Microscopic Simulation of a Group Defense Strategy

Linda Briesemeister and Phillip Porras *
SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025
first.lastname@sri.com

Abstract

We introduce a novel worm containment strategy that integrates two complementary worm quarantine techniques. The two techniques are linked, with one strategy employing the other as an indicator of worm infection. A group defense mechanism shares such indicators among neighboring networks, and when enough corroboration occurs, the network engages in traffic filtering to halt infection attempts.

We present an SSFnet-based microscopic simulation of the containment strategy against random scan worms, and explore various performance characteristics of the group defense mechanism. The simulation results help to characterize the conditions and degree to which the integrated quarantine strategy can both slow worm propagation and prevent the worm from reaching its full saturation potential.

1. Introduction

The increasing speed and sophistication of Internet worms has offered strong motivation toward the development of automated worm defense strategies. Some strategies seek to delay worm propagation by restricting the consumption of resources that a worm utilizes as it attempts to propagate [20, 16, 21]. Some strategies attempt to disrupt or thwart the ability of the worm to discover new susceptible hosts within its search space [13, 18, 5]. Other defenses propose to block worm propagation through cooperative information sharing that allows defensive filtering to leap ahead of the worm as it propagates across the network seeking new victims [1, 11].

In [12] we examined the potential synergies that exist between two competing worm defense strategies by overlaying them within an enterprise network simulation. We observed how the independent overlay of a resource-limiting strategy with a cooperative leap-ahead (LA) defense provides a synergistic effect that reduces worm infection growth rates significantly more than either approach could independently achieve. We presented an analytical argument to explain the basis of this synergy.

In this study, we examine an integrated worm defense strategy, where connection rate limiting (RL) serves as the input source to an LA approach. Here, the RL algorithm is used to identify local worm infections and slow their propagation attempts to remote addresses. Each threshold violation is monitored and shared by the local network using an LA protocol. Each local network attempts to corroborate the occurrences of reported RL threshold violations across the greater network. The repeated experiences of RL threshold violations among peer network domains may cause a local network to enter a defensive posture, such as enabling a filter to suppress incoming infection attempts. We describe how dynamic filters could be derived through the examination of RL-suppressed packets, and suggest how this approach provides substantial worm growth suppression in the presence of random scanning worms at different speeds. For some classes of worms, this solution offers a signature-free approach to both slow the infection growth rate and prevent the worm from reaching its full saturation potential within a group of peer networks.

We examine the effectiveness of this integrated and interdependent worm defense strategy through microscopic network simulations of various random scan worms using the Scalable Simulation Framework (SSF) [15]. We call this approach a microscopic worm simulation not because of the size of the network, but rather that we do not aim to abstract the whole Internet or even an autonomous system (AS) backbone. While we admittedly employ a simplified and abstract network topology, we examine other subtleties of the integrated combination strategy by incorporating into our simulation common network protocols, greater implementation detail of the worm and our algorithms, and modeling of finer network traffic and link characteristics.

* This material is based upon work supported by the National Science Foundation under Grant No. ANI-0335299, through a subcontract with the University of California at Davis, Contract No. 01RA0052.

2. Related Work

Worm containment research to combat the growing threat of large-scale malware epidemics is an active field of study. By *worm containment* we refer to methods that attempt to isolate infected computing assets from the uninfected segments of a network. Detection may be one step in the containment process, followed by techniques such as traffic filtering or address blacklisting, which could be directed at either inbound or outbound communications.

Resource limitation strategies have been proposed to suppress the speed at which worms propagate by throttling the resources that they are known to consume as they seek out new infection targets. In particular, the monitoring and prevention of outbound network connections has been widely studied as an effective strategy to suppress the infection growth rate of worms. Connection rate limiting at the end host [20], the network gateway [16], and within backbone routers [21] has been proposed. Rate limiting through process-based monitoring of outbound connection rates across the total population of processes or from a preselected group of known benevolent processes has also been proposed [6]. Ganger et al. [3], and more recently [19], suggest that limiting the number of network connections not facilitated through DNS lookups provides an effective defense for certain classes of worms. The simulation presented here implements outbound connection rate limiting at the egress of each local network.

