Configuration Management and Security

Steven M. Bellovin and Randy Bush

***Abstract*—Proper configuration management is vital for host and network security. We outline the problems, especially for large-scale environments, and discuss the security aspects of a number of different configuration scenarios, including security appliances (e.g., firewalls), desktop and server computers, and PDAs. We conclude by discussing research challenges.**

***Index Terms*—Configuration management, security, database, compilation, Internet network devices.**

I. Introduction

COMPUTER systems require configuration. This is obvious to anyone who has used even the simplest consumer-grade machines, which demand that you supply your name and the computer's name when they are first unpacked. Anyone who has worked with system or network administrators knows that full configuration is far more complex, including the ability — and sometimes the need — to set options that most people don't even know exist.

What is less obvious, even to many system administrators, is that configuration management is a vital part of system and site security. Not only do security devices themselves require configuration, many rather ordinary aspects of life with computers (e.g., the fact that laptops are sometimes turned off and hence can't be reconfigured) have implications for security.

Security configuration has another unique feature: it has strong interactions with business and personnel policy. Many security decisions are concrete instantiations of management decisions. Incorrect configuration can endanger business relationships or have legal ramifications.

Conversely, by definition security involves dealing with enemies. That is, someone will try to counter your moves. Managing security configuration is not simply a matter of designing the right configuration and distributing it, contending only with Murphy's Law; instead, the administrator must contend with active attempts to subvert configurations, evade them, or drive the system into an improbable state not anticipated by the configuration. Note well that "enemies" can include a site's own employees.

This, then, is the challenge of security configuration management: to devise and deploy policies that carry out organizational requirements — including, of course the ability to get work done[1] — despite lack of cooperation from many insiders, and active opposition both from outsiders and from honest-but-frustrated insiders.

A. Paper Structure

We start by examining the security implications of configuration management at scale. Large-scale systems are always different; this section examines how this is true for security configuration.

The heart of the paper explores security configuration management for five different scenarios: security appliances, network infrastructure, ordinary user computers, servers, and PDAs. Each of these pose different configuration management challenges, and of course any of these problems are worse for large-scale sites.

We follow with a real-world case study, examining how a major ISP used a database-driving configuration scheme, for both security and correctness reasons.

We conclude with a few parting thoughts, including suggestions for future research directions.

II. Challenges of Large-Scale Systems

A. The Problem of Scale

Often, a difference in scale is qualitative, not simply quantitative. Managing the configuration of 100 machines is a different problem than managing one or two; managing 1000 is different still.

It's tempting to say that that isn't true, that each individual machine owner can manage its configuration. That is, we are not dealing with 1000 machines as a group, we instead have 1000 problems we already know how to solve. There are at least four problems with this approach.

The first is that individually-administered machines are often poorly administered. Most computer users are not professional system administrators. Just as many home machines are misconfigured, from a security perspective and an ordinary correctness perspective, doing the same in a corporate environment will result in very many misconfigured machines.

The second problem is consistency. In many environments, enterprise configurations are centrally specified. Leaving the configuration up to individuals will inevitably result in misconfigurations, ranging from choice of software — did everyone install the mandatory corporate applications? Will they be updated? — to certain security settings that the machine users may be unaware of or even resent. To give just one example, many companies disable USB mass storage devices, to make it harder for someone to exfiltrate sensitive corporate data.

Together, these raise a third problem: interactions. That is, two configurations that by themselves are correct can interact in dangerous ways. Martin et al. give an example where

Manuscript received 4 April 2008; revised 6 December 2008.

Steven M. Bellovin is with the Department of Computer Science, Columbia University, New York, NY (e-mail: smb@cs.columbia.edu).

Randy Bush is with the Internet Initiative of Japan, Tokyo, JP (e-mail: randy@psg.com).

Digital Object Identifier 10.1109/JSAC.2009.090403.

[1] For another approach, see http://www.dilbert.com/strips/comic/2008-04-04/

0733-8716/09/$25.00 © 2009 IEEE

correct Java behavior and correct firewall behavior for the FTP protocol [1] combine to produce a security hole [2].

