

AUTOMATED SECURITY HARDENING OF RED HAT ENTERPRISE LINUX V5 IN ACCORDANCE WITH DISA STANDARDS



Scott C. Zimmerman, CISSP

CSC Identity Labs

szimmerm@csc.com

CSC Papers

2011

Keywords: Red Hat Enterprise Linux, RHEL, Department of Defense, DoD, Defense Information Systems Agency, DISA, DIACAP, C&A, Accreditation, Script, Hardening

ABSTRACT

While use of Red Hat Enterprise Linux (RHEL) is becoming widespread within the U.S. Department of Defense (DoD), there is a dearth of effective products — COTS or otherwise — to facilitate the securing of RHEL systems in accordance with Defense Information Systems Agency (DISA) standards. An Information Security Engineer Leader with CSC's Identity Labs developed a tool that would allow a system administrator to automate as much of the hardening process as possible — to bring a typical RHEL v5.X server into compliance with DoD Information

Assurance standards, quickly, efficiently and successfully. Following an external security audit, DoD auditors praised the project for the extremely low number of findings on the RHEL systems, stating that they had never before seen such a new and complex system perform so well on their tests. This paper, which describes the project and tool, and associated scripts, should be useful to those responsible for achieving DIACAP accreditation of RHEL systems.

INTRODUCTION

Currently the use of Red Hat Enterprise Linux (RHEL) is becoming more widespread within the Department of Defense. While several tools are available for securing Microsoft Windows™ systems, there is a genuine dearth of products available – COTS or otherwise – to facilitate securing RHEL systems in accordance with DISA standards. The team tested the leading COTS security hardening product, at a cost of \$360 per server, and found that it did not find or remediate issues effectively during testing. As a result, a Security Engineering Lead with CSC's Identity Labs began to develop a tool in-house which would allow a system administrator to automate as much of the hardening process as possible.

The goal of the development was to bring a typical RHEL v5.X server into compliance with the DISA Unix Security Checklist and with the DISA Security Readiness Review (SRR) script. The first iteration of the tool – called *h-script* – was fairly simple and addressed only file ownership and permissions. Subsequent versions of the tool introduced the replacement of configuration files and the addition/removal of software packages, users, and services. Eventually the *h-script* functionality was integrated with the Red Hat Satellite server deployment solution. The combination of Satellite, *h-script*, and other functionality allows the CSC Identity Labs engineers to deploy a DISA-compliant RHEL Operating System on server blade hardware **in under twenty (20) minutes.**



THE RESULTS

Assuming a deployment of fifty (50) RHEL servers (and assuming the RHEL licenses have already been purchased):

- Using the scripted approach by itself will cost \$0 and will result in a DISA-compliant security posture, greatly facilitating DIACAP accreditation. Using the COTS tool will cost \$18000 (\$360 per server for 50 servers) and will result in a non-compliant security posture.
- Using the scripted approach in conjunction with Red Hat Satellite will cost \$9000. **This represents a CAPEX savings of 50% over using the COTS product.**

Using the scripted approach in conjunction with Red Hat Satellite will take 12.5-16.7 labor hours; the manual process will take 300-400 labor hours. **This represents a labor savings of approximately 96% over using a manual process.**

	Manual hardening	COTS tool	Automated hardening
Time/labor	6-8 hours per system	4-5 hours per system	15-20 minutes per system
CAPEX	\$0	\$360 per system	\$0
Effectiveness	Varies by individual	Large number of critical findings	<10 Total findings

Table 1: Comparison of Manual Hardening, COTS tool, and Automated Hardening

The results of the automated security hardening process were very encouraging. Here are the SRR results from a typical Domain Name Server which is placed in a Demilitarized Zone (DMZ):

Finding Counts – Security Readiness Review (SRR) output

CAT I = 1/177 (number of findings/number of items checked by SRR)

CAT II = 1/368

CAT III = 0/63

CAT IV = 0/0

Category I findings are the most critical. The single finding in this case was a false positive.

Category II findings are slightly less critical than Category I but are still very important. The single finding in this case was the result of the Linux system not being equipped with anti-virus software; the finding would be mitigated to a Category III and thus would not interfere with DIACAP accreditation.

CONCLUSION

The automated hardening process – in conjunction with Red Hat Satellite – is much less expensive, much faster, and much more effective than using a COTS security hardening product or performing all security hardening tasks manually.

The tools described in this paper can be used to harden RHEL to meet Department of Defense and DISA standards in a consistent, automated, and highly effective manner.



PART I — BACKGROUND

Red Hat Enterprise Linux – hereafter RHEL – is a commercially-supported Linux Operating System (OS) that has gained considerable ground in the data center in recent years. As a company, Red Hat provides a licensing plan, a patch/update repository, technical support, training, and other items associated with a shrink-wrap software vendor.

