Automatically Deducing Propagation Sequences that Circumvent a Collaborative Worm Defense

Linda Briesemeister and Phillip A. Porras
Computer Science Laboratory, SRI International
333 Ravenswood Avenue, Menlo Park, CA 94025
firstname.lastname@sri.com

ABSTRACT

We present an approach to the question of evaluating worm defenses against future, yet unseen, and possibly defense-aware worm behavior. Our scheme employs model checking to produce worm propagation sequences that defeat a worm defense of interest. We demonstrate this approach using an exemplar collaborative worm defense, in which LANs share alerts about encountered infections. Through model checking experiments, we then generate propagation sequences that are able to infect the whole population in the modeled network. We discuss these experimental results and also identify open problems in applying formal methods more generally in the context of worm quarantine research.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*

General Terms

Security, Design

Keywords

Model checking, malicious code, worm defense, network security

1. INTRODUCTION

While the Internet continues to foster technological progress, this ever-connected world also bears the risk of self-propagating code causing computer epidemics. Researchers are developing countermeasures in response to this threat, by both examining the algorithms of real-world worms and performing experimentation using captured worm traffic traces. However, these approaches limit experimentation to anticipated or previously seen worm strategies. To study worm defenses in light of yet unseen worm behavior, we advocate reasoning about fundamental properties of worm propagation and quarantine in a formal framework.

As a starting point to this grander vision, we focus on the potential of model checking as a means to produce worm propagation sequences against a modeled worm defense without considering any specific worm strategy. As an example, we model a collaborative defense scheme and run the model checker through a range of varying parameters. We evaluate generated propagation sequences that defeat the defense and are able to infect the whole population in the modeled network. As a human interpreter of these sequences, we hypothesize about underlying worm intent that would exhibit behavior matching the observations. We also discuss cases in which the model checker proves that the defense successfully saves at least one of the population. This assertion holds over all possible worm propagation sequences.

We suggest that this approach is applicable both for developing more challenging test cases when evaluating many contemporary worm defenses and for understanding the degree to which more customized worm propagation sequences can be designed to circumvent (or optimally tolerate) a given worm defense strategy. In addition, we enumerate several open research problems that we have identified during our initial study, which we deem important to overcome to facilitate greater use of formal methods in worm defense analysis.

2. RESEARCH IN DYNAMIC WORM QUARANTINE STRATEGIES

The increasing speed and sophistication of worm epidemics has offered strong motivation to develop *quarantine* techniques that can detect and react to worm outbreaks as they propagate through a community of networks. Detection may be one step in the quarantine process, followed by techniques to corroborate an attack pattern and institute a containment strategy. Throughout the quarantine process there remain significant research challenges in all aspects of detection, response coordination within a narrow *reaction time* [11], and containment strategies, such as port-based filtering, IP blacklisting, or shared dynamic content filters [5].

Over the last few years researchers have proposed several collaborative quarantine schemes, which they argue are effective in curbing the infection rates of certain classes of large-scale worm epidemics. GrIDS [19] is an early example of a hierarchical detection scheme to identify coordinated worm activity, with an overlay sensor network that reports connection anomalies up through a distributed management structure. Nojiri et al. [12] propose a cooperative alert sharing scheme using a *friends protocol* under which each egress router within an enterprise selects a static peer group with which to share worm indicators, and in turn is also selected by other domains to receive reports. When an egress router receives a certain number of alerts over a temporal window, it enters a filtering posture to block inbound worm traffic. Anagnostakis et al. [2] propose a variant scheme called COVERAGE, in which each egress router randomly selects a set of remote nodes to poll for worm reports at periodic intervals. Briesemeister and Porras [3], propose a combined rate limiting and group sharing protocol, whereby a connec-

tion rate limiting scheme such as [21, 18, 7] is used both as the source of peer alert production, and as a fundamental mechanism through which fast-spreading worms are slowed down enough to allow peer groups an opportunity to corroborate and engage their defenses. Kannan et al. [8] propose a cooperative framework for firewall-level peer cooperation across the Internet, suggesting that even in minimal deployments, the participating firewalls can achieve high degrees of containment under certain worm propagation strategies.

