Todd Garrison – Practical Assignment for GSNA

Version 1.0

Research in Audit, Measurement Practice, and Control

System being audited:

I chose to audit a Silicon Graphics Inc. Indy for the practical. I do not have extensive experience with SGI systems and feel that the practical assignment should be as much a learning exercise as that of a demonstration of knowledge gained from the SANS course I attended.

The specifics of the system are as follows:

- IRIX 6.5.4
- SGI Indy (180 MHz R5000 processor)

It should be noted that the operating system being used is not the most current. This is due to a practical limitation of cost. This system was bought at a popular on-line auction site for under $300, and Silicon Graphics Inc. charges $700 for the latest version of their operating system. Since this system is being used at my home and not office, there is little benefit to purchasing a newer version of the operating system. This will prove to be an additional challenge to the audit because there are additional vulnerabilities and risks associated with this older version of the IRIX operating system.

Current State of Practice

Silicon Graphics maintains a portion of their website dedicated to security (http://www.sgi.com/support/security/). The website has pointers to several FAQ (frequently asked questions) documents as well as references to books on general security measures. It is the author's opinion that the outlined practices and tools mentioned on the SGI website are outdated in comparison to other Unix operating systems. The tools mentioned are generally several years old and the reference material mentioned is not necessarily current. This is not meant to imply, however, that these practices are inefficient, only that they are older. These are 'tried and true' methods of securing Unix systems, but information security is a moving target and the practices in use two or three years ago sometimes are irrelevant or less useful than newer tools and methodologies.

There are several checklists on the website and I will take those into account as I create my audit checklist.

By default the IRIX operating system is designed for convenience and subsequently it is very insecure when initially installed. Because SGI systems have very powerful graphics abilities they are commonly used in government installations and universities for complex design and modeling purposes. Because of this fact, there are many organizations that have created security checklists. One such list is maintained by Texas A&M University. This is the primary list I will be using for creating my list. Their list can be viewed online at: http://www-viz.tamu.edu/~sgi-faq/faq/html-1/security.html.

Why are current methods and techniques in need of improvement?

While the current state of practice is not necessarily flawed in any way, it does not meet my specific needs. The system is not shared, is physically secure and has only one primary user.

Obviously my security needs are different from those of a university or government agency. The primary focus of the security lists I have found on the Internet is either a multi-user environment, where the SGI system is used as a server, or a shared environment where many users may be at the console (such as a laboratory environment.) My environment is different in that the network itself is generally more hostile and there is only one primary user of the system. The system will only be used as a client in the network environment for accessing the web and reading e-mail. The network on which the system exists is hostile because it is host to honeypots and other systems for performing information security related research.

The strategy used in this audit checklist is that anything that is not necessary should be disabled. This helps in many cases where there may be a security hole discovered in a specific service, but since it is not available to an attacker to begin with it would not matter.

I have focused heavily on network security. This has different implications for hardening a host than say an ISP that provides shell accounts on a system. For example, I found **no** texts explaining how to use 'ipfilterd' to enable packet filtering on the host itself. This is important because my general philosophy about host security is that a system administrator should still be able to sleep at night even if the firewall is pulled out of the picture. Much of the work that has been done is just 'best practice' for Unix – tripwire, TCP Wrappers, turning off of services and managing System V init scripts. But I have not found any text that shows what is and is not dangerous to run. Because of this I have developed a paranoid configuration that trusts nothing to the network, that way even if they **are** secure, the services in question are still not available.

Objective Measurements

Since this system is to only be used as a workstation that is not shared between users and provides no network services the checklist is simplified. Two major areas of interest are the focus of this audit. The first is preventing attack and the second is the detection of compromise or abnormal circumstances.

Allowed Network Traffic:

The system will offer no response to unnecessary network traffic inbound from outside the local network. These rules are in addition to network firewall rules (meaning that packet filtering should be performed on the machine itself, so that if the firewall or local network is compromised the system is still protected.)

Allow inbound SSH

Outbound traffic to anywhere is limited to:

ICMP echo	FTP	Streaming media (rtsp)
SSH	http	DNS
Syslog	SMTP	S-Imap (SSL)

The purpose behind packet filtering at the host itself is to strengthen the system against network based attacks. Vulnerable services cannot be accessed if they are not available. This assists in mitigating the risks of denial-of-service (availability) and compromise (integrity) by eliminating the possibility of attacking host-based services from both the local and remote networks.

Other checks to be performed:

- OpenSSH is installed
- No root logins are allowed, except from console
- Logins without passwords are denied
- Syslog to central location
- Static ARP entries for syslog server, default route and SMTP server.

Ensure password-protected logins for the following usernames (or ensure the account is locked):

sysadm	cmwlogin	diag
uucp	sys	adm
lp	nuucp	auditor
dbadmin	sgiweb	rfindd
EZsetup	demos	OutOfBox
guest	4Dgifts	

- Use of shadow passwords.
- Sendmail is not listening on the network (only checks queue every 15 minutes)

Ensure the following are disabled in /etc/inetd.conf:

ftp	telnet	shell
login	exec	finger

bootp	tftp	ntalk
tcpmux	echo	discard
chargen	daytime	time
daytime	mountd/1,3	sgi_mountd/1
rstatd/1-3	walld/1	rusersd/1
rquotad/1	sprayd/1	sgi_snoopd/1
sgi_pcsd/1	sgi_pod/1	ttdbserverd/1
tcpmux/sgi_scanner	tcpmux/sgi_printer	

Use chkconfig to ensure the following services are turned off during system boot:

array	autoconfig_ipaddress	autofs
fontserver	gated	named
lp	mrouted	nostickytmp
nds	nfs	rfindd
proclaim_relayagent	proclaim_server	routed
proxymngr	rarpd	snmpd
rsvpd	rwhod	videod
timed	ypmaster	yp
automount	ypserv	isdnd

Other Objective Measurements:

- NMAP will show no open ports from external network (UDP or TCP).
- Nessus will show no vulnerabilities when machine is scanned.
- TCP Wrappers are installed and deny connection attempts from outside the local network.
- TCP Wrappers are configured for all enabled services in the '/etc/inetd.conf' file.
- The chkconfig setting noiconlogin should be set to 'ON'.

Subjective Measurements

- All accounts use reasonably strong passwords (at least six characters total length, at least two alpha characters and at least one special character or number.)
- All available security-related patches are performed.
- System is protected by firewall from any inbound connections.
- System is monitored with a network intrusion detection system to detect both inbound and outbound attacks.
- The system must be using the most recent version web-browser.
- Tripwire is installed and checks all **critical** files on the system (the example configuration for IRIX is sufficient.)