Finally — and this point is often ignored by those not in the system administration business — many machines (and in some environments, most machines) are not ordinary desktop PCs. Servers, virtual machines, and even many embedded systems require configuration management, too. Network-connected printers can be hacked as easily — and in some cases more easily — than ordinary desktop PCs, and may in fact run some of the same software.

Some form of centralized configuration management is necessary, then, both for correct operation and security. While there are many ways to accomplish this, a number of requirements must be satisfied:

Security The configuration management scheme must be secure, in many senses of the word. Its databases must be protected against unuathorized access or modification, the communications with the managed hosts must be authenticated, integrity-protected, and often encrypted; the commands to the managment system must be properly authenticated.

Database-Driven Different machines have different roles; as such, they require different configurations. A desktop machine may have no need for a web server; depending on the user's needs, though, it may need a database server. Other machines — say, mail servers — may need neither, but may need other specialized software.

Robustness There *will* be configuration failures. Machines may be down during attempted updates; other, unmanaged changes may have occurred; changes — specifically including vendor-supplied patches — may fail to install. If the configuration changes are security-relevant, this could have serious consequences.

B. It's Turtles All the Way Down

It is clear from the above description that managing the configuration of a large number of machines requires effort: people, software, and data. What is less obvious is that configuration management systems themselves require management, and perhaps their own layer of management systems.

For one thing, in a large-enough enterprise a single configuration management system is insufficient. Not only can it not handle the load, a secondary machine is needed for redundancy.

In a geographically distributed environment, database maintenance at least must be geographically distributed. *Every* new computer's configuration needs must be entered into the database; this is hard to do when the administrators are unaware of the very existence of the machine, let alone its role and needs. For that matter, in many environments it is necessary to conduct audits, manual and electronic, to detect new machines and enroll them appropriately in the configuration management system.

Finally, configuration needs change. New software is obtained, new applications are developed, etc. Who, 10 years ago, would have anticipated the need to configure desktop machines with the identity of the corporate IM server? Today, forward-looking IT managers are planning for corporate SecondLife® proxies.

III. Scenarios

Different configuration scenarios have different needs. It is instructive to examine them in detail. Note that these needs exist regardless of the existence of configuration management systems; however, as will become clear, some of the problems described are much more serious in large-scale environments.

A. Security Appliances

The most obvious class of computers with security-sensitive configurations is security appliances: firewalls, filtering routers, etc. Such devices not only provide security for the rest of the enterprise, they have stringent security needs of their own.

The hardest problem in this environment is independent of the mechanics of the configuration. Rather, it is what the configuration should be. It in turn depends on the network topology and the desired security policy. Neither is obvious; both change. Security policies, for example, change in response to new software, new business needs, and new threats. A joint venture with some other company will frequently change both: dedicated links may be set up, and firewall rules must be adjusted to permit access to some but not all resources. (The alternative to a new, dedicated link is a virtual private network connection. These are generally cheaper but carry their own set of security-sensitive configuration needs.)

Complex firewall configurations are known to be error-prone [3]. This is likely to remain true even if fully distributed firewalls are used [4]; those solve the topology problem but not the policy issue.

Given a proposed firewall policy, it is important to verify it. While automated tools (see, for example, [5], [6]) can detect certain classes of errors, such tools cannot differentiate between a policy that is plausible and a policy that reflects the actual desire. That is, there is no a priori difference between allowing port 80 access to the official corporate web server and allowing port 80 access to a sensitive internal web server; an automated algorithm cannot distinguish between the two cases. Verifying topology changes is even harder [7].

Ultimately, both problems are more complex in geographically dispersed organizations. It is hard for a central system administrator to know which servers should be accessible to which outside parties, let alone when new links have been installed to some far-off location.

Security appliances themselves need security protection [8]. Maintaining such configurations is itself a difficult problem. Indeed, one recent survey [9] reported that ISPs called management of access control lists (ACLs) their "most critical" problem. Recent work on automatic generation of ACLs [10] may help.

B. Infrastructure

Even ordinary infrastructure nodes (i.e., those with no special security role) have security-sensitive configurations. If nothing else, an adversary who controls these elements can disrupt connectivity throughout the organization. A more sophisticated attacker can reroute traffic through an eavesdropping node. Correct configuration — that is, correct passwords, correct access control lists, suitable logging, etc., can go a long way towards preventing such attacks.

