GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Security Essentials Bootcamp Style (Security 401)"
at http://www.giac.org/registration/gsec

Niles Mills
Practical Assignment
GIAC Security Essentials Certification (GSEC)
Version 1.4b (amended August 29, 2002)
Option 2 - Case Study in Information Security

Title: Comprehensive Anomaly Detection (CAD)

Note: This paper assumes that the reader has a working knowledge of Linux administration, shell scripting, Perl programming and TCP/IP networking.

Abstract

A mid-size city in North Carolina had all of its servers and workstations directly connected to the Internet and was under continual attack (city administration, police department, fire department, water and power, etc.). With loss of service and data exposure as key concerns, the city was considering a set of traditional firewalls to mitigate the risk. An additional concern was that those firewalls could be compromised as well (without the city's knowledge) and leave them just as exposed as before installing the firewalls. Coupled with the security risks was a practical issue of a limited municipal budget, which steered the solution towards the world of open source.

When researching possible open source solutions, one utility stood out as an example of what we needed to accomplish: portsentry[1]. Portsentry[16] is a port-monitoring tool that is able to take action when a change occurs in the signature of a machine's ports. Put another way, if an intruder accesses a port that is not in the allowed port list, portsentry[16] can automatically add a packet filtering rule (among other responses) to block the intruder from any further connection efforts. Watching how portsentry[16] reacted led to the idea that we could build a set of firewalls that could monitor their own health once they were connected to the Internet. This monitoring went beyond the usual combination of Tripwire[25] and HID/NID systems and included the ability to take automated action in response to detecting a change in the known and expected state of each firewall. The actions were ranked by the severity of the detected change and could range from simple notification to network disconnect to full system shutdown. Several years after implementation, the firewalls have deflected a steady stream of attacks and in one case, a firewall took itself off the Internet when an ftp service was compromised, proving the value of the monitoring daemons.

Before Snapshot

Before beginning the project, the city was running Oracle on a HP9000 (HP/UX), accounting systems on IBM RS6000's (AIX) and MIPS-based minis (RISCOS), Microsoft Exchange, primary/backup Domain Controllers and fileservers on PC-based servers (Windows NT) and various applications on several hundred PC's running different flavors of Windows 95/98. PC's were networked via a mix of Microsoft networking and an aging Novell network. All of these machines were connected directly

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

to the Internet via an ISP-supplied router with each machine owning a static, external IP address. The city's intranet was a spider web of hubs and switches within buildings, bridges/frame-relay to connect some buildings and a private cable system (city-owned head-end unit) to connect other buildings.

Microsoft Windows-based machines were compromised on a daily basis via direct attacks (NT null-password attacks), email-delivered backdoors (Back Orifice) as well as a continual parade of virii and worms. Although there was no direct evidence that the AIX/HPUX/RISCOS machines were being hacked, anyone with access to the same wire as the city's subnet could have easily lifted a telnet login stream and logged in to those servers. Also at risk was the exposure of the police department's data to the rest of the city. In addition to wanting to isolate itself from the rest of the city, the police department was interested in establishing an independent, direct internet connection to the state's Justice department for access to online driving records and criminal records. With the city's security problems, however, the state would not allow the connection to be set up.

As an initial solution, the city asked their ISP to block external access to the usual Microsoft ports (137/udp, 138/udp, 139/tcp) as well as telnet (23/tcp) and Oracle (1521/tcp). The ISP assured the city that the ports were being blocked, but nmap[13] scans of the city's address space showed that the ports were still visible to the outside world.

At this juncture, the city elected to solve their problems without the ISP's help and a decision was made to install a set of firewalls.

During Snapshot

Building the firewalls consisted of installing GNU/Linux on PC-class hardware, configuring the Linux kernel[10], installing the Comprehensive Anomaly Detection (CAD) utilities and most important of all -- creating a complete signature of each firewall before ever connecting it to the internet.

1. Operating System (GNU/Linux) Installation

Three Dell PC's were purchased, each with 350 MHZ Pentium II CPUs, 64 MB of RAM, 8 GByte disks and two 3Com NICs. The Slackware[18] GNU/Linux distribution (Slackware v3.6, Linux kernel[10] 2.0.35) was installed on each PC. The kernel was configured to disable IP-forwarding and to only include those drivers necessary to support the installed hardware. All services except for ftp were disabled in /etc/inetd.conf and all daemons were disabled in the /etc/rc.d startup scripts except for named (DNS), sendmail[17] (SMTP), ssh and CERN[8] (HTTP proxy server). A TCP wrapper[24] was used for the ftp service. At the time these firewalls were built, named was still in the 4.7 source tree and susceptible to root compromise, so it was installed under /chroot/named and invoked via chroot. The firewalls are currently running the latest 9.X version of the BIND[6] source.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Since the firewalls were configured to be non IP-forwarding, there would be no way for PC's behind the firewall to surf the internet, so the CERN[8] http proxy server (v3.0) was installed. Although CERN[8] did not have the throughput proxy performance of SQUID[23] or Apache[22], it was selected due to its excellent security record. Recent changes to the way modern websites behave (Microsoft ASP-based servers, in particular) dealt a blow to CERN's[8] ability to proxy all pages and it was replaced with an Apache[22] proxy server from the Apache V1.3 source tree.

All routing on the firewalls is based upon static routes created at boot-time. No dynamic routing protocols (RIP, etc.) are installed. And although these firewalls were built some time ago, they have been kept current with the latest versions of the Linux 2.2 kernel[10], zlib[21], openssl[15], openssh[14], BIND[6], Apache[22], sendmail[17], wu-ftpd[20] and related utilities.

2. Network Topology

The city consists of two logical groups: the police department and everyone else. The network and firewalls were laid out as follows:

FW1 = Firewall between City and Internet
FW2 = Firewall between City and Police Department
FW3 = Firewall between Police Department and State Justice Department

Internet <> FW1 <> City <> FW2 <> Police Dept <> FW3 <> State Justice Dept.

