Pushing Boulders Uphill: The Difficulty of Network Intrusion Recovery

Michael E. Locasto
George Mason University
mlocasto@gmu.edu

Matthew Burnside
Columbia University
mb@cs.columbia.edu

Darrell Bethea
UNC Chapel Hill
djb@cs.unc.edu

Abstract

One of the most significant unsolved problems for network managers and system administrators is how to repair a network infrastructure after discovering evidence of an extensive compromise. The technical issues are compounded by a breathtaking variety of human factors. We present a study of three significant compromises of a medium-scale network infrastructure. We do so as a way to expose the difficulties — both technical and human — inherent in intrusion recovery. Most network users take a "secure" network infrastructure for granted. Real events show that this level of faith is unwarranted, as is the belief that intrusions are or can be completely repaired, especially in the absence of research on network recovery mechanisms that explicitly take the needs of support staff into account. We conclude with lessons learned and some detailed suggestions for tools that can help bridge this gap.

"Damage control is much easier when the actual damage is known. If a system administrator doesn't have a log, he or she should reload his compromised system from the release tapes or CD-ROM."
– Firewalls and Internet Security:
Repelling the Wily Hacker [6].

1 Introduction

This paper presents a case study of the impact of social pressure, technical experience, bias, and other constraints on both individual and group risk assessment and decision-making during the recovery efforts from three significant network intrusions at a single site in March of 2007, December of 2007, and March of 2008.

Although many people enjoy the benefits of access to information and communication through networked systems, most take the security and reliability of these infrastructures (residential ISPs, workplace IT departments, the IT infrastructure of educational institutions, *etc.*) for granted. Users do not often see the impact of computer break-ins and intrusions beyond the occasional sensational story that reaches the front page of some major news outlet. Skilled attackers work hard not to be noticed. System administrators worth their salt work even harder to make sure intrusions are prevented. Institutions have deep concerns about negative publicity.

As a result, users have a misguided understanding of the frequency of such attacks and the difficulty of maintaining and repairing a network. Users may incorrectly assume that IT staff can fully repair the damage or harm (think of copied intellectual property, computer cycles used, reputations lost) caused by an attacker. Even researchers in the systems security space may summarily dismiss the task as a simple, if somewhat lengthy, system administration job, and thus unworthy of investigation. It is our opinion that the problem of coordinating the repair and restoration of network infrastructures is a major unaddressed problem that embeds a number of unanswered research questions involving the intersection of human factors and technical challenges.

1.1 Dual Nature of the Problem

Compromises of medium or large networked systems (such as the infrastructure supporting a research department, college, or university) are difficult to analyze and respond to for a number of reasons. As a result of the diversity of the problem and the lack of research into methods that deal with both technical and human factors, network intrusion recovery is more of an art than a science. The state of the art often involves manually reinstalling machines from read-only media, as the traditional text on firewalls [6] reminds us in the quote above. Even when this process *is* automated, it still resets systems to some initial state, thus deleting valuable data that may not have been backed up, or information that would be of some use in a forensic investigation. At this point, we must resist the temptation to treat the problem as solved by turning to some technical solution (*e.g.,* automated network-based OS installations, "ghosting" software, or recent research on an automated process for working backward from the attack to undo the damage caused [13, 7]). Both technical and human factors introduce obstacles that simply executing a software application cannot overcome.

Even with the assumption that we can reliably detect an intrusion, there are many technical issues related to repairing a wide variety of hosts, nodes, objects, and artifacts. These issues, and the decisions necessary to address them, are compounded by a number of human factors. The workflow we depict in Figure 1 and the issues listed below are representative rather than exhaustive.

First, even with deep auditing information, it can be difficult to describe the extent of an intrusion within the context of a single system. Second, determining the extent of the damage throughout the network requires replicating or extracting those conditions to widen the scope of the detection process. Once the process of detecting an attack and determining its scope have been accomplished, then the process of recovery presents an overwhelming series of choices and possibilities. As we can see from the incidents described later in the paper, this process is not strictly linear. Thinking of detection as "accomplished" rather than "ongoing" is misleading.

Planning and implementing a recovery can involve a variety of changes to systems, hardware, applications, and network topology. Individual systems require forensics and may need to be isolated, removed, updated, or reconfigured. Software applications may need to be reconfigured or have patches applied, which raises the twin issues of which applications to fix in what order and what patches to generate or obtain (and what order to apply them). The network topology may need to change: new routers, switches, or other equipment may need to be introduced or existing equipment reconfigured. Firewall rules may need to be introduced or modified. Existing IDS sensors could require retuning. During this entire process, the team must test and verify each step.

We begin to see recovery as a complicated, fluid process. Response teams often labor under a compressed time frame to fix as large a part of the intrusion in as short a time as possible. The forensics process experiences pressure to finish quickly to reduce service downtime. The recovery team's training and skill level, along with the vagaries of interpersonal relationships, can constrain what types of actions are realistic. Promotion, demotion, hiring, or termination decisions can affect someone's willingness to engage in extensive recovery actions. In addition, attacks rarely occur at convenient times; if the incident occurs near social events or holidays, time pressure can greatly increase.

Although some technical fixes may be "obvious", both internal (to the team) and external (*i.e.,* the team's customers and employers) vested interests in maintaining the network status quo can prevent the implementation of these fixes. The team must be familiar with the preferences, attitudes, and biases of the user or customer population in order to "sell" the repair to them. Finally, the reputations of the team, individuals, customers and users, and institution requires careful consideration.

1.2 Contributions

This paper offers evidence that illustrates what might otherwise be an overlooked point by information security researchers: intrusion recovery is not a simple systems administration task. Intrusion recovery, while a large technical challenge, is further complicated by human–level issues, and we highlight specific issues involved in the incidents we describe. In addition to our analysis, we provide the research community with three real (rather than artificial, contrived, or based on conjecture) threat and recovery scenarios. Intrusion recovery systems are relatively neglected in the research literature; we believe the community should focus on creating mechanisms that deal with recovery as a system composed of both humans and computer systems.

1.3 Background and Related Work

Complete technical solutions to the problem of recovering from *realistic* intrusions in the research literature are sparse, although both classic [25, 24, 5] and more recent [23, 11] examples of post-mortem intrusion analysis do exist. Spafford's analysis [24] of the Morris Worm and Cheswick's annotated log of the Berferd case [5] can be seen as catalysts for changing the way computer scientists and network researchers thought about trust and security on the fledgling Internet. The analysis of these incidents helped spur the adoption of stronger authentication mechanisms, the use of firewalls to implement host communication policies, and research on basic auditing tools and intrusion sensors. Singer [23] recounts how even a well-designed infrastructure managed by an experienced, professional network security team can be compromised. This latter analysis helps illustrate just how difficult and time-consuming it can be to completely remove an attacker from a system. In particular, the attacker described in Singer's article would repeatedly find another avenue into the infrastructure just when the admins thought they had adjusted their security posture appropriately.

