico II University, in Napoli, Italy. He has been a Member of the Technical Scientific Committee and the CSIRT of the Italian NREN GARR. He is an active researcher in the fields of high performance/evolutionary networking and network security.

Ugo Fiore received his degree in Physics from Federico II University, Italy. He joined the network management/operation centre of Federico II University in 2001. His current research interest is modelling and design of algorithms and protocols aimed at increasing performance and security in the core networks.

1 Introduction

Computer worms and viruses are the first and the only form of 'artificial life' to have had a measurable impact on our society, since it has been widely experienced that the massive worldwide spreading of very fast and aggressive worms may easily disrupt or damage the connectivity of large sections of the internet, affecting millions of users. There are few reactions to the above threat. Risk may be kept at a minimum by applying the patches that remove the security defects exploited by worms and viruses in their propagation, as soon as those patches are made available. But software bugs seem to always increase as computer systems become more and more complicated and not all people have the habit of keeping an eye on the patch releases or engage themselves to constantly keep their systems up-to-date. What's worse, the relatively homogeneous software base in almost all the networked nodes in the internet, coupled with the current high-bandwidth connectivity, provides an ideal climate (Nachenberg, 2000) for self-propagating attacks. Furthermore, the ability of worms to spread at rates that make very difficult or actually preclude human-directed reaction has elevated them to a first-class security threat to

all networked systems. The emergence of mobile computing makes things worse, since it allows malicious code to move from network to network without traversing any firewall or border protection. There are thousands of residential publicly available access points, most with minimal or no security, since they lack any reasonable access control and are thus extremely vulnerable to anyone who wishes to use them. A mobile computer may be plugged into an already infected or unprotected network, infected by a worm and then disconnected without any evidence. Later it may be brought into a protected office environment, where it connects to the network behind the firewall. Its packets are never examined for the worm, since they do not go through the firewall and it is thus free to infect the entire network. In response to these threats, there is considerable effort focused on developing technical means for detecting and containing worm infections before they can cause such damage. Routers and firewalls on the network border may be configured to block incoming packets whose content contains worm signatures or the ports used by the worms in their diffusion. Still, that may only happen after a worm has spread and has been analysed and this may take too much time to be effective with next-generation fast-spreading worms. Users attempt to shield their own machines from such attacks by using antivirus programs and personal firewalls, with a mixed record of success at the best. One of the main problems with these solutions is that they rely on manual configurations and human intervention and may fail to react in time to defend against an attack. The main computer worm and virus threat is in their very fast self-replication and spreading activities across the infected networks and the only way to cope with them is deploying the proper containment strategies through the networks at the same speed or at a comparable one. Thus, there are strong reasons to believe that it is possible to build better computer security systems by adopting design principles that are more appropriate for the imperfect, uncontrolled and open environments in which most computers currently exist.

Most computer security issues can be viewed as the problem of distinguishing *self* (legitimate traffic, authorised actions, original source code, uncorrupted data, etc.) from *non-self* (intruders, computer viruses, spoofing, worms, etc.). Nature, more specifically the natural immune system, has been solving a similar problem for hundreds of millions of years, using methods quite different from those typically used to protect computers and networks. For example, consider the human immune system. It is composed of many unreliable, short-lived and imperfect components. It is autonomous. It is not 'correct', because it sometimes makes mistakes. However, in spite of these mistakes, it functions well enough to help keep most of us alive for 70 years, even though we encounter potentially deadly parasites, bacteria and viruses every day. Accordingly, to address the widespread worm and virus infection problems, we propose a network immune system that takes much of its inspiration from nature, thinking that a deeper understanding of the natural immune system can help us design a more robust and practical 'computer immune system'. Such a system would incorporate many elements of current security systems, augmenting them with an adaptive response layer. Parts of this layer might be directly analogous to mechanisms present in the immune system; others will likely be quite different from those found in biology, even if they are based on similar principles. The nodes in our system cooperate in detecting and informing each other of ongoing attacks and of the actions necessary to the defence, driving, when possible, the automatic software update to fix the exploited vulnerabilities on the infected hosts. To evaluate our proposal, we formulated a simple worm propagation and containment model, based on the above principles and evaluated our system using numerical solution and sensitivity analysis. Our observations show that our framework seems to be robust and effective against viruses and worms. From the above experience, we argue that building self-reacting distributed containment systems that, like the natural immune systems, can detect and prevent in a matter of minutes widespread network infections will be one of the most promising and challenging lines of defence against next-generation more aggressive worms.

2 Worms and viruses

A worm is a self-replicating, segmented and distributed computer program spreading from host to host via an available network connection. Most often, worms exploit vulnerabilities in the host computer's operating system or network service handlers that can accept network requests or outside connections. More properly, a worm has been defined (Nazario et al., 2001) as a software component that is capable, under its own means, of infecting a computer system and using it, in an automated fashion, to infect other systems. Worms are viruses that can spread on their own. In contrast, viruses rely on passive means of transfer. For example, a virus can spread by tricking a user into executing an e-mail attachment or otherwise executing an infected file. When the file is copied from an infected host to another host, it will infect the new host when the file is opened for the first time. Thus, malicious agents that spread via human interaction are not typically classified as worms. Worms were originally considered a benign paradigm to ensure the longevity of distributed applications and originally ran only on machines that supported either general or special purpose remote execution facilities. Two factors have changed both the perception and reality of worms to be largely malignant platforms for distributed applications. Firstly, even when programmers' motives are pure, small bugs can cause worms to proliferate and grow more rapidly than it was desired and overwhelm the resources of a distributed remote-execution system, as in fact occurred on the internet in November 1988. Secondly, most worms or viruses no longer use legitimate remote execution interfaces to acquire a bounded number of nodes. Rather, they exploit bugs and loopholes and install themselves on machines where they are unwanted. They often try to grow without bound, attempting to infect every machine accessible to them. In the best case, they simply steal CPU

