Securing a Network Device Support Server Running Debian Linux

Douglas Ridgeway

Securing a Network Device Support Server Running Debian Linux.

Douglas Alan Ridgeway
GSEC, GCIH, GCFW

February 28, 2005

GIAC Unix Security Administrator (GCUX)
Version 3 - Option 1

Table of Contents

Abstract

This paper fulfills the requirements for SANS Securing Unix (GCUX) Certification. It covers building a Debian Linux tftp server in a secure manner. It includes policy based auditing and monitoring the server with a syslog infrastructure. Inorder to accomplish the security goals of the fictitious business GIAC Enterprises.

Define the Environment

GIAC Enterprises is the parent company of “El Pesky Taco”, a fast food chain specializing in San Diego style fish tacos. There are over 100 “El Pesky Taco” (EPT) locations along the West Coast and the Southwestern United States. Each EPT location has various Point of Sales (POS) terminals for ringing up orders and managing the books for the local store. Each EPT location is using an IPSec VPN tunnel to connect to the corporate home office of GIAC Enterprises. Every night the POS for each location will update the days' sales, inventory, etc. to the home office.

At the home office (GIAC Enterprises), the following hardware is used:
Cisco Pix 525 Firewall/VPN end point
4 AMD Opteron PCs with Red Hat Linux and Postgresql database for receiving the nightly updates
5 Pentium IV PCs with Debian Linux to support security and network operations such as Cfengine policies, backup server, central logging server, network intrusion systems and network systems monitoring
25 Pentium IV PCs with Windows XP for various office staff
2 Pentium IV PCs as Windows 2000 Domain controllers

For each EPT site the following hardware is used:
Cisco Pix 501 Firewall/VPN end point
3 to 5 Posiflex Point of Sale Terminals
1 Pentium IV PC with Red Hat Linux and Postgresql database for the terminals
1 Pentium IV PC with Windows XP as a machine for office work
1Shuttle SK43G Athlon XP Barebone with Debian Linux to support backup/restore firmware/configuration/network sniffs and to act as the local syslog for the Cisco Pix 501 for each EPT location

Below are a list of the machines which are relevant to the network security of GIAC Enterprises. The list includes the location of the machine, the name of the machine, the job it performs, the description of its use and the software used to fulfill the job.

Location: GIAC home office

Name: netman001
Job: Unix System Policy Server
Description: This server contains files which enforce the acceptable policy for each Linux machine in the GIAC Enterprises network.
Software: Cfengine (to enforce policy)

Location: GIAC home office
Name: netman002
Job: Central logging server
Description: This server runs Syslog-NG, a replacement syslog server. The firewall at each EPT site sends it's syslog messages to a Linux machine on the local network. Then the machine relays the syslog messages to this central server. The server analyzes the messages to generate alerts for problems and build reports regarding the status of the firewalls.
Software: Syslog-NG (the TCP syslog server) and Swatch (a tool which sends alerts when it's regular expressions match an entry in any log file)

Location: GIAC home office
Name: netman003
Job: Network Intrusion Detection Server
Description: This server monitors the network for malicious packets and sends alerts to the central syslog when identified.
Software: Snort (a network intrusion detection system)

Location: GIAC home office
Name: netman004
Job: Network Device Backup Server
Description: This server contains network device and POS device firmware, configuration files and network analyzer files for every firewall owned by GIAC Enterprises. It is the central repository for firmware for the devices of each store. After each store has the image backup to the local machine, it will be copied by scp to this server to be stored in Subversion and tape backup.
Software: Subversion (a file versioning system), Veritas Netbackup (tape backup software)

Location: One machine in each of the 100 EPT locations.
Name: supportwall001- supportwall100
Job: Network Device Backup Server for each local EPT store.
Description: Each store has this server to colect syslog messgaes from the firewall, wirte the messages to both the local syslog, and forward the same messages to the central syslog in the GIAC home office. It also runs a tftp server to backup and restore the firmware and configuration of the PIX firewall. The tftp server can also collect libpcap compatible network sniffs generated by the

firewall, when needed. When choosing the strategy of where the tftp server should be located in the network, the administrators debated the merit of one central tftp server to support all the firewalls vs. each location having it's own network device support machine. The factor which solidified the decision to place a network device support machine in each location is the ability to continue to support the firewall if the VPN was disconnected. Supporting the firewall includes collecting syslog messages and allowing the local manager to use the tftp server to help the network administrators reinstate the firewall's VPN. Since the machines would need occasional updates in the firewall's firmware and/or configuration, the administrators were concerned that sometimes the update could break the VPN. Once the VPN was broken, the costs involving downtime for accounting and last minute flight/hotel to each location was greater than keeping a low cost machine at each location to support the firewall. If the VPN disconnects, the manager could be contacted, given the current password for the firewall Web browser GUI, access to the minimal Linux user account, and then follow the steps as per the Managers Manual given as part of their training. The manual shows how to login to the Linux machine, "ls" the /tftpboot directory to find the name of the last working firmware/configuration, then load it to the firewall. The network administrators are confident that this process allows for significant cost savings and minimal downtime.
Software: atftpd (to backup and restore firewall firmware and configurations), Syslog-NG (to collect and forward the syslog information), and Samhain (to track changes in key files, to search for rootkits and track login/logout events)

The creation of the machines supportwall001- supportwall100 is the focus of this paper. The operating system of supportwall(001-100) is Debian Linux with a Linux 2.4.18 kernel. Unnecessary default "small services" were shutdown and removed where possible. The job to start services is left to the startup scripts and the policy enforced by Cfengine. OpenSSH Secure Shell server was added. The syslog server was upgraded to Syslog-NG. Cfengine agent was installed to configure the machine and maintain a standard security policy.

Define the New Software or Service

The service to be installed on supportwall(001-100) is atftpd. atftpd is a Trivial File Transfer Protocol (TFTP) server. PIX firewall's os is upgraded by updating the firmware with a tftp server. The PIX also backs up its configuration in a single text file. Both the firmware and the configuration can be backed up and restored with a TFTP server. The TFTP server is also needed to support/store libpcap compatible file captures generated by the Cisco firewall network analysis.

Security Issues Introduced by the New Software

Installing atftpd introduces a range of security problems. The SANS Top 20

Vulnerabilities[1] list does not specifically address any issues with atftpd. It does however mention tftp servers in section "W1. Web Servers & Services". The main advice given is to restrict or remove unneeded services which could help an attacker. The Top Twenty lists possible unneeded programs of which tftp servers are included. The SANS Top Twenty mentions tftp servers as a possible problem because they were originally designed without any concept of security. Listed below are security issues which need to be addressed for both tftp servers in general and specific issues with atftpd .

TFTP General

TFTP servers were created at a time when little thought was given to security. Their design emphasized open access and speed. As a result, the following features of TFTP are security risks:

- The packets are sent in "clear text"
- TFTP does not authenticate access
- TFTP servers have traditionally suffered from "buffer overflows" and "directory traversal vulnerabilities"
  http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0183
  http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0498
  http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0380
- The service needs to be started as root to access a low port number.

atftpd Specific Vulnerabilities

- ATFTPD Remote Filename Length Buffer Overrun Vulnerability
  http://www.securityfocus.com/bid/7819
  http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0380
- ATFTP TFTP-Timeout Command Line Argument Local Buffer Overflow Vulnerability
  http://www.securityfocus.com/bid/7906
  http://www.securityfocus.com/bid/7902

The buffer overflows listed for tftp/atftpd are a serious threat. Buffer overflows can be used to execute arbitrary code. If the buffer overflow is accessible remotely it allows an attacker to take control of the machine.

Understanding the security impact of the listed vulnerabilities

The combined two lists of vulnerabilities impacts security in different ways. The steps to remediate the vulnerabilities are different. To differentiate the remediation of the vulnerabilities they will be grouped in the following manner: Programming Bugs and Insecure Features.

1 SANS Top 20 Vulnerabilities Version 5.0 October 8, 2004

Programming Bugs

Each of the security issues mentioned with a bugtrack ID and/or a CVE/CAN ID are programming bugs. These bugs allow an attacker to bypass normal security processes and either gather information the attacker normally would not have or execute arbitrary code of the account which owns the running process. This is a difficult problem to prevent, because it is not totally predictable. No one expects to have a security bug in their program. Even experts in security programming will make a mistake[2]. This is why security requires steps to harden the os and the service. Then maintain security by adding security patches on a regular basis and enforce security configuration by policy.

Insecure Features

TFTP sends all packets in clear text, does not authenticate access to the server, and requires a "low" IP port number so only the root account can start the service. This is the design of TFTP. The clear text can be a problem if a network sniffer is running. This will allow an attacker the opportunity to see the information transferred over the network. In this case the information can be a firewall configuration. This is the type of information which can be used by an attacker to find more holes in the network. Unauthenticated access to a TFTP server can be dangerous since it will allow an attacker to upload to or download from the server without any hindrance. So if we have a firewall configuration in the TFTP server directory, the attacker could copy it to examine possible holes in the network. Also if an attacker can load a program onto the machine and get it to execute as root, he can take control of the machine. TFTP needs to bind to a low port number, which requires the root account. Running TFTP as root is risky if an attacker finds a programming bug in the TFTP server. The attacker can use a script to make use of the bug and send commands to the OS as root. All of the above need individual steps to remediate the insecurities.

Implement the Solution

Remediation for the vulnerabilities

To remediate the issues with Programming Bugs the following tasks need to be completed:

- Run the latest patched code. This will fix known bugs. It prevents worms and low talented attackers from gaining access to the system.
- Run atftpd in a chroot()ed environment in order to keep future vulnerabilities contained in a minimized compartment of the OS. While a skilled attacker may compromise the specific service, a properly built chroot()ed environment will prevent the attacker from getting access to the system binaries and kernel. Keeping an attacker out of the main system binaries and kernel will prevent the attacker from installing backdoors and hidden tools in the main OS.

2 OpenBSD is known for it's commitment to secure by default installation. On June 26, 2002 the default install of OpenBSD installed with a remote security hole. http://www.openssh.org/txt/iss.adv

- Create and enforce a policy based configuration to harden the OS. Items which need to be addressed by the policy are correct setup of the chroot()ed environment, the ability to monitor if the process is running or not running and address it, the ability to assure the correct .conf files are in place, and the ability to report to syslog if something is not correct.
- Monitor the integrity of the system to confirm specific files have not changed both with a policy and with a change control tool.
- Use a tftp server which properly drops the root privilege and changes its UID to a different account.
- Define permissions in the chroot()ed directory which limit the account "tftpuser" to only the /tftpboot directory.
- Use a tftp server which cooperates with the restricted permissions.
- Check the local network for possible sniffers before using the tftp server.

To remediate the issues with Insecure Features the following steps need to be addressed:

- Place the tftp server in location physically close to the firewall it supports. Within the CIA-Triad (Confidentiality, Integrity, Availability) tftp servers do not provide confidentially or integrity. Confidentiality can be compromised because tftp servers lack a means to encrypt the contents of the data sent. Tftp servers lack integrity in that they use the UDP protocol which is often referenced as an unreliable protocol due to it's inability to assure the receiving side has the same packet as the sender intended. To address both issues, the tftp server will be located in each store.
- Test the machines on the network for promiscuous mode. All machines on the local store network are configured as per the company network administrators which excludes the installation of a network sniffer. Both the Windows and Linux machines are scripted to run a program to check if the network card is in promiscuous mode. The Windows machines use PromiscDetect[3] and the Linux machines use ifstat[4]. Before anything is sent to the tftp server (backups of the system firmware or the firewall configuration), the network admins also run Sentinel[5]. Hence if none of the machines on the local network are in promiscuous mode, it is unlikely that a network sniffer is running.
- Use TCP wrappers to allow the ip address of the frewall access to the tftp server and deny access to all other ip addresses. Since tftp does not have a method to authenticate access, the next best method of access control is to control which machines can use the tftp server.

3 PromiscDetect. A utility to check if the windows (NT/2000/XP) machine is running in promiscuous mode. http://ntsecurity.nu/toolbox/promiscdetect/

4 Ifstat: A utility to check if the Linux machine is running in promiscuous mode. http://la-samhna.de/misc/index.html

5 Sentinel: A unix tools to remotely check if machines are in promiscuous mode. http://www.packetfactory.net/Projects/sentinel/

- Log the use of the tftp server to syslog. When the setup is complete both tcp wrappers and the tftp server will log the use of the tftp server. The configuration allows the network administrators can use the logs to audit attempted use and the actual use of the tftp server. This allows an audit trail to track the possibility of an attacker. A good audit trail can inform the administrators of what actions have taken place, giving them the information to evaluate the incident.