Most of the effort toward evaluating the efficacy of such quarantine schemes has focused on analyzing the impact of these schemes on infection growth rate in the presence of common worm propagation strategies. In particular, researchers study proposed worm defense algorithms in the context of naïve or generic randomly propagating epidemic strategies, or at best attempt to mirror the propagation strategies of previously experienced worms such as Code Red [24] and Slammer [20]. Often, simulation is employed as a cost-efficient way to examine the growth rate impacts of a quarantine algorithm against a modeled epidemic [1, 10, 23]. However, simulation provides little insight into how the defense performs on strategies other than the specific propagation strategy encoded within the simulation. One objective of our present study is to employ model checking to exhaustively examine the behavior of a proposed worm defense across the complete space of all possible infection sequences.

3. COLLABORATIVE DEFENSE AND WORM PROPAGATION

To illustrate our approach, we present an exemplar group-based defense, in which local area networks (LANs) cooperate by alerting other LANs about a worm infection they might encounter. We assume the defense algorithm is integrated into the network infrastructure, typically running on an egress router of a LAN. In our model, the egress router has the capability to switch into a *defensive posture* filtering successfully all incoming and outgoing traffic from the offending worm packets. To arrive at a sufficiently abstract model, we do not specify how to accomplish perfect filtering but we attribute a cost to it and preclude the algorithm from simply staying in the defensive posture. Others are developing automatic signature generation systems such as EarlyBird [17] and Autograph [9], which could be used for filtering in an implementation of our scheme.

In presenting our model, we will henceforth not distinguish anymore a LAN from individual LAN components such as an egress router within the LAN executing the algorithm and end hosts within the LAN that are susceptible to the worm. We refer to all these meanings under the term of a *node*. A node can be infected (i.e., at least one end host within the LAN is infected), can be filtering (i.e., the egress router drops offending packets), and can send alerts to other nodes (i.e., peer-to-peer communication among egress routers).

Next, we describe the group-based worm defense as a distributed algorithm running on a set $\{1..N\}$ of N nodes. The parameter G denotes the group size with $2 \leq G \leq N$. The defense algorithm initially selects the preceding[1] $G-1$ peers to which it will later send alert messages. Thereby we reach optimal coverage over the population of nodes, in that each node becomes a peer of the same number of other nodes. Two parameters $s = 3$ (severity of an incident) and r (corroboration level) define an alert threshold $\alpha = s \cdot r$.

Each node maintains a counter a to denote its current alert level. The value of a is initially set to zero. Upon receiving an alert from another node, a node adds s to the alert level but ensures that a is not more than the maximum α. When the algorithm detects an internal worm infection (e.g., employing outbound rate limiting and an aggressive worm violates the threshold) it sends an alert to all its peers and increments the alert level as if receiving an external alert. If increasing the alert level on either occasion reaches the alert threshold α, the node switches into defensive posture. At the end of a time step, the counter a is decremented by one but not if below zero. If a reaches zero, the node turns off the defensive posture.

In addition, after sending an alert, the algorithm enforces a period of s time steps, in which it remains quiet. A local variable c denotes the hold-off counter used to implement this duration of hold-off. During hold-off time, a node does not react upon detecting a local worm infection. The hold-off mechanism ensures that no single node forces its peer group to enter the defensive posture.

Refer to Figure 1 for a more detailed flow diagram that implements the distributed and collaborative worm defense algorithm sketched out above. As the flow diagram uses a state-transition system notation to describe the algorithm, it is relatively straight-forward to convert it into the SAL language used for model checking. We performed this translation manually but had two people checking the implementation independently. Therefore, the model checking experiments below are based on a faithful implementation of the algorithm presented here.

In our formal model, we combine the state-transition system of the defense algorithm outlined above with a nondeterministic worm propagation model. Per time step, an infected node either chooses another node to which it will attempt to spread or decides to lay dormant, but it is otherwise not limited to any specific propagation strategy. As a result, all choices in the execution of the combined model can be attributed solely to the worm behavior because the transitions of the defense are all deterministic.

4. AN EXPERIMENT TO GENERATE SUCCESSFUL PROPAGATION SEQUENCES

We designed an experiment of using model checking to generate worm propagation sequences. Model checking is a method to algorithmically verify whether a model satisfies a specification. A model checker tool usually takes as input a state-transition system (model) and a temporal logic formula (specification) to either prove the assertion or generate a counterexample.

