Global Information Assurance Certification Paper

Case Study: Secure Infrastructure for Partner Data Sharing

David B. Smith
December 27, 2003
GSEC Version 1.4b Option 2

Table of Contents

Introduction

This case study follows the design and implementation of a secure infrastructure for data sharing between partners. Data is considered sensitive and confidential. Partners both send and retrieve data via batch FTP and real-time applications.

This case study focuses on the infrastructure need to support this type of data transfer, not the actual securing of the data itself (encryption, data validation, etc) or the regulatory issues in securing certain types of information. The need for this project was caused by a growth in data sharing between my company and multiple partners.

Note: Due to the sensitive nature of this information and requirements from my company's CISO, I have excluded complete policies and some configuration files for services.

Before

Data sharing between partners has greatly increased over the past few years. Previously, private data networks were created, or third party clearinghouses were used to share data. With the widespread use of the Internet it is now possible for many institutions to cheaply and effectively share data between companies for purposes such as marketing, outsourcing and transaction processing.

The original infrastructure consisted of a proxy-based firewall that connected to a SGI Origin 200 running IRIX 6.5 with the bundled FTP server. Partners had a dedicated account and were prevented from seeing each other's files by using Unix permissions. Standard system files were available for download by the partners. The proxy-based firewall was running Gauntlet on a NT 4.0 server and was kept current on available service packs and security patches. There were also some Netscreen 10s acting as VPN/firewall devices for individual partners.

Encrypted data files were sent and received via the FTP server. Once delivered, an application server would pick up the file, decrypt it and load the data into a database. See Figure 1 for a diagram of the infrastructure.

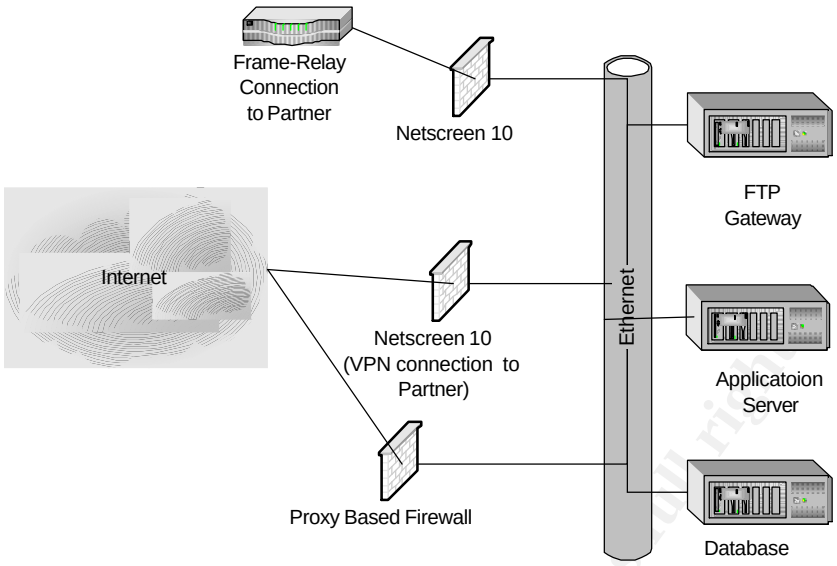

Figure 1. Before

Security Risks

- Although the data is stored securely there is no DMZ for inbound FTP access. The database and application servers were on the same network and were open to significant risk if the FTP server was compromised.
- An outdated firewall product was being used to proxy the connection. Furthermore, firewall logs were only kept on the firewall itself allowing for easy deletion and/or modification of the logs in case of a compromise.
- The proxy server did not limit connections by IP address, although two authentications were required (both the proxy and ftp server)
- The FTP server allowed all users to download unsecured system files. Although data was securely stored, many system files could be accessed.
- Planned partner growth was substantial for the next year and the management aspects of having multiple firewall/vpn devices created a significant risk for bad policy creation.
- The FTP daemon was a stock installation. Although security patches were applied regularly, the configuration was very inflexible.

This implementation evolved of some initial data transfer requirements and the need to rapidly deploy a solution when the company was first starting up.

