Evaluation of collaborative worm containment on the DETER testbed

L. Li, P. Liu, Y.C. Jhi, G. Kesidis
College of Information Sciences & Technology
Computer Science and Engineering and Electrical Engineering Depts
Pennsylvania State University, University Park, PA 16802
lli,pliu@ist.psu.edu,jhi@cse.psu.edu,kesidis@engr.psu.edu

Abstract

The advantage of collaborative containment over independent block or address blacklisting on worm defense has been advocated in previous worm studies. In this work, we will evaluate two collaborative worm containment proposals and present some of the results of our DETER emulation experiments. In the first one, proactive worm containment (PWC), security agents block all suspicious hosts on the network on receiving alerts of a worm and run "relaxation analysis" on those blocked hosts afterwards. Emulation experiments will evaluate PWC's ability to stop the propagation of fast local worms and to reduce scan traffic of fast global scanning worms. The second proposal, which detects and contains a scanning worm based on the concept of dark port, focuses on stealthy worms that target only specific local networks or enterprise networks. Emulation experiments run on the DETER testbed demonstrate the efficiency of local scanning worms and their elevated threat to enterprise networks. The effectiveness of a collaborative containment strategy based on dark port detection is evaluated using DETER emulation and compared with that of individual address blacklisting.

1. Introduction

While most conventional worm containment proposals depend much on novel detection technique or containment strategy, we believe that a collaborative approach in which end-hosts or sub-networks share security alerts and act proactively could be superior, especially in those contexts where the priority of network security is put higher than those of temporary availability loss. Works in [18] and [3] have demonstrated the advantage of group defense over independent containment. In this study, we will run emulation experiments to evaluate the effectiveness of two worm containment schemes we proposed that champion a collaborative approach. The first one, Proactive Worm Containment, or PWC [8], is based on two key concepts: early blocking and post-hoc relaxation. In abstract terms, the PWC system contains a suspicious host after an early-but-immature worm alert is raised by a worm agent running on the host. A potentially false alert is to be repaired by the subsequent relaxation phase. To be more proactive, the worm alert is propagated to all other worm agents in the network once it is raised. Upon receiving a propagated worm alert, an agent proactively blocks its host and starts the relaxation phase. Early containment reduces the number of outbound scans, which otherwise could infect another victim in the local network or somewhere in the Internet.

The second proposal, dark port detection and containment, takes aim at a new kind of worms that target specifically local or enterprise networks. For the defense against this new threat, we have proposed a new dark port detection scheme that can effectively monitor all suspicious scans within an enterprise network as soon as those scans arrive at the local routers that are supposed to deliver those packets to their intended final targets. A selective or full block action then can be taken to block dark-port-scan originating hosts, cells, or the full network on specific port numbers. We will run evaluation experiments to compare the new dark port proposal with the individual blacklisting containment with various detection latencies.

This paper is organized as follows. Section 2 reviews the related work on the worm detection, worm containment, and enterprise worms. The procedure of running a worm experiment on the DETER testbed and related tools are briefly discussed in section 3. In section 4, we introduce the PWC system and present evaluation results of PWC. In section 5.2, we show the threat of a new class of local-only scanning worms and introduce our new dark port detection framework. A cooperative containment strategy based on the new dark port detection is presented in section 6, together with results of its DETER evaluation. In the final section, we conclude the paper and list some future research topics.

2. Related work

Existing worm containment techniques can be roughly broken down into following classes: *Rate limiting* [25, 4] is to limit the sending rate of scan-like traffic at an infected host, which may introduce longer delays for normal traffic;

Signature-based worm filtering [21, 17, 15, 9, 22, 26] relies on worm signatures to prevent scans from entering/leaving a LAN/host, which may be evaded by a proper use of polymorphism [7]. PWC can effectively slow down or even stop the propagation of a fast scanning worm before a reliable worm signature is generated and applied. The idea of early response before having enough evidence for worm containment also appeared in the dynamic quarantine scheme in [27]. But unlike PWC, dynamic quarantine is not a collaborative approach and it cancels containment on a suspicious host after a period of time without any further analysis.

In many cases, worm signature generation is helped by a worm detector. Honeypot or honeynet [5] can detect suspicious worm activities by monitoring scans that are targeting at unused or dark network addresses. For detection and defense in local or enterprise networks, a double honeypot can be used to collect worm samples for signature generation [22]. In [6], an enhanced honeypot based detection, HoneyStat, combines network security alerts with computer OS alerts to improve the detection accuracy and reduce false positives.

