Operations with Degraded Security

Modern systems aren't designed to support some ongoing operations after their security is compromised. Using the ResiliNets model, the authors discuss five strategies for operating in a degraded security environment.

Simson L. Garfinkel and George Dinolt
Naval Postgraduate School

A well-developed literature describes how to use redundancy to build storage and communications systems that are resistant to failure. Most of this work stems from Moore and Shannon's seminal 1956 article about the construction of reliable electric circuits from less reliable relays.[1] Instead of failing, systems with redundancy can enter a degraded mode of operation until the failed component can be replaced. Although such approaches have made modern computers incredibly reliable, attempts to apply these techniques to security have largely failed. Despite the widespread adoption of the so-called *defense-in-depth* strategy, the best metaphor for security remains that of a chain—only as strong as its weakest link.

But why should security engineering fundamentally differ from reliability engineering? We believe it's because modern systems aren't designed to support some form of ongoing operation after their security is compromised. Degraded operation of a RAID system decreases performance, but it doesn't corrupt the computer's file system. In contrast, security degradation from a successful hacker penetration typically results in changes so pervasive that the most cost-effective recovery strategy is to declare the system a total loss, wipe the hard drive, and reinstall the system's software and data. Some people amusingly call this approach "nuke from orbit and reinstall," a reference to the 1979 film *Alien*.

We argue that a critical part of living with insecurity is the ability to continue using systems in a controlled and meaningful manner, even after they've been compromised. We use the term *degraded security* to describe this state. Instead of advocating redundancy and defense in depth, we apply James Sterbenz's ResiliNets (resilient networks) model to describe the tasks of managing a system that might be attacked.[2] Prior related work has applied ResiliNets to denial-of-service (DoS) attacks but not to other security threats (see the "Related Work in Network Resilience and Intrusion Tolerance" sidebar); here, we show that this framework can be used for understanding a range of operations in a security-degraded state.

ResiliNets

Designing for operations in a security-degraded environment makes it easier for users and managers to live with the inherent insecurity that permeates all computer systems. Systems can be designed to function after an attack—for example, the system might use various forms of redundancy to protect data, or it could store multiple copies of critical data or software in fast read-write storage, in slower write-once or write-protected storage, or even on remote systems in a different security domain. If an attack modifies one version, another version might still be available.

ResiliNets envisions a four-part cycle—defend (treat networks with processes designed to increase the probability of an attack's failure), detect (identify service degradation or attack), remediate (halt the immediate damage), and recover (restore the system to normal operation). As Figure 1 shows, two back-

 1540-7993/11/$26.00 © 2011 IEEE COPUBLISHED BY THE IEEE COMPUTER AND RELIABILITY SOCIETIES

Related Work in Network Resilience and Intrusion Tolerance

Literally, an object is resilient if it returns to its previous shape after being deformed. Applied to computer networks, this term typically describes systems that can continue providing service after experiencing challenges such as erroneous equipment reconfiguration, denial-of-service attacks, and node loss.

Unlike traditional challenges to reliability, security attacks can be the result of malicious intent. Although it makes sense to model a system's reliability by assuming the chance of failure and detection for each event is independent, this isn't the case when a system is under attack by an intelligent adversary. When modeling reliability, it's reasonable to assume that any component can fail. When modeling security, it's reasonable to assume that attackers will target the components whose failure will cause the most damage. When modeling reliability, failures aren't likely to be correlated in a manner that will ensure maximum damage or disruption. In the realm of security, attackers frequently correlate their attacks specifically for such purposes.

Michael Fry, Mathias Fischer, and Paul Smith note that challenge identification is a critical aspect of network resilience in the security domain.[1] For example, both a distributed denial-of-service (DDoS) attack and a flash crowd are detected in the same manner, but the two require different remediation and recovery strategies. (A DDoS attack is best remediated through aggressive filtering, whereas flash crowds are better solved through TCP rate limiting.) Thus, attackers have an added incentive to disrupt identification processes to prolong an attack's effect.

Although resiliency and security have been areas of research in mesh and sensor networking for more than a decade, little of this work focuses on operations after security has been compromised. We can find only two exceptions: Hao Yang and colleagues' 2005 work, a proposed architecture in which secret keys are bound to geographic locations,[2] and Arvind Seshadri, Mark Luk, and Adrian Perrig's SAKE protocol, which allows compromised nodes to reestablish their secrecy and authenticity keys, even if the attacker can read and modify the memory of both nodes before the protocol executes.[3]

Others have explored some aspect of operations in a security-degraded environment. Alysson Neves Bessani and colleagues explored a series of architectures for fault and intrusion tolerance and removal, including Crutial (Critical Utility Infrastructural Resilience),[4] which provides for recovery of compromised systems through the use of a subsystem that allows a protected and trusted part of each node to communicate with other nodes in a timely and reliable manner.