Logging functions on switches are important when tracing certain kinds of attacks. In particular, logging MAC addresses and traffic volumes can help isolate the source of denial of service attacks and address spoofing attacks (see [11], [12]). Again, this is configuration that must be managed.

Even without malicious attacks, maintaining router configurations is difficult. When something goes wrong with routing — and that is not at all improbable, in complex configurations — technicians who repair the problem will often forget to update the master configuration database. To this end, some sites periodically poll routers for their configurations, comparing them to the officially-authorized versions [13]. In effect, this acts like a Tripwire system ([14], [15], [16]) for routers.

The harder problem, though, is conflicting security policies. The issue isn't so much technical (i.e., see [17] or [18]) as it is political: whose policy should dominate? That is, given a disagreement between overall corporate policies and local needs, which should control? Technical mechanisms can identify the problem but not solve it. We note here that the decisions made can affect both the security of the corporate network and the productivity of local business units.

C. Desktops and Laptops

Desktop and laptop computers present a more familiar security configuration challenge. The administrator must ensure that proper programs, patches, and anti-virus update are installed. However, many of the issues we have seen before occur in this space as well, including policy conflicts. The dimensions of the problem differ, however.

The first issue is that there is a new player: the authorized user of the machine. Rightly or wrongly, individuals with personal computers regard them as personal machines, and frequently use them for unofficial purposes. Frequently, this includes ignoring official policies [19]. Such misbehavior can extend to the very highest ranks [20]. Mechanisms for ensuring desktop and laptop configurations must account for employee non-cooperation and even willful misbehavior.

A second issue is that the timing of patch application must be controlled. Patches — even security patches — are often incompatible with essential applications. Proper configuration, then, must be capable of either preventing normal vendor patches from being installed (see, for example, the Microsoft tool that allowed corporations to delay installing Service Pack 2 to Windows XP [21]) or forcing installation if there is an exigent security threat [22].

Beyond that, laptops and corporate-owned home desktop machines are often inaccessible. Hotel and home firewalls and Network Address Translator (NAT) boxes almost always prevent connections into such machines; in many cases, they block outbound connections to the corporate configuration server. (We note with amusement that such a server must frequently be located outside the corporate firewall, to permit upgrades by machines too insecure to be allowed within the firewall.)

D. Servers

Servers, on the other hand, are much simpler in many ways. The owner is rarely the adversary, they are generally quite accessible, etc. Patch timing must still be controlled; additionally, they suffer from a different problem: heterogeneity.

Desktop machines within an organization tend to be quite similar. Most people need most of the same applications; unused applications that do not contain privileged programs or network listeners are rarely a major problem. Servers, on the other hand, differ widely. Web server machines need a web server daemon, of course; depending on the nature of the web pages being served, they may also need databases, Perl or PHP interpreters, Wikis, etc. Secure configuration of any of these is tricky; a configuration management system (or person) for servers must be able to handle any and all of these. That is, it must first know which components apply to which servers; second, it must know how to combine the necessary security configuration aspects with the server-specific parameters. For example, a web server must be configured with things like a document root directory, policies regarding execution of CGI or ASP scripts, keys and certificates, access control policies, etc. Almost always, these will be different on different servers, but they are all quite critical for security.

E. PDAs and Phones

The newest configuration challenge for organizations is the PDA or "smartphone". These devices are often personally owned, but used for business purposes, such as checking email. Some, such as Apple's iPhone, were not initially designed with corporate needs in mind [23]:

> Many IT groups have banned the iPhone from their workplaces, complaining that there is no way to force employees to protect their iPhones with passwords and that they can't erase sensitive corporate data from remote locations if the device is stolen or lost. Additionally, they say the iPhone doesn't support the software many businesses use and that it only works on one cellular carrier's network.

It is not clear yet just how the configuration of such devices will be managed. It is clear that as they gain access to corporate networks, such management will be necessary.