3. CAD Installation

The CAD utilities were designed, written and tested over a several month period on an offline (not connected to the Internet) system. When they were complete, installation of the CAD system on the firewalls was accomplished by copying the CAD source tarball from a CD into the /tmp directory of each firewall. The tarball's contents were extracted, creating the following subdirectories in the firewall's /root directory:

Directory	Functionality
fence/admin	This directory contains routines that are common to all the individual utilities (described below in this section), as well as a template for creating new CAD utilities. Definition of users, interfaces, firewall rules and responses to actions are all set in this directory. The files that require modification are listed below in sections 3.1 through 3.4.
fence/attr	This utility monitors the attributes of every file found in each of the directories found in the $PATH variable. Monitored objects include links, files, pipes, inodes and directories. At run time, discrepancies

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

are considered to be of SEVERITY_MEDIUM, resulting in the external NIC being taken offline after notifications are sent to users.

fence/cops This is the veteran COPS[9] utility. It checks directory and file ownership, groups and permissions from the vantage point of correct configuration for system data (/etc/passwd, /etc/groups, uucp setup, mail setup, ftp configuration, world writable files that shouldn't be, etc.). If you can make it through COPS, you're in pretty good shape.

fence/daemons This keeps track of which daemons are running, as well as the MD5[12] checksum of the in-memory image of each daemon. The /proc filesystem allows you to treat running executables as though they are disk images, allowing the comparison of the MD5 value of a true disk image against the in-memory /proc/PID#/exe image. Daemons that have been altered after being loaded are thus easily identified.

fence/dev The dev utility keeps track of the entries in the /dev directory structure, notifying the system when new devices appear or existing devices drop off. Changes to the /dev structure after boot time could indicate a serious breach.

fence/diskspace The diskspace utility keeps track of the amount of free diskspace, free memory and consumed swap space. Getting this utility set up takes some time as it is not trivial to figure out what the 'normal' memory usage is. This is typically a matter of setting limits, letting the diskspace utility trigger an alert, analyzing the alert and adjusting the limits again. It took several months to get this setting right on each firewall.

fence/dirs The dirs utility tracks the expected directory structure of the firewall, reporting on newly created and missing directories. The presence of a new directory is considered an indication of a breach.

fence/files The files utility works in the same way as the dirs utility, tracking the expected file contents of directories, reporting on new files and missing files. Unlike the dirs utility, which is pretty easy to set up, it took several months to get the false alerts written into the 'skip' file for this module. Once it was done, though, it became a valuable addition to the tool set.

fence/fw This directory contains the firewall 'fw.run' kernel packet-filtering setup utility and is described in section 3.1, below.

fence/ftp This utility keeps track of all files uploaded to the ftp server directories.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

fence/kernel This utility keeps track of the kernel run-level, setting an alert if the level changes.

fence/links Much like the CAD dirs and files utilities, this module keeps track of soft links, notifying when new links appear or existing links are deleted, both of which indicate breaches. In addition, it will notify when the hard object underneath a soft link is deleted (ie., bad soft link), again indicating a breach.

fence/logins The logins module keeps track of all unexpected logins by validating against a set of allowed userids and ttys. Exceptions are treated as malicious logins.

fence/logs This directory contains the 'logs' utility which monitors the size of files in the /var/log directory. Since each firewall is running a log rotation/truncate utility (home-grown, 'trim_all'), excessive logfile size is an indication of unwanted activity, signalling a potential breach.

fence/md5 This utility keeps a Tripwire[25]-like database of MD5[12] values for selected directories and files. The utility is driven by a set of tables that are created once at system generation time. Every 10 minutes, the MD5[12] values of every file in the database are checked against the database and exceptions are taken to be an indication of a breach. A 'replace.value' utility allows one or more files to have their MD5[12] values updated in the database when making approved changes to the firewall.

fence/ports The 'ports' utility watches all open server ports, validates them against a list of known ports and notifies when unexpected ports appear as server ports. There is also a 'skip' mechanism that lets the administrator use the lsof[11] utility to accept some transient server ports as non-malicious. The 4.7 BIND named daemon was notorious in this regard and generated quite a few false alerts when this utility was first installed. Netdate can also generate false alarms with transient server ports.

fence/rcommands

This utility scans the entire drive for the presence of any r-prefixed commands. Although this appears to be a redundant utility, overlapping the CAD 'files' utility, this is different in that it scans the entire drive, while the 'files' utility scans a selected list of directories. In general r-prefixed utilities should not be anywhere on the system (rlogin, rdist, etc.).

fence/setuid This utility scans the entire drive for the presence of any new setuid commands. Although this appears to be a redundant utility, overlapping

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

the CAD 'files' utility, this is different in that it scans the entire drive, while the 'files' utility scans a selected list of directories.

fence/snort This module runs snort[19] on every interface, archives the results every 12 hours and aggregates the activity into attack-report summaries. This utility currently is only used as an administrative reporting utility, but could easily be extended to use the CAD notify function as well.

Four files required customization for each firewall:

3.1 fence/admin/fw.run (kernel packet-filtering rules)

/fence/admin/fw.run is a bash script that employs a table-driven approach to configure the kernel's packet filtering rules. The tables are embedded within the script itself (see Appendix for full source) and are processed at run time by the script reading itself to determine addresses and ports. An example of a data table within the fw.run script is

```
_networks_clean() {
  # Field 1:  IP Address block for interface
  # Field 2:  Comment
  192.168.100.0/24  Intranet
}
```

The fw.run script requires the private IP address block of the clean-side intranet, the address block of the external NIC, as well as a list of the services that will be advertised on the firewall. The script itself contains extensive instructions on configuration.

