Global Information Assurance Certification Paper

Lou Rabon

SANS GSNA

Practical Assignment Version 1.2

Auditing EnGarde Secure Linux v1.1

February 08 2002

Table of Contents

Introduction

Until recently, securing a Linux distribution relied on assembling different sources of information (i.e. CERT & SANS lists), and applying one's own knowledge of security practices and exploits to safeguard an out-of-the-box Linux machine from malicious threats on the internet. A "newbie" administrator had few choices when tasked with securing a Linux machine. Fast forward to the last two years, and the introduction of "hardening scripts" allowed less-experienced administrators an option to make up for their lack of security knowledge. This is also around the time that Secure Linux distributions started making their way into the marketplace. EnGarde Secure Linux is one of the newest entries into this field.

EnGarde Secure Linux (hereby referred to as "ESL") touts itself as the "most robust and secure candidate for the task"[1] of providing a secure Linux distribution. It is the goal of this paper to come up with a checklist that takes all aspects of *nix security into account and applies them to the device that will be audited in Assignment 2. *Please note: The goal of this paper is not intended to pass a final judgement on the EnGarde OS as a whole, but merely to judge it's security capabilities in the capacity listed under "device to be Audited" on page 19.*

Assignment 1 - Research in Audit, Measurement Practice, and Control

I. Current state of practice

Due to the newness of ESL, I was unable to find any previous audits or checklists that related directly to this distribution of Linux, so numerous resources were scoured to come up with the following checklist. I've intentionally included some older articles and pulled from a wide range of sources to ensure a well-rounded audit checklist:

- Auditing RedHat Linux 7.0 (workstation), Mary A. Laude, SANS Institute Practical for GSNA Certification
- Auditing Linux**,** Krishni Naidu, http://www.sans.org/checklist/linux_check.htm
- UNIX Security Checklist v2.0, Australian Computer Emergency Reponse Team, http://www.auscert.org.au/Information/Auscert_info/Papers/usc20.html
- Linux Administrators Security Guide, Kurt Seifried, http://www.seifried.org/lasg/
- Securing and Optimizing Linux: RedHat Edition, Gerhard Mourani, OpenDocs Publishing

In researching all of these sources, I've written down the most important points in each and combined them to make what I consider to be an extremely thorough checklist. Surprisingly, some checklists left out what others included. Some stressed the importance of securing the file system over securing the individual services, and others hit on all aspects but did not go into enough detail. I would give the "Securing and Optimizing Linux: RedHat Edition" book the most credit for explaining the methods behind the "madness" of security very thoroughly, as well as being one of the more detailed sources for every aspect of securing a Linux distribution. Although it was centered on RedHat, I found it to apply just as equally to EnGarde and other distributions of Linux. Plus, it was a free download which was an added bonus.

Even after drawing from the amount of sources I did, I'd have to say that there were still shortcomings in each approach. Generally, almost all of the above sources did not fully agree as to how to lock down the file system. Some recommended deleting files altogether, while others relied solely on a umask setting in /etc/profile to lock down executables, while still others had the

[1] "About EnGarde Security", http://www.engardelinux.org/security.html

immutable bit on and made sure it was owned and only useable by root. Some effectively combined these methods. All mentioned third-party tools as an afterthought rather than an intergral part of hardening the OS.

Another shortcoming of many of the audits and practices I came across was the indiscretion with which they recommended disabling certain systems. In some cases, it would lead to an almost unusable system for all users except root. I hope to have overcome these shortcomings by combining what I feel is a balance between deploying the system with as much functionality as possible, but with the highest security in mind. Some checklists were also out of date, which makes some of their recommendations obsolete. EnGarde also has some proprietary security built in that could be missed by not having complete knowledge of the OS.[2]

Engarde Secure Linux Features

One of the conventions of ESL distribution is the use of a web-based tool called Guardian Digital Webtool to configure the major portions of the operating system right out of the box. GD Webtool administration is achieved by browsing to the SSL-enabled IP address of the server on port 1023. There is also a user-administered port on 1022 for password changes and certificate download for SSH management. You cannot connect to this port without SSL, which decreases the chances of plain-text passwords and such being transferred across the network.

Remote administration and configuration can also be achieved through SSH, the secure alternative to telnet. An added security feature is the fact that SSH access is only granted with a certificate that must be generated by the ESL server and installed on the user's SSH client. This PKI-based approach ensures a very high level of remote administration security.

Updating is easy with the Guardian Digital Secure Network. An update script is run via the web console, and all dependencies are resolved. I put the update tool through it's paces, and it was able to update all of the packages with ease. There was a kernel update that needed to be applied which required the download of a perl script to be run at the command line, but this was very simple to use and it successfully updated the kernel. No in-depth knowledge of the Linux OS was needed to install this kernel update, so even a very inexperienced administrator could have applied it following the step-by-step directions from Guardian Digital. One caveat is that packages that are not installed by Guardian Digital are obviously not included in the update; this could lead to a false sense of security when administrators install unsupported binaries and expect that they too are being updated.

ESL comes out of the box with the Linux Intrusion Detection System (LIDS), an excellent tool for locking down and monitoring sensitive system resources (http://www.lids.org/). ESL's default configuration of LIDS is used to write-protect the following resources from all users, including root:

[2] See "Developers Comments" in Appendix C.

```
/bin
/boot
/etc/fstab
/etc/init.d
/etc/inittab
/etc/lids
/etc/lilo.conf
/etc/postfix/postfix-script
/etc/pwdb.conf
/etc/rc
/etc/rcS
/etc/securetty
/etc/sysctl.conf
/lib
/usr
/usr/bin
/usr/sbin
/usr/sbin/userpass
/usr/sbin/webtool_cron_helper.pl
/usr/webtool/miniserv.pl
/usr/userpass/userpass.pl
/usr/local
/sbin
/var/lib/rpm
```

Using LIDS, the following directories are denied all access by all users and hidden (except when administered through the GD Webtool):

```
/usr/userpass
/usr/webtool
/etc/webtool
```

For logging, ESL uses syslog-ng, the "next-generation" syslog daemon (http://www.balabit.hu/en/downloads/syslog-ng/). This a much more fine-tuned solution than the previous option, syslogd. The information recorded by syslog-ng is:

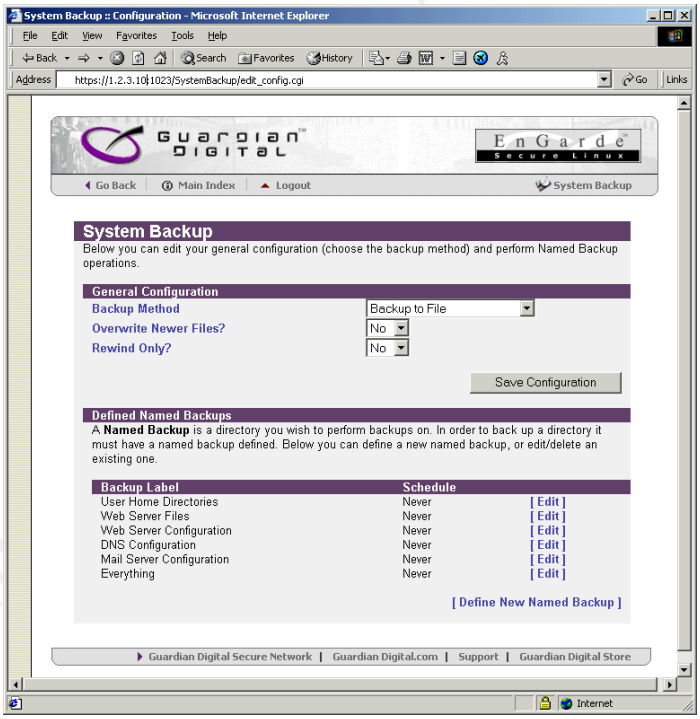
```
destination authlog { file("/var/log/auth.log"); };
destination syslog { file("/var/log/messages"); };
destination cron { file("/var/log/cron.log"); };
destination kern { file("/var/log/kern.log"); };
destination uucp { file("/var/log/uucp.log"); };
destination mail { file("/var/log/mail.log"); };
destination spooler { file("/var/log/spooler.log"); };
destination sudo { file("/var/log/sudo.log"); };
destination boot { file("/var/log/boot.log"); };
destination debug { file("/var/log/debug.log"); };
destination messages { file("/var/log/messages"); };
destination loginlog { file("/var/log/login.log"); };
destination console { usertty("root"); };
destination console_all { file("/dev/tty12"); };
destination tty8-log { file("/dev/tty8"); };
```