IV. A Real-World Case Study

A few large ISPs recognized many of these issues long ago and built their operations support systems accordingly (Figure 1). Their systems rested on three major principles: databases, compilation, and monitoring.

A. Databases

The first principle, as suggested earlier, is that all configuration should be database-driven. All relevant information, whether it is the physical hardware in any location or administratively-assigned parameters such as the IP addresses assigned to a given customer, are recorded in a database. *All* changes must be recorded in the database; if the actual, physical configuration differs, then by definition it is wrong, not the database. The contrasting — and all too common —

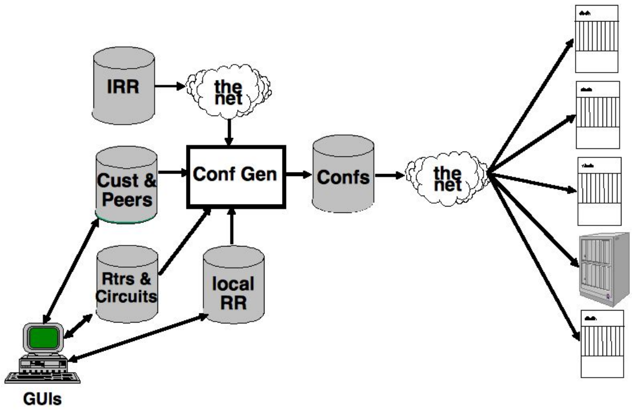

Fig. 1. Overall data flow for a database-driven configuration system.

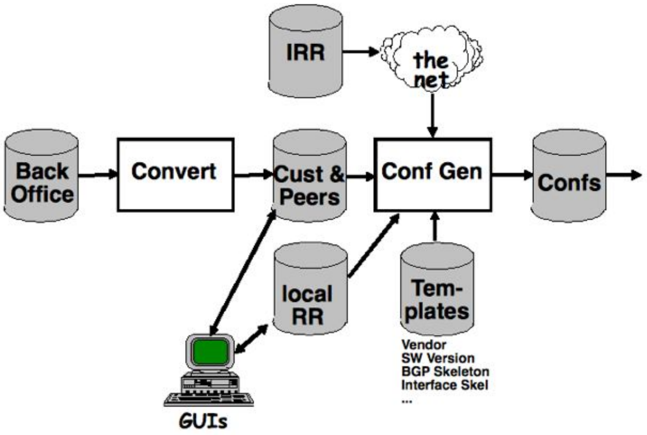

Fig. 2. Generating configuration.

alternative, the notion that "the network is the database of record", was explicitly rejected.

This approach has many advantages. Perhaps the most important is correctness. For one thing, the database represents an *engineered* approach to design: physical circuits that were selected to meet some higher-level goal, such as path diversity, will be used as intended. Additionally, various data items need only be entered once; there is no chance of inconsistency if two or more copies of some item disagree.

Another important advantage was long-term cost savings. While it certainly costs something to build, operate, and administer this infrastructure, it saves on operational expenses: hand-built configurations are time-consuming and error-laden; correcting such errors is itself an expensive proposition.

An important security advantage is responsiveness: if a security incident is occurring, new configurations can be pushed out to all nodes very quickly. For example, the Slammer worm was stopped when many ISPs blocked UDP port 1434 via access control lists on their routers. Distributing such ACLs, in a system like this, is simply a matter of changing the standard inbound packet ACL in one place and initiating the distribution. Even new firmware images can be sent out as needed.

The databases (Figure 2) contain

- device descriptions for routers etc. with models, versions, etc.
- interface descriptions
- circuit descriptions and bindings to interfaces
- IP address allocations bindings to interfaces and customers
- BGP peering relationships to customers and peers
- DNS delegations to customers

The data are generated and maintained by many units of the organization, from circuit management, asset management (routers), routing engineers, IP allocation engineers, etc.

B. Compilation

The compilation process is similar to that for ordinary programming languages; however, the programs themselves are very different. In this environment, there is a separate "main program" called a *device list* for each device type, such as a router or switch. In addition, because different models of device from different vendors must be supported, there are configuration templates for devices (e.g. a Juniper M5 configuration skeleton). There are also templates for snippets of specific configuration details such as BGP peerings, Packet over SONET (POS) interfaces, etc. The device lists drive the generation process; it in turn finds the BGP peerings, interface details, etc. in the databases, and uses those data to drive the appropriate snippet templates to generate the configurations.