cycles. However, they can easily have more destructive payloads: delete files and/or otherwise damage the host machines, steal sensitive data, participate in a denial of service attack, even act as vehicles for the massive diffusion of other mischievous software such as backdoors, etc. Recent incidents have demonstrated the ability of worms and other malicious agents to infect large numbers of hosts, exploiting vulnerabilities in the broadly homogeneous deployed software base (Zou et al., 2002) and causing immense damage to the network-dependent commercial, military and social services infrastructure of nations throughout the world. Thus, even when the worm carries no malicious payload, the direct cost of recovering from the side effects of an infection epidemic can be substantial. Thus, countering worms has recently become the focus of increased research. The improved connectivity of modern computer systems exacerbates the problem, because viruses and worms can now use networks as a very effective medium for their propagation. They can sweep quickly through thousands of hosts, an effect that is far more damaging than what would occur in a more traditional, stand-alone or local-area networked computing environment. As the number of connected devices increases, the threat of self-propagating computer malicious agents grow despite the industry's best efforts to eradicate them. The analogy between worm and virus-related computer security problems and biological processes was recognised as early as 1987, when the term 'computer virus' was introduced (Cohen, 1987). Indeed, many parallels can be drawn between biological systems and computer networks. Thankfully, computer worms currently are not as effective at spreading as their real life counterparts because of the passive nature of biological infections (Nazario et al., 2001). This is because biological agents can survive in the transport media such as blood, bodily fluids, etc., just waiting for a host to pick them up. Computer worms rely on active methods of infecting host that requires searching through the network (scanning) for their next target. The analogy of the internet worms to that of living systems is not lost when it comes to classifying how worms behave. Albanese et al. (2004) describes a system that they believe can be used to classify any past, present or future worm. Based on life functions, the criteria used to classify worms are:

1 infection
2 survival
3 propagation and
4 payload.

Infection is accomplished using two general methods – the worm either exploits a flaw in the software on a running system or a user's action on the system executes the worm code. The second life function, survival, is essential for any autonomous agent that wishes to spread throughout a system or network. To this end, many worms employ techniques that are designed to hide their existence and foothold on a system. The longer a worm can operate undetected, the more likely it is to accomplish its objectives. Survival alone is not enough. The worm must also be able to spread effectively and this is something that can be done only once a worm has gained control of the host. Many viruses work by harvesting e-mail addresses on the host computer and sending e-mails with infected attachments to these addresses; when the attachment is executed, the virus infects the new host and begins its life functions all over again. With the popularity of music and application file sharing on networks such as Kazaa or Napster, it is becoming easier to trick users into downloading infected files. Once the files are executed or viewed, the worm is unleashed. The newly infected host then places more infected files on the file-sharing network. Additionally, media players such as WinAmp are showing that they too are vulnerable to buffer overflow exploits from carefully crafted music play lists. Finally, the most aggressive tactic employed by computer worms is the scanning of remote computers for known vulnerabilities to exploit. Many methods of detecting worm activity are based on the detection of anomalous network traffic associated with scanning. Worms that have the ability to coordinate their propagation and control other infected machines can also accomplish many complex propagation tasks using just a small set of simple abilities. The ability to communicate on the network is vital to nearly every other worm function. Any capability that requires sending or receiving information relies on being able to communicate on the network. Care is often taken to hide these communication channels, even by means of encryption. As wireless networks are becoming increasingly common and the internet is evolving towards a model based on nearly ubiquitous coverage and transparent mobility, the availability of wireless PDAs and cell phones powerful enough to download and run Java code is creating new powerful and often hidden vehicles and possibilities for infection propagation. In fact, with wired networks, there is an explicit step that a user takes to connect up to the protected network: he plugs in a cable. Perhaps users can be educated to regard this as an event with security significance, requiring them to perform extra actions, such as a virus scan. With wireless networks, there is no such user-initiated event. Typically, wireless nodes automatically detect and join local wireless networks and typically users do not necessarily even know it has happened.

On the other side, the most obvious defence against worms and viruses is to prevent their attacks by repairing the vulnerabilities they are based on, before those can be exploited. Typically, software vendors develop and distribute reparative patches to their software as soon as possible after learning of a vulnerability. Customers can then install the patch and prevent attacks that exploit the vulnerability. Software patching has not been effective as a first-line defence against large-scale worm attacks, even when patches have been available long before the worm outbreak. Generally, people have been reluctant to patch their systems immediately, because patches are perceived to be unreliable and disruptive to apply. Experience has shown, in addition, that administrators often do not install patches until long after they are made available, if at all (Rescorla, 2003). As a result, attacks such as the widely publicised CodeRed, SQL Slammer, Blaster and Sasser worms, that exploit known vulnerabilities, for which

patches had been available for pretty some time, have nevertheless been quite successful, causing widespread damage by attacking the large cohort of still vulnerable hosts. In fact, more than 90% of the attacks today are exploiting known vulnerabilities (Arbaugh et al., 2000).

Consequently, to prevent widespread infection in the internet, any viable containment strategy will require automated detection of pathological traffic anomalies generated by infections and triggering, via a cooperatively deployed communication facility, immediate system updates (by applying all the available patches) on the networked host to ensure just-in-time reaction to worm epidemics.

3 The detection and containment paradigm

Traditional antivirus techniques focus typically on detecting the static signatures of viruses. While these techniques are somewhat effective in their own right, they do not address the dynamic nature of a worm infection within the context of the underlying system. In a computer network, a worm can propagate through the network quickly and it might infect and damage many, perhaps all, machines before the severity of the situation is recognised. A valuable mechanism for mitigating the damage would be to detect the presence of an infection in a network at an early stage and to have the network react to the attack in real time. A number of challenges exist in developing such a scheme. On one side, activities such as signature matching, static access control and formal verification on system configuration and current state are not really effective in coping with the extreme flexibility and adaptability of recent worms/viruses, operating in the highly dynamic modern computer and networking environment. Computers and in particular the networked ones, are not static systems: vendors, system administrators and users constantly change the state of a system. Programs are added and removed and configurations are changed. Clearly all the above verification activities will be at least impractical on such a dynamic system making any ‘perfect’ implementation of security policies based on access controls, classic firewalls and audit trails impossible. Accordingly, we believe it is necessary to build better computer security systems by adopting new design principles that are more appropriate for the above described imperfect, uncontrolled and open computer and networking environment. Natural immune systems provide a rich source of inspiration for computer security in the age of the internet, since they have many features that are desirable for the environments in which most computers currently exist. These include distributability, diversity, disposability, adaptability, autonomy, dynamic coverage, anomaly detection, multiple layers, identity via behaviour, no trusted components and imperfect detection. Immune systems in nature are made primarily out of cells, which monitor and interact with other cells. It can be viewed as a distributed detection system that consists primarily of special independent detector or ‘sentinel’ cells, called lymphocytes. Moving the analogy up another level, we could regard each computer as a cell, with a network of mutually trusting computers being the organism. Host-based security, such as system hardening or better, patching, can be thought of as the normal defences a cell has against attack. The innate immune system corresponds to the network’s defences, such as packet filtering routers and firewalls. We can implement an adaptive immune system layer by creating a set of coordinating detector machines acting as lymphocytes in the natural immune system. These machines would monitor the state of other machines on the network. When an anomaly was detected, the problematic machine could be isolated (perhaps by reconfiguring firewalls and/or routers filtering policies), reupdated and rebooted or shut down. If the true source of the anomaly were outside the network, a ‘lymphocyte’ border node could block or rate-limit the incoming traffic flows associated to the hostile activity to protect the internal machines, actively doing battle with the outside malicious host and preventing or at least slowing the infection spread. Lymphocytes in the immune system are able to determine locally the presence of an infection. No central coordination takes place, which means there is no single point of failure. The immune system is highly adaptable and has the ability to detect pathogens that it has never encountered before, that is, it performs anomaly detection and retains the ability to recognise previously seen pathogens through its immune memory. A computer immune system should be similarly adaptable, both learning to recognise new intrusions and remembering the evidences and discriminators of previous attacks. We believe that the ability to detect intrusions or violations that are not already known is the most important feature of any security system. Our architecture, makes use of three types of components: a set of anomaly detection sensors, also called *detectors* acting as ‘sentinel’ lymphocyte cells, a cooperative communication facility between the components and a software update component, that can also be called *effector* that will be installed on each host and can be used to drive software patching, via the proper automatic update procedure when signaled in consequence of an ongoing infection. This last component act as the natural capacity of an individual cell to self-repair itself and in most cases become immune. Several types of sensors may be employed concurrently:

> Passive sensors, installed on independent boxes, perform eavesdropping on traffic to and from the hosts operating on their network segment or at least on their routing domain to immediately identify any anomaly or condition due to a probable worm outbreak and to inform all the modes participating to the artificial immune system about the dangerous condition, eventually triggering, when accepted, an automatic update in the installed software base.
>
> Active sensors, operating on the corporate routers and firewall routers, actively monitor the traffic flows passing through them and apply, when an anomalous traffic condition generated by a worm attack is detected, the proper countermeasures in terms of IP or port based filtering or rate-limiting. Active sensors, when handling a detected infection process, operate

like passive sensors in cooperatively spreading the alert information through the nodes participating to the network immune system to ensure just-in-time triggering of the proper actions.

The whole paradigm is simply sketched in Figure 1 below.

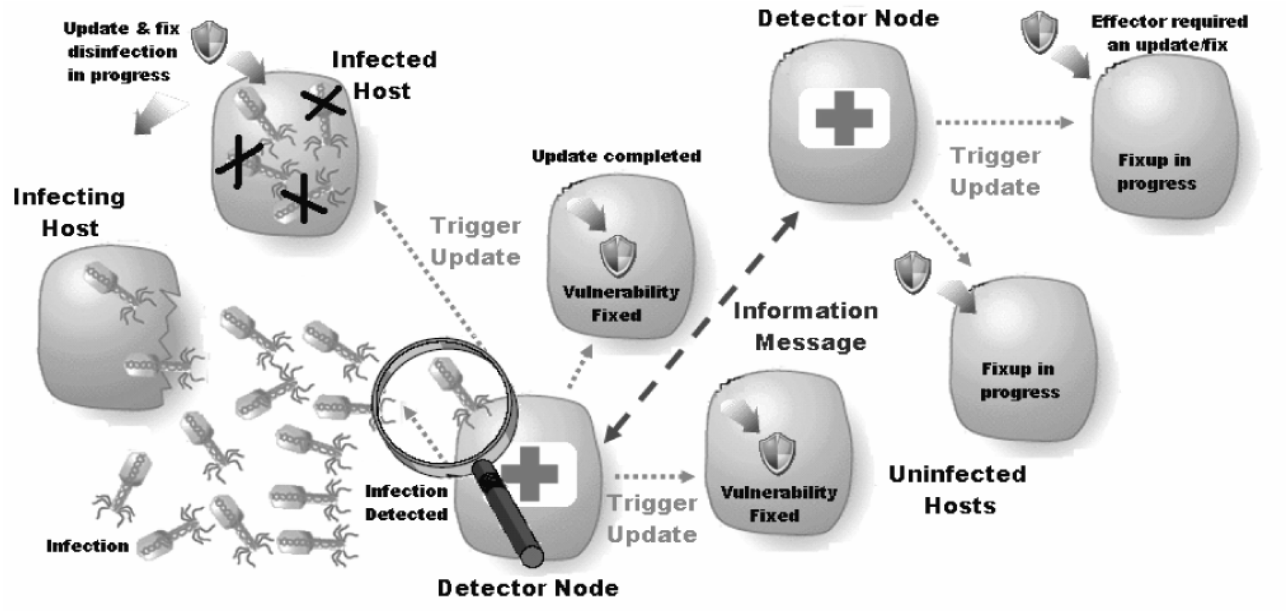
3.1 Anomaly detection and first epidemic containment

Most worms that have been observed so far in the internet have the following common characteristics. Firstly, they generate a substantial volume of identical or similar traffic to the targets, although polymorphic worms may not follow this pattern. Secondly, the worm infects vulnerable hosts for propagation. Thirdly, many worms, for example, Code Red and SQL Slammer, use random scanning to probe vulnerable hosts. Therefore, scans generated by this type of worm can reach inactive IP addresses. Our anomaly detection strategy focuses on the above characteristics and is based on continuously analysing some properly chosen health parameters, directly reflecting the network behaviour in the presence of worms and checking them against a 'sanity' per-time limit threshold. The most significant parameters used in our anomaly detection facility are the *outgoing flow* and the *connection failure rate*.

The outgoing flow of each node represents the number of outbound nodes that an internal machine may contact per time interval. This choice is motivated from the observation that during normal operation the volume of outbound flows to unique machines is relatively small and that this volume generally greatly increases when a machine is infected by a scan-based worm in proportion to the aggressiveness with which the worm seeks susceptible nodes. The assumption holds very well for random-scan worms, but not as well for topology, contagion, flash worm victims or slow spreading worms that may not necessarily provoke an abrupt raise in the outbound communication patterns of their victims. However, by choosing a good threshold value and using some well-crafted heuristics to discriminate some classical worm behaviours, like trying access, eventually according to a locality principle, to the same set of ports on many hosts on the network, the above technique may reach a satisfactory level of success also with the other worm or virus types.