We exploit the latter to automatically produce propagation sequences that are successful against a modeled worm defense. Given the combined model of defense algorithm and worm propagation in Section 3, we express in a linear temporal logic (LTL) [13] formula (see Property 1) what it means that the worm is successful .

PROPERTY 1 (WORM WINS). *Eventually, all nodes be-*

[1]When reaching zero we start over from N.

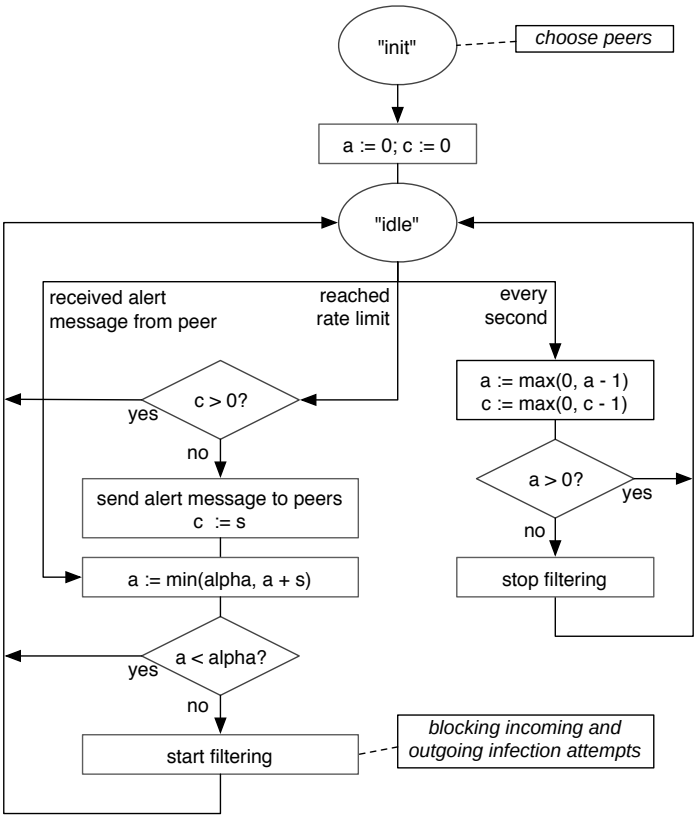

Figure 1: Flow diagram of collaborative defense algorithm running on each node

come infected. Formally,

$$\Diamond(\forall j \in \{1..N\} : \mathit{Infected}[j])$$

To generate a propagation sequence successfully infecting everyone, we run the model checker against the *negation* of Property 1.[2] Then, counterexamples produced by the model checker are guaranteed to exhibit Property 1. In other words, we ask the model checker to prove that our defense will forever protect at least one node, and each counterexample produced represents a winning infection strategy that the defense designer must carefully consider. Again, this result is significantly different than what is achieved through simulation in that simulation explores only a specific propagation strategy (e.g., random scanning) over a limited temporal window.

While we vary the parameter combinations, the model checker does not always produce a counterexample and instead proves the negation of Property 1 in some cases that we discuss first.

For $r = 1$ and $G = 1$, no alerts are generated and it is easy to see that the worm wins as only the routers of infected LANs have a chance to switch into filtering mode.

For $r = 1$ and $G > 1$, the model checker proves that at least one node is always spared. In these cases, the initially infected node sends an alert to at least one other node in the first time step, which forces recipients immediately into filtering ($r = 1$) before they are infected. These alerts repeat with periodicity $s = 3$ before nodes back off, so that they remain filtering from then on. The worm then never has a chance to infect these nodes, and hence no winning strategy for the worm exists.

For $r > 1$, there generally exists a winning strategy for the worm, which the model checker produces as a counterexample. Within the experiments that we were able to complete, only a few peculiar combinations of parameters preclude the worm from winning as seen in Table 1. All experiments were carried out modeling the system in the SAL specification language [14] and employing the SAL model checker [6] on a Linux machine with two Pentium Xeon 3.6 GHz processors with 16 GB of memory. We suspect that for $r \geq 2$, the negation of Property 1 can be proven if G and N are sufficiently large. Unfortunately, the performance limitations of the model checker prohibit the full exploration of the state space corresponding to large parameter values. In future studies, we hope to construct and (manually) prove the general formula for these cases.