US Department of Defense systems generally must have documented plans and procedures to recover from security compromises. These are described in DoD Instruction 8500.2 as part of the "availability" requirements.[5] Deployed systems should have documents that clearly describe the essential functions and prioritized plans for restoration in the event of a security compromise. Despite these requirements, we haven't seen plans that seriously contemplate continued operations of systems with known security compromises. One reason could be that the failure to meet these requirements is rarely evident without extensive analysis. A second reason might be the lack of available and understood technical approaches for meeting these requirements. Cost is yet another factor; meeting these requirements increases costs but provides no apparent benefit to system users.

Finally, the US National Institute of Standards and Technology has provided guidance[6] on developing contingency and recovery plans for US federal government systems in the face of various kinds of failures and attacks. Unfortunately, the details of plans that discuss security measures generally aren't available to the public.

References

1. M. Fry, M. Fischer, and P. Smith, "Challenge Identification for Network Resilience," *65th EURO-NF Conf. Next Generation Internet* (NGI 10), IEEE Press, 2010, pp. 1–8.
2. H. Yang et al., "Toward Resilient Security in Wireless Sensor Networks," *Proc. Int'l Symp. Mobile Ad Hoc Networking and Computing* (MobiHoc 05), ACM Press, 2005, pp. 34–45.
3. A. Seshadri, M. Luk, and A. Perrig, "SAKE: Software Attestation for Key Establishment in Sensor Networks," *Proc. Distributed Computing in Sensor Systems* (DCOSS 08), LNCS 5067, Springer, 2008, pp. 372–385.
4. A.N. Bessani et al., "The Crutial Way of Critical Infrastructure Protection," *IEEE Security & Privacy*, vol. 6, no. 6, 2008, pp. 44–51.
5. Department of Defense Instruction 8500.2, 6 Feb. 2003; www.dtic.mil/whs/directives/corres/pdf/850002p.pdf.
6. NIST Special Publication 800-53, Revision 3, "Recommended Security Controls for Federal Information Systems and Organizations," Computer Security Division, Information Technology Laboratory, Nat'l Inst. of Standards and Technology, May 2010; http://csrc.nist.gov/publications/nistpubs/800-53-Rev3/sp800-53-rev3-final.pdf.

ground processes—diagnose (system operators perform root-cause analysis of observed faults) and refine (evaluation of the previous five steps)—mediate and constantly improve this cycle.[2]

As noted earlier, we define degraded security as a system or network state after a successful attack but before efforts to recover from the attack have been completed. We use this definition in the ResiliNets framework but with an important caveat: whereas much of the previous work in resilience has focused on DoS attacks and appropriate responses, we argue that the primary result of such attacks is to degrade performability[3] or, in security terminology, availability. Here we focus on other security properties.

Some will argue that all computer systems and networks are continually in a state of degraded security,

a claim that isn't useful or probably even entirely true. Although a computer system on a disconnected network that has just been loaded with a widely available operating system might have security vulnerabilities, it isn't in a state of degraded security.

People make decisions based on previous experience and prior knowledge, thus personal experience with malware and prior knowledge of major websites besieged by DoS attacks deeply affect many professionals' approach to cybersecurity. One result, we hypothesize, is that many professionals don't understand the difficulty of detection or recognize a need for remediation that is distinct from recovery. But when using Sterbenz's framework, three discrete operations in a security-degraded environment emerge: detection, remediation, and recovery. Although many references combine remediation and recovery in a single step, they should be distinguished. Remediation describes the way in which normal operations are intentionally altered while security is degraded; recovery encompasses the specific actions and mechanisms that are employed to restore the system to its preattack or corrected state.

Detection doesn't seem like a difficult problem if the prototypical model for a security incident is a distributed DoS (DDoS) attack or website defacement. Increasingly, however, malware is equipped with technical measures that let it evade detection. Such malware is sometimes called the *advanced persistent threat*, and we clearly need much better strategies for detecting it. Once certain types of malware have infected an operating system, the only way to find them is to boot a trusted operating system from a separate medium and search for the malware using an independent scanner. In practice, this is done by booting a desktop or server from a CD-ROM that proceeds to scan the system's hard drive. But whereas several antivirus vendors support offline scanning, it's rare to find organizations that proactively remove their systems from service to scan for malware.