Installing atftpd in a secure manner

Before describing the installation of the tftp service, it is necessary to describe the features of the OS. Debian Linux can be installed as one of three types of distribution levels. The levels are “stable”, “testing”, and “unstable”. The various levels have code names. Each code name at this time is based on a character from the movie “Toy Story”. The current “stable” release is known as “Woody”. Unstable is always known as “Sid”. The current “testing” level is known as “Sarge”. The differences among the levels are how the current packages are released, tested and the support for security. Unstable (Sid) changes almost daily. It has all the latest configuration and updates. If a user desires to run the latest software versions, the latest shared libraries and a wider array of software, then unstable is the level a user should consider. Running unstable is risky because programs can unexpectedly break. Since unstable changes almost daily, it can tax the resources of the administrator supporting it. The testing level is where packages have been accepted from unstable and is now being tested and prepared for the next stable release. Debian does not recommend testing for a production server. It's main risk is both security and big fixes are not updated very quickly. The focus for testing is getting the whole OS working well. At the last phase of testing the developers start considering how they will address big fixes by applying a patch or just promote a package from unstable to testing. Hence the recommended way to run Debian for a production server is to use the stable level. Stable stays very current on security patches. The main disappointment many administrators have for the stable distribution is it's lack of the most current version of programs. Stable's focus on “stability” prevents the official packaging system from offering the latest versions of many programs.

An administrator of Debian Linux does not have to be limited to just the packages from the official Debian branch. The administrator can do any of the following: 1) Add to the /etc/apt/source.list file a url to a 3rd party package source. Hence one should verify they can trust the source before installing the packages. 2) Setup a test machine to download the source, compile the code and then build the .deb package. Then the packages can be hosted on a local archive. 3) Download and compile the code to the specific machine.

The administrators for this network prefer option 2 to manage packages with the

Debian apt tools. They have built an infrastructure to proxy the .deb files on a local server. They proxy/cache the files to minimize the downloading of files over the Internet. Hence the first time installation can get the latest files without contacting a server outside the company network. The proxy also provides GIAC Enterprises the ability to build/rebuild a machine with the latest packages if access to the Internet is lost. Since the proxy caches the packages, the packages are copied to a server which is separated from the network for building new machines. The machine in the separate network has a script to start an iptables firewall which only allows ssh outside when the source is the local machine. So when packages need to be updated, the machine is attached to the main network, ssh to the proxy, and then rsync the packages to the cache. The machine is then disconnected from the main network and then reattached to the network for building machines. Then another script is run on the proxy/cache server to change the firewall policy to allow addresses, within the separate network, access to the .deb package archive.

To make a secure installation of Debian Linux the installer must consider the following:

- What distribution level will be used?
- What can be automated for the installation?
- How will the filesystems be arranged?
- What accounts will be needed?
- What will be the name of the machine?
- Will the machine have a dynamic IP address or a static one?
- What drivers will need to be installed?

To create the machine supportwall001 it will run with the distribution level “Stable” (named Woody at this time). This is the vendor recommended practice for production servers. At this current time, the Woody distribution does not have an official automated installation. An outside tool named FAI (Fully Automatic Installation) can be used. It is a mix of PXE booting with Cfengine, Expect, and Perl scripts. It requires a complex configuration and the use of NFS to copy the files. The network administrators decided not to use FAI due to its complexity and its dependence on NFS. The current testing distribution (Sarge) does have an automated preconfiguration file, but that would require the use of the testing distribution which is not recommended for a production server. So the installation of the OS will start as a manual process. Then after the basic installation Cfengine will be used to harden the OS and act as a configuration and audit policy.

When building the filesystem the following layout will be used:

Table 1.0: Filesystem assignment for fstab file			
Drive ID	***Mount point***	***Options***	***Notes***
sda1	/ (root)	defaults, errors=remount-ro	
sda5	/tmp	nodev, noexec, nosuid	To prevent simple trojans from executing
sda6	/home	nodev, noexec, nosuid	To prevent easy introduction of foreign executables
sda7	/var	defaults	The chrooted setup will be on this filesystem. It will need to make two devices to complete the chroot. Also executables will run within /var.
sda8	/opt	nodev, defaults	No devices needed in /opt.
sda9	/usr	nodev, defaults	No devices needed in /usr.
sda3	swap	n/a	n/a

Before settling on the above filesystem assignments various experiments were tested with /dev and /var/chroot as separate filesystems. The experiments produced poor results in stability. The filesystem options specified in the "Options" column will be introduced after the install. The default fstab file will be replaced with an fstab file from the Cfengine server.

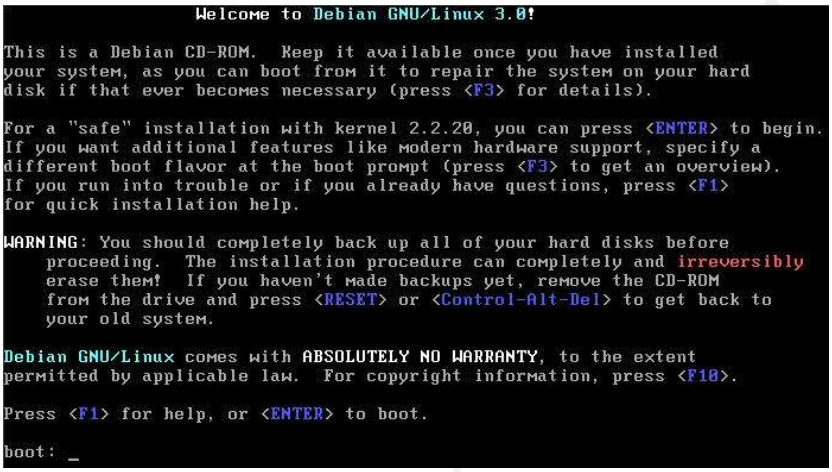
At some point, the Debian installation will prompt the installer to create login accounts. The following accounts will need to be created: root (default superuser), tftpuser (the account the tftp server will use), user (a generic user), and adminuser (an account which will be sudoers version of superuser). During the installation, the root and adminuser accounts will be created. After the installation, the tftpuser and generic user will be created with Cfengine. The tftpuser account will have the shell changed to /bin/false so the account could not be used for login nor offer a shell on a compromised server.

The name of the server during the installation will be supportwall001. Debian installations work best when they have access to a package archive. The installation will take place on a network separated from the rest of the company network. This is the "build machine" network. During the installation, the machine will use DHCP. Then just before it is sent to an EPT store, the DHCP client will be removed and it will be given a static address and routing information.

The drivers to be installed are APM (allows the shutdown command to turn off

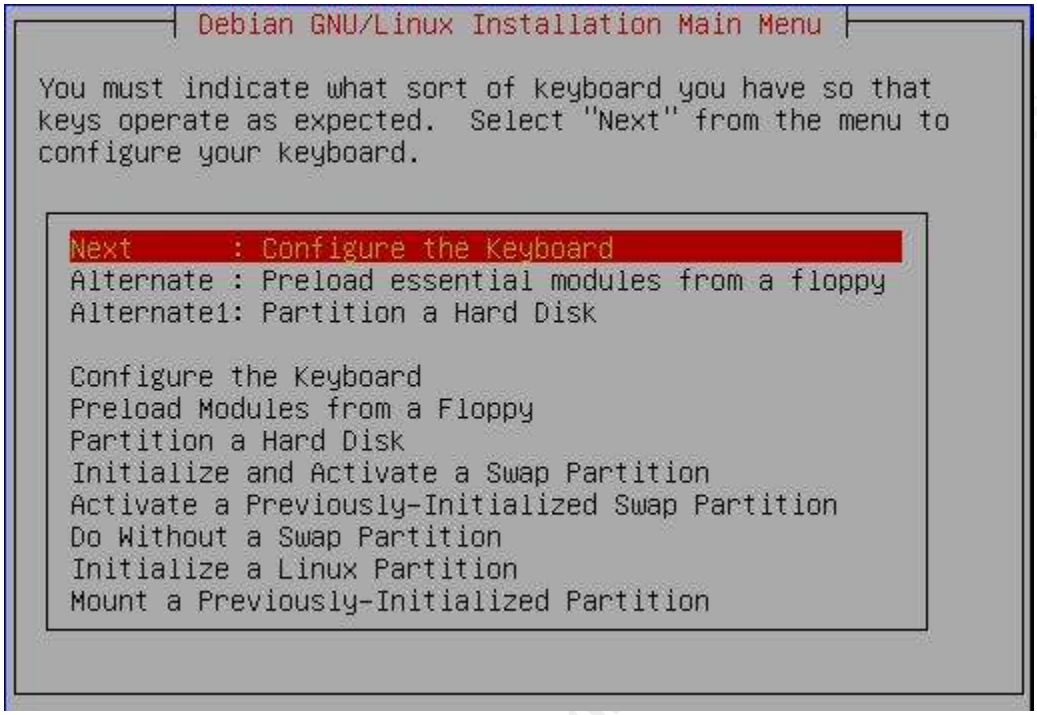
the machine at the end of OS shutdown), ide-scsi (treats IDE drives like scsi devices), and sg (generic scsi support). The default kernel already has the driver for the network interface, so no extra support is needed.

Before the installation the .iso files from the Debian web site have already been downloaded, verified against a MD5 hash and burned to CDs. The machine is connected to the separate installation network. Debian can install the latest files with a “network installation”. This is where a CD or .iso file will boot the server, install a minimal version of Debian, then download the rest of the packages from the Debian servers (or in this case a local cache). Place the CD in the new machine's CD drive, and boot it up. When ready, the prompt will ask for options to boot into the installation.



Type “bf24” and press <Enter>. This will allow the installation to build the OS with the 2.4.18 kernel.

The next screen will prompt for the choice of language. Choose “en – Choose this and press Enter to proceed in English”. In the next menu pick “English (United States)”. Press <Enter> to continue the installation. At this point the install will show the main installation menu. As shown in the next screen shot, choose “Configure the keyboard”.



In the next menu choose, “qwerty/us : U.S. English (QWERT)”.

Now the menu offers the option to partition the hard drive. Choose this option and accept the drive /dev/sda. Click <Enter> on the next few screens until the partition utility cfdisk appears[6]. The screen below shows cfdisk with an unformatted hard drive.

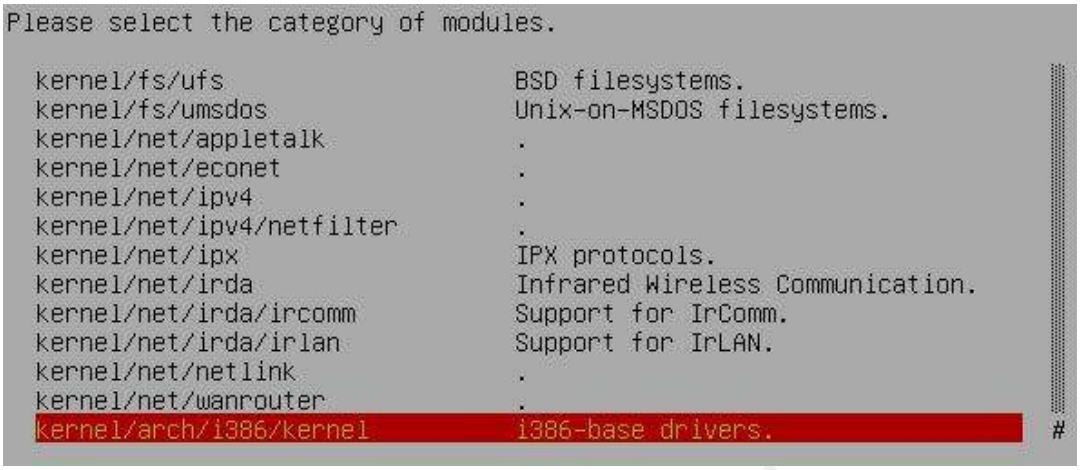
Partition the drive with cfdisk until it looks like the following screen shot.

6 Note the drive size looks like a small hard drive. The installation is setup within VM for purposes of this paper. Hence the small drive size.

Write the partition table to the disk and quit the cfdisk program.

The menu will prompt to “Initiate and Activate a Swap partition”. Choose this option. For each partition, the installer will prompt to run a bad block scan. Choose “yes” for each prompt. This will warn about hard disk problems. If any errors are found it can remap the bad blocks as unusable. Confirm initialize the hard drive. Repeat the same steps for each file system/partition. In addition to the previous steps, each partition will prompt about the format of the file system and the mount point. Choose the EXT3 format and the mount points per partition as described in *Table 1.0: Filesystem assignment for fstab file*.

When finished with the formatting and the mount points, choose the option “Install Kernel and Driver Modules”. The menu will prompt it found the installation CD, do you wish to use it; choose “yes”. Now the menu will offer the option “Configure Device Driver Modules”. The image below shows a menu of driver modules available.



Within the section, kernel/arch/i386/kernel, choose the kernel module “APM – Advanced Power Management BIOS support”. Do the same for kernel/drivers/scsi and choose the kernel modules “ide-scsi – SCSI emulation support” and “sg – SCSI generic support”. There is not a need to select a driver for the network card because the default kernel already includes the specific driver.

After choosing the above options, exit the kernel module menu. Choose the next option, “Configure the network”. When prompted for the hostname, type “supportwall001” and press <Enter>. Choose “yes” when asked if you wish to use DHCP. A banner will display if it succeeds in getting the IP information.