ESL also uses SWATCH, the Simple WATCHer to manage logs (http://www.oit.ucsb.edu/~eta/swatch/). ESL is monitoring the following logs and creating specialized logs in /var/log/audit/ based on the results of SWATCH's log parsing (NOTE: In the interest of space, SWATCH DNS entries have been left out since this server will not be running DNS. Provisions for parsing DNS logs, however, are included with EnGarde's default configuration as well):

```
## PAM errors, logins, etc..
 watchfor /pam_limits/
        exec=echo $0 >> /var/log/audit/pam.log

 watchfor /PAM_pwdb.*login/
        exec=echo $0 >> /var/log/audit/pam.log

 watchfor /PAM.*\(su\).*session opened/
        exec=echo $0 >> /var/log/audit/su_logins.log

 watchfor /PAM.*\(su\).*session\ closed/
        exec=echo $0 >> /var/log/audit/su_logins.log

 watchfor /su\:.*PAM/
        exec=echo $0 >> /var/log/audit/su_fail.log

 watchfor /PAM.*authentication failure.*for\ su\ service/
        exec=echo $0 >> /var/log/audit/su_fail.log

##  check SSH authentication
#
watchfor /sshd\[/
        exec=echo $0 >> /var/log/audit/ssh_authentication.log

##  LIDS stuff
#
watchfor /LIDS\:.*Give\ incorrect\ password/
        exec=echo \"$0\" >> /var/log/audit/lids-failed_password.log

 watchfor /LIDS\:.*Give\ incorrect\ password.*try.*3/
        exec=/usr/bin/swatch-mail

watchfor /LIDS.*switched to/
        exec=echo $0 >> /var/log/audit/lids.log

watchfor /.*LIDS.*/
        exec=echo $0 >> /var/log/audit/lids_general.log

watchfor /lidsadm.*Config\.\ file reloaded/
        exec=echo $0 >> /var/log/audit/lids_general.log

##  Important kernel info
#
watchfor /panic/
        exec=echo $0 >> /var/log/audit/panic.log

##  General kernel info
#
watchfor /kernel/
        exec=echo $0 >> /var/log/audit/kernel.log

##  Postfix mail log for relaying and deny info
#
watchfor /postfix.*relay/,/postfix.*deny/,/postfix.*sender domain must
resolve/,/postfix.*time
out waiting for
input/,/postfix.*stat=Deferred:/,/postfix.*timeout.*waiting/,/postfix.*domain
must exist/
        exec=echo $0 >> /var/log/audit/mail_relay_deny.log

watchfor /postfix.*relay/
        exec=echo $0 >> /var/log/audit/mail_relay_deny.log

watchfor /postfix.*deny/
        exec=echo $0 >> /var/log/audit/mail_relay_deny.log

watchfor /postfix.*sender domain must resolve/
        exec=echo $0 >> /var/log/audit/mail_relay_deny.log

watchfor /postfix.*timeout waiting for input/
        exec=echo $0 >> /var/log/audit/mail_relay_deny.log

watchfor /postfix.*state\=deferred:/
        exec=echo $0 >> /var/log/audit/mail_relay_deny.log

watchfor /postfix.*timeout.*waiting/
        exec=echo $0 >> /var/log/audit/mail_relay_deny.log
```

```
watchfor /postfix.*domain must exist/
        exec=echo $0 >> /var/log/audit/mail_relay_deny.log

watchfor /postfix-script/
        exec=echo $0 >> /var/log/audit/mail_scripts_error.log

watchfor /postfix.*warning/
        exec=echo $0 >> /var/log/audit/mail_warning.log

watchfor /postfix.*fatal/
        exec=echo $0 >> /var/log/audit/mail_fatal.log

watchfor /postfix-script.*stopping/
        exec=echo $0 >> /var/log/audit/mail_startstop.log

watchfor /postfix-script.*starting/
        exec=echo $0 >> /var/log/audit/mail_startstop.log

watchfor /postfix.*bounce/
        exec=echo $0 >> /var/log/audit/mail_bounce.log

## Watch for log errors in klog and syslog-ng
#
watchfor /klog/
        exec=echo $0 >> /var/log/audit/log-msgs.log

watchfor /syslog/
        exec=echo $0 >> /var/log/audit/log-msgs.log

watchfor /swatch/
        exec=echo $0 >> /var/log/audit/log-msgs.log
```

For backups, ESL uses Flexbackup (http://members.home.com/flexbackup/) coupled with afio. Configuration is very easy via the GD Webtool:



Another extremely useful tool included with an out-of-the-box ESL installation is Snort, a Network Intrusion Detection System (NIDS). ESL has no Snort administration on the GD Webtool, so configuration of this service is left to more knowledgable administrators. ESL has smartly decided to run Snort in a chroot jail, further increasing security.

II. Quick definition of subjective and objective measurements

For the purposes of this paper I will define objective measurements as something that a tool will tell you is true or false (i.e. not open to interpretation), and a subjective measurement will be

defined as something that is subject to the opinion of the administrator or environment. So, for instance, if we've determined through an initial interview that port 111 is an unnecessary service and a port scan turns up a service listening on port 111, we've hit upon both an objective and subjective measurement: 1) The decision that port 111 is unnecessary is a subjective measurement, usually made by the system administrator of the target, and 2) the objective and non-disputable results of the port scanner, which turned up port 111 as an open port.

III. Subjective measurements

Subjective measurements initially require an interview to determine exactly what role the device being audited will play. In my experience, it is important to discuss some form of Time-Based Security[3] with the administration of the target site before having them fill this checklist out. This will help narrow their definitions of "essential" services.

The following is one part of a form that I would give to highest-level IT administrator/decision-maker of our target network. This should serve to convert a number of "subjective" measurements into "objective" ones:

What essential services will this device be providing? _____
Will this device be assigned a secondary, or backup role? _____
If so, please explain: _____
Will this device be hosting any services on the internet? _____
If so, please explain: _____
Will remote access be needed? _____
If so, by whom? _____
If users will be logging into this device, what level of access will they need?
(_) Administrator, high-risk (access to root password, all areas of device & network)
How many users with this access: ____
(_) Developer, high-risk (access to compilers, binaries, etc)
How many users with this access: ____
(_) Power User, medium-risk (access to mid-level binaries, ability to run certain programs)
How many users with this access: ____
(_) End User, low-risk (limited access to all areas of the device)
How many users with this access: ____

By getting these answers, we have effectively taken a large amount of the subjectivity out of the audit by defining what is necessary, and therefore we can test this objectively with the proper tools. Throughout the following audit instructions (labeled "Passing Grades for this Audit") I have indicated in italics when a measurement could be considered subjective.

IV. Objective Measurements

Objective measurements for auditing any device are easy, provided you have the tools necessary and know what constitutes a "pass" or "fail". That being said, here are the tools I will use to conduct this audit (NOTE: ESL does not come with the tools necessary to compile binary programs. When possible, i've used statically linked libraries, but in those circumstances where this was not possible I downloaded and installed the development binaries from Guardian Digital):

- Nmap – the best scanner for both internal and external port-scanning
- Nessus – One of the top Vulnerability scanners
- SARA – because, hey, is one vulnerability scanner enough?
- Stat – to determine detailed information about files and directories

[3] Schwartau, Winn. *Time-Based Security*. Interpact Press, 1999.

- Typical *nix commands like find, ps, ls, lsof, netstat, etc
- SING – A great ICMP packet crafter (http://sourceforge.net/projects/sing)
- COPS – for checking file-system security
- Genovex – A good buffer overflow tester (http://www-miaif.lip6.fr/willy/security/sources/genovex.c)
- SNORT – Sniffer and NIDS, included with the default ESL installation
- cgichk – A CGI vulnerability scanner (http://sourceforge.net/projects/cgichk/)
- ESL.audit – I've created a custom shell script to assist with information collection. See Appendix B.

I've broken the audit checklist down into 6 categories. At the end of each category there is a "WRAP-UP" section that will use the aforementioned tools to test the system for compliance, independent of results that have a passing grade on the checklist. For instance, if netstat returns results indicating no unknown services are being hosted, but nmap returns a port listening for DNS requests (port 53), then that part of the wrap-up will fail and it will be necessary to explore why DNS is listening when netstat does not list it (and it's presumably not been configured). This should serve to further remove as much subjectivity from the audit as possible.

V. Passing grades defined for this audit:

Again, please note that these are passing grades related ONLY to the device intended to be audited and not the operating system as a whole.

Section I, OS/Kernel:

- **a.** FTP to ftp.engardelinux.org and review the list of files in /pub/engarde/stable/updates/MD5SUMS. These files should be checked against the RPM versions (#RPM –qa > /rpm.found). PASS if all match up.
- **b.** Run # df > df.found and see where the filesystems are. Pass if /home & /var are on separate partitions from /, /dev, and /tmp. For more information, refer to the chart in Appendix A.

WRAP UP:

- Check http://cve.mitre.org/cve/ for vulnerabilities on kernel version and packages. PASS if no vulnerabilities found.

Section II, Boot Security:

ESL provides two modes at boot-time: secure and standard. The secure mode boots to a LIDS kernel, and it is the default mode. The standard mode boots to a standard, non-LIDS enabled kernel. NOTE: Although it is possible in the default configuration of ESL to boot to "standard" mode without a password, ESL uses **sulogin** to restrict single-user mode without the root password. The potential security risk is the ability of a normal user who has physical access to the box to boot to standard mode, thereby disabling the protection features that LIDS provides. It is therefore necessary to password protect the "standard" mode boot configuration.

