Global Information Assurance Certification Paper

PRACTICAL ASSIGNMENT FOR THE

GIAC CERTIFIED UNIX SECURITY ADMINISTRATOR EXAMINATION (GCUX, Option 1, Version 1.9) "Securing Unix Step by Step"

entitled

"SECURING THE SUN FIRE V100 SERVER, FOR USE AS THE CENTRAL MANAGEMENT SERVER, IN A NETWORK FLIGHT RECORDER INTRUSION DETECTION SYSTEM"

ABSTRACT: This paper outlines one method of building an NFR Central Management Server on a carefully minimized and hardened Sun Fire V100, running Solaris 8. Starting with an out of the box Sun Fire V100, we step through the processes of operating system installation, patching, hardening, application installation, configuration verification, and scripting for on-going maintenance. The expected result of the process is to produce a trim, robust, and somewhat self-monitoring platform, one that is resistant to tampering, effective in performing its mission, and readily reproducible. The machine is expected to present 3 open ports on the network, and to communicate via encrypted channels. It is expected to reveal little about itself, its operations, and intended functions, while resisting attempts to subvert its configuration. It should log events meticulously, and also report these events to a remote syslog server.

By: George Markham, GCIH

(August 12, 2003)

System Description

We start this project with a stock Sun Fire V100 Server in a box, whose ultimate destiny

is to become a secure component in an intrusion detection system. The V100 is an entry-level UltraSPARC-IIe (500mhz) computer platform, designed as a compact, low-cost/high quality enterprise grade server that is:

> "ideal for … anyone who wants to maximize the density of Solaris servers in a rack"[i]. (Sun, P20)

The subject system is equipped with two "40 gig" IDE hard drives, two 10/100mbs network adapters, and 1024M RAM. The server ships in a 1U rack mount chassis. It contains no keyboard, mouse, or display ports – all communication is via two serial ports, two USB ports, and/or two network adapters. System configuration information resides in a "smart card" that plugs in the back, much like a satellite television receiver card This smart card can be considered portable NVRAM[1], and can be used to move host configuration data (hostid[2], MAC address[3], configuration settings, etc.) from one chassis to another[ii]. This delicate operation can be accomplished fairly quickly, as follows:

> " To transfer the System Configuration Card to a new server:
> Power down both the old and the new Netra servers.
> Remove the front bezel from both the old and the new servers.
> Remove the System Configuration Card from the old server and insert it into the new one.
> Replace the front bezel on the new server. You can secure the new System Configuration Card by fixing a tie-wrap through the hole inthe front mounting of the memory card reader. " (Sun Microsystems, supra)

The machine's ultimate role is to become a Central Management Server (CMS) in a Network Flight Recorder Intrusion Detection System (IDS). The CMS is designed so that "Multiple NID Sensors can be managed from a single CMS, and multiple CMS's can

---

[1] Sun's SPARC based systems historically store configuration information in NVRAM - Non Volatile Random Access Memory – a form of static random access memory, backed up by a battery or Electronically Programmable Read Only Memory. Sun Fire servers use the "smart card" for this function.

[2]SUN SPARC based computers each contains a host identification number (hostid) coded into a PROM chip, which is sometimes tied to software license schemes etc. The V100 also uses the smart card to store this data.

[3]MAC address – Media Access Control address – a unique, manufacturer-assigned 48-bit hardware address for network devices. SPARC systems store this in NVRAM, and the kernel reports this value for every NIC in the machine. To force reporting of MAC addresses on a discrete, per-interface basis, you separately configure each interface using the "ether" parameter of the ifconfig command. The *local-mac-address* property in NVRAM also configures this behavior, via the Solaris eeprom command, or the setenv OpenBoot (ok) prompt.

be deployed within and environment." [iii](NFR, P2). The CMS configures and controls this NIDS via encrypted TCP connections. It also concentrates alerts and other traffic information, making it available to management stations via an encrypted TCP connection to one (or more) windows-based Administrative Interface (AI) clients. The CMS can also be configured to export data to an Oracle database, via a product aptly called DBExport. The subject site has chosen not to utilize this option, due to the additional cost of supporting an Oracle installation. NFR support staff have intimated that DBExport will soon be extended to support a wildly popular Open Source database. For basic installation information, hardware and software requirements, etc., refer to The NFR Central Management Server User's Guide, P1-4[iv]. The CMS software to be installed on the subject system is version 3.0.2. This system will be deployed on a medium sized campus network, where it will reside behind a traditional firewall. It will receive its traffic information and alerts from remote sensors that monitor traffic off of mirrored router ports, situated at key points of interest. These remote sensors consist of customer-supplied "PC" hardware, and run an NFR supplied operating system that boots off read-only media, not unlike the popular Knoppix or Trinux Linux distributions. For a graphic schematic, and synopsis of the relationships between the components of this system, their interoperation, etc. refer to the publication NFR Security Solutions, Network Intrusion Detection System, Technical Data[v].

The sensor software version for this installation is NID-300 Series v3.0 (customer sourced hardware). The sensors in this NFR installation are configured to sniff traffic data and to communicate with the CMS on one interface. A second interface is used for system administration and backup transfers only. Command and control is performed via an encrypted TCP connection. This writer is unaware of any attacks that have ever compromised one of these sensors, although they are certainly vulnerable to denial of service attacks, arp spoofing diversions, etc. One can configure the sniffer interface as a dumb interface (without IP assignment etc.), however this site chooses to monitor the management interface for reasons best described as historical. At the time of this writing, NFR does not officially support Solaris 9, so Solaris 8 will be used for this project. No mention of 32/64 bit compatibility issues was noted in the NFR literature reviewed prior to installation, however, one library necessary for NFR CMS system operation was reported missing after installation, and had to be manually loaded into /usr/lib. This is thought to be due to the radical hardening applied to the server, and/or the minimalist installation approach. In any event, it is not a reflection on NFR documentation, but rather an illustration of how complex operating system hardening can be.

The V100 hardware platform offers some firmware security features, which will be enabled during our exercise. Although the system arrived pre-configured with Solaris 8 installed, a fresh, *minimalist* installation will be performed from sealed factory media, followed by application of Sun's Recommended and Security patch cluster. Then will come hardening with the JASS Security Toolkit, followed by installation of Titan. We will add some YASSP packages and miscellaneous support tools. We will then install our

primary application, do some configuration fingerprinting, maintenance, and backup scripting. Finally, we will perform some validation testing of the machine's over-all security posture.

Software Synopsis

To recap, the subject system's final software inventory will include:

A "core" Solaris 8 OE™ (64-bit support) installation, patched with current Recommended and Security Patch Cluster (with a 32bit /usr/lib/libCrun.so.1 inserted into /usr/lib for NFR compatibility[4]).

gzip, version 1.2.4, by GNU Software Project (YASSP/Solaris package version)

md5, by "RSA Data Security, Inc. MD5 Message-Digest Algorithm".

JASS Security Toolkit, version 4.0.0, by Sun Microsystems, Inc.

Titan, Version 4 beta6, by Brad M. Powell, Dan Farmer, and Matthew Archibaldand, et al.

NFR Central Management Server, Version 3.0.2, by Network Flight Recorder, Inc.

ASR Tripwire, version 1.2, by Tripwire, Inc. (YASSP/Solaris package version)

SSH Version OpenSSH_2.3.0p1, protocol versions 1.5/2.0(Compiled with SSL), by the Openssh Project. [5] (YASSP/Solaris package version)

LSOF, Version 4.68, maintained by Vic Abell at Purdue University

Chkrootkit, Version 0.4.1, by Pangeia Informatica Software

[4] On the first practice run, NFR installed but the bin/samd daemon would not execute until a necessary 32-bit library was moved into /usr/lib from patch 108434-13. The file is found in ./8_Recommended/108434-13/SUNWlibC/reloc/usr/lib/libCrun.so.1

[5] This version ships with YASSP, and is subject to certain vulnerabilities. As a consequence, sshd will not be enabled, and no connections outbound will be initiated unless on the secure (crossover) network.

Expected Result and Final System State

At the completion of this project, we expect to have a system secure enough to place on a hostile network, suitable to act as the Central Management Server for a Network Flight Recorder system. The system will be minimized to remove unnecessary programs and services. The networking attributes will be tweaked, using Sun recommended settings, to harden it against common attacks, improve its handling of network connections, and reject undesired types of traffic. It will be configured to log syslog events remotely to a second system. It will be expected to present a total of three network ports open when we are finished: UDP/520, and TCP/1968 and 2010, and no other. It will be expected to integrity check itself for file system alterations, including "rootkitting" and to report its findings regularly via email. It will be expected to prohibit any non-root user from writing in the /usr file system, to prohibit non-root users from performing suid program execution from any partition, except the /(root) file system, and to log all such attempts. It is expected to automatically start NFR, and to be secure from tampering physically. It is expected to reside in a secure facility with controlled access and monitoring. It is expected to require password login to access the EEPROM functionality present in Sun SPARC based computing platforms, and to present appropriate warning banners for all logins.

Risk Analysis

Because this system plays a key role managing an enterprise Intrusion Detection System (IDS), it will require *significant* hardening prior to its deployment. A recent scan of the network (a large university campus) revealed over 13,000 ports, populated at any time with an average of around 9,000 active systems. This population runs an astounding variety of operating systems and applications. An inspection of several floors of one building revealed a host of specialty devices, such as embedded time clocks, blood gas analysis systems, and some advanced, portable ultrasound machines the size of a laptop. A large medical lab houses hundreds of advanced instruments, many of which are networked. There are a large number of CT, MRI, and other diagnostic and treatment tools, most of which are built around networked Sun and SGI workstations. These high-end systems represent a huge investment in patient care, and an extraordinary challenge from a support and patching perspective. Many are highly customized, and cannot be patched without revalidation by their vendors. Validation of medical computing devices in the United States typically involves exhaustive testing, documentation, and approval by entities such as the Food and Drug Administration (FDA) – nothing one does often, or lightly. A whole industry has sprung up to assist in compliance with the implementing regulations, currently found in the Code of Federal Regulations (C.F.R.), at 21CFR, Part 11. For pointers to official FDA documentation governing this huge facet of Medical IT support, see 21CFRpart11.com "Links to FDA Documentation"[vi]. These specialized diagnostic systems are often managed by clinical,

rather than IT people. The emphasis on these systems has traditionally been ease of use, not security, although this is changing. The operating systems presenting ranged from an ancient AIX, version 3.1, to early SunOS, NeXT, and IRIX. Several systems run QNX, including an FDA regulated system, which is classified as a medical device, because it assists in the dispensing of therapeutic products. There are scattered versions of Linux, some dating into the hoary past. There is an IBM mainframe, and thousands of Intel workstations running Windows 95 through XP. A sub-netting team recently uncovered a running DOS 5 machine! Powerful, wireless equipped Mac OS X systems are in use, along with Palms and related gizmos. A good number of switches and routers support the core infrastructure, providing 100MBS service to wall jacks, and back hauling traffic across redundant "gig" copper and fiber links from the edge routers to the core enterprise router housed in the data center. Select systems enjoy copper gig – mainly highly utilized database and enterprise backup servers, with a handful of "hot-dog" workstations showing up lately. There are discrete point-to -point T1 connections to outlying offices, vendors, and partner sites. Topping this collection off is an Internet-attached VPN server, and a fair collection of modems, with locations known and unknown[6]. The LAN itself is connected to a state governmental backbone, and to both the Internet (and Internet 2) via redundant fiber links, with a total combined throughput potential of 110MBS.

The subject network is attached to the Internet and Internet 2, which means we get targeted for intrusion attempts around the clock. Despite enlightened support from management, and the efforts of a dedicated team of security-aware professionals, systems will be compromised. Once hacked, these systems can become springboards to launch further mischief from the "trusted" local address space. The subject LAN, while safer than the public networks, is therefore considered hostile and untrustworthy. The site maintains huge amounts of information, classifiable as health care, financial, academic, human resources, R&D, experimental, law enforcement, and probably a few types not readily classifiable. There is potential for financial loss, as well as personal and institutional embarrassment in the event of a serious breach of security. The potential for regulatory fallout and legal liability to injured parties is significant. Patient treatment could be affected, if, say dosage data were altered in a radiotherapy planning system, or a blood type was changed, and a bad transfusion resulted, etc. Brian Osborne, of ThinkGeek.com reported, in July of this year, that legal precedent exists for governmental entities to be unplugged judicially from the Internet, for ineptitude in security matters no less! It is a very serious business keeping this network secure[vii].

[6] Business networks will have unofficial entry points lurking in the wings, and this one is no exception. Recently a bi-directional modem was found on a Sun-based printing system. The modem auto-answered when dialed, used a simple default password, and offered a ppp session. Support personnel have since stumbled onto a 56K frame relay link, installed by a vendor for support access, which, as of this writing, has not been assessed for cost or security. The moral is simple, *Watch your vendors,and watch them closely*.

In this less than perfect environment, the subject system must monitor the intrusion detection sensor(s), and present the data gathered for review by network security and support personnel. The CMS issues alerts based on various criteria, and logs traffic for forensic examination, when suspected malfunctions, suspicious traffic, or violations of policy are investigated. It could benefit an Evildoer to neutralize this system, and/or prevent it from recording "bad" activity. It could additionally be a rich source of intelligence, if overrun by an Evildoer, being one of the few systems, in an otherwise switched environment, that can readily "sniff" *lots* of passwords off the wire, as well as capture instant messaging traffic, email, etc[7]. The remote sensor(s) transmit their reports, and receive their configuration instructions from the CMS across the LAN they monitor at this time, although this will soon change. Because we use the monitored network as the transport for NFR command and control traffic, it is critical that the CMS be able to withstand brutal attacks. Accordingly, it will be hardened against various denials of service attacks, and will run minimal network services, to present as narrow a profile for exploitation as possible. When complete, the CMS should present only the command and control ports for the NFR system to the network, with UDP port 520[8]. All systems administration of the CMS server will be performed via serial console, and/or a private IP network, connected to the second network interface, which will be manually brought up only when absolutely necessary. The basic process involves first issuing a plumb command, then assigning an ip address and bringing the interface up:

```
#/usr/bin/ifconfig  dmfe1 plumb
#/usr/sbin/ifconfig dmfe1 inet 192.168.0.1 up
#/usr/sbin/ifconfig –a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
dmfe0:  flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4>  mtu  1500
index 2
        inet 10.1.1.113 netmask ffff8000 broadcast 144.30.127.255
        ether 0:3:ba:16:78:55
dmfe1:  flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4>  mtu  1500
index 4
        inet 192.168.0.2 netmask ffffff00 broadcast 192.168.0.255
```

[7] The NFR has huge capabilities for traffic recording, but is not now configured to capture sensitive data, such as email traffic or chat session content. Besides legal, privacy, and policy constraints, sheer overhead and performance issues exist – given the traffic on this LAN, the modest, customer-sourced sensor(s) begin dropping packets if too many back ends are enabled. Given experience with this product, I suggest buying sensors preconfigured, direct from NFR, rather than using castoff PC's – it looks good on paper, but is false economy. Get the good stuff, the preconfigured appliances.

[8] Routed is running in monitor mode on this host, so that any changes in routes detected will be logged. See the man pages for your routed implementation.

ether 0:3:ba:16:78:55

When you are finished with the interface, you can config it down, and unplumb it:
/usr/sbin/ifconfig dmfe1 down
/usr/sbin/ifconfig dmfe1 unplumb

Risk Analysis - Key Security Concerns:

Primary threats to be considered in this exercise:

- Physical security threats – theft, willful destruction, fire, or other acts designed to disable/destroy the NFR/CMS system, so that illicit actions go undetected, unrecorded, or to otherwise destroy the evidence of such actions.
- Unauthorized physical tampering with a goal of achieving covert and unauthorized access to the NFR/CMS system.
- Random hacking, viruses, worms, denial of service attacks, and miscellaneous acts of senseless vandalism not specifically targeting the NFR/CMS, but affecting it nevertheless.
- Network attacks to divert and/or read, decode, or manipulate traffic between the NFR/CMS system, and the administration interfaces it talks to.
- Network based attacks to gain unauthorized access to the CMS server and the data it collects.
- Network based denial of service attacks intended to take the NFR/CMS system off the "air", so that other actions will not be detected / recorded.
- Loss of data needed for investigations and/or troubleshooting due to hardware malfunctions, man-made, or natural disasters.
- Insider manipulations of the NFR/CMS for a variety of possible reasons.
- Risks not contemplated in our analysis – the Great Unknown.