Several cooperative sharing algorithms have been proposed to allow networks to recognize the emergence of propagating attacks. These algorithms can employ a hierarchical coordination center to model threats that span multiple networks. GrIDS [17] is an early example of a hierarchical detection scheme to identify coordinated worm activity and coordinated floods or sweeps. More recently, peer-sharing schemes have been described to allow individuals to independently corroborate the emergence of worm activity and take defensive actions before the worm can compromise their networks. We refer to these as *leap-ahead (LA)* defenses, in that networks have the potential to activate a defensive posture before the worm reaches them, and thus can slow or prevent the worm from reaching its full saturation potential.

COVERAGE [1] is an example peer-sharing scheme, in which each participating network randomly polls a set of remote hosts for reports of worm activity at periodic intervals. Friends [11] is a peer-sharing scheme similar to COVERAGE. However, under the Friends protocol the peer groups are pre-selected, and alerts are pushed to friends asynchronously rather than pulled at periodic intervals. The LA strategy incorporated into our integrated strategy operates in a way similar to Friends, but with several key differentiators. We prevent our sharing protocol from allowing a single detector from inducing filter modes across a group of peer networks, we do not allow a domain to enter defensive mode based solely on local alerts, and we integrate RL thresholds violation monitoring as the worm detection mechanism.

Many worm defense studies evaluate their proposed solutions by analyzing traffic traces or omit more details about the simulation framework they used. Riley et al. [14] employ the Georgia Tech Network Simulator (GTNetS) to simulate worm infection on the scale of the Internet. The paper reports on the detailed implementation of TCP and UDP worms and on the enhancements of the simulator to achieve the desired scale. In contrast to our paper, the authors do not study worm mitigation strategies and in turn, we focus less on the scale of the simulated network.

Liljenstam and Nicol [9] tackle the challenge of worm simulation on the scale of the Internet. The authors use a mixed abstraction-level technique that uses epidemic models to abstract from the details of the worm behavior. In subsequent papers [7, 8] the simulation framework is used to compare so-called active (such as counterworms) with passive (such as patching) worm defense strategies.

3. Group Defense Strategy

The integrated group defense strategy employs an outbound connection rate limiting algorithm at the egress router of each local network. An LA algorithm shares control of the egress router with a rate limiter, and uses RL threshold violations to identify potential worm infections among the local end hosts. The LA protocol broadcasts alerts about RL threshold violations to its peers. The corroboration of RL threshold violations from multiple group members may cause an LA process to enable inbound traffic filters, which may be derived through an examination of the suppressed packets (see Subsection 3.2). Once in the defensive posture, the LA process continues to maintain filtering while peer networks contribute RL threshold violations, and transitions out from this posture if no alerts are raised over time.

Figure 1 depicts our algorithm in more detail. First, two counters, a (alert level) and c (hold-off), are initialized to zero. In every time step, both counters are decremented but never fall below zero (the right logic branch). It is through this temporal decay of a that the LA process transitions from the defensive posture back to the normal state when it fails to receive continued alerts of worm activity.

When a local RL threshold violation occurs, the local LA process sends an alert to its selected peers weighted by a