The rate of failed connection requests from a host or failure rate can be measured by monitoring the failure replies that are sent to the host. The failure rate measured for a normal host is likely to be low. For most internet applications (www, telnet, ftp, etc.), a user normally types domain names instead of raw IP addresses to identify the servers. Domain names are resolved by Domain Name System (DNS) for IP addresses. If DNS cannot find the address of a given name, the application will not issue a connection request. Hence, mistyping or stale web links do not result in failed connection requests. An ICMP host-unreachable packet is returned only when the server is offline or the DNS record is stale, which are both uncommon for popular or regularly-maintained sites (e.g. Yahoo, Google, E-bay, CNN, universities, governments, enterprises, etc.) that attract most of internet traffic. Moreover, a frequent user typically has a list of favourite sites (servers) to which most connections are made. Since those sites are known to work most of the time, the failure rate for such a user is likely to be low. If a connection fails due to network congestion, it does not affect the measurement of the failure rate because no ICMP host-unreachable or RESET packet is returned. On the other hand, the failure rate measured for a worm-infected host is likely to be high. Unlike normal traffic, most connection requests initiated by a worm fail because the destination addresses are randomly picked, thus including those that are not in use or not listening on

Figure 1 The operating paradigm



the port that the worm targets at. Consequently, the failure rate can be conceived as a very good indicator of the scanning rate. Let be the address space size and N the number of hosts that listen on the attacked port(s). With N , the relation between the scanning rate and the failure rate of a worm is:

$$1 \quad \frac{N}{} \tag{1}$$

Hence, measuring the failure rate of a worm gives a good idea about its scanning rate. Given the aggressive behaviour of a worm-infected host, its failure rate is likely to be high, which sets it apart from the normal hosts. More importantly, an approach that restricts the failure rate will restrict the scanning rate, which slows down the worm propagation. We also try to identify some evidence of destination-source infection patterns that can be useful to dynamically determine filtering rules. Simply stated, after a vulnerable host is infected by a worm on a port p (i.e. the host is the destination of an early worm attack), the infected host will send out scans to other hosts targeting at the same port p in a short time (i.e. the infected host is a source of new worm attack). For example, an infected host by Slammer worm on port 1434 also sends out scans to port 1434 of other hosts. Thus, we keep a sliding window of previous network traffic (source and destination addresses of SYN and UDP traffic). Two general items are tracked: for each port witnessed in this traffic, we record the address of the inside destination host and the scanning source from the monitored network. The addresses can be IP or MAC in order to defeat worms that use spoofed IP addresses. If a source scan originates from a host that previously received a scan on an identical port, we increment a counter. If this counter passes a threshold calculated for the network as a result of a training process, we reasonably guess the presence of an active worm infection where the above port is used for worm propagation.

A facility that is able to identify anomalous traffic can be very useful to set up defences, in forms of filters/access-controls and traffic handling policies such as blocking or rate-limiting. Setting up a filter requires detailed knowledge of the scan traffic. Without such knowledge, our active reaction mechanism like rate limiting or blocking can partially protect the network at the earliest stage of the infection, prior of triggering self-repair actions on individual host through software upgrade/patching. Each detector monitors the outgoing flow and outgoing failure rate from traffic flow counters and replies sent to the customer network that the edge router/detector connects to. It identifies the potential offending hosts, the involved ports and measures their connection and failure rates. If any of the measured worm activity indicators of a host exceeds its preconfigured alarm threshold, the active detectors randomly drop a minimum number of connection requests from that host in order to keep the exceeding rate under the threshold. Both a temporal rate-limit algorithm and a spatial rate-limit algorithm can be used to constrain any worm activity to a low level over the longer term, while accommodating temporary aggressive behaviour of normal hosts. Any worm or virus will propagate more slowly under both the temporal rate-limit algorithm and the spatial rate-limit algorithm application. Administrators will have some more time to study the hostile traffic, identifying the infection behaviour and the so-called worm signatures or identification patterns that allow more selective content-based filtering policies to be applied. Substantial research effort has been performed in techniques that explore ways in which local systems and domains may delay worm propagation through the limiting of resources that aggressive worms are known to consume at high rates. Williamson (2002) suggests that throttling the volume of outbound connections that a host is allowed to initiate to new machines can produce a significant reduction in the infection rate, without significantly hindering normal communications. Staniford (2003) refines the outbound connection-throttling concept and provides extensive assessment of its behaviour, while moving the throttling mechanism from the individual host to the domain gateway. Gualtieri and Mosse (2003) propose to dynamically calculate outbound connection rate limits on a perprocess basis, through the observation of connection rates across the total population of processes or from a preselected group of known benevolent processes. Ganger et al. (2002) suggest that the analysis of network connections not facilitated through DNS lookups provide a relevant signature for identifying potential worm traffic and host-layer filters can be directed to such traffic with greater aggressiveness. Wong et al. (2004) explores the application of connection rate limiting to backbone routers, suggesting that the throttling of IP-to-IP connection at the edge offers propagation reduction equivalent to all hosts implementing rate throttling, while offering significant deployment advantages. All these techniques may be applied successfully in our reaction system, but at our very early stage of implementation in which both the detectors and the update agents are far from complete we used a very simple rate limitation facility based on the IPFW package implementation in the FreeBSD operating system, which implements a simple and effective traffic shaping scheme that can be used in our prototype as the rate limiter. The prototype routing capabilities are based on the standard Zebra implementation under FreeBSD and all the implemented features are really first-stage proof of concept exercises.

In our system, each active or passive sensor informs both instantaneously and on a periodic base the other detectors about the observed scanning activity, the potential offenders and the involved services/ports. A continuous, steady increase in the gross scanning activity raises the flag of a possible worm attack. The worm propagation can be further slowed or even stopped by blocking the hosts or ports with persistently high rates. In more detail, the edge routers with the role of active detectors can be configured to block out the addresses or the traffic flow related to specific ports whose indicators exceed for a time t the above fixed threshold values, where t is a system 'tolerance time' parameter selected so that if the worm-infected hosts perform high-speed

scanning, they will be blocked out after a time *t* of activity. Hence if *t* is assigned a sufficiently low (and carefully chosen) value, the worm propagation may be stopped before an epidemic materialises. The *t* parameter can also be made dynamic: once the threat of a worm is strongly confirmed, the edge routers/detectors may decide to reduce *t*, which increases the chance of fully stopping the worm.

