Notes On The Design Of An Internet Adversary

David S. H. Rosenthal[1], Petros Maniatis[2], Mema Roussopoulos[3], TJ Giuli[4],
and Mary Baker[5]

[1] Stanford University Libraries, Stanford, CA,
[2] Intel Research, Berkeley, CA,
[3] Computer Science, Harvard University, Cambridge, MA,
[4] Computer Science Department, Stanford University, Stanford, CA,
[5] HP Labs, Palo Alto, CA

Abstract. The design of the defenses Internet systems can deploy against attack, especially adaptive and resilient defenses, must start from a realistic model of the threat. This requires an assessment of the capabilities of the adversary. The design typically evolves through a process of simulating both the system and the adversary. This requires the design and implementation of a simulated adversary based on the capability assessment. Consensus on the capabilities of a suitable adversary is not evident. Part of the recent redesign of the protocol used by peers in the LOCKSS digital preservation system included a conservative assessment of the adversary's capabilities. We present our assessment and the implications we drew from it as a step towards a reusable adversary specification.

1 Introduction

The LOCKSS[6] (Lots Of Copies Keep Stuff Safe) program has developed and deployed test versions of a system for preserving access to academic journals published on the Web. The fundamental problem for any digital preservation system is that it must be affordable for the long term. To reduce the cost of ownership, the LOCKSS system uses generic PC hardware, open source software, and peer-to-peer technology. It is packaged as a "network appliance," a single-function box that can be connected to the Internet, configured and left alone to do its job with minimal monitoring or administration. The system has been under test at about 50 libraries worldwide since 2000.

Like other Internet hosts, these appliances are continually subject to attack. Although measures [31] have been taken to render the operating system platform resistant to attack, its compromise must be anticipated. The appliances cooperate with each other to detect and repair damage in a peer-to-peer network. The first version [32] of this protocol turned out to be vulnerable to various attacks. We recently redesigned the protocol [24] to make it more resistant to attack.

The redesign needed as input an assessment of the capabilities and strategies of the potential adversaries, but we were unable to find this information off-the-shelf. We present our assessment, and the implications we drew from it, as a contribution to an eventual reusable adversary specification.

---

[6] LOCKSS is a trademark of Stanford University.

2 Adversary Assessment

Military intelligence seeks to develop so-called “appreciations” of a potential adversary’s “capabilities” (what the adversary *could* do) and “intentions” (what the adversary is *expected* to attempt with the capabilities) as a basis for planning [5]. Similarly, plans and techniques for defending distributed systems exposed to the Internet need an appreciation of the capabilities and intentions of the adversary they may encounter when deployed.

Our assessment identified the following probable adversary capabilities:

- Unlimited Power
- Unlimited Identities
- Conspiracy
- Eavesdropping and Spoofing
- Exploiting Common Vulnerabilities
- Uncovering Secrets

2.1 Unlimited Power

Techniques [35] have been described by which a worm could compromise a large proportion of vulnerable Internet hosts in a short time. In practice, even much less sophisticated techniques [26] have proven capable of compromising large numbers of hosts quickly, despite widespread knowledge of both the vulnerabilities themselves and their cures for six months prior to the attack [19]. Further:

- Experience with Code Red [27] shows that at least 1/3 of the compromised hosts remain compromised a month after the start of the attack. Two years after the attack a pool of 20,000 infected hosts was still available [13].
- Experience with Slapper [28] shows that 1/3 of vulnerable hosts were still vulnerable 3 months after the vulnerability was announced and 1 month after the start of the attack.
- Experience with a BIND vulnerability [4] shows that a significant proportion of professionally maintained systems are still vulnerable two months after the vulnerability was made public.
- Advertisements are rumored to be appearing that invite spam senders to rent access to a network of 450K compromised hosts they can use to disguise the origin of e-mails.

So far, these networks of compromised hosts have been used to mount crude but effective [18] network-level denial of service attacks. However, it would be a simple matter for the payload of such a worm to be an application-level attack targeted at a particular victim system. If the worm were based on a vulnerability as widespread (350K+ hosts) as the ones Code Red [27] or Blaster (385K+ hosts) [13] exploited, the attacker could expect on the order of 10K machine-years of computation to be available for the attack on the victim system (30% of systems compromised for 1 month, 10% for 3 months). This is, for example, about 35 times the effort used to win the RSA DES Challenge III in 1999 [9].