The HotAdmin[1] project at UBC has looked at the nature of the job of security administrators [10]. They compare the dynamics of a centralized and distributed secu-

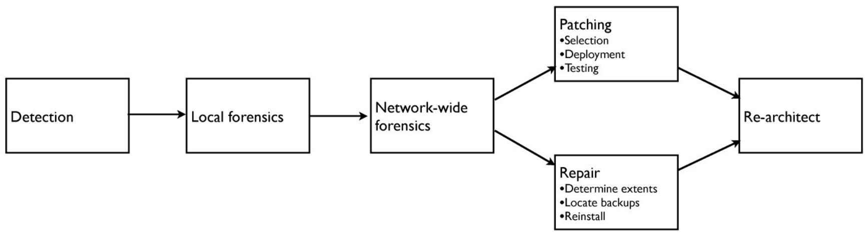

Figure 1: *Response flow.* Our attempt to define a workflow for a technical response at a high level of abstraction. Decision trees at individual parts of this workflow can be partially hidden or incomplete with very large branching factors. This figure belies the fluidity with which a response scenario can take place; detection, diagnosis, and reaction do not form a strictly rigid, step-by-step process. In searching for a way to more easily visualize the relationships between these activities, we compromised at a high level of abstraction.

rity group at an academic organization, and how the transition between the two models worked.

Much research takes a prophylactic stance: networks and systems should be hardened before an attack occurs. Needless to say, proactive hardening, even if it provides strong protection mechanisms like tainted dataflow analysis [16, 26] only partly addresses the problem: the cost of use may be high, the adoption rate low, and the "coverage" of the technique (in terms of classes of attacks defended against) narrow. To date, only memory address space randomization [2] seems to have seen significant deployment, but even this protection mechanism only addresses a certain class of attacks. Other efforts to deal with intrusion recovery discuss ways to provide secure backup, alert logging [7, 19], and audit systems [21]. Meanwhile, Kursawe and Katzenbeisser [14] argue that the prophylactic stance is limited. They introduce a new paradigm where computer users accomplish useful work even though their machines are compromised.

The problem itself appears too large for a single, comprehensive technical solution [9]. System administrators, therefore, are relegated to selecting a hodgepodge of sensors and countermeasures to help defend their networks and restore order when intrusions are finally noticed. The selection process is driven by a variety of possible considerations — not just purely technical issues. These considerations may range from cost and resource constraints on equipment and personnel to "political" factors, personal experience, or recommendations from friends or colleagues. These factors can exert a powerful influence. Although neither of the following cases apply in the incidents we describe, it is not difficult to imagine such situations. For example, a faculty member may have had a role in developing a particular networking technology or intrusion sensor, or an IT company feels obligated to use only their OS or toolset.

Defending a network involves assessing risk and allocating resources to match the perceived threats and costs [4]. In terms of network intrusion recovery, knowing that the network is at a high risk of a compromise does not directly inform the procedures that should be in place for repair. Instead, it may inform strategies for reducing or managing risk, and little research exists on systems for managing the disaster workflow recovery once a network *is* compromised.

The psychology community has spent a significant amount of time studying and trying to understand the process of human decision making under duress. Payne *et al.* [20] provide a good overview of the research in this area, including beliefs about uncertain events, decisions made under risk and uncertainty, and frameworks for decision behavior. Consideration of how security-related decisions are made under stress seem to fall most naturally into discussions about the threat model a system operates under, as bad decisions by the system user could increase the power of the hypothetical attacker.

Finally, as we saw in our attempts to collect information for this case study, the human memory and recollection is notoriously unreliable. The reliability of eyewitness [27] and earwitness [3] testimony has been extensively studied by psychologists; in fact, Wells and Olson [27] point out that the only scientific body of literature on eyewitness reliability exists in the psychology space. In the computer security field, and in the context of rebuilding complex network infrastructures and carrying out a number of both repetitive and complex tasks over a long period of time, human memory is relied upon far too much. Our case study shows that it is possible for initial planning goals, suggestions, or objections to be misunderstood, warped, or forgotten — leaving potentially large gaps in the actual level of security achieved after repairs complete.

2 Methodology

In researching this paper, we interviewed all parties involved in recoveries from three recent attacks on a medium-scale network, including administrative staff and management. We performed an email archive search, and confirmed many of the details of the attack through analysis of disk images from a number of the compromised machines. We performed an initial debrief of the entire IT staff and followed up more extensively with four of the IT staff members. We have continued monitoring the organization's response.

We emphasize that we do not aim to lay blame with individuals, and we refrain from naming the people and organization involved. Each interview subject gave us permission to interview them and report on the process. Our goal is to present the facts of the situation, disposition of the network, and decisions made by the staff in as clear a light as possible as a way to motivate research and development of tools that ease the burden on IT staff during the process of network intrusion recovery.

One of the most significant challenges when responding to an intrusion is performing forensic analysis to determine the exact impact of the attack. In the case of the compromises we discuss here, the nature of the attacks was such that no individual performed a single coherent analysis. Rather, the analysis was performed piecemeal by the various members of the IT staff, and, as such, each had a different view of the impact of the attack. As we discuss later, this fractured view presents the IT staff with problems when attempting to form a coherent response. Furthermore, it presents a problem to us as researchers. In some cases, parties we interviewed had radically different timelines and analysis, even though the interviews took place less than a month after the attacks of December 2007 and within the scope of the March 2008 attacks. Where possible, conflicting statements were reconciled through mechanical methods (email or file modification dates) but some ambiguity remains.

3 Intrusion Incidents

The network on which we focus our attention in this paper is the network for a mid-sized research department at a large university. The network consists of approximately 1000 Windows, Linux, and Solaris workstations, as well as a number of infrastructure servers providing DNS, DHCP, and HTTP, and several general purpose compute clusters accessible via SSH. Approximately 150 of the workstations run Red Hat Enterprise Linux (RHEL) AS v4. These machines are periodically updated from two source machines using `rdist`. For the purposes of load balancing, the two `rdist` masters are each responsible for half of the machine population. User authentication in this environment is centralized. Windows machines authenticate users via Active Directory; the Linux and Solaris machines authenticate users through NIS. At the time of these incidents, the network employed neither a rule–based IDS like Snort nor an anomaly sensor. As a partial result of these incidents, the network will shortly employ a content–based network anomaly sensor. Machines are generally not firewalled (although most end hosts have a local firewall supplied by their OS vendor). The network supports a research environment with a strong tradition of open access. This tradition supplies a political force that has precluded the use of any form of firewall at the network edge. One of our colleagues (not associated with these incidents) pointed out that a firewall is merely a device for implementing policy. If the policy is unclear, then the mere presence of such a device is unlikely to help.

