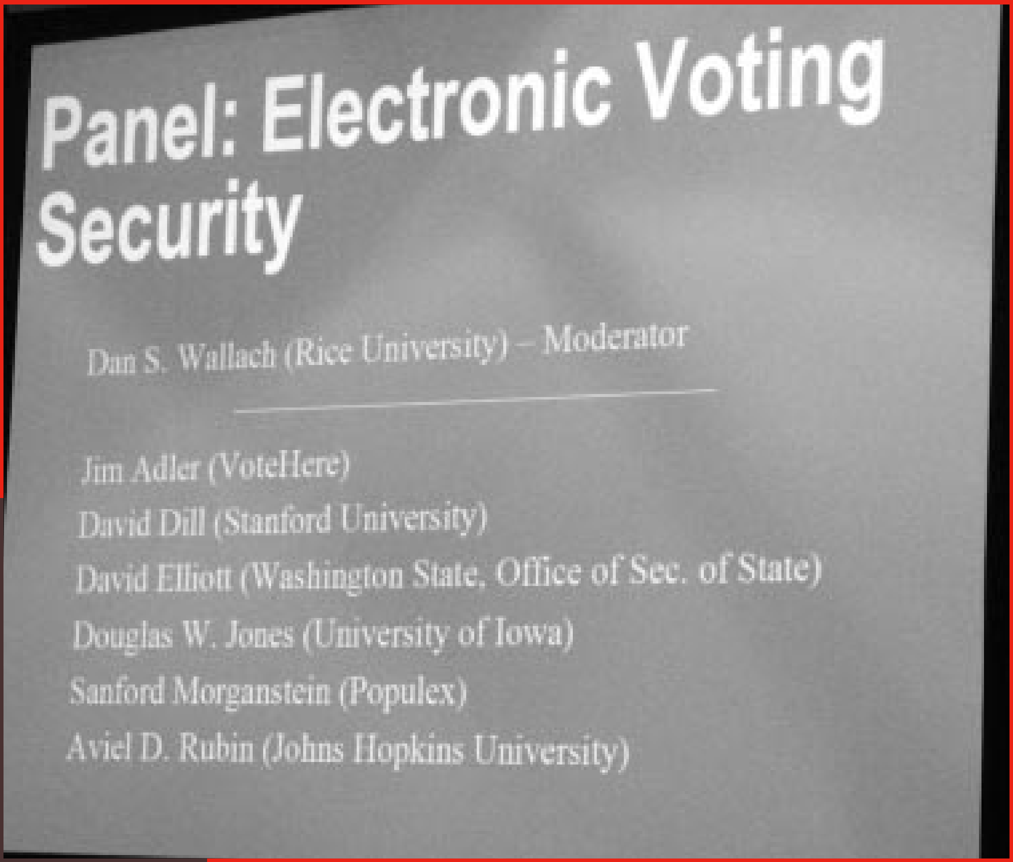
;login:

THE USENIX MAGAZINE

December 2003 • volume 28 • number 6



Panel: Electronic Voting Security

Dan S. Wallach (Rice University) – Moderator

Jim Adler (VoteHere)
David Dill (Stanford University)
David Elliott (Washington State, Office of Sec. of State)
Douglas W. Jones (University of Iowa)
Sanford Morganstein (Populex)
Aviel D. Rubin (Johns Hopkins University)

inside:

Focus Issue: Security

Guest Editor: Rik Farrow

USENIX
The Advanced Computing Systems Association

life without firewalls

by Abe Singer

Abe Singer is a computer security manager at the San Diego Supercomputer Center, and occasional consultant and expert witness. He is responsible for operational security for SDSC, involved with the Teragrid security working group, and is doing research in security measurement and logfile analysis"

Abe@SDSC.edu

Here at the San Diego Supercomputer Center (SDSC) we recently went almost four years without an intrusion on our managed machines.[1] And it was some time between that one and the previous one.