There is a practical difficulty for the adversary hoping to use these pools of compromised hosts as a resource for attacking a given system. Many other adversaries with other targets are in competition for the resource, which is not infinite although it may be large. This difficulty, however, is not a comfort to the designer of system defenses, whose worst-case analysis must assume that all available resources may be used for a single-minded attack against his system.

2.2 Unlimited Identities

Given the relative ease by which an adversary can compromise and control a large number of hosts across the Internet, we must assume that the adversary can pose as an unlimited number of identities, e.g., IP addresses. The adversary can either directly use the compromised host's IP address or make the compromised host spoof other IP addresses on the same subnet. Even if ingress filtering [7] were turned on in all routers across the Internet, the cost for a host to spoof an IP address on the same subnet is negligible.

There is a practical difficulty for the adversary in that he can only steal identities on subnets in which he maintains a presence, either legitimately or through compromise. This difficulty is not a comfort to the designer of system defenses who must assume that the adversary can have a presence in thousands of subnets spread across the Internet.

The assessment above is not unique to IP addresses. Email addresses, identity certificates, DNS domains are just as easy for an adversary to hoard or spoof or both. Techniques for making this more difficult or time-consuming for an adversary include client puzzles and reverse Turing tests [20], but the adversaries are adapting to them. For example, it is now rumored that reverse Turing tests can be forwarded to a service run by porn sites, which exploit their customers to solve them and return their responses.

2.3 Conspiracy

The Fizzer worm uses IRC [11] to communicate with a central control site. It would be possible for a worm to use peer-to-peer communication techniques instead, avoiding the difficulties the Fizzer worm suffered when its IRC channel was subverted by its enemies [8].

It has to be assumed, therefore, that all the adversary's identities mask a single distributed adversary with instantaneous self-awareness. Any state, such as messages sent, received, or observed by one identity acting on behalf of the adversary is immediately made available to all other identities.

In addition, it must be assumed that some apparently benign identities are conspiring with the adversary. Anything known to these "spies," including supposed secrets such as session keys, is known to the adversary.

It is practically difficult for the adversary to distribute information rapidly and completely among the components of a distributed system with as many nodes as there are compromised hosts. This difficulty is not a comfort for the designer of system defenses, who must assume that the adversary can succeed in getting the critical information to the nodes that need it.

2.4 Eavesdropping and Spoofing

A single compromised host on a subnet can eavesdrop on traffic to and from all hosts on the same subnet. It can also send spoofed messages on behalf of the co-located hosts, as well as send messages with spoofed source addresses from anywhere in the Internet to co-located hosts. By doing so it can often abuse trust relationships mediated by IP addresses. This behavior is very difficult to detect and prevent when compromised hosts are not regularly monitored and maintained.

2.5 Common Vulnerabilities

Even if the design of the system's defenses is perfect, the designer cannot assume that their implementation is as perfect. It is likely that, at some point, an exploitable implementation vulnerability will be discovered. A well-designed flash worm exploiting it can compromise the vast majority of the vulnerable hosts in a very short time.

In different contexts including traditional Byzantine Fault Tolerance [3], Distributed Hash Tables [2] and sampled voting [24] it has been shown that systems with more than about 1/3 faulty or malign peers cannot survive for long. Given this, even in fault-tolerant systems, peers need to be assigned at random one of at least four independent implementations if the system is to survive the discovery of an implementation vulnerability. Rodrigues et al. [30] describe a framework within which independent implementations can be accommodated in a fault-tolerant system.

It is important to note that even a perfectly designed and implemented system cannot avoid vulnerabilities brought about by human operators who are coerced to misbehave. An invulnerable computer system, though unimaginably hard to build, is certainly easier to imagine than an incorruptible human.

2.6 Uncovering Secrets

Most systems rely on secret-based encryption systems to preserve system integrity. The assumption is that the adversary does not know and cannot in a timely fashion obtain any of the secrets.

This is not a robust assumption. A recent survey [29] purported to show that the vast majority of commuters at a London station would reveal their passwords if offered a ball-point pen. The adversary may conspire with an insider, he may be the beneficiary of lax security by insiders such as poor password choice [12], he may steal authentication tokens, and, given the resources we assume, he may even use brute-force techniques to break the encryption.