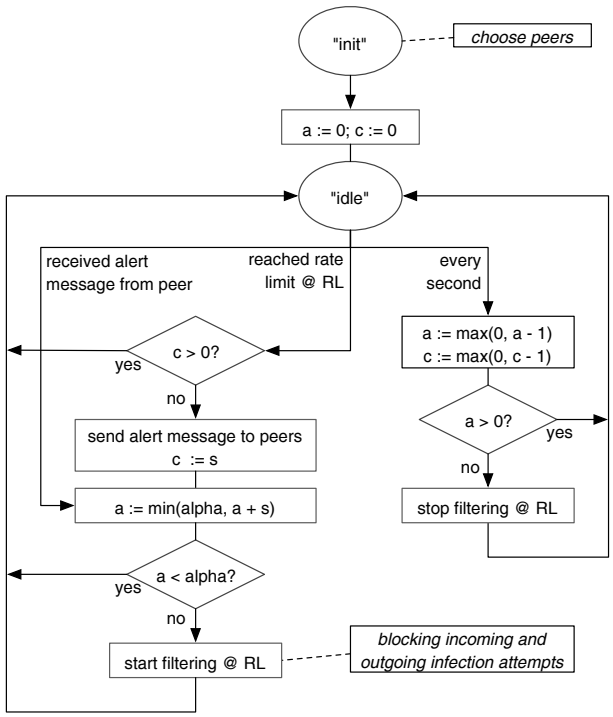

Figure 1. LA flow diagram

parameter s (severity). A hold-off counter c is employed by the LA process to suppress repeated RL threshold warnings for a period of s seconds. This mechanism prevents a single end host that generates a stream of RL threshold violations over multiple seconds from causing the entire group to enter the defensive posture. Rather, the algorithm requires corroboration from multiple peer contributors before any network may enter the defensive posture.

When receiving alerts or when the RL threshold is violated outside of the hold-off period ($c = 0$), the alert level a increases by s, but never above the alert threshold $\alpha = r \cdot s$. When the alert level reaches α, we say that the host is in defensive posture, which may include the enabling of a filter that selectively blocks traffic matching the characteristics of the packets that are suppressed by the rate limiter (see Subsection 3.2). With respect to the simulations performed in this study, we assume that the filtering mechanism will completely block subsequent worm traffic, which in practice may not be achievable without the use of broad traffic filters and may itself be subject to worm countermeasures. Once the alert level decays to zero, the defensive posture is abandoned. For our simulation, this means that we drop our guards and let traffic flow freely within the limits of the RL component.

The alert threshold α defines the corroboration required to enter defensive posture. This may prove more effective in preventing false positives from impacting the defense strategy. On the other hand, a smaller α permits defenders to react much more quickly to arising alerts and gives the defense a better chance of leaping ahead of the spreading worm.

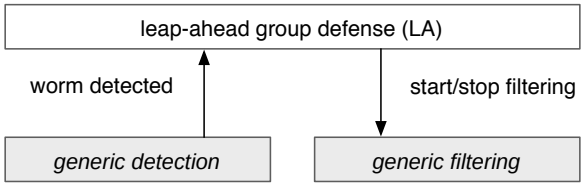

Figure 2. LA interaction with generic modules

Another parameter of LA is the size of the group of peers that exchange alert messages. Each LA selects $F = G - 1$ peers at the beginning of operation. To reach optimal coverage over the population of LA hosts, we employ the following scheme to ensure that each LA host becomes a peer of the same number of other LA hosts. We arrange the IP addresses of all known LA hosts in a ring of ascending order and then select the next $G - 1$ addresses from a given LA's address to be the peers to which this LA sends messages. In this study, we deploy the RL/LA functionality at routers that serve as a gateway for a group of end hosts.

3.1. Generic Worm Detection and Filtering

In principle, our group defense strategy works with any kind of worm detection and filtering mechanism. One could replace the RL portion in our algorithm—which currently implements both, detection and filtering—simply by altering the notification "reached rate limit @ RL" in the flow diagram of Figure 1 to suit another worm detection. In the same manner, the filtering component that implements defensive posture, can be exchanged for example, with an implementation of the scheme described in detail in Subsection 3.2. The specification of the LA algorithm then directs the notifications "start filtering @ RL" and "stop filtering @ RL" to the appropriate module. Figure 2 shows how our LA algorithm can interact with generic worm detection and filtering components.

3.2. Dynamic Filter Generation

The basis for connection rate limiting algorithms is the observation that a surge in end host transmissions to unique remote addresses offers a reasonable heuristic to diagnose

a potential infection of certain classes of worms. While refinements of this approach have been suggested, which include considerations such as DNS assistance [3, 19], the approach is by no means free of false positives. In practice, non-worm-related threshold violations may produce delays in end host operations that should not cause permanent harm.

We leverage the RL threshold violation as one potential data point for worm activity, where it is the corroboration of multiple RL threshold violations from across the peer group that provides the leap-ahead algorithm with an opportunity to potentially formulate and insert an inbound connection filter.

For connection filtering to be applicable, the LA component must observe some degree of correlation among the packets of the end hosts that trigger the RL violations. For example, consider the Slammer worm, which employs a buffer overflow packet on UDP port 1434 and a random scan propagation strategy. Each locally infected end host will cross the RL threshold as its instantiation of Slammer attempts to propagate. For worms that similarly produce a strongly correlated infection packet, a dynamic filtering strategy for assisting the LA component could be implemented as follows.