Risk Analysis – Striking An Acceptable Balance

As discussed above, the management server must be secure, as good or better than the firewall, because it contains not only great volumes of forensic data, but huge capacity to help an attacker – owing to its special position on the network. For this system, the considerations of security militate against installation of any of the usual tools that make administration easier – X-Windows, network backup clients, remote administration tools, NFS storage resources, etc. For this reason, the only open network ports on this machine will be those required to operate the CMS, and assure it's correct operations:

- 520/udp the communication port for routed, which will be configured to watch for and log any routing table alterations.

- 1968/tcp - the communications process between the CMS and the NID(s)
- 2010/tcp - the communications process for the Administrative Interface(s)

Authorized users of this machine are limited to: One System Administrator, who installs, maintains, and operates the NFR System, the Lead Security Engineer, and his subordinate Security Engineer, who use the data to monitor network usage. Only the System Administrator logs in to the machine proper, the rest use the NFR Administration interface (AI), to configure and interrogate the remote sensors, view alerts, query logs of past events, etc. Only the root user will be given command line access to this system, therefore normal user requirements play a extremely limited role in configuring this system. For example, we will be installing an older ssh version because we already have it as a trusted source binary, we do not wish to install compilers on this platform, and we know the only use to be made of it will be to access an intermediate host, via a crossover cable, for maintenance and backup purposes. If we were supporting less aware users, we would *never* consider using this dated tool, due to known vulnerabilities. We need the machine to be secure, but we also had a finite time frame to bring it online, and balanced the fact that only a knowledgeable user would be on the machine at command line level vs. the added delay of building and validating a new ssh distribution.

Risk Analysis - Threat Mitigation Rationale and Strategy

No amount of work will protect a machine 100% of the time against every possible threat, but with thoughtful planning, and layered defenses, many risks can be mitigated considerably. A malicious person with enough time and physical access to hardware can do anything to a machine they please. In order to protect it from possible harm, it must be kept physically secure. Further, because the machine is to be managed mainly from a hardware serial port, it will be kept physically secure to protect its console from unauthorized access and manipulation. The machine will be housed in an access controlled computer facility, with 24/7 staffing, automatic fire suppression, back up power, and video monitoring / recording of all points of ingress and egress. Badge readers will control access, and log swipes at the two points of entry to the datacenter.

Because disasters do occur, data will be backed up, using the standard Solaris ufsdump and restore commands. The contents of each production partition will be saved to the /backup partition as ufsdump[9] archives, and then moved off via scp[10] to a intermediate

[9] ufsdump is the Solaris standard backup tool, which creates nice archives you can restore from interactively. It supports the notion of backup levels (0=everything, higher numbers represent essentially incremental snapshops). With this tool, some extra drive space, and cron, you can deploy an easy to use backup scheme, without opening up to

system, for subsequent backup using the normal enterprise backup scheme. For a humorous and practical primer on ufsdump, the uninitiated may enjoy David Ashley's "Practical ufsdump & ufsrestore explanations for Solaris 8"[viii]. The scheme used here involves a level 0 dump to a disk archive, which is then moved manually to a less restricted platform for backup via standard enterprise tools, and includes offsite media storage. This is where tape media is rotated to a secure, third-party facility as a hedge against a site-wide catastrophe. These backup tapes are placed in sealed containers, which are secured with unique, serially numbered plastic seals before shipment. These seals insure that any tampering with the tapes will be readily detectable. One may obtain these serialized seals from Troxler Labs[11]. A tamper evident seal will also be used, slightly trimmed to fit the card slot on the Sun, to lock down the V100's system configuration card against any tampering.

A malicious person equipped with the right skills and tools can divert, hijack, record, and playback even encrypted network sessions in a switched environment. For this reason, the production network connection to this machine will be configured with port level security, so the router port will communicate only with the MAC address of the CMS. Each network device port serving the sensor(s) will also be so configured, in order to better secure integrity of communications. You can learn more about the specifics of this procedure from several networking gear manufactuers' support sites. See Foundery Networks, Inc. (makers of Big Iron[ixx] routers), HP. Inc. (makers of ProCurve[xi] switches), or Cisco[xii], Inc. The site-specific process will not be illustrated; however, as it is performed by another group, for compartmentalization purposes[12].

A malicious person could capture and record traffic; therefore, all traffic to and from the CMS will be encrypted. This reduces the likelihood intercepted traffic will ultimately provide information that could be useful in playback, man in the middle, or other attacks. The NFR software utilizes encryption between the sensors, the CMS, and the AI/Console(s), although it is important to generate a strong encryption phrase, to maximize its value. One weakness of the NFR is that this phrase resides in clear text on the CMS host.

---

network agents, or supporting standalone tape drives, media, etc. We will provide a simple script. Review the man pages for ufsdump for details on use, restore functions, and options.

10 scp – the secure shell replacement for rcp, also known as secure copy. The command is used as follows: scp local-file user@remote.host.ip:remote/path/filename

[11]Cost is currently 10 seals for 6 Dollars. See Troxler Labs, Inc. 2002. "Tamper-evident Security Seals",URL: http://www.troxlerlabs.com/tampseal.html (July 23, 2003)

[12] Compartmentalization, as used here, means to divide roles across organizational units, so that no one group knows all of the keys, methods, and procedures used to protect sensitive information. Backbone configuration information is compartmentalized, administrators don't configure switches, network engineers don't generate host keys, etc.

Because of the increased sharing of information between attackers, even relatively unskilled persons and automated attack tools can perform sophisticated attacks against vulnerable network services. To guard against such threats, the CMS will run the bare minimum network services, and those presented must be able to withstand typical buffer overflow, format string, denial of service, and related attacks.

Because of the widespread availability of session hijacking tools, privilege escalation exploits and the like, no interactive connection via traditional tools such as telnet, ssh, X-Windows, etc. will be offered on this system. All management functions will take place from serial connections, using command line interfaces. Any file transfers will be initiated from the CMS host, and will traverse a private network consisting of a crossover cable.

To protect against Trojan horse and remote administration tools, no third party software will be installed unless it comes from a verified, trusted source. All software will have its' md5 hash validated before installation. All downloaded software or other data will go to an intermediate system, which will scan it for viruses, Trojans, and authenticity before transfer to the CMS via the crossover cable network. That network will consist of a crossover cable from the intermediate system to the dfme1 network interface of the CMS. Sun uses two Davicom DM9102A auto-negotiation network cards in the V100 series server. It appears to the system as one or more interfaces named dfme*n*, so the first instance would be dfme0, the next dfme1, etc. The dfme device driver is configured for speed settings etc by either editing the file dfme.conf, or via the ndd command. For help configuring this interface, consult SunMicrosystems, Inc. "Platform Notes: The dfme Fast Ethernet Device Driver[xiii]".P5-18.

Step by Step Configuration Guide

Hardware setup is fairly straightforward on the Sun Fire™ ™ V100, which ships as a pre-configured, pre-installed, headless UltraSPARCIIi system, housed in a 1U purple rack mount chassis. To secure the machine against unauthorized configuration changes, certain steps unique to this platform were performed. First, the system configuration card (located in the back, next to the power button) ships with a nylon wire tie holding it in place – this is replaced with a Troxler Labs serialized, tamper-evident seal, which, once locked must be cut to be removed. You have to trim down the end of the tab a bit to make it fit, but once in place it positively prevents anyone from swapping configuration cards on the machine without disturbing the seal. We record this number on our configuration notes. Next a standard category 5 station cable is plugged into the RJ45 port to the immediate right of the system configuration card port labeled "A LOM", and attached to a standard 9-pin D shell serial port with a special adapter. Sun ships

two with the Sun Fire V100, a 9 and 25 pin, we use the 9 here. The Sun Part Number for the 9-pin adapter is 530-3100-01 (an RJ45 to DB9 female serial connector). If you need a 25 pin, it is Sun Part number 530-2889-03(Rev 02), and is an Rj45 to DB 25 male adapter. If you have trouble connecting to a Sun via serial cable, let me refer you to the "Sun Source" of all Sun serial port wisdom, The *Stokely Unix Serial Resources* page[xiv], a legendary site of great value to Sun administrators, old and new alike. Once your cabling is in order, set the serial port to 9600 baud, 8 data bits, no parity bit, 1 stop bit, and no flow control. Opening a connection as a vt100 should produce a usable connection. If not, try nudging it with a enter, control-c, or other break sequence. If the machine has booted and is running Solaris, you can perform the equivalent of a Stop-A by hitting the sequence "#.". On a new machine, with the right settings for the serial port, you should eventually get a prompt like this:

```
lom>
```

If the machine has been set up and has users defined to the Lights Out Management (LOM) subsystem (much like the OpenBoot interface, a simple command line shell of sorts) you will have a prompt like this:

```
LOMlite console
Please login:
```

We will assume the former for now, and show you how next to define users to LOM, so that it requires a password to manipulate the LOM subsystem. On a new Sun FireV100, there will be no users defined, and anyone can configure the machine. This is a Very Bad Thing™ that we wish to correct, so we connect our cable as above, and open a terminal session to a V100 fresh out of the box and plugged in (without the power switch being toggled, the LOM monitor is already active). Commands are in blue, comments appear following a #.

```
lom>
lom>usershow
1: [not defined]
2: [not defined]
3: [not defined]
4: [not defined]
# no users are defined on this V100 yet.
lom>
lom>useradd root
lom>
# add user id "root"
LOM event: +4d+5h17m56s user added 1
lom>usershow root
1: root: -
```

```
2: [not defined]
3: [not defined]
4: [not defined]
# verify the user is added, do not  botch this process.
lom>userperm root cuar
lom>
#give root user all four permissions, where:
# c = console - can exit LOM, get Solaris console/shell prompt
# u = user admin – add/delete/modify users with userperm command
# a = admin permission –  configure LOM device configuration
# r = reset permission – reset server and power cycle it too.
# - = no permission assigned.
LOM event: +4d+5h18m27s user permissions changed 1
lom>usershow
1: root: acru
2: [not defined]
3: [not defined]
4: [not defined]
lom>useradd toor
lom>
#create a backup user too, just in case root gets boogered up. We will omit #the
permissions setup for toor, but it will mirror “root” when done.
LOM event: +4d+5h19m5s user added 2
lom>usershow
1: root: acru
2: toor: -
3: [not defined]
4: [not defined]
lom>password toor
Invalid command. Type 'help' for list of commands.
lom>password root
Invalid command. Type 'help' for list of commands.
#Note that LOM can't set a password until you logout and login again.
lom>logout
# this exits, now you get to login again
LOMlite console
Please login: root
Enter password: (not echoed)
lom>
#now you have to login, since one or more users exists...
LOM event: +4d+5h20m36s user logout 1
#note LOM reports your last logout – helps watch for vandals
lom>
LOM event: +4d+5h20m44s user login 1
```

```
lom>usershow
1: root: acru
2: toor: -
3: [not defined]
4: [not defined]
lom>password
Enter current password:     (it is blank, press enter)
Enter new password:         (not echoed for security)
Reenter new password:       (not echoed for security)
Password changed
lom>
LOM event: +4d+5h21m1s user password changed 1
lom>logout
#note we pressed enter for the current password (its unset), then what we #want the
new password for user root to be. Don't confuse this with the
#system / unix password, this is like a BIOS password on a Intel PC, and
#has nothing to do with Solaris proper. Be sure to escrow or otherwise
#preserve this password with the system password for DR purposes, #following your
site's policies. Loss of this password means ordering a
#new PROM from Sun and waiting for it, anxiously...

LOMlite console
Please login: root
Enter password:       (not echoed)
lom>
LOM event: +4d+5h22m38s user logout 2
#LOM keeps a count of your logins too, nice....
lom>
LOM event: +4d+5h22m41s user login 1
```

The Sun Fire V100 is now secured from having its "nvram"[13] tampered with, has a pair of user id's defined, and has password protection enabled for the LOM console. We now move on to perform a basic Solaris 8 OE installation[14]. A word of caution is in order here – Solaris can be tricky even for seasoned administrators – if you are new to Sun equipment, allow some time for *practice*. Certain Solaris 8 installations, on certain architectures of Sun equipment, will fail to configure themselves to automatically boot unless an OpenBoot variable is set correctly *before* the installation, with either eeprom (from Solaris command line) or setenv (from ok prompt). After installation, an error message appears, then the system will try, and fail, to boot:

[13]The System Configuration Card contains site customizations, including the MAC address the machine reports to the network, and the system IDPROM.
[14]For this project we will perform a "core" installation, which leaves out a tremendous amount of software that otherwise complicates hardening.

```
Installing boot information
        - Installing boot blocks (c0t0d0s0)
        - Updating system firmware for automatic rebooting
WARNING: Could not update system for automatic rebooting.
```

Thanks are due to certain intrepid system administrators, “Osama Ahmed”, “Charles Gagnon”, and others, who posted[xv] about this problem on the Sunmanagers[15] mailing list. From them we learn that we must insure the correct setting of the *diag-switch?* OpenBoot variable *before* booting the installer off Disk 1 of 2 of the Solaris 8 Software set. Note this is *NOT* the CD labelled “Solaris 8 Installation” CD, but the first one of two in the Installation set proper. Be careful with the eeprom / setenv commands, with them, it is possible to *really* hose things up. A good methodology is to check each settings current value, record it, then alter it and see what happens. This way, you can back out of bad changes gracefully. There is a man page for eeprom that is helpful. To fix this particular problem and thereby insure post-install booting on a Sun Fire V100, type:

```
# eeprom diag-switch?
diag-switch?=true
# eeprom diag-switch?=false
# eeprom diag-switch?
diag-switch?=false
#
lom>break
Type 'go' to resume
ok boot cdrom
```

Time to reinstall a fresh copy of Solaris 8. Using the break command to perform the equivalent of a Stop-A key sequence on a keyboard-equipped Sun, we drop to the *ok* prompt, boot from the cdrom and begin installation. The system resets, then OpenBoot goes to work. Now is a good time to record your serial number, hostid, and also your MAC address – your network engineer will want it for switch port programming most likely. Some software requires the hostid to generate a license key, so it is good stuff to have documented. You will see something like this:

```
Sun Fire V100 (UltraSPARC-IIe 500MHz), No Keyboard
OpenBoot 4.0, 1024 MB memory installed, Serial #5nnnnnnn.
Ethernet address 0:3:nn:nn:nn:nn, Host ID: 83nnnnnn[16].
```

[15] The S*unmanagers* mailing list is *indispensable* for newbies to Sun hardware. You can subscribe here:
http://www.sunmanagers.org/mailman/listinfo/sunmanagers

[16] All unique information is sanitized to obscure host specific configuration information.