We have a decent-sized organization – a few thousand machines, over 6000 users worldwide. We think we do a pretty good job of keeping our systems secure.

And here's a little secret: We don't use firewalls.

Why not? Well, because, basically, they're not worth it, and they don't provide us with the protection we need. Because it's about *effective* security – solutions which actually work against actual threats, which scale, and which are robust.

This article outlines our strategy for effective security. While not all of it is applicable in all environments, I hope some (maybe a lot) of it is useful to others.

Space limitations prevent me from going into detail about our implementation. Expect more in future articles.

1. We do have "unmanaged" (user-managed) machines that have been compromised. This will be explained in more detail below.

The Myth of Firewalls

There is a pervasive myth that firewalls are necessary for effective security. Firewalls have become a panacea, and are assumed to magically protect everything. The net result is often that a network is *less* secure.

"Use a firewall" is a common recommendation from security literature and practitioners. VISA's Cardholder Information Security Program requires online merchants to "install and maintain a firewall."[2]

2. VISA Cardholder Information Security Program, *http://usa.visa.com/business/merchants/cisp_index.html*.

The myth that a firewall is necessary for effective network security is so prevalent that many believe you are doing something wrong if you don't have one.

For example, I was helping a private research lab construct a comprehensive security plan, which focused on infrastructure protection. They hired a new CTO, who informed us that he wanted a firewall, because whenever he discussed security with his peers at other organizations, they were incredulous that he did not have one, and he felt that his reputation was suffering as a result. Some time later, at a technical staff meeting, the issue of security came up, and he announced that they were now secure because they had a firewall. (They weren't, and some of the staff called him on it.)

I used to be a full-time consultant, and I can't tell you how many times a customer (or potential customer) would say something similar: "We're safe – we have a firewall."

But firewalls *aren't* the magical bullet-proof vest that the public seems to think they are. Just look at the spread of worms throughout the Net: Code Red, Blaster, etc. Look at Web sites that have been defaced by vandals, or had credit card or social security numbers stolen. Many of these compromised sites had firewalls.

Why does the myth persist? First, firewalls are sexy. It's much c00l3r to play with a firewall than to patch machines. Second, we live in the "magic pill" culture, where we look for a one-time quick fix, and management wants to hear that the problem is solved.

Why Firewalls Don't Work

So what is a firewall? Early firewalls were just packet-filtering routers, which could only filter IP addresses (not port numbers). Then came port filtering, proxy firewalls, and stateful packet filtering (SPF). To some people, using NAT is considered a firewall.

A firewall works only as well as it's configured. A firewall which allows traffic through on every port provides no protection whatsoever. One which does not allow any traffic through will probably protect against external attacks, but isn't otherwise useful.

A firewall works only as well as it's configured.

So a firewall has to allow some traffic through. And if the machine receiving the traffic behind the firewall has a vulnerability, it can be compromised.

Now at this point the firewall-o-philes will be saying, "Oh, but a proxy firewall would solve that problem" or "Stateful packet filtering is the answer." Those technologies also only work as well as they're configured or programmed.

A proxy has the *capability* to examine the data and only allow "legitimate" data through, but it has to be programmed to differentiate between what is legitimate and what is not. And that program has to be maintained, bugs have to be fixed, and so on.

And you still have to protect the service against attacks from other hosts on the same network *behind* the firewall.

Firewalls which rely on chained rule sets are vulnerable to cascade failures – a change in one rule can have an effect on every rule which follows it. I've seen systems that relied on a firewall to block services which were only supposed to be available on the local network, but which were made available to the entire Internet due to the unforeseen result of a firewall rule change.

Firewalls also have performance issues; they usually only work if all traffic is directed through them, and usually not at the speeds found, for example, on OC-192 connections. Firewalls do not work well or at all in a high-speed network with multiple paths.

Finally, having a firewall should not be used as an excuse for poor system administration.