3.2 fence/admin/nics (determines internal/external NICS)

The fence/admin/nics file is a list of the firewall's network adapters and identifies which adapter is the 'clean-side' NIC and which is the 'dirty-side' NIC. While it might seem that one could determine which is which at run time by using the RFC-1918 private addresses as a clue, that doesn't work when the firewall is a barrier between two private address networks. This is in fact the case for FW2 in the city's infrastructure. The nics file contains clear documentation of the record format for the data.

3.3 fence/admin/services (describes severity levels)

The fence/admin/services file is a list of all the scripts that will use the CAD system's notify function. Each line in the script contains the script name that will call notify, as well as an associated severity, described in full later in this paper. The services file contains embedded instructions on how to add entries. The services files is an important construct because it allows utilities that do not ship with the CAD system to also use the CAD's notify function. Again, this is also explained further on in this paper.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

3.4 fence/admin/users (describes whom to notify)

The fence/admin/users file is a list of users that should be notified for the SEVERITY_LOW and SEVERITY_MEDIUM events. The list also distinguishes between local and remote users.

4. Firewall Signature Creation and CAD Startup

After all software had been installed (GNU/Linux, CAD, daemon upgrades), each utility in the CAD system was run to initialize its piece of the firewall signature. This was a simple matter of cd'ing into each of the /fence directories listed in section 3 (above) and running "./create". Running the create script would create a flatfile dataset in each subdirectory. The flatfile contained the signature of that subsystem when the firewall was clean. To get the CAD system running, entries were added to root's crontab database for each CAD utility and the system was ready for connection to the internet.

One of the liabilities of monitoring a comprehensive signature for a firewall is the care you must take when making approved changes to the system. One must always be thinking about what part of the firewall's signature is being changed before it is changed. As an example, if you are going to upgrade the sendmail daemon (a too-frequent task), you know that you'll be changing the MD5 values of the /sbin/sendmail daemon and probably those of the /etc/mail/*.cf files, too.

There are two ways to deal with that. One way is to make the changes and let the system trip an alarm and then clear the alarm quickly before the system shuts down. The other method is to know what the time window is between the MD5 checks and then get the work done within that window, replacing the MD5 values with the 'md5' utility's 'replace.value' function. Either way, it requires careful work to keep the signature intact and the system online.

5. CAD top-level Monitor/Response System

The CAD system operates by running the individual tests (described in section 3, above) on a frequent schedule via the cron scheduling facility. When a test completes with no errors, it exits quietly. When a test encounters an error, indicating that the signature of the firewall has been altered, the test first generates a message file with a description of the error and then calls the fence/admin/notify script with the following syntax:

fence/admin/notify name_of_calling_script full_path_to_message_file

After logging the event in an audit trail (fence/admin/audit_trail), the notify script then looks in the fence/admin/services table to determine what action to take. A few lines from the services table will serve to show its use:

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

/root/fence/dev/test	SEVERITY_LOW
/root/fence/md5/test	SEVERITY_HIGH
/root/fence/ports/test	SEVERITY_MEDIUM

Based upon the SEVERITY associated with the script that encountered the error, the notify script will take the following actions:

SEVERITY_LOW	email users in fence/admin/users table
SEVERITY_MEDIUM	email users, shut down internet-side NIC
SEVERITY_HIGH	email users, shut down all NICs
SEVERITY_EXTREME	message to console, halt machine

In the case of SEVERITY_LOW and SEVERITY_MEDIUM events, an administrator can still connect to the firewall through the intranet-side NIC, resolve the condition and keep running. In the case of SEVERITY_HIGH and SEVERITY_EXTREME events, physical access to the machine is necessary to allow restart, forensic analysis and subsequent recovery.

* An interesting note is that any root-priveleged process can take advantage of the CAD system's fence/admin/notify process, by simply making sure that fence/admin/services has an entry for the calling script. This means that the CAD system can be easily extended beyond the utilities that come packaged with it.

6. Individual CAD Utilities

The individual CAD utilities share a common format. Each subdirectory typically contains the following files:

Scripts:

list	A ksh or bash script to scan for items of interest.
create	A ksh or bash script that runs the 'list' script to create a flatfile named 'allowed'.
test	A ksh or bash script that first runs the 'list' script to create a flatfile named 'current' and then runs an intelligent comparison of the 'allowed' and 'current' flatfiles.

Datafiles:

allowed	A list of the allowed objects, created by the 'create' script.
skip	A flatfile that contains known exceptions that we don't want to trigger errors.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

current A list of the current objects, created by the 'test' script.

diffs A list of differences between the 'allowed' and 'current' flatfiles, taking into account the exceptions found in the 'skip' flatfile. When the 'diffs' file is greater than zero bytes in length, the individual CAD utility will call the fence/admin/notify script so that appropriate action can be taken.

The differences between utilities typically are found in the test script, which can vary from a simple 'ls' output to a complex analysis of 'lsof' and 'netstat' data. Each CAD script is documented internally. An example of one CAD utility (ports) is included in this paper's Appendix VI. All of the utilities can be reviewed by downloading the CAD utility set. (See reference [7], below).

After Snapshot

The firewalls achieved the three goals of isolating the city's machines from the internet, isolating the police department's machines from the rest of the city and allowing the police department to set up a direct connection to the state's Justice department. To date, although the snort[19]-based logging has shown many deflected attacks, there has been just one intrusion, wherein a ftp server (wu-fptd) on the primary, outward-facing firewall was compromised by a buffer-overflow attack, granting the attacker a root shell. Fortunately, the directory scanning daemon detected the directory creation that is part of the attack and as a response, disconnected the firewall from the internet, eliminating the root login of the attacker.