Another type of worm detection scheme, failed scan detection, is also taking advantage of worms' blind scanning behavior. While the early version of failed scan detector used to rely on the returned ICMP packets from the destination host or network [16], most recent proposals detect failed scans by keeping logs of all connection requests on the sender site and singling out these requests that have not received response after a pre-defined period of time [20, 24]. Compared with these failed scan detection proposals, our dark port detection can detect a failed local scan much faster since the security agent at the destination site can report a failed scan right after its arrival and there is no need to wait for a connection to go time-out at the sending site.

Recently, network testbeds are becoming available and researchers began to use them to simulate and emulate worms and study their behavior in the Internet [23] or in enterprise networks [13]. The advantage of cooperative or group defense over independent firewalls has been evaluated by simulation experiments such as [18].

3. DETER worm emulation with virtualization

Network testbeds such as DETER provide a simulation and emulation platform with the highest level of flexibility and fidelity in term of hardware and network configurations, code compatibility, and network metrics. In [13], we reported our virtual node approach to leverage limited testbed resources to emulate worm propagation in a large enterprise network. The general method and steps of running a DETER emulation experiment are briefly reviewed here.

3.1. Setup an Experiment Using the ESVT Tool

The ESVT GUI tool provides an integrated environment to conduct an interactive worm or other network experiment on a testbed. It is a component based topology editor, NS2/TCL script generator, worm experiment designer, and a visualization tool of experimental results. At the first step, it can be used to draw the topology based on a prototype network. The toolbox of programs includes network components such as computer/host node, switch/router node, network/Internet interface, and link. Computer nodes can be defined as susceptible or non-susceptible. To request network resources from the testbed and apply network topology on the testbed, a NS (network simulator) style TCL script file needs to be submitted to the testbed control plane. The ESVT GUI has a TCL script generation function that can output and convert a network topology into a DETER TCL script to simplify this task.

3.2. Our Virtual Node Design

Employing a one-to-one emulation approach entails substantial resources that a normal testbed cannot support. In [13], we compared our virtual node design with other kinds of virtualization methods such as VMWare and Emulab VM and concluded that the performance of the virtual node design in realistic LAN simulation is comparable with the all-real-node scenario, while consuming much fewer resources than other virtualization approaches.

In our virtualization approach, one switched LAN in the real topology is emulated by one virtual-node application running on a testbed host. The virtual node program consists of a number of virtual end-systems, each representing one real host and having a unique virtual IP address. Inside the program, each virtual end-system is implemented by a thread which simultaneously executes a user-defined procedure (such as worm traffic generator) and a background traffic generator. The parameters of background traffic of each virtual node that can be configured include the maximum sending rate, the number of simultaneous sessions, packet inter-arrival distribution, and port number/protocol distribution.

We adopted a similar design for the timing of incoming worm scans from the Internet interface as reported in [13, 12]. The Internet interface node recreates and injects scanning traffic into the enterprise network under test. The speed and pattern of worm traffic were based on simulation results from our extended KMSim model [1, 10]. KMSim worm model, an extension of Kermack-McKendrick epidemic model, takes enterprise networks as the unit of analysis so that various distributions of worm susceptibles and network topology characteristics can be accounted for. Suppose that the total scan rate, $S(t)$, of a worm is obtained using the KMSim simulation, under the assumption of a uniform distribution, the scan-rate from the Internet directed at the enterprise under simulation could be approximated as $(A/2^{32})S(t)$, where A is the size of the address space of the enterprise network. The data source for the background traffic was mainly from the enterprise traffic report in [11].

4. PWC evaluation on DETER

PWC is a proactive worm containment solution for enterprises. Motivated by the observation that a worm uses a sustained outgoing packet rate, PWC gains infection awareness seconds before a signature or filter can be generated [8] and broadcast worm alerts to all worm agents. To overcome denial-of-service possibly caused by containment based on received worm alerts, PWC performs relaxation analysis that detects and releases contained-but-uninfected hosts. PWC has following features: *minimal denial-of-service*; *signature-free*; *lightweight*; and *evasion resilience*.