Press <Enter> and then the menu will show the option, “Install the base system”. Choose this option. The installer will start the minimal install. The screen should now look like the following graphic.

At the end of this, choose the option, “Make System Bootable”. Choose to install LILO in the MBR. When finished, choose to reboot the system.

On boot up, a new set of prompts will appear. The first default prompt will state the installation has succeeded, press <Enter>. The next menu will ask if the clock is set to GMT; choose “yes”. Next it will ask to enable MD5 passwords; choose “yes”. Then it will ask to enable shadow passwords; choose “yes”. Next it will prompt for the password for root. Choose a strong password and press <Enter>. The install will prompt to confirm the password by reentering it. Next the installation will prompt for a new user. Choose “yes”. Create the account “user “. The installation only allows for one new user, so the tfpuser will need to be added after the installation with Cfengine. The next few screens of the installer will ask if certain packages can be removed. Instruct the installer to remove the “pcmcia” packages and not use the “ppp” packages. Next the screen prompts if the desire is to keep using the CDs for the rest of the installation. Choose “no”. Then it will ask for another “apt source”; choose “yes”. The menu will list the options to add to apt. Choose “edit sources by hand”. This will open the text editor nano with the file /etc/apt/sources.list. The following should be added to the sources.list file.

deb http://deb-proxy:9999/main stable main non-free contrib

Then save the file. The installation will prompt if another apt source it required, choose no. Then it will prompt to get security updates from security.debian.org; choose Yes. This configuration will setup this supportwall001 to first get all packages from the proxy/cache server. The security updates are also be available from the cache. Residing in the same network is a mirrored version of the Cfengine policy server. This allows the new machine to act with the Cfengine server as if it were on the main network.

In the final stages of the install, it will ask to run utilities to setup programs. The first of which is named “tasksel”; choose “no”. Next is dselect; choose “no”. Then confirm to remove the packages selected earlier for removal. Next it asks what kind of prompts you desire from “debconf”, choose “noninteractive”. Choosing

noninteractive just places the default settings for the program without any prompting from the user. The configuration files to be used are kept on the Cfengine server, so there is not a need for interaction. Choose yes to erase previously downloaded .deb files. Default the installation assumes the machine will have a mail server. The default mail server is Exim. Cancel the installation of Exim by choosing the option number (5). Now the machine is ready for login.

Login as root. Then install OpenSSH (apt-get install ssh). The ssh packages includes Secure Copy (scp), Secure FTP (sftp), the Secure Shell (ssh) client, and the Secure Shell server (sshd). This program allows encrypted remote shell access and encrypted remote file copy over the network. Next install Cfengine (apt-get install cfagent2). Cfengine2 is not in the official Debian Stable archive. This package of Cfengine2 was precompiled and cached on the apt-proxy server. To run Cfengine, confirm the following directory is in place; /etc/cfengine. By default this directory is empty. When Cfengine is working it will have two files in this directory: update.conf and cfagent.conf. The file update.conf is a minimal file. It's purpose will be to validate contact between the cfagent client and the Cfengine policy server (netman001) and download the latest cfagent.conf file. The update.conf file is always parsed before cfagent.conf. This allows the network administrator to write a policy in a central location and have the policy distributed to the whole network. To get the first file, scp the update.conf to the /etc/cfengine/ directory (scp user@netman001:/user/cfengine/update.conf /etc/cfengine/update.conf).

Here is the update.conf file. The "#" is the indicator for comments (like comments in Perl).

Control: #Cfengine .conf files always start with the control: section

IfElapsed = (5) #IfElapsed is the number of minnutes to expire before the agent can parse the file again. It is used to prevent Denial of Service attacks against cfagent.

domain = (giac.com) #Inform Cfengine about the default domain

policyserver = (sans-woody-01.giac.com) #Inform Cfengine which machine is the Cfengine file server.

cfservd_key = (root-192.168.1.202.pub) #Set this variable to look for the name of the key from the file server.

actionsequence = (copy) # Actionsequence defines which action and the order they are executed. This one only runs an action of copy.

classes: #Creating classes are Cfengine's style of making decisions. It uses them in place of "if-elseif-else" statements.
get_cfservd_key = (FileExists (/var/lib/cfengine2/ppkeys/${cfservd_key})) #This class is created if the cfservd key exsists in the named directory.

Copy: # The purpose of this action is to copy the latest cfagent.conf file. Both classes require the coping of the file to be encrypted. Hence the public keys need to be exchanged. If the key for the file server does not exsist, then the property for "trustkeys" is set to true. Cfagent will accept the key sent as the true key from the server. If the key already exsists, then it does not trust any new keys from the server. It only trusts the current key.

```
!get_cfservd_key::
        /var/cfengine2/configs/linux/debian/etc/cfagent.conf
        type=checksum
        dest=/etc/cfengine/cfagent.conf
        server=${policyserver}
        encrypt=true
        trustkey=true
get_cfservd_key::
        /var/cfengine2/configs/linux/debian/etc/cfagent.conf
        type=checksum
        dest=/etc/cfengine/cfagent.conf
        server=${policyserver}
        encrypt=true
        trustkey=false
```

Since update.conf is in place, the command cfagent can be run. This will make the client and server exchange the public keys and get the latest cfagent.conf file. This is what the following screen shot displays the files in the directories before cfagent is run.

```
supportwall001:~# ls /etc/cfengine/
update.conf
supportwall001:~# ls /var/lib/cfengine2/ppkeys/
localhost.priv  localhost.pub
supportwall001:~#
```

Only update.conf is in /etc/cfengine/ and public and private keys for the local host are in /var/lib/cfengine2/ppkeys/. Also the following output from lsof[7] shows the processes running by the default install.

7 List of Open Files (lsof) is not installed by default in Debian Woody. It's output is easier to read then using "ps" or "netstat" regarding which processes are running and are available over the network. Hence lsof was installed prior to running cfagent in this graphic. The rest of the output in this paper will be from a copy of lsof which was installed by the cfagent.conf file.

```
supportwall001:~# lsof -i
COMMAND    PID USER   FD   TYPE DEVICE SIZE NODE NAME
dhclient-  133 root    6u  IPv4     77       UDP *:bootpc
inetd      201 root    4u  IPv4    254       TCP *:discard (LISTEN)
inetd      201 root    5u  IPv4    255       UDP *:discard
inetd      201 root    6u  IPv4    256       TCP *:daytime (LISTEN)
inetd      201 root    7u  IPv4    257       TCP *:time (LISTEN)
inetd      201 root    8u  IPv4   3222       TCP *:smtp (LISTEN)
sshd      1494 root    4u  IPv4   5429       TCP *:ssh (LISTEN)
supportwall001:~#
```

By default some of the small services are installed. Also during the install the the mail server was not configured, but lsof still shows smtp running. Dhclient and ssh were chosen during the install so it is expected to see them listed.

To show that all running processes which believe their root directory is / (which is the default) run the command “lsof -d rtd”. The screen shot below confirms all running processes see / as their root directory.

```
dhclient-  133 root  rtd    DIR    8,1 4096    2 /
syslogd    190 root  rtd    DIR    8,1 4096    2 /
klogd      193 root  rtd    DIR    8,1 4096    2 /
inetd      201 root  rtd    DIR    8,1 4096    2 /
atd        230 root  rtd    DIR    8,1 4096    2 /
cron       233 root  rtd    DIR    8,1 4096    2 /
getty      237 root  rtd    DIR    8,1 4096    2 /
getty      238 root  rtd    DIR    8,1 4096    2 /
getty     1001 root  rtd    DIR    8,1 4096    2 /
getty     1002 root  rtd    DIR    8,1 4096    2 /
getty     1003 root  rtd    DIR    8,1 4096    2 /
bash      1005 root  rtd    DIR    8,1 4096    2 /
sshd      1494 root  rtd    DIR    8,1 4096    2 /
lsof      1602 root  rtd    DIR    8,1 4096    2 /
lsof      1603 root  rtd    DIR    8,1 4096    2 /
supportwall001:~#
```

After cfagent runs, we can see the public key (root-192.168.1.202.pub) for the Cfengine server has been received.

```
supportwall001:~# ls /var/lib/cfengine2/ppkeys/
localhost.priv  localhost.pub  root-192.168.1.202.pub
supportwall001:~#
```

Here are the changes after cfagent is finished parsing and acting on the update.conf and cfagent.conf files.

```
supportwall001:~# lsof -i
COMMAND   PID USER   FD   TYPE DEVICE SIZE NODE NAME
dhclient- 143 root    6u  IPv4     74       UDP *:bootpc
syslog-ng 196 root    3u  IPv4   1573       UDP *:syslog
sshd      215 root    3u  IPv4    271       TCP *:ssh (LISTEN)
ntpd      218 root    4u  IPv4    382       UDP *:ntp
ntpd      218 root    5u  IPv4    383       UDP supportwall001:ntp
ntpd      218 root    6u  IPv4    384       UDP 192.168.1.2:ntp
supportwall001:~#
```

All the small services are removed. Inetd is not invoked. Only dhclient, sshd, ntpd, syslog-ng, and the main program atftpd are running. So in running this cfagent policy file the only network available processes running are the ones needed for this server. The unneeded network available services have been shutdown and/or removed. Hence the cfagent.conf policy enforces one of the most important rules in computer security. “Run only necessary network available services.”

We can also verify that atftpd is running in a chroot()ed directory by using lsof. Below are the results of “lsof -d rtd”

```
bash      1005 root  rtd    DIR    8,1 4096     2 /
sshd      1494 root  rtd    DIR    8,1 4096     2 /
syslog-ng 1977 root  rtd    DIR    8,1 4096     2 /
atftpd    1993 root  rtd    DIR    8,7 4096 97761 /var/chroot/tftpd
ntpd      2039 root  rtd    DIR    8,1 4096     2 /
```

Notice the “root directory” for the other programs listed is “/”. But for atftpd, the root path is “/var/chroot/tfptd”. This shows that the cfagent policy was able to copy the needed files, devices, socket, libraries, directories, and to start the process chrooted. The cfagent.conf file will also check ps every time it runs. If it does not see atftpd running, it will attempt to restart the process in the same chrooted directory.

Now that the results of running the cfagent policy have been presented, the details of which actions were implemented by the cfagent policy to secure supportwall001 will be examined.

The cfagent.conf file is a configuration policy. It has a syntax which is simpler than most scripting languages. The syntax main purpose is to describe the desired configuration of a machine. Cfengine understands the details for each OS to configure the machine as per the policy. Specific lines from the cfagent.conf file which demonstrate the specific point will be listed. Appendix A lists the entire cfagent.conf policy file in context.

As with the update.conf file, the cfagent.conf file starts with the section named "control:". The relevant section of "control" is the actionsequence. This is the "plan of action" for the policy. It tells Cfengine which steps are needed to be taken to complete the policy. Below is the actionsequence for the cfagent.conf for the new machine.

```
control:

        actionsequence = (
                shellcommands.dmesg
                shellcommands.install_debs
                shellcommands.remove_debs
                tidy.deb_cleanup
                tidy.clean_temp
                copy.conf_files
                processes
                tidy.clean_temp
                directories.chroot_dirs
                copy.chroot_copy
                shellcommands.make_dev
                shellcommands.make_tftpuser
                files.chroot_files
                tidy.afterchroot
                processes.start_chroot_atftpd
                files.generic_monitoring
        )
```

Following the actionsequence, below are the steps implemented to secure supportwall001.

The action in the actionsequence is "shellcommands" then followed by its class name. This action "shellcommands" means to run some kind of command line program. The first class is "dmesg". As listed below the action and class are paired by a period. Hence it has a syntax of "<action>.<class>." Cfengine assists with security by refusing to run any shell command unless it has the full path. Each class defines the sequence of shellcommands executed by the policy.

shellcommands.dmesg

By default Debian Linux will post some types of errors on the console. This action will prevent errors from printing on the console by default. This action does not address any specific security issue. The code is

```
# Prevent Dmesg messages from outputing to the console
        dmesg::
                "/bin/dmesg -n 1"
```

shellcommands.install_debs

Here is where the policy defines which programs (using apt-get) should be installed. Cfengine does not understand apt-get as a native function. So the administrators had to create variables and classes for shellcommands to make the policy file know how to recognize which programs are installed, how to install them, and which programs to install. This action and the action to follow both rely on the program apt-show-versions. The script below checks if the apt-show-versions is installed. If not, Cfengine will install it and then proceed to parse the list variable to check for installed packages. Below is some of the related code.

```
#List variable naming the packages to install
install_package = ( apt-show-versions;lsof;syslog-ng;wget;atftpd;swatch;ntp-simple )
check_installed = ( "/usr/bin/apt-show-versions" )
apt_install = ( "/usr/bin/apt-get -y install" )

#Install the Debian packages listed in the variable install_package. Notice each install will write to syslog if
it installs a package.

install_debs::
        "$(check_installed) > /dev/null || $(apt_install) apt-show-versions | logger -t 'FCE Shell Says: '"
        "$(check_installed) | grep 'not installed' || $(apt_install) $(install_package) | logger -t 'CFE Shell
Says: '"
```

shellcommands.remove_debs

This is similar to the previous action, except it will remove known unwanted packages. The list variable contains names of packages the network administrators do not want installed on the machine. A feature of apt-get is to hold a cache of package names on the local machine. So if a new packages is available, the user needs to use “apt-get update” for the new package name. Hence all available package names can be located in the cache. So if other packages need to be removed, the administrators can investigate the names in the cache and add them to the list. The program apt-list-versions will also search this cache and report if the program is up to date, has an upgrade available, or is not installed. Hence the Cfengine action will take advantage of this status to know if the program needs to be uninstalled. The related code is below.

```
#Packages to remove. Note: Since this is 2.4 kernel ipchains is removed. Netfilter is included but not
enabled by default.
remove_package = ( portmap;nfs-common;ppp;pppoe;finger;telnet;telnetd;ipchains;exim )
apt_remove = ( "/usr/bin/apt-get -y remove" )

#Install the Debian packages listed in the variable install_package
        remove_debs::
                "$(check_installed) | grep -E \(upgradeable\|uptodate\) > /dev/null && $(apt_remove)
$(remove_package) | logger -t 'CFE Shell Says: '"
```

The action “tidy” is Cfengine's way to delete the files or directories.