A sample code segment containing a template is shown in Figure 3. Note that, due to proprietary issues (network configuration is seen as a non-trivial competitive advantage), this is not from a major ISP but from the private code of one of the authors' small research networks. Hence the database and code are degenerate examples of a large scale system. Note the configuration generation loops over all peers in the peer table in Figure 4. (Loops and conditionals at compile-time are not, of course, new; IBM's PL/1 compiler and Basic Assembly Language for the Systems/360, among others, had such things more than 40 years ago.) When run against the database shown in Figure 4, it produces output beginning:

```
neighbor 191.42.180.10 {
  peer-as 6666;
  import [ no-bogons dampening peer-6666 ];
  }
neighbor 191.42.180.12 {
  peer-as 1234;
  authentication-key "fnardle";
  import [ no-bogons dampening peer-1234 ];
  }
neighbor 191.42.180.13 {
  peer-as 1243;
  authentication-key "fnoofle";
  import [ no-bogons dampening peer-1243 ];
  }
neighbor 191.42.180.22 {
  peer-as 1423;
  authentication-key "fnuffle";
  import [ no-bogons dampening peer-1423 ];
  }
```

as well as massively long access control lists (ACLs) used to filter prefixes allowed from each peer.

```
printf (POLICY "policy-statement peer-%d {\n", $AS );
printf (POLICY "  term ok {\n");
printf (POLICY "    from {\n");
foreach $PREFIX (@PFX) {
  printf (POLICY "    route-filter $PREFIX exact;\n");
  }
printf (POLICY "      }\n");
printf (POLICY "    then accept;\n");
printf (POLICY "    }\n");
printf (POLICY "  term final {\n");
printf (POLICY "    then reject\n");
printf (POLICY "    }\n");
printf (POLICY "  }\n");

printf (PEERS " neighbor %s {\n", $IP);
printf (PEERS " peer-as $AS;\n");
if ( $MD5 ne "NONE" ) {
  printf (PEERS " authentication-key \"%s\";\n", $MD5);
  }
printf (PEERS " import [ no-bogons dampening peer-%d ];\n",
  $AS);
printf (PEERS " }\n");
```

Fig. 3. A code template segment for a Juniper router. This one is actually a Perl script invoked by the compiler.

ASN	IP Address	IRR	AS-MACRO	MD5 Key
6666	191.42.180.10	CW	AS666	NONE
1234	191.42.180.12	VERIO	AS-BIGONE	fnardle
1243	191.42.180.13	RADB	AS-CATS	fnoofle
1423	191.42.180.22	EPOCH	AS-DOGS	fnuffle

Fig. 4. Excerpt from Peering Database

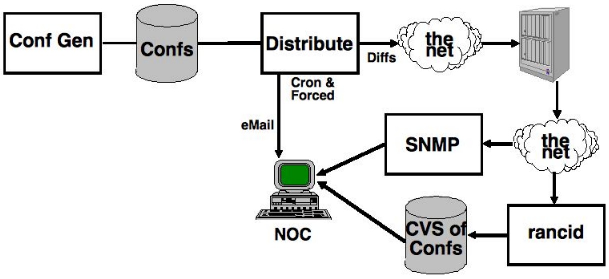

Fig. 5. Distribution and monitoring structure.

In addition to provider-specific databases, public databases, such as the Internet Routing Registry (see http://www.irr.net/), are used as a source of information about network topology and prefix allocation to create routing filter ACLs. Because this is too complex to do manually, those providers who do not automate do not do proper filtering. This leads to incidents such as the YouTube/Pakistani accident, where a singe provider's mistake caused routing trouble for much of the Internet.

Many devices have operational idiosyncrasies that must be accomodated. For example, some devices reset some or all subsystems when asked to fully reload a configuration; e.g., many Cisco routers reset BGP sessions even though nothing about BGP has changed). Consequently, the generation system usually makes a "diff" file with only the changes that need to be pushed to the device.