3.2 Information sharing and communication

A second major direction has been toward the design of cooperative information sharing and reaction facility, by using the proper models, to help recognise the emergence of a propagating worm or virus and then take coordinated action before it can saturate the network. This component can be viewed as the overall, cooperative response of the natural immune system cells to pathogens after their detection by lymphocyte cells. Thus, when the immune system detects a pathogen (that in our case will be a hostile traffic activity affecting some hosts and ports), both if it is already known or it has not encountered before, it undergoes a primary response, during which it 'infers' the structure of the specific pathogen, that is, it evolves a set of lymphocytes (in our case determines filtering policies on the router/firewalls nodes) with high affinity for that pathogen, through a process called affinity maturation. In our networked computer world, the affinity maturation process can be associated to the activity of automatically generating service/ports or host-based filtering or rate limiting rules, from the observation of the current traffic flows and evaluating with properly-crafted heuristics or inference rules (constituting the immune knowledge base) the shift from an acceptable baseline. This process will be driven independently from each 'lymphocyte' node with all the nodes cooperating and exchanging information through the communication facility (knowledge spreading). A detector node communicates with agents within its local networking domain by using a classic multicast facility, sending its action triggers to a multicast group known to any participating agent. This implies a very small communication overhead not affecting the detector activity. Local communication between the detectors is based on the same mechanism, using, of course, a different group address. Inter-domain communication, which is needed only between detector nodes, is generally more critical. It needs to be implemented according to the same simple and scalable paradigm, a multicast routing facility available on the active detector nodes, which will be the multicast routers participating in the information exchange. Thus, all the activity messages from active or passive sensors/detectors toward agents are sent to the locally scoped multicast address 224.0.0.111 that all the agent facilities installed on the network hosts listen to. Information update messages between the detectors are sent to the multicast group 224.0.1.111, both periodically and on event basis. Messages are sent without any reliability enforcement strategy by using the UDP service on port 61111. There are two type of messages, that can be differenced from the message type and structure: the *activity message* (type 0 02), typically sent to the agents to trigger an update, a restart or a shutdown, according to the value inserted in the *Action* field and the *information update* (type 0 01) messages that can be used to transport information about one or more traffic flows and related filtering control and anomaly detection status (Figure 2).

Figure 2 Information and activity communication messages

Message Type=0x01(1 octets)
Number of flow entries (1 octet)
First Flow entry (21 octets)
…
N-th Flow entry (21 octets)

Message Type=0x02 (1 octets)
Action (1 octet)

The Traffic Flow entry, describing the information to be transported in inter-detector messages, allows sensors to express any detected conditions in terms of traffic discriminator list (like an ACL), traffic rate, detected alert conditions and suggested action. It supports multiple protocol filtering, based on a combination of source/destination IP prefix with exact, range or wildcard based matching and source/destination TCP or UDP port. A Traffic Flow entry is defined below (Figure 3):

Figure 3 The traffic flow entry

Protocol (1 octet)
Traffic rate (4 octets)
Action (1 octet)
Alert flags (1 octet)
Source Prefix (4 octets)
Source Prefix Length (1 octet)
Destination Prefix (4 octets)
Destination Prefix Length (1 octet)
Source Port (2 octets)
Destination Port (2 octets)

The *Traffic rate* field is a 4-octet value that specifies the measured 5 min sustained traffic rate (in Kbits) of the traffic flow.

The *Alert flags* field is an 8-bit mask reporting the status of each implemented worm activity indicator (0 threshold not exceeded, 1 exceeded) for the specified flow. At the moment only the first two bits (outgoing flow and outgoing connection failure rate, respectively) are meaningful.

The *Action* field specifies whether this traffic flow entry should be filtered through rate limiting (value 0 01) or definitely blocked (value 0 10). A zero action value specifies the suggestion to remove any filter for the specified flow.

The *Protocol* field is an octet (value 0 255) specifies the protocol number to which the specific traffic control rule is referred.

The *Source* and *Destination Prefix Length* fields (1 octet) indicate respectively the length in bits of the portion of the Source and destination address prefixes that must be matched in the traffic control. This allows wildcard-based prefix matching. A length of zero indicates a prefix that matches all addresses (with prefix itself of zero octets).

The *Source* and *Destination Prefix* fields (4 octets each) contain respectively the source and destination IP prefixes eventually used in the filtering entry. If the corresponding Prefix Length octet is zero, as defined above for 'match any' source or destination prefix entries, this field is a do not care.

The *Source* and *Destination Port* field (2 octets, unsigned) indicate the source and destination TCP or UDP port referred in the traffic control filter. The field is considered as un-specified with a 'don't care' value for any protocol other than TCP (value 6), UDP (value 17) or ICMP (value 1). When the *Protocol* value is ICMP the source and destination port fields contain respectively the ICMP type and code values (Table 1).

Table 1 Variable meaning of the port fields

Protocol	*Source port*	*Destination port*
0 (IP)	NA	NA
1 (ICMP)	ICMP Type	ICMP code
6 (TCP)	TCP source port	TCP destination port
17 (UDP)	UDP source port	UDP destination port

3.3 The immunisation: update agents

The update agents installed on the networked hosts act as the natural capacity of an individual cell to self-repair itself and in most cases, when a patch is available and correctly applied, become immune. Agents are triggered from active and passive detectors/sensors through the multicast communication facility to drive automatic updates, shutdown or reboots according to the action message received. Clearly, this kind of immunisation, obtained by patching, can protect a system from the effects of a known worm or virus for which a proper patch is already available through the network, but in a large network it is essentially impossible to be sure that all the nodes are properly immunised. This raises the question of what the effect might be of immunisation that is only effective on certain nodes. Dealing with this question might also allow insight into the conscious use of selective immunisation where the task of immunising a complete network is infeasible.

We also investigated the idea that the update agents could be configured to trigger the update in a preventive fashion, for instance every time that a fresh (wired or wireless) network access is gained. However, for this strategy to be effective, every other network activity should be quarantined while the update process is running and this requires too tight a control over the protocol stack components. The ideal solution would be the substitution of the protocol stack itself with a purposely-crafted one, along the lines of (Williamson, 2002). However, this quarantine time can be perceived as a annoyance by users, who can be thus motivated to look for ways to bypass the agent, with destructive effects.

We realised a very simple and incomplete prototype of update agents running on MS Windows XP and RedHat Linux hosts that can be used to trigger, under the activity message stimulus, automatic update or shutdown of the involved host.