When end host E triggers RL threshold at time t:

- Select port/protocol pair (A/B) as the dominant target port/protocol used by E's outbound packets during t
- Select datagram size S as the average datagram and D as the standard deviation of datagrams sent by E during t
- Report an RL threshold violation to LA with E, A/B, S, D

When the LA process at Router N observes $a \geq \alpha$:

- Construct filter F such that F denies incoming A/B packets
- When $D < \epsilon$, augment F to filter packets of size $S \pm \epsilon$

Regardless of whether inbound worm filtering is formulated through automated or manual processes, the selectiveness of the filters will greatly depend on the degree to which the worm produces a common traffic pattern. However, selective filter generation may be substantially more difficult when worm infections manifest greater packet diversity.

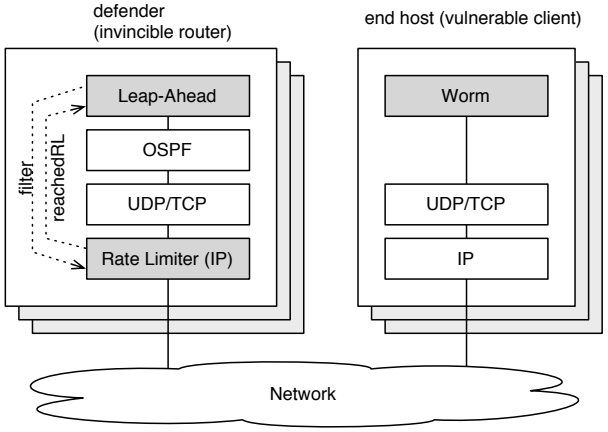

Figure 3. Protocol stacks implemented in SSFNet

4. Simulation and Results

We study our group worm defense using a simulator written in Java implementing the SSFNet [15]. In contrast to many worm simulations, we focus on the worm behavior at a microscopic level. Our motivation here is to study the behavior of the worm defense at the edge of the network rather than the propagation of the worm in the context of the broader Internet.

Figure 3 shows how we implemented—gray colored boxes denote our additions to SSFNet—the proposed algorithms RL and LA on a defender host. Both protocols communicate with each other in the following situations. First, the rate limiter notifies LA when the rate limit has been reached. Second, the leap-ahead algorithm instructs RL when to start filtering packets in defensive posture. We also implemented an application-layer random-scanning UDP worm that runs on vulnerable hosts.

The network topology consists of $M = 100$ LANs, each containing one egress router acting as the defender of this LAN and $k = 10$ vulnerable end hosts. These 100 LANs are connected through a small backbone network of 24 routers. The backbone network is organized into two tiers with 20 routers in an outer ring each connecting to five LANs and four routers in an inner ring each connecting to four routers of the outer ring.

The worm implementation uses only one UDP packet to infect other hosts. It scans the address space between 0 and $2^{16} - 1$ randomly with three different rates: 10 (low speed), 100 (medium speed), and 1000 (high speed) scans per second.

The RL component running at the IP layer of the egress router allows each end host of the connected LAN to communicate with at most 10 different remote IP addresses per second. Packets to other remote IP addresses get dropped after this limit has been reached during one time step. The RL protocol also implements inbound and outbound filtering of worm messages on behalf of the LA component when the router activates defensive posture. This filtering occurs independently of the rate limiting.

The LA algorithm chooses nine peers ($G = 10$) to which to send alerts. It employs a severity of $s = 3$ to alerts and incidents, and varies $r = 2, 3, 4, 5$, which results in values for $\alpha = 6, 9, 12, 15$, respectively.

Each simulation run starts with one end host being initially infected and runs for 200 simulated seconds or until all vulnerable end hosts are infected. We compare simulation runs without any defense (baseline), with a rate-limiting defense only, and with the combined RL/LA approach. We carry out 100 runs per parameter set.