- **a.** Check /etc/lilo.conf for the presence of "restricted, password=<password>". under the "standard" boot configuration. PASS if present.
- **b.** Check /etc/shutdown.allow. PASS if only root is listed.
- **c.** Also check lilo.conf to see that only root has read/write access to this file. Use stat to ensure compliance (Mode: 0600).
- **d.** Is lilo.conf immutable? By default, LIDS should be protecting this file from all users, including root. To test, open the file in vi and add any character at the bottom, then try to write the file (using both :w and :w!). PASS if the file is unable to be written to.
- **e.** Check that root is the only user to have access to /etc/rc.d/init.d by using stat (Mode should=600):
  - # stat /etc/init.d/* > initd.stat
- **f.** FAIL if /etc/issue and /etc/issue.net contain commands to show the kernel version. PASS if a disclaimer is displayed.
- **g.** Check /etc/security/console.apps directory. PASS if there are no console-equivalent apps such as shutdown, reboot, halt listed.
- **h.** Use stat to print out results of /sbin/reboot, halt, and shutdown. PASS if Mode=700 or equivalent.

WRAP UP:

- Log out of root account, and log in as a normal user (audit). Attempt to:
  - edit the /etc/lilo.conf, /etc/initab file and /boot/vmlinuz files
  - reboot the server using /sbin/reboot. /sbin/shutdown and /sbin/halt
  - Press CTRL-ALT-DELETE.
  - su to root
- su to root and reboot the server. Upon reboot, attempt to enter single-user mode by

typing “standard -b” the the boot: prompt. PASS if you are prompted for a password.

- Attempt to edit the /boot/vmlinuz file

Section III, File System:

The LIDS installation protects a large amount of files in a default ESL installation (see figure above). This checklist is designed to take into account the things LIDS does not protect.

a. Search for the presence of /etc/exports. PASS if file is not found.
- # find / -name export* -print > exportfs.found

b. # cat /etc/fstab > fstab.found and ensure restricted access on the /tmp and /home directories. PASS if fstab has the following:

```
/dev/hda3       home     ext2    rw,nosuid,nodev,noexec
/dev/hda1       tmp      ext2    rw,nosuid,nodev,noexec
```

c. Find the RPM program and ensure only root has execute permissions (PASS if Mode=700 or equivalent),

stat /bin/rpm > rpm.stat

d. Find all files with SETUID and SETGID permissions:
- # find / -type f \(-perm -04000 -o -perm -02000 \) \-exec ls –lg {} \; > setuid.found
- FAIL if any of the listed programs are not Mode 4711 or 2711
- *SUBJECTIVE:* FAIL if there are an inordinate amount of listed programs

e. *SUBJECTIVE:* Find unsuspected hidden or “unusual” files.
- # find / -name ".. " -print –xdev >> hidden.found; find / -name ".*" -print -xdev | cat –v >> hidden.found
- FAIL if any unknown or unsuspected programs are found (other than .bashrc, .profile, etc.)

f. *SUBJECTIVE:* Find world-writable and world-readable files & directories. PASS if these files are only located in expected areas, such as web page and mail directories.
- # find / -type f \(-perm -2 -o -perm -20 \) -exec ls -lg {} \; >> readwrite.found
- # find / -type d \(-perm -2 -o -perm -20 \) -exec ls -ldg {} \; >> readwrite.found

g. PASS if no unowned files are found:
- # find / -nouser -o –nogroup > unowned.found

h. Find .rhosts files. PASS if none are found
- # find /home -name .rhosts > rhosts.found

i. Ensure logs are protected from writing by all but root:
- PASS if stat shows /var/log/* mode=640 or equivalent.

j. Ensure /etc/hosts.allow and hosts.deny is protected from all but root:
- PASS if stat shows /etc/hosts* mode=640 or equivalent.

k. Using stat, make sure /tmp directory permissions are secured. PASS if Mode=1777

l. Check /etc/profile for the existence of a umask statement. PASS if the following is present:

```
if [ `id -gn` = `id -un` -a `id -u` -gt 14 ]; then
            umask 027
else
            umask 077
fi
```

m. Issue the following command to ensure all files in the /dev directory are “special” files. PASS if only MAKEDEV is found:
- # find /dev -type f -exec ls -l {} \; > normaldev.found

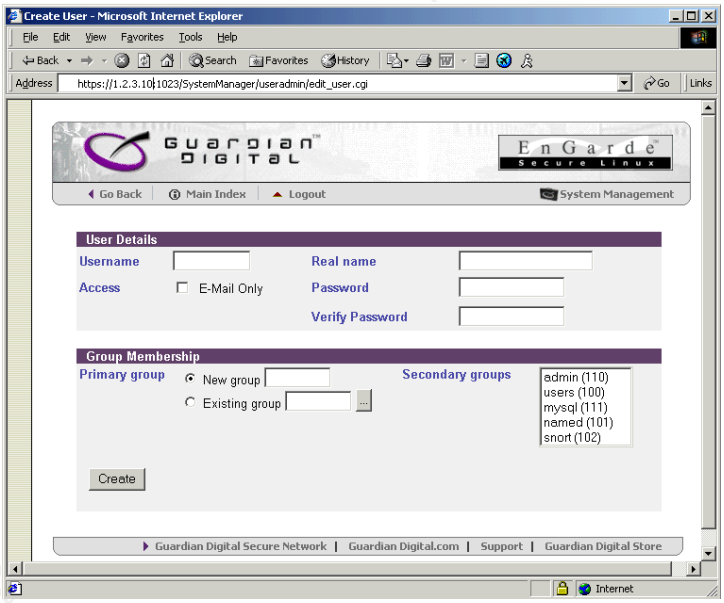
n. Issue the following command to find any files with the “character” or “block” attribute in the /dev directory to prevent unauthorized device access. Pass if none are found:
- # find / \(-type b -o -type c \) -print | grep -v '^/dev/' > specialdev.found

WRAP UP:

- Use COPS to ensure compliance. Print results and check errors. FAIL if unforeseen problems (include details).
- Use Tiger to supplement COPS scan.
- Use genovex to try to overflow a SETUID binary.
- Use lids.test script to test
- Log in as our test user (audit) with normal user privileges and bang on the file system. PASS if unable to do any of the following:
  - try to read anything in the /etc file
  - try to disable LIDS (http://www.team-teso.org/advisories/teso-advisory-012.txt)[4]
  - try to run RPM
  - try to view/modify log files
  - try to modify binaries (ls, etc.)

Section IV, Users:

A consistent user policy must be created in every environment in which users will be accessing the server. ESL includes an "email only" option in it's Webtool configuration in which users are given a /bin/false shell, which prevents access and logs any attempts to login locally:



There are many areas that can still be locked down by an experienced administrator, however, so the following should serve to protect unauthorized user access even further.

a. *SUBJECTIVE:* All local account logins should be restricted except those specified in etc/security/access.conf. PASS if access.conf includes the following:

```
# Disallow console logins to all but a few accounts.
-:ALL EXCEPT {user user user}:console
```

b. Check /etc/passwd for plain-text passwords. FAIL if these exist.

c. Check /etc/shadow for blank passwords (2nd field should have MD5 hash or some other

[4] Special thanks to stealth (stealth@segfault.net) for the lids-hack instructions and binaries (http://stealth.7350.org/lids-hack.tgz).

character). FAIL if this is blank.

- Ex: user1:1IZ9GKqMs$1AATYcqsneXYRI/yucOq/.:11685:0:99999:7::: **PASS**
  user2:*:11661:0:99999:7::: **PASS**
  user3:!!:11661:0:99999:7::: **PASS**
  user4::11685:0:99999:7::: **FAIL**

d. Check /etc/login.defs for minimum length of 7 characters. FAIL if less than 7:

- Ex: PASS_MIN_LEN 8 **PASS**

e. Stat the login.defs file and make sure it's Mode is 640 or equivalent. PASS if only root can read and modify the file.

f. *SUBJECTIVE:* check /etc/passwd for unused accounts that have a shell other than /bin/false. PASS if none found.

- Example: User "lpr" is present, but print services are not enabled – FAIL

g. Under /etc/security/limits.conf, the following lines should be present:
PASS if the following limits or equivalent are present:

```
# prevent core dumps
*       hard    core    0
#limit user processes per user to 150
*       hard    nproc   20
# limit size of any one of users' files to 40mb
*       hard    fsize   10000
# limit user logins
*       hard    maxlogins       5
# limit max resident set size
*       hard    rss       5000
```

h. Check /etc/profile for HIST and HISTSIZE. FAIL if number is larger than 20.

i. PAM should be enabled. Check by doing an ldd on /bin/login. PASS if libpam is present.