Over the time period covered in this case study, the network was administered by an IT staff of three to five people, with a single manager. This staff works independently within the context of the larger IT organization of the university. The IT manager is highly experienced in managing staff and infrastructure and had previously completed a vast overhaul and update of the infrastructure to bring some amount of order to what was an otherwise disorganized physical and virtual space.

There is a high turnover, and staff members come from widely disparate backgrounds; some are students with little to no experience, while some are highly knowledgeable and very experienced. The network is complex for its size and has a number of systems, including the accounting system, which remain unchanged from the late 1990s. New staff, even if highly experienced, often take months to gain a complete understanding of the intricacies of the network.

3.1 March 2007 Attack

In March of 2007, an attacker attempted to use a kernel exploit to gain root privileges on several of the RHEL workstations. The attack was discovered when several of these attempts failed, raising alerts. For each machine on which the attack failed at least once, the IT team were able to use system logs to determine the origin of the attacker and the compromised user accounts he was using to access the machine.

The failed attacks were not all the same, however; the attacker was revising his methods, and there was no way to determine if he had succeeded. The staff checked the logs of other susceptible machines (those harboring the same vulnerability, but showing no indication of failed attacks). While staff could uncover no indication that the attacker had connected to the machines, it is possible that he altered the logs after gaining root access.

3.2 March 2007 Response

It is possible that the attacker never succeeded. Regardless, the safest response in this situation, recognized by all members of the IT staff, would have been to reinstall all vulnerable machines with a patched version of the operating system. There were, however, external constraints that prevented this approach. The attacks occurred in the middle of the semester and involved many machines heavily used by classes. Thus, the staff needed to carry out a solution as quickly as possible to avoid disruption to the Department's academic mission.

Most of the systems are nearly identical, with the exception of the servers and the `rdist` masters. Reinstalling the `rdist` masters would have been time consuming and error-prone, as the `rdist` distribution architecture in use was archaic and proprietary, and those most familiar with it were no longer employed.

Furthermore, reinstalling the workstations using the `rdist` new-install process would have taken far too much time, as each install generally took about a half day, and due to network bottlenecks (much of the install was network-based), no more than four or five machines could reasonably be installed at any given moment.

The IT staff's primary insight was that there were two classes of vulnerable machines: servers and workstations. The attack required a user-level shell account on the target machine in order to work, and the attacker had compromised at least one or two student accounts (as indicated from the logs of the failed attacks). Student accounts, however, do not have access to the servers, so the likelihood of an infection on those machines was less than on any given workstation, as long as the staff assumed that the attacker had not compromised any administrator accounts. The workstations, on the other hand, were mostly identical, only differing in a few configuration files. By isolating those files, the staff believed they could clone workstations from other workstations and avoid the bottleneck to the master `rdist` servers.

The staff shutdown each server and ran several rootkit checkers. They also performed some manual log inspection for any indication of an attack. Seeing none, they patched the servers and brought them back online.

The staff then performed a standard (half-day) new install on a single workstation via the master server. While this new, clean workstation was installing, the staff used the time to analyze workstations of many different configurations to determine the minimal set of configuration files that would differ per machine. They also burned approximately twenty Linux LiveCDs. Once the first workstation was finished installing, the team went to each remaining workstation, booted to a LiveCD, and inspected the configuration files which were to be left untouched to verify that they contained nothing malicious.

The staff members then downloaded and ran a script from the local intranet. This script erased most world-writable locations on the machine (`/tmp`, parts of `/var`, etc.). It then synchronized the remainder of the local filesystem (with the exception of the wiped partitions and the workstation-unique configuration files) directly from a known-clean workstation. Staff then re-configured and re-installed the bootloader and restarted the workstations.

Once the single clean workstation had been cloned, it was possible to use the newly cloned machines themselves as `rdist` masters for other machines. For example, by choosing masters within the same room, on the same local switch, it allowed for a dramatic decrease in the amount of time for the entire recovery. Note the level of detail and manual effort involved in starting and evolving the repair and recovery process, including a heuristic learned only through direct experience with reinstalling machines in a localized fashion.

At this time, staff considered the problem resolved and returned to normal day-to-day operations. However, we saw in our interviews that some members of the staff recognized, even at the time, that they were unsure whether the attack had been truly cleaned up. Furthermore, there was no record keeping and no analysis or formal discussions regarding installation of additional security measures such as an intrusion detection system.

3.3 December 2007 Attack

Early in 2007, four new machines arrived at the department, intended for use in high-performance graphics research. Each machine was equipped with a high-end NVIDIA graphics card. No official Linux drivers for these graphics cards existed, so staff used unofficial drivers. In early December 2007, all four machines stopped working. The IT staff installed updated (and now official) graphics drivers, which solved the problem until all four machines crashed the next day.

The staff pushed out updates to all RHEL machines through its two `rdist` servers, `starsky` and `hutch`. `starksy` is the primary master `rdist` server and `hutch` is a secondary. The infrastructure accomplishes upgrades with a two stage process. In the first stage, the active RHEL installation on `starsky` is upgraded. This live operating system is manually imaged and the image copied to `hutch`. A `cron` job on each of these machines pushes the upgraded image out to half of the 150 machines. The unfortunate consequence of this architecture is that a compromise on `starsky` would be pushed out automatically to the entire network. The staff installed the updated NVIDIA driver on `starsky` to prevent it from being overwritten on the graphics machines after the next `rdist`.

In addition to handling the NVIDIA issue, the staff also upgraded the kernel on `starsky` from version `2.6.9-55.0.9.EL` to `2.6.9-55.0.12.EL`. At 4 AM, the cron job delivered the upgrade to all 150 machines. On 10 December 2007, the staff discovered that both `starsky` and `hutch` had crashed. The staff attributed the failure to the recent upgrade, and investigating it was added to the end of a long task list for one of the staff members. Both machines crashed again on several subsequent nights.