For each run, we collect the following metrics over time. First, we measure the percentage of vulnerable end hosts currently infected, ρ. Second, we record the number of defenders whose rate limit was reached (at one or more end hosts) during the past second. Our metric R then denotes the percentage of defenders who reached the rate limit. Third, we count the number of defenders who are filtering at each second. For filtering and not filtering defenders, we also count the number of defenders having none and at least one infected end host in their LANs. The metric L captures the percentage of defenders who are in filtering mode at the end of each time step.

Figure 4 shows the simulation results of ρ for a high-speed and medium-speed worm. We compare our approach to simulations without any defense and to simulations employing rate limiting only. The rate limiting slows the spreading of the worm, but eventually the whole population becomes infected. For different values of α our approach successfully curbs the spreading of the worm. As expected, a small value of α leads to slower infection spread than a large value of α because a low alert threshold permits defenders to switch into defensive posture (filtering) much more quickly. However, with a high value of α we achieve greater corroboration and thus believe the strategy to be more resilient to false positives.

Given the results in Figure 4 for high-speed and medium-speed worms, the results of ρ for the low-speed worm (Figure 5, top) seem surprising at first. Why does our approach not leap ahead of a slow worm even better than a fast worm? The answer lies in the underlying rate-limiting component that works at the scanning rate of the simulated worm. However, comparing RL alone to a defenseless network, the rate

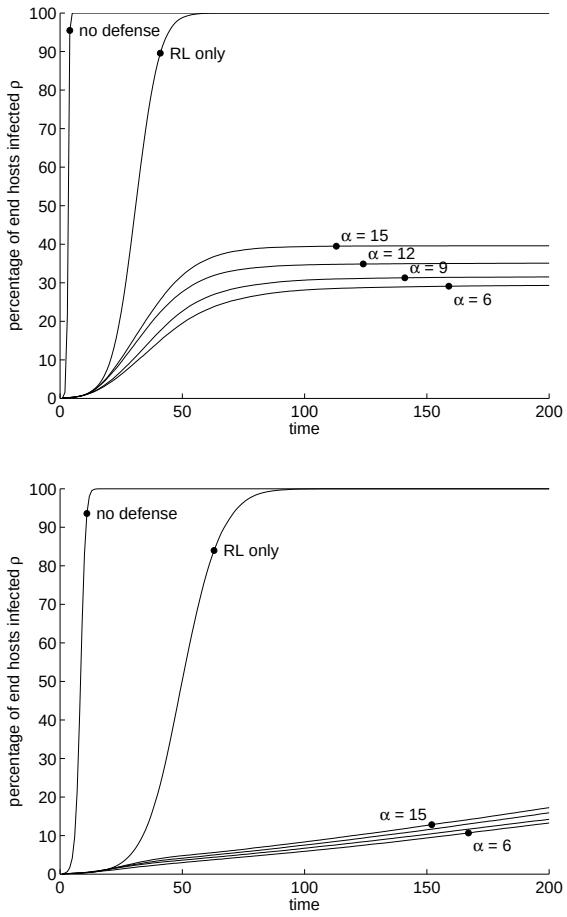

Figure 4. Results for ρ under high-speed (top) and medium-speed worm (bottom) for $\alpha = 6, 9, 12, 5$ compared with defenseless network and rate limiting only

limiting does not seem to have much of an impact. Hence, it seems unusual that the combined approach curbs worm spreading at all. The simulation traces show us that even though the worm sends only 10 packets per second, these packets get scheduled and delayed in the network and may turn up later at the defending router and then potentially trigger the rate limit. Figure 5, bottom, depicts the percentage of defenders who have at least one associated end host exceeding the rate limit threshold. As more and more end hosts become infected the rate limit metric R grows, meaning that more defenders become aware of it and eventually activate filtering to curb the spread of the infection.