4.1. PWC Algorithm

Each host in an enterprise network protected by PWC runs a worm agent, or a PWC agent, that performs detection and suppression of worm scans released from its host. The PWC agents are coordinated by a PWC manager that has two roles: first, it distributes authenticated worm alerts reported by the PWC agents to all the PWC agents in the enterprise network; second, it is a certificate authority in authentication between each PWC agent and the PWC manager and vice versa. PWC can handle multiple simultaneous worm alerts raised by different worms in one contain/relax procedure.

Similarly to other worm detection techniques such as virus throttle [25], a PWC agent calculates the rate of most recent outbound connection attempts[1] sent to unique IP addresses in order to gain awareness of worm infection on its host. When the rate exceeds λ connections per second, it raises an alert and reports it to the PWC manager. The PWC manager then propagates the alert to the rest of PWC agents in the enterprise network. When a PWC agent receives an alert from the PWC manager, the PWC agent properly authenticates it before accepting it.

A PWC agent starts containing its host either on raising or on accepting an alert propagated from the PWC manager. When a PWC agent is containing its host, it buffers TCP SYN and UDP packets sent to new destination addresses, allowing traffic through existing TCP connections and packets to known destinations. On initiating containment, the PWC agent also starts *relaxation analysis* for limited duration τ seconds, to check if the outbound connection rate is sustained. If the relaxation analysis detects a sustained connection rate, the PWC agent silently discards buffered connection attempts and performs relaxation analysis again; otherwise, the containment is relaxed and buffered connection attempts are forwarded.

[1]*outbound connection attempts* include outgoing TCP SYN and UDP packets.

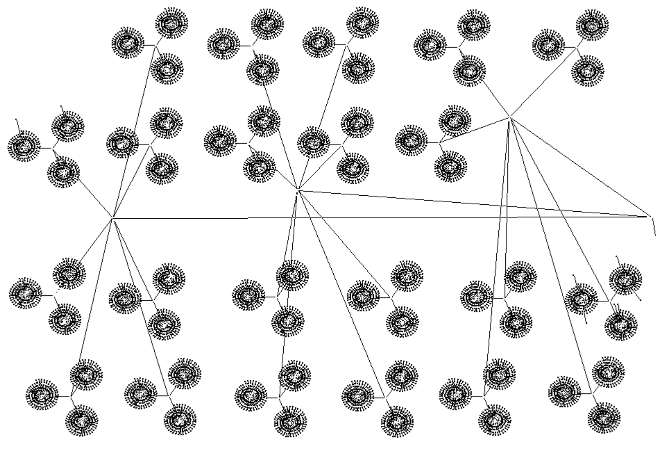
Figure 1. Enterprise network with 10000 nodes

To test the effect of PWC on containing fast scanning worms, we conducted two enterprise network emulation experiments on a network testbed, DETER [2]. The first experiment is to test whether PWC can effectively contain a fast local-scanning worm, while the second is to study how PWC limits high volume scanning traffic originated from suspicious hosts.

4.2. Experiment topology

To conduct a detailed worm propagation experiment using emulation and simulation method, the scale of the network has to be large enough to capture the interplay between the worm and defense, and the configuration of the network has to be typical of enterprise networks. For this purpose, we utilized our ESVT toolkit [14] to design a hypothetical enterprise network, which included one Internet stub-link, 22 internal routers, 66 switched LANs, 7 servers, and more than 10000 end-hosts. Of the hosts, 110 hosts (about 1%) are susceptible to the worm that we intentionally injected into the network. Thanks to the virtualization, emulation of this enterprise network on the DETER testbed only took 93 physical nodes.

4.3. Emulation results

The prototype for the PWC evaluation implements three components: worm detection based on unique destination addresses, worm alert broadcast, and proactive containment upon receiving a worm alert. The parameters of PWC for the emulation experiments are as follows: sending rate threshold 20 scans/second; vulnerability window 1 second. We implemented worm alert ‘broadcast’ by building a list of LAN

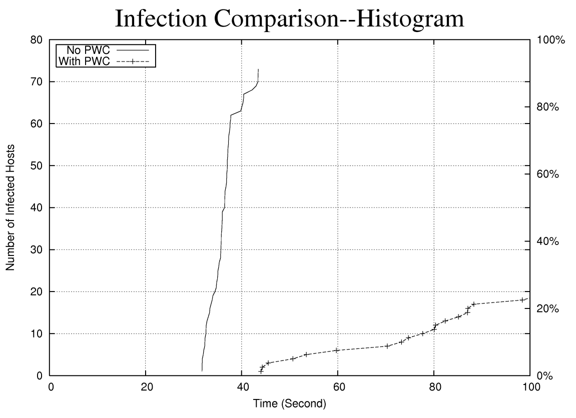

Figure 2. Testbed Emulation Results: the effect of PWC on containing a local scanning worm