Methods and Commands to Determine Compliance

Allowed Network Traffic

The easiest method to limit network activity on IRIX is to use ‘ipfilterd’ (which should not be confused with ipf [BSD] or ipfilter/netfilter [Linux].)

Enabled at startup?

First, the daemon must be enabled. IRIX uses a SystemV init script to start ‘ipfilterd’ on boot. The command ‘chkconfig | grep ipfilterd’ should return something similar to ‘ipfilterd on’ when executed. There are three possible states for ‘ipfilterd’. The first is that it is enabled (which is what is desired.) The second is that it is disabled (off). And the third is that it has not been configured for use with ‘chkconfig’ and will return nothing when ‘chkconfig’ is queried (which is not desirable.)

In the case that ‘chkconfig’ does not know about ‘ipfilterd’ or that it is disabled it can be enabled with the command: ‘chkconfig -f ipfilterd on’.

Packet filtering rules:

The file ‘/etc/ipfilterd.conf’ specifies what network connections are allowed or denied. It should be noted that ‘ipfilterd’ is not a “stateful” packet filter. Therefore you must use certain TCP flags to emulate this behavior. While this is not optimal, it is better than nothing! Here is an example ‘ipfilterd.conf’ file, which would meet our audit guidelines. It is outside the scope of this document to explain all of the filtering options of ipfilterd, but if the reader is interested the MAN page for the program contains pointers on where to find more information.

The IP address in the following example should be changed to the address of the machine being audited.

```
#
# Loopback . . .
accept -i lo0
#
#########
#
#  outbound only
#
#########
#
# DNS is necessary outbound only
accept -i ec0 ip.src 10.0.245.1 and udp.dport 53
accept -i ec0 ip.dst 10.0.245.1 and udp.sport 53 and udp.dport > 1023
accept -i ec0 ip.src 10.0.245.1 and tcp.dport 53
accept -i ec0 ip.dst 10.0.245.1 and tcp.sport 53 and tcp.flags != SYN and tcp.dport > 1023
#
# Web traffic is all right:
accept -i ec0 ip.src 10.0.245.1 and tcp.dport 80
accept -i ec0 ip.dst 10.0.245.1 and tcp.sport 80 and tcp.flags != SYN and tcp.dport > 1023
accept -i ec0 ip.src 10.0.245.1 and tcp.dport 443
accept -i ec0 ip.dst 10.0.245.1 and tcp.sport 443 and tcp.flags != SYN and tcp.dport > 1023
accept -i ec0 ip.src 10.0.245.1 and tcp.dport 8080
accept -i ec0 ip.dst 10.0.245.1 and tcp.sport 8080 and tcp.flags != SYN and tcp.dport > 1023
accept -i ec0 ip.src 10.0.245.1 and tcp.dport 8000
accept -i ec0 ip.dst 10.0.245.1 and tcp.sport 8000 and tcp.flags != SYN and tcp.dport > 1023
#
# FTP may be needed ***THIS IS PASSIVE FTP ONLY***
accept -i ec0 ip.src 10.0.245.1 and tcp.dport 21
accept -i ec0 ip.dst 10.0.245.1 and tcp.sport 21 and tcp.flags != SYN and tcp.dport > 1023
accept -i ec0 ip.src 10.0.245.1 and tcp.dport 20
accept -i ec0 ip.dst 10.0.245.1 and tcp.sport 20 and tcp.flags != SYN and tcp.dport > 1023
#
# Allow streaming media
accept -i ec0 ip.src 10.0.245.1 and tcp.dport 7070
accept -i ec0 ip.dst 10.0.245.1 and tcp.sport 7070 and tcp.flags != SYN and tcp.dport > 1023
accept -i ec0 ip.src 10.0.245.1 and tcp.dport 554
accept -i ec0 ip.dst 10.0.245.1 and tcp.sport 554 and tcp.flags != SYN and tcp.dport > 1023
#
# Mail outbound and s-imap
accept -i ec0 ip.src 10.0.245.1 and tcp.dport 25
accept -i ec0 ip.dst 10.0.245.1 and tcp.sport 25 and tcp.flags != SYN and tcp.dport > 1023
accept -i ec0 ip.src 10.0.245.1 and tcp.dport 993
accept -i ec0 ip.dst 10.0.245.1 and tcp.sport 993 and tcp.flags != SYN and tcp.dport > 1023
#
# allow ssh OUT
accept -i ec0 ip.src 10.0.245.1 and tcp.dport 22
accept -i ec0 ip.dst 10.0.245.1 and tcp.sport 22 and tcp.flags != SYN and tcp.dport > 1023
#
# allow ping
accept -i ec0 ip.src 10.0.245.1 and icmp.type == ECHO
accept -i ec0 ip.dst 10.0.245.1 and icmp.type == ECHOREPLY
#
#########
#
#  inbound rules
#
#########
#
accept -i ec0 ip.dst 10.0.245.1 and tcp.dport 22
#
#########
# All which is not specifically allowed is denied.
#
reject -i ec0 tcp
reject -i ec0 udp
reject -i ec0 icmp
```

Is OpenSSH Installed?

The easiest method of checking is to issue the command 'ssh –v' which will display the version of OpenSSH being used. If OpenSSH has been installed from a pre-compiled package the binary may not be in your path. The most likely location is '/usr/freeware/bin/ssh'. If OpenSSH is not already installed it is available from SGI's freeware collection located at http://freeware.sgi.com/ or at the OpenSSH web site at http://www.openssh.com/.

Restrict 'root' Logins to the Console.

The line 'CONSOLE=/dev/console' should not be commented out in the file '/etc/default/login'. Additionally, SSH will allow this to be over-ridden. The file '/etc/sshd_config' should have the line 'PermitRootLogin without-password' instead of 'PermitRootLogin yes'. The reason behind this is detailed later in the 'tripwire' section.

Users Must Have a Password to Login

The line ‘PASSREQ=YES’ as well as the line ‘MANDPASS=YES’ should exist in ‘/etc/default/login’. By default these are both set to NO.

System Logging on a Central Server

The file ‘/etc/syslogd.conf’ should contain an entry similar to this ‘*.debug   @loghost’ where ‘loghost’ is the hostname of a syslog server.

Static ARP Entries for Syslog Server, Mail Gateway and Default Route

- Determine the IP addresses for the machines in question.
- Examine the output of the command ‘arp –an’ to see if the hosts exist in the output.
- If the text ‘permanent’ is printed after each entry, then the system is configured properly.
- Otherwise the system will require the entries to be added.

In the situation that static ARP is not configured follow these steps to enable it.