During

Introduction

Obviously this implementation had significant security risk as well as future traffic policy management concerns. When assigned the task of fixing this problem, I had the following requirements:

- The ability to send and receive files securely without the user having knowledge of other users or access to other system files.
- The ability to access our FTP server via VPN, private circuit or the Internet.
- The ability to support multiple vendor VPN solutions, including intra-vendor solutions.
- The ability to access information from partners via middle-ware applications or direct SQL access via stored-procedures.

Network Design and Vendor Selection

To address the risks with the implementation I designed a centralized DMZ based network with a firewall controlling access between all zones. When doing a vendor investigation, my criteria was:

- Flexible policy management, including complicated NAT handling.
- Strong VPN support, especially inter-operability.
- Fairly inexpensive per port firewall cost. The company has a number of private 56K frame-relay connections to individual partners.

I investigated solutions from Cisco, Checkpoint and Netscreen. At the time Netscreen and Cisco had either a small port density or a high per port cost, which effectively eliminated them from competition. Since there was a non-standard requirement of a large number of low bandwidth connections and existing Sun hardware, Checkpoint FW-1 was chosen based on cost and performance. Although at the time of implementation Checkpoint-NG was available, it was considered too new for implementation based on business risk. FW-1 version 4.1 was considered a stable, well-deployed and tested firewall solution.

At the time of initial implementation, VPN interoperability was still somewhat challenging, so I decided to dedicate an individual VPN device for each partner. That allowed us the flexibility to have open VPN tunnels and then restrict traffic at the firewall. This design also allowed us to control outages and upgrades on a per partner basis and easily roll back Netscreen software in the event of a new VPN interoperability problem (which has happened). Low-end Netscreen appliances were used for that purpose. This also kept the design fairly standard, even though the technology for connectivity was direct, via a private connection with a router, or via VPN. FW-1 was only used for traffic control.

Figure two (below) shows the proposed network design. Each vendor connection and the FTP gateway has a private /28 network dedicated to them on a separate network port. In this configuration vendors coming in over a VPN or private connection could choose to use private space, our non-routable public-space, or to dedicate one of their network blocks for this network. By having this design we were able to provide a flexible solution to all of our partners. In many

cases this flexibility allowed us to rapidly deploy new connections because the partner only need to make minimal changes inside their network. All traffic to and from our network is NATed to an address on the private network. The firewall also has a network connection to the Internet for FTP connections. Additionally, during the implementation, a new requirement came up to allow for a partner co-located database, it was easily incorporated into the design.