Next, we discuss one counterexample obtained from model checking. It exemplifies a strategy that a "smart worm" can take to circumvent the group-based defense. Figure 2 depicts the infection and alert propagation of this counterexample with $N = 10$ nodes and a group size of half the population $G = 5$. The small value of the corroboration level $r = 2$ makes the defense very sensitive to alerts and overprotective—nevertheless, model checking reveals a propagation sequence to infect the entire network before any node can transition into its defensive posture.

In the first three time steps, the worm spreads to loca-

[2]Negation of Property 1: *It is always the case that there exists an uninfected node.*

Table 1: Model checking results for partial parameter space

$r =$		2									3								
$N =$		2	3	4	5	6	7	8	9	10	2	3	4	5	6	7	8	9	10
$G =$	2	x	x	x	x	x	x	x	x	x									
	3		x	x	x	x	x	x	x	?		x	x	x	x	x	x	?	?
	4			x	x	x	x	x	x	x			x	x	x	x	x	?	?
	5				x	x	x	x	x	x				x	x	x	x	?	?
	6					p	x	x	-	-					x	x	x	?	?
	7						p	p	x	-						x	x	x	?
	8							p	p	p							x	x	?
	9								p	-								p	?
	10									p									p

(p: proved, x: counterexample, -:out-of-memory, ?: run not finished, gray: parameter combination not applicable)

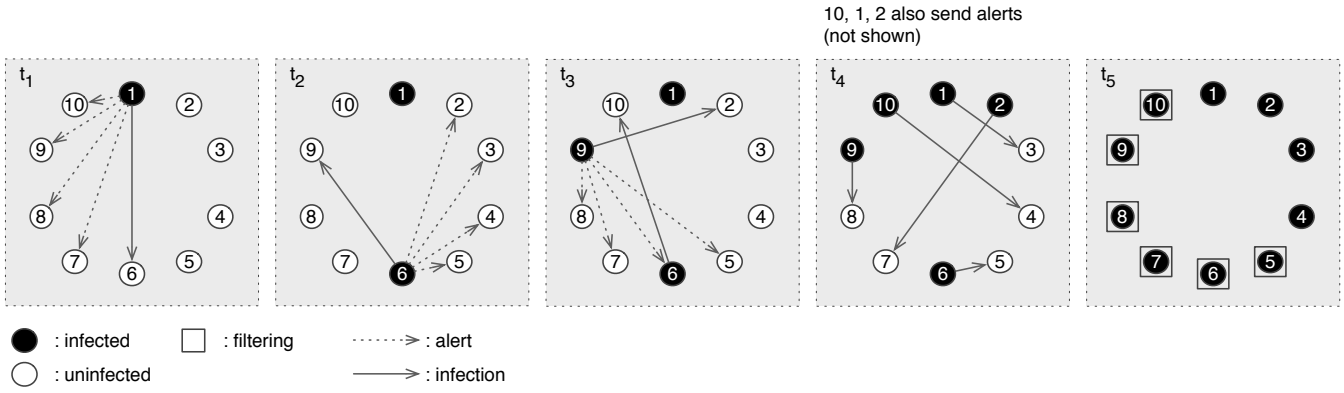

Figure 2: Successful infection propagation sequence for $r = 2$ and $G = 5$

tions that do not overlap with the peers receiving alerts at the same time (since $r = 2$, a coinciding infection and alert reception causes immediate filtering). A second interesting observation is node 1 selecting node 6 as the first infection target. Choosing node 6 as its first victim has the important implication that neither the alerts of node 1 nor the subsequent alerts from node 6 at t_2 overlap with each other. Note also that the worm in node 1 lies dormant in t_2 and t_3. This initial infection sequence prevents overall alert levels from rising too fast and pushing any node into defensive posture. At t_4, when half of the population is infected but none of these are filtering, the worm launches a concerted attack against the remaining uninfected population, where every infected node targets a different susceptible node.