As the push to the device could have surprising or even unhappy results, it is customary to first email changes to a human a few hours in advance of changing the device configurations. This allows for last-minute corrections. This process is usually done on daily basis, but the periodicity and timing varies between providers and across geography and topology. A large provider will tend to roll changes out slowly across their network, often over a number of hours.

Occasionally, an emergency fix has to be pushed to one or more devices. To that end, the system has controls to cause an immediate rebuild and push. The engineer changes the database entries to represent the changes desired, presses the button to generate the new configuration, hopefully checks it for errors, and then tells the system to push it to the device.

Although we do not know of any ISPs who actually do this, using a compilation-based approach makes it easy to use more sophisticated mechanisms to verify and build a configuration. Narain et al. [24] describe use of a requirement solver to build configurations. Another possibility is construction of optimizing compilers. For example, the scheme described in [10] could be adopted. It would be very interesting to incorporate network routing metric optimizations à la Fortz et alia [25].

C. Monitoring

When device configurations are fully generated, the canonical version of the configuration is on the management system, not the device (Figure 5). This means that the device configuration should not change unexpectedly. To verify this, there are tools, e.g., `rancid`, (http://shrubbery.net/rancid/) which are used to pull the running configuration from the device on a periodic basis (traditionally once an hour), archive it, and diff it from the previous and/or canonical configuration. Any significant deviations usually cause email to be sent to engineers. The usual cause of such deviations is an over-eager technician; however, this monitoring system can also catch changes made directly to devices by outsiders.

The simple use of `rancid` to gather and diff running configurations from all devices is not necessarily tied to a network database or configuration generation system. Therefore, while thorough network configuration is done by very few ISPs, `rancid` or its equivalent is used by most ISPs above trivial size.

A change made by an outsider is, of course, a form of intrusion. This means that standard intrusion detection and tracing techniques can be applied. The IP address of the machine from which unauthorized changes were made can be traced. (This assumes, of course, that the network element or the console server connected to it do sufficient logging, itself an important security requirement.) Furthermore, the differences between the authorized configuration and the one installed by the intruder can give valuable clues as to goals and motives.

Database-driven configuration can also *reduce* operational alerts: the database can serve as input to a root cause analysis system. Root cause analysis is a difficult enough problem to start with; trying to do it without accurate knowledge of the precise running configuration is likely impossible. Using a single database both to generate element configurations and as input to the analysis system is much more likely to succeed.

One interesting side effect of a system rigorously driven from a data base is that security and other audits and validations can be done against the database, as opposed to scraping and digesting raw device configurations. If nothing else, this means that the analyzer knows the precise meaning of each datum; there is no need for the analysis engine to guess where some value came from. Suppose, for example, that an actual router configuration's source address filter ACL ([11], [26]) lists an address block different from the one assigned to that customer, per the routing ACL. Is that due to a typographical error, or is it because the customer has more than one address block, but can only advertise one of them to the router? In a database-driven configuration system, the answer would be unambiguous.

As the software toolbase has clear definitions and details of all devices, interfaces, interface configurations, etc., it can also generate and help maintain changes in the database used to control the real-time monitoring stsrems used by a netork operations center (NOC). Custom monitoring systems can directly access the data in the network database, or the needed data for externally developed (free or public, open or closed source) software systems can be generated by glue-ware from the internal network database.

V. Conclusions

It is clear that configuration, and hence configuration management, are crucial to security. We assert that it is equally clear that even in moderate-size environments, it should not be done by hand. Too much can go wrong.

Three elements are necessary for successful management of security configuration: clear policies; accurate knowledge of all computers and network elements; and proper management software. To be sure, all of these are needed for other types of configuration management. However, in a security setting, a failure can have consequences far beyond the failure of one device or system.

In other words, for organizational security, configuration management is vital. However, we do not know how to do this. The challenges, and hence the need for research, are severalfold.

First, we must understand what configurations should be like, and how they should be set. This is not just a question of file contents and the like; the human element — how people understand and specify configurations — is at least as important.

Second, we need to be able to abstract and parameterize configurations. That is, we need to be able to create meta-configurations, and merge these with the knowledge of the machines that actually exist.