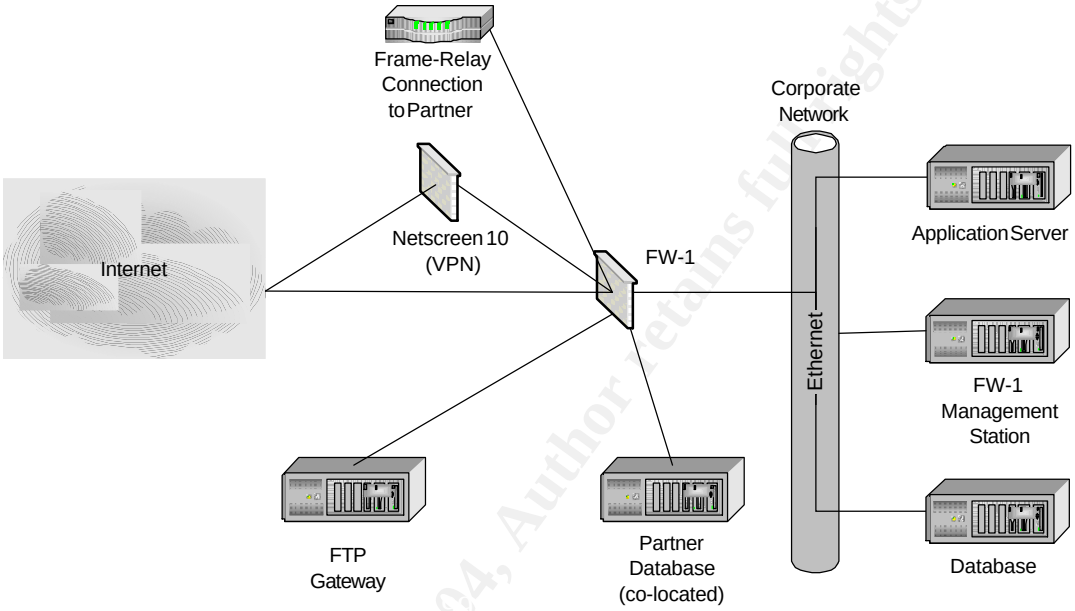

Figure 2.
Proposed

Checkpoint FW-1 Installation

FW-1 was deployed on two Sun Ultra 2 Enterprise Servers running Solaris 7. Using Lance Spitzner's Beginner's guide to Armoring Solaris, I created two secure, patched servers running Solaris 7[1]. I removed packages based on Spitzner's Solaris 7 documentation[2]. Once completed, I shutdown all network services by commenting out all of /etc/inetd.conf and removing many of the startup files. SSH was installed for command line access. Once finished the only processes running were init, sched, fsflush, pageout, ttymon, cron, sshd and syslogd.

Additionally the following software was installed:

- Tcpdump for troubleshooting purposes.
- Sun Microsystems nddconfig per Sun's Operating System Network Settings for Security[3].

- Sun Microsystems SE performance toolkit for performance monitoring.

See Appendix 1 for more information regarding server configuration.

The Checkpoint FW-1 software was installed via the software CDROM. One system became the enforcement point and the other a management station for logs and policy control. Both were licensed and the GUI clients were configured via the cpconfig utility on the management station. User accounts were also added for each firewall administrator. The latest service packs were downloaded from Checkpoint FW-1's support site and applied.

FTP Server installation

The operating system for the FTP server was setup the exact same way as the firewall servers, except for the installation of the Checkpoint FW-1 Software. ProFTPD was chosen as our ftp daemon because of its security posture, configuration flexibility and logging capabilities.

ProFTPD was compiled on a separate server and installed on the system. To eliminate access between partners, the <Anonymous> configuration directive was used. The anonymous configuration directive causes ProFTPD to create a chroot environment, which prevents the user from seeing any other files on the system. Since ProFTPD was chosen over other FTP servers, system files did not have to be copied into the directory for anonymous access to work. This is because ProFTPD opens all required files before performing the chroot. By setting the User and AnonRequirePassword fields, a valid Unix user account is required. Otherwise, the user would only need to enter an email address to gain access to environment.[4] Also, users were limited from reading or deleting files as well as creating or deleting new directories. This minimizes risk in case of a password compromise, especially over the Internet where passwords are being passed in plain text. Below is an example configuration for a user. For more information on installing and configuring ProFTPd please see http://proftpd.linux.co.uk/localsite/Userguide/linked/userguide.html

```
<Anonymous /data/ftp/testuser>
    User                                testuser
    Group                               ftp
    AnonRequirePassword                 on
    MaxClients                          1
    RequireValidShell                   off

    <Directory /*>
        AllowOverwrite on
        <Limit STOR CWD>
            AllowAll
        </Limit>
        <Limit READ RMD DELE MKD>
            DenyAll
        </Limit>
```

```
    </Directory>
</Anonymous>
```

Checkpoint FW-1 Policy Build Out

Policy

After all the software was installed and initialed configured, I started to create the firewall's policy. To test and install the policy, the firewall and FTP server were connected to the internal network and DMZ respectively; the firewall was not connected to the Internet until the servers were ready for production.

The following access was required (scrubbed policy names in parenthesis):

- Administrative access via SSH to the firewall from a select group of systems (Firewall_Admins)
- Direct Oracle SQL access to a partner's database co-located in our facilities (F_Server) from our internal network (I_Networks)
- HTTP/HTTPS access to a group of servers (S) connected via a Netscreen VPN from our internal network (I_Networks)
- Inbound FTP access from server group S to a private NAT for our FTP server (S_NAT)
- Internet access from various FTP clients (FTP_Clients) to the external NAT for our FTP server (ftp)
- SSH/FTP access from our internal networks to the FTP server

In addition to these policies, FW-1 comes configured with some default accept rules that are less than ideal for security purposes. It is always a good policy to disable these defaults. However, it is important to note that when changing defaults it applies to all enforcement points[5]. Defaults can be configured via the Properties dialog off the Policy menu. Also by default, FW-1's policy is to accept traffic; rule number 2 prevents access to the firewall and rule number 8 (always the last rule) drops all unmatched traffic. All firewall policies should have these rules. A screen capture of the policy is below (Figure 3).