The issue was finally explored on 13 December 2007, and the recent patches were rolled back on `starsky`. That night, both machines crashed again. This was a strong indication that the patches were not the problem, so an attempt was made to re-upgrade `starsky`. The upgrade failed when, during kernel compilation, the `mkdir` command returned an error. On the morning of 17 December 2007, exploration of this error determined that `mkdir` failed when attempting to create directories consisting only of numeric characters. IT staff began to suspect a rootkit. Booting to a LiveCD confirmed that suspicion: several files, including `mount`, had been replaced.

The hypothesis of the IT staff is that the rootkit installed by the attacker conflicted with the kernel module of the NVIDIA driver. If the attack took place in the first week of December, the rootkit would have been pushed to the graphics machines, a conflict ensued, and the machines crashed. Installing the driver on `starsky` caused that machine to crash too. The near-simulaneous kernel update obscured the real issue.

3.4 December 2007 Response

Discussion and planning for the response took place in a hallway at around 1pm on 17 December 2007. The planning group was assembled informally and consisted of the IT manager, three IT staff, and two authors of this paper, who happened to be nearby.

Initial discussion surrounded disagreements on the scale of the attack and the nature of the exposure. There was a brief argument over whether the `rdist` servers could be re-imaged and a clean install pushed out to all machines. This idea was discarded because it was recognized that all 150 machines would have to be reformatted from scratch. Planning began on how that process would take place, and a number of questions were raised immediately. What, if any, changes should be made to the system architecture? If changes are made, in what order, and to which machines, should those changes be rolled out? Who will be involved? Staffing shortages imply that any changes beyond the simplest would take weeks or months to put in place. How will changes affect end users? Finals week is in progress, so taking large numbers of machines offline is undesirable.

Discussion immediately centered around whether the staff should either stick with Red Hat Enterprise Linux or move the machines to another operating system. We note that were was no *a priori* reason to blame RHEL for the intrusion, and we question whether this was an appropriate first topic for the response team to examine. OpenBSD was proposed and discarded, primarily due to the IT staff's unfamiliarity with the operating system. One member of the staff was familiar with Ubuntu, had a working Ubuntu installation (an experiment to support a new authentication infrastructure) and argued for this option. The IT staff has high turnover, so there was no RHEL expert currently employed and there were no individuals present who were capable of competently comparing RHEL and Ubuntu. Lacking any quantitative comparisons, no strong opposing voices emerged, and the Ubuntu motion carried.

Discussion moved on to the user directory and authentication system. The existing mechanism was based on NIS. As we mention above, one member of the IT staff had a pre-built LDAP server in place, so movement to LDAP was quickly agreed upon, especially because this provided a reason for the Ubuntu switch.

The agreement of those in the meeting was that a new network, independent of the existing network, had to be created, and each account had to be re-created with fresh authentication credentials (passwords, SSH keys) in the new network. Since it was finals week, most machines were under heavy use. An underused 8-machine cluster was proposed as a testbed for the deployment, and the group agreed that that cluster should become the testbed for the Ubuntu rollout.

Now that an overall plan was in place, the next question was one of prioritization. Since it was possible that the attack had been an insider attack (perhaps aimed at gleaning final exam information), the highest priority was to build clean Ubuntu images for the faculty. Thus, the faculty and finals remained the first critical concerns.

The December 17 meeting then broke up, and the IT staff began work. The first public disclosure of the attack happened one hour later when the IT manager emailed all faculty and PhD students informing them of the intrusion. All faculty passwords were to be changed.

On December 18, all PhD students teaching classes were informed that they would have to undergo the same procedure outlined for faculty the previous day. The department was also notified about the impending staffing shortage; half the IT workforce were leaving for jobs in the finance industry at the end of the year (in three days). Faculty cell-phone numbers were requested so staff could text message them new passwords. Installation of Ubuntu and LDAP on the test cluster began.

Students and staff then departed for the holiday break. The IT staff returned on December 27, and a Ubuntu rollout on another computer cluster began (this time, a general-purpose lab). The next day, Solaris machines were upgraded to Solaris 10. On January 8, all guests and visitors were moved to the new system.

3.5 March 2008 attack

The March 2008 attack was detected by a member of the IT staff who noticed a new account named `mysqld` with root privileges on an important web server. Examining the contents of the home directory of this account showed several interesting files.

1. `.bash_history` containing what is probably a partial record of the attacker's behavior.
2. `ali.txt` containing the results of an NMAP scan for port 5555 (`freeciv`) across a /16 network.
3. `bot.pl` An IRC-based bot engine.
4. `dos.pl` A simple denial-of-service engine.
5. `xpl.c` Source code for the vmsplice Linux local privilege escalation exploit.

The `mysqld` account appeared in the `lastlog` history, along with the attacker's source IP address. Searching for that address in the Apache web server logs indicated that the attacker had repeatedly requested several files in a directory containing a common PHP web application, which was several revisions out of date, with remote exploits in the wild. The attacker added a copy of the `nsTView` remote web administration tool to the web app directory, leaving it set up with the default password.

The Apache logs also indicated that the attacker had downloaded a file he had created called `secret.txt`, containing the username and password for the web application's MySQL database, and the IP address for the remote host on which the database was running. Unfortunately, logging was disabled on the MySQL database, so investigations are limited in that direction. It is unknown whether the attacker ever connected to that database, or used one of several MySQL privilege escalation attacks to examine any of the other databases on that server.

We do note that, given the age of the web-application exploit, we believe that it is unlikely this is the first attacker to come in through this vulnerability. Furthermore, the `nsTView` remote web administration tool was using a default password, so multiple attackers may have come in through that route.

3.6 March 2008 response

The response to the attack began by removing `mysqld` from `/etc/passwd` in order to disable it. The MySQL server daemon was shut down shortly thereafter. The owner of the vulnerable web application was then contacted and it too was shut down. These responses were performed quickly – within two hours of the attack first being detected – and then the response turned to a policy discussion. What architecture and policy changes need to take place to prevent such attacks in the future? Several alternatives have been discussed, including undertaking a manual review of all web applications, prohibiting web applications entirely, making patching the mandatory responsbility of users running web applications, and moving the web infrastructure to a "read-only" style web site that is periodically refreshed from virtual machine snapshots. Users remain responsible for checking that their software is patched.

4 Incident Analysis

We next highlight some of the key decisions, discuss why they were not based purely on technical considerations, and suggest research directions aimed at helping automate and ease the process of decision making and reasoning under uncertain beliefs and knowledge. Note that our purpose is not to pass judgment on a particular decision by labeling it good or bad: the central goals of our analysis are to observe how non-technical factors influence decisions and to highlight what kinds of technical systems might help manage that influence.