Essentially, this worm operates in two phases: strategic placement of worms and possible periods of dormancy such that infections avoid group overlap, followed by a coordinated overrun of the remaining nodes. While the attack seems to require sophisticated coordination by the worm, this strategy is both generalizable and can be efficiently precalculated, assuming the worm acquires access to the preselected group map. Each worm instantiation is subsequently handed an "infection schedule" of which nodes it is allowed to infect and at what time interval it is allowed to attack. The generated propagation sequences suggest the danger of static preselection of the group structure. One solution is to periodically and randomly alter the set of peers to prevent worms from reliably avoiding group overlap. COVERAGE [2] is an example of a quarantine strategy that randomly recomputes the group to share alerts at periodic intervals.

5. FORMAL METHODS IN WORM DEFENSE ANALYSIS

Formal modeling and model checking early in the design stage of a quarantine algorithm can help identify infection propagation sequences for which the algorithm or its parameters are poorly suited to address. In Section 4, we discover a strategy from one such propagation sequence that we call the "preplacement and overrun" strategy. The example illustrates a threat that can be both generalized and scaled up to defeat cooperative defenses that employ static peer groups.

Model checking, unlike simulation and operational testing, allows one to assert or refute which desired protection properties and other design expectations hold over the entire space of infection propagation within a modeled network. While testing and simulation provide good insight into the potential impact of a worm defense on the growth rates of certain classes of worm epidemics, they are very limited in their ability to assert behavioral characteristics of the worm defense beyond their observation time horizon or to validate the efficacy of the defense beyond the scope of the specific worm infection strategies that are simulated or tested. In [4], we present the results of an effort to use model checking to answer questions regarding algorithm correctness in an exemplar cooperative quarantine defense and validate or refute various quarantine properties against a fully nondeterministic, exponentially growing worm infection.

However, formal methods and model checking also have significant limitations and open research challenges that may limit their broader use in evaluating contemporary worm defenses. Readily available and standard model checking used with our state-transition system does not support simultaneous nondeterministic behavior among both the worm propagation and the worm defense. Nondeterministic defense and attack models dramatically increase state space size and make it impossible to attribute any counterexample discovered by the model checker as being a result of an intelligent worm infection sequence rather than poor defensive behavior. To solve this problem we are exploring the development of customized search algorithms that guide the model checker in constructing the counterexample. Solving this problem is critical as randomized algorithms offer a valuable defense against worms that seek to exploit threshold values or circumvent group structures.

We are also developing model checker extensions to generate more than one counterexample from one parameterized model and specification to collect more data on successful infection propagation sequences. Sheyner and Wing's work [16, 15, 22] attempts such an endeavor in a similar context. In our situation, we expect the number of counterexamples to be infinite and thus need to conceive methods of bounding and scoping in a meaningful way. Having a collection of observed infection propagation sequences, one needs to further consolidate them by detecting isomorphisms and other redundancies.

Another key challenge is to methodologically infer worm behavior from the observations. In this initial study, we applied manual and ad hoc analysis to draw conclusions about underlying worm strategies that would exhibit behavior matching the counterexamples. For future use, we plan to investigate whether existing methods of meta-programming apply, for example, constructing decision trees or hidden Markov models by using observations as training data.

When applying model checking to worm propagation and quarantine protocols, we recognize a rather high level of abstraction. The semantics of the network infrastructure, such as firewalls, routers, and end hosts, are significantly abstracted, as are the influences of traffic dynamics, congestion, quarantine message volume, or other network characteristics that affect worm propagation speed. Future studies will address means of translating between the abstraction levels and provide mechanisms to apply results back to the real world.

Finally, model checking as a means of exhaustive search in an exponentially growing state space suffers naturally from issues of scale. Based on the vast literature on model checking in general, where this problem repeatedly arises in other application contexts, we hope to arrive at larger-scale systems by exploiting homogeneity in the system and human-guided customization of the system model. Also, one may be able to prove hypotheses derived from smaller-scale, toy-like network models with the aid of theorem provers.

6. CONCLUSION

We present an initial study of applying model checking in the context of dynamic worm quarantine. Contrary to many simulation and evaluation studies, our formal model does not assume any specific worm propagation strategy but rather encompasses all possible infection sequences. Using a linear temporal logic formula as the specification for the model checker, we automatically generate infection propagation sequences that successfully penetrate the whole modeled network.

We evaluate different runs of the model checker with varying parameters of the system model. Studying one exemplar propagation sequence leads to an interpretation of how the worm defense algorithm can be improved to preclude similar infection patterns from succeeding in the future.

