ISSN:2231-0711
www.ijcset.net

Modeling and Automated Containment of Worms

P.Daniel Ratna Raju
HOD CSE Dept
Priyadarshini Institute of Technology & Science
ratnaraju.daniel@gmail.com

G.Neelima
Asst. Prof CSE Dept
Acharya Nagarjuna University,
neelima_raju2003@yahoo.co.in

Abstract -While much recent research concentrates on propagation models, the defense 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 immunization 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 defense against more aggressive worms.

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

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 pact 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. Most computer security issues can be viewed as the problem of distinguishing *self* (legitimate traffic, authorized 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 everyday. 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 The nodes in our system cooperate in detecting and informing each other of ongoing attacks and of the actions necessary to the defense, 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 defense 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 (Navarro 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 loop holes and install themselves on machines where they are unwanted. They often try to grow without bound, attempting to infect every machine accessible to them. 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 (Navarro 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. On the other side, the most obvious defense 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 vulnerability. Customers can then install the patch and prevent attacks that exploit the vulnerability. Software patching has not been effective as a first-line defense 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 publicized Code Red, SQL Slammer, Blaster and Sesser 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 recognized 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.

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 analyzing some properly chosen health parameters, directly reflecting the network behavior 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 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 favorite 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 connection fails due to network congestion, it does not 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 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 _N (1)

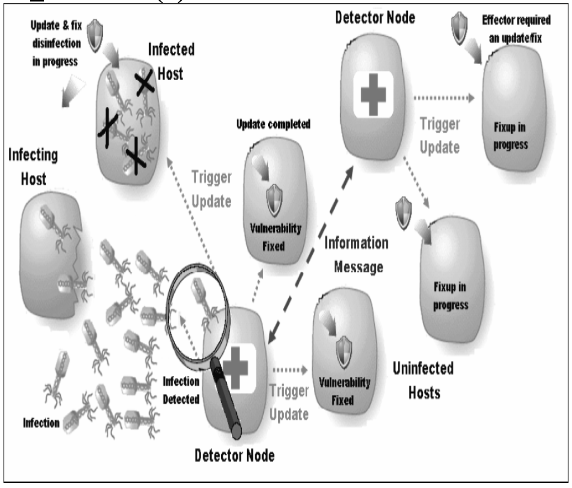


Figure 1 The operating paradigm

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 materializes. 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 recognize the emergence of a propagating worm or virus and then take coordinated action before it can saturate the network. (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).

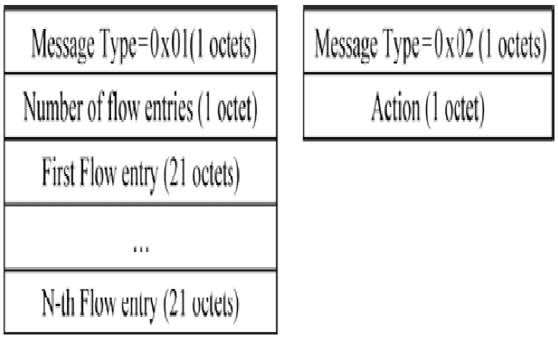
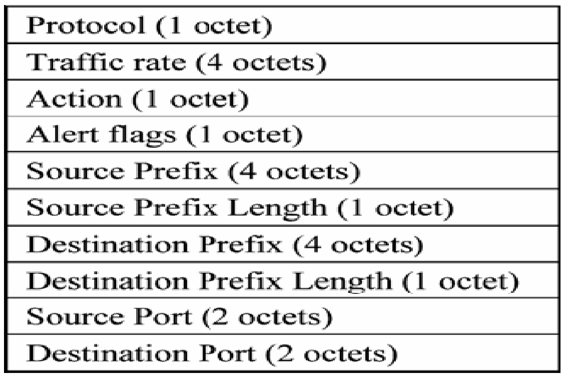

Message Type=0x01(1 octets)
Number of flow entries (1 octet)
First Flow entry (21 octets)
...
N-th Flow entry (21 octets)
Message Type=0x02 (1 octets)
Action (1 octet)

Figure 2 Information and activity communication messages

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):



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)

Figure 3 The traffic flow entry

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

4 Modeling and Proof of Concept

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 behavior of the observed phenomenon.