Edit the file ‘/etc/init.d/network’ and look for the section of the file that matches the following:

```
;;

‘stop’)
```

Immediately before this section, type exactly the following:

```
if $IS_ON static-arp ; then
    /etc/static-arp.sh
fi
```

Then issue the command ‘**chkconfig –f static-arp on**’

And create the file (by editing it) ‘/etc/static-arp.sh’. The contents should look like this:

```
#!/bin/sh

/usr/etc/arp –s 10.0.245.2 0:a:b:c:1:2
```

Which contains an entry for each ARP to IP address mapping you would like at system startup. Finally ensure your new shell-script is executable by issuing the command ‘chmod 0755 /etc/static-arp.sh’.

Check for Logins Without Passwords

The following shell script will produce a list of users without passwords:

```
#!/bin/sh
cat /etc/passwd | awk -F: '{print $1}' | while read login
do passwd -s $login | grep NP | awk '{print $1}'
done
```

For each account without a password, the login should be locked. Issuing the following command can do this:

```
passwd –l <username>
```

Where <username> is the login name of the offending account. Be careful not to lock the ‘root’ account!

Use of Shadow Passwords

Examine the ‘/etc/passwd’ file, if the second field contains an ‘x’ on all accounts then shadowing is enabled. Here is an example:

```
tgarris:x:24720:20:Todd Garrison:/usr/people/tgarris:/bin/csh
```

If shadow passwords are not already enabled, the ‘pwconv’ utility will automatically convert the system over to shadow passwords.

Sendmail is not Listening for Connections

Execute the command ‘ps –ef | grep sendmail’, if the result contains the string ‘-bd’ the daemon is configured to listen for connections from the network, and should be fixed.

```
  root   818    1    0 15:38:07  ?     0:00 /usr/lib/sendmail –bd –q15m
```

Is an example of a sendmail daemon that is configured to listen for incoming mail and should be remedied.

Modify the file ‘/etc/init.d/mail’ so that the strings ‘-bd –q15m’ reads only ‘-q15m’.

Unneeded Services are Disabled in Inetd

The command ‘grep –v \# /etc/inetd.conf | sort’ will give a list of all the services enabled via inetd. The returned list will need to be manually checked against the list of disallowed services.

To remedy an out-of-compliance service, the ‘/etc/inetd.conf’ file should be edited and for each line corresponding to an invalid service should have a pound sign (#) placed at the beginning of the line.

Other Services not Needed at System Boot

There are many superfluous services enabled by default. While some of these may be needed in certain circumstances, for example integration with Novell Directory Services, they are not needed on this network.

To check for whether the listed services are enabled, issue the command ‘chkconfig’ and each service will be printed and listed as either ‘on’ or ‘off’. If the service is not listed, it is implicitly turned off.

For services that should **not** be enabled, but are - the command ‘chkconfig <servicename> off’ will disable them at the next system boot.

Tripwire

The easiest method of checking that tripwire is installed is to attempt a database check. If the check does not work, or the tripwire binary is not found then it should be installed.

To test simply issue the command:

```
/usr/freeware/bin/tripwire
```

If this does not work, then download and install tripwire 1.2 package from http://freeware.sgi.com/, then execute the following commands:

```
mkdir /usr/adm/tcheck
cp /usr/freeware/lib/tripwire/irix-tw.config \
      /usr/adm/tcheck/tw.config
chmod 0700 /usr/adm/tcheck
/usr/freeware/bin/tripwire –initialize
mkdir /usr/adm/tcheck/databases
mv ./databases/tw* /usr/adm/tcheck/databases
```

Test tripwire by executing –

```
/usr/freeware/bin/tripwire
```

For security reasons, you do not want to trust the local copy of the tripwire database. Copy the database to another system via scp where your other tripwire configurations are stored using a unique name. Configure SSH trust from remote system to allow root access without a password (this should be done in a manner that only trusted systems can access the root account without a password, and other systems cannot login as root even with the password. See the configuration option named ‘PermitRootLogin without-password’ for your SSH Daemon.)

On your secure server that will initiate the tripwire database check, configure a cron job for regular execution using ssh and scp that will copy the necessary files to the machine and perform the database check.

For example a shell script similar to this would do the following:

```
scp /var/tw-db/tw-10.0.245.1-db \
      root@10.0.245.1:/usr/adm/tcheck/tw.db_Indy
scp /var/tw-db/irix-tripwire-6.5.4 \
      root@10.0.245.1:/usr/freeware/bin/tripwire
ssh root@10.0.245.1 ‘/usr/freeware/bin/tripwire’ | mail –s \
      ’10.0.245.1  Nightly Tripwire’ root
```

It should be noted that this is a very simplified version of the actual script – there is no error handling in the above example, and it is not dynamic (only works for one host.) Whereas a more optimal solution would have the ability to work with multiple hosts and multiple databases driven by a configuration file. Obviously such an example is outside of the scope of this document and the above example is just that – an example.

NMAP Scan

The utility nmap should be used to scan the host from outside, as well as inside the local network. From outside the local network, no connections should appear and internally only SSH should work. Before scanning the system it should be rebooted because of all the initialization script changes that have been made in the case of failing previous audit items.

The commands:

```
nmap –sS –v –O –P0 10.0.245.1
nmap –sU –v –P0 10.0.245.1
```

Will show sufficient output. If the test fails, re-check the previous audit items regarding inetd, ipfilterd, and startup scripts and if necessary disable anything that has been missed.

Nessus Scan

Installation and configuration of Nessus is outside the scope of this document. It is assumed the auditor has been provided with the proper tools to perform the job at hand. Since nmap has already been run against the host, the portscan options in Nessus should be turned off. Select ‘All Plugins’ and do not attempt to perform a domain name authoritative transfer. The scan should be performed from both outside and from within the local network to ensure all network vulnerabilities have been eliminated.

TCP Wrappers Deny All but Local Access

The file /etc/hosts.allow should contain only the lines:

```
ALL: 10.0.245.0/255.255.255.0
ALL: 127.0.0.1
```

And the file ‘/etc/hosts.deny’ should contain only the line:

```
ALL: ALL
```

Using TCP Wrappers with Remaining INETD Services

If after the check of the ‘inetd.conf’ file there are any services left, they should if possible be configured to use TCP Wrappers. First confirm that TCP Wrappers is installed. The binary should be located at ‘/usr/freeware/bin/tcpd’. If the file does not exist download and install it from **http://freeware.sgi.com/**. Second edit the file ‘/etc/inetd.conf’ which should have entries similar to the following (note that the example does use finger, which should not be enabled according to the instructions in this document – it is used for the purpose of demonstrating the setup of tcpd only.)

```
finger stream tcp nowait root /usr/freeware/bin/tcpd /usr/etc/fingerd -L
```

The default setting in ‘inetd.conf’ looks like this (if this is how it is setup, it should be fixed.)

```
finger stream tcp nowait root /usr/etc/fingerd fingerd -L
```

All Accounts use Strong Passwords

By default IRIX will enforce the minimum requirement for passwords. In order to ensure un-locked system accounts and user accounts that have been set by root each password in the system should be expired. This will force the users of the system to change their password and will ensure they are subject to the rules required for strong passwords.