The difficulty of doing these raises a third issue: can we create configuration mechanisms that are more amenable to such specification? Today's systems were designed for hand configuration, with few nods to automation. Can we do better?

Fourth, we need effective ways to understand exactly what our networks really consist of. Note that this needs to be done in ways that protect privacy, not so much for business reasons as because often, consumer-grade machines are used for business purposes. It may be acceptable for corporate machines to respond to "who's out there, and what are your capabilities" messages; it certainly is not acceptable in a consumer environment.

Finally, we need to build robust, secure systems that implement all of the above. As the cost of developing such systems is far more than a medium or small network provider can afford, and seems to be beyond the means of the majority of large providers, it would be a public good if one or more such systems were available as open source.

References

[1] J. Postel and J. Reynolds, "File transfer protocol," Internet Engineering Task Force, RFC 959, Oct. 1985. [Online]. Available: http://www.rfc-editor.org/rfc/rfc959.txt

[2] D. Martin, S. Rajagopalan, and A. D. Rubin, "Blocking Java applets at the firewall," *Proc. Internet Society Symposium on Network and Distributed System Security*, pp. 16–26, 1997.

[3] A. Wool, "A quantitative study of firewall configuration errors," *IEEE Computer*, vol. 37, no. 6, pp. 62–67, 2004.

[4] S. M. Bellovin, "Distributed firewalls," *;login:*, pp. 39–47, November 1999.

[5] A. Mayer, A. Wool, and E. Ziskind, "Fang: A firewall analysis engine," in *Proc. IEEE Symposium on Security and Privacy*, 2000, pp. 177–187.

[6] ——, "Offline firewall analysis," *International Journal of Information Security*, vol. 5, no. 3, pp. 125–144, 2006.

[7] B. Cheswick, H. Burch, and S. Branigan, "Mapping and visualizing the internet," in *Usenix*, 2000. [Online]. Available: http://www.cheswick.com/ches/papers/mapping.ps.gz

[8] "Hackers hijacking routers and blackmailing firms to regain access," *NetworkWorld*, 2 April 2008, http://www.networkworld.com/community/node/26129.

[9] *Worldwide Infrastructure Security Report*. Arbor Networks, 2007, vol. III. [Online]. Available: http://www.arbornetworks.com/report

[10] E. Sung, S. Rao, G. Xie, and D. Maltz, "Towards systematic design of enterprise networks," in *Proc. ACM CoNEXT Conference*, Madrid, Spain, December 2008. [Online]. Available: http://faculty.nps.edu/xie/papers/ent-design-conext08.pdf

[11] P. Ferguson and D. Senie, "Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing," Internet Engineering Task Force, RFC 2827, May 2000. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2827.txt

[12] S. M. Bellovin, "Security problems in the TCP/IP protocol suite," *Computer Communications Review*, vol. 19, no. 2, pp. 32–48, April 1989. [Online]. Available: http://www.cs.columbia.edu/~smb/papers/ipext.pdf

[13] "RANCID — Really Awesome New Cisco confIg Differ," 2007, http://www.shrubbery.net/rancid/.

[14] G. Kim and E. H. Spafford, "The design and implementation of Tripwire: A file system integrity checker," in *Proc. 2nd ACM Conference on Computer and Communications Security*, November 1994. [Online]. Available: http://ftp.cerias.purdue.edu/pub/papers/gene-kim/Tripwire.pdf

[15] ——, "Writing, supporting, and evalutaing tripwire: A publically available security tool," in *Proc. Usenix* UNIX *Applications Development Symposium*, 1994. [Online]. Available: http://www.usenix.org/publications/library/proceedings/appdev94/kim.html

[16] ——, "Experiences with Tripwire: Using integrity checkers for intrusion detection," in *Proc. Systems Administration, Networking, and Security III*, 1994. [Online]. Available: http://ftp.cerias.purdue.edu/pub/papers/Tripwire/Tripwire-appdev.pdf

[17] A. Russo, R. Miller, B. Nuseibeh, and J. Kramer, "An abductive approach for analysing event-based requirements specifications," in *Proc. 18th International Conference on Logic Programming*, 29 July-1 August 2002. [Online]. Available: http://www.doc.ic.ac.uk/~ar3/iclp2002.pdf