4.1 Observations

Lesson 1: Cross-layer, anomaly-based intrusion detection seems valuable for detecting stealthy attacks. This type of detection is far more comprehensive than system call sequence monitoring and involves the fusion of alert streams from multiple levels of system abstraction.

All three attacks were discovered manually through symptoms and side–effects of each attacker's activities rather than traditional intrusion sensors like Snort or a commercial anti-virus product. At the time, the network did not employ a traditional network IDS, and little in the way of automated detection beyond some syslog monitoring scripts, but neither was the active attack sequence something detectable by a network intrusion detetion or a desktop anti-virus software system. In the March 2007 attack, abnormal kernel activity prompted an investigation by an IT staff member. In the December 2007 attack, crashes noticed by the Graphics research group led to the eventual discovery of the rootkit. The March 2008 attack was noticed by an IT staff member discovering a new privileged user account by accident — prompted by a trouble ticket filed by a senior professor asking why some standard mount points were failing.

This situation suggests that alert and educated IT staff and users are critical to uncovering stealthy attacks. We acknowledge that the sample size of incidents is small and purposefully focused on extensive intrusions (rather than well-known worm infection attempts). This lesson should be taken as a call to focus on creating anomaly sensors that span multiple levels of a system. For example, a system that correlates a user's inability to mount their regular partitions with anomalous network or host traffic can help build evidence for a comprehensive anomaly. The research challenge here is to move beyond AD techniques that rely solely on various flavors of system call sequence modeling.

Lesson 2: Staff do not have the luxury of complete forensics.

From an end-user viewpoint, this lesson was rather surprising at first, perhaps because we believe computer systems to be more flexible than they are in reality. Although we knew that undertaking an effective forensics process is challenging, we were surprised at the nasty dilemna of trying to analyze a host that one also wants to keep running. A tension exists between short-term operational demands to keep services running and long-term demands from the ISP to keep a network clean. Disks and machines have to be kept in use; we suspect that many organizations lack the luxury of taking them offline for extensive cleanroom analysis. Hot swappable and mirrored disks do offer a way to keep a machine online while also looking at a snapshot of the current content, but not all organizations can afford this type of redundancy for all their machines.

For example, if a critical server has been infected, the IT staff might decide that it is more important to quickly reinstall the server and restore normal operation than to analyze the malware in any depth. But while operational demands are important, the forensic analysis they preclude might reveal information which ultimately proves more critical still – perhaps it establishes that the infected server also infected other servers, or it might show that a compromised administrator account was the initial source of the intrusion, meaning that a reinstallation alone will not solve the overall problem.

An ISP often imposes constraints on real-time analysis of infected machines. IT staff may wish to analyze an infected machine's traffic to see if any other machines are communicating with it (and thus might be infected). But ISPs are often more concerned with limiting damage caused by an infected host. They will sometimes insist upon removing it from the network immediately, especially in academic environments, where the university is directly responsible for most hosts on the network. Large public ISPs may be less demanding to match their reduced liability.

Lesson 3: Visualizing a decision surface can help inform overall strategy and planning.

After detecting an incident or intrusion, it is difficult to immediately identify and execute the appropriate next steps; a staff is effectively in the middle of diagnosis. Staff may be torn between a number of actions, including continuing diagnosis and forensic efforts, fixing the immediate problem or small–scale symptoms of an attack (turn off a particular service, unplug a particular machine, remove a login entry from `/etc/passwd`), and fixing the larger–scale symptoms or root causes of an intrusion.

In the medium to long term, staff members needed a system that could direct the implementation of the solutions they had arrived at. To a certain extent, such a system includes standard "trouble ticket" or issue tracking software. In contrast to such an "obvious" tool, the technology that the staff actually used to plan out recovery activities for the December 2007 attack included a whiteboard and a marker. The whiteboard was inadvertently erased. The marker remains at large. Interestingly enough, usability research on display–centered group activities has found that displays are important in the planning stages of the activity, but grow progressively less useful as the plan is enacted [12].

In the short term, staff members needed a system that could direct planning activities by giving them a feel for the magnitude and location of various pitfalls (whether human or technical in nature). We suggest the concept of a decision surface composed of **process clocks** (a visual representation of task complexity using an estimate of task difficulty to shade in a graph node) as one way to achieve this high level view of the difficulty of the terrain ahead. We have found that standard decision trees and swim-lane diagrams are not quite appropriate for this goal, but we are left without any ready alternatives.

Decisions, and the reasons for making them, can be difficult to articulate and defend. Describing a decision making process can leave one lost for words — sometimes elements of the decision were based on intuition, flashes of inspiration or emotion, a complex sequence of data analysis, or deep contemplation and personal reflection. However hard it is to describe the process of making a decision, we have found that visualizing the elements of a decision is even harder. One of our main inspirations for writing this paper was the lack of a way for our system administrator to assess — at a quick glance — the difficulty of the terrain ahead of her, including parts of the decision surface where human and technical factors would conspire to greatly increase (or even decrease!) the complexity of the available alternatives. We have asked a number of our colleagues for their best method of visualizing a decision, and we have repeatedly drawn blanks. We consulted Edward Tufte's work[2] in hopes of gaining some insight into visualizing the elements of information involved in a decision, but most information visualization principles did not seem directly applicable to this problem of visualizing a *process* (rather, a collection of processes).

As a result, we are attempting to define a model for visualizing a decision surface that would take into account the properties we observed to be important in guiding the process of network intrusion recovery: amount of human involvement, estimated effort for task completion, ordering dependencies of tasks, potential disruptions. We start by seeking to construct what we call a *decision surface*: a two dimensal plane akin to topographical maps projecting three dimensions onto a flat surface. The peaks, valleys, and plains of a decision surface convey at a glance where difficult or complex decision points lie. Knowing how to compose a decision surface, however, especially in light of future attacks, is a difficult exercise.

Lesson 4: Rapidly setting and executing a diagnosis and recovery agenda requires an unobtrusive, pervasive incident recording and modeling system that can help manage cognitive traps like availability bias and the shortcomings of human memory. Since human memory and recall is far from perfect, multiple points of view supply sometimes conflicting details of attacks and do not assist efforts in forensics, auditing, or planning for the next attack. Recovery graphs may provide one way to encode intrusion scenarios and the human response to them for later auditing.

The crucial first minutes after an intrusion discovery, in which there is no complete information about the attacker's entry point(s), history of actions, short and long-term intent, or current level of activity, hold the potential for panic, an overwhelming amount of data to analyze, and a paralyzed thought process. Involving too many people in the decision–making process after a serious intrusion is discovered can distract the person in charge. The hallway discussion on 17 December involved multiple people, ideas, and proposals. The system administrator involved with our case study achieved a certain level of success at repairing the network only because she was able to rapidly sift through the different proposals that the team members articulated.