The following shell script will facilitate this, be sure to run this after you lock any accounts without passwords. *Be sure to warn your users before doing this – it may upset them if done without warning!*

```
#!/bin/sh

cat /etc/passwd | awk –F: ‘{print $1}’ | while read login
do passwd –s $login | grep –v LK | grep –v root \
      | awk ‘{print $1}’ | while read forcechange
      do passwd –f $forcechange
      done
done
```

Note: it is not advisable to cut and paste the above command into a shell. MS-Word changes the single-quote marks to make grammatically correct matching quotes (single quote and a tick).

All available security-related patches are performed.

Using Bugtraq (http://www.securityfocus.com) the following vulnerabilities were found which may affect this version of IRIX:

Bugtraq id: 262 *(IRIX midikeys Root Vulnerability)*
Action required: remove suid bit on midikeys binary

Bugtraq id: 530 *(SGI arrayd.auth Default Configuration Vulnerability)*
Action required: issue the command ‘chkconfig array off’

Bugtraq id: 1031 *(SGI InfoSearch fname Vulnerability)*
Action required: issue the command ‘/bin/chmod 500 /usr/lib/infosearch/bin/infosrch.cgi’

Bugtraq id: 1106 *(IRIX Performance Copilot Vulnerabilities)*
Action required: issue the command ‘chkconfig pmcd off’

Bugtraq id: 1529 *(IRIX lpstat Buffer Overflow Vulnerability)*
Action required: remove suid bit on ‘/usr/bin/lpstat’

Bugtraq id: 1530 *(IRIX inpview Race Condition Vulnerability)*
Action Required: issue the command ‘versions remove InPerson’ to remove the service, it can be checked with the command ‘versions –b InPerson’ which will display if it is installed.

Bugtraq id: 1526 *(IRIX gr_osview Buffer Overflow Vulnerability)*
Action required: remove the suid bit on ‘/usr/sbin/gr_osview’

Bugtraq id: 1528 *(IRIX dmplay Buffer Overflow Vulnerability)*
Action required: remove the suid bit on ‘/usr/sbin/dmplay’

Bugtraq id: 1572 *(IRIX telnetd Environment Variable Format String Vulnerability)*
Action required: install the patch located at ‘ftp://patches.sgi.com/support/free/security/patches/6.5.4/patch4060.tar’

Bugtraq id: 2548 *(Multiple Vendor BSD ftpd glob() Buffer Overflow Vulnerabilities)*
Action required: ensure ftpd in disabled in ‘/etc/inetd.conf’

Bugtraq id: 2656 *(IRIX 'netprint' Arbitrary Shared Library Usage Vulnerability)*
Action required: remove suid bit on ‘/usr/lib/print/netprint’

Others found on http://www.sgi.com/support/security/

BIND Vulnerabilities
Action required: install the patch located at ‘ftp://patches.sgi.com/support/free/security/patches/6.5.4/patch4193.tar’

System is protected by firewall from any inbound connections.

????

System is monitored with a network intrusion detection system to detect both inbound and outbound attacks.
????

Most Recent Web-Browser

Start the Netscape browser by typing 'netscape' on the command line. Once the browser is open, type 'about:' in the address field. This will display the version number. Compare the version number displayed against the latest download (for version 4.x) for IRIX located on the Netscape web site (http://www.netscape.com/). If the version is not current, then download and install the latest version into '/usr/X11/bin' which will overwrite the existing Netscape installation. You may have to remove the file '/usr/lib/X11/app-defaults/Netscape' if the version installed was the copy provided with the original operating system distribution.

Audit Checklist

The next section shows the checklist that an auditor would use when performing the audit. It is brief so that an administrator or the system owner can rapidly assess the results of an audit. Instead of presenting the checklist twice, I present it completed showing the results of the audit once performed.

Application of Audit Techniques to a Real World System

Checklist for IRIX 6.5.4 Workstation Audit

Auditor : Todd Garrison
Date : 6/25/2001
System Name : Indy
System IP : 10.0.245.1

Reference work instruction sheet for specifics about each item audited.

Pass/Fail	Fixed?	Audit Item
Fail	Yes	Is ipfilterd configured properly?
Pass	-	Is OpenSSH installed and configured?
Pass	-	Are root logins restricted to console?
Pass	-	Are users required to have a password to logon to the system?
Pass	-	Syslog messages are sent to a centralized location?
Pass	-	Static ARP entries exist for Syslog, Mail, and Default Router?
Fail	Yes	No accounts without passwords exist?
Pass	-	Are shadow passwords enabled?
Pass	-	Sendmail is configured to only process queue?
Fail	Yes	Unneeded services disabled in /etc/inetd.conf?
Fail	Yes	Unneeded services disabled at system boot?
Pass	-	Tripwire is installed and database check performed remotely?
Pass	-	TCP Wrappers deny all but local network access inbound?
Fail	**No**	TCP Wrappers govern access to all services remaining in '/etc/inetd.conf'?
Pass	-	Users are using strong passwords?
N/A	-	Removed suid bit on midikeys binary?
Pass	-	The 'array' service is disabled at system boot?
N/A	-	Users have no execute access to the file '/usr/lib/infosearch/bin/infosrch.cgi'?
Fail	Yes	The 'pmcd' service is disabled at system boot?
Fail	Yes	Removed suid bit on '/usr/bin/lpstat'?
Fail	Yes	InPerson software uninstalled?
Fail	Yes	Removed the suid bit on '/usr/sbin/gr_osview'?
Fail	Yes	Removed the suid bit on '/usr/sbin/dmplay'?
Fail	Yes	SGI patch 4060 installed?
Fail	Yes	The ftpd entry in '/etc/inetd.conf' is disabled?
Fail	Yes	Removed suid bit on '/usr/lib/print/netprint'?
Fail	Yes	SGI patch 4193 installed?
Fail	Yes	Using most recent web browser?
Pass	-	System is protected by firewall?
Pass	-	System is monitored by NIDS?
Pass	-	Both external and internal nmap scans are consistent with desired network policy?
Pass	-	Internal and external scan by Nessus show no vulnerabilities?

Excerpts From the Audit

Most of the audit was done from a remote machine using SSH. I have included the commands I typed in bold-faced fonts, and items of interest in red bold-faced fonts.