Taking a closer look at the behavior of the defenders, we plot the percentage of defenders in filtering mode L as a thick line in Figure 6. The top graph of Figure 6 shows L

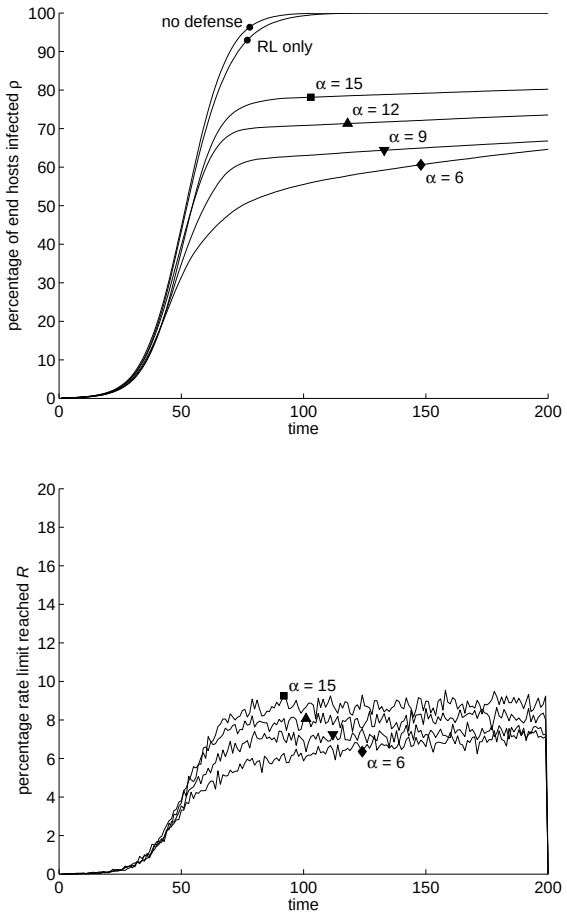

Figure 5. Results for ρ (top) and R (bottom) under low-speed worm for $\alpha = 6, 9, 12, 5$; ρ compared with defenseless network and rate limiting only

for the case of a slow worm and $\alpha = 6$, which corresponds to the situation in Figure 5. Indeed, as the percentage of defenders whose rate limit has been reached increases, more and more defenders switch into active filtering of inbound and outbound worm traffic.

In Figure 6, we also distinguish, within each group of filtering and not-filtering defenders, between those guarding a network without any infection (lighter gray) and a network with at least one end host infected (darker gray). The area above the thick line corresponds to defenders not filtering, whereas the area below denotes filtering defenders. With this distinction, we conclude the following. Many defenders reach their defensive posture after about 80 seconds but with just occasional triggers from a slow worm operating at the detection level. However, the defensive posture