Decision making at this point should be aided by automated processes that help manage the signal-to-noise ratio; in studies on decision–making, the manner in which information is organized often appears more important than simply getting increased amounts of information [20].

Furthermore, during our interviews, we observed that details of the attacks and the responses often differed wildly between individuals. Individuals often disagreed on dates — one person confused an attack from March 2007 with one from May 2006 and provided a mixture of details from both. In other cases, individuals presented radically different reports on which actions were taken. Two members of the IT staff disagreed on the date and method of detection of the December 2007 attack, while another viewed it as a continuation of the March 2007 attack. Without a coherent view of the state of the network, it is difficult for staff to make informed decisions to guide the attack response. One suggestion is that a staff member be tasked to record all the actions of a recovery process, but such a role can prove problematic for organizations that have staff shortages and tight budgets.

Even though researchers have proposed work on attack scenarios and attack trees [18, 22], relatively little attention has been paid to analyzing the process of a response. Automatically increasing the rate and types of events logged after an intrusion is discovered and the recovery process is started can assist efforts to revise a disaster recovery plan. More logging can make sure that key decisions are clearly recorded and not subject to human recollection of events occurring during a stressful time of rapid change and high rates of information. This type of recording is substantially different than ensuring that `/var/log/messages` collects more OS–level events. We propose the concept of *recovery graphs* to help capture and encode the sequences of events following the start of a recovery effort.

The lack of a human-centered post-intrusion journaling system suggests that research to design and develop new systems that record human–level events, judgments, recollections, and intentions is needed. Such systems must interact with humans seamlessly: they cannot place an additional burden on already-busy personnel. Categorizing, tagging, and cross-referencing events and information generated during the post-intrusion recovery process can help form a coherent view of what has happened and is happening to the network.

Lesson 5: Designing and maintaining a disaster recovery plan can aid recovery efforts, but the plan must be continuously — not periodically — updated.

The IT staff did not have *a priori* knowledge of what procedures should be enacted to combat or rectify the intrusion or to process and prioritize information about the incident. While the lack of a disaster recovery plan is a major operational shortcoming, disaster recovery plans alone are not a panacea. Like any proactive defense method, the plan may be incomplete, outdated, or unlikely to work given the current personnel. For example, the IT manager in our case study faced a critical personnel shortage due to events unrelated to the intrusion: half the staff was leaving for new jobs in a matter of days.

A disaster recovery plan must constantly evolve. Each new attack, vulnerability, or patch affects the recovery details. Similarly, employee turnover, improved employee skill set, and application deployment require modifications to the plan. The question of how often to update the disaster recovery plan is a risk analysis and assessment task that must balance the needs of the staff to accomplish everyday system administration tasks against spending an inordinate amount of time planning for disasters that might never occur.

The open research question here is how personnel changes, catalogs of personnel skill, and lists of resources, sensors, countermeasures, toolsets, and inventory can drive an *automated* (and potentially real-time) update of the disaster recovery plan. We recognize that this research goal is rather ambitious (some readers have called it unrealistic — although existing pervasive recording systems [8] indicate otherwise), but we stress that this type of problem is precisely where the research gap is: little or no work in this space looks at ways to combine both humans and computers into a cohesive system where the computational elements are responsive, proactive, and transparent to the human as they go about their main tasks. In our minds, such a research direction is new and exciting, especially in a subfield where the bulk of the research looks at tweaking IDS parameters, considering an endless array of new features, or slicing up botnets in a variety of ways.

Nevertheless, the need to improvise can lead to creative solutions. For example, one of the most interesting countermeasures taken by the system administrator in the December 2007 attacks was to find an alternative distribution channel for new login credentials. The administrator sent text messages to the bulk of the user population with their new account password. This side channel is inexpensive (we estimate ten cents per message for roughly one thousand users), and it served quite nicely to distribute authentication material to users who were physically dispersed over the winter break.

Lesson 6: Decisions about appropriate technology shifts are driven by informal personal inclinations rather than quantitative (or even qualitative) comparisons of system properties.

Making changes to a complex and corrupted infrastructure requires (besides a quality analysis of the intrusion) a good understanding of the benefits offered by selecting one technology over another. For example, when the staff discussed whether to change computing platforms from RHEL to Ubuntu, the decision was made without any point–by–point comparison of the *security* benefits of either system. Although a question was raised about whether or not Ubuntu incorporated SELinux by default, as RHEL does, it was neglected (a symptom of the need for a recording system). The staff expressed comfort with Ubuntu's package management software and indicated that one staff member had already prototyped an Ubuntu system that would support stronger authentication. While good package management software can greatly ease the job of system administration, we feel that it is not the primary or only factor in a security–related decision. In this instance, however, the intrusion presented an opportunity for the IT staff to increase the security of the system.

Note that this decision represents an astoundingly rapid shift; even though the underlying platform is Linux, the actual delta is significant (placement of system files and scripts, customizations and patches to the kernel, *etc.*). Even such a minor shift stretches the limits of possibility; for example, deciding to switch the infrastructure to Windows or *BSD would not have been possible in the same amount of time the RHELv4/Ubuntu shift was accomplished.

This lesson points to the need for research into models or techniques to help estimate or otherwise provide some quantitative measure of how the defense posture will be affected after choosing to implement technology X over technology Y. Techniques like attack graphs [18, 22] and event-correlation [17] may help by focusing attention in important places, but at that point we need to begin the process of helping to make recovery decisions.

In this case study, the IT staff did not perform even a cursory examination of the release notes of the latest versions of the operating systems under consideration. While the circumstances and the time pressure demanded a quick decision, it would be best if the IT staff were not placed in such a bind to begin with. Providing systems that automated these types of comparisons and parameterizing them with the details of the intrusion or incident can assist staff efforts to make rational, informed, and *technical* decisions rather than strictly intuitive ones.

We can think of at least three research directions stemming from this lesson. It may be possible to use Natural Language Processing techniques to compare the release notes of the latest versions of two (or more) pieces of software for items that may impact the security posture of the organization. Second, a more realistic goal may be to create a system that data mines bug report databases and vulnerability mailing lists for items that are relevant to the security of the software systems under consideration. Finally, if the source of the intrusion can be traced to a weakness in a particular software package, it may be possible to work forward to predict other vulnerable components in that software [15].