Only a portion of the audit is shown here, this is to ensure the document has a manageable length.

Is ipfilterd configured properly?

```
plato# ssh tgarris@10.0.245.1
tgarris@10.0.245.1's password:
Last login: Mon Jun 25 10:26:57 2001

Indy 1% su -
Password:

Indy 1# cat /etc/ipfilterd.conf

#
# Loopback . . .
accept -i lo0
#
#########
#
#  outbound only
#
#########
#
# DNS is necessary outbound only
accept -i ec0 ip.src 10.0.245.1 and udp.dport 53
accept -i ec0 ip.dst 10.0.245.1 and udp.sport 53 and udp.dport > 1023
accept -i ec0 ip.src 10.0.245.1 and tcp.dport 53
accept -i ec0 ip.dst 10.0.245.1 and tcp.sport 53 and tcp.flags != SYN
and tcp.dport > 1023
#
# Web traffic is alright:
accept -i ec0 ip.src 10.0.245.1 and tcp.dport 80
accept -i ec0 ip.dst 10.0.245.1 and tcp.sport 80 and tcp.flags != SYN
and tcp.dport > 1023
accept -i ec0 ip.src 10.0.245.1 and tcp.dport 443
accept -i ec0 ip.dst 10.0.245.1 and tcp.sport 443 and tcp.flags != SYN
and tcp.dport > 1023
accept -i ec0 ip.src 10.0.245.1 and tcp.dport 8080
accept -i ec0 ip.dst 10.0.245.1 and tcp.sport 8080 and tcp.flags != SYN
and tcp.dport > 1023
accept -i ec0 ip.src 10.0.245.1 and tcp.dport 8000
accept -i ec0 ip.dst 10.0.245.1 and tcp.sport 8000 and tcp.flags != SYN
and tcp.dport > 1023
#
# FTP may be needed ***THIS IS PASSIVE FTP ONLY***
accept -i ec0 ip.src 10.0.245.1 and tcp.dport 21
accept -i ec0 ip.dst 10.0.245.1 and tcp.sport 21 and tcp.flags != SYN
and tcp.dport > 1023
accept -i ec0 ip.src 10.0.245.1 and tcp.dport 20
accept -i ec0 ip.dst 10.0.245.1 and tcp.sport 20 and tcp.flags != SYN
and tcp.dport > 1023
#
# Allow streaming media
accept -i ec0 ip.src 10.0.245.1 and tcp.dport 7070
accept -i ec0 ip.dst 10.0.245.1 and tcp.sport 7070 and tcp.flags != SYN
and tcp.dport > 1023
accept -i ec0 ip.src 10.0.245.1 and tcp.dport 554
accept -i ec0 ip.dst 10.0.245.1 and tcp.sport 554 and tcp.flags != SYN
and tcp.dport > 1023
#
# Mail outbound and simap
accept -i ec0 ip.src 10.0.245.1 and tcp.dport 25
accept -i ec0 ip.dst 10.0.245.1 and tcp.sport 25 and tcp.flags != SYN
and tcp.dport > 1023
accept -i ec0 ip.src 10.0.245.1 and tcp.dport 993
accept -i ec0 ip.dst 10.0.245.1 and tcp.sport 993 and tcp.flags != SYN
and tcp.dport > 1023
#
# allow ssh OUT
accept -i ec0 ip.src 10.0.245.1 and tcp.dport 22
accept -i ec0 ip.dst 10.0.245.1 and tcp.sport 22 and tcp.flags != SYN
and tcp.dport > 1023
#
# allow ping
accept -i ec0 ip.src 10.0.245.1 and icmp.type == ECHO
accept -i ec0 ip.dst 10.0.245.1 and icmp.type == ECHOREPLY
#
#########
#
#  inbound rules
#
#########
#
# allow ssh IN - telnet too until SSH works . . .
accept -i ec0 ip.dst 10.0.245.1 and tcp.dport 23
accept -i ec0 ip.dst 10.0.245.1 and tcp.dport 22
#
#########
# All which is not specifically allowed is denied.
#
reject -i ec0 tcp
reject -i ec0 udp
reject -i ec0 icmp
```

Our first problem – telnet is allowed by 'ipfilterd'. So the file was edited and changed so that only SSH was allowed inbound.

```
Indy 3# vi /etc/ipfilterd.conf

Indy 4# chkconfig |grep ipfilterd

        ipfilterd            on
```

Good, 'ipfilterd' is enabled at startup.

Is OpenSSH installed and configured?

```
Indy 5# whereis ssh

ssh: /usr/freeware/bin/ssh /usr/freeware/catman/u_man/cat1/ssh.Z

Indy 6# /usr/freeware/bin/ssh -v

OpenSSH_2.5.1p2, SSH protocols 1.5/2.0, OpenSSL 0x0090600f

        . . .
```

OpenSSH 2.5.1 will do just fine. Actually, we could have probably skipped this step since we were able to connect via SSH. But it is good to know that it is OpenSSH being used since there are no licensing issues to be dealt with.

Are root logins restricted to console and are users required to have a password to logon to the system?

```
Indy 7# more /etc/default/login
        . . .
# If defined, only allows root logins on the device specified.
# MUST NOT be defined as either "/dev/syscon" or "/dev/systty"!
CONSOLE=/dev/console
#
# Must all accounts have passwords?  If YES, and user has no password,
they
# will be prompted for one at login time.
PASSREQ=YES
        . . .
# Like PASSREQ, but locks out user if they have no password.
MANDPASS=YES
#
        . . .
```

Everyone must use a password, and root must use the console.

Syslog messages are sent to a centralized location?

```
Indy 9# more /etc/syslog.conf

        . . .
*.debug                 @wintermute.localnet
```

Here we see that syslog is indeed being shuffled off to another host. The machine accepting syslog was inspected and it was confirmed that messages were not only arriving, but being checked for significant events using Psionic Logcheck (http://www.psionic.com/).

Static ARP entries exist for Syslog, Mail, and Default Router?

```
Indy 10# arp -an

? (10.0.245.2) at 0:10:5a:2:32:a0 permanent
        . . .
```

Here we see the syslog server has been configured with a static ARP address. Other entries have been removed to eliminate redundancy. Why is this being done? Well to make a long story short the layer-2 implementation of Ethernet is very insecure. For a good idea of what attacks are available take a look at dsniff (http://www.monkey.org/~dugsong/dsniff/).