- Example: libcrypt.so.1 => /lib/libcrypt.so.1 (0x00127000)
  libpam.so.0 => /lib/libpam.so.0 (0x00155000) **PASS**
  libdl.so.2 => /lib/libdl.so.2 (0x0015d000)
  libpam_misc.so.0 => /lib/libpam_misc.so.0 (0x00161000)
  libc.so.6 => /lib/libc.so.6 (0x00165000)
  /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x00110000)

j. Check /etc/pam.d/login to ensure pam_limits.so is utilized.
PASS if the following line is included in this file:

```
        session    required     /lib/security/pam_limits.so
```

k. Check /etc/pam.d/ files for the existence of pam_console.so with the following to identify which files it may be in:

- # find /etc/pam.d -exec grep -q 'pam_console.so' {} \; -print > pam_console.found

l. Root account should have a timeout set to 900 seconds (15 minutes) or below. Check /root/.bashrc for the line TMOUT=1800. FAIL if no timeout is set, or it's below the specified limit.

m. User accounts should have timeouts of 600 seconds (10 minutes) or below. FAIL if no timeout is set, or it's below the above limit.

n. Check /root/.bashrc for HISTSIZE=10 or lower. FAIL if more than a 10 command history is available.

o. Check /etc/crontab permissions using stat. FAIL if permissions are below 640 and/or root does not own the file.

p. Check /etc/shells for the presence of /dev/null, and check permissions on the file. PASS if /dev/null is present, Mode=640 or equivalent, and root owns the file.

WRAP UP:

- Test "login" command to log in as other users
- Review Tiger results

- Try creating simple passwords and check for error messages
- Try logging in at the console with user accounts not included in /etc/access.conf

Section V, General Network Services:

During the initial installation of ESL, it is necessary to explicitly pick which services will be started when the server initializes. Again, this is where an effective policy will play a role in determining which services are necessary. ESL comes configured with two features to protect unauthorized access, libwrap and ipchains. This checklist will ensure these are running properly and also that other known insecure services are prevented from starting.

a. Stat the xinetd.conf file. Is it owned and writeable only by root? FAIL if this is false.
b. Run # chkconfig --list | grep 'on' > chkconfig.found. Check existence of inetd.conf or xinetd.conf. FAIL if either service is not present.
c. *SUBJECTIVE:* Check chkconfig.found for any unnecessary services. Next, run netstat –l > netstat.listen; ps –ax > ps.found and compare. FAIL if unnecessary services are present on either.
d. Run # lsof –i > lsof.listen and ensure only expected services are listening. Also, compare results to chkconfig.found, ps.found and netstat.listen. PASS if no new services appear.
e. Check the /etc/xinetd.conf and the /etc/xinetd.d/ directory for the presence of any unnecessary services. FAIL if unnecessary services are present.
f. Check /etc/hosts.deny for the uncommented presence of ALL:ALL. PASS if present.
g. Check /etc/hosts.allow for presence of trusted hosts only. PASS if blank, or known IPs are listed.
h. For the services this server is intending to host externally (in this case SPOP3, SSH, SMTP) is xinetd.conf wrapping these services? PASS if services are unable to be connected to unless entry is added to hosts.allow.
    - telnet to port 22 of the server from an untrusted ip (an IP not explicitly named in /etc/hosts.allow) and check the response. PASS If you see:

```
Trying 1.2.3.10...
Connected to 1.2.3.10.
Escape character is '^]'.
Connection closed by foreign host.
```

FAIL if you see:

```
SSH-1.5-OpenSSH_2.3.0p1
```

    - Also, check /var/log/messages. PASS if you see a log entry that resembles the following:
    - Dec 30 18:59:27 1.2.3.10 sshd[4337]: refused connect from 195.74.192.154
i. Check for presence of linuxconf. FAIL if present.
    - # find / -name linuxconf -print
j. Check /etc/host.conf for the following line (FAIL if not present):
nospoof on
k. Check /proc/net for the existence of ip_fwchains. FAIL if not present.
l. *SUBJECTIVE:* Run # ipchains –L > ipchains.list to examine the firewall rules. Are ICMP messages restricted? Namely, type 5 Redirects. PASS if ICMP messages are restricted to a select few:

```
Chain forward (policy ACCEPT):
target     prot opt     source               destination          ports
ACCEPT     icmp ------  anywhere         anywhere         destination-unreachable
ACCEPT     icmp ------  anywhere         anywhere         time-exceeded
ACCEPT     icmp ------  anywhere         anywhere         echo-reply
REJECT     all  ----l-  anywhere         anywhere          n/a
```

m. Check ipchains.list for the presence of the private address spaces from RFC1918. Test with the following command (FAIL if packet accepted):
- # ipchains -C input -i eth0 -p tcp -s 192.168.2.12 4353 -d 1.2.3.10 80 > ipchains.test

n. Check the ipchains list for the presence of the equivalent of "deny all unless otherwise accepted". PASS if the following line is present at the end of the chain:
DENY all ------ anywhere anywhere n/a

o. *SUBJECTIVE:* Check /etc/sysctl.conf for the presence of the following lines (FAIL if they are not present as shown):

```
# Enable ignoring ping request
net.ipv4.icmp_echo_ignore_all = 1
# Enable ignoring broadcasts request
net.ipv4.icmp_echo_ignore_broadcasts = 1
# Disables IP source routing
net.ipv4.conf.all.accept_source_route = 0
# Enable TCP SYN Cookie Protection
net.ipv4.tcp_syncookies = 1
# Disable ICMP Redirect Acceptance
net.ipv4.conf.all.accept_redirects = 0
# Enable bad error message Protection
net.ipv4.icmp_ignore_bogus_error_responses = 1
# Enable IP spoofing protection, turn on Source Address Verification
net.ipv4.conf.all.rp_filter = 1
# Log Spoofed Packets, Source Routed Packets, Redirect Packets
net.ipv4.conf.all.log_martians = 1
```

p. Check /proc/sys/net/ipv4/icmp_echo_ignore_broadcast_requests, PASS if 0 is present.
q. Check all interfaces under /proc/sys/net/ipv4/conf for the presence of 1 in any of the accept_source_routing files. FAIL if 1 is present.
- # find /proc/sys/net/ipv4/conf/*/accept_source_route -exec cat {} \; -print > sourceroute.icmp

r. Check /proc/sys/net/ipv4/tcp_syncookies, PASS if 1 is present.
s. Check all interfaces under /proc/sys/net/ipv4/conf for the presence of "1" in any of the accept_redirects files. PASS if 0 is present:
- # find /proc/sys/net/ipv4/conf/*/accept_redirects -exec cat {} \; -print > redirects.icmp

t. Check all interfaces under /proc/sys/net/ipv4/conf for the presence of "1" in any of the rp_filter files. PASS if 1 is present:
- # find /proc/sys/net/ipv4/conf/*/rp_filter -exec cat {} \; -print > rpfilter.found

u. Check all interfaces under /proc/sys/net/ipv4/conf for the presence of "1" in any of the log_martians files. PASS if 1 is present:
- # find /proc/sys/net/ipv4/conf/*/log_martians -exec cat {} \; -print > martians.found

v. Check /proc/sys/net/ipv4/ip_forward, PASS if 0 is present.

WRAP UP:

- Use nmap internally and externally. Attach results. FAIL if results are unexpected, and explain.
- Use Nessus. Attach results. FAIL if results are unexpected, and explain.
- Use SING to test ICMP answers

Section VI, Specific Network Services:

ESL uses the following TCP/IP applications:

- Postfix for SMTP
- stunnel for Secure POP3 and IMAP services
- xntpd for NTP
- VsFTP for FTP
- Apache for HTTP
- OpenSSH for SSH

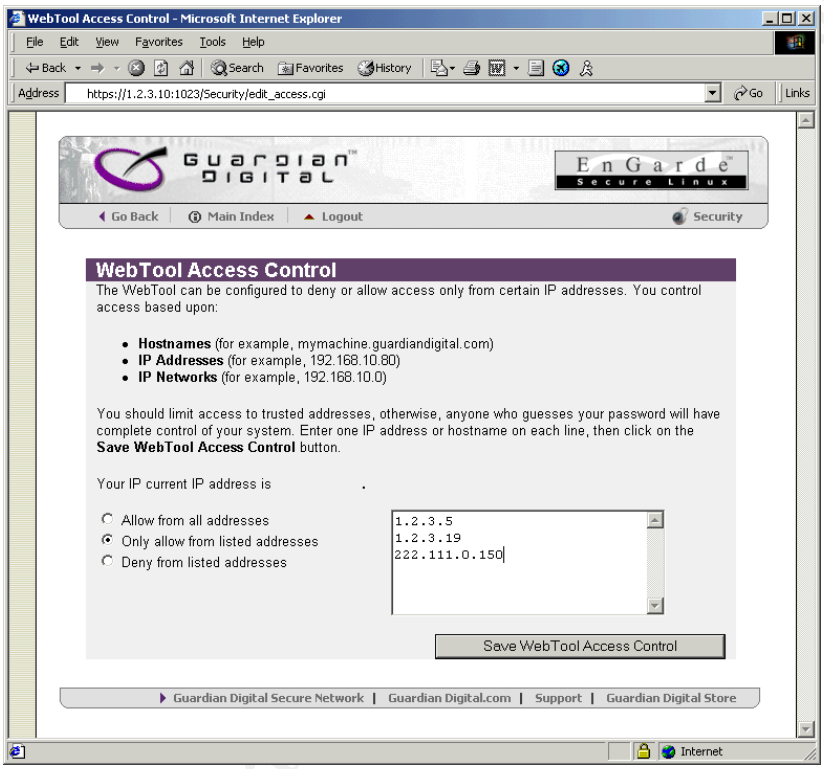
This section is solely for the services that will be used on the server that will be audited. Please note: HTTP has not been included, even though the GD Webtool is running under the http daemon, due to the fact that the GD Webtool is not being hosted as a traditional client under Apache and therefore auditing and securing it is a different matter. The default ESL Apache configuration is protected in the expected configuration:

- root owns the httpd binaries and directories
- only root has execute access
- .htaccess and .htpasswd are protected in /etc/httpd/conf/httpd.conf
- All Webtool CGI is located in /usr/webtool, which is hidden and protected by LIDS.