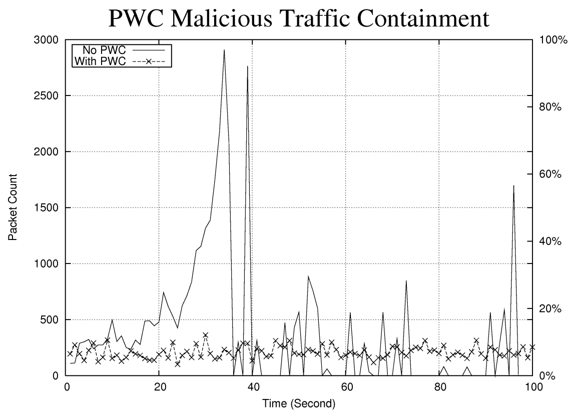

Figure 3. Testbed Emulation Results PWC's effect on containing high volume traffic

gateways (virtual node program) and sending a message to all addresses on this list sequentially.

The first experiment tested the effect of PWC on containing a local scanning worm. An Internet node injected infectious worm packets into the enterprise at rates following our Blaster simulation[12]. Inside the enterprise, the worm randomly chose an initial IP address and began to scan sequentially from that address. Without PWC, the infection ratio was high (72 out of 110 after 120 seconds) and the speed of infection was rapid (about 10 seconds to reach the peak) as shown in Figure 4.3. While with PWC enabled, we saw a marked difference in the same 120-second experiment: the number of infected hosts was reduced to 18 and the speed of infection was much slower, suggesting that the fast local infection was contained and most infection was caused by the incoming scan from the Internet node.

The second experiment was to study the effect of PWC on reducing high volume scanning traffic. In this experiment, infected hosts scanned randomly to global addresses. Because of its random-scanning nature, most of worm scanning traffic was directed to the Internet interface where worm traffic may congest and impact normal traffic. From Figure 4.3, we see that in the case without PWC, there were several peak traffic periods caused by worm scanning and traffic as a whole suffered increased delay after. In the case with PWC, there was no abnormally high traffic volume and the aggregate traffic rate was smooth.

It is worth noting that relaxation analysis was not implemented and there was no additional signature-based worm defense (such as EarlyBird) deployed in the emulation. For this reason, the blocked hosts were kept blocked through the 120-second experiment. The maximum number of hosts that were blocked was about four hundred out of ten thousand. We believe there would be an significant improvement in terms of infection rate, availability recovery, and traffic filtering if those methods were deployed.

5. Dark port detection for locally scanning worms

Most scanning worms chose the whole IPv4 address segment as their target scanning space, though the scanning strategies may vary. As a result of this vast scanning space, even a very fast scanning worm could not finish scanning the whole space efficiently by just one infected host. Take the example of the Slammer worm, it will take one Slammer infected host $2^{32}/10,000 = 429496.7$ seconds to scan the whole space with a speed of 10,000 scans per second. Luckily, there are many concurrent active worm infectives that can work together to jointly scan the space so the total scan and infection time can be reduced. The efficiency of worms to reach full infection or scan the whole address space will be greatly increased when a worm is particularly designed to target an enterprise network whose address space is much more limited. For a /16 network with 256 /24 subnets, a worm can adopt pure enterprise-wide random scanning, hybrid or preferential local random scanning, or enterprise-wide sequential scanning. We will first run testbed emulation experiments to explore the efficiency of local scanning in an enterprise network environment and the danger such local scanning worms pose to the network.