During the course of our model checking experiments, we identified various open problems in applying formal methods more generally in the context of worm quarantine research. We discuss these key challenges and give a few pointers to potential remedies. Overall, we hope to motivate the practice of formal methods as an alternative and complementary approach to the daunting problem of worms infesting computer networks.

7. ACKNOWLEDGMENTS

We like to thank Ashish Tiwari for his help in developing the SAL model and suggesting interesting properties for model checking. This work was partly supported by the National Science Foundation under Grant No. ANI-0335299, through a subcontract with the University of California at Davis, Contract No. 01RA0052.

8. REFERENCES

[1] M. Abdelhafez and G. Riley. Evaluation of worm containment algorithms and their effect on legitimate traffic. In *Third IEEE International Workshop on Information Assurance (IWIA)*, March 2005.

[2] 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.

[3] L. Briesemeister and P. Porras. Microscopic simulation of a group defense strategy. In *Proceedings of Workshop on Principles of Advanced and Distributed Simulation (PADS)*, pages 254–261, June 2005.

[4] L. Briesemeister, P. A. Porras, and A. Tiwari. Model checking of worm quarantine and counter-quarantine under a group defense. Technical Report SRI-CSL-05-03, SRI International, Computer Science Laboratory, October 2005.

[5] M. Cai, K. Hwang, Y.-K. Kwok, S. Song, and Y. Chen. Collaborative Internet worm containment. *IEEE Security and Privacy Magazine*, 3(3):25–33, May/June 2005.

[6] L. de Moura, S. Owre, H. Rueß, J. Rushby, N. Shankar, M. Sorea, and A. Tiwari. SAL 2. In R. Alur and D. Peled, editors, *Computer-Aided Verification, CAV*, volume 3114 of *LNCS*, pages 496–500. Springer, July 2004.

[7] 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.

[8] J. Kannan, L. Subramanian, I. Stoica, and R. H. Katz. Analyzing cooperative containment of fast scanning worms. In *Proceedings of USENIX Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI)*, pages 17–23, July 2005.

[9] H.-A. Kim and B. Karp. Autograph: Toward automated, distributed worm signature detection. In *USENIX Security Symposium*, pages 271–286, 2004.

[10] 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.

[11] D. Moore, C. Shannon, G. Voelker, and S. Savage. Internet quarantine: Requirements for containing self-propagating code. In *Proceedings of the 2003 IEEE Infocom Conference (INFOCOM)*, April 2003.

[12] 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.

[13] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science*, pages 46–67, 1977.

[14] The SAL intermediate language, 2003. Computer Science Laboratory, SRI International, Menlo Park, CA. `http://sal.csl.sri.com/`.

[15] O. Sheyner. *Scenario Graphs and Attack Graphs*. PhD thesis, Carnegie Mellon University, April 2004.

[16] O. Sheyner and J. Wing. Tools for generating and analyzing attack graphs. In *Proceedings of Formal Methods for Components and Objects*, Lecture Notes in Computer Science, pages 344–371, 2004.

[17] S. Singh, C. Estan, G. Varghese, and S. Savage. The EarlyBird system for realtime detection of unknown worms. Technical Report CS2003-0761, UC San Diego, August 2003.

[18] S. Staniford. Containment of scanning worms in enterprise networks. *Journal of Computer Security*, to appear.

[19] 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.

[20] N. Weaver, I. Hamadeh, G. Kesidis, and V. Paxson. Preliminary results using scale-down to explore worm dynamics. In *Proceedings of the 2004 ACM Workshop on Rapid Malcode (WORM)*, pages 65–72, New York, NY, USA, 2004. ACM Press.

[21] 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.

[22] J. M. Wing. Scenario graphs applied to security. In *Proceedings of Verification of Infinite State Systems with Applications to Security (VISSAS)*, Timisoara, Romania, March 2005. Summary paper.

[23] Y.-K. Zhang, F.-W. Wang, Y.-Q. Zhang, and J.-F. Ma. Worm propagation modeling and analysis based on quarantine. In *Proceedings of the 3rd International Conference on Information Security (InfoSecu)*, pages 69–75, 2004.

[24] C. C. Zou, W. Gong, and D. Towsley. Code red worm propagation modeling and analysis. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS)*, pages 138–147, New York, NY, USA, 2002. ACM Press.