No.	Source	Destination	Service	Action	Track
1	Firewall_Admins	inf-chk- 01	SSH	accept	Long
2	Any	inf-chk- -01	Any	drop	Long
3	I_Networks	F Server	sqlnet1	accept	Long
4	I_Networks	S	http https	accept	Long
5	S	S _NAT	ftp	accept	Long
6	FTP_Clients	ftp	ftp	accept	Long
7	I_Networks	inf-misc 04	SSH ftp	accept	Long
8	Any	Any	Any	drop	Long

Figure 3. Firewall Policy

NAT Rules

At its most basic definition, Network Address Translation (NAT) is the translation of an IP addresses on one network into different IP addresses on another network.[6] There are more complicated implementations that allow various levels of control when performing NAT on a packet.

FW-1 Implements the follow types of NAT[7]:

- *Source Static translates the source IP address in an IP packet to a specific IP address. This is a 1 to 1 address translation for connections that originate from "inside" the firewall. Return traffic, as necessary, is allowed back.*
- *Hide is a many-to-1 translation. To do this, the source port of the packet is always changed to something else. Based on this source port (which "replies" will be sent back to), the firewall will know where to direct the return traffic. Most standard applications (e.g. telnet, http, ftp, https) work fine, but any application that requires a connection initiated from the outside will not work with the hide translation, however, anything using an IP datagram other than TCP or UDP may not work correct (ICMP is handled properly, though).*
- *Destination Static translates the destination IP address in an IP packet to a specified IP address. This is a 1 to 1 address translation for connections that originate from "outside" the firewall. Return traffic, as necessary, is allowed back.*

FW-1 handles all NAT rules last. Policy checks and OS routing occur first. Therefore for certain types of NATs to work correctly a static proxy ARP entry is required. You could also add a static route to the upstream router point to the firewall. However with this particular implementation, access to upstream routers

wasn't always available[8]. Static ARP entries were always used to minimize configuration complexity,. In addition to creating the NAT rules, a host-based route on the firewall itself was required to correctly route the translated packet on the firewall. To facilitate management of these entries, a script was created and placed in /etc/init.d and hard linked into /etc/rc2.d.

The NAT rules were as follows (see Figure 4 for a screen capture):

- NAT rules 1 and 2 are hide rules for SQL server and HTTP/HTTPS outbound (corresponds to policy rules 3 and 4)
- NAT rules 3-5 dealt with FTP access. Internally, the FTP server (inf-misc-04) needed to be exposed as its internal address (rule 3). For all other Internet traffic from FTP_Clients a one-to-one static NAT entry was created.
- NAT rules 6 and 7 are for FTP access via a VPN connection with a partner. The partner chose a specific RFC-1918 compliant address that easily integrated into their network architecture.