5.1. Test-bed emulation of local worm propagation

The topology we used for emulation is the same network we used for PWC evaluation in section 4. We configured 959 hosts to be vulnerable to the worm attack and they were randomly distributed among 66 sub-networks. We ran three

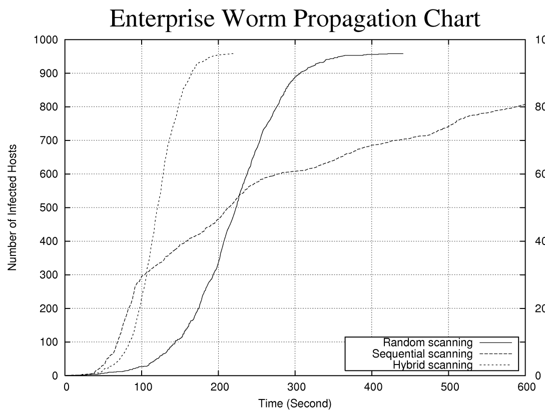

Figure 4. Comparison of local scanning strategies

experiments on this topology with different scanning strategies: the first with a pure enterprise-wide random scanning at a rate of 2 scans per second; the second with a pure enterprise-wide sequential scanning starting from a random address at a rate of 2 scans per second; and the third with a hybrid strategy at a combination of one enterprise-wide random scan per second and one sequential scan per second starting from a sub-network address. There was one initial seed infective in all three experiments. The results are depicted in Figure 4.

Thus, we found that the pure random scanning was more efficient than the pure sequential scanning. The hybrid scanning strategy was the fastest among three strategies and it took less than 200 seconds to infect the majority of susceptible hosts. For such a fast locally scanning worm, there is an urgent need for an effective early-warning detection scheme.

5.2. Dark port scan detection

The danger of worms targeting enterprise networks has been demonstrated through emulation experiments. While there have been some discussions on the topic of enterprise worms and their detection and defense in the literature, none has specifically looked at a pure enterprise-targeting local worm. For this reason, we propose a dark port worm detection scheme and go over the main ideas of this scheme briefly here.

A dark port detection system is comprised of a central security console and a series of soft firewalls. The soft firewalls are installed on the network components connecting a sub-network or cell and the enterprise backbone network, normally a router or a switch which has the ability to monitor all inbound and outgoing packets and can block any one of them. Instead of looking at outgoing packets or connection attempts at the sender site, the soft firewall only inspects incoming connection attempts for the worm detection. The de-

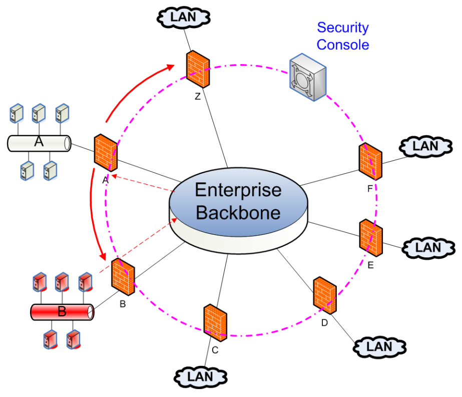

Figure 5. Soft firewall for enterprise networks

tection is rather simple in that any connection attempt whose tuple of (destinationIP:protocol:portnumber) is not on a safe list of the soft firewall could be deemed suspicious. The real task of detection is so on the creation and maintenance of the safe list, or white list of services that are running on the hosts protected by the firewall. Figure 5 is an illustration of proposed soft firewalls in an enterprise network. When firewall A detects a suspicious scan that tries to sneak into cell A by checking its safe list of running services, it will send alerts to the central security console. The security console will issue containment orders to relevant firewalls when some threshold has been reached. Firewall B can then block the corresponding host, or particular service port number, or all hosts in the cell.

Real world traces collected on our lab computers showed that most intra-network sessions involved only a limited set of servers and port numbers. Traces in Table 1 were collected on two normal Windows user hosts and one Linux file server. We removed all packets related to L2R (local to remote) sessions and only kept unicast intra-enterprise sessions. Even the busiest host (trace 1) only contacted 14 distinct local IPs in a period of more than six hours and these local sessions were focusing on 13 TCP and UDP services. A blind scanner will be detected efficiently using the proposed dark port detection.

5.3. Detection of worm propagation

The universal deployment of a dark port detector on subnetwork firewalls facilitates the detection of single random scanners. Detection of single scanner, however, is not the sole purpose of dark port detection system and cannot be used as the evidence and rationale for a collaborative containment action. Only when the system detects an increasing number of hosts participating in scanning and/or timing information indicates actual worm propagation, the IDS will detect the existence of an ongoing worm in the network. For this purpose, we need a sensitive detector for the propaga-

Table 1. Real world intra-enterprise traces

Traces	Length	Distinct Dest IPs	Distinct Destination Ports
Windows Host 1	21951s	14	13 (22,53,88,123,135,137,138,139,389,445, 995,1026,2967)
Windows Host 2	14101s	6	4 (22,53,137,443)
Linux Trace	59633s	2	2 (53,514)

tion of worm infections.

