Formally Specifying Design Goals of Worm Defense Strategies

EXTENDED ABSTRACT

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

1 Introduction

There are many key challenges to developing the apparatus and methodologies necessary to evaluate the emerging suite of approaches to large-scale worm defense. Within the DETER/EMIST initiative [3], challenges that have arisen during the development of our experimental framework include the need to support experiment repeatability [17], greater scalability in network topology [16, 8], and greater realism in traffic dynamics [1]. Among these key challenges, we also seek to expand the rigor with which we model the protection claims of the worm defense algorithm, particularly as we design tests that we hope can fully stress and evaluate the protection claims of the algorithm of interest.

To date, most of the work in understanding the behavior of malicious code propagation and defense has centered exclusively on understanding the effects of a proposed malware countermeasure on the global infection growth rate given a specific modeled network and malicious code scenario. In this study we consider how to more rigorously express design goals regarding the local impact of a defensive algorithm from the perspective of those who participate in the defense. We contrast this perspective of *local benefit* from what we view as the current tradition of evaluating worm defense performance based on assessing growth rate impact on an abstracted topology of global population.

Current worm defense performance analyses often provide little insight into understanding the potential negative impacts of a defensive strategy on the local network. For example, two worm defense strategies that are evaluated against a worm that operates using a particular propagation strategy and speed may very well be found to equally reduce the global infection growth rate on a given network. However, in this work we consider performance concerns such as whether one defense may disrupt the communication ability of noninfected systems more than the other. It may also be the case that while both perform equivalently on a given infection sequence, one may be more susceptible to circumvention by worms that employ specially crafted infection sequences. The question of finding stressful infection sequence cases given a specific worm defense algorithm is a critical problem, and using model checking to search for such sequences in a systematic way has, in the small scale, yielded some useful results [5].

We perform an initial exploration toward more formal definitions of the design goals of collaborative worm defense algorithms such as those to achieve dynamic quarantine. We believe that our analytical modeling approach can inform future simulation and emulation experiments in ways that will lead to more challenging tests of the protection claims of a system under evaluation in the DETER framework. We begin by discussing current approaches to worm evaluation, and briefly survey the design space of current worm defense algorithms. We then discuss the basic definition of quarantine in the context of worm defense algorithms, and suggest more precise definitions of quarantine that can capture increasingly stringent requirements for a worm defense algorithm. We suggest how formalizing such definitions could help analyze a worm defense algorithm and produce insight into designing simulation and emulation experiments that are more targeted to stressing the design goals of a defense algorithm under evaluation.

2 Worm Defense Evaluation

Most of the effort toward evaluating the efficacy of malware defense schemes has focused on analyzing the impact of these schemes on infection growth rate in the presence of common worm propagation strategies [1, 11, 4, 18]. 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

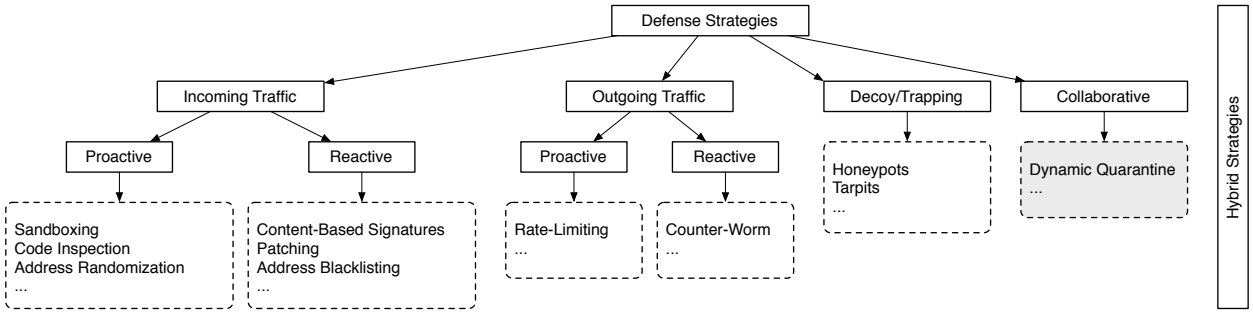

Figure 1: Design space of defense strategies (extended from [7])

such as Code Red [19] and Slammer [12]. Often simulation is employed as a cost-efficient way to examine growth rate impacts of a malware defense against a modeled epidemic. However, simulation provides little insight into how the defense performs on propagation strategies other than the specific propagation strategy modeled within the simulation. Simulation is also an inherently insufficient methodology for asserting algorithm behavior beyond the scope of the simulated time window.