There are still outstanding issues that surround the potential compromise of Microsoft Windows-based machines from behind the firewall. The non-forwarding primary firewall is helpful when a PC is compromised and turned into a malware-driven server, as the newly created server port can not be seen from outside the firewall. For the same reason, compromised PC's can not establish outbound connections other than to the SMTP server or the HTTP proxy server. And there lies the vulnerability -- a malware application that wants to send information home via the HTTP proxy or via email can not be stopped. Also, a malware application can not be prevented from wandering around the city's intranet, scanning and attempting to break into other machines behind the firewalls. Thus far the city's effort to deal with this liability has centered around running virus detection software on all Windows-based machines, but that is only as good as the current virus definitions.

Alert readers will have noted that the monitoring, detection and response processes are all running as root, which is a liability in itself. It would take a non-trivial effort to change the daemons to run as non-root, as some of the system information needed by the monitoring daemons requires root privileges. Still, every little bit helps when you are putting layers of protection in place, so whatever could run as non-root would improve the security of the system.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

All in all, the CAD project was successful, eliminating the vulnerabilities identified at the beginning of the task. Subsequent to the completion of the CAD project, the cablemodem-based infrastructure was replaced with a GNU/Linux-based VPN, the police department began selling services to other police departments via GNU/Linux-based VPNs, malicious email attachments were made safe via procmail and a remote ssh/expect-based NOC was constructed for external monitoring and software deployment, but those are topics for other papers.

References

[1] Craig Rowland. Port Scan Detection and Active Defense System. Feb 2002.
URL: ttp://web.archive.org/web/20020207005420/www.psionic.com/products/portsentry.html (Aug 20 2003)
Note: The original PSIONIC website has been absorbed into Cisco and is now only available at the Internet Archive Project (Wayback Machine).

[2] William R. Cheswick, Steven M. Bellovin. Firewalls and Internet Security. 1994.

[3] D. Brent Chapman and Elizabeth D. Zwicky. Building Internet Firewalls. Nov 1995.

[4] Mark Grennan. Firewall and Proxy Server HOWTO. Feb 2000.
URL: http://www.ibiblio.org/pub/Linux/docs/HOWTO/Firewall-HOWTO

[5] Daniel Lopez Ridruejo. The Linux Networking Overview HOWTO. July 2000.
URL: http://www.ibiblio.org/pub/Linux/docs/HOWTO/Networking-Overview-HOWTO

Source Code:

[6] Internet Software Consortium. BIND utilities.
URL: http://www.isc.org/products/BIND/

[7] Niles Mills. CAD. 2000.
URL: ftp://www.dnsppp.net/pub/nmills/cad.tgz

[8] European Laboratory for Particle Physics. CERN proxy/webserver.
URL: ftp://ftp.dnsppp.net/pub/nmills/CERN.tgz
Note: This source is no longer available at CERN. This tarball is what we used for our project.

[9] Dan Farmer. COPS. 1992.
URL: ftp://ftp.dnsppp.net/pub/nmills/cops-1.04.tar.gz
Note: This source is no longer available at Purdue's FTP server. This tarball is what we used for our project.

[10] Linus Torvalds et. al. Linux kernel.
URL: ftp://ftp.kernel.org/pub/linux/kernel/

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

[11] Vic Abel, Ray Shaw and others. lsof.
URL: ftp://lsof.itap.purdue.edu/pub/tools/unix/lsof

[12] Ronald L. Rivest (MIT). md5
URL: ftp://ftp.dnsppp.net/pub/nmills/md5.tgz
Note: This is not the latest MD5 source, but reflects what we used for the project.

[13] fyodor@insecure.org. nmap scanner.
URL: http://www.insecure.org/nmap/nmap_download.html

[14] Tatu Ylönen and many others. SSH. OpenSSH.
URL: http://www.openssh.com/portable.html

[15] Mark J. Cox, Ralf S. Engelschall, Stephen Henson, Ben Laurie and many others. openssl.
URL: http://www.openssl.org/source/

[16] Craig Rowland. portsentry.
URL: ftp://ftp.dnsppp.net/pub/nmills/portsentry-1.0.tar.gz
Note: The original PSIONIC website has been absorbed into Cisco and is no longer online. This tarball is the one we used for the project.

[17] Eric Allman and others. sendmail SMTP server.
URL: ftp://ftp.sendmail.org/pub/sendmail/

[18] Patrick Volkerding. Slackware GNU/Linux.
URL: http://www.slackware.com/getslack/

[19] Marty Roesch and many others. snort.
URL: http://www.snort.org/dl/

[20] University of Washington. wu-ftpd ftp server.
ftp://ftp.wu-ftpd.org/pub/wu-ftpd/

[21] Jean-loup Gailly and many others. zlib compression library.
URL: http://www.gzip.org/zlib/

[22] Apache web/proxy server.
URL: http://httpd.apache.org/download.cgi

[23] SQUID: http://www.squid-cache.org/Versions/v2/2.5/
Note: not used for project. Only noted as a reference.

[24] Wietse Venema. TCP-Wrappers.
URL: ftp://ftp.porcupine.org/pub/security/

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

[25] www.tripwire.org. Tripwire checksum utility.
URL: http://www.tripwire.org/downloads/index.php

Appendices

Appendix I - fence/admin/services

```
# services
#
# Field 1: full path to service
# Field 2: severity, as defined in notify script
#
# Notes:
#
# 1. Fields are whitespace-delimited.
# 2. Record order is not important.
# 3. Comments begin with '#' and are ignored.
# 4. Comments begin in column 1 only.
# 5. Empty records are ignored.

/root/fence/dev/test            SEVERITY_LOW
/root/fence/ftp/test            SEVERITY_LOW
/root/fence/logins/test         SEVERITY_LOW
/root/fence/logs/test           SEVERITY_LOW
/root/fence/md5/test            SEVERITY_HIGH
/root/fence/attr/test           SEVERITY_LOW
/root/fence/ports/test          SEVERITY_MEDIUM
/root/fence/dirs/test           SEVERITY_MEDIUM
/root/fence/files/test          SEVERITY_MEDIUM
/root/fence/links/test          SEVERITY_MEDIUM
/root/fence/rcommands/rcommands  SEVERITY_LOW
/root/fence/rlogin/test         SEVERITY_LOW
/root/fence/setuid/setuid_chk    SEVERITY_MEDIUM
/root/fence/cops/test           SEVERITY_LOW
/root/fence/snort/test          SEVERITY_LOW

# End services
```

Appendix II - fence/admin/nics

```
# nics
#
# Field 1: short hostname, as returned by "hostname --short | tr [A-Z] [a-z]"
# Field 2: device name for nic (as defined by ifconfig)
```

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
# Field 3: clean or dirty
#
# Notes:
#
# 1. Fields are whitespace-delimited.
# 2. Record order is not important.
# 3. Comments begin with '#' and are ignored.
# 4. Comments begin in column 1 only.
# 5. Empty records are ignored.