4.2 Where Do we Go From Here?

Accounts discussing the trapping and tracking of attackers in improvised honeypots form part of the classic network security literature [25, 5]. Just as these accounts relate the first examples of a honeypot and computer forensics, the improvisation required in these early responses forecast exactly the plight of network administrators to-day: when faced with a real attacker, decisions must be made quickly and accurately, and the decisions may conflict with the desires of other stakeholders. At the times of these early incidents, almost no tools existed to help trace hacker activity. Tools were improvised from the ground up, and their descendants and offshoots have become part of a standard set of tools. Now, network intrusion recovery faces an even larger challenge: create a suite of tools that take into account not only the engineering challenges of repairing a network, but also the human issues surrounding this process.

We have five specific suggestions for the construction of tools that could reasonably see use in the near term:

1. A way of visualizing complex decisions at a high level. A decision surface could help convey terrain information rather than just a branching factor of standard decision trees.

2. An unobtrusive, pervasive, and real-time activity monitor capable of efficiently and reliably recording both computer events and human actions during the recovery process.

3. A standard for encoding intrusion scenarios derived from the data captured by the above system, including the behavior of both human actors and computer systems in terms of the information structures they maintain and the sequences of actions they take. The keys to this standard are both fidelity and portability, so that these scenarios can be run on simulation infrastructures that employ, for example, different virtual machine hosts.

4. An environment for executing, analyzing, and reviewing these scenarios. For this environment, we can turn to recent work in the arena of computer game design [1] that focuses on the simulation of realistic crowds (rather than randomly milling zombies or predictably scripted bots) for a variety of purposes, including realistic storylines, evacuations from buildings or transportation vehicles in a crisis, and automated assessment of the usability and ergonomics of functional living or working space. This type of tool is useful for both post-mortem analysis to learn from the incident and to ensure that a recovery plan was fully enacted.

5. A toolset for automatically analyzing relevant security properties of alternative solutions. We do not see a panecea here; rather, it is likely that a collection of tools, each specialized to assessing the quality of a particular type of solution, is appropriate.

In each case, these tools help a team of administrators remove guesswork and uncertainty from the process of recovery. We also see a need for a way to input organizational changes to drive changes in a disaster recovery plan, but as we relate above, this task may prove to be too challenging, even if we manage to cobble together some combination of MindMap[3] and a trouble-ticket system.

5 Discussion

This paper has benefited greatly from both formal and informal feedback and reviews. Here, we would like to address some of the meta-issues and high-level concerns that various readers have raised. Fundamentally, we see this paper as the start of a two-pronged effort: first, to formally document intrusion recovery scenarios and second, to create systems that help support intrusion recovery efforts or that streamline the process of intrusion recovery.

The most obvious shortcoming of this paper is that we examine a single organization. It is hard to assess if the same specific troubles affect other organizations, but from our experience and anecdotal evidence, this similarity seems to be the case, at least for academic networks as well as some corporate networks we are familiar with. Some readers have suggested that the nature of the network itself suggests an administrative staff unconcerned with security, and thus it provides an unreasonable organization to base a case study on. Given our direct experience with the personnel involved in these incidents, we believe this is an unfair criticism of their efforts. Strategic security adjustments *are* important to the staff, but so are the day to day struggles — on a tight budget — to keep an infrastructure with many diverse interests and stakeholders operational. Furthermore, at least one other system administrator purposefully and publically runs a network without firewalls [23], like our subject network. Therefore, we suggest that this network is in fact typical of academic-style, open-access networks, and we do not claim that this network is the ideal model for drawing conclusions about, for example, a highly sensitive military network. Nevertheless, recovering from an intrusion remains a common problem, and the travails of the least prepared of us can help even those who are most prepared understand the risks they face.

We anticipate documenting new incidents as well as incidents from other organizations. We are in contact with the technical staff of our institutions to help broaden the scope of this research. We intend to start an archive of structured encodings of these scenarios. Such an archive can support comparisons between organizations as they respond to similar incidents and chose different tradeoffs.

We recognize that incidents similar to the ones we cover in this paper occur every day in many organizations worldwide. Far from making the details we expose here mundane, this reality underscores the fact that this topic is of critical concern. Furthermore, to the best of our knowledge, no one is documenting these incidents in detail or examining how these details change over time. Descriptions of these incidents in the research literature are rare; we cited those that we could find.

Our belief is that the details of these incidents (and how organizations recover from them) are even more revealing and of as much interest as their high–level structure. Furthermore, since organizations have little incentive (in fact, they have potentially large legal and financial disincentives) to share the details of these incidents, academic research into methods of intrusion recovery remains uninformed and undirected. No concretely specified collection of intrusion recovery scenarios exist, and this lack leaves most discussions about the best way to recover from an attack at the level of hand–waving.

Other readers have suggested a variety of areas for further work and improvements, from performing a user impact survey to estimating the economic impact of the intrusions on the organization. We refrain from including this type of analysis precisely because there are no widely-accepted frameworks (although some nascent research proposals do exist [4]) for providing realistic, standardized estimates of costs for losses due to security incidents. Informal industry studies often hyperinflate their estimated costs to serve some agenda, be it marketing a particular security tool, the worth of their own survey, or to provide "evidence" that the problem is worth significant public or private investment. Our goal in this paper is simply to tease out how the technical and human complexities in specific, real-life scenarios interact — not to provide some exotic finanical estimation instrument, especially as none of the authors has any meaningful training in economics.

Finally, one aspect of intrusion recovery that we did not discuss is that of gathering forensic evidence to support criminal prosecution. The prevailing wisdom in this area is twofold. First, many attackers tend (or appear based on the attack source IP address) to be from jurisdictions outside of the US; as the organization we deal with is located in the US, it is unlikely that any such evidence would have been utilized in a criminal trial. Furthermore, many organizations hesitate to bring charges, because doing so requires that the incident become public knowledge. Neverthless, retaining log files and disk images of compromised machines can assist efforts to uncover a larger pattern of malicious activity. In any event, the IT staff was far more concerned with rebuilding the infrastructure and denying access to the intruder than preserving any chain of evidence (Section 4 discusses how IT staff find themselves in a bind when it comes to forensics).

6 Conclusion

Currently, repairing a network infrastructure after a serious intrusion is costly because cleanup is largely a manual process, and the complexity of information systems makes it difficult to automatically trace the extent of the attack. Furthermore, the psychological and sociological aspects of the problem are grossly understudied. Systems involve people, and their security decisions and risk assessments are often based on reasons that are not purely technical. The purpose of this case study is not to question whether the IT staff could have done a better job, or if the organization should have had a more robust network to begin with.