It's main use in this policy is to keep the /tmp directory clean.

tidy.deb_cleanup

The main purpose behind the class deb_cleanup is to remove items the package installer never bothered to clean up. In this case it is the removal of the mail server exim. The related code is below.

```
deb_cleanup::
#All the directories and files in this class show that exim
#does not do a very good job at cleaning up after itself when
#it is uninstalled.
        /etc/ppp
        age=0
        pattern=*
        rmdirs=all
        recurse=inf

        /etc/cron.d/
        pattern=exim
        age=0

        /etc/cron.daily/
        pattern=exim
        age=0
```

tidy.clean_temp

The class clean_temp will delete whatever files may have been left in the temp directory. This helps if any of the packages leave files with information which should not be disclosed. Due to the similarity to tidy.deb_cleanup the code will not be listed below.

copy.conf_files

This action contacts the Cfengine files server and downloads the .conf files which have an impact on security. Some files can be replaced and will be in effect immediately. Other files need to have a command executed to load the contents of the .conf file. Cfengine provides the ability to issue the command based on the file download. The relevant code is shown below.

```
# The variable needed to remotely copy files
policyhost = ( 192.168.1.202 ) #The IP address of the cfengien file server.
dir_copy_etc = ( /var/cfengine2/configs/linux/debian/etc ) The location of the files to be copied.

$(dir_copy_etc)/sysctl.conf # The file to copy.
dest=/etc/sysctl.conf # The destination and the final name of the file.
mode=640 #The permissions for the file
encrypt=true # Encrypt  the transportation of the file.
type=checksum # Only copy the file if the MD5 checksum does not match.
server=$(policyhost) # Tell Cfengine to copy the file from the file server.
```

```
define=load_sysctl # If this file is copied, then enabel this class. The class will be executed in the next
round.
syslog=true # Report the actions for syslog

shellcommands:
##
load_sysctl:: # This class is enabled when a the file above is copied.
        "/sbin/sysctl -p" # This command tells sysctl to reload sysctl.conf.
```

The following names in italics are file names which are stored on the Cfengine server and replaced on the new machine as per the previous copy code. The differences in the files from the default files will be listed below the file name. Also comments (#) are listed to describe the impact on the security of the new machine.

---

sysctl.conf

```
# Log all packets which are Redirected, Spoofed or Source Routed
net/ipv4/conf/default/log_martians = 1
#Do not accept source routed packets
net/ipv4/conf/default/accept_source_route = 0
#Do not send redirects
net/ipv4/conf/default/send_redirects = 0
#Enable protection from spoofing
net/ipv4/conf/default/rp_filter = 1
#Prevent sending ICMP redirects
net/ipv4/conf/default/shared_media = 0
#Prevent accepting ICMP redirects
net/ipv4/conf/default/secure_redirects = 1
#Prevent accepting ICMP redirects
net/ipv4/conf/default/accept_redirects = 0
#Do not forward packets
net/ipv4/conf/default/forwarding = 0
#Prevent sending redirects
net/ipv4/conf/default/send_redirects = 0
#Do not ignore non RFC 1122 compliant responses.
net/ipv4/icmp_ignore_bogus_error_responses = 0
# Ignore ICMP broadcasts to prevent SMURF attacks
net/ipv4/icmp_echo_ignore_broadcasts = 1
# Do not ignore all icmp echo messages.
net/ipv4/icmp_echo_ignore_all = 0
```

issue, issue.net, motd

All three files are used to state a “Terms of Use” policy. The two issue files state the information before login. Issue is for the local login and issue.net is for remote login. Motd, is modified to issue the terms of use for after the login.

Both files generally state that use of this machine requires the user to understand the machine may be monitored. The two issue files request that any users who do not agree with the terms, should not login. The motd file requests any user who does not agree with the terms listed should logout immediately. The issue and motd files used are available appendix C and D.

inetd.conf

The is the file which controls the “super server”. By default all the software the administrators wanted to remove, was listed in inetd.conf (except atftpd was also listed in inetd.conf). If it was practical to use inetd, then the administrators would have configured the machine to use xinetd in it's place. Xinetd features better logging options and compiled support for tcp wrappers. But for this configuration, ssh, ntp and Syslog-NG came with their own scripts. Also dhclient is only installed for a temporary amount of time. The tftp server, atftpd install was originally setup with inetd. The administrators removed atftpd from inetd for two reasons: 1. The administrators wanted Cfengine to monitor the process so it could start the service atftpd on it's own terms, separate from other processes. 2. running the service from inetd/xinetd prevents lsof from listing the root path of the running program. Instead of showing atftpd from it's chrooted path, it only shows inetd from the default root path. Hence the chrooted service is easier to audit when it is running separate from inetd/xinetd. The file inetd.conf only contains comments to prevent it from starting services.

syslog-ng.conf

This .conf file controls the upgraded syslog server, Syslog-NG. The main changes to the configuration are the adding of the socket to / var/chroot/tftpd/dev/log and the configuration to allow the syslog to keep both local logs, listen for remote logs and forward all the logs to a separate syslog server.

ntp.conf

All servers need to keep synchronized accurate time. Keeping the time synchronized allows for easier auditing of the log files. Since remote logging is enabled, the central syslog needs to parse messages from different machines. If the clocks are set for different times, then retracing all the steps would be difficult. NTP allows all the servers to be set to the same time so log auditors can verify events in the logs with the correct order.

fstab

The fstab file tells the kernel how to mount the filesystems. As described in table 1.0, it contains options to limit what certain mounted filesystems can do. This increases the security of the machine by limiting the methods an attacker can use to trick a user into executing malware. The fstab file is available in appendix B.

sudoers

The sudoers file defines access for each account on the system. It can provide a user account the ability to run just three commands acting as the root account. It can allow a user to copy files acting as the UID of the tftpuser. The sudoers file also defines which accounts need to retype the password to use sudo. Another feature with sudo is logging every command typed to syslog. This allows the auditing of users actions during the users login. The sudoers fiel is available in appendix F

securetty

This file defines which physical attached ttys are allowed to login directly as root. In the case of this file, only tty12 is allowed. The Cfengine policy file will run the command “who” (which shows the logged in accounts opened tty) and pipe the results to syslog. The remote syslog parser (swatch) will alert that someone is logged in to tty12, so the administrators will be aware of direct root access. The main reason to leave a single local tty available for login is in case the sudoers file is corrupted, login options for root login are still possible.

sshd_config

This file defines the options the ssh daemon. The previous file limits the root login for the local machine. The direct root login for ssh is also restricted in this file. It is restricted with the option “PermitRootLogin no” Other settings chosen in this file are, “UsePrivilageSeparation yes”, “RhostsAuthentication no”, “PermitEmptyPasswords no”, “PasswordAuthetication yes”, “X11Forwarding no”, “banner /etc/issue.net”, and “PrintMotd yes”. All the options combine to restrict the login in the same manner as a local login. The sshd_config file is available in appendix E.

crontab

The Debian Linux style of cron uses a master file known as the crontab. Then in the directories /etc/cron.daily, /etc/cron.weekly, /etc/cron.monthly are jobs which only need to be run in those specific time frames. Hence if the job only needed to be run once a week, then place the shell script in the cron.weekly directory. If the

job needed to be run more often, then the file /etc/crontab needs to be edited. On this machine the cfagent command will be run every 15 minutes. This provides a potential attacker a 15 minute window. When cfagent is run at the end of 15 minutes, any policy violation will it will be noted in syslog.

inittab

This file affects the init process. The items of note in the file are disabling the ability to reboot the machine with a “ctrl+alt+del” and the requirement to type the root password for single user mode. The lines are listed below.

```
# Add “#” to the beginning of the line to disable “ctrl+alt+del”
#ca::ctrlaltdel:/sbin/shutdown -t3 -r now

#Require root password in single user mode
~~:S:wait:/sbin/sulogin
```

---

processes

The action processes matches a regular expression against the output of “ps”. The tutorial states , “the options sent to ps are "-aux" for BSD systems and "-ef" for system 5 “. Since Linux supports both styles of “ps”, the Cfengine debug shows that it favors the BSD style[8].

```
cfengine:supportwall001: Running process command /bin/ps auxw
cfengine:supportwall001: Signalled process 1494 (sshd) with SIGCONT
```

Cfengine attempts to match the process name and then take action based upon the success or failure to match the name of the process. Of the three processes in this action, two do not have classes. Without the classes both commands for sshd and syslog-ng will be executed. The relevant code is listed below.

```
"sshd" signal=cont restart "/etc/init.d/ssh start" syslog=true
"syslog-ng" signal=cont restart "/etc/init.d/syslog-ng start" syslog=true
```

Both processes have the same properties. The signal property, means if the Cfengine matches the expression, then it sends the assigned signal. The restart property tells Cfengine if the expression is not matched, then run the following command to start the process. Both actions will log the use of the properties to syslog. This allows any critical stopped services to be started again and also have syslog record the actions taken.

8 Note: It seems the tutorial needs to be updated. The PS command issued is “ps auxw”. The debug option is cfagent -v and it allows the command line options of the process to be listed.

tidy.clean_temp

Once again Cfengine cleans out the temp directory.

directories.chroot_dirs

This action will create the directories listed below if they are not created already. In this case they are just making the directory structure for chroot()ing the tftp service.

```
chroot_dirs::
#Setup the directories for chroot.
/var/chroot/tftpd/dev
/var/chroot/tftpd/etc/tftpd
/var/chroot/tftpd/lib
/var/chroot/tftpd/tftpboot
/var/chroot/tftpd/usr/lib
/var/chroot/tftpd/usr/sbin
/var/chroot/tftpd/var/run
/var/cfengine/
```

copy.chroot_copy

This action copies files from the source directories to the destination directories within the /var/chroot/tftpd/ directory. Since the network administrators of GIAC Enterprises find value in working with .deb packages from the Debian archive, they decided not to compile there own static binaries. They decided to stay with dynamic libraries. Hence the libraries, /etc/*.conf files, binaries and other supporting files all need to be copied into the correct directories. The decision to copy each file was based on extensive testing on lab machines. The files needed by atftpd was discovered by attempting to run atftpd chrooted while using strace to log the system calls. When the atftpd process died, the logs for strace would show the last file not found. Then that file would be copied to the chrooted directory and the test would start again. The advantage of this process is if security patches are issued for the libraries or the main binary, the ability to upgrade is very simple. The files copied includes a modified version of the passwd file, modified group file, and the tcp wrappers program with a version of hosts.allow and hosts.deny for use within the chrooted directory.