No accounts without passwords exist?

```
Indy 12# more password-audit.sh
#!/bin/sh
cat /etc/passwd | awk -F: '{print $1}' | while read login
do passwd -s $login | grep -v LK | grep NP | awk '{print $1}'
done
```

This is a little shell script I threw together for this audit. The script gets all of the user names from /etc/passwd and then determines if the account has no password. It only prints the accounts without passwords.

```
Indy 13# ./password-audit.sh

lp
nuucp
demos
OutOfBox
guest
```

We find five accounts without passwords, so those are subsequently locked.

```
Indy 15# passwd -l lp
Indy 16# passwd -l nuucp
Indy 17# passwd -l demos
Indy 18# passwd -l OutOfBox
Indy 19# passwd -l guest
```

Are shadow passwords enabled?

```
Indy 20# cat /etc/passwd

root:x:0:0:Super-User:/:/bin/csh
sysadm:x:0:0:System V Administration:/usr/admin:/bin/sh
cmwlogin:x:0:994:CMW Login UserID:/usr/CMW:/sbin/csh
diag:x:0:996:Hardware Diagnostics:/usr/diags:/bin/csh

      . . .
```

The passwd file looks good, all we need to know is if the shadow file really exists.

```
Indy 21# ls -l /etc/shadow

-r--------    1 root     sys          540 Jun 25 12:52 /etc/shadow
```

Sendmail is configured to only process queue?

```
Indy 22# ps -ef |grep sendmail
```

```
    root        818          1  0 15:38:07 ?       0:01
/usr/lib/sendmail -q15m
```

Unneeded services disabled in /etc/inetd.conf?

```
Indy 25# grep -v \# /etc/inetd.conf | sort

bootp   dgram   udp     wait    root    /usr/etc/dhcp_bootp dhcp_bootp
-o /etc/config/dhcp_bootp.options
chargen dgram   udp     wait    root    internal
chargen stream  tcp     nowait  root    internal
daytime dgram   udp     wait    root    internal
daytime stream  tcp     nowait  root    internal
discard dgram   udp     wait    root    internal
discard stream  tcp     nowait  root    internal
echo    dgram   udp     wait    root    internal
echo    stream  tcp     nowait  root    internal
exec    stream  tcp     nowait  root    /usr/etc/rexecd         rexecd
finger  stream  tcp     nowait  guest   /usr/freeware/bin/tcpd
/usr/etc/fingerd -L
ftp     stream  tcp     nowait  root    /usr/etc/ftpd   ftpd -l
login   stream  tcp     nowait  root    /usr/etc/rlogind        rlogind
mountd/1,3    stream  rpc/tcp wait/lc    root    /usr/etc/rpc.mountd
mountd

                  . . .
20 lines deleted from output to save space.
                  . . .

telnet  stream  tcp     nowait  root    /usr/freeware/bin/tcpd
/usr/etc/telnetd
tftp    dgram   udp     wait    guest   /usr/etc/tftpd  tftpd -s
/usr/local/boot /usr/etc/boot
time    dgram   udp     wait    root    internal
time    stream  tcp     nowait  root    internal
ttdbserverd/1   stream  rpc/tcp wait root ?/usr/etc/rpc.ttdbserverd
rpc.ttdbserverd
walld/1     dgram   rpc/udp wait    root    /usr/etc/rpc.rwalld
rwalld
```

There is plenty of garbage in the inetd.conf file, this means there are plenty of unneeded services available on this system.

```
Indy 26# vi /etc/inetd.conf

      . . .
```

After following the guidelines for what should be disabled, and editing the configuration file – here is what is left.

```
Indy 28# grep -v \# /etc/inetd.conf

sgi-dgl stream  tcp     nowait  root/rcv        /usr/etc/dgld   dgld -
IM -tDGLTSOCKET
sgi_videod/1 stream rpc/tcp wait    root    ?/usr/etc/videod
videod
```

```
sgi_fam/1   stream  rpc/tcp wait    root    ?/usr/etc/fam
fam
sgi_xfsmd/1 stream  rpc/tcp wait    root    ?/usr/etc/xfsmd     xfsmd
```

I am not entirely sure what these services do, and now they are suspect. I will have to do more research on whether or not they are necessary for this system.

Unneeded services disabled at system boot?

```
Indy 30# chkconfig

        Flag                 State
        ====                 =====
        acct                 off
        array                off
        audit                off
        autoconfig_ipaddress off
        autofs               on
        automount            off
        cachefs              on
        desktop              on
        fontserver           off
        gated                off
        ipaliases            on
        ipfilterd            on
        isdnd                off

            . . .
        30 lines deleted to save space
            . . .

        static-arp           on
        timed                on
        timeslave            off
        verbose              off
        videod               off
        visuallogin          on
        vswap                off
        windowsystem         on
        xdm                  on
        xfwp                 off
        xlv                  on
        yp                   off
        ypmaster             off
        ypserv               off

Indy 31# chkconfig autofs off
Indy 32# chkconfig lp off
Indy 33# chkconfig nfs off
Indy 34# chkconfig rfindd off
Indy 35# chkconfig routed off
Indy 36# chkconfig snmpd off
Indy 37# chkconfig timed off
Indy 38# chkconfig noiconlogin on
```

Seven unneeded services found, and disabled – noiconlogin set to 'on'. This setting prevents a list of all user accounts found in '/etc/passwd' from being displayed at the X login screen.

Tripwire is installed and database check performed remotely?

```
Indy 39# ls -l /usr/freeware/bin/tripwire

-rwxr-xr-x    1 root     sys       167104 Jun 24 16:08
/usr/freeware/bin/tripwire

Indy 40# /usr/freeware/bin/tripwire

### Phase 1:   Reading configuration file
### Phase 2:   Generating file list
### Phase 3:   Creating file information database
### Phase 4:   Searching for inconsistencies
###
###                     Total files scanned:           8083
###                           Files added:             3
###                           Files deleted:           0
###                           Files changed:           7978
###
###                     After applying rules:
###                           Changes discarded:       7962
###                           Changes remaining:       22
###
added:   -rw-r--r-- root           9 Jun 25 09:18:02 2001
/etc/hosts.deny
added:   -r-------- root         532 Jun 25 12:52:06 2001 /etc/oshadow

      16 lines deleted

changed: crw--w--w- tgarris        0 Jun 25 10:22:17 2001 /dev/pts/3
changed: crw--w--w- tgarris        0 Jun 25 13:05:52 2001 /dev/pts/4
changed: crw--w--w- tgarris        0 Jun 25 13:05:52 2001 /dev/ttyq4
changed: crw--w--w- tgarris        0 Jun 25 10:22:17 2001 /dev/ttyq3
changed: drwxrwxrwt sys         4096 Jun 25 13:08:05 2001 /tmp
### Phase 5:   Generating observed/expected pairs for changed files

      . . .
```

I have truncated the output from tripwire here. It is apparent that the database is pretty recent, but there are a few files that should not be audited for changes. Most of the lines showing changed files are from changes made during the audit. The database should be updated and re-uploaded to the tripwire server once the audit is completed.