The installer tries to detect your terminal type, and with luck, the terminal mapping is decent, and your screens come across clear enough to navigate. First thing you select are Language and Locale environment settings, then are asked to approve terminal type. ANSI and VT100 seem to work equally well with most terminal emulators, if you have a real serial terminal, something else may be in order. Most screens are answered with function keys. Sadly, most function keys don't respond, so you must use the escape key and a number key together to hobble along. Escape+2 generally accepts the screen and escape+4 generally is the customize option. Escape-6 is help. Menu options are selected/deselected with the space bar. There are exceptions, read each carefully until completely comfortable. The arrow and tab keys are used to navigate around, especially in the network and disk partitioning menus. Configuration proceeds with basic network configuration information, and a choice of interfaces. Interface dmfe0 will be our LAN attached interface, with dmfe1 used only for private transfers of patches and other administrative functions temporarily requiring a network. We will not use IPv6 or Kerberos for this system, and configure accordingly. You will see a summary screen before you actually commit the network settings. When presented with name service type, select “none” for now, and configure DNS manually if you need it. Time Zone and date settings come next, configure appropriately for your site. Next we see a quaint little notice:

```
Starting Solaris installation program...
Searching for JumpStart directory...
not found
Warning: Could not find matching rule in rules.ok
Press the return key for an interactive Solaris install program...
```

Press return. If the system has ever had Solaris on it, it will ask if you wish to upgrade or do an initial install – choose initial to repartition and overwrite any existing data, then select the “Standard” installation option. You will then be asked about Geographic region support. Customizing packages for installation is best described as tedious, but you need to do it here. You will then see a screen like this:

```
Select the Solaris software to install on the system.

NOTE:After selecting a software group, you can add or remove software by
customizing it. However, this requires understanding of software dependencies
and how Solaris software is packaged. The software groups displaying 64-bit
contain 64-bit support.

[ ] Entire Distribution plus OEM support 64-bit 1503.00MB
[ ] Entire Distribution 64-bit .....1468.00 MB
[ ] Developer System Support 64-bit .....1416.00 MB
[ ] End User System Support 64-bit .....991.00 MB
```

```
[X] Core System Support 64-bit .....408.00       MB(F4 to Customize)
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqq
F2_Continue    F3_Go Back          F4_Customize        F5_Exit    F6_Help[X]
```

Selecting the “core” installation set is a good start, then use escape-4 to customize it, removing everything you can that isn't marked with a ! or a /. Some experimentation is necessary. In the end, for this install, the core set comprised about 110M of files in 107 packages. We will remove some more packages later in the process. For now, with the machine isolated as it is, we can leave well enough alone, and focus on adding what we need to complete the hardening process. Next use spacebar to select the disk or disks you wish to perform the install on

```
Disk Device   (Size)Available              Space
=============================================
[X]c0t0d0   (38162 MB)     boot disk      38162 MB  (F4 to edit)
[X]c0t2d0   (38162 MB)                    38162 MB
```

Now we partition the hard disk(s) – this machine has two physical drives, so we select each in turn. We configure one, c0t0d0, as the boot drive, and the other, c0t2d0, as a backup disk for keeping copies of things we want to make sure we have in case of failures etc. Solaris physical device names can be daunting even for professionals to decrypt, and are beyond the scope of this work. Have a look at the entry for the root disk:

```
#ls -la c0t0d0s0
total 1
lrwxrwxrwx 1 root root  42 Jul 23 14:38 c0t0d0s0 ->../../devices/pc
i@1f,0/ide@d/dad@0,0:a,raw
```

Pretty wild stuff, but you see the pattern - a pci controller, an ide disk, and a raw device? Logical device names are somewhat less baffling, and are used more often by administrators. The disk-partitioning tool will use them too. For our boot disk, parsing left to right past /dev and /rdsk, we encounter /c0t0d0s0 – which breaks out as:

```
c0 = controller        0
t0 =  target number  0
d0 = disk  number    0
s0 = slice number    0
```

So, c0t0d0s0 = controller 0, target 0, disk 0, slice 0. See the pattern? For years Suns were SCSI-centric, and it still shows. The installer then tells you the original boot device, and asks you to select the boot device, which is the device the / partition is on, in this case c0t0d0.

[X]c0t0d0 (F4 to select boot device)

Next you are asked if you want to reconfigure the system hardware (EEPROM) to point to the new boot device, as discussed previously. Generally you answer “yes” to this question, unless you have some reason to do otherwise, and are sure that you do not want to do this. If there are previously made file systems on the disk(s), the installer will ask if you want to preserve existing data. For this install the answer is “no”. Next you are asked if you want auto-layout to layout your file systems, and warns that manual layout requires “advanced” skills. For this install we press esc-4 and manually configure the file systems.

File system/Mount point	Disk/Slice	Size
/	c0t0d0s0	2000 MB
/usr	c0t0d0s1	2500 MB
overlap	c0t0d0s2	38162 MB
swap	c0t0d0s3	1000 MB
/opt	c0t0d0s4	1000 MB
/var	c0t0d0s5	2000 MB
/nfr	c0t0d0s6	29661 MB

The installer asks if you want to mount software from a remote server (we did not), gives a final summary of your configuration, and gives you the choice of automatically rebooting after installation or not. Answer that question and press esc-2 to being installation. When installation ends, the system reboots. Upon reboot, we next configure two items that apply to all SPARC systems, an eeprom password, and a eeprom warning banner. This is accomplished, naturally, with the eeprom command:

```
#eeprom  oem-banner
oem-banner: data not available.
```

This should return “data not available” unless a banner has been set already. To set one, you enter the command this way:

```
#eeprom oem-banner=”This system for authorized use, access is \
 monitored, use in excess of authority is prohibited. Call 555-555-1212  \
 for more information on policies and prohibitions.”
```

As with all banners, if in doubt, ask first before installing one, and be ready to defend your request with facts. Verify the setting is enabled with this command:

```
#eeprom oem-banner?
```

This will return the value false until set equal to true like this:

```
#eeprom oem-banner?=true
```

Set the security-mode with:

```
#eeprom security-mode=command
Changing PROM password:
New password:
Retype new password:
```

We can select three settings for this variable, none, full, or command. We don't use full generally, as the system will not auto-boot without human intervention to enter the password. If you need that behavior, this is where you set it up. None is out too, that leaves us wide open to a host of devilishness, so we choose command mode security. Be sure and read the following warning, twice, before you do this. You will probably have to reset this someday, but remember this is not like the root password at all: If you lose or forget this password, you have to order a new PROM from Sun, and install it before the system will work again. This changes the HOSTID from what I hear, and that affects some software licenses. Don't expect management to understand if their pet system is down over the weekend while you sort through all this esoteric stuff. If you have a critical application running on SPARC equipment, having a spare PROM around the shop is not a bad idea at all. This ritual is conducted with great circumspection, and a notepad. That said, the magic command:

```
#eeprom security-password=
Changing PROM password:
New password:
Retype new password:
```

The next time you boot the V100 (or similar machines) and are at the LOM> prompt, you can test this out as follows:

```
lom>break
/
Type  'go' to resume
Type boot , go (continue), or login (command mode)
> login
Firmware Password:
Type  help  for more information
ok boot
```

You are now in command mode, as if you pressed Stop-A on a regular Sun. Type boot to reload the system. Remember the LOM user and the eeprom user are two different

abstractions, each with unique, if somewhat overlapping capabilities. The next step is to load some useful tools that are not present on the system - gzip and md5. The machine is connected via crossover cable to a Mac iBook used to stage "known good"[17] installation files. We use a crossover cable for both simplicity and security. One clear explanation for what a crossover cable does which I ran across at "littlewhitedog.com", put it this way:

> "when you connect two machines together without the use of a hub or switch, a crossover cable is required - because both 'ends' are essentially the same - a NIC Card. The crossover function must take place somewhere, and since there is no hub or switch to do it for you, the cable must[xvi]. (Littlewhitedog.com).

The Macintosh iBook, with appropriate software, provides a full Unix environment, including both X11 as well as their gorgeous Aqua interface, in a very small package. Using ftp initiated *from* the Sun, we fetch a known good copy of gzip for Solaris 8, and a package containing md5 via the crossover network. Trust has to begin somewhere, and md5 gives us the ability to validate our files are unsullied, by comparing previously calculated signatures against the files moved to the Sun. We maintain an archive of binary freeware packages that have been built locally, or obtained from otherwise trusted sources. By comparing the md5 signature against our list of those signatures, one can be reasonably sure that the file in question is an un-tampered with copy of the desired original. This is a chicken-or-egg situation, in that we have to have md5 installed before we can trust gzip. Using the pkgadd command we install the md5 package. It is an interactive process. Inputs and answers are in blue type, as are comments.

```
# pkgadd -d ./md5-6142000-sol8-sparc-local
The following packages are available:
  1  SMCmd5     md5          <-program name
             (sparc) 6142000  <-version number
Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]: 1
Processing package instance <SMCmd5> from </backup/md5-6142000-sol8-sparc-
local>
md5
(sparc) 6142000
RSA Data Security
The selected base directory </usr/local> must exist before
```

[17]For the purposes of trusting software, the term "known good" in this paper means files that are known to be from legitimate sources, have been either locally built from source code that has been approved for production use, bought from approved vendors and received through trusted channels of commerce, or otherwise verified as to source and content, and for which we have appropriate usage licenses in place.

Installation is attempted.
Do you want this directory created now [y,n,?,q] y
Using </usr/local> as the package base directory.
Processing package information.
Processing system information.
Verifying disk space requirements.
Checking for conflicts with packages already installed.
Checking for setuid/setgid programs.
Installing md5 as <SMCmd5>
Installing part 1 of 1.
/usr/local/bin/md5 <- pkgadd now tells where it is putting things, nice.
/usr/local/doc/md5/README
/usr/local/doc/md5/md5-announcement.txt
/usr/local/doc/md5/md5.1.ps
/usr/local/doc/md5/md5.1.txt
/usr/local/doc/md5/rfc1321.txt
/usr/local/doc/md5/test.rfc
/usr/local/man/man1/md5.1
[verifying class <none>]
Installation of <SMCmd5> was successful. <- A Good Install!

With md5 now in /usr/local/bin, we check our gzip program against the md5 signature from our list of "known good" archive file signatures on the Mac iBook:

[gmarkhams-Computer:~] root#cat gzip.md5
MD5 (gzip) = dc87b6b52c0d0b8de36bd70b0f48c062
#
and compare with values on the Sun:
/usr/local/bin/md5 ./gzip
MD5 (./gzip) = dc87b6b52c0d0b8de36bd70b0f48c062

The gzip we have checks out as uncorrupt, so next we patch Solaris using the current Recommended and Security Patch set, downloaded from Sun[18] and staged on the Mac previously. It is best to patch before hardening, or making any other changes on the system because, sometimes, patches install or enable things we might not want them to. An example of such behavior is cited in the Sun "BluePrints Online" publication *Operating Environment Minimization for Security*, by Alex Noordergraaf[xvii], where it was reported:

> The kernel patch, 106541 for Solaris 7 OE, is an example of why patches must

[18]Get the recommended patches for Solaris 8 from Patches for the OS from Sun Microsystems, Inc. 2003.URL:http://sunsolve.sun.com/pub-cgi/show.pl?target=patches/patch-license&nav=pub-patches (July 25, 2003)

be installed before any minimization or security hardening is performed. The README and pkgmap of this patch shows that the following files will be updated when the patch is installed:

- /etc/rc2.d/S71rpc
- /etc/syslog.conf
- /etc/init.d/rpc

The presence of any of these files may either enable a service that has previously been disabled (rpc, automounter, or volume manager) or overwrite a file with specific configuration information in it (syslog.conf).

Think about the implications of Mr. Noordergraaf's statement. It is worth noting that if, as part of your hardening process, you had configured a remote syslog server before patching, your changes would have been wiped out by the patch. Turning on RPC is not a Very Good Idea™. Patch first, harden after - that should be the standard operating procedure. If you must patch later on, better save system state first, then compare everything afterwards for changes.

While we considering /etc/syslog.conf, let us add a remote loghost to our file, so a copy of all syslog events get sent to another host. Hackers in the process of rootkitting a box have been known to bolt for the door when their kit reports a remote loghost is being fed syslog info. Much like a Beware of Dog sign, the @hostname line in your syslog.conf will deter a lot of invaders. This is because even if they neutralize your primary defenses, smash your stack, and take over the target system, they also have to take over the remote loghost, or accept that someone may already be alerted to their activities. For many, the risk is just too high, so they bail. Or clobber your box, and then bail. I go ahead and put in an entry for high detail logging. You could run more than one remote loghost too, maybe sorting by the reporting facility, etc. Oh, one caveat: be sure and use tabs, not spaces, for the whitespace in your syslog.conf. See the syslog.conf man page for your system if you are curious why this is. To add the remote log host entry:

```
echo "*.debug         @10.1.1.1" >> /etc/syslog.conf
```

This is a good time to install a few optional packages that will make life easier on us, particularly when it comes to file transfers. Even though we are not going to run a sshd on the box, we want an ssh client available for doing things like moving backup files off the system, grabbing new patches from the intermediate host, etc. We will be installing the Openssh that bundles with YASSP, Jean Chouanard's Solaris Security Package. This version is older, version 2.30p1, and no longer recommended for use on hostile networks. If this system were going to run the ssh daemon, sshd, and offer interactive logins, we would take the time to add the several other packages required to run a

current version of OpenSSH. We would also have to configure a privilege separation user as well, frustrating the goal of operating system minimization further. To keep the installed packages to an absolute minimum, a value judgment is being made here, and simplicity is winning this one. We will, therefore, disable execution of sshd after installation, and will never communicate with the old client across anything but the crossover cable to the iBook. Like Clint Eastwood once said in some movie of his: "A man has to know his limitations.". So too should we realize the limits of our technologies, and respect them. Also required for this option will be the installation of SUNWzlib and SUNWzlibx – the 32 and 64 bit compression libraries. We first add these packages, off the Solaris 8 Software cd, disk 2 of 2. Mounting the cdrom, we then use pkgadd to load the packages:

```
#mount –F hsfs –o ro /dev/dsk/c0t3d0s2 /cdrom
#cd /cdrom/Solaris_8/Product
#pkgadd –d . SUNWzlib SUNWzlibx
```

The Installer asks for some answers, yes for them all works fine. We then cd into the YASSP[19] directory, and install just the package for openssh:

```
#cd ./yassp
#pkgadd –d ./openssh_sparc
```

Note that this will install startup scripts in /etc/rc2.d and /etc/init.d that will have to be turned off to prevent sshd from starting automatically at system boot. Disable them by renaming and removing execute bits:

```
#mv /etc/rc2.d/S75sshd /etc/rc2.d/OUT.S75sshd; chmod –x /etc/rc2.d/OUT*
#mv /etc/init.d/sshd /etc/init.d/OUT.sshd; chmod –x /etc/init.d/OUT*
```

Next we need to examine, and possibly configure /etc/ssh_config, and disable /usr/local/sbin/sshd from executing as well. The ssh_config file sets options for ssh clients, using defaults configured into the client unless overridden in the config file. For complete information on this file, see the online manual pages[xviii]. You may want to generate keys for yourself using ssh-keygen. Disable sshd execution with

```
#chmod –x /usr/local/sbin/sshd
#mv /usr/local/sbin/sshd /usr/local/sbin/OUT.sshd
```

[19] YASSP – Yet Another Solaris Security Package, by Jean Chouanard, et al. The package untars into ./yassp. Get your copy at URL: http://www.yassp.org . It bundles a number of useful programs, RCS, WVTCPD, GZIP, and ASR-TRIPWIRE to name a few. An older tool now, but still great stuff, done by some of the most knowledgeable Solaris folks around.

We also add the freeware lsof[20] package to our box, for use in examining the open files on the machine in depth. Our lsof was obtained from SunFreeware.com, and was approved for production use after signature checking, examination, and testing in the lab. More on its use later, when we validate our configuration, and setup ongoing maintenance processes.

Next we will remove a number of Solaris packages that are not needed for this installation. For information on exactly what specific packages are, see the Solaris 8 2/02 Operating Environment Package List[xix]. First we list the installed packages using pkginfo, and redirect to a textfile in /tmp:

```
#pkginfo > /tmp/pkglist; cat /tmp/pkglist|wc –l
110
```

We then read this list, comparing each package to the Sun Package List for our distribution (2/02) and with some educated guessing, we try to remove each unnecessary package, avoiding any related to device drivers that we know we need and required stuff. This can be difficult. In Table 1, at the end of this work, I offer the full list merely as general guidance; your mileage will vary – possibly wildly! First we printed the original package list, it came to a total of 110 packages. After careful work, we reduced this to a total of 57 packages! See Table 2, at the end of this work, for the final listing. This is a significant reduction in overall code bloat, and all the potential vulnerabilities associated with it. For an excellent read on the techniques of minimization, see Alex Noordergraaf's write up for Sun Blueprints Online[xx]. Mr. Noordergraaf points out something that often wholly escapes the attention of wood-be system hardeners, that is, the nagging question, did your installation actually complete correctly:

> "…it is important that the installation logs on the server be examined for any errors or configuration problems." He goes on to share that "The begin.log contains all pre-OS installation operations, while the finish.log contains all post-OS installation steps. Usually the finish.log contains the most pertinent messages." (Noordergraaf, P11)

Trying to harden a broken installation is no fun at all, so we dutifully cat the logs looking for anything obviously broken. While speaking to a Jumpstart installation, the logs for our install are located in the same place: /var/sadm/system/logs. Essentially it boils down to a bit of educated guessing, and much reading of documentation – if it sounds

[20] Lsof lists all open files on your machine, and is a fantasic way to see what is working with what on your box. Get your lsof Solaris binary from SunFreeware.com, URL:http://www.sunfreeware.com . As with any pre-built binary, check the thing out before you put it on a critical machine. We originally examined this one in the lab for a while, sniffing traffic and trying various things until we were satisfied. Building from code is best if - you have time and spare platforms.

useful to the OS, leave it alone, and remove one package at a time till it breaks something. I might have culled more, and got away with it, but when I took the SUNWaudd package out, the kernel cried about missing symbols, so I stopped, thankful it would still boot. Again, your mileage will vary, and vary by hardware platform, installed hardware options, and lunar phase most likely. This stuff really moves you into the Realm of the Mystics, and qualifies you for your Guru's Turban:

> "It's easy to make your own wizard's pointy hat ; the turban a little trickier. You can find both in various stores.[xxi]" (Nemeth, Mike. "Hats for System Administrators; more SA tools").