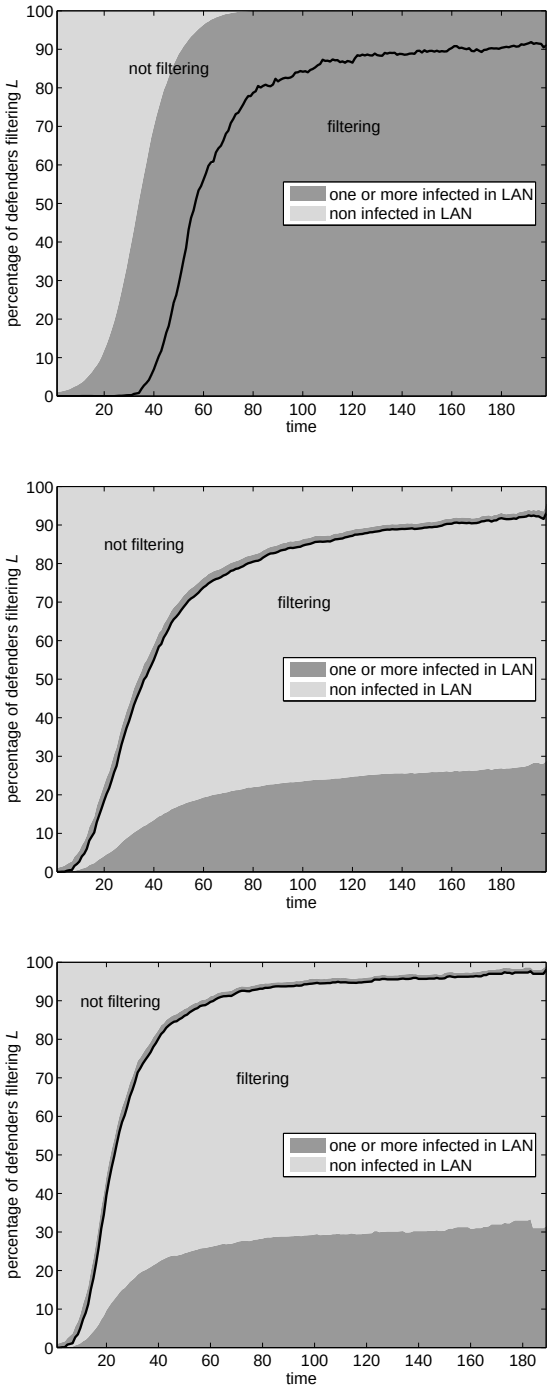

Figure 6. Results for L (thick line) under a low-speed (top) and medium-speed (middle) and high-speed (bottom) worm for $\alpha = 6$

comes too late, as each filtering defender has at least one infection in its own network. These infections continue to spread locally even while defenders are filtering outgoing traffic. Therefore, the leveled curves of ρ in Figure 5 are deceiving in that the worm will reach saturation eventually, albeit much more slowly. Note that even when the combination of both dark areas add up to 100%, which means that all defenders have infected hosts in their networks, the metric ρ does not have to reach 100% at the same time as some of the end hosts in an "infected" LAN may still not be infected.

In Figure 6, the middle and bottom graphs show L and how the defenders' states are locally infected or not for the medium-speed and high-speed worm, respectively. Again, we display the results for $\alpha = 6$. In contrast to the graph at the top, the majority of filtering defenders do not have an infected end host in their networks. This metric demonstrates that our RL/LA approach indeed leaps ahead of the worm and causes defenders to move into defensive posture *before* their networks become affected. The results for other values of α are similar and are omitted here.

4.1. Confidence Intervals

We determine the quality of the simulation output by using independent replications analysis [4, 2]. We stopped every simulation run when a certain criterion was met. Thus, these runs fall into the category of terminating simulations. For each run, the pseudo random number generator (Mersenne Twister [10]) starts with a new initial seed and hence produces a replication independent from other runs. If the number of independent replications is large enough, a central limit theorem allows us to assume that the replicate outputs are approximately i.i.d. (independent, identically distributed) normal. We then compute the half-width of the $100(1-\alpha)\%$ confidence interval as stated in (1) where $t(d,p)$ represents the p quantile of Student's t distribution with d degrees of freedom. We set the confidence level to 95% corresponding to $\alpha = 0.05$.

$$M_p \pm t(n-1, 1-\frac{\alpha}{2}) \cdot \sqrt{\frac{V_p}{n}} \quad (1)$$

where M_p : mean of data with parameter p
V_p : standard deviation of data with parameter p
n : number of independent replications

It is beyond the scope of this paper to report on all confidence intervals obtained through our simulation. As an example, we list in Table 1 the confidence intervals for metric ρ at the end of the simulation runs. These confidence intervals are roughly maintained over the course of a simulation run. For 100 independent runs that we carried out for each parameter set, the margin of error of ρ varies between 2.8 and 4.1 around the mean values determined for ρ.

5. Conclusion

We examine an integrated worm defense strategy, where connection rate limiting (RL) serves as the input source to a leap-ahead (LA) approach. We present the algorithms in detail and discuss some possible extensions such as constructing dynamic filters.

We carry out microscopic simulations of our defense strategy against random scanning worms of various speed. A combination of metrics shows that the defense leaps ahead of medium-speed and high-speed worms and effectively curbs the spread of infection. For a slow worm scanning at the connection rate limit, the threshold is violated less often. Still, the defense can achieve corroboration, shifting the defenders across the network into defensive posture. However, a closer look at a combination of simulation metrics reveals that in the case of a slow worm, all local networks in defensive posture contain at least one infected host, which will eventually lead to full saturation of the network. We conclude that in this case the defense is not able to leap ahead of the worm for the lack of infection indicators.

References

[1] K. G. Anagnostakis, M. B. Greenwald, S. Ioannidis, A. D. Keromytis, and D. Li. A cooperative immunization system for an untrusting internet. In *Proceedings of the 11th IEEE International Conference on Networks (ICON'03)*, October 2003.

[2] J. M. Charnes. Statistical analysis of output processes. In *Proceedings of the 25th Winter Simulation Conference*, pages 41–49. ACM Press, December 1993.