TCP Wrappers deny all but local network access inbound?

```
Indy 41# cat /etc/hosts.allow

ALL: 10.0.245.0/255.255.255.0
ALL: 127.0.0.1

Indy 42# cat /etc/hosts.deny
```

```
ALL: ALL
```

TCP Wrappers looks good, all communication is limited to the local network.

TCP Wrappers govern access to all services remaining in '/etc/inetd.conf'?

What about the services left running from inetd? I am not sure to be truthful. Since they are not opened up to the network (via ipfilterd) and are unlikely to be (they are non-interactive) I feel it is safe to leave as is for now until more investigation can be done. A method of testing these services should be devised and tested to see if TCP Wrappers interfere with their operation.

```
Indy 43# grep -v \# /etc/inetd.conf

sgi-dgl stream  tcp     nowait  root/rcv        /usr/etc/dgld   dgld -
IM -tDGLTSOCKET
sgi_videod/1 stream rpc/tcp wait    root    ?/usr/etc/videod
videod
sgi_fam/1   stream  rpc/tcp wait    root    ?/usr/etc/fam
fam
sgi_xfsmd/1 stream  rpc/tcp wait    root    ?/usr/etc/xfsmd     xfsmd
```

Now we start working through the vulnerabilities found on Bugtraq.

Removed suid bit on midikeys binary?

```
Indy 44# whereis midikeys
midikeys:
Indy 45# find / -name midikeys -print
```

Apparently 'midikeys' is not installed on this system.

The 'array' service is disabled at system boot?

```
Indy 46# chkconfig |grep array
        array               off
```

This check should have already been done. This is duplication.

Users have no execute access to the file '/usr/lib/infosearch/bin/infosrch.cgi'?

```
Indy 47# ls -l /usr/lib/infosearch/bin/infosrch.cgi

Cannot access /usr/lib/infosearch/bin/infosrch.cgi: No such file or
directory
```

Info Search is not installed either.

The 'pmcd' service is disabled at system boot?

```
Indy 48# chkconfig |grep pmcd
        pmcd                on

Indy 49# chkconfig pmcd off
```

PMCD should have been listed in the earlier section dealing with startup services.

InPerson software uninstalled?

```
Indy 53# versions -b InPerson

I = Installed, R = Removed

   Name                    Date        Description
I  InPerson              04/22/2000  InPerson Desktop Conferencing, 2.2.1

Indy 54# versions remove InPerson

Reading product descriptions .. 13%
Reading /var/inst/hist
Reading product descriptions .. Done.
Pre-installation check .. 8%
Checking space requirements .. 16%
Pre-installation check completed
Installing/removing files ..  16%
Removing selected InPerson.man subsystems
Installing/removing files .. 31%
Removing selected InPerson.sw subsystems
Installing/removing files .. 94%
Removing orphaned directories
Running exit-commands .. Done.
Installations and removals were successful.
```

InPerson was found and uninstalled.

We finish this part of the audit with some file permissions of known buffer overflows.

```
Indy 50# ls -l /usr/bin/lpstat
-rwsr-xr-x    1 root     sys        23208 Apr 23  2000 /usr/bin/lpstat
Indy 51# chmod 0755 /usr/bin/lpstat

Indy 55# ls -l /usr/sbin/gr_osview
-rwsr-xr-x    1 root     sys       160376 Apr 23  2000
/usr/sbin/gr_osview
Indy 56# chmod 0755 /usr/sbin/gr_osview

Indy 57# ls -l /usr/sbin/dmplay
-rwsr-xr-x    1 root     sys        75284 Apr 23  2000 /usr/sbin/dmplay
Indy 58# chmod 0755 /usr/sbin/dmplay

Indy 59# ls -l /usr/lib/print/netprint
-rwsr-xr-x    1 root     sys        18276 Apr 23  2000
/usr/lib/print/netprint
Indy 60# chmod 0755 /usr/lib/print/netprint
```

NMAP Scans

The scan shown here was performed from the local network. The flags used during the scan are: -sS – Syn scan; -v – verbose output; -O – Operating System Identification using TCP fingerprinting; -P0 – do not ping the host before scanning it. Results were as expected:

```
wintermute# nmap -sS -v -O -P0 10.0.245.1

Starting nmap V. 2.30BETA18 by fyodor@insecure.org (
www.insecure.org/nmap/ )
Initiating SYN half-open stealth scan against indy.sgi.localnet
(10.0.245.1)

Adding TCP port 22 (state Open).
The SYN scan took 1141 seconds to scan 1517 ports.
For OSScan assuming that port 22 is open and port 30202 is closed and
neither are firewalled
Interesting ports on indy.sgi.localnet (10.0.245.1):
(Ports scanned but not shown below are in state: filtered)

Port       State       Service
22/tcp     open        ssh

TCP Sequence Prediction: Class=64K rule
                         Difficulty=1 (Trivial joke)

Sequence numbers: 9FBBA01 9FCB401 9FDAE01 9FEA801 A009C01 A019601
Remote operating system guess: IRIX 6.2 - 6.5

Nmap run completed -- 1 IP address (1 host up) scanned in 1146 seconds
```

It is interesting to note that even with the 'ipfilterd' rules in place that the operating system identification function of nmap was still successful. Also because of the packet filtering on the system the scan took an incredibly long time – about 20 minutes. I will not duplicate scan information for UDP or outside the network, but they were exactly as expected and took even longer to perform.

Audit Evaluation

Overall I feel that the audit is successful in its primary goal. Which is to prevent persons from outside the local network from accessing the machine. There were several steps taken to satisfy this goal: firewall, packet filtering at the host, TCP Wrappers and the disabling of unnecessary services. More could be done to secure the host, but I was unable to find anything akin to Bastille (for hardening Linux) for IRIX.

There were several major shortcomings in the presented audit methodology:

1. The procedure shown assumes the auditor is allowed to fix problems that are found immediately.
2. The procedure assumes that the auditor is a competent Unix administrator.
3. The audit does not fully address local security in the context of a user who already has permission to use the host but may subject the system to some type of misuse.
4. There was some duplication in several of the tests to be done.

5. The audit was most certainly not comprehensive, however relevant and recent guides for IRIX security administration are rare so there is little to compare the audit against.

Given the outcome of the audit, I would feel comfortable placing the system in a hostile environment where it is subject to attack from outsiders. I would not feel comfortable with the system in a multi-user context because too few steps were taken to detect and mitigate attacks from an authorized user.