Counting the number of distinct scanning hosts is a straight-forward method for worm detection, but only counting the number of scanning hosts does not consider infection timing information which is an important indicator of threat severity. The inter-arrival time of successive worm scanners may be a good measure to detect the existence and urgency of worm propagation. For a pure random- scanning worm in a /16 network, the expected time for a new infection can be calculated using a formula similar to equation (4) in [19]:

$$T_k = \frac{log(1 - \frac{1}{N_s - I_{k-1}})}{\sigma I_{k-1} log(1 - 2^{-16})} \qquad (1)$$

Here N_s is the total number of susceptible hosts, I_{k-1} is the number of infected hosts at current time, and T_k is the expected wait time for the next infection. The formula can be further simplified as $T_k = \frac{N}{\sigma I_{k-1}(N_s - I_{k-1})}$, where N is the amount of local network address space (2^{16} for a /16 network). It is easy to find that when the number of infected hosts increases, the expected time for the next infection decreases. This phenomenon of decreasing inter-arrival times can be contrasted with the behavior of other background scan noise where no consistent trend in inter-arrival times of distinct scanners is expected. Using data collected by honeypot computers deployed in an actual enterprise network, we plotted the curve of inter-arrival times of background scanners together with the inter-arrival times of a random scanning worm in a /16 network in Figure 6. For the generation of worm inter-arrival times, we set the total number of susceptible hosts to be 1,000 and the scan rate per worm victim to 2. From the figure, we can see though the inter-arrival times of background scanners sometime fell to a rather low value, and there is no consistent trend among them over a long period of time, compared to the strong decreasing trend in the early stages of worm propagation. This feature was the basis for our worm propagation detection in our emulation experiment.

6. Dark port Containment and DETER evaluation

When the aggregated level of suspicious scan activities is above some pre-defined threshold, the central security console will decide appropriate actions to deter possible worm

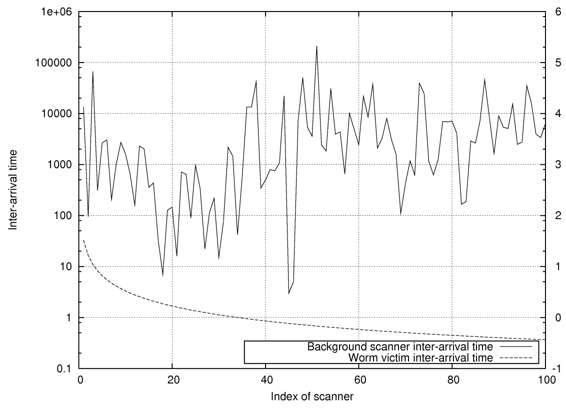

Figure 6. Comparison of inter-arrival time among distinct scanners between a random scanning worm and background noise

propagation. The set of actions could include doing nothing, blocking individual hosts from initiating any new connection attempt, blocking particular service port numbers from accepting new connections, or blocking all new service requests. Implication of blocking all new service requests or on some specific port numbers could be severe since it may disrupt important network transactions within the enterprise boundary. To avoid taking such drastic block action, some simple intuitive or rule-of-thumb decision rules can be used. For example, when security alert analysis indicates that there are rather widespread scanning activities within the network and the majority of scans are focusing on one service port number, the decision should be sending commands to all security agents to block any new connection requests targeting that particular port number.

We built a prototype defense system which implemented only selective block actions and ran emulation experiments to test the effectiveness of it on containing a hypothetical local-only scanning worm. Network configuration and density of susceptible population were the same as we used for propagation experiments in section 5.1. Each infected host sent out a combination of one enterprise-wide random scan and one sub-network random scan per second. We assumed zero false negative in dark port detection, which means that each scan on a non-existent service port was reported to the