Now, having said all that, I'm going to back down a little and say that firewalls do have their uses. They're just not the magic pill that people think.

Lack of Metrics

The truth is we don't have any real scientific measurement of how effective different security practices are. There's no data that shows that, to make up an example, firewalled sites have 34.5% fewer compromises than those without. There's no data that shows how much longer it will take, on average, to compromise a "secure" system configuration vs. a default "out of the box" configuration.[3]

3. Marcus Ranum's cat may have some numbers.

(We could really settle some religious wars if we *did* have such data.)

SDSC's Approach

History

Like almost everyone, we had to feel the pain before getting smart about security. About 10 years ago, SDSC had recurring problems with intruders. As fast as an intruder could be kicked out, another one got in. Finally, we decided to take drastic measures: We shut down the entire network. Machines were reinstalled and not allowed back on the network until they were considered secure. Our Cray was down for three weeks, the longest it had ever been down (before *or* after). Our director, Sid Karin, said, "I never want to do this again." Meaning that whatever we did should work over the long term, not just be an immediate fix.

A security policy and strategy should be based on a realistic risk analysis.

Sid also said that we should have an open environment, where people can do what they want to do. (He actually said that if someone wants root access, why not give it to them?) But he recognized that one person's actions on one machine can have an impact on other machines on that network.

So we developed a long-term strategy to keep our machines and network secure, while providing resources that are open and usable. We did not do everything immediately. Rather, we implemented what we could, and improved things as new technologies became available. We also recognized that there are people who just want to do their own thing, either because they have a need that doesn't fit with our environment or because they have an ego problem, so we also provided a way for those people to manage their own resources.

The SDSC Environment

SDSC is a facility which provides supercomputing resources for scientific research, and does research in high-performance computing technology.[4] SDSC is also part of TeraGrid and Internet2. SDSC does not do any government-classified work.

4. The San Diego Supercomputer Center, *http://www.sdsc.edu*.

We have about 6000 users, of which only about 300 are on-site. The rest are at other institutions around the globe (mostly in the US).

We have several thousand systems on-site and about five petabytes of near-line storage, plus several hundred terabytes of spinning disk on a SAN. Our network supports 10Gb Ethernet (no, that's not a typo) internally and to other TeraGrid sites. We have multiple OC-192 connections to various sites. In other words, LOTS of bandwidth.

We do not insist on absolute homogeneity (more on that below). On-site users get their choice of desktop: Linux, Solaris, Windows, or Macintosh. Many users have more than one desktop machine. Our infrastructure machines are a combination of Linux and Solaris. Oh, and we have some IRIX machines used by the Visualization group.

We currently support the following OS revs:

RedHat 7.2, 7.3, 8, 9; Solaris 7, 8, 9; IRIX 6.5; Windows NT4, 2000, 2003, XP (for some specific applications); MacOS 9, X

We treat hardware as commodity devices. System configuration is independent of the hardware. If a system dies, it is replaced with new hardware and auto-installs. User downtime is kept to a minimum, as are support staff resources.

Risk Analysis

A security policy and strategy should be based on a realistic risk analysis. Our analysis looks at the assets we are trying to protect and the threats to those assets.

ASSETS

To SDSC, the most important thing to protect is the integrity (and, where necessary, the confidentiality) of our data. In this context, "data" means both user and system data (mess with system data and you've compromised the machine).

We also need to protect our resources: bandwidth, CPU, and data storage capacity. Many of the script kiddies out there don't want our data (or yours), they just want our disk space and bandwidth.

Finally, we want to protect our reputation as a site that does security well and provides high performance and reliability to its users.

THREATS

We have several threat vectors to deal with. First, there are the Evil Internet Hackers™, who want to break into our machines for whatever reason. As mentioned above, some just want to use our resources, but some are targeting us specifically because we've got a high profile,[5] such as those that hold us responsible for Kevin Mitnick's most recent incarceration (as of this writing ;-).