For extra credit, check out the platform this says it was written on using view/source. Yes, Mike looks like *a* real Unix Guru too, from his website, check out his resume. I like his musings; this guy must be a real Living Treasure. His tactics work, too.

Now we move on to patching, note that the Recommended and Security patch set is fairly large, so unless using the SunSolve CD's, better plan on a bit of download time. The patches arrive as a .zip archive, so we use the native unzip tool to decompress them. The result is a directory structure called 8_Recommended, which is full of subdirectories named for the patch numbers they contain, e.g. 108901-08. A file, CLUSTER_README, contains a list of patches, what they do, and etc. It is best to read it. Patching is one of those tasks best performed in single user mode, so we type:

```
# init S <--- Tell system to go to Single user mode
#INIT: New run level: S
The system is coming down for administration.  Please wait.
Unmounting remote filesystems: done.
Killing user processes: done.
INIT: SINGLE USER MODE
Type control-d to proceed with normal startup,
(or give root password for system maintenance):(passwd) <--enter
single-user privilege assigned to /dev/console.
Entering System Maintenance Mode
Jul 28 14:03:58 su: 'su root' succeeded for root on /dev/console
Sun Microsystems Inc.   SunOS 5.8       Generic Patch   October 2001
#./install_cluster  <--- start install, and answer prompting questions
```

The system will inform you that it needs a certain amount of free disk space, and will then go about its work patching the system. Path numbers will scroll by, indicating success or failure as each patch is processed. Some patches will fail, as all are not appropriate for the configuration of all systems. Get a soda and watch for this message:

```
For more installation messages refer to the installation logfile:
 /var/sadm/install_data/Solaris_8_Recommended_log
```

```
Use '/usr/bin/showrev -p' to verify installed patch-ids.

Refer to individual patch README files for more patch detail.
Rebooting the system is usually necessary after installation.
```

Reboot your system and see how it goes. Note that, as Seán Boran reports[xxii], the showrev command is included in packages (SUNWadmfw, SUNWadmc) that do not install on the "core" installation, so you must use patchadd -p to list your patch information on a "core" Solaris 8 install. So far, so good. Now it is time to apply the Solaris Security Toolkit, "JASS". JASS is a security tool that examines your Solaris system and brings it to a known security state. The version we are applying is 4.0.0. JASS is available from Sun as either a compressed tar, or Solaris package file, and ships with adequate documentation to get you going quickly. This installation uses the package version, which is installed with pkgadd. Once the package installs, you invoke JASS from its installation directory as follows:

```
#./jass-execute -d secure.driver
```

JASS produces a lot of output, and is somewhat self-documenting as a result. Log the console output to a file and you have an excellent configuration reference that tells every action taken by JASS. JASS will reset the root password to "t00lk1t", so when it completes, you want to reset it to whatever your local policy requires and then reboot. If you want to control this behavior up front, follow the directions in the JASS FAQ, which provides:

> To override this default behavior, define the " JASS_ROOT_PASSWORD " variable in the user.init file. The desired encrypted password should be assigned as the value for this variable. [xxiii]"

On startup, the first thing we notice is a *lot* more console activity, as the system reports the various tweaks being applied to it now:

```
Setting /dev/arp arp_cleanup_interval to 60000
Setting /dev/ip ip_forward_directed_broadcasts to 0
Setting /dev/ip ip_forward_src_routed to 0
Setting /dev/ip ip_ignore_redirect to 1
```

and so, on it goes, for a page or so. Each of these tweaks has been well documented on the net, a Google™ search on the character string of each reveals a world of information. The author's experience has shown that the JASS defaults for the secure.driver have been well researched by some impressive Sun Gurus, and are very hard to beat. I enjoy experimenting, but in production use, I go with the JASS suggestions every time. Eventually you will receive the new warning banner, just installed by JASS, which you can (and should) customize by editing /etc/issue:

```
|-------------------------------------------------------------------------------|
| This system is for the use of authorized users only.                          |
| Individuals using this computer system without authority, or in               |
| excess of their authority, are subject to having all of their                 |
| activities on this system monitored and recorded by system                    |
| personnel.                                                                    |
|                                                                               |
| In the course of monitoring individuals improperly using this                 |
| system, or in the course of system maintenance, the activities                |
| of authorized users may also be monitored.                                    |
|                                                                               |
| Anyone using this system expressly consents to such monitoring                |
| and is advised that if such monitoring reveals possible                       |
| evidence of criminal activity, system personnel may provide the               |
| evidence of such monitoring to law enforcement officials.                     |
|-------------------------------------------------------------------------------|
```

Warning banners are, generally, a Very Good Thing™, and are often required to prosecute abuses of computer systems, although the policy varies widely. Just what is, and is not a proper warning banner can be a complex question. Arkansas, the state in which the subject system resides, recently delegated warning banner standards to The Arkansas Office of Information Technology, which defines them as:

> "A warning banner is verbiage that a user sees at the point of access to a system which sets the right expectations for users regarding acceptable use of a computer system and its resources, data, and capabilities[xxiv]."

Boran, supra, also reports that JASS can install patches for you, if you place them in the Patches subdirectory of the JASS installation directory. I prefer to maintain total control of such processes, for reasons stated previously, and do all my patching manually – before running JASS. Ongoing maintenance patching will require you to re-run JASS to re-validate your security settings. Your mileage will certainly vary. JASS is not the only such tool either, there are two others that come to mind, YASSP, previously mentioned, and Titan. YASSP includes OpenSSH, Academic Source Release TripWire, and some other packages[21] that make it a good place to start, especially if you don't have the time or equipment to build your own executables. For Sun system hardening, the tool Titan is real spiffy. Titan is the creation of many folks work, currently distributed under copyright held by Brad M. Powell, Dan Farmer, and Matthew Archibald. It is a collection

[21] YASSP – Yet Another Solaris Security Package, by Jean Chouanard. Recent vulnerabilities in OpenSSH would militate against installing older versions for use in hostile networks. As mentioned before, you get YASSP at: URL: http://www.yassp.org (Aug 5, 2003).

of scripts that tighten up security on Solaris systems. These scripts were named Titan, according to the fish.com website because:

> "I could go off and give the ``its an acronym'' speech about it standing for ``Toolkit for Interactively Toughening Advanced Networks and Systems'' but to tell the truth, I just kept getting requests from my fellow workers at the time (hi Pete, hi Alan) for ``those scripts to tighten down the OS''. The name just sort of stuck."[xxv]

After downloading Titan and it's md5 signature we check the md5 signature for our file:

```
# /usr/local/bin/md5 Titan,v4.0BETA6.tar.gz
MD5 (Titan,v4.0BETA6.tar.gz) = 2d019b53d4a14dcf623974c81544919e
# cat Titanchecksum
MD5 (Titan,v4.0BETA6.tar.gz) = 2d019b53d4a14dcf623974c81544919e
```

The md5 checks out, so we untar the archive, and examine the documentation. Titan Version 4, beta 6 is moved to the location where it will reside (/usr/titan4b6 on the subject system) and is initialized with:./TitanConfig –i. When asked if you want Titan to backup all the files it modifies, I generally answer yes. The script advises, and rightly so, that you must protect these backup files from unauthorized access, as /etc/shadow[22] is among the items Titan saves. Look for these files in the directory where Titan lives, under ./Backup, organized by run date. An examination of the files in this list will tell you what files Titan proposes to manipulate. The next step is to customize the Titan configuration files for your systems needs. Look over the files named sample.*, where * is server, firewall, etc. Select one that is approximately appropriate for your security needs, and use it as a starting place. You can then run Titan against it, and examine the output to see what Titan proposes to do to your system. I find the sample.firewall serves my needs well for hosts such as the subject system. Once you work through a configuration you like (and that will work for your machine), you can change the –v flags in the customized configuration file to –f to make Titan actually apply the proposed fixes. Feed the file to Titan as follows:

```
#Titan –c ./config.file
```

Titan will then check and correct things on your system, using the modules supplied in the configuration file. This will take a while, longer for larger file systems but not unduly long. The run on the subject system was about 4 minutes or so. You can then run:

```
#TitanReport user@host.com
```

[22] /etc/shadow is where the encrypted password hashes are stored on many Un*x systems, if you have these, a computer, and time, you can extract the clear text passwords using any of a number of password "retrieval" programs.

This will generate a report on your system and mail it to the address given. In short order you will have a report of your systems state in the mail. We now reboot the machine to insure all settings take effect. Please note that later in this process, we will be making the /usr filesystem read only – this will break TitanReport (and anything else that wants to write to /usr), so install it on a read-write filesystem if you wish to run reports as part of ongoing system maintenance. Upon logging back in, we find that, indeed, we have a very compact, trim system:

```
# ps -ef
    UID   PID  PPID  C    STIME TTY      TIME CMD
   root     0     0  0   12:15:49 ?           0:17 sched
   root     1     0  0   12:15:49 ?           0:00 /etc/init -
   root     2     0  0   12:15:49 ?           0:00 pageout
   root     3     0  0   12:15:49 ?           0:00 fsflush
   root   299     1  0   12:16:19 ?           0:00 /usr/lib/saf/sac -t 300
   root   302   299  0   12:16:19 ?           0:00 /usr/lib/saf/ttymon
   root   307   300  0   12:20:05 console     0:00 -sh
   root   152     1  0   12:16:16 ?           0:00 /usr/sbin/in.routed.orig.0730031209 -q
   root   391   307  0   12:38:06 console     0:00 ps -ef
   root    50     1  0   12:15:54 ?           0:00 /usr/lib/sysevent/syseventd
   root   291     1  0   12:16:18 ?           0:00 /usr/sbin/auditd
   root   257     1  0   12:16:17 ?           0:00 /usr/sbin/inetd -s -t
   root   300     1  0   12:16:19 console     0:00 /usr/bin/login
   root   270     1  0 12:16:18 ?       0:00 /usr/sbin/cron
   root   267     1  0 12:16:18 ?       0:00 /usr/sbin/syslogd -t
   root   282     1  0 12:16:18 ?       0:00 /usr/lib/utmpd
```

As you can tell, there is precious little running on this machine at the moment - an average Solaris default install will generally list 80+ processes. Touching the file /etc/notrouter will keep the machine from thinking it is a router automagically, so we do that now with #touch /etc/notrouter <enter>. The configuration file for inetd has been pared down significantly from the default version, which runs over 180 lines, by these wonderful hardening scripts:

```
# Configuration file for inetd(8).  See inetd.conf(5).
# To re-configure the running inetd process, edit this file, then
# send the inetd process a SIGHUP.
# Internet services syntax:
#<service_name> <socket_type> <proto> <flags> <user> <server_pathname> <args>
#
# Ftp and telnet are standard Internet services.
#
ftp        stream  tcp     nowait  root    /usr/sbin/in.ftpd      in.ftpd -l
```

```
telnet   stream  tcp     nowait  root    /usr/sbin/in.telnetd    in.telnetd
```

Notice only the entries for telnet and ftp remain in the once large file! We are not going to be running these daemons either, so inetd will be deactivated as well. The inetd process starts in two places, /etc/init.d/inetsvc, and /etc/rc2.d/S72inetsvc, so lets have a look at them - note there are several versions now that JASS has run:

```
#ls  /etc/init.d/inetsv*
/etc/inetsvc
/etc/init.d/inetsvc.JASS.20030729120533
/etc/init.d/inetsvc.ORIG.200307301209
```

JASS has already pared it way down, so that now we have a rather minimal script left:

```
#cat /etc/init.d/inetsvc
#!/sbin/sh
case "$1" in
'start')
/usr/sbin/ifconfig -au netmask + broadcast +  <-This appears redundent
/usr/sbin/inetd -s -t
;;
'stop')
        /usr/bin/pkill -x -u 0 'in.named|inetd'
        exit 0
        ;;
*)
        echo "Usage: inetsvc { start | stop }"
        exit 1
        ;;
esac
```

The S72inetsvc script looks exactly the same. There is nothing left but inetd control stanzas, except one lone ifconfig command, the need for which is not readily apparent. On this install, the inetd service startup was disabled by renaming the /etc/init.d/inetsvc to OUT.inetsvc and running chmod -x on it to disable execution. Ditto the /etc/rc2.d/S72inetsvc file. It is possible that the lone ifconfig might be needed at some point in time by something, but after reboot an ifconfig -a showed the dmfe0 interface up and happy with proper netmask etc:

```
# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232
index 1  inet 127.0.0.1 netmask ff000000
dmfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu
1500 index 2 inet 10.1.1.113 netmask ffff8000 broadcast 10.1.127.255 ether
```

0:3:xx:xx:xx:xx

The IP and MAC addresses shown are altered for site confidentiality. As you may notice, much of this stuff is trial and error at first. There is probably an instance where that ifconfig command is needed, so we save the script just in case. As the final bullet for inetd, we rename and chmod -x the executable in case someone ever gets a rootshell and tries to start it manually:

```
#mv /usr/sbin/inetd /usr/sbin/OUT.inetd
#chmod -x /usr/sbin/OUT.inetd
```

Then we create /etc/gateways , put a norip directive for each interface, and set the defaultrouter file as we want it to be for production use. This is to prevent Solaris from running Router Information Protocol, an internal gateway protocol for sharing routing information on a LAN.

```
#echo “norip dfme0” > /etc/gateways
#echo “norip dmfe1” >> /etc/gateways
#echo “10.1.1.1” >> /etc/defaultrouter
#mv /usr/sbin/in.rdisc /usr/sbin/OUTin.rdisc
#chmod -x /usr/sbin/OUTin.rdisc
```

Next we want to edit /usr/sbin/in.routed.orig.*timestamp*[23] so that the line with routed -q becomes routed -q -v /var/adm/routelog. This produces some reassuring, timestamped information for us in /var/adm/routelog:

```
Jul 30 14:19:41 ADD dst 10.1.0.0 via 10.1.1.113 metric 1 if dmfe0 state
INTERFACE|CHANGED|SUBNET UP
```

In the event routing ever changes, that change will be reflected in the log file. Now lets have a look at the active processes on the system:

```
#  ps -ef
    UID   PID  PPID  C    STIME TTY      TIME CMD
   root    0    0  0 14:19:32 ?       0:17 sched
   root    1    0  0 14:19:32 ?       0:00 /etc/init -
   root    2    0  0 14:19:32 ?       0:00 pageout
```

[23]Titan renames the in.routed executable to in.routed.orig.timestamp, where timestamp is the time Titan ran, and invokes routed with the -q option so that routing information changes are ignored. We want to add a logfile option, just in case anything funny ever persuades routed to change its information, there will be a record of it. We do this by adding -v /var/adm/routelog to the command after -q, then making sure that file exists by touching it.

```
   root     3     0  0 14:19:32 ?        0:01 fsflush
   root 294     1  0 14:19:43 ?        0:00 /usr/lib/saf/sac -t 300
   root 297 294  0 14:19:44 ?        0:00 /usr/lib/saf/ttymon
   root 307 298  0 14:49:39 console 0:00 ps -ef
   root 295     1  0 14:19:43 console 0:00 /usr/bin/login
   root   50     1  0 14:19:37 ?          0:00 /usr/lib/sysevent/syseventd
   root 298 295  0 14:19:58 console  0:00 -sh
   root 276     1  0 14:19:43 ?         0:00 /usr/lib/utmpd
   root 287     1  0 14:19:43 ?         0:00 /usr/sbin/auditd
   root 151     1  0 14:19:41 ?        0:00 /usr/sbin/in.routed.orig.0730031209 -q –v
/var/adm/routelog
   root 262     1  0 14:19:43 ?        0:00 /usr/sbin/syslogd -t
   root 269     1  0 14:19:43 ?        0:00 /usr/sbin/cron
```