At the other end of the spectrum, operational testing can provide detailed insight regarding the pragmatic issues of worm defense overhead, management, and impact to normal operations, as well as a greater understanding of the duress that the malicious code outbreak causes. However, testing is generally recognized as being an intensive activity to establish and control, particularly in the assessment of defenses that are intended to scale and span over large networks. Wide variability of topology configurations and worm behavior is also expensive to fully explore, and testing is often more effective in answering direct questions about specific environments and test cases than in assessing behavioral properties across a range of conditions. Emulation environments may offer a middle ground for worm defense analysis that is closer to reality than formal modeling or simulations. Among the goals of the DETER emulation environment is to reduce the cost of creating complex worm emulation experiments that can more accurately capture the dynamics of a defense algorithm, while reducing the effort and equipment costs associated with live testing.

3 Understanding the Design Space of Worm Defenses

Before we illustrate how one may approach stating design goals using a particular kind of worm defense—namely collaborative, dynamic quarantine techniques—we focus on understanding the design space first. The design space can encompass many diverse defense strategies. We assume that across the design space, one can employ a similar approach of formally defining design goals using abstractions adapted to the class of defenses.

Inspired by Brumley et al.'s [7] worm defense strategy taxonomy, we present a modified and extended form as depicted in Figure 1. The first level of distinction between defense strategies now encompasses strategies focused on incoming[1] and outgoing[2] traffic but also a category of decoy-based techniques such as honeypots and tarpits, and collaborative defenses such as dynamic quarantine. On the next level remains the distinction between *proactive* and *reactive* strategies. Proactive defenses are not specific to a worm outbreak or a vulnerability in contrast to reactive ones. We provide examples for each type of defense to illustrate our understanding of the categories. In addition to this hierarchy of strategies, we identify hybrid strategies to be orthogonal in the design space.

For the remainder of this paper, we exemplify how to formulate design goals for worm defense evaluation using the class of dynamic quarantine strategies. There have been a number of algorithms that approach malicious code defense by proposing to contain, or *quarantine*, the infected population from the uninfected [13, 2, 4, 10, 9]. The general intuition behind such strategies is that as an epidemic spreads among a collaborative subpopulation in the global network, infection indicators are exchanged among the network population in a manner that may allow members to recognize the epidemic and adjust their security postures appropriately. For example, Dash et al. [9] explore the detection efficacy of such strategies in the presence of slow scanning worms that may propagate at rates below what any single entity might recognize as the epidemic spread. In prior work, we suggest how collaborative strategies might be mixed with techniques that can throttle a propagation enough to increase the potential for corroboration to occur [15, 4].

[1] Formerly "Protection."
[2] Formerly "Local Containment."

4 Design Goals of Quarantine-based Defense Techniques

We use the general class of dynamic quarantine algorithms to illustrate how we may more rigorously understand the protection claims of large-scale network defense algorithms in general. Such understanding is particularly important in the context of DETER/EMIST, as we consider how to more strenuously evaluate a defense in search of its benefits and disadvantages, and as we attempt to fairly assess competitive defense schemes from more than a single dimension.

4.1 Quarantine Definitions

For the sake of our evaluation question, let us assert that the desired outcome of any quarantine scheme is to isolate the infected population from the uninfected, thus slowing, or ideally halting, the infection spread. Let us further assert that the act of quarantining an individual from the community comes at a nontrivial cost, in our case this cost may include both the coordination overhead of implementing the quarantine policy and the loss of otherwise legitimate communications that cannot be performed while the target entity remains under quarantine. We can state a more rigorous definition of the desired worm defense property using a formal language, such as Linear Temporal Logic (LTL) [14]. As a reminder, LTL introduces two temporal operators, $\Diamond$ ("eventually") and $\Box$ ("henceforth"), to the set of usual logic connectives.

For example, given a network of size N and an algorithm that asserts it can detect and quarantine (i.e., completely filter or block targeted communications from) an infected subpopulation within N, we can express this property as follows.

Property 1 (Weak Quarantine)**.** *Eventually every infected member of N is quarantined from N. Formally,*

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

A key term in our definition is the word "eventually," meaning that infected members are not born quarantined, but rather are detected and transitioned to a quarantined state. In a more detailed approach—for example, using Interval Temporal Logic—one could define acceptable time bounds for such a transition. Under current evaluation methods, we strongly consider the question of how many members of N eventually resided in the infected set as well as the rate at which they were added to this set (i.e., infection growth rate analysis of the global network).