5. Probes of takedown.com, *http://security.sdsc.edu/incidents.*

In addition to external, anonymous threats, we have to worry about our own users, whom we really don't trust any more than the "outsiders." Sometimes our users willfully violate policy because they just don't care. Others are light on clue.

And sysadmins can do stupid, careless, or malicious things; we have to find a way to contain them, too (I think I hear the sound of cats and a herder).

POLICY

The general SDSC security policy is pretty short. It basically says that the security policy is to protect the confidentiality and integrity of our users' data and to provide reliable service.

In order to provide this protection, we must protect our file servers. Our policy requires that only machines considered "secure" according to a reference system can be on networks that can communicate with the file server. We refer to these systems as "managed" machines or "reference systems." How we build a reference system is described below.

We also realize that accounts can be compromised by the sniffing of passwords, and that an account compromise can lead to a system compromise. So we instituted a policy that says that no authentication protocols which use plaintext passwords or other secrets which can be intercepted and reused can pass between our "trusted" networks and other networks. I call this the "no plaintext passwords" policy.

Since a compromise of one system can lead to compromise of other systems, our policy also requires anyone with a system on any of our networks to report any suspected compromise to the security group for investigation.

For privileged access, the user must explain why they need the access and sign an acceptable-use agreement that is also signed by their supervisor. The root password is only given out where absolutely necessary, such as to people who need to log in to a system console during the installation process. For all other cases, limited "sudo" access is given for UNIX access, and a local administrator (not domain administrator) is provided for Windows users. The user is also given a little lecture to make clear that they are not to use their privileges for activities beyond the reason they gave for access – that access/ability is not authorization. The agreement form also indicates this.

Finally, in order to accommodate users who do not want to comply with our reference guidelines, we have a section of the network which we call "The Outback," in which anyone can install a system. The user and their supervisor must sign a form indicating that they take responsibility for their system, and that if it is compromised we may take custody of the system for forensic analysis and, potentially, as evidence in a criminal case.

We have, on occasion, had to strip a user of privileged access. We also have seized numerous Outback machines over the years (although the rate *has* decreased over the last year or so).

SDSC's Security Approach

Our security approach revolves around a few basic strategies described here.

Scalable Configuration Management

Most (all?) default vendor installations of operating systems have security vulnerabilities. Systems with default installations will eventually get compromised.[6]

6. See *http://worm.sdsc.edu/*.

Another common point of security failure is secure configurations that are lost after a system is reinstalled, upgraded, or overwritten by a vendor patch. Also, sometimes the configuration change fails to get installed on all machines – 99 were patched, but the 100th forgotten, or a machine that was down for several weeks and stored in a closet is booted onto the network, and it does not have the fixes that were put on the other running systems.

Our system configurations are based on a "reference system" and managed using automated configuration management software (cfengine).[7]

7. cfengine, *http://www.infrastructures.org/cfengine/*.

We use cfengine to correct things that the vendor gets wrong, to install locally built software, and to install/maintain configuration files (e.g., fstab). Every system runs cfengine upon boot, and each night, files or permissions that have been modified by hand on a system will be detected by cfengine and restored to their "reference" state – the system is effectively self-healing. Cfengine operates on "classes" of machines – a change put into the reference will be automatically installed on all machines in that class. And since cfengine logs the changes it makes and makes backup copies of any files it modifies, it serves as an automated intrusion detection and recovery system.

Now, those of you who paid attention when I was ranting about firewalls might point out that a misconfiguration will cause a failure across *all* systems; that is correct. But since most host-based protective measures are not interdependent (chained), we are not as susceptible to the cascade failures mentioned above. And, yes, we could open up a window of vulnerability on our machines. But we think the trade-off is worth the risk, as we don't end up with machines on the network whose configuration state is unknown. And putting the fix into the reference guarantees that it will be propagated to all machines.