[3] G. R. Ganger, G. Economou, and S. M. Bielski. Self-securing network interfaces: What, why and how. Technical report, Computer Science Department, Carnegie Mellon University, 2002.

[4] D. Goldsman. Simulation output analysis. In *Proceedings of the 24th Winter Simulation Conference*, pages 97–103. ACM Press, December 1992.

[5] S. P. Gorman, R. G. Kulkarni, L. A. Schintler, and R. R. Stough. Least effort strategies for cybersecurity. Technical report, George Mason University, 2003.

[6] M. Gualtieri and D. Mossé. Limiting worms via QoS degradation. Technical report, Computer Science Department, University of Pittsburgh, 2003.

[7] M. Liljenstam and D. M. Nicol. Comparing passive and active worm defenses. In *Proceedings of the First International Conference on the Quantitative Evaluation of Systems (QEST)*, pages 18–27, September 2004.

α	slow-speed worm $\rho(t = 200)$ [%]	medium-speed worm $\rho(t = 200)$ [%]	high-speed worm $\rho(t = 200)$ [%]
6	64.6 ± 3.1	13.2 ± 2.9	29.3 ± 3.9
9	66.8 ± 4.1	14.2 ± 2.8	31.5 ± 3.8
12	73.5 ± 4.0	15.9 ± 3.1	35.1 ± 3.6
15	80.2 ± 4.0	17.2 ± 3.2	39.6 ± 3.7

Table 1. Example of confidence intervals for ρ

[8] M. Liljenstam and D. M. Nicol. Comparing passive and active worm defenses. In *Proceedings of the 1st International Conference on Quantitative Evaluation of Systems (QEST)*, pages 18–27, 2004.

[9] M. Liljenstam, Y. Yuan, B. J. Premore, and D. M. Nicol. A mixed abstraction level simulation model of large-scale Internet worm infestations. In *10th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 109–116. IEEE Computer Society, 2002.

[10] M. Matsumoto and T. Nishimura. Mersenne Twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8(1):3–30, 1998.

[11] D. Nojiri, J. Rowe, and K. Levitt. Cooperative response strategies for large scale attack mitigation. In *DARPA Information Survivability Conference and Exposition*, pages 293–302, 2003.

[12] P. Porras, L. Briesemeister, K. Skinner, K. Levitt, J. Rowe, and Y.-C. A. Ting. A hybrid quarantine defense. In *Proceedings of the 2004 ACM workshop on Rapid malcode (WORM)*, pages 73–82. ACM Press, 2004.

[13] N. Provos. A virtual honeypot framework. In *Proceedings of the 12th USENIX Security Symposium*, pages 1–14, August 2004.

[14] G. F. Riley, M. I. Sharif, and W. Lee. Simulating internet worms. In *Proceedings of the 12th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 268–274, 2004.

[15] Scalable Simulation Framework. http://www.ssfnet.org/.

[16] S. Staniford. Containment of scanning worms in enterprise networks. *Journal of Computer Security*, 2004.

[17] S. Staniford-Chen et al. GrIDS—A graph based intrusion detection system for large networks. In *Proceedings of the 19th National Information Systems Security Conference*, volume 1, pages 361–370, October 1996.

[18] H. Wang, C. Guo, D. Simon, and A. Zugenmaier. Shield: Vulnerability-driven network filters for preventing known vulnerability exploits. Technical report, Microsoft Research, Technical Report MSR-TR-2003-81, 2004.

[19] D. Whyte, E. Kranakis, and P. van Oorschot. DNS-based detection of scanning worms in an enterprise network. In *Proceedings of the 12th Network and Distributed System Security Symposium (NDSS)*, pages 181–195, February 2005.

[20] M. M. Williamson. Throttling viruses: Restricting propagation to defeat malicious mobile code. In *Proceedings of the 18th Annual Computer Security Applications Conference*, page 61. IEEE Computer Society, 2002.

[21] C. Wong, C. Wang, D. Song, S. Bielski, and G. R. Ganger. Dynamic quarantine of Internet worms. In *Proceedings of the International Conference on Dependable Systems and Networks DSN-2004*, June 2004.