4 Modelling and proof of concept

We have used mathematical models of worm transmission to estimate the infectiousness of a worm from the outgoing flows, to assess the likelihood of an outbreak when a case is introduced into a susceptible population and to draw preliminary conclusions about the impact of our control measures. Accurately modelling and analysing the effects that large-scale worm infestations have on the network infrastructure is challenging since, in addition to issues related to heterogeneity and change in the internet, a worm that infects tens or hundreds of thousands of machines gives rise to an inherently large-scale phenomenon and requires the model to be of appropriate scale to correctly model the propagation dynamics. By analogy between human and computer infection dynamics, the well-understood mathematical laws regulating the spread of infectious pathogens can be straightforwardly adapted to describe computer worms. The propagation dynamics of worms spreading by uniform random scanning, such as Code Red and Slammer, lend themselves well to a coarse form of modelling based on epidemic models. This drastically reduces resource requirements compared to a pure packet-level model of the whole system. On the other hand, epidemic models alone include no information about the underlying network and its possible effects on the propagation, for example, bandwidth constraints on scans as observed during propagation or router failures observed due to worm traffic. Moreover, additional details are required to create meaningful, realistic test data for a significant analysis. Consequently, we have chosen to combine a coarse-level model of the worm propagation and scan traffic with detailed models of selected parts of the network. Thus, the worm propagation through the internet can be modelled 'macroscopically' through an epidemic model, while the agent coordination behaviour can be described in detail through the classic network model, based on detailed abstractions of routers and protocols. The two model levels use different time advancement mechanisms: the network model is event-oriented and the epidemic model is time-stepped. However, this does not present a problem since a single recurrent event timer can be used to advance the epidemic model. The available epidemic infection models that can be used to describe the spread of the worm as it infects hosts in the internet are based either on a discrete stochastic (time-stepped) approach or on a deterministic one (using differential equations). For 'sufficiently large' populations, such as the global internet, it is common to

approximate the stochastic model by the better continuous-state, continuous-time deterministic model, capturing the mean behaviour of the observed phenomenon The notation used in such model is reported in the following table (Table 2).

Table 2 Notations used in this paper

Notation	*Definition*
	Size of the address space
N	Total number of vulnerable hosts
	Average worm scan rate
	Average worm connection failure rate
$S(t)$	Number of Susceptible hosts
$I_U(t)$	Number of Infected hosts that are undetected
$I_D(t)$	Number of Infected hosts that detected
$P(t)$	Number of Patched hosts
	Pairwise infection rate
	Average patching rate (baseline)
	Average patching rate multiplier
	Scanning detection delay

The SIR model (Daley and Gani, 1999) is a deterministic, compartmental model assuming that individuals in a populations of N hosts fall into three classes, *susceptible individuals* $S(t)$ who are vulnerable but not yet infected, *infected individuals* $I(t)$ who can pass immediately the pathogen to others and *removed individuals* $R(t)$ who were infected but have recovered (the recovery or removal rate being). In the SIR model, the allowed state transitions are *Susceptible Infected Removed.* Another assumption is that infections occur randomly and homogeneously in proportion to the density of susceptible and infected individuals and the *transmission coefficient* , that is, *the pairwise rate of infection.* Consider a worm that uniformly scans and address space of addresses. At time t, the number of infected hosts is $I(t)$. Each of these infected hosts incessantly scans for potential victims. Let be the average scan rate. The probability that a random scan hits a host is $N/$. But since only $S(t)$ hosts out of N are vulnerable, the pairwise rate of infection is $I(t)$ $(N/$ $)(S(t)/N)/(I(t)$ $S(t))$ / . The epidemic process is described by the following differential equations:

$$\begin{aligned} \dot{S} &\quad IS \\ \dot{I} &\quad IS \quad R \\ \dot{R} &\quad I \end{aligned} \tag{2}$$

The basic SIR model can be extended in many ways (Anderson and May, 1991), to take into consideration, for example, population variation by births or deaths, infection latency, quarantining and immunisation. While therapy and vaccination hardly depend on the same chemical agent, the act of applying a software patch will remove both susceptible and infectious computers from the worm grasp. So we renamed the Removed compartment as Patched and distributed into it individuals taken from the Susceptible and Infected compartments. In their model, Zou et al. (2002) make a distinction between cure and prevention and use two different equations to describe the two processes. This is correct on the time scale where patching is done by human intervention, because people will have a stronger motivation to update systems known to be under a worm's control. However, since we are concentrating on the initial phases of an infection and we envision a very limited human intervention, we have applied the same equation to both the categories. We also made the simplifying assumption that patching is evenly spread across the population. Next, we consider the effect of early detection. As with human epidemics, infectious or exposed individuals may be detected or go undetected. We split the Infected compartment in two parts and place infected hosts in the Infected-Undetected compartment I_U if they have not yet been detected and in the Infected-Detected compartment I_D, if they have been detected. We assume that detection is made with a single mechanism throughout the system, but there can be local variations in the detection interval. Let denote the average detection time. The mean rate for progression from I_U to I_D is then $1/$. When scanning is detected, the effect of rate limiting can be modelled as follows. The rate-limiting factor, , is applied as a multiplier to the scan rate and thus to the pairwise rate of infection . Thus, the equation for the evolution of Infected individuals remains of the same form, provided that the 'effective' infected population I_U I_D is considered in place of I. We assume (as it is all too often the case) that the worm exploits a vulnerability that has already been disclosed to the community. Appropriate patches have thus been developed and are available at the time of infection. Patching takes place (without intervention by the agents) at a linear rate , coherently with experimental data gathered in the early-middle stages of infection. When agents trigger updates and patching, the slope of the patching curve gets magnified by a factor .

$$\dot{P} \quad \begin{matrix} t \\ t \end{matrix} \tag{3}$$

Since the agents communicate between them, this activity takes effect in the whole system nearly at the same time, that is, as soon as the first infection is detected, that is at time (scanning starts at t 0). We operate under the hypothesis that the communication latency (the time needed before an agent receives the relevant information) and the actuation latency (the time needed before an agent's commands are actually executed) are negligible as compared with the detection time. Combining all of the above, we write down the equations for our model:

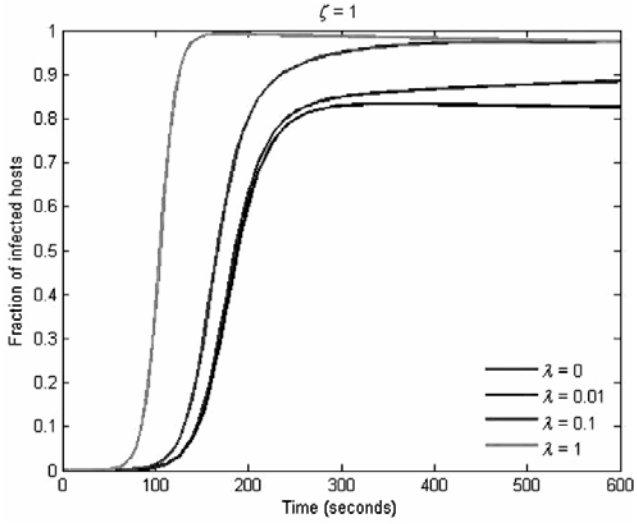
$$\begin{aligned} \dot{S} &\quad \dot{I}_U \quad \dot{I}_D \quad \dot{P} \\ \dot{I}_U &\quad I_U \quad I_D \; S \quad \frac{I_U}{} \quad \frac{I_U}{N \quad P}\dot{P} \\ \dot{I}_D &\quad \frac{I_U}{} \quad \frac{I_D}{N \quad P}\dot{P} \\ \dot{P} &\quad \begin{matrix} t \\ t \end{matrix} \end{aligned} \tag{4}$$

Numerical solution, experiments and sensitivity analysis with respect to some parameters is carried on in the forthcoming section.