Note also that syslog is set to deny communications from off-system with the -t option. Two of the processes are our own login and ps, so this is a trim machine indeed. A quick check with netstat -an shows only routed on UDP port 520:

```
#netstat -an

UDP: IPv4
   Local Address        Remote Address     State
-------------------- -------------------- -------
      *.520                                Idle
      *.*                                  Unbound
      *.*                                  Unbound

TCP: IPv4
   Local Address        Remote Address    Swind Send-Q Rwind Recv-Q  State
-------------------- -------------------- ----- ------ ----- ------ -------
      *.*                  *.*              0      0 24576      0 IDLE
      *.*                  *.*              0      0 24576      0 IDLE

TCP: IPv6
   Local Address                 Remote Address              Swind Send-Q R
wind Recv-Q   State      If
--------------------------------- --------------------------------- ----- ------ ----- ------ ----------- ---
--
      *.*                           *.*                        0      0 2 4576      0 IDLE
```

Now we will scan the machine for active listeners and record the profile for future reference. We use nmap[24] version 3.0, running on FreeBSD 4.7[25] for our scanner:

[24] nmap is a great portscanner and much more. For a copy, go to URL:

```
# nmap -sT -p 1-65535 10.1.1.113
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
All 65535 scanned ports on  (10.1.1.113) are: closed
```

It appearing that we have tightened the system up, we now move on to installing the NFR CMS software. Following the documentation[xxvi], we first create the user that will operate the NFR CMS – who we creatively call "nfr". On a core install of Solaris, there is no windowing system, so we make users the old fashioned way:

```
# useradd -d /nfr -s /bin/ksh nfr
# grep nfr /etc/passwd
nfr:x:100:15::/nfr:/bin/ksh
```

We then set the passwd for nfr with passwd nfr <enter> and add a group entry as well with groupadd -g 100 nfr. Next we vi /etc/group and add nfr to the end of the newly created nfr group specification, so the last line reads: "nfr::100:nfr". The user "nfr" is now a valid system user with login shell and group id, so we chown -R nfr /nfr and chgroup –R, and ln -/nfr /opt/nfr because NFR's default install directory for Solaris 8 is /opt/nfr, and we sure don't want to break anything. There are times we need a shell for this user, or we would disable it completely. Next mount the cdrom manually (none of the magic volume manager stuff is running on this box) with the command mount -F hsfs -o ro /dev/dsk/c0t3d0s2 /cdrom. Your cdrom device may be addressed by another name, so some experimentation may be in order. Once mounted, cd into the /cdrom/thirdparty/Solaris directory. This is where the special libraries needed by NFR are stored[xxvii]. Copy off the file gcc_stdc_sun.tar.Z to a directory on your server and uncompress/untar it, then look for two library files:

```
#cd /backup
#cp /cdrom/thirdparty/Solaris/gcc_stdc_sun.tar.Z /backup/
#uncompress  gcc_stdc_sun.tar.Z
#tar xvf ./gcc_stdc_sun.tar
#cd gcc_stdc_sun
```

This puts two libraries into /backup, move them to /usr/lib

```
#cp libgcc_s.so.1 /usr/local/lib
#cp libstdc++.so.3.0.0 /usr/local/lib
#cd /usr/lib
```

---

http://www.insecure.org/nmap/

[25] FreeBSD is a free Unix-like operating system based on Berkley 4.4 Lite. It runs on a variety of hardware, and is an interesting alternative to Linux. Learn more at URL:http://www.freebsd.org

```
#ln -s /usr/local/lib/libstdc++.so.3.0.0 libstdc++.so.3
#ln -s /usr/local/lib/libgcc_s.so.1 libgcc_s.so.1
```

This creates symbolic links from /usr/lib to /usr/local/lib for the two necessary libraries. Next in /nfr we add a line to the NFR users' *.profile*[26] file a line containing: LD_LIBRARY_PATH=/usr/local/lib. Next switch user to nfr and install the application:

```
#su – nfr
$/cdrom/central/install
Where do you want to install the NFR software?
[ default: /opt/nfr ]
/nfr
There are some files there already:
lost+found: No such file or directory
Install anyway? [no]
yes
Extracting files...
39090 blocks
Enter your NFR central management station licence key
xxxxxx-xxxxx
Enter the name of this central NFR, or press Return to use
the name "mongo"
Using name mongo
Enter the IP address of this central NFR
10.1.1.113
Here is your current disk usage:
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/dsk/c0t0d0s6    29905967   22887 29584021     1%    /nfr
How much disk space will you make available to NFR for data storage?
Specify the number of 1K blocks or use M or G for megabytes or
gigabytes, respectively.  [ default: 800M ]
29000000
Creating the "nfr" user
Enter a password for the nfr administrative user:
Enter it again:
------------------------------------------------------------------------
If you want NFR to start automatically at boot time,
become root and run these commands
cp /nfr/nfrstart.sh /etc/init.d/nfr
ln -s ../init.d/nfr /etc/rc0.d/K10nfr
ln -s ../init.d/nfr /etc/rc1.d/K10nfr
```

[26] The .profile file executes when the traditional shell starts up, and is used to set up a users environment.

```
ln -s ../init.d/nfr /etc/rc2.d/S90nfr
ln -s ../init.d/nfr /etc/rcS.d/K10nfr

Installation of NFR software is now complete.
The software is installed in /nfr
You can now add remote nodes with bin/add_remote.
```

Installation is completed, you can unmount the cdrom. Now su – root and run the commands suggested, if you want NFR to start at boot time:

```
#su – root
#cp /nfr/nfrstart.sh /etc/init.d/nfr
#ln -s ../init.d/nfr /etc/rc0.d/K10nfr
#ln -s ../init.d/nfr /etc/rc1.d/K10nfr
#ln -s ../init.d/nfr /etc/rc2.d/S90nfr
#ln -s ../init.d/nfr /etc/rcS.d/K10nfr
#exit
```

Now become the NFR user again, add one or more remotes using bin/add_remote from /nfr. It is OK to run this script anytime, just stop nfr first with:

```
$ bin/stop_nfr
$ bin/add_remote
Enter the name of the remote NFR: spookie.somewhere.com
Enter the IP address or DNS name of the remote: 10.1.11.210
Please choose one of the following:
   slr - Secure Log Repository
   nid - Network Intrusion Detection Appliance
Enter the type of remote [nid]: nid
NID remote selected
Please choose one of the following:
   nid-100
   nid-200
   nid-300
Enter the type of NID [nid-100]: nid-300
Enter the encryption passphrase for spookie.somewhere.com contacting this
host
777TheBadHacker!WillNeverGuessThis888
NID was selected
Installing rcfg/spookie.somewhere.com.
```

We noticed a bug that leads us to conclude that NFR does not fully support 64 bit Solaris 8. After restart we got this console message:

```
# file ld.so.1: bin/samd: fatal: libCrun.so.1: open failed: No such file or directory
```

A search for this library finds it in some directories that the recommended patches archive created when unpacked:

```
# find / -name "libCrun.so.1" -print
/backup/8_Recommended/108434-13/SUNWlibC/reloc/usr/lib/libCrun.so.1
/backup/8_Recommended/108434-13/SUNWlibC/reloc/usr/lib/libCstd.so.1
/backup/8_Recommended/10843513/SUNWlibCx/reloc/usr/lib/sparcv9/libCrun.so.1
/backup/8_Recommended/10843513/SUNWlibCx/reloc/usr/lib/sparcv9/libCstd.so.1
```

We took the ../usr/lib 32-bit version, not the one in ../usr/lib/sparvcv9/. Do NOT try to use the 64 bit version with NFR, it did not run!! When you have the most recent one, copy it to /usr/lib and use the file utility to make sure it is 32 bit, not 64bit:

```
$ file /usr/lib/libCrun.so.1
/usr/lib/libCstd.so.1: ELF 32-bit MSB dynamic lib SPARC Version 1, dynamically
linked, not stripped
```

With this file in place in /usr/lib, NFR happily starts up whenever you reboot the box. A quick check with netstat shows the NFR processes listening:

Configuration Validation and Ongoing Maintenance

I take the view that, in today's hostile environment, configuration validation and routine maintenance have converged. You really can't sit back and ignore things until something melts down, you have to be proactive, the objective being to "nip trouble in the bud" so to speak. So, with that philosophy, we being by examining what the system is doing at this moment, then move on to setting up some backup and integrity assessment tools and scripting them to run and report back to us what the system is doing on a daily basis. Let's start with our old friend, netstat:

```
# netstat -an

UDP: IPv4
   Local Address        Remote Address     State
-------------------- -------------------- -------
      *.520                                Idle
      *.*                                  Unbound
      *.*                                  Unbound
TCP: IPv4
   Local Address        Remote Address    Swind Send-Q Rwind Recv-Q  State
-------------------- -------------------- ----- ------ ----- ------ -------
      *.*                  *.*             0      0 24576      0 IDLE
```

```
*.1968          *.*          0      0 24576      0 LISTEN
*.2010          *.*          0      0 24576      0 LISTEN
*.*             *.*          0      0 24576      0 IDLE
```

This is our expected result, three ports active. Good. Confirming with external nmap scans (tcp and udp) , we see:

```
# nmap -sT 10.1.1.113
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on  (10.1.1.113):
(The 1600 ports scanned but not shown below are in state: closed)
Port       State       Service
1968/tcp   open
2010/tcp   open        search
Nmap run completed -- 1 IP address (1 host up) scanned in 124 seconds
```

```
#nmap –sU 10.1.1.113
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on  (10.1.1.113):
(The 1467 ports scanned but not shown below are in state: closed)
Port       State       Service
520/udp    open        route
Nmap run completed -- 1 IP address (1 host up) scanned in 245 seconds
```

Note that the scans run were short, that is, not all of the possible 65535 ports were scanned on this run, we merely confirmed externally what netstat already reported. To scan all ports, use the –p1-65535 command switch to nmap. In order to validate that traffic to and from the NFR/CMS is being encrypted, a 4 port hub is introduced, and the V100, the iBook, a WindowsXP machine and a FreeBSD workstation are attached. A sniffer was started on the FreeBSD workstation, using tcpdump. On the WindowsXP system, we initiated a NFR Administrative Interface (AI) session to the CMS software running on the V100. Tcpdump recorded the traffic as we logged into the NFR CMS and issued a few commands. We then analyzed the results. Host 10.1.1.113 is the NFR/CMS, while the host 10.1.1.111 is the Administrative Interface client. In these traces, service ndtp maps in /etc/services to tcp port 2010 (FreeBSD 4.7). Packet traces are pretty long and involved, but the point we want to make is that the communications are encrypted. Using the hex dump feature of tcpdump, and setting the *snaplength* value to 2048, we turn an eye on communications between the systems. I will bold those items of interest in the trace below, so they stand out a bit. Notice that the NFR AI client is using a custom http 1.0 client to send encrypted command and control messages to the NFR/CMS :

#tcpdump -X -i fxp0 -s 2048 host 10.1.1.113

```
0x0020          5018 4470 4d03 0000 5573 6572 2d41 6765   P.DpM...User-Age
0x0030          6e74 3a20 4e46 5220 436f 6e73 6f6c 652f   nt:.NFR.Console/
0x0040          3120 2832 2e31 2920 2857 696e 646f 7773   1.(2.1).(Windows
0x0050          290d 0a41 6363 6570 743a 202a 2f2a 0d0a   )..Accept:.*/*..
0x0060          436f 6e74 656e 742d 7479 7065 3a20 6170   Content-type:.ap
0x0070          706c 6963 6174 696f 6e2f 782d 7777 772d   plication/x-www-
0x0080          666f 726d 2d75 726c 656e 636f 6465 640d   form-urlencoded.
0x0090          0a43 6f6e 7465 6e74 2d6c 656e 6774 683a   .Content-length:
[snip]
0x0020          5018 60f4 019e 0000 4854 5450 2f31 2e30   P.`.....HTTP/1.0
0x0030          2032 3030 204f 4b0d 0a54 696d 653a 2031   .200.OK..Time:.1
0x0040          3036 3030 3331 3932 320d 0a0d 0a          060031922...
[snip] From another trace:
0x0020          5018 4470 443c 0000 504f 5354 202f 2048   P.DpD<..POST./.H
0x0030          5454 502f 312e 300d 0a                    TTP/1.0..
16:36:41.475369 10.1.1.113.ndtp > 10.1.1.111.1258: . ack 18 win 24820 (DF)
0x0000          4500 0028 a5f1 4000 4006 7143 901e 01f0   E..(..@.@.qC....
0x0010          901e 016f 07da 04ea 4052 96de 57e1 d3d4   ...o....@R..W...
0x0020          5010 60f4 1b9a 0000 0000 0000 0000        P.`...........
16:36:41.475850 10.1.1.111.1258 > 10.1.1.113.ndtp: P 18:424(406) ack 1 win 17520
(DF)
0x0000          4500 01be 9120 4000 8006 447e 901e 016f   E.....@...D~...o
0x0010          901e 01f0 04ea 07da 57e1 d3d4 4052 96de   ........W...@R..
0x0020          5018 4470 f330 0000 5573 6572 2d41 6765   P.Dp.0..User-Age
0x0030          6e74 3a20 4e46 5220 436f 6e73 6f6c 652f   nt:.NFR.Console/
0x0040          3120 2832 2e31 2920 2857 696e 646f 7773   1.(2.1).(Windows
0x0050          290d 0a43 6f6e 7465 6e74 2d74 7970 653a   )..Content-type:
0x0060          206e 6672 5f64 6573 0d0a 436f 6e74 656e   .nfr_des..Conten
```

Notice the Content-type = nfr_des, followed by some gibberish. It appears to be announcing to the world that some form of DES encryption is in use here, although, perhaps, it is a diversionary lie. Whatever it is, it brings a measure of peace of mind when you can see your data is, in fact, not moving in clear text. It is no fun to sniff a so-called encrypted connection, only to learn that, in fact, the transmissions were not being encrypted, for one reason or another. As we alluded to before, it is good to be aware of the limitations of your systems. Notice in the trace below, we did manage to recover a valid username from the data stream – nfr. This is the default userid for the NFR subsystem on the Sun side, which is always useful in a system-cracking context. The next time I install NFR, you can bet that default user will change. So from a intelligence perspective, we now know this is an NFR system, so if we know of any special vulnerabilities, we could start to hammer on them specifically with this information. Try some buffer overflows on the listeners, etc. Or better yet, attack the AI station, since it is windows based, and then use it to reconfigure the NFR system. They could tighten

up this command and control method a bit more - it gives away a lot that it doesn't need to. Still, it is encrypted, and seeing that is reassuring.

```
0x0070   742d 436f 6469 6e67 3a20 6170 706c 6963      t-Coding:.applic
0x0080   6174 696f 6e2f 782d 6e66 725f 7374 7265      ation/x-nfr_stre
0x0090   616d 0d0a 5573 6572 6e61 6d65 3a20 6e66      am..Username:.nf
0x00a0   720d 0a54 6963 6b65 743a 2025 3939 3e25      r..Ticket:.%99>%
0x00b0   3043 2538 373c 2544 4225 4136 2542 3325      0C%87<%DB%A6%B3%
0x00c0   4639 2546 3325 4442 2531 4476 2538 4525      F9%F3%DB%1Dv%8E%
0x00d0   3130 2539 3325 3038 2546 3730 2546 4178      10%93%08%F70%FAx
0x00e0   715e 2532 3654 6749 2538 3625 4634 2543      q^%26TgI%86%F4%C
0x00f0  4125 3143 5425 4243 2538 4245 5733 2542      A%1CT%BC%8BEW3%B
0x0100   3525 3944 2531 410d 0a43 6f6e 7465 6e74      5%9D%1A..Content
0x0110   2d6c 656e 6774 683a 2031 3839 0d0a 0d0a      -length:.189....
0x0120   67fa 81cb d7cd a53a 8fbb 3d4c 7f4b 4c5f    g......:..=L.KL_
0x0130   7019 ed21 c5de eb9e 4c73 0a1d f9a7 dcb3      p..!....Ls......
0x0140   4be4 00ff 46f5 9e8a c87c 1cd5 26c7 3780     K...F....|..&.7.
0x0150   aedb fb20 c377 63f4 04b8 eb20 b5f2 6f75     .....wc.......ou
```

One more examination, this time with nmap, gives us a accurate guess of the OS type and version, so we have gathered significant intelligence about the machine using common tools and methods:

```
# nmap -O 10.1.1.113
Starting nmap V. 3.10ALPHA4 ( www.insecure.org/nmap/ )
Interesting ports on 10.1.1.113:
(The 1604 ports scanned but not shown below are in state: closed)
Port       State      Service
1968/tcp   open       ndtp
2010/tcp   open       search
Remote operating system guess: Solaris 8 early access beta through actual release
Uptime 0.391 days (since Mon Aug 4 07:20:47 2003)
```

This is why we harden systems, to make it just a bit harder, to give us a little extra hope. Now that the system software, patching, hardening and application installation and validation are completed, we will move tripwire over from our intermediate host, initialize our database, and move a copy of it back to the intermediate host for storage against future runs as a baseline snapshot. Since we are not installing compilers on this machine, and do not have an up to date version of Tripwire for Solaris handy for this exercise, we are using an older version of tripwire, Academic Source Release, Version 1.2.1(patch 2), the one that shipped with the YASSP distribution, for illustration purposes.