GD Webtool

a. *SUBJECTIVE*: The administrative and root passwords will have been set at the start of the audit. Attempt to login to the Webtool with the root password. PASS if the passwords are different.

b. *SUBJECTIVE*: Log in to the Webtool administrative program and check under Security:Webtool Access Control:Allow only from listed addresses. Only known administrative IP addresses should be specified. PASS if access is restricted by listed IPs:



SSH

c. *SUBJECTIVE:* Ensure /etc/hosts.allow has the IP addresses of all SSH users.

d. Ensure root cannot login by checking /etc/ssh/sshd_config
- PASS if "PermitRootLogin no" is present.

e. *SUBJECTIVE:* The /etc/ssh/sshd_config file should have the following settings or equivalent:

```
ListenAddress 1.2.3.10
HostKey /etc/ssh/ssh_host_key
ServerKeyBits 1024
LoginGraceTime 600
KeyRegenerationInterval 3600
PermitRootLogin no
IgnoreRhosts yes
IgnoreUserKnownHosts yes
StrictModes yes
X11Forwarding no
PrintMotd yes
SyslogFacility AUTH
LogLevel INFO
RhostsAuthentication no
RhostsRSAAuthentication no
RSAAuthentication yes
PasswordAuthentication no
PermitEmptyPasswords no
AllowUsers audit
```

SMTP

f. By default, Postfix does not relay. Ensure by telnetting to port 25 of the mail server and typing the following commands (PASS if relay is denied):

```
helo l.com
250 mail.audit-machine.com
mail from:spammer@spam.com
250 Ok
rcpt to: testaccount@yahoo.com
554 <testaccount@yahoo.com>: Recipient address rejected: Relay access denied
```

g. Postfix also does not recognize the EXPN or HELP commands, but it does recognize VRFY. Check /etc/postfix/main.cf for the existence of a command to diasble VRFY requests.
- PASS if disable_vrfy_command = yes exists.

h. Ensure postfix is chrooted with the presence of the following lines in /etc/postfix/master.cf:
- smtp inet n - y - - smtpd
- smtp unix n - y - - smtpd

POP3

i. SSL-enabled POP3 (SPOP3) should be installed. Type the following line to ensure this is the case (FAIL if POP3 is on port 110):
- **# netstat –ant** or **netstat -alt**

j. Ensure service is wrapped by xinetd by removing any IPs listed and trying to connect. PASS if access is denied and a log error is created in /var/log/messages resembling the following:
- Jan 6 13:28:03 mail stunnel[2752]: Connection from testhost:1839 REFUSED by libwrap

NTP

k. Is NTP configured for a chroot jail?
- Check /var/chroot for ntp entries. FAIL if no entries exist.
- Check /etc/init.d/xntpd for chroot entries. FAIL if not chrooted.

WRAP UP:
- Try logging into FTP, see if service starts and/or allows access.
- Brute force GD Webtool using Webcracker.
- Check http://cve.mitre.org/cve/ again for vulnerabilities on all hosted services.
- Use a sniffer to decipher packets going outbound to webtool.
- Run cgichk to check for cgi vulnerabilities.
- Use SARA and NESSUS to scan for vulnerabilities.

Section VII, Logging, Monitoring and Backups:

ESL uses Swatch and syslog-ng to monitor logs. These are excellent tools that provide added functionality for the task of administering logs. The Swatch program is configured to parse all server logs and break them down into distinct log files stored under the directory /var/log/audit. In addition, a Cron job is setup to run daily to pull information from the /var/log/messages file and summarize it, using a perl script named logsum, which resides in /usr/bin. Backups are an integral part of the default install, and a Backup option is one of six icons available from the default GD Webtool page, further stressing the importance of backing up.

a. Is the tripwire database set?
- Run # tripwire –m c to ensure the database runs and is set up correctly. PASS if a

report is generated.

b. Are tripwire logs being sent to the administrator?
- Run # tripwire --test –e test@mydomain.com PASS if there are no errors and the message reaches the intended address.

c. Are tripwire reports scheduled and running daily?
- Check /etc/crontab for the existence of the following line:

```
### Run Tripwire
0 6 * * * root /usr/sbin/tripwire -m c -t 1 -M > /dev/null 2>&1
```

- Also check /var/lib/tripwire/report for reports. PASS if reports are present and they're running at the scheduled time.

d. Are LIDS alerts configured?
- Run # lidsadm –S -- -LIDS to disable LIDS, then edit /etc/lids/lids.net to ensure the administrator email is configured. PASS if "MAIL_TO=" has an administrator's email address.

e. Is Cron running a logrotate script?
- Run # find /etc -name logrotate –print and see if logrotate is in any of the cron* directories. PASS if logrotate exists.

f. Is the administrator receiving daily log summaries from Swatch?
- Check /var/log/audit for a daily.summary file dated within the last 24 hours using # ls -l /var/log/audit/daily* > logsum.date
- Also check /etc/logsum.conf for the line EMAIL=<administrator email> using # cat /etc/logsum.conf | grep 'EMAIL =' > logs.email.
- PASS if both are present.

g. Check through the GD Webtool to ensure that backups are scheduled on a regular basis.
- Under System Backup:Configuration , Check Backup Label "Everything", Check for "Every Day (Incremental)" or equivalent.
- Check /var/BACKUP directory for *afio-gz files
- PASS if both conditions are met.

WRAP UP:
- Turn off LIDS and modify a binary, turn LIDS back on and run Tripwire. Was the change caught?
- Port scan from an external host. Check /var/log/messages for scan
- Review LIDS and Swatch email for information.

Assignment 2 – Application of Audit Techniques to a Real World System

I. Device to be audited:

I am auditing a Guardian Digital Secure Internet Basic 1050-S1 (http://store.guardiandigital.com/html/eng/83-AA.shtml) server running Engarde Secure Linux version 1.1. The only essential services for this server, as determined by the policy of the environment, will be NTP, SPOP3, SSH, and SMTP. The physical security of the server is ensured through an additional security policy that restricts physical access to the server to all but the administrator of the site. The following is also relevant:

- Unless otherwise noted, the user will be logged in as root for most of the testing.
- The untrusted IP address of this server will be noted as 1.2.3.10, which is not it's real IP.
- SNORT Intrusion Detection will not be configured since this machine will be fully facing the internet and not on any internal network segment.
- Although EnGarde Linux comes with an excellent GUI via HTTPS (known as the Guardian Digital Webtool), this audit will be conducted almost entirely from the command prompt to ensure the correct settings. Please see the ESL documentation for instructions on using the GD Webtool to configure the server.
- The following post-installation settings are the only things that have been changed from the default "out-of'the-box" installation:
    - The Update agent has been configured and all updates have been run via the webtool.
    - The webtool password and root password have been set.
    - A number of users have been set up as "email-only".
    - Since no users will be logging in locally, the umask entry in /etc/profile should read be set to 027 or 077. ESL defaults umask values to 002 and 022, so this must be changed initially.
    - A user named "audit" has been created as a normal user with SSH access.
    - Postfix and stunnel have been configured to allow SMTP and SPOP3 traffic.
    - TCP_Wrappers have been configured to allow only the administrative IP for SSH access, and ALL for SPOP3 access.
    - SSH has been setup to allow two temporary users to log in: lou, the account that will "su" to root and audit, the normal user account created for testing normal user access, although no users will be setup to actually remotely connect to this server for anything but email.
    - gcc and compilers were installed to compile certain security tools.
    - An Everything backup has been run, and backups have been scheduled through the GD Webtool consistent with the policy of the site (In this case, a scheduled daily incremental backup).
    - Tripwire has been run at the end of the initial configuration.
- A custom script has been run to collect much of the data. The generated files that are accessed are in the notes field of the audit. This is for the convenience of the auditor, but these files can be checked without the script as well (see Appendix B). In the interest of space I will attach the files to this document rather than explicitly list the results.