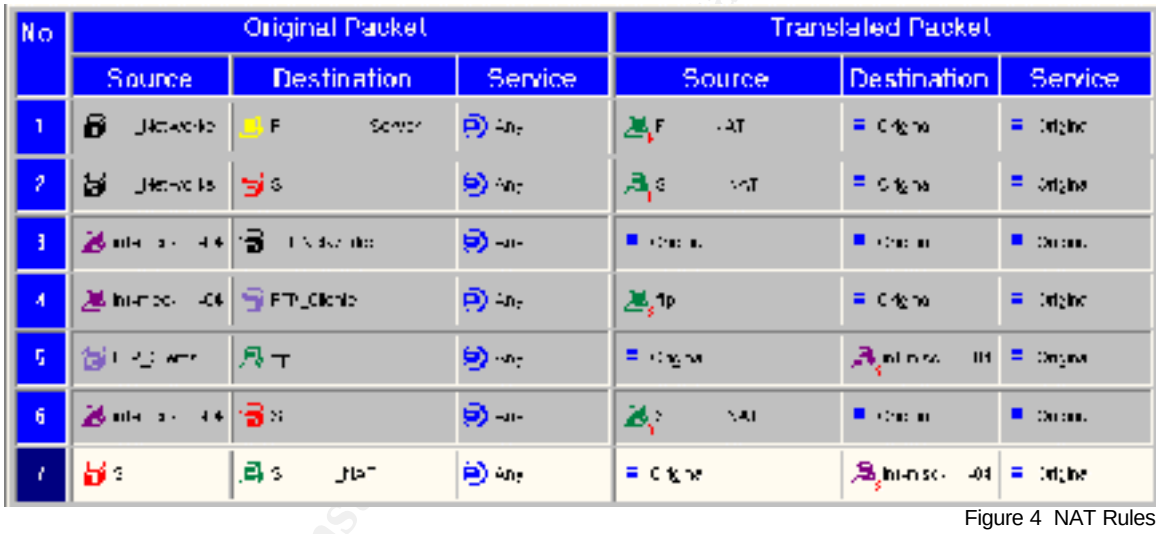

Figure 4 NAT Rules

Spoof Detection

Spoof Detection with FW-1 is somewhat challenging to implement, especially when combined with NAT. In the interfaces tab of the enforcement point object, you can set specific networks and what traffic is allowed.

FW-1 allows the following[9]:

- *Any (the default) - All addresses are considered valid on this interface. Note that IP Options checking is still performed in this mode (which is how a lot of packets are "spoofed" from the Internet).*
- *No Security Policy - Do not enforce **any** security policy on this interface. Not only does this include anti-spoofing, but this includes your policy as well. Use with extreme caution!*

- *This Net - Probably the most mis-understood of the options. What this specifically means is "the logical network this interface is on." Contrary to popular belief, there is no magic to this as it is defined by the interfaces IP address and netmask per the configuration screen. All other networks are not considered valid for that interface.*
- *Specific - A group of network objects (networks, hosts) that defined the "valid addresses" for this interface. Typically used where there are multiple networks reachable from this interface and/or when Network Address Translation is used. If a host reachable from this interface has a "translated" IP address, you will need to include the "translated" IP address in this interface's "valid addresses" setting.*
- *Others - This is used on your interface facing your Internet connection. Specifically, it means "all IP address not specified on other FireWall interfaces as valid."*
- *Others + - This allows you to specify IP addresses that appear on both your internal and external interfaces. This is usually needed when you are doing NAT in certain situations, running OSPF on both the internal and external interfaces, or running VRRP.*

When doing one-to-one NATs with IP spoof detection enabled, you will see rejects into your DMZ based on a spoof alert. This is because the NAT happens last with FW-1 (with Checkpoint-NG it happens first, which alleviates this problem). When the packet enters the DMZ network and goes through spoof detection, it still has its NAT address. To work around this problem, a network group was created with all of the FTP server's NAT addresses and the security policy was set to Others + FTP_NATS (the group).

Testing

Once the policy and rules were built out, connectivity and security tests were performed. A laptop was connected to the external interface of the firewall and network scan was performed using NMAP. Although the test results were satisfactory, the management station's log buffer ran out of space. This caused the management station to stop logging packets. This seems like a critical design error on Checkpoint's part and from searching on the Internet it still looks like its apparent in NG. To work around this problem, two kernel variables need to be set in /etc/system for the FW-1 module[10]:

> *To resolve the issue, make a backup copy of the /etc/system file, then open the /etc/system file, and add the following lines (or increase existing parameter values, represented by hex numbers):*
>
> *1. set fw: fw_msg_q_max = 0x10000*
>
> *By default, this parameter is set to 0x200. There is no danger in increasing fw_msg_q_max as much as needed, if 0x10000 did not resolve*

the problem.

2. set fw: fw_log_bufsize = 0x80000

By default, this parameter is set to 0x14000. The fw_log_bufsize can be increase d to 512K (0x80000).

3. Reboot the machine

Once the firewall was up and running, we also ran into some problems with Oracle connection pooling when connecting to the partner owned, co-located database. After some investigation, we figured out that the Oracle client would open up multiple connections when necessary and leave them open after they became idle. After 60 minutes, FW-1 would idle them out of the state table. Then, when the Oracle client needed to use them again, it would run into problems and crash. The solution we implemented to this problem was two-fold. First, I upped the timeout limits for port 1521 (sqlnet1) to six hours. To do this I added `ADD_TCP_TIMEOUT(1521,3600*6)`, to lib/init.def in the FW-1 directory on the management server[11]. Secondly, a developer added checks to the code to test and reopen connections if necessary. Although this is still required, its greatly reduced by the increased timeout value.