Aggressive Patching

Most publicized compromises (especially worms) have taken advantage of vulnerabilities for which patches were already available, in some cases for months or years. Aggressive patching could probably solve 90% of most companies' security problems.[8]

8. Marcus Ranum's horse would probably disagree with that number.

Security patches are installed as soon as possible. We prioritize patches based on a combination of whether the vulnerability is remote or local, and whether or not an exploit exists.

Patches are tested on a single machine, then on willing victims' (users) desktops for Microsoft systems, and then are distributed to all appropriate hosts with automated tools.

NO PLAINTEXT PASSWORDS/STRONG AUTHENTICATION

The other vector for attack is via a compromised account where the password for that account had been sniffed from the network, or guessed. Additionally, many intruders will install a sniffer, regardless of their main purpose, to opportunistically find other accounts and machines to compromise.

A compromised account is one of the most difficult to detect. How can one easily determine whether a given login session is the legitimate user or an intruder?

We use a combination of solutions to provide authenticated services where passwords are either encrypted or not transmitted (e.g., SSH, Kerberos).[9]

9. See Abe Singer, "No Plaintext Passwords," *;login:*, November 2001, vol. 26, no. 7, for details of how we eliminated plaintext passwords. *http://www.usenix.org/publications/login/2001-11/*.

Practices

REFERENCE SYSTEMS

The general process of creating a reference system is to first install an appropriate selection of system software (usually, the vendor's procedures will be used for this), then add SDSC-specific software (e.g., cfengine), then modify everything to fix security problems and establish the functionality we require. Key to making this work is a high degree of automation, which allows the easy replication of the system.[10]

10. Jeff Makey wrote this in an unpublished document describing our reference system.

Our reference system includes the following:

- Automated configuration management using cfengine (see description, below).
- Time synchronized to atomic clocks using Network Time Protocol. This is essential when mounting a central NFS file system. Synchronized time is also useful for forensic purposes in analyzing file timestamps and correlating syslog entries from multiple hosts.
- Centralized account management. NFS requires that UIDs be consistent across clients in order to prevent inadvertent access to protected files. Centralized account management keeps UIDs consistent across all our machines.
- A password-changing program that rejects easy-to-guess passwords. Our password changing uses cracklib[11] to test passwords that could be guessed with crack, instead of trying to crack them after the fact. Why look for crackable passwords when you can prevent people from using them in the first place?
- Detailed logging to a central host. All syslog facilities are forwarded to a central loghost and archived on our storage system (yes, we have eight years of logs stored!). Centralized logging preserves log data in the event that a system is compromised and local copies of the logs are modified by the intruder. Centralized logging also allows us to monitor logs for interesting activity across all hosts.
- Most services (including RPC) protected by TCP Wrappers. Some services which should only be used within our networks are limited to just those networks. TCP Wrappers also provides consistent logging of accepted and refused connections for services, even those accessible from anywhere.
- SSH.[12]
- Kerberos 5 authentication for Telnet, FTP, rlogin, & SSH.[13]
- Email notices sent to administrators at shutdown and boot time. Getting boot and shutdown notices helps alert us to potential problems with a host. It also reminds us to check hosts that have been down for a period of time and ensure that they are fully patched.
- Sudo.[14] Most users who need privileged access are given it via sudo. Very few people actually have the root passwords. Sudo assists with accountability by logging

11. Cracklib, *http://www.crypticide.org/users/alecm*.

12. See Singer, "No Plaintext Passwords."

13. See Singer, "No Plaintext Passwords."

14. Sudo, *http://www.courtesan.com/sudo*.

15. Sudo can be subverted. We handle this through policy.

the commands performed as root, plus it allows us to limit what the user can do, where appropriate.[15] Additionally, keeping the list of people with the root password to a minimum keeps us from having to change the root password quite so frequently.