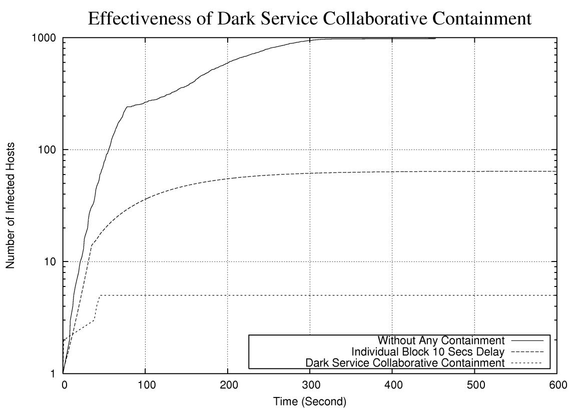

Figure 7. Effectiveness of Dark port Containment

central security console right after it arrived at the receiving soft firewall. The central security console maintained the following records: the number of dark port alerts per service port, the times of new distinct IPs sending the first scan, and the number of distinct cells that had been reported sending scans per service port. The method for propagation detection we used in the emulation was based on the fact of decreasing inter-arrival infection times we introduced in the last section. The actual algorithm is in the form of sequential likelihood ratio test and the details and its evaluation are presented in a separate paper. When the worm propagation measurement is above the threshold, the central security console would issue a selective block command on the related port(s) to all soft firewalls. The thresholds on the rate of new infections and distinct scan-sending cells should be adjusted according to the local network traffic to have a satisfactory detection performance in real world application. Figure 7 shows the effect of our containment.

Collaborative containment performed well and outperformed individual block or address-blacklisting, one of few workable containment strategy for this local scanning worm, as shown in the figure. The dark port collaborative containment activated selective port block on all hosts after a few hosts were infected and none was infected after the containment. Independent individual containment with a delay of 10 seconds resulted a final size of about 60 infected hosts. The detection based on failed scans only is also less prone to counter-detection by forged benign scans by a 'smart' worm. The drawback of selective block on all cells is the potential service disruption on normal hosts. To fully account for the loss of network service by worm containment actions and find an optimal defense strategy, we will need a quantitative evaluation framework to run the cost-benefit analysis of different containment strategies.

7. Summary and future work

Testbed emulation experiments demonstrated the efficiency of locally scanning worms with advanced scanning strategies. The defense to this threat entails early detection and collaborative containment. Selective block containment based on a new dark port detection scheme showed promise in this regard through experiments run on the DETER testbed. Also in this article, we presented the evaluation results of another collaborative containment scheme that targets fast scanning worms. The experimental results clearly demonstrated the effectiveness of collaborative containment on worm propagation.

Our future work includes fine-tuning of selective block strategy to reduce service disruption. Improving the communication security between collaborating security agents is another important task. Also, we will explore related issues of botnet emulation and defense evaluation in our future work.

References

[1] EMIST project. http://emist.ist.psu.edu.

[2] T. Benzel, B. Braden, D. Kim, C. Neuman, A. Joseph, K. Sklower, R. Ostrenga, and S. Schwab. Experience with DETER: A testbed for security research. In *Proc. 2nd International IEEE/Create-Net Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCOM'06)*, Barcelona, Spain, March, 2006.

[3] L. Briesemeister, P. Porras, and A. Tiwari. Model checking of worm quarantine and counter-quarantine under a group defense. Technical Report SRI-CSL-05-03, Computer Science Laboratory, SRI, 1992.

[4] Shigang Chen and Yong Tang. Slowing down Internet worms. In *ICDCS '04: Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, pages 312–319. IEEE Computer Society, 2004.

[5] Evan Cooke, Michael Bailey, Z. Morley Mao, David Watson, Farnam Jahanian, and Danny McPherson. Toward understanding distributed blackhole placement. In *WORM '04: Proceedings of the 2004 ACM workshop on Rapid malcode*, pages 54–64, New York, NY, 2004. ACM Press.

[6] D. Dagon, X. Qin, G. Gu, W. Lee, J. Grizzard, J. Levin, and H. Owen. Honeystat: Local worm detection using honeypots. In *Proceedings of The 7th International Symposium on Recent Advances in Intrusion Detection (RAID 2004)*, Sophia Antipolis, France, September 2004.

[7] Prahlad Fogla, Monirul Sharif, Roberto Perdisci, Oleg Kolesnikov, and Wenke Lee. Polymorphic blending attacks. In *Proc. 15th USENIX Security Symposium*, 2006.