System designers should not treat encryption as a panacea [33]. An individual analysis is needed of the consequences of compromise of each key in the system, if only to assess the precautions appropriate for its protection.

3 Intentions

We have presented an assessment of some of the putative adversary's capabilities. We must now assess his possible intentions. What might the adversary be intending to achieve by exploiting these capabilities?

Our initial attempt classifies possible adversary intentions into five classes: Stealth, Nuisance, Attrition, Thief, and Spy.

3.1 Stealth

The *Stealth* adversary's goal is to damage the system by affecting its state. A necessary sub-goal is to avoid detection before the damage is complete, for example to dodge an intrusion detection system.

3.2 Nuisance

The *Nuisance* adversary's goal is to discredit the system by continually raising intrusion alarms. There is no intention to cause any actual damage to the system or prevent it from functioning. An attack from the Nuisance adversary might, for example, be intended to get the victim's system administrators to disable or ignore the intrusion alarms as a prelude to other forms of attack.

3.3 Attrition

The *Attrition* adversary's goal is to prevent the system from functioning for long enough to inflict damage on the organization it supports. Some forms of the adversary are referred to as "Denial of Service," but this has come to mean a technique rather than a goal.

The Blaster worm was an Attrition attack, attempting to mount a flooding attack on a Microsoft website from its 385K infected hosts. The MiMail virus is an Attrition attack against a set of anti-spam services [18].

3.4 Thief

The goal of the *Thief* adversary is to steal services provided by the system (possibly over long time periods) or steal valuable information protected by the system. The Thief is different from the Stealth adversary in that he does not necessarily want to alter the state of the system, nor does he want to bring the system down or subvert it. The Thief of services wants unauthorized access to resources for as long as possible without being detected. The Thief of information hopes that his intrusion remains undetected for as long as possible.

The Sobig series of viruses [36] is believed to be a Thief who steals services from victim machines by using them as a spam-sending network. It is also thought to be used to mount Attrition attacks on anti-spam services [17].

3.5 Spy

The *Spy* adversary's goal is to observe as much about the system as possible: who participates, where users are located, and what transactions take place. The Spy could be a powerful corporation wanting to harass or prosecute users. The Spy could also be a government collecting information on the on-line activities of its citizens.

4 Rules Of Thumb

We summarize these assessments with some conservative "rules of thumb." The assumptions underlying them are a worm infecting three times as many hosts as Code Red, with the bulk of the infection lasting four days, and 10% still infected after three months. The adversary can:

- exert bursts of computational effort lasting 100 hours and using 1,000,000 hosts,
- sustain computational effort over 100 days using 100,000 hosts,
- masquerade behind 1,000,000 IP addresses,
- eavesdrop on and spoof traffic from 10% of the hosts in the victim system for 100 days.
- break 100 well-chosen DES keys.

5 Implications

Our adversary is very powerful, posing a number of important implications. First, it is economically infeasible to test, or even simulate, attacks of this scale. Assurance that a system does not fail under expected attacks is not likely to be available or credible. Design should focus on:

- Graceful, or at least survivable, failure.
- Assisting diagnosis, perhaps by using bimodal behaviors [1] to raise alarms.
- Assisting recovery.

Second, the adversary can mount a full-scale attack with no warning. Rate-limiting techniques [32, 34, 37] are important in slowing the rate of failure enough to allow for human intervention before failure is total.

Third, the adversary can appear as huge numbers of new peers or clients. Limiting the rate at which the system accepts new peers or clients using techniques such as "newcomer pays" [10] may help slow the failure.

6 Related Work

Researchers in many different fields have tackled the task of characterizing malicious adversaries. In this section, we outline only a few of the approaches we have identified in the literature.

- *Cryptography* typically uses game-theoretic analyses to construct sets of "games" resulting in the adversary behavior observed by benign protocol participants, and investigate whether those sets contain games with malign participants.
- *Protocol design* typically uses exhaustive search of the transitive closure of the state space of the protocol without explicitly modeling an adversary's capabilities or intentions. Finite-state analysis takes the same approach in an automated fashion, with some notable successes (see, for example, an automated analysis of authentication protocols [25]).
- *Distributed systems theory* typically works backwards from a bad state of the system (e.g., a state in which an exploit has been used to damage the system) to identify the sequence of events that must have happened to arrive at that state. The system has to be specified in a suitable formalism (e.g., Lamport's TLA+ [14] or Lynch and Tuttle's Input/Output Automata [21, 22]), but in some cases it is possible to conduct an invariant analysis without a full system specification.
- *Fault tolerance* typically places broad limits on the adversary (e.g., "no more than 1/3 of the nodes can be malign" in the case of byzantine fault tolerance [15]). In other cases, nodes with similar failure modes can be grouped together into distinct equivalence classes with respect to failures (e.g., in Malkhi and Reiter's work on quorum systems [23]). These can be loosely considered an adversary model.