The modified version of the passwd file is

```
tftpuser:x:1001:100:tftp account:/dev/null:/bin/false
```

Since the only account listed in the passwd file in the chrooted directory is the

tftpuser account, it minimizes the disclosure of information if the tftp server were compromised. In the same manner the group file is also stripped of all the groups except the tftpgroup.

```
tftpgroup:x:1001
```

The next files are the hosts.allow and hosts.deny files. The atftpd service is precompiled with support for TCPwrappers. The TCPwrappers are a mechanism for access control to the specific service. The tcpd binary has been copied to the chrooted directory so separate rules can be applied to the TCPwrappers on the chrooted service separate from the TCPwrappers rules of the main OS. The relevant sections of the two files in the chroot()ed directory are listed below.

```
hosts.allow
in.tftpd : 192.168.1.1 #The address will be the inside nic of the local firewall

hosts.deny
ALL: paranoid
ALL : ALL
```

After the above files are copied, the files /etc/ld.so.cache, /usr/share/zoneinfo/America/Los_Angeles, and /etc/nsswitch.conf are copied from the local machine to the chrooted directories.

shellcommands.make_dev

To properly chroot a service, devices need to be created in /dev. This action will issue the shellcommands to create the devices “null” and “zero”.

```
make_dev.!test_chroot_null::
"/bin/mknod /var/chroot/tftpd/dev/null c 1 3"

make_dev.!test_chroot_zero::
"/bin/mknod /var/chroot/tftpd/dev/zero c 1 5"
```

The class of test_chroot_(null|zero) will only execute the code if the class returns “0”. In other words it only attempts to create the device if the device has not been created.

shellcommands.make_tftpuser

Next we check if the tftpuser account and tftpgroup group have been created. If not, then they are created.

```
make_tftpuser.!make_tftp_acct::
```

```
"/usr/sbin/groupadd tftpgroup"
"/usr/sbin/useradd -c 'account for the tftp server' -d /dev/null -s /bin/false -p ch@ng3A5aP tftpuser"
```

Next Cfengine will confirm the shell path for the account is always “/bin/false” so that the account can not return a shell.

```
make_tftpuser.!edit_tftp_sh::
"/usr/bin/chsh -s /bin/false tftpuser"
```

The classes edit_tftpa_sh and make_tftp_acct were defined with the following regular expressions and the following variable:

```
#############################################################
classes:
#############################################################
make_tftp_acct = ( Regcmp("tftpuser[[:print:]]+","${tftpuser_true}") )
edit_tftp_sh = ( Regcmp("tftpuser[[:print:]]+:/bin/false","${tftpuser_true}") )

tftpuser_true = ( ExecResult(/bin/sh -c "/bin/cat /etc/passwd | /bin/grep 'tftpuser'") )
```

files.chroot_files

Here is where Cfengine sets up the files and directory permissions for the chrooted directory. The goal is to give the least amount of privilege as possible. All the files are owned by root except for the contents of /tftpboot. The contents will be controlled by the process atftpd, which will use the account tftpuser and the group tftpgroup. The permissions are shown below.

```
chroot_files::

        /var/chroot/tftpd/.
        mode=510
        owner=root
        group=root
        action=fixall
        recurse=0
        syslog=true

        /var/chroot/tftpd/etc/.
        mode=440
        owner=root
        group=root
        action=fixall
        recurse=inf
        syslog=true

        /var/chroot/tftpd/dev/.
        mode=660
        owner=root
        group=root
```

```
action=fixall
recurse=inf
syslog=true

/var/chroot/tftpd/lib/.
mode=550
owner=root
group=root
action=fixall
recurse=inf
syslog=true

/var/chroot/tftpd/usr/.
mode=110
owner=root
group=root
action=fixall
recurse=inf
syslog=true

/var/chroot/tftpd/tftpboot/.
mode=770
owner=root
group=root
action=fixall
syslog=true

/var/chroot/var/.
mode=550
owner=root
group=root
recurse=inf
action=fixall
syslog=true
```

Note that all the permission modes end with “0”. The tftp server drops root and assumes the account tftpuser. If the server is compromised, the attacker would assume the account tftpuser. The permissions prevent tftpuser from doing anything within the chroot directory except for inside the directory /tftpboot. Hence the OS is protected by various layers of defenses should the tftp server become compromised.

tidy.afterchroot

When the files in the previous action were copied, Cfengine may also have copied empty subdirectories. This step removes the empty subdirectories. Below is a small sample of the code.

```
afterchroot::
```

```
/var/chroot/tftpd/usr/lib
pattern=*
age=0
rmdirs=sub
recurse=inf
filter=chrootremovedirs

############################################################
filters:
############################################################
{ chrootremovedirs
Type: "dir"
Result: "Type"
}
```

processes.start_chroot_atftpd

Here is where Cfengine starts the tftp service, in a chrooted environment, as a daemon, and drops root privileges in exchange for the user tftpuser.

```
start_chroot_atftpd::
"atftpd" signal=cont restart "/usr/sbin/chroot /var/chroot/tftpd /usr/sbin/atftpd --daemon --port 69 --user
tftpuser –verbose=7" syslog=true
```

Below is the result of a successful configuration where atftpd drops root privileges and assumes the UID[9] of tftpuser. Output of “ps aux” shows the process is owned by tftpuser.

```
tftpuser  6846  0.0  0.2  1548  756 ?        S    22:12   0:00 /usr/sbin/atftpd
tftpuser  6851  0.0  0.2  1548  756 ?        S    22:12   0:00 /usr/sbin/atftpd
```

Once Cfengine finishes, the pix firewall should be able to write it's configuration right to the tftp server. The result of a “write net” from the Cisco Pix shows the configuration was written.

```
Result of firewall command: "write net"

Building configuration...
TFTP write 'test-file' at 192.168.1.2 on interface 1
[OK]
```

9 The class workbook mentions that the source code of the program should not use seteuid when dropping root privileges. It should use setuid. To verify this I downloaded the source code from the Debian archive and grepped the files for both seteuid and setuid. Only setuid was found. It is located in the file tftpd.c.

```
setgid(group->gr_gid);
setuid(user->pw_uid);
/* Reopen log file now that we changed user, ...
```

Then the file written to the tftpserver shows the file has tftpgroup as the group and tftpuser as the owner.

```
supportwall001:/var/chroot/tftpd# ls -l tftpboot/
total 8
-rw-------    1 tftpuser tftpgrou     8131 Feb 15 22:35 test-file
supportwall001:/var/chroot/tftpd#
```

The chrooted /dev/log socket allows the tftp server to write to the main OS syslog. Here is the syslog entry written by atftpd with a successful load of the pix firewall configuration (by running “write net” on the Pix firewall) to the tftp server.

Feb 16 16:49:11 supportwall001 tftpd[10596]: Fetching from 192.168.1.1 to test-file
Feb 16 16:49:11 supportwall001 tftpd[10596]: received WRQ <filename: test-file, mode: octet, >
Feb 16 16:49:11 supportwall001 tftpd[10596]: sent ACK <block: 0>
Feb 16 16:49:11 supportwall001 tftpd[10596]: received DATA <block: 1, size: 512>
....<snip>...
Feb 16 16:49:11 supportwall001 tftpd[10596]: received DATA <block: 16, size: 451>
Feb 16 16:49:11 supportwall001 tftpd[10596]: sent ACK <block: 16>
Feb 16 16:49:11 supportwall001 tftpd[10596]: Server thread exiting

files.generic_monitoring

While Cfengine is not a total replacement for Tripwire, it's file integrity features are a bonus. Since the cfagent is already monitoring the system, it may as well report if the files used in the shellcommands action and some other standard auditing tools can be trusted. If at any time Cfengine writes an error regarding one of the files, the administrators can then start to run a comparison using the full file integrity tool and/or binary comparisons with files from a baseline backup. Below is a sample of some of the files monitored. To see a complete list, find the cfagent.conf file sample in appendix A.

```
generic_monitoring::
#Below are files we wish to monitor the integrity. If these are compromised we are in trouble

/usr/sbin/chroot
mode=550
owner=root
group=root
action=fixall
recurse=0
syslog=true
checksum=md5
define=

/usr/sbin/cfagent
```

```
mode=550
owner=root
group=root
action=fixall
recurse=0
syslog=true
checksum=md5

/usr/sbin/lsof
mode=550
owner=root
group=root
action=fixall
recurse=0
syslog=true
checksum=md5
```

After the cfagent.conf is finished parsing, the following manual steps need to be completed.

1) Change the password of the tftpuser account. While still logged in as root, run "passwd tftpuser". Type a strong[10] password when prompted.
2) Create the account useradmin.
   "useradd -c 'admin account' -d /home/useradmin -s /bin/sh useradmin"
   Then change the account's password, "passwd useradmin". Type a strong password when prompted.
3) Reboot the machine. This allows the /etc/fstab file to be mounted and will show if any problems with startup scripts, .conf files or the running of cfagent have taken place. When the BIOS settings come up, edit the settings to enter a password. The pasword will now be requested to edit the BIOS settings in the future. Then document the settings for future reference. When the reboot is finished, the network administrators should be able to login as useradmin. The sudoers file allows the account useradmin to run commands as root without an extra login.
4) Configure lilo.conf [11] to use a password. Just uncomment the line "password = <some password>" and replace the "<some password>" with a password. Be sure to document the password for future use. Then type /sbin/lilo on the command line to make the changes take effect.
5) Edit the file /etc/network/interfaces to reflect the following changes:
   #iface eth0 inet dhcp
   iface eth0 inet static

10 Settings in the PAM configuration allow for enforcing a password policy. There was a problem with getting this working correctly on the Woody distribution. Since the accounts are mainly handled by administrators, it was decided to not use the PAM policy and enforce regular password audits as motivation for strong passwords.

11 GRUB is a newer bootloader, but is not the default bootloader for Debian Woody. Since the needs of this machine are modest, it was decided to leave the default boot loader for this machine.

address <*.*.*.* The real local network address>
netmask 255.255.255.0
broadcast <*.*.*.255>
gateway <*.*.*.* The gateway address>

6) Edit the IP address in the chrooted host.allow file to the ip address of the new firewall.
7) Remove the DHCP client dhclient, “apt-get remove dhclient”
8) Install Samhain. This program is a centrally managed file integrity management program. From this point all future changes will be handled with Samhain acting as a change control management tool.
9) Shutdown the machine and move it to the real network.
10) Test the remote syslog by running “logger -t 'test label' 'This works' ”. Then grep the logs on the central syslog for this message. Make changes where needed.
11) Run a security update. To make sure only security related files will be downloaded, edit the /etc/apt/source.list file and comment out all lines except, “deb http://security.debian.org/ stable/updates main contrib non-free”
Then run “apt-get update” and then “apt-get upgrade”. If any programs are updated, use Samhain to check which files have changed. If any are .conf files, scp a copy to the central Cfengine server. See if any of the settings in the new .conf file removes any of the security changes made. If so, edit the new files to contain the settings. Edit the cfagent.conf file to accept new hashes. Run cfagent to download and update the files. Rerun Samhain to update the latest hashes. Reedit the cfagent.conf to not accept new hashes. The whole point behind this exercise is to set the machine from a starting baseline. Now that all the files have been changed and the current hashes have been made, all future changes can be managed with a change control process. The only expected future changes are security updates, changes in the /etc/network/interface file when the machine gets to the EPT store and changes to the passwords.

Audit Plan

The described TFTP server configuration builds a foundation for the future. It is able to do so by having a documented baseline. The baseline is measured by the cfagent.conf file, copies of the /etc files and the file hashes kept in Samhain. The cfagent.conf file can be used two different ways to audit the system. First, the normal cfagent.conf, which runs every 15 minutes from cron, sends information about changes it makes to syslog. Hence Cfengine is constantly auditing the machine configuration. Secondly, the cfagent.conf file can be used to audit the system by copying the file and edit it for the following:

- Add comments about the date, purpose and use of this file in the cfagent.conf files
- Search and replace all the “action=fixall” to “action=warnall” or “action=warn”

pending syntax for the action
- Disable the classes to restart the services to reread the conf file
- Comment out the current cfservd server and use the same variable to a test cfservd server
- Sign the file with a GnuPG key to verify future integrity of the file.
- Add the newly edited file and the /etc files to a versioning system in this case the versioning system is Subversion

The steps listed allow the administrator to pull the file from subversion, place it in the current cfservd[12] server. The administrators also copy the /etc files and place them on the test cfservd server. Then when the client machine picks up it's latest cfagent.conf it will send warnings to syslog for all the items which are different than the original policy. This action allows the administrators to find if the original file triggers new alerts vs. the current cfagent.conf file. It will also show what configuration may have changed since the original setup. If changes are found, cfagent.conf file and the archived /etc files have all the information needed to evaluate if the files in the versioning system should be updated and saved as new versions for future baselines or if a problem/compromise has occurred.

Below are some lines in the syslog which show that cfagent.conf found items not within the policy running the edited baseline cfagent.conf file.

```
#Format under “action=fixall”
Feb 20 09:40:42 supportwall001 cfengine:supportwall001[7490]: Object /usr/sbin/lsof had permission 755, changed it to 550

#Format under “action=warnall”
Feb 20 09:34:33 supportwall001 cfengine:supportwall001[7490]:  /usr/sbin/lsof has permission 755
Feb 20 09:34:33 supportwall001 cfengine:supportwall001[7490]:  [should be 550]
```

The lines above show that in either case if the action is “fixall” or “warnall” Cfengine will record the event. On the remote syslog server, Swatch is monitoring such events. The administrators created signatures for Swatch to match the error messages Cfengine may generate. The Swatch signatures to match the above error messages is below.

```
watchfor /[[:print:]]+cfengine:[[:print:]]+\[[[:digit:]]+\]:[[:print:]]*\/usr\/sbin\/(lsof|tcpd) (has|had) permission [[:digit:]]{3}/
echo blue

watchfor /(changed it to [[:digit:]]{3}|should be [[:digit:]]{3})/
echo blue
```

The first signature matches the messages stating the directory path and the file

12 This process can also be reversed by updating the cfagent.conf file on the test server to the original cfservd server variable. Hence the recommendation to comment out the original variable.