4.1 Analysis and results

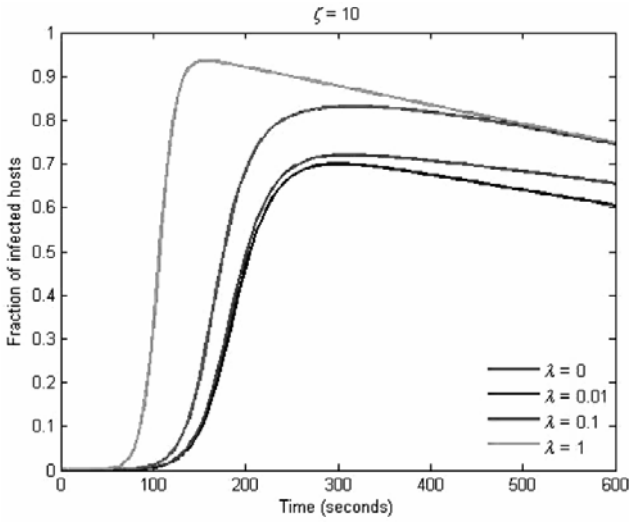
We chose to analyse Slammer in our model, since its average scanning rate of 4000 scans/sec makes it a prototype of a very aggressive worm. The scanning speed of Slammer was mainly due to the code being contained in a single UDP packet, so Slammer could broadcast scans without having to wait for any response. A TCP-based worm, on the other hand, would have to wait at least a packet round-trip time for successful connections to be established. Worse yet, it must wait for unsuccessful connection attempts to time-out. It thus can be expected to have a significantly lower average scan rate. We used a scan rate of 4000 scans/sec, a baseline removal rate 5.2 removals/sec, a total population of N 120,000 hosts, a detection time 20 sec and $I(0)$ 1 initially infected host. Figure 4 shows $I(t)$ $I_U(t)$ $I_D(t)$ for different rate-limiting factors, when no stimulated patching is in place. It should be noted that the choice of rate-limiting factors is prudent: even a rate-limit of 0.01 brings a 100 Mbit/sec Ethernet connection at the speed of a DSL line.

Figure 4 Sensitivity to with no stimulated patching



We observe that rate limiting alone cannot stop a worm as aggressive as Slammer. Even a small rate-limiting factor, though, can delay the infection for some valuable time before communication links become saturated and exchange of information becomes difficult. In addition, it was an important design requirement that the system should provide some benefit even in a limited version, without stimulated patching, because it may be argued that users may resist the installation of an agent on their machines. This would probably be an issue in some environments, but we believe that user resistance would not be higher than what can be expected of protocol stack implementation that limit the number of hosts contacted. Figure 5 shows the same data as Figure 4, but with stimulated patching at 10, that is, updates are increased by a factor of ten.

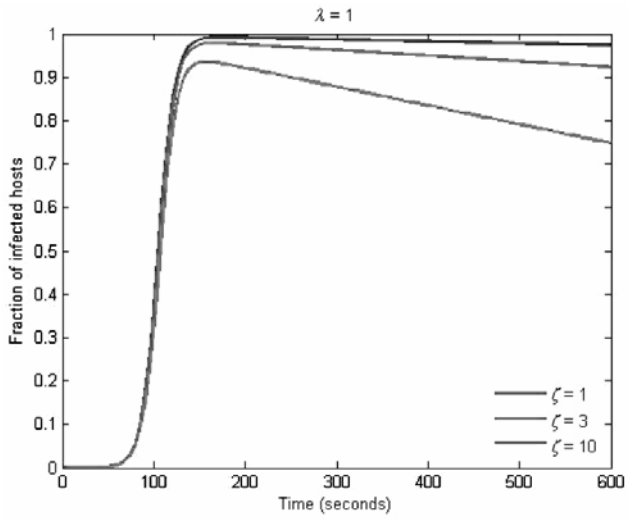
Figure 5 Sensitivity to with stimulated patching



Here, the rate-limiting factor influences the maximum percentage of hosts affected, as well as the time at which this maximum is reached. Again, a rate-limiting factor of 0.01 achieves good performance. This can be favourably weighed in consideration of the confined effects that false positives may have. A machine legitimately doing traffic that resembles scanning would not undergo a complete block, only a slowdown.

Figure 6 above illustrates $I(t)$ $I_U(t)$ $I_D(t)$ for different patch rate increase factors, with no rate limiting in place. We see that a factor of 10 is needed for the infection to be contained somewhat. Figure 7 shows again $I(t)$ for different patch rate increase factors, but with rate limiting fixed at 0.01.

Figure 6 Sensitivity to with no rate limiting

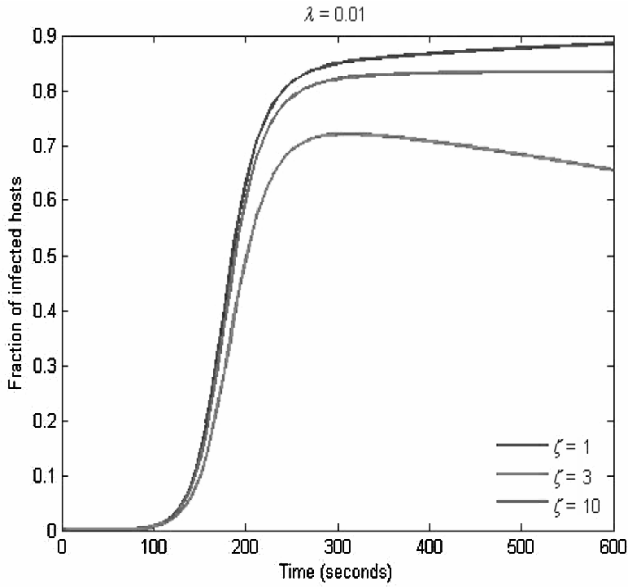


The effects of stimulated updating are amplified by rate limiting. At 10 the infection is restrained to 70% of the total population and recovery time is acceptable.