- Identd with encrypted responses. This tool allows a remote site to collect ident information, but that information is encrypted and does not provide useful information to an attacker. However, if the remote site has a problem with connections from *our* site, they can send us the encrypted ident string which *we* can decrypt and use to track down what is happening.
- A modified "xhost" program with "+" functions disabled. The "+" function authorizes *any* connection from a remote host, regardless of who owns the process at the other end.
- A version of "su" that uses group 0 as an access list. Some vendor versions of "su" only allow users who are in group 0 (called "system," "root," or "wheel" on some OSes) to su to root. Those that don't have the vendor version replace with one that does. This helps prevent an unauthorized user from becoming root even if they *have* the root password.

Building a Reference System

We have a reference for each operating system we manage. We start by installing the OS, using the vendor's installer. We then remove software that we don't need and that (1) starts daemons, (2) has setuid/setgid programs, (3) has files with ownership or permissions we don't like, or (4) is duplicated by our own versions. We reboot the system to make sure everything comes up as we expect, and lather, rinse, repeat, until the system is configured the way we like.

We then build an auto-installer using the vendor's software (e.g., kickstart for RedHat) for the system as configured. The auto-installer also installs all necessary patches and runs cfengine once the base installation is done. Cfengine is put in a startup script to run at boot time and as a cron job to run once a day.

Centralized Data Storage

Key to keeping secure reference systems is maintaining the integrity of the reference. Additionally, maintaining the integrity of user data is necessary.

We use a central NFS file server. Since the NFS protocol has some weaknesses (file handles can be sniffed/guessed), we only let the NFS server talk to hosts that are considered "secure" – reference systems. File systems are exported to the client's IP address in order to avoid DNS spoofing.

Furthermore, the NFS server does not have a default route – it only has routes to the "trusted" networks which only have hosts that we manage. An attacker is not able to establish a two-way connection to our NFS server.

The reference partition of the file server is export read-only to all hosts, so that a compromise of a host cannot be used to compromise the reference. Administrators must be able to log in to the file server in order to make changes to the reference.

We export file systems read-only where possible, but other file systems on the file server, such as user home directories, are exported read-write. The root user on all clients is mapped to "nobody" on the file server, since root on one machine could be used to compromise files exported to another machine.

Windows Systems

I'm not going to say a lot about Windows here, only a little bit about how we manage them . . .

We use Ghost and SMS to install and manage our Windows system: Ghost to install systems, SMS to install patches on existing systems.

I don't think firewalls are completely useless.

A Place for Firewalls

Okay, as I said above, I don't think firewalls are completely useless. It's really about proportion of effort. I think most organizations spend more than 90% of their effort (money *and* time) on firewalls, and it should probably be less than 5%. Firewalls can provide an extra layer of protection (provided you know what you are protecting *against*). And some people say that firewalls are for machines that cannot protect themselves, such as printers and maybe Windows machines.

We *do* perform some packet filtering on our network. We have anti-spoof filters (on ingress) to prevent someone on the outside from sending packets that appear to come from the inside. We keep Windows machines on their own subnet and only allow certain Microsoft protocols within that subnet.

We are also playing with firewalls for some applications. One is for users who bring in their laptops. We currently put them on an open, external network like The Outback (see above). We are experimenting with a firewall that allows outbound connections but no inbound, to provide some measure of protection for the user machines. The firewall may reduce their exposure, and it provides us with a choke point to monitor and block misbehaving machines from attacking the rest of our network.

We may also use a firewall for the Windows network, as new attacks pop up so frequently, and some of them are difficult or impossible to control from the host.

Conclusion

Firewalls don't necessarily provide as much security as popularly believed. Securing individual hosts can provide better security and functionality than using a firewall. Hosts are protected from each other in addition to the Internet. Use of scalable configuration management, no plaintext passwords, and aggressive patching can provide host-based security in a scalable, cost-effective manner. It has worked for us; maybe it can work for you.