"had" or "has" permission. The next permission error matches the follow up message given if the action is "warnall". Below is an example of Swatch matching the signatures.

```
Feb 20 10:54:25 sans-woody-02 cfengine:sans-woody-02[7979]:  /usr/sbin/lsof has
permission 777
Feb 20 10:54:25 sans-woody-02 cfengine:sans-woody-02[7979]:  [should be 550]
Feb 20 10:55:34 sans-woody-02 cfengine:sans-woody-02[7992]: Object /usr/sbin/lso
f had permission 777, changed it to 550
```

The network administrators invested time to build regular expressions to match violated policy with Cfengine. Hence under most cases where Cfenigne may report a policy violation the administrators can be properly notified.

Since the cfagent.conf file only tracks a limited number of files with it's own hashes and only measures the configuration of the most critical parts of the system, Samhain is also used to build a baseline file audit of the machine. Samhain is a Tripwire style integrity checker. The databases can prevent tempering by being signed with GnuPG keys. The administrators will take advantage of this feature to be assured the data can be trusted. Since one of the last steps was to take a snapshot of the entire machine, the database created will also be copied and entered into Subversion. This allows for a record of all the original file hashes to be measured against the current machine's file in case the current database on the compromised machine can not be trusted due to corruption or an attacker's actions. Samhain can log to syslog, so Syslog-NG will forward the logs to the remote log server. When the network administrators created the swatch expression file, they also included expressions for swatch. Below is an expression to match an error if the file changes.

```
watchfor /path=<[[:print:]]+>[[:print:]]+chksum_old[[:print:]]+chksum_new[[:print:]]+/
echo blue
```

Here is the output of swatch successfully matching the errors generated for a changed file then sent to the remote syslog.

```
CRIT   :  [2005-02-20T22:17:53-0800] msg=<POLICY [ReadOnly]>, path=</etc/sysctl.
conf>, size_old=<648> size_new=<651> ctime_old=<[2005-02-02T19:00:16]>, ctime_ne
w=<[2005-02-21T04:49:54]>, mtime_old=<[2005-02-02T04:49:32]>, mtime_new=<[2005-0
2-21T04:49:54]>, chksum_old=<8A57038B01EF92D2CF698E310EDE83DE06292AB275EA40E5>,
chksum_new=<7E4598FBA074C6D14CEC6F8962AB03B4BDCC873702F46A78>,
```

Another task in the audit is to test the output of "lsof -i" (the listening network services) against the results of a port scanner. Before the task is run, tcpd in the chrooted directory will be edited by adding the scanning machine.
Then two scans will be run: one normal UDP scan and one normal TCP scan.

The UDP scan
nmap -sU -O -P0 -T Normal -n 192.1.2 -p 1-1024
(The 1022 ports scanned but not shown below are in state: closed)
Port State Service
69/udp open tftp
123/udp open ntp

The TCP scan
nmap -sT -O -P0 -T Normal -n 192.168.1.2
(The 1553 ports scanned but not shown below are in state: closed)
Port State Service
22/tcp open ssh

So a nmap scan confirms the output of "lsof -i". Another benefit of this test is to check the syslog to see if tcpd in the chrooted directory logged this event when the address of the nmap machine is denied.

Feb 21 10:03:17 supportwall001 tftpd[2019]: Connection refused from 0.0.0.0
Feb 21 10:03:17 supportwall001 tftpd[2019]: Server thread exiting

The audit confirms the new service is working properly, atftpd can log to syslog, the chrooted version of tcpd can log to syslog, the service drops root permissions, the permissions for files and directories are monitored and enforced, the final step is to run an image backup. The network administrator brings out a boot disk for a program named "g4u" (Ghost for Unix). The machine is rebooted to this disk and it runs DD on the drive sending the data to another tftp server. This image will be stored on tape. If the machine becomes compromised in a manner in which the FBI becomes involved, we can use DD to reimage the compromised machine and deliver both a before image and an after image. The FBI can compare the image for both the forensic investigation and to present evidence in court if needed. The audit covers tasks which confirm the proper function of the new service and the future possibility of legal action.

Ongoing Maintenance Plan

As described in the audit plan, the maintenance plan will also use Cfengine and Samhain to check for security problems. In addition to the previously stated features of Cfengine, Cfengine will also check for the following issues:

Current setuid/setgid files. Of the files and directories listed in the various actions in Cfengine, it will log which files are setuid to the file /var/log/setuid.today[13]. The files are copied using scp. Currently a network administrator is looking over the logs to find patterns to match for an updated version of the swatch expression file.

13 The log is rotated. Hence the files are designed in the following manner: setuid.today, setuid.yesterday, setuid.changes (shows diffs of the changes), then all older logs are setuid.changes<number>.

Cfengine also has a feature to find hidden non-alphanumeric names.

NonAlphaNumFiles = (on)

The best description of the feature is from the Cfengine reference guide, section 4.9.43.

> *If enabled, this option causes Cfengine to detect and disable files which have purely non-alphanumeric filenames, i.e. files which might be accidental or deliberately concealed. The files are then marked with a suffix* `.cf-nonalpha` *and are rendered visible.*

Cfengine will also check for suspicious file names in directories it is already monitoring. In the “control:” section the action “SuspiciousNames = (<list of names>)” will send warnings to syslog if it finds a file which matches a name in the list. In addition, this feature watches for file names which are hidden by just containing spaces or dots. The names in the list may include rootkits, popular scanning and cracking tools, etc. While the list is limited to literal names and not regular expressions, the list will have TARed names of root kits listed on PacketStorm. This may catch any recently downloaded rootkits. These two features increase security maintenance by proactively looking for the tools and methods an attacker may use and sending the information to syslog for an administrator to verify.

Similar to CFEngine, Samhain also has extra features to assist in maintaining the security of the system. It will also monitor setuid files. It will look for new kernel module rootkits, and it will monitor login/logout events.

It is realized that supportwall001's security is largely dependent upon the syslog server working. Hence each week a test of the syslog server is run by adding to the cfagent.conf file to run the shell command “/usr/bin/logger ${host} 'is working'”. The working syslogs report the following line to the local syslog “<machine name> is working”. The log entry will make it's way back to the central syslog. The entrys will be enumerated as the machines with a working configuration. Machines which did not report as working will be investigated.

Once a week the rotated log files on the specific machines are scp and periodically checked against the central logs. The administrators also confirm that the machine does not have firewall images or configurations sitting on the machine since that information should be kept on the central server, netman004. Since the installation of the machine can be created in a short time, the machine is not backed up on a regular basis. It is only backed up as an image if the configuration changes. If the configuration does change, then just like the end of the installation, a DD image is created and saved for future forensic purposes.

As part of regular network security maintenance, the administrators run network vulnerability scans using Nessus, port scans with nmap and promiscuous mode scans with Sentinel.

Summary

While the current installation plan builds a secure and auditable Linux machine, more can be done to improve security. The following areas should be considered for improvement: hardening the os with kernel patches, improve syslog event tracking and reporting, and adopt a best practices methodology to measure the overall success of “Confidentiality, Integrity, Availability (CIA)” .

Kernel patches such as SElinux and GRsecurity add new functions and utilities to the Linux kernel. Both patches support Mandatory Access Control (MAC), where a policy defines which resources a program may use. This is a finer grain control mechanism over standard permissions. Both patches can allow one program to access a resource while denying another program the same resource. Both patches have removed the need for setuid and setgid program. GRsecurity brings additional features which apply to the goals of GIAC enterprises. GIAC enterprises desires to have the ability to log any event. GRsecurity adds greater enforcement of chrooted environments. It restricts actions such as no chroot within a chroot and if /proc is mounted, within the chroot it still prevents viewing any processes which are not contained within the chroot. By restricting the listed actions, GRsecurity removes the normal weaknesses of chrooting. It also has the ability to log all executable actions within a chrooted environment. So if the chrooted service is not used very often (just like the tftp server), finding an executable running in the chrooted directory would be a very interesting event to investigate. GRsecurity also supports logging denied resources, mounting and unmounting, signals, failed forks, IPC creation & removal, and change in time. Like tcpwrappers, it also supports logging the ip address of the logged events sent from a remote source. So not only does GRsecurity increase the ability to enforce restrictions, it increases the ability to audit the machine. This compliments the logging infrastructure of GIAC enterprises. The most sweeping improvement with GRsecurity is it's inclusion of PAX. This feature greatly reduces the ability of an attacker to compromise a machine with remote buffer overflows. It does so by adding randomization to the stack and a “canary” value to prevent execution if the “canary” is accessed instead of the actual address.

While a lot of effort has been put into building a logging infrastructure to a central location for GAIC enterprises, the only event monitoring at this point is running swatch and daily reading of logs. More effort in this next year will be spent on correlating the events in the syslog. Some of the open source tools to be evaluated are Simple Event Correlator (SEC), fwlogwatch, and Prelude Log

Monitoring Lackey (PLML). Each has different strengths. SEC is a good replacement for the general log monitoring of swatch. It looks not only for specific events, but can be instructed to match events with other events. This adds to the power of log monitoring by adding some intelligence beyond simple matching. fwlogwatch is designed for firewall and Snort logs. Fwlogwatch is not good for general syslog but it can parse specific logs for the cisco firewalls and the snort sensors. PLML can work with general syslog and also parse application specific log messages. It can parse SSH, GRSecurity, and various UNIX specific logs. The network administrators will investigate how to best combine all three open source tools since each has it's own unique strength. They plan to explore proprietary vendors' options to see if their programs can match the features of the three open source tools.

The network administrators are investigating the IT Infrastructure Library (ITIL) for a set of best practices by which to measure their success. Topics in ITIL covers managing Incident, Service Level, Change, and Availability. The network administrators will start by reading the book, *The Visible Ops Handbook: Starting ITIL in 4 Practical Steps*[14] This will give them the information to evaluate the current practices and look for steps to improve them. Even if GIAC Enterprises does not adopt the entire library, they will improve their evaluation and auditing by embracing the concepts in this book.

GIAC enterprises continues to search for ways to enhance the Confidentiality, Integrity and Availability (CIA) of the network infrastructure. The future security enhancements of applying kernel patches, improving syslog tracking and reporting, and adopting network best practices will build upon GIAC Enterprises existing foundation. The corporation values their ability to manage risk by hardening the OS from installation, using automated policy to configure and audit the OS, and monitoring the integrity of the OS.

14 Authored by Kevin Behr, Gene Kim, and George Spafford, copyright June 2004.

List of References
Web Sites

Zdziarski, Jonathan A. “Chrooting daemons and system processes HOW-TO.” *Nuclear Elephant.com*. 12 Mar. 2003. 1 Nov. 2004 <http://www.nuclearelephant.com/papers/chroot.html>.

Friedl, Steve. "Best Practices for UNIX chroot() Operations." *Unixwiz.net*. Pre Nov. 2004, 1 Nov. 2004. <http://www.unixwiz.net/techtips/chroot-practices.html>.

Bauer, Kirk. “Automating Security with GNU Cfengine.” *Linux Journal*. 5 Feb. 2004, 1 Nov. 2004. <http://www.linuxjournal.com/article/6848>.

Burgess, Mark. "Cfengine-Tutorial". *GNU Cfengine*. Pre. Dec 2004, Dec 10 2004 <http://www.cfengine.org/docs/cfengine-Tutorial.html>.

Burgess, Mark. "Cfengine-Reference". *GNU Cfengine*. Pre. Dec 2004, Dec 10 2004 <http://www.cfengine.org/docs/cfengine-Reference.html>.

Seifried, Kurt "Linux Administrator's Security Guide: Installation". *Kurt Seifried Information Security*. 31 Aug. 2001, 14 Jan. 2005 <http://www.seifried.org/lasg/installation/>.

"Securing Debian Manual". *Debian.org*. 23 Jan. 2005, 25 Jan 2005 <http://www.nl.debian.org/doc/manuals/securing-debian-howto/>.

Andreasson, Oskar. "Ipsysctl tutorial 1.0.4." *Frozen Tux.* 2002, 17 Dec. 2004 <http://ipsysctl-tutorial.frozentux.net/ipsysctl-tutorial.html>.