After

The goal of this project was to remove some substantial risks identified with the beginning implementation, which grew more out of business needs than out of a controlled project. Revisiting the risks that were present:

Risk: Although the data is stored securely there is no DMZ for inbound FTP access. The database and application servers were on the same network and were open to significant risk if the FTP server was compromised.

Mitigation: A DMZ was created and a new FTP server was placed in the DMZ. Spoof detection was also implemented to prevent unauthorized network access.

Risk: An outdated firewall product was being used to proxy the connection. Furthermore, firewall logs were only kept on the firewall itself allowing for easy deletion and/or modification of the logs in case of a compromise.

Mitigation: The firewall was upgraded to a significantly deployed, well-tested stateful inspection firewall. Firewall logs were migrated to a separate server.

Risk: The proxy server did not limit connections by IP address, although two authentications were required (both the proxy and ftp server)

Mitigation: Although installing FW-1 removed the double authentication, the new implementation only allowed connections from a select group of known hosts.

Risk: The FTP server allowed all users to download system files. Although data was securely stored, system files could be accessed. The FTP daemon was a stock installation. Although security patches were applied regularly, the configuration was very inflexible.

Mitigation: A new, securely installed Solaris box was implemented with ProFTPD. ProFTPD allowed for secure, auditable access and users were only able to see files they had been granted specific access to.

Risk: Planned partner growth was substantial for the next year and the management aspects of having multiple firewall/vpn devices created a significant risk for bad policy creation.

Mitigation: Although VPN devices were still being used, all security policy and control was rolled up to one centrally managed firewall.

Additionally, after the successful roll out of this project further enhancements were applied including:

- Firewall based IDS (http://www.spitzner.net/intrusion.html)
- DMZ traffic generation alert – the DMZ should never initiate certain types of traffic, if it does alert and lock down. (http://www.spitzner.net/rules/rule11.html)

Upgrading our software and operating systems, as well as creating a DMZ for all access points into the network significantly reduced my company's risk. As a policy all confidential data on the FTP server is encrypted so a compromise would not constitute a significant loss for either company.

Since the deployment of this solution, a significant number of partners have been added and there haven't been any major change to the design. Although not 100% bullet proof, this solution balanced business risks with needs and allowed us to significantly enhance our security posture. Ideally, a IDS implementation watching both the DMZ and Internal network would add detection in case of attacks. Furthermore, Tripwire or another similar product on both the management station and enforcement point would allow for modified file detection and alerting.

Currently, a project is underway to upgrade the firewall and add separate IDS capabilities to the design.

Appendix 1 – Detailed Server Information

Final Package Listing

```
application ANCrules       Adrian's Rules & Tools
application CPdtm-41       Check Point Policy Server
application CPfw1-41       Check Point VPN-1/FireWall-1
application RICHPse        The SymbEL Interpreter
application RICHPsex       The SE eXtensions Package
application SMCrsync       rsync
application SMCtcpd        tcpdump
application SMCtop         top
system      SUNWadmc       System administration core libraries
system      SUNWadmfw      System & Network Administration Framework
system      SUNWcar        Core Architecture, (Root)
system      SUNWcsd        Core Solaris Devices
system      SUNWcsl        Core Solaris, (Shared Libs)
system      SUNWcsr        Core Solaris, (Root)
system      SUNWcsu        Core Solaris, (Usr)
system      SUNWdoc        Documentation Tools
system      SUNWesu        Extended System Utilities
system      SUNWhmd        SunSwift SBus Adapter Drivers
system      SUNWkvm        Core Architecture, (Kvm)
system      SUNWlibC       Sun Workshop Compilers Bundled libC
system      SUNWlibms      Sun WorkShop Bundled shared libm
system      SUNWloc        System Localization
system      SUNWman        On-Line Manual Pages
system      SUNWos86u      Platform Support, OS Functionality (Usr)
system      SUNWploc       Partial Locales
system      SUNWploc1      Supplementary Partial Locales
system      SUNWqfed       Sun Quad FastEthernet Adapter 32bit Driver
system      SUNWqfedu      Sun Quad FastEthernet Adapter Driver Headers
system      SUNWscpu       Source Compatibility, (Usr)
system      SUNWswmt       Install and Patch Utilities
system      SUNWter        Terminal Information
```