There are differences in the newer and older versions, but the end result is the same –

a method to detect unauthorized file system modifications. You get a ASR Tripwire binary with your YASSP distribution – in Solaris package format no less. Extract the YASSP archive, and look for the Solaris package prftripw_sparc. You can install it as part of an overall YASSP install, or if you are not installing YASSP, as is our case here, just install the ASR Tripwire with:

```
pkgadd –d ./prftripw_sparc [27]
```

We keep a stripped down tarball of ASR Tripwire in the toolbox, with everything it needs to install and run from /secure/tripwire. Using this archive, we install as follows:

```
#mkdir /secure
#cd /secure
# tar xvfp asrtw.tar
x ./tripwire, 0 bytes, 0 tape blocks
x ./tripwire/databases, 0 bytes, 0 tape blocks
x ./tripwire/siggen, 86260 bytes, 169 tape blocks
x ./tripwire/tripwire, 158932 bytes, 311 tape blocks
x ./tripwire/tw.config.Solaris, 16330 bytes, 32 tape blocks
```

Lets depart now from installing things, and visit the way we mount our filesystems. The file /etc/vfstab sets up how filesystems are mounted, and controls various properties for them - like read only, logging, etc. We are particularly interested in three – ro (read only), logging (adds robustness, and speeds up system boot up after a crash, at slight performance cost), and nosuid (prevents execution of files that run as users other than themselves). You can discern these files by doing an ls –al on them and looking for the tell-tale s bit. When executed, it runs as the user who owns it, in this case root:

```
# ls -la /bin/login
-r-s--x--x   1 root     bin        29292 Jan  5  2000 /bin/login
```

The /etc/vfstab looks like this:

```
#device         device          mount           FS      fsck    mount   mount
#to mount       to fsck         point           type    pass    at boot options
```

[27] When using pkgadd with Solaris packages that are set up in a directory format, you use pkgadd –d . *packagename* to add them, as is the case on the Solaris cdrom. When using pkgadd on packages from third parties that are arranged as one file, you will use pkgadd –d ./*packagename* . If unsure of the format, use file *packagename* – if it reports "directory" use the dot, if it reports English text or ASCII text, use ./ to select the package. Another way to tell is to head the first line, if it says something like # PaCkAgE DaTaStReAm, it is a file and the ./ is the form to use on it.

```
#
#/dev/dsk/c1d0s2 /dev/rdsk/c1d0s2          /usr        ufs        1     yes      -
fd                      -                  /dev/fd     fd      -     no             -
/proc                   -                  /proc       proc          -     no       -
/dev/dsk/c0t0d0s3       -                  -           swap          -     no       -
/dev/dsk/c0t0d0s0       /dev/rdsk/c0t0d0s0 /           ufs     1     no     logging
/dev/dsk/c0t0d0s1       /dev/rdsk/c0t0d0s1 /usr        ufs     1     no     logging,ro
/dev/dsk/c0t0d0s5       /dev/rdsk/c0t0d0s5 /var        ufs     1     no     logging,nosuid
/dev/dsk/c0t2d0s0       /dev/rdsk/c0t2d0s0 /backup     ufs     2     yes    logging,nosuid
/dev/dsk/c0t0d0s6       /dev/rdsk/c0t0d0s6 /nfr        ufs     2     yes    logging,nosuid
/dev/dsk/c0t0d0s4       /dev/rdsk/c0t0d0s4 /opt        ufs     2     yes    logging,nosuid
swap                    -                  /tmp        tmpfs   -     yes size=512m
```

The nosuid option will not be applied to /, because it will also mean “no devices”. Who knows, if /dev could be a separate partition, it just might work – fodder for cold winter nights in the lab. I suspect you could do this, on the theory that root is root and will run whatever it likes, SUID bit or not, which is how is handles /usr/bin/ps later in the project. Root runs ps, it logs the fact to syslog, and executes anyhow. What we can protect, we do, and watch the rest closely. Be aware that *if you botch an entry in the /etc/vfstab, your system may not boot*, and certainly will not be happy. If you break it, boot from distribution media, mount the root partition, and clean up the mess best you can with a text editor, or recreate it line by line with echo “fstab entry 1, 2, etc” >> /etc/vfstab if you have trouble with getting an editor up.

The next step is to check the tw.config file to make sure it is suitable for your system. These config files can be bedeviling, once you get one that works for a given OS, I am sure you will keep it around as a template for subsequent installs. For help with ASR Tripwire, Version 1.2 (The YASSP Version) configuration file creation, see the man page for tw.config, or download a nice manual for Version 1.3 online at Tripwire’s website[xxviii]. Be patient, it takes a while to get these things set up. Once you have the configuration file set up, you create the initial database by moving to your Tripwire install directory and typing:

```
./tripwire -initialise -c tw.config
```

This creates a database of all files and their computed signatures, then stores it in./databases/tw.db_*hostname* This file contains the Keys to the Kingdom, so to speak, move both it and your configuration file, tw.config, off to secure media at once. Store these as you would system passwords. In this case, the file will be moved off the system, and burned onto cdrom. When our nightly cron script runs, it will invoke the executables off read only media. While I could just mount the file systems read only, it is much harder to alter a read only cdrom than to remount a file system generally. We backup the files with tar again, move them to the iBook, extract them and burn them onto a cdrom, which then goes right back into the Sun. We also add some other files to

the cdrom, files that are typically tampered with by attackers, such as ls, find, ifconfig, netstat, etc. Create your Tripwire tarball like this:

```
# tar cvf /tmp/tw.cdrom.tar /secure/tripwire/*
a /secure/tripwire/databases/ 0K
a /secure/tripwire/databases/tw.db_mongo 755K
a /secure/tripwire/siggen 85K
a /secure/tripwire/tripwire 156K
a /secure/tripwire/tw.config 16K
a /secure/tripwire/tw.config.Dist 13K
```

I also put the fantastic tool lsof[28] on this cd as well. While not a Solaris binary, this tool will show every open file descriptor on the machine that the kernel can see (some rootkits will mask their processes, of course). The more things you watch, the less likely it is you will miss something as massive as a rootkitting. Speaking of which, it doesn't hurt to have a copy of all your critical binaries on a handy cdrom, so you might as well copy them over and burn them while you are at it. The contents of /bin and /usr/bin are not that oppressive, and will be appreciated in a pinche. Another great tool for this cdrom is Chkrootkit. Chkrootkit will look for many rootkits by analyzing the system for their "signatures", or special characteristics. Get a copy at the chkrootkit.org website[29]. Might as well run it via crontab everynight. Download and build on Solaris 8 takes about 5 minutes total, and it runs very fast. To build it, you simply:

```
#gzip –d ./chkrootkit.tar.gz
#tar xvf ./chkrootkit.tar
#cd chkrootkit-(version)
#make sense [xxix] (Murilo and Steding-Jessen)
```

If all goes well, in a couple of minutes you wind up with several small executables. Chkrootkit is the one that you invoke to start the processing. If your system has log files in odd places, you might alter some of the paths in the code, however, it supports a lot of systems well out of the box. Solaris is one. You invoke chkrootkit by typing:

```
#./chkrootkit –h
Usage: ./chkrootkit [options] [test ...]
Options:
    -h          show this help and exit
    -V          show version information and exit
```

[28] lsof – a utility that lists files opened by processes on your system. Get your copy free from Vic Able's site at Purdue. Keep in mind their ftp server requires a reverse DNS lookup to work for your IP address to open a connection up.
URL: http://www-rcd.cc.purdue.edu/~abe/
[29] The Chkrootkit.org website is at URL:http://www.chkrootkit.org (Aug 1, 2003)

```
-l              show available tests and exit
-d              debug
-q              quiet mode
-x              expert mode
-r dir          use dir as the root directory
-p dir1:dir2:dirN path for the external commands used by chkrootkit
```

A run of chkrootkit produces a decent audit of your system too, checking for known signs of tampering that are consistent with “rootkitting”. Invoke it like this:

```
# ./chkrootkit
ROOTDIR is `/'
Checking `amd'... not found
Checking `basename'... not infected
Checking `biff'... not found
Checking `chfn'... not infected

[snip] checks tons of binaries

Searching for sniffer's logs, it may take a while... nothing found
Searching for HiDrootkit's default dir... nothing found
Searching for t0rn's default files and dirs... nothing found

[snip] checks for sniffer logs and rootkits by the score! Scary….

Checking `sniffer'... Checking `wted'... nothing deleted in /var/adm/wtmpx
unable to open wtmp-file /var/adm/wtmp
Checking `w55808'... not infected
Checking `scalper'... not infected
Checking `slapper'... not infected
Checking `z2'... unable to open wtmp-file wtmp

[Here we learn that I forgot to create a wtmp file!]

# touch /var/adm/wtmp; touch /var/adm/wtmpx
# chmod 600 /var/adm/wt*
# ls -la /var/adm/w*
-rw-------   1 root     root         0 Aug  7 13:33 /var/adm/wtmp
-rw-------   1 adm      adm     211668 Aug  7 13:33 /var/adm/wtmpx
```

It just goes to show that complex systems are really easy to mis-configure, despite the attention this machine has received, a basic configuration step was skipped. So, we create a wtmp and wtmpx file now, then return our focus to tripwire. I think we will go ahead and put chkrootkit on a crontab, too. When we run our tripwire script, it will scan

the system, compare it against the state preserved in the database, then do an lsof the system for all open files and such, count and record them, and send the report to the Administrator for review. I also tail the /var/adm/message log file and append that to our report.
The end result is an email you can refer back to anytime you have lingering doubts about the system state.

```
#!/bin/sh
# do tripwire scan of system
# BOURNE SHELL, Solaris 8 2/02
# open source please share give away and use commercially as you wilt
TODAY=`date +%m%d%y`

/bin/rm /tmp/tw.report*
# clean previous leftovers off tmp

/usr/bin/echo "To: gmarkham@nowhere.com " > /tmp/tw.report.$TODAY
/usr/bin/echo "Subject: Tripwire Report for NFR/CMS" >> /tmp/tw.report.$TODAY
/usr/bin/echo "******************************" >>/tmp/tw.report.$TODAY
# setup email headers, to change recipients, edit To:
# line, separate multiple addresses with comma and space
# the **** line separates headers from text, tripwire may insert characters that
# appear as bogus headers  which break email so keep a line of something here
# that doesn't end in a :. Tripwire will flag files added: deleted: changed: etc. in report

 /usr/sbin/mount -F hsfs -o ro /dev/dsk/c0t3d0s2 /cdrom
/cdrom/tripwire/tripwire -c /cdrom/tripwire/tw.con >> /tmp/tw.report.$TODAY

# run tripwire off cdrom . Your cdrom may have another name, and directory
# paths may vary according to how your ASR Tripwire was configured

/usr/bin/echo "******* /var/adm/messages file on NFR/CMS *********"  >>
/tmp/tw.report.$TODAY
/usr/bin/tail -100 /var/adm/messages >> /tmp/tw.report.$TODAY
# put some formatting in, add last 100 logfile entries to message

/usr/bin/echo "*** lsof listing line count for  NFR/CMS ***" >> \
 /tmp/tw.report.$TODAY

/cdrom/lsof  | tee /tmp/filelist | wc -l >> /tmp/tw.report.$TODAY
/usr/bin/echo "*** lsof detail ****" >> /tmp/tw.report.$TODAY
/usr/bin/cat /tmp/filelist >> /tmp/tw.report.$TODAY
# do lsof of system, get total # of open items + provide detail listing in case
# significant change seen. Line count gives thumbnail indication of volume of open files
```

```
# a deviation from norms is a clue to look deeper.

/usr/bin/echo "******* Run Chkootkit ******" >> /tmp/tw.report.$TODAY
/cdrom/chkrootkit >> /tmp/tw.report.$TODAY

/usr/lib/sendmail  -t </tmp/tw.report.$TODAY
#feed sendmail a recipient list with -t flag, and report in one file
/usr/sbin/umount /cdrom
# mail report and umount the cdrom but do not eject.
```

We enable this script to run daily at 23:58 by adding a line to the root crontab:

```
58 23  *       *       *       /usr/local/bin/runtw.sh
```

The man page for crontab, section 5, gives the format for the crontab, reading left to right, as follows: minute (0-59), hour (0-23), day of the month (1-31), month of the year (1-12), and day of the week (0-6 with 0=Sunday). So based on this, each evening at 23 58 we run this script. While we are on crontabs, lets install a backup script – all the security on earth is worthless if a jet engine falls through the roof and smashes our pretty toy. This machines backup plan is simple, save the system to a second internal disk, then move that data off manually once a week to the iBook using scp, as shown previously. Once on the iBook, it gets sucked into the enterprise backup system, and ultimately, goes offsite. We use a simple script to implement the ufsdump command like this:

```
#!/bin/sh
today=`date +%m%d%y`
/usr/bin/echo "To: gmarkham@somewhere.com" >              /backup/backup.log
/usr/bin/echo "Subject: NFR/CMS Backup Log" >>           /backup/backup.log
/usr/bin/echo "**************************************" >>  /backup/backup.log
#setup email headers and divide between them and data

/usr/bin/df -k  >> /backup/backup.log
# get disk space, report, dump following filesystems
#  cmd ( level=0, f=save to file) file=filesystem-name.date
#-----------------------------------------------------------------------
/usr/sbin/ufsdump 0f /backup/root.$today / && /usr/bin/echo " / dumped... " >> \
/backup/backup.log

/usr/sbin/ufsdump 0f /backup/usr.$today /usr && /usr/bin/echo "/usr dumped..." >> \
/backup/backup.log

/usr/sbin/ufsdump 0f /backup/var.$today  /var && /usr/bin/echo "/var dumped..." >> \
/backup/backup.log
```

```
/usr/sbin/ufsdump 0f /backup/opt.$today  /opt && /usr/bin/echo " /opt dumped. " >> \
/backup/backup.log
/usr/bin/su – nfr –c “bin/nfr/stop_nfr”
#switch user and shutdown NFR system for good backup
/usr/sbin/ufsdump 0f /backup/nfr.$today /nfr  && /usr/bin/echo "/nfr dumped ... " >> \
/backup/backup.log
/usr/bin/su – nfr –c “bin/start_nfr”
#now restart NFR

#dump file system utilization info and then dump system to /backup with ufsdump

/usr/bin/compress -f /backup/*.$today >>          /backup/backup.log
#squish a little bit to save space and be compatible with everyone

/usr/lib/sendmail -t < /backup/backup.log
#send report to admin
```

To receive your email, you will have to configure sendmail appropriately for your installation. We only run it from the command line here, rather than as a daemon, for security reasons. Generally, make sure you have a mailhost entry set in /etc/hosts, and a basic sendmail.cf file configured according to your local customs. Checking your site’s MX record(s) will reveal the appropriate mailhost(s). If everything is in its place, we get a nice email each time the backup runs:

```
To: gmarkham@somewhere.com
Subject: NFR/CMS Backup Log
****************************************
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/dsk/c0t0d0s0     1985487   52812  1873111    3%    /
/dev/dsk/c0t0d0s1     2507143   75491  2381510    4%    /usr
/proc                       0       0        0    0%    /proc
fd                          0       0        0    0%    /dev/fd
mnttab                      0       0        0    0%    /etc/mnttab
/dev/dsk/c0t0d0s5     1985487   47921  1878002    3%    /var
swap                  1857912      16  1857896    1%    /var/run
/dev/dsk/c0t2d0s0    38476820  662393 37429659    2%    /backup
swap                   524288       8   524280    1%    /tmp
/dev/dsk/c0t0d0s6    29905967   53325 29553583    1%    /nfr
/dev/dsk/c0t0d0s4      962367    3260   901365    1%    /opt
 / dumped...
/var dumped...
 /opt dumped...
/nfr dumped ...
```

This installs in the root crontab too. Use crontrab –e to open it up for editing. The man pages for cron and crontab will help get you straight on the files format. Keep in mind its best to put complete paths to anything in your scripts.