Instead, the lessons we should learn are that real security problems — those whose scope is sometimes too large to comprehend and deal with in any single research publication, are brushed aside as either too large to be interesting, or too close to human and organizational problems to be strictly "systems" security issues. With this case study, we hope to show that interesting possibilities for systems security research exist. Fundamentally, we think that human decisions should be assisted with automated methods that help filter and classify the available information. The problem of network intrusion recovery is a particularly thorny exercise in researching, designing, and creating usable security mechanisms.

Acknowledgements

We would like to thank our shepherd, Nicole Velasquez, for helping us resolve the issues and insightful comments raised by the reviewers. We deeply appreciate the cooperation and help of the IT staff that provides the subject of this paper. Theresa Menzel provided extensive feedback and anecdotal evidence from her experiences with intrusion incident handling. Locasto is supported in part by grant 2006-CS-001-000001 from the U.S. Department of Homeland Security under the auspices of the I3P research program. The I3P is managed by Dartmouth College. The opinions expressed in this paper should not be taken as the view of the authors' institutions, the DHS, or the I3P.

References

[1] Badler, N., Allbeck, J., Zhao, L., and Byun, M. Representing and Parameterizing Agent Behaviors. In *Proceedings of Computer Animation* (June 2002), pp. 133–143.

[2] Bhatkar, S., DuVarney, D. C., and Sekar, R. Address Obfuscation: an Efficient Approach to Combat a Broad Range of Memory Error Exploits. In *Proceedings of the* 12^{th} *USENIX Security Symposium* (August 2003), pp. 105–120.

[3] Campos, L., and Alonso-Quecuty, M. L. Remembering a Criminal Conversation: Beyond Eyewitness Testimony. *Memory 14*, 1 (2006), 27–36.

[4] Carin, L., Cybenko, G., and Hughes, J. Quantitative Evaluation of Risk for Investment Efficient Strategies in Cybersecurity: The QuERIES Methodology. *IEEE Computer* (2008).

[5] Cheswick, B. An Evening with Berferd, in which a cracker is lured, endured, and studied. In *Proceedings of the Winter USENIX Conference* (January 1992).

[6] Cheswick, W. R., and Bellovin, S. M. *Firewalls and Internet Security: Repelling the Wily Hacker*. Addison-Wesley, 1994.

[7] Dunlap, G. W., King, S., Cinar, S., Basrai, M. A., and Chen, P. M. ReVirt: Enabling Intrusion Analysis Through Virtual-Machine Logging and Replay. In *Proceedings of the 2002 Symposium on Operating Systems Design and Implementation (OSDI)* (February 2002).

[8] Gemmell, J., Lueder, R., and Bell, G. The mylifebits lifetime store. In *ETP '03: Proceedings of the 2003 ACM SIGMM workshop on Experiential telepresence* (New York, NY, USA, 2003), ACM, pp. 80–83.

[9] Grizzard, J. B., Krasser, S., Owen, H. L., Dodson, E. R., and Conti, G. J. Towards an approach for automatically repairing compromised network systems. In *Proceedings of 3rd IEEE International Symposium on Network Computing and Applications* (August 2004), IEEE, pp. 389–392.

[10] Hawkey, K., Muldner, K., and Beznosov, K. Searching for the Right Fit: Balancing IT Security Management Model Trade-Offs. *IEEE Internet Computing* (May/June 2008), 22–30.

[11] Hilzinger, M. Fedora: Chronicle of a Server Break-in. `http://www.linux-magazine.com/linux_magazine_com/online/news/update_fedora_chronicle_of_a_server_break_in`, March 2009. Linux Magazine.

[12] Huang, E. M., Mynatt, E., and Trimble, J. P. Displays in the Wild: Understanding the Dynamics and Evolution of a Display Ecology. In *Proceedings of the* 4^{th} *International Conference on Pervasive Computing* (May 2006).

[13] King, S. T., and Chen, P. M. Backtracking Intrusions. In 19^{th} *ACM Symposium on Operating Systems Principles (SOSP)* (October 2003).

[14] Kursawe, K., and Katzenbeisser, S. Computing Under Occupation. In *New Security Paradigms Workshop* (September 2007).

[15] Neuhaus, S., Zimmermann, T., and Zeller, A. Predicting Vulnerable Software Components. In *Proceedings of the* 14^{th} *ACM Conference on Computer and Communications Security (CCS)* (2007).

[16] Newsome, J., and Song, D. Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploits on Commodity Software. In *Proceedings of the* 12^{th} *Symposium on Network and Distributed System Security (NDSS)* (February 2005).

[17] Ning, P., Cui, Y., and Reeves, D. S. Analyzing Intensive Intrusion Alerts Via Correlation. In *Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID 2002)* (October 2002).

[18] Ou, X., Boyer, W. F., and McQueen, M. A. A Scalable Approach to Attack Graph Generation. In *Proceedings of the* 13^{th} *ACM Conference on Computer and Communications Security (CCS)* (October 2006).

[19] Ozgit, A., Dayioglu, B., Anuk, E., Kanbur, I., Alptekin, O., and Ermis, U. Design of a log server for distributed and large-scale server environments.

[20] Payne, J. W., Bettman, J. R., and Johnson, E. J. Behavioral Decision Research: A Constructive Processing Perspective. *Annual Review of Psychology 43* (1992), 88–131.

[21] Provos, N. Improving Host Security with System Call Policies. In *Proceedings of the* 12^{th} *USENIX Security Symposium* (August 2003), pp. 207–225.

[22] Sheyner, O., Haines, J., Jha, S., Lippmann, R., and Wing, J. Automated Generation and Analysis of Attack Graphs. In *Proceedings of the IEEE Symposium on Security and Privacy* (May 2002).

[23] Singer, A. Tempting Fate. *USENIX login; 30*, 1 (February 2005), 27–30.

[24] Spafford, E. H. The Internet Worm: Crisis and Aftermath. *Communications of the ACM 32*, 6 (June 1989), 678–687.

[25] Stoll, C. Stalking the Wily Hacker. *Communications of the ACM 31*, 5 (May 1988), 484.

[26] Suh, G. E., Lee, J. W., Zhang, D., and Devadas, S. Secure Program Execution Via Dynamic Information Flow Tracking. *SIGOPS Oper. Syst. Rev. 38*, 5 (2004), 85–96.

[27] Wells, G. L., and Olson, E. A. Eyewitness Testimony. *Annual Review of Psychology 54* (2003), 277–295.

Notes

[1]`www.hotadmin.org`

[2]`http://www.edwardtufte.com/tufte/`

[3]`http://freemind.sourceforge.net/wiki/index.php/Main_Page`