Instead, most organizations handle malware infestations reactively: when they discover malware, they consider it a "best practice" to skip remediation entirely and proceed to recovery; they often can't, or won't, take the time for diagnostics because they want their machines back online as soon as possible. Such protocols are a poor match for sophisticated threats. Cleaning a system destroys critical information that might let an experienced security professional determine if the attack is the result of garden-variety malware or a sophisticated attempt to exfiltrate information or control internal systems. Such procedures might even assist the attacker—for example, the "golden master" CD could be the source of the very malware or vulnerability that the attacker is exploiting.

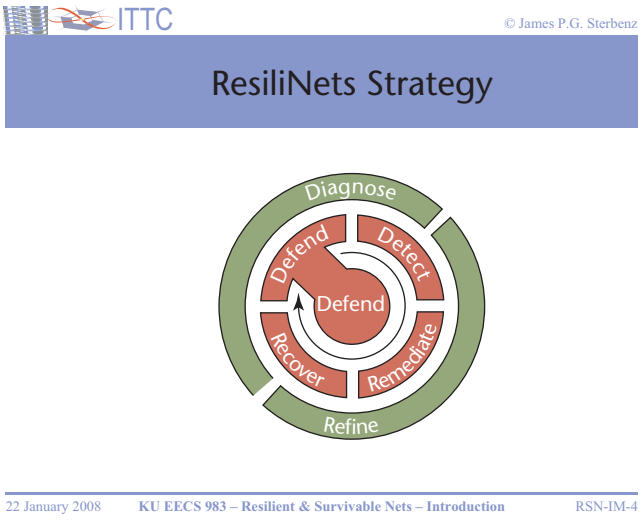

Figure 1. ResiliNets. James Sterbenz's model views resilience as a two-phase process. The inner phase is a never-ending cycle of network defense, detection, remediation, and recovery. The outside phase is aimed at process improvements: diagnosing the root causes of successful attacks and refining the tools and human behavior in use.

For these reasons, distinguishing remediation from recovery is critical. In many cases, it's simply infeasible to immediately recover a system—instead, it must operate for a period of time in its compromised state while other remediation procedures are put in place to minimize the compromised system's impact on the organization as a whole. Continued operation of a compromised system also makes it possible to gather additional information to determine the extent of the attack, just as a police officer might observe a burglar to determine if he or she has accomplices. Such an approach might be vital to supporting the diagnose and refine steps.

Remediation Strategies for Security-Degraded Operations

Different organizations follow different approaches when they suffer security compromises. Our experience is that there's often a significant difference between what organizations think they're doing, what written policies say they should do, and what they actually do. Using the ResiliNets framework, we analyze here five different approaches that we've encountered in recent years. Each represents an approach for living with insecurity. Unfortunately, none of the approaches provides the security guarantees that organizations require due to fundamental shortcomings in today's computing systems.

Strategy 1: Ignorance Is Bliss

Security is at most a secondary concern for most computer users. Ignoring compromises and sim-

ply tolerating the security degradation is a common strategy—assuming that the underlying system maintains sufficient functionality following a successful attack.

This strategy is widespread in homes and many small businesses, but we've seen it in large organizations as well. For example, in early 2011, we were handed a USB storage device while visiting an organization and were surprised to discover it contained a remote-control Trojan known to spread via USB devices. The device's owner insisted that it was clean because it didn't trigger his organization's antivirus software. We uploaded the virus to VirusTotal, discovered that the program was in fact malware, and provided the VirusTotal report to the organization's security staff. We never received a response. Months later, we learned that multiple machines inside the organization had persistent connections to cloud-based command-and-control nodes operated by a criminal enterprise. The organization, it seems, had fallen victim to the advanced persistent threat.

Strategy 2: Response and Recovery

In January 2001, VeriSign mistakenly issued two code-signing digital certificates to a person posing as a Microsoft employee. Microsoft eventually issued a security bulletin describing the situation and an operating system patch that caused Windows to ignore the certificates.[4]

Although Microsoft could have executed an internal remediation strategy to determine the degraded security's impact, we know of no effort at any Microsoft customer site to identify systems that might have been compromised by the certificates. Likewise, we know of no attempts to diagnose the root cause of the attack or refine tools and human response.