58 22 * * * /usr/local/bin/backup.sh

Each backup run brings us a report of file system space, and the fact that our backup has run – very reassuring. Running off cdrom, we get our file scans run without worrying about the integrity of our database or configuration files. Our other scripts rely on Tripwire to watch over their well-being. If we wish to update the Tripwire database or configuration file, we work with the binaries on the cdrom, and just copy the new database back to the iBook, burn it all onto a new cd, and carry on. While we are at it, our backup script gives us an lsof of the file system as well – the report is tedious, so we use wc –l to do a line count of open files. This count gives us a thumbnail sketch of systems status. For example, if the lsof line count has been consistently running say 375, and one morning it pops up to 500, something is going on – a clue to have a closer look at things. This cuts down on the background noise. Some of my co-workers demand that you put whatever you want in the subject line of your mail to them, so they do not have to open each item. That could apply to these kinds of mailings too – for you fancy script writers out there.

Your report from tripwire will list files like those below by status since last database update. Tripwire can run in interactive update mode too, so you can correct your database as it drifts over time. Be sure to read up on it and use it a while to get comfortable with it.

files added:
added: lrwxrwxrwx root 3 Aug 6 08:22:57 2003 /usr/local/bin/slogin
[snip]
files deleted:
deleted: -rwxr-xr-x root 49304 May 24 19:01:47 2001 /kernel/drv/sparcv9/ses
[snip]
and files changed:
changed: drwxr-x--- root 512 Aug 6 23:03:22 2003 /var/spool/mqueue
[snip]
It then reports on attributes in detail:
Attr Observed (what it is) Expected (what it should be)
==========
==========
/kernel/drv ←name of file object reported on
st_mtime: Tue Aug 5 12:47:50 2003 Mon Jul 28 14:42:31 2003 ←attribute detail
st_ctime: Tue Aug 5 12:47:50 2003 Mon Jul 28 14:42:31 2003
[snip]

At the end of tripwires report, we get the log file tailings:
******* /var/adm/messages file on NFR/CMS *********
Aug 6 17:05:39 mongo syslogd: going down on signal 15
Aug 6 17:05:46 mongo genunix: [ID 672855 kern.notice] syncing file systems...
[snip]
Then we get the lsof total line count:
*** lsof listing line count for NFR/CMS ***
429 ← total number of open files on system, marked increase = activity, maybe a Bad Thing™ - too few = something stopped running. Investigate deviations.

*** lsof detail **** ←next report is detailed listing of all open file descriptors COMMAND
PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
sched 0 root cwd VDIR 136,0 1024 2 /
init 1 root cwd VDIR 136,0 1024 2 /
init 1 root txt VREG 136,0 550000 270354 /sbin/init
[snip] Lastly comes the output from chkrootkit, much truncated for brevity:
ROOTDIR is `/'
Checking `amd'... not found
Checking `basename'... not infected
Checking `biff'... not found
[snip]
Searching for Anonoying rootkit default files and dirs... nothing found
Searching for ZK rootkit default files and dirs... nothing found
Searching for anomalies in shell history files... nothing found
[snip]
Checking `scalper'... not infected
Checking `slapper'... not infected
Checking `z2'...
nothing deleted

While no one likes reading these kinds of emails, in the event something happens, the clues you will glean can mean the difference between having an answer for the boss or giving them that pitiful, "Gee, I dunno what happened…" look. Jobs are too scarce anymore for that slackness to fly. To take an old saw, and paraphrase it a bit "Nothing fails like failure". Failure with no clues as to why is, well, downright disrespectful. We owe it to both ourselves, and our employers, to have answers. Documentation like this helps when the place is on fire. So, in that spirit, we continue our validation of our system configuration. We perform a full, 65535 port tcp, and a 10000 portudp portscan with nmap[30] against the system, using a similar script. For general scripting help, you can refer to Steve Parker's "Bourne Shell Programming Tutorial[xxx]. The following script,

[30] For a copy of nmap, see the insecure.org website: URL http://www.insecure.org/nmap/

which is intended to runs on another host, requires nmap and executes via a crontab entry. Feel free to use, modify and share as needed:

```
#!/bin/sh
# do nmap scan of a host, mail it to sysadmin
# BOURNE SHELL, FreeBSD v4.7
# Open source please share give away and use commercially as you wilt

TODAY=`date +%m%d%y`
/bin/rm /tmp/nmap.report.*
# clean previous leftovers off tmp

echo "To: gmarkham@someplace.com" >              /tmp/nmap.report.$TODAY
echo "Subject: Portscan Report for NFR/CMS" >>   /tmp/nmap.report.$TODAY
# echo is shell built-in here, how much search path can that have?
#  this sets up the recipient and subject of email
#  to change recipients, edit To: line, separate multiple addresses with
#  comma and space. Subject works the same way

/usr/local/bin/nmap -sT -p1-65535 -o /tmp/nmap.report.tmp 10.1.1.113
/bin/cat /tmp/nmap.report.tmp >> /tmp/nmap.report.$TODAY

#  use the -o option to write a nmap log file in human readable format
# then append to the days report. Can generate xml too

/usr/local/bin/nmap -sU -p1-10000 -o  /tmp/nmap.report.tmp 10.1.1.113
/bin/cat /tmp/nmap.report.tmp >>  /tmp/nmap.report.$TODAY

/usr/sbin/sendmail -t </tmp/nmap.report.$TODAY
# send the report to someone who cares. Use -t sendmail flag to feed
# recipient name to sendmail in the report body. Your sendmail might
# well live someplace else, /usr/lib is popular…
```

So, according to a preset schedule, a report id delivered to a set of eyeballs that is short and conscise:

```
To: gmarkham@someplace.com
Subject: Tripwire Report for NFR/CMS
# Log of: /usr/local/bin/nmap -sT -p1-65535 -o /tmp/nmap.report.tmp 10.1.1.113
Interesting ports on  (10.1.1.113):
Port    State       Protocol  Service
1968    open        tcp       unknown
2010    open        tcp       search
```

```
# Log of: /usr/local/bin/nmap -sU -p1-10000 -o /tmp/nmap.report.tmp 10.1.1.113
No ports open for host  (10.1.1.113)
Port    State       Protocol  Service
520     open        udp       route
```

This scan reports as expected. If anyone opens up a traditional port on the machine, we will see it reflected in our email. While there are tools now that do not reply on open ports for communications, this will ferret out more blatant invasions. One note – the UDP scans of some Operating Systems with nmap take a LONG time, owing to behaviors configured into the TCP stacks of various systems regarding how they respond to multiple port unreachable messages. The manual page for nmap (Version 2.07, FreeBSD v 4.7) warns clearly:

> "Unfortunately UDP scanning is sometimes painfully slow since most hosts impliment a suggestion in RFC 1812 (section 4.3.2.8) of limiting the ICMP error message rate"

Our Solaris 8, as tweaked, sure as heck isn't going to tolerate a fast UDP scan. This is a Very Good Thing™. In practice some nmap UDP scans against some hosts will time out before completion, so you may have to tweak what you scan for a bit, depending on what system you scan from. I cut this one off at 10000, and let it run through the night. We will also try to invoke an suid file, off a filesystem configured not to permit it. Starting with telnet, we see we can't connect:

```
[gmarkhams-Computer:~] root# telnet  10.1.1.113
Trying 10.1.1.113...
telnet: connect to address 10.1.1.113: Connection refused
telnet: Unable to connect to remote host
```

Cool, how about ssh? We did install it, but disabled sshd startup as you may recall. From the iBook we try to initiate ssh session to the NFR/CMS:

```
[gmarkhams-Computer:~] root# ssh -l root 10.1.1.113
ssh: connect to address 10.1.1.113 port 22: Connection refused
```

Good. At least nothing we have run has reactivated sshd. From our port scans we knew this already. Lets play bang the keyboard. We login as root, giving bad passwords. After each bad try, the system redisplays warning banners, alerts the console, sends an alert to syslog, and after 3 events, it gives us a time out before we get to try again. This is as expected:

```
mongo console login: root
Password:
Login incorrect
```

Aug 7 15:23:25 mongo last message repeated 1 time
Aug 7 15:23:30 mongo login: REPEATED LOGIN FAILURES ON /dev/console, root

Syslog output in /var/adm/messages:

Aug 7 15:25:36 mongo login: [ID 821011 auth.notice] Login failure on /dev/console, root
Aug 7 15:25:54 mongo last message repeated 2 times
Aug 7 15:25:59 mongo login: [ID 561941 auth.crit] REPEATED LOGIN FAILURES ON /dev/console, root

Next we login and check syslog for appropriate messages, we discover something nice, the attempt to execute a suid file has been logged:

Aug 7 15:31:56 mongo login: [ID 644210 auth.notice] ROOT LOGIN /dev/console
Aug 7 15:31:56 mongo genunix: [ID 809163 kern.info] NOTICE: quota, uid 0: setuid execution not allowed, dev=8800000001
Aug 7 15:31:56 mongo genunix: [ID 809163 kern.info] NOTICE: mail, uid 0: setuid execution not allowed, dev=8800000001

So, we now get syslog event tracking for every suid execution, sweet. Can you say “got root detector”? Lets try some file system monkey business of our own. The system /etc/vfstab reads as follows:

```
#device        device         mount                    FS      fsck    mount   mount
#to mount      to fsck        point                    type    pass    at boot options
#/dev/dsk/c1d0s2       /dev/rdsk/c1d0s2       /usr       ufs     1       yes     -
fd                     -       /dev/fd                 fd      -       no      -
/proc                  -       /proc                   proc    -       no      -
/dev/dsk/c0t0d0s3      -                               -       swap    -       no      -
/dev/dsk/c0t0d0s0      /dev/rdsk/c0t0d0s0      /       ufs     1       no      logging
/dev/dsk/c0t0d0s1      /dev/rdsk/c0t0d0s1      /usr    ufs     1       no      logging,ro,nosuid
/dev/dsk/c0t0d0s5      /dev/rdsk/c0t0d0s5      /var    ufs     1       no      logging,nosuid
/dev/dsk/c0t2d0s0      /dev/rdsk/c0t2d0s0      /backup ufs     2       yes     logging,nosuid
/dev/dsk/c0t0d0s6      /dev/rdsk/c0t0d0s6      /nfr    ufs     2       yes     logging,nosuid
/dev/dsk/c0t0d0s4      /dev/rdsk/c0t0d0s4      /opt    ufs     2       yes     logging,nosuid
swap - /tmp tmpfs - yes size=512m
```

Testing the nosuid function, we try and run a few setuid commands that reside in /usr, while doing a tail –f /var/adm/messages on our remote syslog server:

loghost>#tail –f /var/adm/messages
Aug 8 08:13:53 mongo.somewhere.com genunix: [ID 809163 kern.info] NOTICE: ps, uid 0: setuid execution not allowed, dev=8800000001
Aug 8 08:13:56 mongo.somewhere.com last message repeated 3 timesAug 8 08:14:15

mongo.somewhere.com genunix: [ID 809163 kern.info] NOTICE: lsof, uid 0: setuid execution not allowed, dev=8800000001

Our ps ran, even though the file system was mounted nosuid. Why? Because root can execute suid programs because, well, it is root. The file system routinges still logs the attempts ro run them however, and then the system runs them anyhow. Any other user would not have it so lucky when running ps. Watch nfr try it:

```
$ id
uid=100(nfr) gid=15(users)
$ ps
ksh: ps: cannot execute
```

This greatly enhances security, because ps is a fantastic tool for watching command line activity on a system. Another thing to keep in mind about remote logging is that syslogd on some systems, FreeBSD being one, requires you to specify not only the remote IP:/netmaskmask of the server for which you with to log events, but the service port number as well, i.e. your syslogd command line to allow remote logging datagrams to come in on a nonstandard port or 32595 would be:

```
#syslogd –a 10.1.1.113/32:32959
```

The –d debug flag can be added to syslogd to let it run on your terminal and report each action taken – very helpful when you just can't get your messages to log. If you are real paranoid, you can tee the output off to a log file and watch for attackers trying to smash your syslogd too. File system writing and modification of timestamp info is also impossible on /usr now:

```
# touch newfile
touch: newfile cannot create
# touch zcat
touch: cannot change times on zcat
```

The file system is resistant to change, that's good. Our validation appears to have been a success, at least as far as we take it in this paper. The real test will be how long this machine endures on the network to which it is attached. As a final, and ongoing task, we must regularly review any vulnerability reports for each of the software products we are using on this machine. If not already subscribed to an appropriate mailing list for tracking software bugs and exploits, we get that way fast. In the fast paced world we live in today, it is easy to miss vulnerability reports from individual vendors. Two excellent, general-purpose lists you can count on for up to date information are:

Listname	Current Subscription Page
BUGTRAQ	URL: http://www.securityfocus.com/subscribe?listname=1

Vulnerability specific list, highly technical and authoritative

SANS NewsBites URL: http://portal.sans.org/
Vulnerabilities with a mix of general security related news.

This concludes the instant exercise in configuring the Sun Fire V100 as a hardened IDS component. I hope you find the information contained herein useful in your work.