[8] Y. Jhi, P. Liu, L. Li, Q. Gu, J. Jing, and G. Kesidis. Proactive containment of fast scanning worms through white detection. In *Proceedings of 3rd International Conference on Security and Privacy in Communication Networks*, September 2007.

[9] G. Kc, A. Keromytis, and V. Prevelakis. Countering code-injection attacks with instruction-set randomization. In *Pro-*

ceedings of the 10th ACM conference on Computer and communications security, pages 272 – 280, October 2003.

[10] G. Kesidis, I. Hamadeh, and S. Jiwasurat. Coupled Kermack-Mckendrick models for randomly scanning and bandwidth saturating Internet worms. *ACM TOMACS*, 2007.

[11] S. Kornexl, V. Paxson, H. Dreger, A. Feldmann, and R. Sommer. Building a time machine for efficient recording and retrieval of high-volume network traffic. In *Proc. Internet Measurement Conference 2005*, Berkeley, CA, Oct, 2005.

[12] L. Li, S. Jiwasurat, I. Hamadeh, P. Liu, G. Kesidis, and C. Neuman. Emulation of sequential scanning worms in large enterprises. In *Proc. 2nd International IEEE/Create-Net Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCOM'06)*, Barcelona, Spain, March, 2006.

[13] L. Li, S. Jiwasurat, P. Liu, and G. Kesidis. Emulation of single packet UDP scanning worms in large enterprises. In *Proc. 19 International Teletraffic Congress (ITC19)*, Beijing, China, August, 2005.

[14] L. Li, P. Liu, and G. Kesidis. Visual toolkit for network security experiment specification and data analysis. In *VizSEC '06: Proceedings of the 3rd international workshop on Visualization for computer security*, pages 7–14, New York, NY, USA, 2006. ACM Press.

[15] Z. Li, M. Sanghi, Y. Chen, M. Y. Kao, and B. Chavez. Hamsa: Fast signature generation for zero-day polymorphic worms with provable attack resilience. In *Proceedings of IEEE Symposium on Security and Privacy*, 2006.

[16] M. Liljenstam, D. Nicol, V. Berk, and R. Gray. Simulating realistic network worm traffic for worm warning system design and testing. In *Proc. ACM WORM*, Washington, DC, 2003.

[17] J. Newsome, B. Karp, and D. Song. Polygraph: Automatic signature generation for polymorphic worms. In *IEEE Security and Privacy Symposium*, May 2005.

[18] P. Porras, L. Biesemeister, K. Levitt, J. Rowe, K. Skinner, and A. Ting. A hybrid quarantine defense. In *Proc. ACM WORM*, Washington, DC, Oct. 29, 2004.

[19] Moheeb Abu Rajab, Fabian Monrose, and Andreas Terzis. Worm evolution tracking via timing analysis. In *WORM '05: Proceedings of the 2005 ACM workshop on Rapid malcode*, pages 52–59, New York, NY, USA, 2005. ACM Press.

[20] S. Schechter, J. Jung, and A. Berger. Fast detection of scanning worm infections. In *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID)*, September, 2004.

[21] Sumeet Singh, Cristian Estan, George Varghese, and Stefan Savage. Automated worm fingerprinting. In *OSDI*, pages 45–60, 2004.

[22] Y. Tang and S. Chen. Defending against Internet worms: A signature-based approach. In *Proc. of IEEE INFOCOM'05*, Miami, Florida, May 2005.

[23] N. Weaver, I. Hamadeh, G. Kesidis, and V. Paxson. Preliminary results using scale-down to explore worm dynamics. In *Proc. ACM WORM, Washington, DC*, Oct. 2004.

[24] N. Weaver, S. Staniford, and V. Paxson. Very fast containment of scanning worms. In *Proc. 13th USENIX Security Symposium*, 2004.

[25] Matthew M. Williamson. Throttling viruses: Restricting propagation to defeat malicious mobile code. In *ACSAC*, pages 61–68. IEEE Computer Society, 2002.

[26] V. Yegneswaran, J. Giffin, P. Barford, and S. Jha. An architecture for generating semantic-aware signatures. In *Proc. 14th USENIX Security Symposium*, 2005.

[27] Cliff Changchun Zou, Weibo Gong, and Don Towsley. Worm propagation modeling and analysis under dynamic quarantine defense. In *WORM '03: Proceedings of the 2003 ACM workshop on Rapid malcode*, pages 51–60, New York, NY, 2003. ACM Press.