[18] A. Bandara, E. Lupu, and A. Russo, "Using event calculus to formalize policy specifications and analysis," in *IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, June 2003. [Online]. Available: http://www.doc.ic.ac.uk/~ar3/EventCalculusPolicy2003.pdf

[19] J. E. Dunn, "Study: Lax laptop policies create security concerns," *Computerworld*, November 1 2004. [Online]. Available: http://www.computerworld.com/securitytopics/security/story/0,10801,97094,00.html

[20] "Report of investigation: Improper handling of classified information by John M. Deutch," Central Intelligence Agency Inspector General, 1998-0028-IG, February 18 2000. [Online]. Available: http://www.fas.org/irp/cia/product/ig_deutch.html

[21] J. Leyden, "Eight patches lined up for MS April patch batch," *Channel Register*, 8 April 2005. [Online]. Available: http://www.channelregister.co.uk/2005/04/08/ms_april_patch_preview/

[22] S. Beattie, S. Arnold, C. Cowan, P. Wagle, C. Wright, and A. Shostack, "Timing the application of security patches for optimal uptime," in *Proc. USENIX 16th Systems Administration Conference*, 2002. [Online]. Available: http://www.cse.ogi.edu/~crispin/time-to-patch-usenix-lisa02.ps.gz

[23] B. Worthen, "Why IT hates the iPhone," in *Wall Street Journal*, March 31 2008, p. R4. [Online]. Available: http://online.wsj.com/article/SB120647580478363231.html

[24] S. Narain, S. Malik, G. Levin, and V. Kaul, "Declarative infrastructure configuration synthesis and debugging," *Journal of Network Systems and Management, Special Issue on Security Configuration*, vol. 16, no. 3, September 2008. [Online]. Available: http://www.argreenhouse.com/papers/narain/DeclarativeConfiguration.pdf

[25] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional IP routing protocols," *IEEE Commun. Mag.*, vol. 40, pp. 118–124, 2002.

[26] F. Baker and P. Savola, "Ingress filtering for multihomed networks," Internet Engineering Task Force, RFC 3704, Mar. 2004. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3704.txt

Steven M. Bellovin is a professor of computer science at Columbia University, where he does research on networks, security, and especially why the two don't get along. He joined the faculty in 2005 after many years at Bell Labs and AT&T Labs Research, where he was an AT&T Fellow. He received a BA degree from Columbia University, and an MS and PhD in Computer Science from the University of North Carolina at Chapel Hill. While a graduate student, he helped create Netnews; for this, he and the other perpetrators were given the 1995 Usenix Lifetime Achievement Award (The Flame). He is a member of the National Academy of Engineering and is serving on the Department of Homeland Security's Science and Technology Advisory Committee; he has also received the 2007 NIST/NSA National Computer Systems Security Award.

Bellovin is the co-author of "Firewalls and Internet Security: Repelling the Wily Hacker", and holds several patents on cryptographic and network protocols. He has served on many National Research Council study committees, including those on information systems trustworthiness, the privacy implications of authentication technologies, and cybersecurity research needs; he was also a member of the information technology subcommittee of an NRC study group on science versus terrorism. He was a member of the Internet Architecture Board from 1996-2002; he was co-director of the Security Area of the IETF from 2002 through 2004.

Randy Bush is a Senior Researcher and Network Operator at Internet Initiative Japan, the first commercial IPv6 deployment in the world. He specializes in IPv6 deployment, network security, protocols, and network measurement especially routing. Randy has been in computing for 45 years, and has a few decades of Internet operations experience. He was the engineering founder of Verio, which is now NTT/Verio. He has been heavily involved in transferring Internet technologies to developing economies for over 20 years.

He was a chair of the IETF WG on the DNS for a decade and served as a member of the IESG, as co-chair of the IETF Operations and Management Area for six years. Randy was the first Chair of the NANOG Steering Committee, a co-founder of AfNOG, on the founding Board of Directors of ARIN, helped start AfriNIC, and has participated in APNIC, RIPE, et alia since each was founded.