However, we must also remember that there are costs associated with quarantining a host. For example, in a competitive evaluation we would most likely prefer an algorithm that avoids quarantining uninfected nodes over an algorithm that appears to minimally discriminate who gets quarantined, even if the latter defense produces a smaller final infection set. Furthermore, Property 1 does not require that there exists any uninfected population at all. We thus call Property 1 the *weak quarantine* property because it is satisfiable by a defense algorithm that quarantines the entire population upon first sign of infection or by an algorithm that waits until all are infected and simply quarantines the corpses.

While weak quarantine is a necessary property to hold among dynamic quarantine algorithms, it does not provide a sufficiently interesting evaluation criterion. Rather, it may be of greater interest to explore the conditions under which a quarantine algorithm provides some degree of benefit to those within a network in which the quarantine defense operates. From the local perspective, benefit would most likely be in the form of increasing the probability that the local site's end nodes avoid infection if the site participates in the defense. That is, a minimally desirable property of a quarantine-based defense would be that it saves at least one member in the population given an infection outbreak. We express this property as follows.

Property 2 (Beneficial Quarantine)**.** *Eventually every infected member of N is quarantined from N and there exists an uninfected member within N. Formally,*

$$\Diamond((\forall j \in \{1..N\} : \mathit{Infected}[j] \Rightarrow \mathit{Quarantined}[j]) \\ \wedge\, (\exists k \in \{1..N\} : \neg\, \mathit{Infected}[k]))$$

This property captures a more desirable end result in that an algorithm that can satisfy this property for a given epidemic can spare at least one member of the network the cost of recovering from the malware infection. Unfortunately, here again such a property could be satisfied by a quarantine defense that simply imposes universal quarantine on all members of the network, as long as at least one member is quarantined before transitioning to the infected state. In such a case, one might view the cure as severe as the disease.

One direct way to strengthen our expression of a quarantine-based defense is to add to Property 2 the requirement that at a minimum, the uninfected node must not be in the quarantined state. This has the effect of eliminating algorithms that impose universal quarantine to all members of N regardless of their infection status, and ensures an increase in the probability that a member of N will be saved from infection should the defense algorithm be imposed. An algorithm that can satisfy this additional

requirement for a given network of size N in the presence of a specific epidemic is said to provide strong local benefit, which we can express as follows.

Property 3 (Strong Beneficial Quarantine)**.** *Eventually every infected member of N is quarantined from N and there exists an uninfected and not filtering member within N. Formally,*

$$\Diamond((\forall j \in \{1..N\} : \mathit{Infected}[j] \Rightarrow \mathit{Quarantined}[j]) \\ \wedge (\exists k \in \{1..N\} : \neg \mathit{Infected}[k] \wedge \neg \mathit{Quarantined}[k]))$$

While we have thus far focused on the more rigorous expression of local benefit in our assessment of the defense strategy, we have not addressed concerns regarding how long such a benefit should last. For example, Nojiri et al. [13] propose the use of temporal decay functions that allow a local site to automatically remove the defensive posture after some interval of time. One concern is that while an algorithm may succeed in satisfying Property 3 during at least one point in the modeled epidemic, its value may be greatly reduced if the algorithm subsequently enters a cycle of transitioning members of N in and out of a quarantined posture that eventually allows the epidemic to saturate all of N. We can express a design goal that the achievement of strong beneficial quarantine should hold over time by applying the "always" LTL operator to Property 3 as follows.

Property 4 (Strong Permanent Quarantine)**.** *Eventually every infected member of N is quarantined from N, and henceforth there exists an uninfected and not quarantined member within N. Formally,*

$$\Diamond((\forall j \in \{1..N\} : \mathit{Infected}[j] \Rightarrow \mathit{Quarantined}[j]) \\ \wedge \Box(\exists k \in \{1..N\} : \neg \mathit{Infected}[k] \wedge \neg \mathit{Quarantined}[k]))$$

One way for an algorithm to achieve permanent quarantine is for it not to allow automated transitions out of the quarantined state, which one may view as too high a cost to be of practical value. Alternatively, some epidemic and detection models may enable situations in which the infected and quarantined members of N produce a continuing flow of alarm signals that preserve the quarantine posture for the life of the epidemic. In either case, one practical concern in evaluating permanence properties is that they are not easily assessed by simulation, emulation, or testing, as these techniques can assert the behavior of algorithms only during their finite analysis windows.