4.1 Analysis and results

We chose to analyze 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.

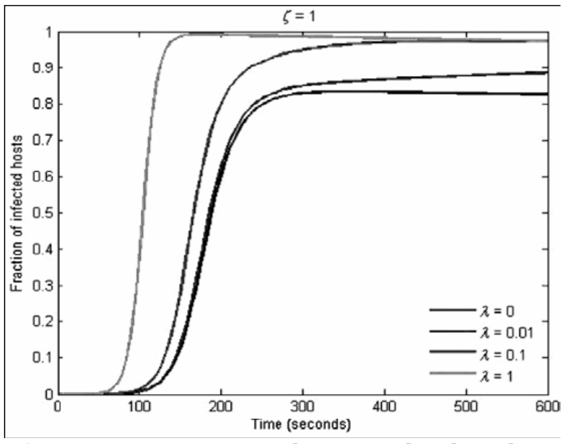


Figure 4 Sensitivity to with no stimulated patching

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.

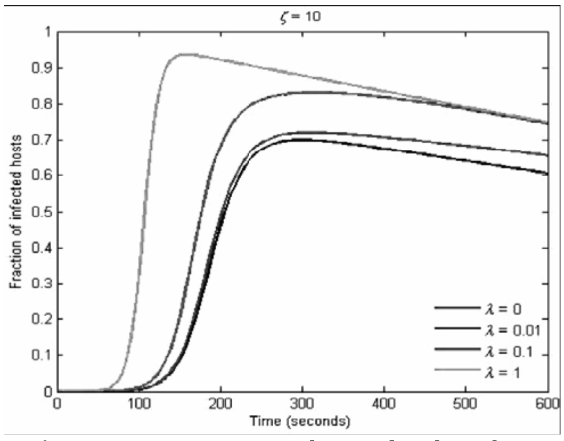


Figure 5 Sensitivity to with stimulated patching

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.

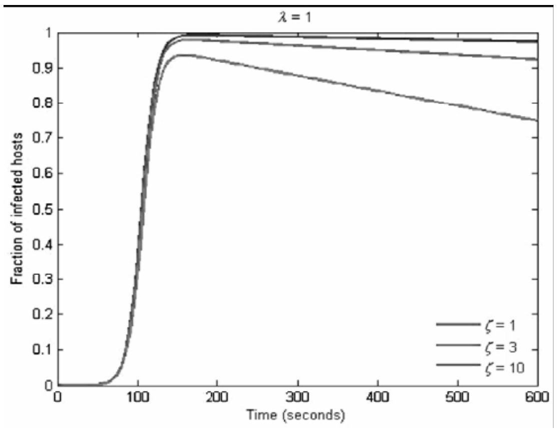


Figure 6 Sensitivity to with no rate limiting

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; Stanford et al., 2002). In particular, Zou et al. (2002) proposed a modified ‘two-factor’ epidemic model that accurately described the Code Red 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 immunization processes with two separate compartments whose evolution is regulated by distinct differential equations. However, the problem of conceiving effective automatic defense 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 fo 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 immunization (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 immunize quickly enough to prevent runaway infection and allow human intervention.

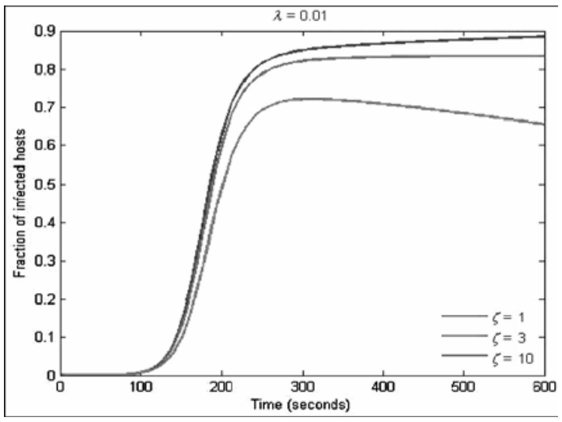


Figure 7 Sensitivity to with drastic rate limiting

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 behavior 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 defense 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 behavior both at the detector and effecter 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',*IEEEComputer*, 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 responsestrategies 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.