Previous work on defending systems against attack classifies adversaries as either "computationally bounded or unbounded" and considers the time interval over which the adversary collects or modifies state [6]. Although the pool of vulnerable machines on which an adversary can draw is in fact limited, it is large enough and the repair rates low enough that the adversary may be considered effectively unbounded in effort and time.

RFC3607 [16] describes how a worm payload can be used for cryptanalysis, and identifies the first such payload observed in the wild.

7 Conclusion

We have presented what we believe is a conservative assessment of the putative adversary the designers of defenses for an Internet system must take into account. This adversary is based on reasonable extrapolations from the observed behavior of worms exploiting vulnerabilities in applications and systems that are widely deployed on the Internet, and on the assumption that the payload of such worms might be targeted at the system under consideration. We believe that discussion of this and alternative adversary assessments leading to some consensus as a basis for future designs would be valuable.

Our adversary is powerful enough to pose design, implementation and testing problems well beyond those current technology can solve. It appears that designing systems to survive attacks of this magnitude unimpaired is unlikely to succeed. Further, even if the design appeared to succeed, testing implementations to assure that success was manifest in practice is unlikely to be affordable. A more reasonable goal may be to slow and delay the process of failure under attack to allow for human intervention.

8 Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 9907296, however any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

The LOCKSS program is grateful for support from the National Science Foundation, the Andrew W. Mellon Foundation, Sun Microsystems Laboratories, and Stanford Libraries.

Vicky Reich has made the LOCKSS program possible.

References

1. Kenneth Birman, Mark Hayden, Oznur Ozkasap, Zhen Xiao, Mihai Budiu, and Yaron Minsky. Bimodal Multicast. *ACM Transactions on Computer Systems*, 17(2):41–88, 1999.
2. Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach. Secure Routing for Structured Peer-to-Peer Overlay Networks. In *Proceedings of the 5th Usenix Symposium on Operating Systems Design and Implementation (OSDI 2002)*, pages 299–314, Boston, MA, USA, December 2002.
3. Miguel Castro and Barbara Liskov. Practical Byzantine Fault Tolerance. In *Proceedings of the 3rd Symposium on Operating Systems Design and Implementation*, pages 173–186, New Orleans, LA, USA, February 1999. USENIX Association.
4. CNN. Men & Mice Research on BIND Security. `http://www.cnn.com/2001/TECH/internet/03/02/lax.on.DNS.idg/index.html`.
5. Department of the Army. *AR 530-1 Operations Security.* Headquarters, Washington, DC, USA, October 1985. Available at `http://www.fas.org/irp/doddir/army/ar530-1.htm`.
6. Roger Dingledine. The Free Haven Project: Design and Deployment of an Anonymous Secure Data Haven, Master's thesis, MIT, Jun 2000.
7. Paul Ferguson and Daniel Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing RFC 2267, January 1998.
8. Fizzer Task Force. Press Release 18 May 2003. `http://www.irc-unity.org/`.
9. Electronic Frontier Foundation. RSA Code-Breaking Contest Again Won by Distributed.Net and Electronic Frontier Foundation. `http://www.eff.org/Privacy/Crypto_misc/DESCracker/HTML/19990119_deschallenge3.html`.
10. Eric J. Friedman and Paul Resnick. The Social Costs of Cheap Pseudonyms. *Journal of Economics and Management Strategy*, 10(2):173–199, Summer 2001.
11. Tyrel L. Haveman. About Fizzer. `http://www.debugoutput.com/fizzer/about`.
12. Daniel V. Klein. "Foiling the cracker": A survey of, and improvements to, password security. In *Proc. of the USENIX UNIX Security Workshop*, pages 5–14, Portland, OR, USA, August 1990.