4.2 Design-Time Evaluation of Rigorous Functional Claims

The ability to formally express the functional expectations of our algorithm not only provides vital input in helping to enumerate applicable test cases and evaluation metrics within the DETER evaluation framework, but these properties can be assessed very early in the algorithm design stage. For example, one can employ model checking of an algorithm design to search for specific input streams, such as a worm infection sequence, that lead to a contradiction in a desired protection property. Unlike simulation and emulation, model checking allows one to assert or refute which protection properties and other design expectations hold over the entire infection sequence space, at least within the confines of a small-scale network model. In [6], we present the results of an effort to use model checking to evaluate various protection claims within one exemplar quarantine-based defense, both formally validating and refuting various properties, including the permanence properties that are outside the scope of simulation and emulation, against a fully nondeterministic, exponentially growing worm infection.

Another considerable benefit to applying model checking early in the design of malware defenses is the ability to utilize the counterexamples produced during proof contradiction to generate evaluation test cases. In the context of model checking worm defenses, a worm infection sequence that contradicts a quarantine property may reveal a worm propagation strategy, which if exposed to the defense within an operational setting could bypass the defense's efforts to contain the worm. We speculate that we can eventually develop a future modeling system that, given a specific worm defense strategy, can explore the full space of potential infection sequences to identify an optimal sequence that will circumvent or at least stress the protection claims of a defense algorithm under evaluation. In [5] we demonstrate this idea by employing model checking to generate infection sequences that violate a formally stated quarantine property of a modeled quarantine-based defense. While the implications of such a modeling system are quite concerning, as it may result in future tendencies to limit the open sharing of a deployed worm defense design to avoid maliciously intended adversary modeling, we believe that overall the ability to systematically search for test sequences to fully stress the protection claims of a defense algorithm can benefit the defense to a greater degree than the misuse of such techniques.

5 Conclusion

As the DETER/EMIST program progresses in its development of an evaluation framework to examine the protection properties of large-scale network defenses, one need that arises is that of developing methods to express

just what the evaluatable protection properties are for a given defense algorithm. In this extended abstract, we observe that in the case of worm defense systems, algorithm evaluation is typically centered on measuring the impact that the defense has on the global network infection rate. While infection rate reduction is clearly a critical metric, we suggest that there are other dimensions to evaluating the characteristics of competing defense algorithms.

We illustrate one potential direction in enumerating key protection properties of interest in malware defense algorithms. To do this we attempt to define a general protection property of quarantine-based defense, and observe that we can increasingly strengthen the property to eliminate unwanted defense behavior. For example, we can extend a basic notion of quarantine to include the requirement to ensure an increase in the probability of avoiding both infection and quarantine. We can express the notion of persistent protection, though such properties would not be evaluatable using current simulation and emulation techniques. We also discuss a related study that employs model checking early in the design stage to both formally validate or refute desired protection properties, and could be useful for informing the generation of test case scenarios within the DETER evaluation framework.

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] R. Bajcsy, T. Benzel, M. Bishop, B. Braden, C. Brodley, S. Fahmy, S. Floyd, W. Hardaker, A. Joseph, G. Kesidis, K. Levitt, B. Lindell, P. Liu, D. Miller, R. Mundy, C. Neuman, R. Ostrenga, V. Paxson, P. Porras, C. Rosenberg, J. D. Tygar, S. Sastry, D. Sterne, and S. F. Wu. Cyber defense technology networking and evaluation. *Communications of the ACM*, 47(3):58–61, 2004.

[4] 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.

[5] L. Briesemeister and P. A. Porras. Automatically deducing propagation sequences that circumvent a collaborative worm defense. In *Proceedings of the 25th International Performance Computing and Communications Conference (Workshop on Malware)*, pages 587–592, April 2006.

[6] 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.

[7] D. Brumley, L.-H. Liu, P. Poosankam, and D. Song. Design space and analysis of worm defense strategies. In *Proceedings of the 2006 ACM Symposium on Information, Computer, and Communication Security (ASIACCS 2006)*, March 2006.

[8] S. Cheetancheri, D. Ma, T. Heberlein, and K. Levitt. Towards an infrastructure for worm defense evaluation. In *Proceedings of the 25th International Performance Computing and Communications Conference (Workshop on Malware)*, pages 559–566, April 2006.

[9] D. Dash, B. Kveton, J. M. Agosta, E. Schooler, J. Chandrashekar, A. Bachrach, and A. Newman. When gossip is good: Distributed probabilistic inference for detection of slow network intrusions. In *Proceedings of the 21st National Conference on Artificial Intelligence*, July 2006. (To appear.).

[10] 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.

[11] 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.

[12] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the Slammer worm. *IEEE Security and Privacy*, 1(4):33–39, July–August 2003.

[13] 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.

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

[15] 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, 2004.

[16] 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, 2004.

[17] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI)*, pages 255–270. USENIX Association, December 2002.

[18] 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.

[19] 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, 2002.