George Markham, GCIH, August, 2003

TABLE 1 STOCK SOLARIS PACAKGES
(110 packages)

tools	OPENssh	Open SSH
application	SMClsof	lsof
application	SMCmd5	md5
system	SMEvplr	SME platform links
system	SMEvplu	SME usr/platform links
system	SUNWadmr	System & Network Administration Root
system	SUNWatfsr	AutoFS, (Root)
system	SUNWatfsu	AutoFS, (Usr)
system	SUNWauda	Audio Applications
system	SUNWaudd	Audio Drivers
system	SUNWauddx	Audio Drivers (64-bit)
system	SUNWbzip	The bzip compression utility
system	SUNWcamos	Central America OS Support
system	SUNWcamow	Central America OW Support
system	SUNWcar	Core Architecture, (Root)
system	SUNWcarx	Core Architecture, (Root) (64-bit)
system	SUNWced	Sun GigaSwift Ethernet Adapter (32-bit Driver)
system	SUNWcedx	Sun GigaSwift Ethernet Adapter (64-bit Driver)
system	SUNWcg6	GX (cg6) Device Driver
system	SUNWcg6x	GX (cg6) Device Driver (64-bit)
system	SUNWcsd	Core Solaris Devices
system	SUNWcsl	Core Solaris, (Shared Libs)
system	SUNWcslx	Core Solaris Libraries (64-bit)
system	SUNWcsr	Core Solaris, (Root)
system	SUNWcsu	Core Solaris, (Usr)
system	SUNWcsxu	Core Solaris (Usr) (64-bit)
system	SUNWdfb	Dumb Frame Buffer Device Drivers
system	SUNWdmfex	Sun Davicom 10/100Mb Ethernet Driver (64-bit)
system	SUNWdtcor	Solaris Desktop /usr/dt filesystem anchor
system	SUNWensqr	Ensoniq ES1370/1371/1373 Audio Device Driver (Root) (32-bit)

system	SUNWensqx	Ensoniq ES1370/1371/1373 Audio Device Driver (Root) (64-bit)
system	SUNWeridx	Sun RIO 10/100 Mb Ethernet Drivers (64-bit)
system	SUNWesis	Latin Spanish install software localization
system	SUNWesu	Extended System Utilities
system	SUNWfcip	Sun FCIP IP/ARP over FibreChannel Device Driver
system	SUNWfcipx	Sun FCIP IP/ARP over FibreChannel Device Driver (64 bit)
system	SUNWfcp	Sun FCP SCSI Device Driver
system	SUNWfcpx	Sun FCP SCSI Device Driver (64-bit)
system	SUNWfctl	Sun Fibre Channel Transport layer
system	SUNWfctlx	Sun Fibre Channel Transport layer (64-bit)
system	SUNWftpr	FTP Server, (Root)
system	SUNWftpu	FTP Server, (Usr)
system	SUNWged	Sun Gigabit Ethernet Adapter Driver
system	SUNWglmr	rasctrl environment monitoring driver for i2c, (Root) (32-bit)
system	SUNWglmx	rasctrl environment monitoring driver for i2c (Root) (64-bit)
system	SUNWhmd	SunSwift SBus Adapter Drivers
system	SUNWhmdx	SunSwift SBus Adapter Drivers (64-bit)
system	SUNWi1cs	X11 ISO8859-1 Codeset Support
system	SUNWi2cr	Device drivers for I2C devices, (Root, 32-bit)
system	SUNWi2cx	Device drivers for I2C devices, (Root, 64-bit)
system	SUNWidecr	IDE device drivers
system	SUNWidecx	IDE device drivers (Root) (64bit)
system	SUNWider	IDE Device Driver, (Root)
system	SUNWigsr	IGS CyberPro2010 Device Driver (ROOT)
application	SUNWigsu	IGS CyberPro2010 DDX (OW) Driver and Utilities
system	SUNWigsx	IGS CyberPro2010 64-bit Device Driver (ROOT)
Application	SUNWjass	Solaris Security Toolkit 4.0.0
system	SUNWkey	Keyboard configuration tables
system	SUNWkmp2r	PS/2 Keyboard and Mouse Device Drivers, (Root, 32-bit)
system	SUNWkmp2x	PS/2 Keyboard and Mouse Device Drivers, (Root, 64-bit)
system	SUNWkvm	Core Architecture, (Kvm)
system	SUNWkvmx	Core Architecture (Kvm) (64-bit)
system	SUNWlibms	Sun WorkShop Bundled shared libm
system	SUNWlmsx	Sun WorkShop Bundled 64-bit shared libm
system	SUNWloc	System Localization
system	SUNWlocx	System Localization (64-bit)
system	SUNWluxdx	Sun Enterprise Network Array sf Device Driver (64-bit)
system	SUNWluxop	Sun Enterprise Network Array firmware and utilities
system	SUNWluxox	Sun Enterprise Network Array libraries (64-bit)
system	SUNWm64	M64 Graphics System Software/Device Driver
system	SUNWm64x	M64 Graphics System Software/Device Driver (64-bit)
system	SUNWmdi	Sun Multipath I/O Drivers
system	SUNWmdix	Sun Multipath I/O Drivers (64-bit)

system	SUNWnisr	Network Information System, (Root)
system	SUNWnisu	Network Information System, (Usr)
system	SUNWpd	PCI Drivers
system	SUNWpdx	PCI Drivers (64-bit)
system	SUNWpl5u	Perl 5.005_03
system	SUNWqfed	Sun Quad FastEthernet Adapter Driver
system	SUNWqfedx	Sun Quad FastEthernet Adapter Driver (64-bit)
system	SUNWrmodu	Realmode Modules, (Usr)
system	SUNWses	SCSI Enclosure Services Device Driver
system	SUNWsesx	SCSI Enclosure Services Device Driver (64-bit)
system	SUNWsior	SuperIO 307 (plug-n-play) device drivers, (Root)
system	SUNWsiox	SuperIO 307 (plug-n-play) device drivers, (Root) (64-bit)
system	SUNWsndmr	Sendmail root
system	SUNWsndmu	Sendmail user
system	SUNWsolnm	Solaris Naming Enabler
system	SUNWssad	SPARCstorage Array Drivers
system	SUNWssadx	SPARCstorage Array Drivers (64-bit)
system	SUNWswmt	Install and Patch Utilities
system	SUNWuaud	USB Audio Drivers
system	SUNWuaudx	USB Audio Drivers (64-bit)
system	SUNWudf	Universal Disk Format 1.50, (Usr)
system	SUNWudfr	Universal Disk Format 1.50
system	SUNWudfrx	Universal Disk Format 1.50 (64-bit)
system	SUNWusb	USB Device Drivers
system	SUNWusbx	USB Device Drivers (64-bit)
system	SUNWwsr2	Solaris Product Registry & Web Start runtime support
system	SUNWxcu4	XCU4 Utilities
system	SUNWxwdv	X Windows System Window Drivers
system	SUNWxwdvx	X Windows System Window Drivers (64-bit)
system	SUNWxwkey	X Windows software, PC keytables
system	SUNWxwmod	OpenWindows kernel modules
system	SUNWxwmox	X Window System kernel modules (64-bit)
system	SUNWzlib	The Zip compression library
system	SUNWzlibx	The Info-Zip compression library (64-bit)

TABLE 2 FINAL PACKAGE LISTING
(57 packages total)

tools	OPENssh	Open SSH
application	SMClsof	lsof
application	SMCmd5	md5
system	SMEvplr	SME platform links
system	SMEvplu	SME usr/platform links
system	SUNWadmr	System & Network Administration Root
system	SUNWbzip	The bzip compression utility
system	SUNWcar	Core Architecture, (Root)
system	SUNWcarx	Core Architecture, (Root) (64-bit)
system	SUNWcsd	Core Solaris Devices
system	SUNWcsl	Core Solaris, (Shared Libs)
system	SUNWcslx	Core Solaris Libraries (64-bit)
system	SUNWcsr	Core Solaris, (Root)
system	SUNWcsu	Core Solaris, (Usr)
system	SUNWcsxu	Core Solaris (Usr) (64-bit)
system	SUNWdmfex	Sun Davicom 10/100Mb Ethernet Driver (64-bit)
system	SUNWensqr	Ensoniq ES1370/1371/1373 Audio Device Driver (Root) (32-bit)
system	SUNWensqx	Ensoniq ES1370/1371/1373 Audio Device Driver (Root) (64-bit)
system	SUNWesu	Extended System Utilities
system	SUNWftpr	FTP Server, (Root)
system	SUNWftpu	FTP Server, (Usr)
system	SUNWged	Sun Gigabit Ethernet Adapter Driver
system	SUNWglmr	rasctrl environment monitoring driver for i2c, (Root) (32-bit)
system	SUNWglmx	rasctrl environment monitoring driver for i2c (Root) (64-bit)
system	SUNWhmd	SunSwift SBus Adapter Drivers
system	SUNWhmdx	SunSwift SBus Adapter Drivers (64-bit)
system	SUNWi2cr	Device drivers for I2C devices, (Root, 32-bit)
system	SUNWi2cx	Device drivers for I2C devices, (Root, 64-bit)
system	SUNWidecr	IDE device drivers
system	SUNWidecx	IDE device drivers (Root) (64bit)
system	SUNWider	IDE Device Driver, (Root)
Application	SUNWjass	Solaris Security Toolkit 4.0.0
system	SUNWkey	Keyboard configuration tables
system	SUNWkvm	Core Architecture, (Kvm)
system	SUNWkvmx	Core Architecture (Kvm) (64-bit)
system	SUNWlibms	Sun WorkShop Bundled shared libm
system	SUNWlmsx	Sun WorkShop Bundled 64-bit shared libm
system	SUNWmdi	Sun Multipath I/O Drivers

system	SUNWmdix	Sun Multipath I/O Drivers (64-bit)
system	SUNWpd	PCI Drivers
system	SUNWpdx	PCI Drivers (64-bit)
system	SUNWpl5u	Perl 5.005_03
system	SUNWrmodu	Realmode Modules, (Usr)
system	SUNWsior	SuperIO 307 (plug-n-play) device drivers, (Root)
system	SUNWsiox	SuperIO 307 (plug-n-play) device drivers, (Root) (64-bit)
system	SUNWsndmr	Sendmail root
system	SUNWsndmu	Sendmail user
system	SUNWsolnm	Solaris Naming Enabler
system	SUNWswmt	Install and Patch Utilities
system	SUNWuaud	USB Audio Drivers
system	SUNWuaudx	USB Audio Drivers (64-bit)
system	SUNWudf	Universal Disk Format 1.50, (Usr)
system	SUNWudfr	Universal Disk Format 1.50
system	SUNWudfrx	Universal Disk Format 1.50 (64-bit)
system	SUNWusb	USB Device Drivers
system	SUNWusbx	USB Device Drivers (64-bit)
system	SUNWzlib	The Zip compression library
system	SUNWzlibx	The Info-Zip compression library (64-bit)

REFERENCES

[i] Sun Microsystems, Inc. "The Sun Fire V100 User Guide". Dec. 2001. URL: http://www.sun.com/products-n-solutions/hardware/docs/pdf/816-2756-10.pdf (July 22,2003)

[ii] Sun Microsystems, Inc. "Miscellaneous - Netra System Configuration Card". undated. URL: http://sunsolve.sun.com/handbook_pub/Devices/Miscellaneous/MISC_SysConf_Netra.html (July 22, 2003).

[iii] NFR Security, Inc.. "FS-Products". undated. URL: http://www.nfr.com/solutions/FS-Product.pdf (July 22, 2003)

[iv] NFR Security, Inc. "NFR Central Management Server, Version 3.0 User's Guide". 10-15-2002. URL: https://support.nfr.com/nid-v3/ddb/cms/docs/NFR_CMS_UG.pdf (Aug 1, 2003)

[v] NFR Security, Inc. "(nfr)(security) solutions: network intrusion detection system : technical data". undated. URL: http://www.nfr.com/solutions/technical.php (July 23,2003)

[vi] 21CFRpart11.com. "Links to FDA Documentation". undated. URL: http://www.21cfrpart11.com/pages/fda_docs/ (August 3, 2003)

[vii] Osborne, Brian. (ThinkGeek.com). July 1, 2003. "US Interior Dept. ordered to pull plug on its 'Net connections". URL: http://www.geek.com/news/geeknews/2003Jun/gee20030701020631.htm (July 23,2003)

[viii] Ashley, David. "Practical ufsdump & ufsrestore explanations for Solaris 8 ". 5-23-2002. URL: http://www.datasync.com/~daniel/how_to_ufsdump_ufsrestore.htm (August 10, 2003)

[ix] Hp, Inc. "HP TopTools for Hubs and Switches User Guide". 2000. URL: http://h20000.www2.hp.com/bc/docs/support/SupportManual/bpt01033/bpt01033.pdf (July 24, 2003): P 181-200.

[x] Foundry Networks, Inc. "Foundry Switch and Router Command Line Interface". undated. URL: http://www.foundrynet.com/services/documentation/srcli/MAC-port-security_cmds.html#128771 (July 24, 2003)

[xi] HP, Inc. "HP TopTools for Hubs and Switches User Guide,". 2000. URL: http://h20000.www2.hp.com/bc/docs/support/SupportManual/bpt01033/bpt01033.pdf (July 24, 2003): P 181-200

[xii] Cisco, Inc. "Cisco Connection Documentation". July 14, 2003. URL: http://www.cisco.com/univercd/home/home.htm (July 24,2003).

[xiii] Sun Microsystems, Inc. "Platform Notes: The dmfe Fast Ethernet Device Driver(part No. 816-2128-11)" December 2001, Revision A, Sun Microsystems, Inc., P 5-18.

[xiv] Stokely, Celeste. "Unix Serial Port Resources: Sun Serial Port." 2002. URL: http://www.stokely.com/unix.serial.port.resources/A-B-Ycablepinout.html (July 23,2003)

[xv] Ahmed, Osama, Gagnon,Charles, et al. (Sunmanagers mailing-list). 02-28-2002. "Jumpstart: Could not update system for automatic rebooting", URL: http://www.sunmanagers.org/pipermail/summaries/2002-February/002472.html (July 22, 2003)

[xvi] Unknown Author at littleblackdog.com. "Hotwo: Making a Crossover Cable". 09-22-2002. URL: http://www.littlewhitedog.com/reviews_other_00009.asp (July 25, 2003)

[xvii] Noordergraaf, Alex. "Solaris™ Operating Environment Minimization for Security: A Simple, Reproducible and Secure Application Installation Methodology – Updated for Solaris 8 Operating Environment". 11-2002. URL: http://www.sun.com/blueprints/1100/minimize-updt1.pdf (July 25, 2003)

[xviii] The OpenSSH Project. "OpenSSH Manual Pages". 2002. URL: http://www.openssh.org/manual.html (Aug 1, 2003)

[xix] Sun Microsystems, Inc.. "Solaris 8 2/02 Operating Environment Package List". Feb. 2002. URL: http://www.sun.com/bigadmin/content/packagelist/s8u7PkgList/p2.html#SPARCPACKAGES-1 (August 1,2003)

[xx] Noordergraaf, Alex. "Solaris Operating Environment Minimization for Security: A Simple, Reproducible and Secure Application Installation Methodology". NOV. 2000. URL: http://www.sun.com/blueprints/1100/minimize-updt1.pdf (July 24, 2003)

[xxi] Nemeth, Mike. "Hats for System Administrators; more SA tools". undated. URL: http://www.geocities.com/SiliconValley/4841/hats.html (Aug 5, 2003).

[xxii] Boran, Seán. "Hardening Solaris with Jass". Jan. 27, 2003. URL: http://www.boran.com/security/sp/Solaris_hardening4.html (July 23, 2003)

[xxiii] Sun Microsystems, Inc. "Solaris Security Toolkit (JASS) FAQ". July 25, 2003. URL: http://www.sun.com/software/security/jass/faq.html (July 25, 2003)

[xxiv] The State of Arkansas. "Standard Statement – Warning Banner(Draft)". May, 2003. URL: http://www.techarch.state.ar.us/domains/security/standards/warning_banner_standard_statement.pdf (August 2, 2003)

[xxv] Farmer, Dan. "Titan". undated. URL: http://www.fish.com/titan/TITAN_documentation.html (July 25, 2003)

[xxvi] NFR Securty, Inc. "NFR Central Management Server Version 3.0. User's Guide",12-18-2002.
URL: https://support.nfr.com/nid-v3/ddb/cms/docs/NFR_CMS_UG.pdf P1.1 – 1.3.

[xxvii] NFR Securty, Inc. "NFR Central Management Server Version 3.0. User's Guide",12-180-2002. URL:
https://support.nfr.com/nid-v3/ddb/cms/docs/NFR_CMS_UG.pdf P1.2 – 1.3.

[xxviii] Tripwire, Inc. " Tripwire™ Academic Source Release 1.3.1 for Unix User Manual" April 30, 1999. URL: http://www.tripwire.com/files/downloads/asr/Tripwire-1.30-docs.pdf (August 4, 2003)

[xxix] Murilo, Nelson and Steding-Jessen, Klaus "Chkrootkit-Version 0.41 README". 6-20-2003. URL:
http://www.chkrootkit.org/README (August 3, 2003)
[xxx] Parker,Steve. "Bourne Shell Programming Tutorial". 2002. URL: http://steve-parker.org/sh/sh.shtml (July 20, 2003).

Last Updated: September 23rd, 2017

Upcoming Training

SANS 2018	**Orlando, FL**	**Apr 03, 2018 - Apr 10, 2018**	**Live Event**
SANS OnDemand	**Online**	**Anytime**	**Self Paced**
SANS SelfStudy	**Books & MP3s Only**	**Anytime**	**Self Paced**