Campi, Nathan. "Syslog-ng FAQ" *Campin dot Net.* Pre. Dec. 2004, Dec 19 2004 <http://www.campin.net/syslog-ng/faq.html>

Wichmann, Rainer. "Samhain manual" *Samhain Labs*. Pre. 2004, Feb 3 2005 <http://la-samhna.de/samhain/manual/>

“The Twenty Most Critical Internet Security Vulnerabilities” *SANS Institute*. 2004 Oct. 8, 2004 Nov. 3 <http://www.sans.org/top20/>

Books

Brotzman, Lee E., Ranch, David A., et al. *Securing Linux Step-By-Step*. Version

1.0. Bethesda: Sans Institute, 2000.

Nemeth, Evi, Garth Snyder, Scott SeeBass, Trent R. Hein, et al. *Unix System Administration Handbook*. 3rd ed. Upper Saddle River: Prentice-Hall PTR, 2001

Garfinkel, Simson and Gene Spafford. *Practical Unix & Internet Security* 2nd ed. Sebastopol: O'Reilly & Associates, 1996

Appendix A

cfagent.conf

```
control:

        #This improves general security of the OS.
        NonAlphaNumFiles = ( on )
        SuspiciousNames = ( 0x333openssh-3.7.1p2.tar.gz 4553-invader-2.1.1.tar.gz n-du.tgz doorman-
0.8.tgz )


        #This is to prevent DDOS if set higher then "0"
        IfElapsed = ( 0 )

        #Set classes and variables for the whole file
                domain = ( giac.com )
                Syslog = ( on )
                Split  = ( ; ) #using a semicolon to prevent reading the passwd file as a list

        ##Prepares cfengine create the classes on demand
        AddInstallable = (
                load_sysctl
                load_inittab
                load_lilo
                load_inetd
                load_sshd
                load_syslog_ng
                load_inittab
                start_ntp
                warn_files
                )

        # The variable needed to remotely copy files
        policyhost = ( 192.168.1.202 )
        dir_copy_etc = ( /var/cfengine2/configs/linux/debian/etc )

        #Set package install and remove shellcommands:
        check_installed = ( "/usr/bin/apt-show-versions" )
        app_path_remove = ( "/usr/bin/which" )
        apt_remove = ( "/usr/bin/apt-get -y remove" )
        apt_install = ( "/usr/bin/apt-get -y install" )
        install_package = ( apt-show-versions;lsof;syslog-ng;wget;atftpd;swatch;ntp-simple;gnupg;sudo )
        remove_package = ( portmap;nfs-common;ppp;pppoe;finger;telnet;telnetd;ipchains;exim )

        #Set variables for chrooting
        chroot_dir = ( "/var/chroot/tftpd/" )
        tftpuser_true = ( ExecResult(/bin/sh -c "/bin/cat /etc/passwd | /bin/grep 'tftpuser'") )

        #Build a database for checksums
        ChecksumDatabase = ( /var/cfengine/cfdb )
```

```
        ChecksumPurge = ( on ) # Leave on to confirm missing files. It will syslog warning if file is
missing
        ChecksumUpdates = ( on ) #Chaneg to off after the install.

        actionsequence = (
                shellcommands.dmesg
                shellcommands.install_debs
                shellcommands.remove_debs
                tidy.deb_cleanup
                tidy.clean_temp
                copy.conf_files
                processes
                tidy.clean_temp
                directories.chroot_dirs
                copy.chroot_copy
                shellcommands.make_dev
                shellcommands.make_tftpuser
                files.chroot_files
                tidy.afterchroot
                processes.start_chroot_atftpd
                files.generic_monitoring
        )

############################################################
classes:
############################################################
        test_chroot_null = ( ReturnsZero(/bin/sh -c "/bin/ls /var/chroot/tftpd/dev/null > /dev/null") )
        test_chroot_zero = ( ReturnsZero(/bin/sh -c "/bin/ls /var/chroot/tftpd/dev/zero > /dev/null") )
        make_tftp_acct = ( Regcmp("tftpuser[[:print:]]+","${tftpuser_true}") )
        edit_tftp_sh = ( Regcmp("tftpuser[[:print:]]+:/bin/false","${tftpuser_true}") )


############################################################
alerts:
############################################################
        load_sysctl::
        "load_sysctl was run. sysctl.conf must have changed"

        load_inittab::
        "load_inittab was run. inittab must have been changed"

        load_lilo::
        "load_lilo was run. lilo.conf must have been changed"

        load_inetd::
        "load_inetd was run. inetd.conf must have been changed"

        load_syslog_ng::
        "load_syslog_ng was run. syslog-ng.conf must have been changed"

        load_sshd::
        "load_sshd was run. sshd_conf must have been changed"
```

```
        load_inittab::
        "load_inittab was run. inittab must have been changed"

        start_ntp::
        "start_ntp was run. ntp.conf must have been changed"

        warn_files::
        "one of the files the in class generic_monitoring changed. Please investigate"

############################################################
shellcommands:
############################################################

# Prevent Dmesg messages from outputing to the console
        dmesg::
                "/bin/dmesg -n 1"


#Install the Debian packages listed in the variable install_package
        install_debs::
                "$(check_installed) > /dev/null || $(apt_install) apt-show-versions | logger -t 'FCE Shell
Says: '"
                "$(check_installed) | grep 'not installed' || $(apt_install) $(install_package) | logger -t 'CFE
Shell Says: '"

#Install the Debian packages listed in the variable install_package
        remove_debs::
                "$(check_installed) | grep -E \(upgradeable\|uptodate\) > /dev/null && $(apt_remove)
$(remove_package) | logger -t 'CFE Shell Says: '"

        load_sysctl::
                "/sbin/sysctl -p"

        load_inittab::
                "/sbin/init q"

        load_lilo::
                "/sbin/lilo"

        load_inetd::
        "/bin/kill -HUP `/usr/sbin/lsof -t -c inetd`"

        load_syslog_ng::
        "/etc/init.d/syslog-ng restart"

        load_sshd::
        "/etc/init.d/ssh restart"

        load_inittab::
        "init q"
```

```
        #syslog_who_last::
        #       "who --login -u --mesg | logger -t 'who command says: '"
        #       "last | logger -t 'last command says: '"

        start_ntp::
        "/etc/init.d/ntp restart"

        make_dev.!test_chroot_null::
        "/bin/mknod /var/chroot/tftpd/dev/null c 1 3"

        make_dev.!test_chroot_zero::
        "/bin/mknod /var/chroot/tftpd/dev/zero c 1 5"

        make_tftpuser.!make_tftp_acct::
        "/usr/sbin/groupadd tftpgroup"
        "/usr/sbin/useradd -c 'account for the tftp server' -d /dev/null -s /bin/false -p ch@ng3A5aP tftpuser"
        "/bin/echo we ran make_tftp_acct. The variable tftpuser_true is ${tftpuser_true}"

        make_tftpuser.!edit_tftp_sh::
        "/usr/bin/chsh -s /bin/false tftpuser"
        "/bin/echo we ran edit_tftp_sh"

############################################################
processes:
############################################################

        "sshd" signal=cont restart "/etc/init.d/ssh start" syslog=true

        "syslog-ng" signal=cont restart "/etc/init.d/syslog-ng start" syslog=true

        start_chroot_atftpd::
        "atftpd" signal=cont restart "/usr/sbin/chroot /var/chroot/tftpd /usr/sbin/atftpd --daemon --port 69 --
user tftpuser --verbose=7" syslog=true


############################################################
tidy:
############################################################
deb_cleanup::
#All the directories and files in this class show that exim
#does not do a very good job at cleaning up after itself when
#it is uninstalled.
        /etc/ppp
        age=0
        pattern=*
        rmdirs=all
        recurse=inf

        /etc/cron.d/
        pattern=exim
        age=0
```

```
        /etc/cron.daily/
        pattern=exim
        age=0

        /etc/rc0.d/
        pattern=*exim
        age=0

        /etc/rc1.d/
        pattern=*exim
        age=0

        /etc/rc2.d/
        pattern=*exim
        age=0

        /etc/rc3.d/
        pattern=*exim
        age=0

        /etc/rc4.d/
        pattern=*exim
        age=0

        /etc/rc5.d/
        pattern=*exim
        age=0

        /etc/rc6.d/
        pattern=*exim
        age=0

        /etc/init.d/
        pattern=exim
        age=0

clean_temp::
        /tmp
        pattern=*
        recurse=inf
        age=0

        /var/tmp
        pattern=*
        recurse=inf
        age=0

        afterchroot::

                /var/chroot/tftpd/usr/lib
                pattern=*
                age=0
```

```
                rmdirs=sub
                recurse=inf
                filter=chrootremovedirs

        /var/chroot/tftpd/lib
                pattern=*
                age=0
                rmdirs=sub
                recurse=inf
                filter=chrootremovedirs

        /var/chroot/tftpd/usr/sbin
                pattern=*
                age=0
                rmdirs=sub
                recurse=inf
                filter=chrootremovedirs

        /var/chroot/tftpd/etc
                pattern=*
                age=0
                rmdirs=sub
                recurse=inf
                filter=chrootremovedirs
############################################################
filters:
############################################################
{ chrootremovedirs

Type: "dir"
Result: "Type"
}


############################################################
copy:
############################################################

conf_files::
#These are various files which affect the security of the system.

        $(dir_copy_etc)/sysctl.conf
        dest=/etc/sysctl.conf
        mode=640
        encrypt=true
        type=checksum
        server=$(policyhost)
        define=load_sysctl
        syslog=true


        $(dir_copy_etc)/issue
```

```
dest=/etc/issue
mode=600
owner=root
group=root
encrypt=true
type=checksum
server=$(policyhost)
syslog=true


$(dir_copy_etc)/issue.net
dest=/etc/issue.net
mode=600
owner=root
group=root
encrypt=true
type=checksum
server=$(policyhost)
syslog=true


$(dir_copy_etc)/inetd.conf
dest=/etc/inetd.conf
mode=644
owner=root
group=root
encrypt=true
type=checksum
server=$(policyhost)
define=load_inetd
syslog=true

$(dir_copy_etc)/motd
dest=/etc/motd
mode=640
owner=root
group=root
encrypt=true
type=checksum
server=$(policyhost)
syslog=true

$(dir_copy_etc)/syslog-ng.conf
dest=/etc/syslog-ng/syslog-ng.conf
mode=640
encrypt=true
type=checksum
server=$(policyhost)
define=load_syslog_ng
syslog=true

$(dir_copy_etc)/ntp.conf
```

```
dest=/etc/ntp.conf
mode=640
encrypt=true
type=checksum
server=$(policyhost)
define=start_ntp
syslog=true

$(dir_copy_etc)/fstab
dest=/etc/fstab
mode=600
owner=root
group=root
encrypt=true
type=checksum
server=$(policyhost)
syslog=true

$(dir_copy_etc)/sudoers
dest=/etc/fstab
mode=600
owner=root
group=root
encrypt=true
type=checksum
server=$(policyhost)
syslog=true

$(dir_copy_etc)/securetty
dest=/etc/fstab
mode=600
owner=root
group=root
encrypt=true
type=checksum
server=$(policyhost)
syslog=true

$(dir_copy_etc)/sshd_config
dest=/etc/sshd/sshd_config
mode=600
owner=root
group=root
encrypt=true
type=checksum
define=load_sshd
server=$(policyhost)
syslog=true

$(dir_copy_etc)/inittab
dest=/etc/inittab
mode=600
```

```
owner=root
group=root
encrypt=true
type=checksum
define=load_inittab
server=$(policyhost)
syslog=true

chroot_copy::

$(dir_copy_etc)/chroot_passwd
dest=$(chroot_dir)etc/passwd
mode=640
encrypt=true
type=checksum
server=$(policyhost)
syslog=true
backup=false

$(dir_copy_etc)/group
dest=$(chroot_dir)etc/group
encrypt=true
type=checksum
server=$(policyhost)
syslog=true
backup=false

$(dir_copy_etc)/hosts.allow
dest=$(chroot_dir)etc/hosts.allow
encrypt=true
type=checksum
server=$(policyhost)
syslog=true
backup=false

$(dir_copy_etc)/hosts.deny
dest=$(chroot_dir)etc/hosts.deny
encrypt=true
type=checksum
server=$(policyhost)
syslog=true
backup=false

/etc/ld.so.cache
dest=$(chroot_dir)etc/ld.so.cache
type=checksum
syslog=true

/usr/share/zoneinfo/America/Los_Angeles
dest=$(chroot_dir)etc/localtime
type=checksum
syslog=true
```

```
        /etc/nsswitch.conf
        dest=$(chroot_dir)etc/nsswitch.conf
        type=checksum
        syslog=true


#Think about using a filter here instead of an include
        /lib/
        dest=$(chroot_dir)lib/
        type=checksum
        syslog=true
        include=ld-*
        include=libc-*.so
        include=libc.so.*
        include=libnsl*
        include=libnss_compat*
        include=libnss_files*
        include=libnss_nis*
        include=libpthread*
        include=libwrap*
        recurse=1

        /usr/lib/
        dest=$(chroot_dir)usr/lib/
        type=checksum
        syslog=true
        include=libpcre.so.*
        recurse=1

        /usr/sbin/
        dest=$(chroot_dir)usr/sbin/
        type=checksum
        syslog=true
        include=atftpd
        include=in.tftpd
        include=tcpd
        recurse=1



############################################################
directories:
############################################################
        chroot_dirs::
#Setup the directories for chroot.

        /var/chroot/tftpd/dev

        /var/chroot/tftpd/etc/tftpd
```

```
        /var/chroot/tftpd/lib

        /var/chroot/tftpd/tftpboot

        /var/chroot/tftpd/usr/lib

        /var/chroot/tftpd/usr/sbin

        /var/chroot/tftpd/var/run

        /var/cfengine/

#################################################################
files:
#################################################################

        generic_monitoring::
        #Below are files we wish to monitor the integrity. If these are compromised we are in trouble

        /usr/sbin/chroot
        mode=550
        owner=root
        group=root
        action=fixall
        recurse=0
        syslog=true
        checksum=md5
        define=

        /usr/sbin/cfagent
        mode=550
        owner=root
        group=root
        action=fixall
        recurse=0
        syslog=true
        checksum=md5

        /usr/bin/chfn
        mode=550
        owner=root
        group=root
        action=fixall
        recurse=0
        syslog=true
        checksum=md5

        /usr/bin/chsh
        mode=550
        owner=root
        group=root
        action=fixall
```

```
recurse=0
syslog=true
checksum=md5

/usr/bin/ls
mode=555
owner=root
group=root
action=fixall
recurse=0
syslog=true
checksum=md5

/usr/bin/find
mode=555
owner=root
group=root
action=fixall
recurse=0
syslog=true
checksum=md5

/sbin/ifocnfig
mode=550
owner=root
group=root
action=fixall
recurse=0
syslog=true
checksum=md5

/sbin/killall5
mode=550
owner=root
group=root
action=fixall
recurse=0
syslog=true
checksum=md5

/bin/kill
mode=550
owner=root
group=root
action=fixall
recurse=0
syslog=true
checksum=md5

/usr/bin/killall
mode=550
owner=root
```

```
group=root
action=fixall
recurse=0
syslog=true
checksum=md5

/usr/bin/killall
mode=550
owner=root
group=root
action=fixall
recurse=0
syslog=true
checksum=md5

/usr/sbin/lsof
mode=550
owner=root
group=root
action=fixall
recurse=0
syslog=true
checksum=md5

/sbin/syslog-ng
mode=550
owner=root
group=root
action=fixall
recurse=0
syslog=true
checksum=md5

/usr/sbin/tcpd
mode=550
owner=root
group=root
action=fixall
recurse=0
syslog=true
checksum=md5

/var/chroot/tftpd/usr/sbin/tcpd
mode=110
owner=root
group=root
action=fixall
recurse=0
syslog=true
checksum=md5

/var/chroot/tftpd/usr/sbin/atftpd
```

```
        mode=110
        owner=root
        group=root
        action=fixall
        recurse=0
        syslog=true
        checksum=md5

        /usr/sbin/atftpd
        mode=550
        owner=root
        group=root
        action=fixall
        recurse=0
        syslog=true
        checksum=md5

        /usr/sbin/sshd
        mode=555
        owner=root
        group=root
        action=fixall
        recurse=0
        syslog=true
        checksum=md5


#Setup directories for chrooting
        chroot_files::

        /var/chroot/tftpd/.
        mode=510
        owner=root
        group=root
        action=fixall
        recurse=0
        syslog=true

        /var/chroot/tftpd/etc/.
        mode=440
        owner=root
        group=root
        action=fixall
        recurse=inf
        syslog=true

        /var/chroot/tftpd/dev/.
        mode=660
        owner=root
        group=root
        action=fixall
        recurse=inf
```

```
syslog=true

/var/chroot/tftpd/lib/.
mode=550
owner=root
group=root
action=fixall
recurse=inf
syslog=true


/var/chroot/tftpd/usr/.
mode=110
owner=root
group=root
action=fixall
recurse=inf
syslog=true

/var/chroot/tftpd/tftpboot/.
mode=770
owner=root
group=root
action=fixall
syslog=true

/var/chroot/var/.
mode=550
owner=tftpuser
group=tftpgroup
recurse=inf
action=fixall
syslog=true
```