# Remote

fw1     eth0   dirty
fw1     eth1   clean
fw2     eth0   dirty
fw2     eth1   clean
fw2     eth1:0 clean
fw2     eth2   dirty
fw2     eth3   clean
fw3     eth0   clean
fw3     eth1   dirty

# End nics
```

Appendix III - fence/admin/users

```
# users
#
# Field 1: short hostname, as returned by "hostname --short | tr [A-Z] [a-z]"
# Field 2: email_address
# Field 3: local or remote
# Field 4: service name or SERVICES_ALL
#
# Notes:
#
# 1. Fields are whitespace-delimited.
# 2. Record order is not important.
# 3. Comments begin with '#' and are ignored.
# 4. Comments can only begin in column 1.
# 5. Empty records are ignored.

# Remote

fw1     msp@XXXXXXXXXXX.XXX       remote  SERVICES_ALL
fw1     root@localhost            local   SERVICES_ALL
fw2     msp@XXXXXXXXXXX.XXX       remote  SERVICES_ALL
```

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
fw2      westj@localhost          local   SERVICES_ALL
fw2      root@localhost           local   SERVICES_ALL
fw3      msp@XXXXXXXXXXX.XXX        remote  SERVICES_ALL
fw3      westj@localhost          local   SERVICES_ALL
fw3      root@localhost           local   SERVICES_ALL

# Local

mojave    msp@localhost            local   SERVICES_ALL
sawtooth  msp@localhost            local   SERVICES_ALL
camelot   msp@localhost            local   SERVICES_ALL
hatcreek  msp@localhost            local   SERVICES_ALL
bigsur    msp@localhost            local   SERVICES_ALL

# End users
```

Appendix IV - fence/admin/notify

```
#!/bin/bash

# Usage:
#
# /root/fence/admin/notify full_path_to_current_script full_path_to_message_file
>/dev/null 2>&1 &

alias source=.
source /usr/bin/ksh/common.path

_shutdown_delay=10 # minutes

_fence=/root/fence
_services=$_fence/admin/services
_users=$_fence/admin/users
_nics=$_fence/admin/nics
_audit_trail=$_fence/admin/audit_trail
_errors=$_fence/admin/errors
_911=$_fence/admin/911
_hostname=$(hostname --short | tr [A-Z] [a-z])

_error()
{
   cat >>$_errors <<EOF
$(date): $*
EOF

   cat <<EOF
```

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
$(date): $*
EOF
   exit 1
}

#----------------------------------------#
# Get service name and notification type. #
#----------------------------------------#

_argument1=$1
_argument2=$2

if [ x"$_argument1" = x"" ]
then
   _error S001: notify invoked without service-name argument
fi

if [ ! -f $_services ]
then
   _error S002: $_services not found
fi

if [ ! -r $_services ]
then
   _error S003: $_services exists, but is not readable
fi

# (leading/duplicate spaces in records don't matter with "set --")
   _service_count=$(egrep -c ^$_argument1 $_services)

case $_service_count in
   0)
      _error S004: unable to locate service in $_services: $_argument1
      ;;
   1)
      set -- $(egrep ^$_argument1 $_services)
      _service_name=$1
      _severity=$2
      ;;
   *)
      _error S005: duplicate records for service in $_services: $_argument1
      ;;
esac

#-------------------#
# Get message file. #
```

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
#-------------------#

if [ x"$_argument2" = x"" ]
then
   _error M001: $_service_name invoked notify without message-file argument
fi

if [ ! -f $_argument2 ]
then
   _error M002: can not locate $_service_name message-file: $_argument2
fi

if [ ! -r $_argument2 ]
then
   _error M003: $_service_name message-file is not readable: $_argument2
fi

_message_file=$_argument2

#-------------------------------------#
# Perform actions based upon severity. #
#-------------------------------------#

_message_to_audit_trail()
{
   echo $(date): $_severity $_service_name $* >> $_audit_trail
}

_message_to_console()
{
   _date=$(date)
   echo -e "\r"                        >/dev/console
   echo -e "$_date: ALERT: $_severity \r" >/dev/console
   echo -e "$_date: $_service_name \r"    >/dev/console
   echo -e "$_date: $* \r"                >/dev/console

   while read _line
   do
      echo -e "$_date: $_line\r" >/dev/console
   done < $_911

   _message_to_audit_trail message sent to console: $*
}

_user_list()
{
```

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
  # $1 contains 'local' or 'remote'

  _email_list=$(egrep ^$_hostname $_users | fgrep -e SERVICES_ALL -e
$_service_name | fgrep $1 | fgrep -v -e '#' | sed  -e 's/^[[:space:]]*//g' -e
's/[[:space:]]*[[:space:]]/ /g' | cut -f2 -d\  | sort | uniq) # two spaces after '-d\'
}