Disk Layout

```
Filesystem             kbytes    used   avail capacity  Mounted on
/proc                       0       0       0     0%    /proc
/dev/dsk/c0t0d0s0      493688   23042  421278     6%    /
/dev/dsk/c0t0d0s5     1018382  101658  855622    11%    /usr
fd                          0       0       0     0%    /dev/fd
/dev/dsk/c0t0d0s3      493688   88919  355401    21%    /var
/dev/dsk/c0t0d0s6     4509901  110256 4354546     3%    /log
/dev/dsk/c0t0d0s4     1018382  297403  659877    32%    /opt
/dev/dsk/c0t1d0s2     8705501   83046 8535400     1%    /arch
swap                  1422624     376 1422248     1%    /tmp
```

/etc/issue:

```
Authorized Users Only
```

```
The information on this computer and network is the property of Company
X and is protected by intellectual property rights. You must be
assigned an account on this computer to access information, and are
only allowed to access information as defined by the system
administrators. Your activities may be monitored.

Attempts to upload or change information on this system without
authorization, or to download and copy with the intent to defraud are
strictly prohibited and may be punishable under the Computer Fraud and
Abuse Act of 1986.
```

/etc/default/inetinit

```
# @(#)inetinit.dfl 1.2 97/05/08
#
# TCP_STRONG_ISS sets the TCP initial sequence number generation
parameters.
# Set TCP_STRONG_ISS to be:
#       0 = Old-fashioned sequential initial sequence number
generation.
#       1 = Improved sequential generation, with random variance in
increment.
#       2 = RFC 1948 sequence number generation, unique-per-connection-
ID.
#
TCP_STRONG_ISS=2
```

/etc/rc[23].d file listings

```
>  ls /etc/rc2.d
S00fw1bootd       S69-cppreinet     S75cron           S99orcallator
S01MOUNTFSYS      S69inet           S75savecore       S99route
S01cleanstartup   S69zcppostinet    S90checkpointnat  S99sshd
S05RMTMPFILES     S70nddconfig      S95firewall1
S20sysetup        S74syslog         S99elaproxy
> ls /etc/rc3.d/
>
```

(rc3.d is empty)

References

[1] Spitzner, Lance. "Armoring Solaris." 19 August 2001. URL: http://www.spitzner.net/armoring.html (27 Dec. 2003).

[2] Spitzner, Lance. "Core7.txt." 19 August 2001. URL: http://www.spitzner.net/core7.txt (27 Dec. 2003).

[3] Watson, Keith, Noordergraaf, Alex. "Solaris™ Operating Environment Network Settings for Security." December 2000. URL: http://www.sun.com/solutions/blueprints/1200/network-updt1.pdf (27 Dec. 2003).

[4] "ProFTPD configuration manual – Anonymous." URL: http://www.proftpd.org/docs/directives/linked/config_ref_Anonymous.html (27 Dec. 2003).

[5] Spitzner, Lance. "Default Properties." 26 Jan 2000. URL: http://www.spitzner.net/rules/def.html (31 Dec. 2003).

[6] "Network Address Translation." URL: http://searchwebservices.techtarget.com/sDefinition/0,,sid26_gci214107,00.html (27 Dec. 2003).

[7] Welch-Abernathy, Dameon D. "How NAT Works." 12 Nov. 2002. URL: http://oldfaq.phoneboy.com/fom-serve/cache/77.html (30 Dec. 2003)

[8] Welch-Abernathy, Dameon D. "How NAT Works." 12 Nov. 2002. URL: http://oldfaq.phoneboy.com/fom-serve/cache/77.html (30 Dec. 2003)

[9] Welch-Abernathy, Dameon D. "How Do I Define Anti-Spoofing." 20 Nov. 2002. URL: http://oldfaq.phoneboy.com/fom-serve/cache/192.html (30 Dec. 2003)

[10] Stich, Reinhard. "Re: [FW-1] Enlarging NG log message queue." 21 Nov. 2003. URL: http://www.mail-archive.com/fw-1-mailinglist@amadeus.us.checkpoint.com/msg03876.html (1 Jan. 2004)

[11] Welch-Abernathy, Dameon D. "TCP Timeout for a specific service." 19 Nov. 2002. URL: http://oldfaq.phoneboy.com/fom-serve/cache/462.html (1 Jan. 2004)