Appendix B

fstab

```
# /etc/fstab: static file system information.
#
# <file system>  <mount point>  <type>  <options>                  <dump>  <pass>
/dev/sda1        /              ext3    errors=remount-ro          0       1
/dev/sda3        none           swap    sw                         0       0
proc             /proc          proc    defaults          0        0
/dev/fd0 /floppy          auto  user,noauto               0        0
/dev/cdrom       /cdrom         iso9660 ro,user,noauto             0       0
/dev/sda5        /tmp     ext3  nodev,noexec,nosuid,defaults       0       2
/dev/sda6        /home    ext3  nodev,noexec,nosuid,defaults       0       2
/dev/sda7        /var     ext3  defaults                 0        2
/dev/sda8        /opt     ext3  nodev,defaults                     0       2
/dev/sda9        /usr     ext3  nodev,defaults                     0       2
```

Appendix C

motd

This machine is the property of GIAC Enterprises.
It is for authroized use only. Users (authorized or unauthorized) have no explicit or implicit expectation or privacy.

Any or all uses of this system and all files on this system may be monitored, recorded, copied, audited, intercepted, inspected, and disclosed to authorized personel, computer security incident response team (CERT), and/or law enforcement personel (FBI and/or local authorities). By using this system, the user consents to such monitoring, recording, copying, auditing, interception, inspection, and disclosure at the discretion of the authorized site and/or GIAC Enterprises personel.

Unauthorized or improper use of this system may result in administrative disciplinary action and/or civil and/or criminal penalities. By continuing to use this system you indicate your awareness of and consent to these terms and conditions of use. If you do not agree to the conditions stated above, LOG OFF IMMEDIATELY.

Appendix D

issue

This machine is the property of GIAC Enterprises.
It is for authroized use only. Users (authorized or unauthorized) have no explicit or implicit expectation or privacy.

Any or all uses of this system and all files on this system may be monitored, recorded, copied, audited, intercepted, inspected, and diclosed to authorized personel, computer security incident response team (CERT), and/or law enforcement personel (FBI and/or local authorities). By using this system, the user consents to such monitoring, recording, copying, auditing, interception, inspection, and disclosure at the discretion of the authorized site and/or GIAC Enterprises personel.

Unauthorized or improper use of this system may result in administrative disciplinary action and/or civil and/or criminal penalities. By continuing to use this system you indicate your awareness of and consent to these terms and conditions of use. If you do not agree to the conditions stated above, DO NOT LOGIN OR ACCESS THIS MACHINE IN ANY MANNER.

Appendix E

sshd_config

```
# Package generated configuration file
# See the sshd(8) manpage for defails

# What ports, IPs and protocols we listen for
Port 22
# Use these options to restrict which interfaces/protocols sshd will bind to
#ListenAddress ::
#ListenAddress 0.0.0.0
Protocol 2
# HostKeys for protocol version 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
#Privilege Separation is turned on for security
UsePrivilegeSeparation yes

# ...but breaks Pam auth via kbdint, so we have to turn it off
# Use PAM authentication via keyboard-interactive so PAM modules can
# properly interface with the user (off due to PrivSep)
PAMAuthenticationViaKbdInt no
# Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 3600
ServerKeyBits 768

# Logging
SyslogFacility AUTH
LogLevel INFO

# Authentication:
LoginGraceTime 600
PermitRootLogin yes
StrictModes yes

RSAAuthentication yes
PubkeyAuthentication yes
#AuthorizedKeysFile     %h/.ssh/authorized_keys

# rhosts authentication should not be used
RhostsAuthentication no
# Don't read the user's ~/.rhosts and ~/.shosts files
IgnoreRhosts yes
# For this to work you will also need host keys in /etc/ssh_known_hosts
RhostsRSAAuthentication no
# similar for protocol version 2
HostbasedAuthentication no
# Uncomment if you don't trust ~/.ssh/known_hosts for RhostsRSAAuthentication
#IgnoreUserKnownHosts yes
```

```
# To enable empty passwords, change to yes (NOT RECOMMENDED)
PermitEmptyPasswords no

# Uncomment to disable s/key passwords
#ChallengeResponseAuthentication no

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes


# To change Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#AFSTokenPassing no
#KerberosTicketCleanup no

# Kerberos TGT Passing does only work with the AFS kaserver
#KerberosTgtPassing yes

X11Forwarding no
X11DisplayOffset 10
PrintMotd no
#PrintLastLog no
KeepAlive yes
#UseLogin no

#MaxStartups 10:30:60
#Banner /etc/issue.net
#ReverseMappingCheck yes

Subsystem       sftp    /usr/lib/sftp-server
```

Appendix F

sudoers

```
# sudoers file.
#
# This file MUST be edited with the 'visudo' command as root.
#
# See the man page for details on how to write a sudoers file.
#

# Host alias specification
Host_Alias      ATFTPD = supportwall001, supportwall002, supportwall003


# User alias specification
User_Alias      FILECOPIERS = user


# Cmnd alias specification
Cmnd_Alias      MOVEIT = /bin/mv, /bin/ls
Cmnd_Alias      SCRIPTROOT = /home/user/boot7

# Runas alias
Runas_Alias     TFTPGET = tftpuser

# User privilege specification
root    ALL = (ALL) ALL
useradmin       ALL = (root) ALL NOPASSWD;
user    ALL = (tftpuser) MOVEIT
user    ALL = SCRIPTROOT


Defaults syslog=auth, logfile=/var/log/sudo.log
```

Last Updated: August 15th, 2020

Upcoming SANS Training

Click here to view a list of all SANS Courses

SANS FOR508 Sydney August 2020	Sydney, AU	Aug 17, 2020 - Aug 22, 2020	Live Event
SANS Virginia Beach 2020	Virginia Beach, VAUS	Aug 30, 2020 - Sep 04, 2020	Live Event
SANS London September 2020	London, GB	Sep 07, 2020 - Sep 12, 2020	Live Event
SANS Baltimore Fall 2020	Baltimore, MDUS	Sep 08, 2020 - Sep 13, 2020	Live Event
SANS Munich September 2020	Munich, DE	Sep 14, 2020 - Sep 19, 2020	Live Event
SANS Australia Spring 2020	, AU	Sep 21, 2020 - Oct 03, 2020	Live Event
SANS San Antonio Fall 2020	San Antonio, TXUS	Sep 28, 2020 - Oct 03, 2020	Live Event
SANS Northern VA - Reston Fall 2020	Reston, VAUS	Sep 28, 2020 - Oct 03, 2020	Live Event
SANS FOR500 Milan 2020 (In Italian)	Milan, IT	Oct 05, 2020 - Oct 10, 2020	Live Event
SANS Amsterdam October 2020	Amsterdam, NL	Oct 05, 2020 - Oct 10, 2020	Live Event
SANS Brussels October 2020	Brussels, BE	Oct 05, 2020 - Oct 10, 2020	Live Event
SANS Prague October 2020	Prague, CZ	Oct 12, 2020 - Oct 17, 2020	Live Event
SANS London October 2020	London, GB	Oct 12, 2020 - Oct 17, 2020	Live Event
SANS Orlando 2020	Orlando, FLUS	Oct 12, 2020 - Oct 17, 2020	Live Event
SANS October Singapore 2020	Singapore, SG	Oct 12, 2020 - Oct 24, 2020	Live Event
SANS Stockholm October 2020	Stockholm, SE	Oct 19, 2020 - Oct 24, 2020	Live Event
SANS Dallas Fall 2020	Dallas, TXUS	Oct 19, 2020 - Oct 24, 2020	Live Event
SANS Rome October 2020	Rome, IT	Oct 19, 2020 - Oct 24, 2020	Live Event
SANS SEC504 Rennes 2020 (In French)	Rennes, FR	Oct 19, 2020 - Oct 24, 2020	Live Event
SANS Cologne October 2020	Cologne, DE	Oct 26, 2020 - Oct 31, 2020	Live Event
SANS San Francisco Fall 2020	San Francisco, CAUS	Oct 26, 2020 - Oct 31, 2020	Live Event
SANS Geneva October 2020	Geneva, CH	Oct 26, 2020 - Oct 31, 2020	Live Event
SANS SEC560 Lille 2020 (In French)	Lille, FR	Oct 26, 2020 - Oct 31, 2020	Live Event
SANS Tel Aviv November 2020	Tel Aviv, IL	Nov 01, 2020 - Nov 05, 2020	Live Event
SANS London November 2020	London, GB	Nov 02, 2020 - Nov 07, 2020	Live Event
SANS Rocky Mountain Fall 2020	Denver, COUS	Nov 02, 2020 - Nov 07, 2020	Live Event
SANS DFIRCON 2020	Miami, FLUS	Nov 02, 2020 - Nov 07, 2020	Live Event
SANS Sydney 2020	Sydney, AU	Nov 02, 2020 - Nov 14, 2020	Live Event
SANS Krakow November 2020	Krakow, PL	Nov 02, 2020 - Nov 07, 2020	Live Event
SANS Paris November 2020	Paris, FR	Nov 02, 2020 - Nov 07, 2020	Live Event
APAC ICS Summit & Training 2020	Singapore, SG	Nov 13, 2020 - Nov 21, 2020	Live Event
SANS OnDemand	OnlineUS	Anytime	Self Paced
SANS SelfStudy	Books & MP3s OnlyUS	Anytime	Self Paced