One piece that would augment the efficiency of the audit would be better documentation about some of the services running on the host. There were several services that were left enabled in the inetd configuration file. It is not known what their exact purpose was nor if using TCP Wrappers would inhibit the operation of the programs.

But why do I feel that the system is secure enough for it's purpose? Well, let's take my gut feelings and compare them to some 'time-based security' numbers and see how effective the implementation of the audit findings are.

First, what are the threats? What am I concerned about? How likely is the attempt?

- An attacker is trying to break into the machine itself. For nothing other than the sake of doing so, or to use it in another attack (jump point) or denial of service.
- An attacker wants to read my electronic mail. Because I work in security, this is a very real threat.
- Finally, someone I may have offended along the line may want to send a denial of service my way.

The most likely methods of accomplishing these goals are as follows:

- Physically steal the computer. Not yet (<1%).
- Send me a virus or Trojan horse. About once every two months (2%).
- Break into the computer from the Internet. An attempt is made every day on my network – likelihood (100%).

TBS (Time Based Security) is a method of determining how secure a system is by factoring three variables. Protection (P_t) represents how long a system can resist a specific attack. Detection (D_t) represents the amount of time required before an attack is detected. Reaction (R_t) represents how long it takes to stop the attack.

As long as [$P_t > D_t + R_t$] a system can be considered secure. Lets evaluate the attack methods and see how we do! I will only cover a couple of the examples shown above to save time and space.

Break into the computer from the Internet – 100% likelihood of attack

Our detection time is low. There are multiple checks being performed that should detect an attacker successful or not.

D_t = [2min … 30hrs]

- Network Intrusion Detection System – best case 2 minutes reaction time, worst case 6 hours. This allows for detection and pager delivery and assumes that I am not at the system console during attack.
- Syslog anomaly checks – best case 6 hours.
- Tripwire database checks – worst case 24 hours + 6 hours sleep = 30 hours.

R_t = [1min … 6hrs]

- Best case 1 minute – I am at the same location and can disconnect the machine from the network.
- Worst case 6 hours – I could be at the office and be handling another more important incident.

$P_t = \infty$

- Packet filtering at the firewall
- Packet filtering at the host
- TCP Wrappers at the host
- Services are disabled at the host except SSH, which uses TCP Wrappers.

Best case: **[∞ > 3 min]** and worst case: **[∞ > 36 hrs]**
Conclusion: Secure against attack from the Internet.

It is safe to assume that we are not vulnerable to attack from the network. I feel that I should explain further about the infinite expected protection time on this specific setup. There are three requisites for a successful attack over the network. First, a vulnerability must have been discovered and an exploit written for that specific vulnerability. Second, that service must be running on the system for which it is to be used against. Third, the network must allow traffic to reach that service. While it is impossible to eliminate the first item, the second two have certainly been eliminated in this scenario and it was done through four layers of countermeasures (firewall, host packet filtering, TCP Wrappers and the disabling of unneeded services.) This approach is called the 'fortress' mentality – and while it is commonly frowned upon it is suited to the application of this specific system. It would be nice to lower the worst-case detection and reaction times from thirty-six hours, but for this application it is not necessary.

Virus or Trojan horse – 2% likelihood of attack

$D_t = \infty$

- There is no anti-virus software present.

$R_t = \infty$

- Reaction time is unknown, and therefore assumed infinite.

P_t = [99months … ∞]

- Okay this one is tricky. On average, going through my e-mail I receive about one virus every two months. Let's assume for simplicity that about 1% of known virus programs are for Unix based systems. That averages out to about 8 1/4 years before I can expect to see a virus that I am susceptible to. This is worst case and doesn't take into account that IRIX is less popular and hence less likely to be targeted than other Unix operating systems.
- Best-case scenario is that since I am using the most recent version of Netscape and the option for Javascript and Java is disabled for the mail reader unless a vulnerability that overrides these is found the risk is completely mitigated.
- Finally, since I do not run the application as root, the usefulness and ability for the virus or Trojan to spread is limited.

Worst case: **[99months < ∞]** and best case: **[$\infty = \infty$]**
Conclusion: Not completely secure against attack from Trojan or Virus.

However, this is one of those special cases where the costs of countermeasures outweigh the risks. Eight years from now I may get a virus, but I doubt that I will even own this machine in eight years. It is perfectly acceptable to run it without virus protection until the threat profile changes at which time a re-evaluation will be required. The costs of network-based anti-virus solutions far outweigh the value of this system. If this system were running Windows NT the threat profile would change instantly – since 99% of the malicious code I receive in e-mail would be expected to be effective.

Directions for future work

The audit could have been more simplified through the use of scripts. Probably 90% of the audit could have been automated. If I were required to do this specific audit across many machines I would have no choice but to automate the process. While the audit itself was not cumbersome it would be time consuming and mundane work to perform the audit on more than just a handful of machines.

Scripting would also allow for a less experienced administrator or auditor to perform the steps listed with less chance of confusion or mistakes.

IRIX like every other Unix is complex. While I have certainly mitigated much of the risk of a system compromise by performing the audit and implementing the fixes – I certainly have not mastered the intricacies of IRIX security; in order to do that I would have to study more about IRIX systems administration.

Future audits should include topics like system backup, time synchronization (ntp) and system monitoring.

If I could get access to a compiler for IRIX there are more tools for vulnerability assessment and prevention that could be ported or compiled.

A more comprehensive list of exploits and fixes for IRIX would be very useful, and take a great step towards providing better local security. Because IRIX has a smaller user base and security is not one of its main focal points there is little attention paid to new exploits in comparison to a more popular operating system like Windows NT or Linux. This would take more work, but if a better list were available I believe many other people would find it useful.

Finally the most impact would be gained through the use of a more recent version of the operating system. The current release is 6.5.11, which eliminates most of the vulnerabilities found in 6.5.4.

References:

Silicon Graphics Security Site
http://www.sgi.com/support/security/

Texas A&M IRIX Security Checklist
http://www-viz.tamu.edu/~sgi-faq/faq/html-1/security.html

Bugtraq Vulnerability Database
http://www.securityfocus.com

Silicon Graphics Backup, Security and Accounting Guide
http://techpubs.sgi.com/library/manuals/2000/007-2862-001/pdf/007-2862-001.pdf

Example ipfilterd configuration
http://ruff.cs.jmu.edu/uug/stuff/ipfilterd.conf

Silicon Graphics Freeware Collection
http://freeware.sgi.com/

Netscape Communications
http://www.netscape.com/

Psionic Security Software
http://www.psionic.com/

Dug Song's Dsniff Page
http://www.monkey.org/~dugsong/dsniff/

OpenSSH
http://www.openssh.com/

Tripwire
http://www.tripwire.com/

All links are current at time of writing.