The brittle public-key infrastructure (PKI) and certificate authority (CA)-based authentication and integrity systems used by all modern operating systems continue to be significant points of vulnerability. For example, two stolen certificates are credited with the spread of the Stuxnet worm. Today, the typical Internet-accessible computer contains more than a hundred CA certificates, many from organizations that might not be trustworthy. Both root and code-signing certificates have been compromised through a variety of means. Nevertheless, the current trust architecture is such that a single compromised CA or certificate can undermine an entire organization's security.

Strategy 3: Isolate and Treat

An alternative to response and recovery is to isolate the affected components from the rest of the system, similar to medical quarantines of infectious diseases. Once the compromise is detected, the "infected" components don't come into contact with the "healthy" ones.

Isolation works best when that capability is designed into the system from the start. Operation in isolation requires that database systems, Web servers, and other services be removable without bringing down the rest of the enterprise. With appropriate design and configuration, the compromise of one service need not necessarily compromise the entire system.

Isolation isn't without significant costs and risks. For example, during the 1988 Morris worm incident, several organizations disconnected from the Internet or manually shut down their vulnerable Unix-based computers. But those that followed this approach found that they were cut off from the tools that quickly became available to provide testing, analysis, and recovery from the worm. Similar problems happened in 2007 at the Naval War College, when a pervasive malware infestation forced the organization to disconnect its network from the Internet so that each of its Windows-based computers could be properly remediated, significantly hampering the ability of the facility and students to do anything else.

Isolation could be an effective strategy for organizations that need to evaluate an attack's impact, but isolating a compromised system can also provide information to the attacker about detection and data that might need to be protected. In some cases, revealing this information has additional costs and should be avoided.

Eventually, it's time to move from isolation to recovery. Ideally, both can proceed at the same time, with the isolated system remaining operational (but removed) and new hardware being used for the recovery process. Failing this, the compromised system's contents should be archived for analysis at a later point in time. The decision of when to move from isolation to recovery depends on the system, its design, and the environment in which it operates.

Strategy 4: In Situ *Analysis*

Organizations that detect compromised servers might choose to leave the server in place and initiate the covert capture of network traffic for later analysis; they can also monitor file access patterns and take snapshots of the computer's memory and hard drive with special-purpose hardware, software, or a virtual machine monitor. Ultimately, the organization needs to determine the scale of the unfolding incident—just like bedbugs, malware rarely infests a single host.

In situ analysis should be performed according to a detailed malware infestation plan that (hopefully) was created in advance to deal with this situation. Critical tasks should be shifted elsewhere—not just from the infected machine but from the entire subnet or facility where the machine resides. Password-protected ac-

counts accessed from the machine could be proactively locked, but log files should be preserved and flagged for human analysis. The idea here is that identifying the attacker, the damage, and the means of compromise is vitally important but this identification should be done without alerting the attacker, if possible. This is analogous to counterintelligence efforts to unmask a spy.

Attackers might try to hide their activities and subvert monitoring and analysis attempts. For *in situ* analysis to succeed, security professionals need to perform some careful advanced planning using well-designed tools and techniques.

In situ analysis might be most appropriate for organizations that can't reveal that they're aware when they've been successfully attacked. For example, a utility that detects a successful attack against one automated power substation might want to keep this information secret if it has dozens of other substations with similar configurations and vulnerabilities. Likewise, organizations that suspect an insider attack might choose to quietly monitor their activity with the hope of catching that person in the act.

Strategy 5: Hunker Down and Live with It

Sometimes, organizations just have to accept and live with an attack's damage. An analogous situation in the physical world is *battleshort*, in which warfighters bypass safety features to complete a mission, even if doing so risks the destruction of equipment or the loss of life. In the computer security case, the organization will detect the attack or compromise but continue to operate the affected system anyway, without any attempt at remediation.

The battleshort approach is different from *in situ* monitoring and isolation in that no one is necessarily doing an analysis or trying to fix the problem. This approach works best if an organization has plans and procedures in place to handle the situation—without them, battleshort effectively becomes "ignorance is bliss," which can negatively impact the organization's mission.