_email_users()
{
  for _user_type in $* # where $* contains 'local' and/or 'remote'
  do
    _user_list $_user_type # load _users with list of email addresses

    for _user in $_email_list
    do
      _message_to_audit_trail emailing $_user

      _service_dirname=$(dirname $_service_name)

      echo $_extra_message > /tmp/notify.message.$$
      echo                 >> /tmp/notify.message.$$
      cat $_message_file  >> /tmp/notify.message.$$

      mailx -s "$(echo $_severity | sed 's/ERITY_/-/g') $(hostname) $(basename
$_service_dirname)" $_user < /tmp/notify.message.$$
    done
  done

  rm -f /tmp/notify.message.$$
}

_shutdown_nics()
{
  for _nic_type in $* # where $* contains 'dirty' and/or 'clean'
  do
    for _nic in $(egrep ^$_hostname $_nics | fgrep $_nic_type | fgrep -v
DISALLOW_SHUTDOWN | fgrep -v -e '#' | sed -e 's/^[[:space:]]*//g' -e
's/[[:space:]]*[[:space:]]/ /g' | cut -f2 -d\  ) # two spaces after backslash
    do
      _message_to_audit_trail shutting down ${_nic_type}-side NIC: $_nic
      ifconfig $_nic down # DEBUG ONE ITEM
      _message_to_audit_trail ifconfig $_nic down return code: $?
    done
  done
}
```

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
_delay()
{
   _message_to_audit_trail delay for $_shutdown_delay minutes

   _shutdown_delay_seconds=$(expr $_shutdown_delay \* 60)

   sleep $_shutdown_delay_seconds # yes, posix sleep takes 'm' as modified, but this is
more portable

   _message_to_audit_trail delay completed
}

_shutdown_machine()
{
   _message_to_audit_trail shutting down machine

   shutdown -h now # DEBUG ONE ITEM

   _message_to_audit_trail shutdown -h now return code: $?
}

_message_to_audit_trail alert received

case $_severity in

   SEVERITY_LOW)
      _extra_message=""
      _email_users local remote
      ;;

   SEVERITY_MEDIUM)
      _extra_message="$(hostname): DIRTY-SIDE NICS WILL BE SHUT DOWN IN
$_shutdown_delay MINUTES"
      _email_users local remote
      _message_to_console DIRTY-SIDE NICS WILL BE SHUT DOWN IN
$_shutdown_delay MINUTES
      _delay
      _message_to_console DIRTY-SIDE NICS HAVE BEEN SHUT DOWN
      _shutdown_nics dirty
      ;;

   SEVERITY_HIGH)
      _extra_message="$(hostname): ALL NICS HAVE BEEN SHUT DOWN"
      _email_users local
      _message_to_console ALL NICS HAVE BEEN SHUT DOWN
      _shutdown_nics dirty clean
```

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
      ;;

   SEVERITY_EXTREME)
      _extra_message=""
      _message_to_console MACHINE HAS BEEN SHUT DOWN
      _shutdown_machine
      ;;
   *)
      _error A001: unknown severity: $_service_name $_severity
      ;;
esac

# End
```

Appendix V - fence/admin/fw.run

```
#!/bin/bash

_networks_clean()
{
   # Field 1:  IP Address block for interface
   # Field 2:  Comment

   # Please ensure that NIC addresses are in
   # /usr/local/psionic/portsentry/portsentry.ignore

   192.168.100.0/24 Intranet
}

_networks_dirty()
{
   # Field 1:  IP Address block for interface
   # Field 2:  Comment

   # Please ensure that NIC addresses are in
   # /usr/local/psionic/portsentry/portsentry.ignore

         XXX.XXX.XXX.0/24 XXXXXX   SDSL 512KB
   0.0.0.0/0        Internet
}

_forwards()
{
   # Field 1: IP Address
   # Field 2: interface to be used for forwarding
   # Field 3: Comment (optional)
```

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
  192.168.100.3     eth2     mirror
  192.168.100.40    eth2     sparc
  192.168.100.128   eth2     bigsur
  192.168.100.129   eth2     hatcreek
  192.168.100.130   eth2     hatcreek wireless
}

_services_clean()
{
  # Clean-side services.

  # Field 1:  Protocol (tcp, udp, icmp or all)
  # Field 2:  Port (numeric) or port-range (number:number)
  # Field 3:  Service-name (used as comment)
  # Field 4:  Clean-side incoming addresses: Internal-host, netblock or "all"
  # Field 5:  Access-allowed: intranet, extranet or both
  # Field 6:  Comment (optional)
  #
  # Do not enable 8080 proxy, that is driven by /etc/proxy/allowed
  #
  # If "all" is specified, # open up for
  # entire clean-side netblocks.
  #
  # Protocol/port records can be repeated.

  tcp   20        ftp-data      all            both
  tcp   21        ftp           all            both
  tcp   22        ssh           all            both
  tcp   222       ssh-ITIC      all            both
  tcp   23        telnet        192.168.100.32    intranet
  tcp   25        smtp          all            both
  tcp   37        time          all            intranet
  tcp   53        dns           all            both
  tcp   67        bootps        all            intranet
  tcp   110       pop3          all            both      but will be limited to intranet in
_services_dirty
  tcp   119       nntp          all            intranet
  tcp   137       netbios-ns    all            both      but will be limited to intranet in
_services_dirty
  tcp   138       netbios-dgm   all            both      but will be limited to intranet in
_services_dirty
  tcp   139       netbios-ssn   all            both      but will be limited to intranet in
_services_dirty
  tcp   143       imap2         all            intranet
  tcp   220       imap3         all            intranet
```

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
  tcp   445        msoft-ds       all             both      but will be limited to intranet in
_services_dirty
  tcp   515        printer        all             intranet
        tcp   873        rsync          all             intranet
  tcp   1723       pptp           all             both      but will be limited in _services_dirty to
fw6
  tcp   6000:6010   x11           all             intranet
        tcp   6346       gnutella       all             both

  udp   37         time           all             intranet
  udp   53         dns            all             both
  udp   67         bootps         all             intranet
  udp   137        netbios-ns     all             both      but will be limited to intranet in
_services_dirty
  udp   138        netbios-dgm    all             both      but will be limited to intranet in
_services_dirty
  udp   139        netbios-ssn    all             both      but will be limited to intranet in
_services_dirty
  udp   445        win2k-netbios  all             both      but will be limited to intranet in
_services_dirty

        udp   500        ipsec          all             both

  icmp                            all             both
}

_services_dirty()
{
  # Dirty-side services.