Engarde Secure Linux Audit Checklist

Date: **February 4, 2002**

Section I: OS/Kernel

Auditor: Lou Rabon

Item	Pass	Fail	Notes
a. Package & Kernel versions are current	X		rpm.found
b. Are the critical filesystems separated from the user partitions	X		df.found
Wrap Up	X		
SECTION SCORE	**X**		

No problems found.

Section II: Boot Security

Item	Pass	Fail	Notes
a. Is there an opportunity to boot into modes other than default?		X	lilo.conf
b. Is CTRL-ALT-DELETE possible?	X		shutdown.allow
c. Does only root have access to the /etc/lilo.conf?		X	lilo.stat
d. Can lilo.conf be written to by any user?	X		
e. Can normal users access /etc/rc.d/init.d	X		initd.stat
f. Check issue commands for legal disclaimers	X		issue, issue.net
g. Check the /etc/security/console.apps directory	X		consoleapps.found
h. Make sure shutdown programs are only executable by root		X	shutdown.stat
Wrap Up	X		
SECTION SCORE	**X**		

This section receives a passing score although there are a few facts that could be potential problems. If someone other than the administrator were to have physical access to the server, they could reboot it and disable LIDS by booting into standard mode without a password, but there's not much else that they'd be able to do without an exploit or the root password.

Section III: File System

Item	Pass	Fail	Notes
a. Is there an /etc/exports file ?	X		exportfs.found
b. Are the /tmp and /home directories protected through /etc/fstab ?		X	fstab.found
c. Can anyone other than root execute the rpm binary?	X		
d. Find all SETUID and SETGID files		X	setuid.found
e. Find hidden or unusual files	X		hidden.found
f. Find group and world-writable files & directories	X		readwrite.found

g. Find unowned files	X		unowned.found
h. Find .rhosts files	X		rhosts.found
i. Ensure logs in /var/logs are only writable and readable by root	X		varlog.stat
j. Ensure only root has read access to libwrap configuration files	X		etchosts.stat
k. Is the /tmp directory mode correct?	X		tmp.stat
l. Is umask for all users 027 or 077?		X	profile
m. Check for normal files in /dev		X	normaldev.found
n. Check for “special” files outside of /dev	X		specialdev.found
Wrap Up	X		wrapup.secIII
SECTION SCORE	**X**		

The /home and /tmp directories are not protected through fstab, which is an added measure of security to prevent setuid/setgid and executables from being used. The umask is also not “paranoid” mode, but every attempt I made as a normal user to execute files was unsuccessful, hence the passing grade.

Section IV: Users

Item	Pass	Fail	Notes
a. All local logins disallowed?	X		access.conf
b. Check for shadowed passwords	X		passwd
c. Check for blank passwords	X		shadow
d. Password length minimum length 7 or more characters?	X		
e. Check login.defs permissions	X		login.defs
f. Check for unnecessary accounts		X	passwd – *lpr, operator, etc*
g. Are resources limited through limits.conf?		X	limits.conf –*no max stack size set*
h. Check console.perms for device permissions		X	console.perms – *user have access to floppy, etc*
i. Check /etc/profile page for command history limit		X	profile – *HIST=1000*
j. Ensure that PAM is enabled	X		pam.present
k. Check /etc/pam.d/login to ensure pam_limits.so is enabled	X		login
l. Check for the existence of pam_console.so in /etc/pam.d files		X	pam_console.found
m. Make sure .forward files are not writable by users		X	forward.found
n. Check TMOUT on root account	X		bashrc
o. Check TMOUT on user accounts		X	profile
p. Check HISTSIZE for root		X	bashrc
q. Are normal users prevented from starting Cron jobs?	X		crontab.stat
r. Is /etc/shells setup correctly?	X		etcshells.found
Wrap Up	X		wrapup.secIV
SECTION SCORE	**X**		

Despite the large amount of failing grades for this section, I still decided to give it a passing score, due to the fact that most of the problems created by these failing grades are minor, and they are covered by other security features of the OS. For instance, users have access to floppy drives, but since the physical security is assured and users are not allowed to log on, this becomes almost unnecessary. The same holds true for TMOUT values and HISTSIZE. These are important variables that lead to a highly secure server, but taking the role that this server is playing in this environment into consideration, I subjectively decided to pass this section.

Section V: General Network Services

Item	Pass	Fail	Notes
a. Is xinetd.conf protected?		X	*Mode=660*
b. Is xinetd starting on server initialization?	X		chkconfig.found
c. Are unnecessary services starting up?	X		netstat.listen, chkconfig.found, ps.found
d. Does lsof output match previous results from netstat and chkconfig?	X		lsof.listen
e. Is xinetd starting any unnecessary services?	X		xinted.conf, xinetd.dir
f. Is /etc/hosts.deny configured to deny all?	X		hosts.deny
g. Is /etc/hosts.allow allowing only trusted IP addresses?	X		hosts.allow
h. Is xinetd wrapping all hosted services?	X		xinetd.test
i. Is Linuxconf present?	X		linuxconf.found
j. Does /etc/host.conf have IP Spoofing protection?		X	hostconf.nospoof
k. Is a firewall installed?	X		ipchains.found
l. Is ICMP restricted through IP Chains?		X	ipchains.list
m. Are private network address spaces restricted?		X	ipchains.list, ipchains.test
n. Are all unspecified services denied by the firewall?		X	ipchains.list
o. Is /etc/sysctl.conf configured correctly?		X	sysctl.conf
p. Are ICMP broadcast requests disabled?	X		broadcast.icmp
q. Are all interfaces denying source routing?		X	sourceroute.icmp
r. Check for syncookie protection.	X		syncookie.found
s. Check for ICMP redirect capability.		X	redirects.icmp
t. Is IP spoofing enabled in rp_filter for all interfaces?		X	rpfilter.found
u. Are all spoofing, redirects, and source routed packets logged?		X	martians.found
v. Ensure IP Forwarding is disabled	X		ipforward.found
Wrap Up		X	wrapup.secV
SECTION SCORE		**X**	

This section receives a failing score because of the low priority given to firewalling and spoofing/DoS protection. According to the results of this checklist, this server is vulnerable to source-routed packets (see RFC 1122)[5] and IP spoofing. Plus it's relatively low level of firewalling and lack of log_martians means that most attempts will not be logged.

[5] Gerard Mourani, Securing and Optimizing Linux, RedHat Edition, p. 62.

Section VI: Specific Services

Item		Pass	Fail	Notes
Guardian Digital Webtool				
	a. Is the administrative password different that the root password?	X		
	b. Are there specified administrative IP addresses?	X		
SSH				
	c. Ensure /etc/hosts.allow has the necessary IP addresses	X		
	d. Ensure root cannot login through ssh	X		
	e. Check /etc/ssh/sshd_config settings		X	*Permit root login=yes*
SMTP				
	f. Is relaying denied?	X		
	g. Is the VRFY command disabled?	X		vrfy.check
	h. Is postfix chrooted?	X		mastercf.check
POP3				
	i. Is POP3 only listening on port 995 ?	X		netstat.pop3
	j. Is SPOP3 wrapped by TCP Wrappers?	X		
NTP				
	k. Is NTP chrooted?		X	chroot.list, chroot.xntpd
Wrap Up		X		wrapup.secVI
SECTION SCORE		**X**		

In their current state, the services running on this server are secure. Although root is permitted to login to SSH by default, the fact that a certificate needs to be generated through the webtool makes it a minimal risk. NTP should be chrooted, but this does not present a serious security risk at the present time, although it is something I would recommend the administrator of this site do to ensure security.[6]

[6] See "Developers Comments" in Appendix C.

Section VII: Logging, Monitoring and Backups

Item		Pass	Fail	Notes
Logging & Monitoring				
	a. Is Tripwire installed and configured correctly?	X		
	b. Are tripwire logs being sent to the administrator?	X		
	c. Are Tripwire reports scheduled?	X		crontab
	d. Are LIDS alerts configured?		X	
	e. Are logs being rotated?	X		logrotate.found
	f. Is the administrator receiving a copy of the rotated logs?	X		logs.email, logsum.date
Backups				
	g. Have Backups been scheduled and configured?	X		
Wrap Up			X	wrapup.secVII
SECTION SCORE			**X**	

It's great that backups and monitoring are an integral part of ESL, but the level of logging that ESL gives is not adequate to successfully determine if the server is the target of an attack. There is no option in the webtool to set up alerts, or to customize the logging level. Again, this is minimal because the logging is happening to some extent, it just takes a more experienced administrator to figure out what is happening and configure the alerts correctly.