Consider a company that inadvertently distributes malware to all of its desktops. The IT department could shift critical processes to other networks, require hosts to submit to network access scans or use some type of software attestation, wipe HTML and attachments from email, or force collaboration on smart-card-authenticated websites or SharePoint, all while leaving the malware in place. The company might choose this approach if it simply doesn't have time to shut down the network and reimage every machine. Other options might include isolating the system to limit or prevent outside access. For example, the company might increase a firewall's filtering levels, switch to a different communications media, or force a "key change" in the communications in-

frastructure. These options work best if planned (and tested) in advance.

The battleshort approach has significant risks. If the attacker is an insider, then he or she might be able to influence the remediation actions. One way to counteract this vulnerability is to enforce mandatory job rotation during a crisis situation so that more than one person is responsible for each job function. Another option is to ensure a strict separation of (security) duties and enforce it even in times of crisis.

Providing for Operations with Degraded Security

We believe that modern cybersecurity theory and practice must expand to explicitly consider operations in a security-degraded environment. Users need better procedures than simply disconnecting their virus-infected computers from the network, backing up their data, wiping their disks, and performing a clean install of Windows—the virus remediation steps recommended by many popular websites.[5]

Providing the ability to operate in a security-degraded environment might require both hardware and software support, such as firmware that isn't readily reprogrammable or protected subsystems that can't be updated. Developers created such strategies for the One Laptop per Child project;[6] similar protection is in Google's Chromebook, which is designed to detect problems, wipe itself, boot a protected partition, and restore the operating system and the user's data from Google's cloud.

Out-of-band monitoring is another important tool for maintaining situational awareness in a security-degraded environment. Cloud-based systems might provide for monitoring through the virtual machine supervisor; likewise, hardware could provide for monitoring via logs sent through "data diodes" to otherwise disconnected networks.

Despite widespread agreement that security is a process and that perfect security is impossible to obtain, many individuals, organizations, and even governments today operate as if their computer networks were normally in a secure state. We treat our insiders (especially our system administrators) as if they're forever trustworthy. We assume our software is bug-free. And when our systems inevitably become compromised, our immediate priority is to restore operations—not to diagnose what went wrong.

To return to the RAID analogy, the computing profession has been dramatically more successful in modeling and discussing storage degradation than security degradation. This might be because RAID systems have relatively limited and clearly defined means of entering and leaving a degraded state; attackers have many options for creating a security-degraded environment. As such, organizations need to understand their alternatives and formulate a variety of possible responses. Current approaches assume attack detection is easy, ignore remediation, and recommend immediate restoration. Such policies aren't sufficient to address the complexity of today's threats. We must extend our security repertoire to support operation in a security-degraded environment if we are to learn to live with insecurity. □

Acknowledgments

Beth Rosenberg, James Sterbenz, Stephen Magill, and several anonymous reviewers provided valuable comments on previous drafts of this article. The views expressed in this article are those of the authors and don't reflect the views of the Department of Defense or any of its agencies.

References

1. E.F. Moore and C.E. Shannon, "Reliable Circuits Using Less Reliable Relays," *J. Franklin Inst.*, vol. 262, no. 3, 1956, pp. 191–208.
2. J.P.G. Sterbenz et al., "Resilience and Survivability in Communication Networks: Strategies, Principles, and Survey of Disciplines," *Computer Networks*, vol. 54, no. 8, 2010, pp. 1245–1265.
3. J.F. Meyer, "On Evaluating the Performability of Degradable Computing Systems," *IEEE Trans. Computers*, vol. 29, no. 8, pp. 720–731.
4. Microsoft TechNet, "Erroneous VeriSign-Issued Digital Certificates Pose Spoofing Hazard," 22 Mar. 2001; www.microsoft.com/technet/security/bulletin/MS01-017.mspx.
5. "My Computer Has Been Compromised, What Do I Do?" ClamWin Free Antivirus, 2001; www.clamwin.com/content/view/146/27.
6. I. Kristic and S. Garfinkel, "The One Laptop per Child Security Model," *Symp. Usable Security and Privacy*, ACM Press, 2007; http://simson.net/clips/academic/2007.SOUPS.Bitfrost.pdf.

Simson L. Garfinkel *is a professor at the Naval Postgraduate School, where his primary research area is computer forensics and information technology policy. He holds a PhD in computer science from the Massachusetts Institute of Technology. Contact him at http://simson.net/contact.*

George Dinolt *is a professor at the Naval Postgraduate School, where his primary interest is formal modeling and provably secure systems. He holds a PhD in mathematics from the University of Wisconsin, Madison. Contact him at gwdinolt@nps.edu.*