  # Field 1:  Protocol (tcp, udp, icmp or all)
  # Field 2:  Port (numeric) or port-range (number:number)
  # Field 3:  Service-name (used as comment)
  # Field 4:  External-host, netblock or "all"
  # Field 5:  Comment
  #
  # If "all" is specified,
  # open up for entire dirty-side netblock
  # (0.0.0.0/0).
  #
  # Protocol/port records can be repeated.

  tcp   20         ftp-data       all
  tcp   21         ftp            all
  tcp   22         ssh            all
  tcp   222        ssh-ITIC       192.168.100.0/24  intranet
```

© SANS Institute 2003, Author retains full rights.

```
  tcp   25        smtp          all
  tcp   53        dns           all
  tcp   80        www           all
  tcp   110       pop3          192.168.100.0/24  intranet
  tcp   137       netbios-ns    192.168.100.0/24  intranet
  tcp   138       netbios-dgm   192.168.100.0/24  intranet
  tcp   139       netbios-ssn   192.168.100.0/24  intranet
  tcp   445       msoft-ds      192.168.100.0/24  intranet  replaces 137/8/9 for Win2000
  tcp   515       printer       192.168.100.0/24  intranet
        tcp   6346      gnutella      all
        tcp   1723      pptp          all

  udp   53        dns           all
  udp   137       netbios-ns    192.168.100.0/24  intranet
  udp   138       netbios-dgm   192.168.100.0/24  intranet
  udp   139       netbios-ssn   192.168.100.0/24  intranet
  udp   445       win2k-netbios 192.168.100.0/24  intranet  replaces 137/8/9 for
Win2000
        udp   500       ipsec         all

  icmp                          all
}

_proxy_server()
{
  # Field 1:  IP Address for proxy server
  # Field 2:  Comment

  192.168.100.1  Apache
}

_black_list()
{
  # Field 1: IP Block
  # Field 2: Comment

  210.0.0.0/8    Asia APNIC-CIDR-BLK2, Australia
  211.0.0.0/8    Asia APNIC-CIDR-BLK2
  218.0.0.0/8    Asia APNIC4
}

#######################################################
#####                                 #####
##### NO USER-MODIFIABLE CODE BELOW THIS POINT #####
#####                                 #####
#######################################################
```

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
#-----------------------------------------------#
#                STANDARD SETUP                 #
#-----------------------------------------------#

alias source=.
source /root/fence/admin/path
source /root/fence/admin/fw
```

Appendix VI - Example of single CAD utility (ports)

VI (1) create script:

```
#!/bin/bash

source /root/fence/admin/path

cd /root/fence/ports

./list tcp > tcp.allowed
./list udp > udp.allowed
```

VI (2) list script

```
#!/bin/bash

source /root/fence/admin/path

case $1 in
        tcp) netstat -tan | fgrep LISTEN                    | sort | sed 's/  */ /g' | cut -f4 -d" " ;;
        udp) netstat -uan | egrep ^udp   | fgrep '0.0.0.0:*' | sort | sed 's/  */ /g' | cut -f4 -d" " ;;
        *) echo invalid or missing argument ;;
esac
```

VI(3) test script

```
#!/bin/bash

#---------------------#
# Pre-Processing Setup #
#---------------------#

alias source=.                # PRECURSOR_ZERO
source /root/fence/admin/path # PRECURSOR_ONE
```

© SANS Institute 2003, Author retains full rights.

```
cd /root/fence/ports          # PRECURSOR_TWO
_baseline=$(pwd)              # PRECURSOR_THREE

#---------------------#
#   Global Variables   #
#---------------------#

#---------------------#
#      Subroutines     #
#---------------------#

_validate()
{
        # The subroutine is called with "valid" set to "no".
        # If all is well, this routine leave valid unchanged.

        protocol=$1
        port=$2
        service=$3
        interface="$4"

        # debug
        #
        # echo protocol  is $protocol
        # echo port      is $port
        # echo service   is $service
        # echo interface is $interface

        # -P = no port names (ie., don't check /etc/services)
        #      so that the netstat port will match the lsof port

        case $protocol in
                udp)   lsof -P -iUDP |
                          tee ./test.lsof.tmp.$$ |
                     fgrep -e COMMAND -e $port |
                       tee ./test.tmp.$$ |
                         egrep ^$service |
                                 fgrep "$interface" |
                                                          fgrep -q :$port >/dev/null 2>&1
                                     retval=$? ;;

                tcp)    lsof -P -iTCP |
                          tee ./test.lsof.tmp.$$ |
                     fgrep -e COMMAND -e $port |
                       tee ./test.tmp.$$ |
                         egrep ^$service |
```

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
                        fgrep -q \*:$port >/dev/null 2>&1
                            retval=$? ;;

                *)          protocol_lsof=UNDEFINED
                    retval=1 ;;
        esac

        if [ $retval = 0 ]
        then
                # Criteria match approved data.
                valid=yes
        else
                if [ ! -s ./test.tmp.$$ ]
                then
                        # Port was gone before we could check it.
                        valid=yes
                fi
        fi
}