5 Related work

Most research on the internet worms concentrates on studying propagation models (Liljenstam et al., 2002; Staniford et al., 2002). In particular, Zou et al. (2002) proposed a modified 'two-factor' epidemic model that accurately described the CodeRed worm propagation and can be considered as a milestone in representing worm infection processes. They considered the infection factor as a variable, scaling it down as more hosts are infected. They described the recovery and the immunisation processes with two separate compartments whose evolution is regulated by distinct differential equations. However, the problem of conceiving effective automatic defence mechanisms against worms is still an open problem. Moore et al. (2003) have recently studied the effectiveness of worm containment technologies (such as address blacklisting and content filtering) and concluded that such systems must react in a matter of minutes and interdict nearly all the internet paths in order to be successful. Williamson proposed to modify the network stack so that the rate of connection requests to distinct destinations is bounded (Williamson, 2002) thus limiting in advance the infection virulence. The main problem is that this approach becomes effective only after the majority of all the internet hosts are upgraded with the new network stack, that is, in the real world, almost unpractical. Recent work has focused on automated distributed mechanisms for containment (Moore et al., 2003) and disinfection (Nojiri et al., 2003) that may be able to spread fast enough to mitigate the effect of the virus. Some believe that there is reason for guarded optimism. Studies have shown that fairly low-levels of immunisation (Wang et al., 2000) or low-level responses (Kephart and White, 1991) can be enough to contain the virus or significantly slow the spread of the virus. The automated response mechanisms may be able to detect, filter and disinfect or immunise quickly enough to prevent runaway infection and allow human intervention.

Figure 7 Sensitivity to with drastic rate limiting



6 Conclusion

As millions of users migrate between home, office, coffee shop and bookstore, they take with them not only their computer, but also electronic hitchhikers such as fast propagating worms and viruses they picked up in elsewhere, threatening the integrity of all the network environments they access in. This problem will only be exacerbated as wireless coverage expands and nomadic behaviour becomes more and more common. This is a fundamental security threat that must be effectively addressed. Accordingly, our work pursues the idea that only an automated reaction system approach can present an effective defence against fast (or flash) viruses and worms in modern heterogeneous network environments. We started from the consideration that many recent efforts, focused on network and host protection, typically based on a conservative prevention approach, produced solutions that either presented considerable impact on performance or have been demonstrated inadequate or extremely difficult to implement. Our first-reaction adaptive and cooperative approach is inspired by the natural immune system and tries to automatically solve the infection problems at the single host level by patching, when necessary all the vulnerable software, thus circumventing the need for human intervention in time-critical infection containment. Since the proposed system can identify anomalous traffic shortly after the scanning activity starts, this information can be also used to quickly limit the scan rate, for example, by applying suitable access lists aimed at rate-limiting the IP traffic coming from the offending machine. While filters are much more precise in limiting only the scans, leaving legitimate traffic undisturbed, setting them up requires detailed knowledge of the scan traffic. Without such knowledge, rate limiting the suspect host can however protect the network at the earliest stage of the infection, although in some cases false positives may occur. This reaction system, like the natural immune system, is massively parallel and its functioning is truly distributed. Individual components, like the different kinds of human cells, are disposable and unreliable, yet the system as a whole is robust and effective like the results of our analysis assess. Novel or already known infections can be detected and eliminated quickly, using a variety of adaptive mechanisms, which can be further enriched. The system is autonomous, controlling its own behaviour both at the detector and effector levels.

References

Albanese, D., Wiacek, M., Salter, C. and Six, J. (2004) *The Case for Using Layered Defenses to Stop Worms*, National Security Agancy, June.

Anderson, R.M. and May, R.M. (1991) *Infectious Diseases of Humans: Dynamics and Control*, Oxford: Oxford University Press.

Arbaugh, W.A., Fithen, W.L. and McHugh, J. (2000) 'Windows of vulnerability: a case study analysis', *IEEE Computer*, Vol. 33, No. 12, pp.52–59.

Cohen, F. (1987) 'Computer viruses – theory and experiments', *Computers and Security*, Vol. 6, pp.22–35.

Daley, D.J. and Gani, J. (1999) *Epidemic Modeling: An Introduction*, Cambridge: Cambridge University Press.

Ganger, G., Economou, G. and Bielski, S. (2002) 'Self-securing network interfaces: what, why, and how', *Carnegie Mellon University Technical Report*, CMU-CS-02-144, August.

Gualtieri, M. and Mosse, D. (2003) *Limiting Worms via QoS Degradation*, University of Pittsburgh.

Kephart, J. and White, S. (1991) 'Directed-graph epidemiological models of computer viruses', *Proceedings of the IEEE Symposium on Security and Privacy*, pp.343–361.

Liljenstam, M., Yuan, Y., Premore, B. and Nicol, D. (2002) 'A mixed abstraction level simulation model of large-scale internet worm infestations', *Proceedings of 10th IEEE/ACM Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, October.

Moore, D., Shannon, C., Voelker, G. and Savage, S. (2003) 'Internet quarantine: requirements for containing self-propagating code', *Proceedings of 22nd Annual Joint Conference of IEEE Computer and Communication Societies (INFOCOM 2003)*, April.

Nachenberg, C. (2000) 'The evolving virus threat', *Proceedings of 23rd NISSC*, Baltimore, Maryland.

Nazario, J., Anderson, J., Walsh, R. and Connelly, C. (2001) 'Future of internet worms', *Crimelabs Research*, July.

Nojiri, D., Rowe, J. and Levitt, K. (2003) 'Cooperative response strategies for large scale attack mitigation', *Proceedings of the Third DARPA Information Survivability Conference and Exposition*, April.

Rescorla, E. (2003) 'Security holes... Who cares?' *Proceedings of USENIX Security Symposium*, August.

Staniford, S. (2003) 'Containment of scanning worms in enterprise networks', *Journal of Computer Security*, to appear.

Staniford, S., Paxson, V. and Weaver, N. (2002) 'How to own the internet in your spare time', *Proceedings of 11th USENIX Security Symposium*, San Francisco, August.

Wang, C., Knight, J.C. and Elder, M.C. (2000) 'On computer viral infection and the effect of immunization', *Proceedings of the 16th Annual Computer Security Applications Conference*, pp.246–256.

Williamson, M. (2002) 'Throttling viruses: restricting propagation to defeat malicious mobile code', *Proceedings of Annual Computer Security Application Conference (ACSAC'02)*, December.

Wong, C., Wang, C., Song, D., Bielski, S. and Granger, G.R. (2004) 'Dynamic quarantine of internet worms', *Proceedings of the International Conference on Dependable Systems and Networks (DSN-2004)*, Florence, Italy, June.

Zou, C.C., Gong, W. and Towsley, D. (2002) 'Code red worm propagation modeling and analysis', *Proceedings of the Ninth ACM Conference on Computer and Communications Security(CCS)*, November, pp.138–147.