13. Brian Krebs. Microsoft gets help from flaw in worm. `http://www.washingtonpost.com/wp-dyn/articles/A589-2003Aug15.html`, August 2003.
14. Leslie Lamport. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers.* Addison-Wesley, July 2002.
15. Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.
16. Marcus Leech. RFC3607: Chinese Lottery Cryptanalysis Revisited: The Internet as a Codebreaking Tool. `ftp://ftp.rfc-editor.org/in-notes/rfc3607.txt`, September 2003.
17. John Leyden. Sobig linked to DDoS attacks on anti-spam sites. `http://www.theregister.co.uk/content/56/33059.html`, September 2003.
18. Steve Linford. Press Release. `http://www.spamhaus.org/news.lasso?article=13`, November 2003.
19. David Litchfield. Unauthenticated Remote Compromise in MS SQL Server 2000. `http://www.nextgenss.com/advisories/mssql-udp.txt`, July 2002.
20. Luis von Ahn, Manuel Blum, and John Langford. Telling Humans and Computers Apart (automatically). *Communications of the ACM*, 2003. to appear.
21. Nancy Lynch and Mark Tuttle. An Introduction to Input/Output automata. *CWI-Quarterly*, 2(3):219–246, September 1989.
22. Nancy A. Lynch. *Distributed Algorithms.* Morgan Kaufmann, San Francisco, CA, USA, 1996.
23. Dahlia Malkhi and Michael Reiter. Byzantine Quorum Systems. *The Journal of Distributed Computing*, 11(4):203–213, October 1998.
24. Petros Maniatis, Mema Roussopoulos, TJ Giuli, David S. H. Rosenthal, Mary Baker, and Yanto Muliadi. Preserving Peer Replicas By Rate-Limited Sampled Voting. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, pages 44–59, Bolton Landing, NY, USA, October 2003.
25. John C. Mitchell, Mark Mitchell, and Ulrich Stern. Automated Analysis of Cryptographic Protocols Using Murϕ. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 141–153, Oakland, CA, USA, 1997.
26. David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford, and Nicholas Weaver. The Spread of the Sapphire/Slammer Worm. `http://www.caida.org/outreach/papers/2003/sapphire/`.
27. David Moore, Colleen Shannon, and Jeffery Brown. Code-Red: a case study on the spread and victims of an Internet worm. In *Internet Measurement Workshop*, Marseille, France, September 2002.
28. Eric Rescorla. Security Holes... Who cares? In *Proceedings of the 12th USENIX Security Symposium*, pages 75–90, Washington, DC, USA, August 2003.
29. Tim Richardson. Give your password to complete strangers? No problem... `http://www.theregister.co.uk/content/archive/24812.html`, April 2002.
30. Rodrigo Rodrigues, Miguel Castro, and Barbara Liskov. BASE: Using Abstraction to Improve Fault Tolerance. In *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*, pages 15–28, Chateau Lake Louise, Banff, AB, Canada, October 2001.
31. David S. H. Rosenthal. A Digital Preservation Network Appliance Based on OpenBSD. In *Proceedings of BSDcon 2003*, San Mateo, CA, USA, September 2003.
32. David S. H. Rosenthal and Vicky Reich. Permanent Web Publishing. In *Proceedings of the USENIX Annual Technical Conference, Freenix Track*, pages 129–140, San Diego, CA, USA, June 2000.

33. David S. H. Rosenthal and Mark Seiden. Is There An Alternative To Long-Term Secrets? In *Workshop on Adaptive Defense of Computer Networks*, Santa Fe, NM, November 2002.
34. Anil Somayaji and Stephanie Forrest. Automated Response Using System-Call Delays. In *Proceedings of 9th Usenix Security Symposium*, August 2000.
35. Stuart Staniford, Vern Paxson, and Nicholas Weaver. How to 0wn the Internet in Your Spare Time. In *Proceedings of the 11th USENIX Security Symposium*, pages 149–167, San Francisco, CA, USA, August 2002.
36. Bob Sullivan. Sobig spam-virus still spreading. `http://www.msnbc.com/news/931205.asp?cp1=1`, June 2003.
37. Matthew Williamson. Throttling Viruses: Restricting Propagation to Defeat Malicious Mobile Code. In *Proceedings of the 18th Annual Computer Security Applications Conference*, Las Vegas, Nevada, USA, December 2002.