_report_diffs_new()
{
        while read symbol data
        do
                port=$(echo $data | cut -f2 -d:)

                valid=no

                while read record_type service_exception interface_exception
                do
                        if [ x"$record_type" = x"exception" ]
                        then
                                _validate $protocol $port $service_exception
"$interface_exception"
                        fi
                done < exceptions

                if [ $valid = no ]
                then
                        echo port in question is port $port for protocol $protocol
                        echo
                        echo test.tmp.$$:
                        echo
                        sed 's/^/new port: /g' < ./test.tmp.$$
                        echo
                        sed "s/^/$protocol.diffs.new: /g" < $protocol.diffs.new
```

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
                echo
                        case $protocol in
                                tcp) netstat -tan ;;
                                udp) netstat -uan ;;
                                *)   ;;
                        esac
                echo
                ps xaflw
                echo
                echo test.lsof.tmp.$$:
                echo
                cat ./test.lsof.tmp.$$
                echo
                echo $protocol.current:
                echo
                cat $protocol.current
                echo
                echo snort logs:
                echo
            fgrep :$port /var/adm/snort/*/*
                echo
        fi

        rm -f ./test.tmp.$$ ./test.lsof.tmp.$$ # created above in _validate

    done < $protocol.diffs.new
}

#---------------------#
#   Main Processing    #
#---------------------#

./list tcp > tcp.current
./list udp > udp.current

for protocol in tcp udp
do
  diff $protocol.allowed $protocol.current > $protocol.diffs
  egrep -e '^>' $protocol.diffs  > $protocol.diffs.new
  egrep -e '^<' $protocol.diffs  > $protocol.diffs.missing

  > $protocol.errors

  if [ -s $protocol.diffs.new ]
  then
        _report_diffs_new >> $protocol.errors
```

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
   fi

   if [ -s $protocol.diffs.missing ]
   then
      echo Missing $protocol ports: >> $protocol.errors
      echo                          >> $protocol.errors
      cat $protocol.diffs.missing   >> $protocol.errors
      echo                          >> $protocol.errors
   fi

   if [ -s $protocol.errors ]
   then
      echo END >> $protocol.errors

                  # Save some debug info - not needed now that we're using lsof.
                  #
                  # timestamp=$(date '+%Y.%m.%d.%T')
                  # cp $protocol.errors $_timestamp.errors.$protocol
                  # ps xaf       >> $timestamp.errors.$protocol
                  # netstat -tan >> $timestamp.errors.$protocol
                  # netstat -uan >> $timestamp.errors.$protocol

                  /root/fence/admin/notify /root/fence/ports/test
/root/fence/ports/$protocol.errors >/dev/null 2>&1 &
         fi
done

#----------------------#
#     Housekeeping     #
#----------------------#

# End

END-OF-DOCUMENT
```

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Last Updated: April 28th, 2017

Upcoming Training

Click Here to
{Get CERTIFIED!}

SANS Riyadh 2017	Riyadh, Saudi Arabia	May 06, 2017 - May 11, 2017	Live Event
SANS Security West 2017	San Diego, CA	May 09, 2017 - May 18, 2017	Live Event
SANS Northern Virginia - Reston 2017	Reston, VA	May 21, 2017 - May 26, 2017	Live Event
SANS London May 2017	London, United Kingdom	May 22, 2017 - May 27, 2017	Live Event
SANS Melbourne 2017	Melbourne, Australia	May 22, 2017 - May 27, 2017	Live Event
SANS Stockholm 2017	Stockholm, Sweden	May 29, 2017 - Jun 03, 2017	Live Event
SANS Houston 2017	Houston, TX	Jun 05, 2017 - Jun 10, 2017	Live Event
Community SANS Ottawa SEC401	Ottawa, ON	Jun 05, 2017 - Jun 10, 2017	Community SANS
SANS San Francisco Summer 2017	San Francisco, CA	Jun 05, 2017 - Jun 10, 2017	Live Event
Security Operations Center Summit & Training	Washington, DC	Jun 05, 2017 - Jun 12, 2017	Live Event
Community SANS Boise SEC401	Boise, ID	Jun 05, 2017 - Jun 10, 2017	Community SANS
SANS Charlotte 2017	Charlotte, NC	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Rocky Mountain 2017 - SEC401: Security Essentials Bootcamp Style	Denver, CO	Jun 12, 2017 - Jun 17, 2017	vLive
SANS Rocky Mountain 2017	Denver, CO	Jun 12, 2017 - Jun 17, 2017	Live Event
Community SANS Portland SEC401	Portland, OR	Jun 12, 2017 - Jun 17, 2017	Community SANS
SANS Secure Europe 2017	Amsterdam, Netherlands	Jun 12, 2017 - Jun 20, 2017	Live Event
Community SANS Philadelphia SEC401	Philadelphia, PA	Jun 19, 2017 - Jun 24, 2017	Community SANS
SANS Minneapolis 2017	Minneapolis, MN	Jun 19, 2017 - Jun 24, 2017	Live Event
SANS Paris 2017	Paris, France	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS Columbia, MD 2017	Columbia, MD	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS Cyber Defence Canberra 2017	Canberra, Australia	Jun 26, 2017 - Jul 08, 2017	Live Event
SANS London July 2017	London, United Kingdom	Jul 03, 2017 - Jul 08, 2017	Live Event
Cyber Defence Japan 2017	Tokyo, Japan	Jul 05, 2017 - Jul 15, 2017	Live Event
Community SANS Phoenix SEC401	Phoenix, AZ	Jul 10, 2017 - Jul 15, 2017	Community SANS
SANS Cyber Defence Singapore 2017	Singapore, Singapore	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Munich Summer 2017	Munich, Germany	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Minneapolis SEC401	Minneapolis, MN	Jul 10, 2017 - Jul 15, 2017	Community SANS
SANS Los Angeles - Long Beach 2017	Long Beach, CA	Jul 10, 2017 - Jul 15, 2017	Live Event
Mentor Session - SEC401	Macon, GA	Jul 12, 2017 - Aug 23, 2017	Mentor
Mentor Session - SEC401	Ventura, CA	Jul 12, 2017 - Sep 13, 2017	Mentor
Community SANS Colorado Springs SEC401	Colorado Springs, CO	Jul 17, 2017 - Jul 22, 2017	Community SANS