II. Evaluation of the audited system:

After auditing this system as thoroughly as I could, I have to admit I'm very impressed with the default security that ESL ships with. Many of the shortcomings of typical out-of-the-box *nix installs are addressed in this distribution. I would give this system a passing grade, and simply instruct the admin to tighten just a few things up.7 That being said, however, there are a few shortcomings that can be addressed:

- ESL could have included better firewalling.
  - As stated in the audit notes, the default firewall is minimal.
- Better logging should be included.
  - Currently, logging is present, but there are circumstances, as stated in the audit notes, when potential attacks will not be logged.
- Logsum should include more info
  - Logsum is an excellent custom script that parses the logs and sends information, but the information it currently reports on is minimal; more kernel-level messages should be summarized.[8]
- LIDS should protect more services and setuid/setgid programs.
  - LIDS is protecting some files, but it could be used to protect more setuid files and directories such as /tmp to some extent.
- fstab should be locked down further.
  - the /etc/fstab file is not restricting any filesystems.[9]
- Portsentry could be added for more security
  - ESL comes with excellent utilities, but one that is absent is portsentry (http://www.psionic.com/products/).

III. Evaluation of the audit:

There are definitely some other weaknesses that I would fix going forward. Although I obviously feel this is a solid, thorough audit, the following could have been done to ensure an even better picture of the security of this server:

- While I feel this audit should ensure protection from all but the most persistent attackers, I could have spent more time attacking the file system with buffer overflows and using less-popular hacking tools and perl scripts, etc, since the audit used well-known tools that do a good job but don't always find obscure exploits.
- The built in Network Intrusion Detection system was not audited (SNORT). If this server had two interfaces and/or it was configured for a different environment, Section V would likely have received a better grade. **It's essential to note that this audit doesn't apply to all environments.**
- There were many subjective measurements in the audit even though an interview was done to minimize these. I've attempted to cover the subjectivity through the use of tools that would uncover weaknesses. Less subjectivity could have been achieved, however, by defining a thorough security policy.

[7] See "Developers Comments" in Appendix C.
[8] See "Logsum email" section in /audit.results/wrapup.secVII
[9] See Appendix A for recommended fstab settings.

Appendix A:

Directory Structure Recommendations from the Linux Administrator's Security Guide, *by Kurt Seifried: http://www.seifried.org/lasg/*

LASG Directory structure chart

Dir.	nodev	noexec	nosuid	read-only	separate	comments
/	yes	yes	yes	yes	good idea	Ideally you should mount / totally restricted and then have directories like /boot/ separate, this also forces you to configure the directories properly since any "dangerous" directory (for example /dev/) will be "broken" (i.e. nodev would severely break /dev/). This is only recommended if you are going all out.
/boot/	yes	yes	yes	ok	good idea	Critical directory with kernel images, if an attacker replaces your kernel or deletes it you have a lot of problems.
/bin/	yes	no	no	ok	tricky	Directory with important system binaries, do not mount noexec or nosuid, your system will not work correctly. Mounting read-only will prevent trojans, and make updating software significantly more difficult.
/dev/	no	yes	yes	no	yes	Mounting /dev/ with the nodev option will severely break your system, as will mounting it read only. /dev/ is usually less then a few megs, and the ability to write to device files can result in huge damage, and system compromise.
/etc/	yes	yes	yes	no	tricky	Critical directory with system configuration information, usually the first target for an attacker. There should be no binaries in here (although some Unix systems do keep binaries in /etc/, Linux is not one of them). Mounting it read only will not allow you to change passwords, or other important settings so is not recommended.
/home/	yes	good idea	yes	no	yes	/home/ is the primary area where users keep their files and work with them (assuming they can log in), if you provide services like IMAP this is where their mail folders will be. You should make it a separate partition since users have a tendency of eating up space rapidly, as well it will prevent them from making hard links to files and then using setuid programs that dump core for example and wiping out system files. Mounting it noexec is probably a good idea, however depending on the type of work users do it may seriously hamper them, mounting it nosuid is a good idea and shouldn't really affect users.
/lib/	yes	no	yes	ok	good idea	Most programs will rely on libraries in this directory, if they are damaged or compromised you will have a hard time cleaning up. There are executable files in here (.so's, etc.), so setting it noexec would not be advised, but setting it nosuid is probably wise.

Dir.	nodev	noexec	nosuid	read-only	separate	comments
/mnt/	yes	good idea	good idea	ok	no need	/mnt/ is typically used to mount removable devices, such as /mnt/floppy/ or /mnt/cdrom, as such it rarely contains anything other then a few directories, making it separate is not a real issue since anything in it will be mounted as well.
/opt/	yes	no	no	no	good idea	Typically this directory is used for optional packages, vendor software and the like, oftentimes this stuff needs setuid bits to work properly (a good reason to separate it since a lot of vendors have terrible software security).
/proc/						/proc/ is a virtual filesystem
/root/	yes	no	no	no	good idea	root's private playground usually, many people use it instead of /usr/local/ when testing things/etc
/sbin/	yes	no	no	ok	tricky	Directory with other important system binaries, do not mount noexec or nosuid or you will break your system. Mounting read-only will prevent trojans, and make updating software significantly more difficult.
/tmp/	yes	yes	yes	no	yes	Temporary directory for use by users and system, mount read only will break things, make it separate because many exploits consist of making hard links in tmp to files, and then having a program misbehave and destroy/modify the target file maliciously. Binaries, especially setuid binaries should not be allowed in /tmp/ since any user can modify them usually.
/usr/	yes	no	no	ok	good idea	This directory is where the majority of software will be installed, along with source code and other stuff typically, mounting it separately is a good idea since it tends to contain relatively important software (especially in /usr/bin and /usr/sbin). Mounting it read only will prevent an attacker from inserting trojan software, but will make upgrades significantly harder. I wouldn't bother mounting it read only unless you also mount /bin/ and /sbin/ read only.
/var/	yes	yes	yes	no	yes	/var/ is used for a lot of things, least of which includes system logging. This partition should be separate since attackers can attempt to fill it up by flooding the log files, and other user data is stored here, such as mail (/var/spool/mail usually). Software that stores data here includes: Mail servers (Sendmail, Postfix, etc.), INN (Usenet news), Proxy software like Squid (WWW/FTP proxy), and so on. There should be no binaries at all here, just log files and data. Setting it noexec may break programs, Red Hat 7.0 places various binaries used for anonymous ftp with WuFTPD and arpwatch binaries in /var/ for example. You can place those files on another partition and symlink the directories to within /var/.

Appendix B:
ESL Audit Script

```
#!/bin/bash
#
# Engarde Secure Linux Audit script by Lou Rabon
# (lrabon@netstream.ws)
#
# Use this script in conjunction with my SANS
# GIAC certification paper (Lou_Rabon_GSNA.doc).
#
# Usage:
# [root@sparky name]# chmod 700 ESL.audit
# [root@sparky name]# ./ESL.audit
#
# Script assumes /home/audit is created.  I've made it
# easy to change this.  Just change the first line and
# the last 3 lines.
#
# Results will be compressed in a tar file and placed
# in the /var/BACKUP directory and the files will be
# deleted from the /home/audit directory.
# ***IMPORTANT: Since there are many sensitive system
# files in this archive, dispose of it immediately
# after use
#
cd /home/audit
rpm -qa > rpm.found
df > df.found
cp /etc/lilo.conf .
cp /etc/inittab .
stat /etc/lilo.conf > lilo.stat
stat /etc/init.d > initd.stat
cp /etc/issue* .
ls /etc/security/console.apps > consoleapps.found
stat /sbin/reboot >> shutdown.stat; stat /sbin/halt >> shutdown.stat; stat /sbin/shutdown >>
shutdown.stat
find / -name export* -print > exportfs.found
cat /etc/fstab > fstab.found
stat /bin/rpm > rpm.stat
find / -type f \( -perm -04000 -o -perm -02000 \) \-exec ls -lg {} \; > setuid.found
find / -name ".. " -print -xdev >> hidden.found; find / -name ".*" -print -xdev | cat -v >> hidden.found
find / -type f \( -perm -2 -o -perm -20 \) -exec ls -lg {} \; >> readwrite.found
find / -type d \( -perm -2 -o -perm -20 \) -exec ls -ldg {} \; >> readwrite.found
find / -nouser -o -nogroup > unowned.found
cp /etc/security/access.conf .
cp /etc/passwd .
cp /etc/shadow .
cp /etc/login.defs .
cp /etc/security/limits.conf .
cp /etc/security/console.perms .
cp /etc/profile .
ldd /bin/login > pam.present
cp /etc/pam.d/login .
find /etc/pam.d/* -exec grep -q 'pam_console.so' {} \; -print > pam_console.found
find / -type f \( -perm 600 \) -name .forward -exec ls -lg {} \; > forward.found
cp /root/.bashrc ./bashrc
stat /etc/crontab > crontab.stat
cat /etc/shells | grep '/dev/null' > etcshells.found
stat /etc/shells >> etcshells.found
stat /etc/xinetd.conf > xinetd.stat
chkconfig --list | grep 'on' > chkconfig.found
lsof -i > lsof.listen
netstat -l > netstat.listen
ps -ax > ps.found
cp /etc/xinetd.conf .
ls /etc/xinetd.d > xinetd.dir
find / -name linuxconf -print > linuxconf.found
cat /etc/host.conf | grep 'nospoof' > hostconf.nospoof
ls /proc/net | grep 'ip_fwchains' > ipchains.found
ipchains -L > ipchains.list
ipchains -C input -i eth0 -p tcp -s 192.168.2.12 4353 -d 1.2.3.10 80 > ipchains.test
cp /etc/sysctl.conf .
cat /proc/sys/net/ipv4/icmp_echo_broadcast_requests > broadcast.icmp
find /proc/sys/net/ipv4/conf/*/accept_source_route -exec cat {} \; -print > sourceroute.icmp
cat /proc/sys/net/ipv4/tcp_syncookies > syncookie.found
find /proc/sys/net/ipv4/conf/*/accept_redirects -exec cat {} \; -print > redirects.icmp
find /proc/sys/net/ipv4/conf/*/rp_filter -exec cat {} \; -print > /home/audit/rpfilter.found
find /proc/sys/net/ipv4/conf/*/log_martians -exec cat {} \; -print > martians.found
cat /proc/sys/net/ipv4/ip_forward > ipforward.found
cp /etc/services .
```

```
cp /etc/hosts.* .
cp /etc/ssh/sshd_config .
cat /etc/postfix/master.cf | grep 'smtp' > mastercf.check
cat /etc/postfix/main.cf | grep 'vrfy' > vrfy.check
netstat -ant > netstat.pop3
ls /var/chroot > chroot.list
cat /etc/init.d/xntpd | grep 'chroot' > chroot.xntpd
ls -l /var/lib/tripwire/report > tripwire.logs
cp /etc/crontab .
find /etc -name logrotate -print > logrotate.found
ls -l /var/log/audit/daily* > logsum.date
cat /etc/logsum.conf | grep 'EMAIL =' > logs.email
tar -cf /var/BACKUP/audit.results.tar /home/audit/*
chmod 600 /var/BACKUP/audit.results.tar
rm -rf /home/audit/*
```

Appendix C:
Comments from the developers

To: Lou Rabon
From: Dave Wreski
Date: Tuesday, February 05, 2002 9:29 PM
Subject: Re: EnGarde audit

Hi Lou,

I've taken some time and analyzed the report you sent over for submission to SANS. As I mentioned previously, it is clear you've spent quite a bit of time auditing EnGarde, but there are some unfortunate technical inaccuracies that need to be addressed.

I actually spent about a combined two hours reading through it, speaking with our engineers, and writing up our findings. It is at this point that I stopped.

It became clear early on that while you had the best intentions, it was evident you may not had enough information to authoritatively form an educated report on the potential security issues with EnGarde.

It very much seems as if you relied heavily on Internet-based references without having the experience or time to do the necessary research. For example:

- COPS and Tiger are at least five years old, and have long been considered reference-only security tools that no longer serve any practical purpose.

- Some ideas such as mounting separate partitions read-only, or nosuid, etc, may on face value seem like good ideas (in fact they are), the system widely becomes unusable once this is done. Creating separate /bin, /sbin, or even /etc partitions simply will not work (mount, for example, would be located on a partition that hasn't yet been mounted). Additionally, if not implemented correctly or relied upon without understanding the implications, the opposite affect may occur:

http://security-archive.merton.ox.ac.uk/bugtraq-199908/0361.html

- Some of the checks you've used to indicate secure/insecure configurations are for specific situations and environments independant of EnGarde. Others are grossly out of date and don't apply to EnGarde. For example, the $HOME/.forward being immutable was a specific reference in the Unix Security Checklist written years back that applied to a version of sendmail that was vulnerable at the time.[10]

- The file and group permissioning structure that we've developed was done such that normal users could share files with each other, but weren't permitted to write to each others files. The alternative (umask 077), invariably leads to "Why can't I access my own files through the Web site?" or "Oh, can't read my file? I'm not in the office, here's my password." Also, each individual network service has its own set of ACLs that are enforceable by the administrator. An expectation is made that people that are able to access the system at the console (remotely or locally) are 1) only able to write to areas to which only they have

[10] This was fixed after these comments.

access; and 2) understand the implications. More in the next comment.

- References to security issues on files that are mode 0640 instead of 0600 are done that way for a specific reason. For example, the notes regarding the floppy being accessible by normal users is completely wrong. Only if the normal user is added to the floppy group by an administrator after installation. If it were not for the 0640 mode and group 'floppy', the mount command would be required to be set-uid -- a far greater security threat!

- The use of the chroot function is greatly misunderstood. I've attached an email message that passed through bugtraq just today about one such example. This "Secure Linux" is anything but secure, but unfortunately still gets to use words like "Secure" and "Firewall" in their product names. Chrooting all services is a horrible idea and only serves to add to administration and security complexity -- the exact opposite of what you would expect to achieve. The short of it is that they are extremely difficult to be done correctly, vary according to every service, have multitudes of ways of be escaped ("jailbreak"), among others. All daemons that provide integrated chroot functionality (snort, bind, postfix, and others) have been implemented, while others have been thoroughly audited or run with only the privileges they require.

- Likely due to a lack of experience, but the comments that were noted about ICMP being restricted and hosts.conf lacking IP spoofing protection are completely wrong. ICMP DoS protection is enabled at the kernel level through the /proc interface. IP spoofing is configured similarly at the kernel level. The box is also a quite secure firewall -- no protocols are permitted to be forwarded from one network to another without administrative intervention. I believe many of the books you've read and websites you've visited lead you to believe it was necessary to have extensive ipchains rules to create a secure firewall. While EnGarde isn't intended to become a drop-in firewall without creating a security policy, it is indeed a very robust Workgroup gateway firewall. On the contrary, there is a tendancy to take a large amount of firewall rules for granted, and assume the server is secured when in fact that very assumption is what makes it insecure.[11]

- The entire local security section is basically wrong. There is not a way for a normal user to access the system without an administrator having already been logged in before leaving for his coffee break. Unless they hit The Big Red Switch, ctrl-alt-delete won't even work for them, despite your note to the contrary.[11]

You've provided a great value, but it might be a good idea to take a step back and re-evaluate this information to SANS as an authoritative analysis of EnGarde. I would expect their seasoned security professionals to also point out some of the inaccuracies due to lack of a more thorough evaluation.

The greatest thing about EnGarde is that with tools such as LIDS, and the solid foundation upon which it was developed, enables administrators to build a system that specifically applies to their environment. In our lab we have designed such a system that meticulously analyzes each system call a particular daemon may perform, and only granting it access for its pre-defined specific purpose. It makes a horrible box for an ISP with normal users, though.

[11] This was also fixed after this these comments.

Hopefully my comments above are interpreted as constructive. I believe
we would actually be very interested in assisting you more closely as
you go through the audit. Some of the points you have raised are already
included in the next version, while others were oversights that only
another set of eyes would catch.

Finally, I've attached a PDF that I wrote some time ago that more
thoroughly discusses the security features that EnGarde employs. I think
you'll find this useful.

Looking forward to hearing from you.
Best regards,
Dave

--
Dave Wreski
Corporate Manager Guardian Digital, Inc.

References:

Apache.Org Homepage. 20 December 2001. The Apache Software Foundation. "Security Tips for Server Configuration." <httpd.apache.org/docs/misc/security_tips.html>

Arkin, Ofir. "ICMP Usage in Scanning." <www.sys-security.com/archive/papers/ICMP_Scanning_v2.5.pdf>

Borland, Matt. "Locking Down Your Daemons: An Overview of 'chroot jailing' Services in Linux." <rr.sans.org/linux/daemons.php>

CERT Homepage. 20 November 2001. Computer Emergency Response Team. "Unix Security Checklist v2.0." <http://www.cert.org/tech_tips/usc20_full.html>

Guardian Digital*. EnGarde Secure Linux User Manual.* 2002.

Hatch, Brian. "An Overview of LIDS." <www.securityfocus.com/infocus/1496>

Laude, Mary A. "Auditing RedHat Linux 7.0 (workstation)." SANS Institute Practical for GSNA Certification. 10 August 2001. <www.giac.org/practical/Mary_Laude_GSNA.zip>

Mourani, Gerhard. *Securing and Optimizing Linux: RedHat Edition*. Open Network Architecture and OpenDocs Publishing, 7 June 2000.

Naidu, Krishna. "Web Application Checklist." <www.sans.org/checklist/web_check.htm>

Naidu, Krishni. "Auditing Linux." <www.sans.org/checklist/linux_check.htm>

Seifried, Kurt. "Linux Administrators Security Guide." <www.seifried.org/lasg/>

Simes. "How to Break Out of a chroot Jail." <www.bpfh.net/simes/computing/chroot-break.html>

Tarreau, Willy. "Security under Linux: the Buffer Overflow Problem. <http://www.miaif.lip6.fr/willy/security/linux.html>

volatile. "Setuid/Setgid tutorial." <neworder.box.sk/newsread.php?newsid=2380>

Wreski, Dave & Christopher Pallack. "Network Intrusion Detection Using Snort." <www.linuxsecurity.com/feature_stories/feature_story-49.html>