Because of the commercial support aspects of RHEL, it has become more widely used to run servers within the Department of Defense (DoD) IT community; these RHEL servers must be hardened to meet DoD standards before they go into production. Information security standards within the DoD are set by the Defense Information Systems Agency, or DISA. DISA provides a range of security guidance documents and tools to help constituents achieve compliance with their security standards. Chief among the tools available for Linux and Unix systems are the **Security Checklists** and the **Security Readiness Review (SRR)** scripts. The Checklists are used to guide the configuration of the system, and the SRR is used to verify the configuration of the system.

It is important to note that while the DISA tools do a thorough job of indicating areas that are out of compliance, they do **not** automatically rectify any issues that are discovered: all remediation must be completed by hand. This can be very time-consuming, especially when the server farm contains over fifty RHEL systems – unless the process is automated.

Note: the CSC team conducted preliminary testing using the leading COTS hardening product. During testing the product was found to be ineffective at both identifying and remediating issues, leaving a great deal of work to be done manually.

PART II — THE CASE FOR AUTOMATION

The Unix Security Checklist used for this project is named Unix-Sec3-081509.doc, and it was downloaded from <http://iase.disa.mil>. This Checklist is four hundred and fifty-eight (458) pages long and describes a wide variety of tasks that a RHEL administrator will need to complete in order to make his system(s) ready to run the Security Readiness Review script.

During review and application of the Checklist, it became apparent that there were many individual line items that dealt with **file permissions** and **file ownership**. Because these items can be addressed easily from the command line, they were recognized as ideal candidates for automation. The first step was to parse the entire Checklist document and identify all line items that involved file permissions.

Here is an example of a typical permission-related Checklist item:

GEN001380 – /etc/passwd File Permissions

Check /etc/passwd permissions:

ls -l /etc/passwd

If /etc/passwd is more permissive than 644, this is a finding.

PDI:	GEN001380 V0000798	Category:	II	Status Code:	AUTO	Previously:	G048
MAC/Confidentiality Levels:	MAC I – CSP, MAC II – CSP, MAC III – CSP						
IA Controls:	ECCD-1, ECCD-2						
PDI Description:	The /etc/passwd file is more permissive than 644.						
Reference:	UNIX STIG: 3.4						
<input type="checkbox"/> Open <input type="checkbox"/> Not a Finding <input type="checkbox"/> Not Applicable <input type="checkbox"/> Not Reviewed	Comments:						



Note that the Checklist does not indicate definitively that the permissions are set incorrectly: it states what the settings should be to avoid having this particular item come up as an issue when the SRR is run. The SRR will evaluate the existing settings and return a result.

Here is the salient point in this Checklist item:

If /etc/passwd is more permissive than 644, this is a finding.

This indicates that any values greater than 644 in the permissions field will be flagged by the SRR as a security issue; conversely, if the permissions are set explicitly to 644 – or less – the item will **not** be flagged as an issue by the SRR. For example, permissions of 640 would be acceptable but 755 would not.

The root user – the privileged system administrator – can change the permissions on this file manually from the command line, and the command would look like this:

```
linuxbox#>chmod 644 /etc/passwd
```

Any such command can easily be placed into a shell script. A shell script is a particular kind of text file that contains individual commands to be executed in order on a Unix system; on other systems it might be called a 'batch file'. If the user – e.g. root – has sufficient privileges, he can run a script to set permissions and perform other tasks automatically instead of doing them by hand.

Automating a process provides greater consistency and repeatability of results compared to making the changes manually.

The second focus area in the Checklist is **file ownership**. As with file permissions, file ownership attributes can be set from the command line.

Here is an example of a permissions check contained in the Checklist:

GEN001400 – /etc/passwd and/or /etc/shadow File Ownership

Check /etc/passwd ownership:

```
# ls -l /etc/passwd
```

[HP-UX and AIX instructions removed for brevity]

If the /etc/passwd and /etc/shadow (or equivalent) file is not owned by root, this is a finding. If HP-UX /tcb directories and files ownerships are not configured as detailed above, this is a finding.

PDI:	GEN001400 V0000797	Category:	II	Status Code:	AUTO	Previously:	G047
MAC/Confidentiality Levels:	MAC I – CSP, MAC II – CSP, MAC III – CSP						
IA Controls:	ECCD-1, ECCD-2						
PDI Description:	The /etc/passwd and /etc/shadow (or equivalent) file is not owned by root.						
Reference:	UNIX STIG: 3.4						
<input type="checkbox"/> Open <input type="checkbox"/> Not a Finding <input type="checkbox"/> Not Applicable <input type="checkbox"/> Not Reviewed	Comments:						

The salient point in this item is here:

If the /etc/passwd and /etc/shadow (or equivalent) file is not owned by root, this is a finding.

As with the file permissions example above, the root user can change the ownership of this file manually from the command line; the command would look like this:

```
linuxbox#>chown root:root /etc/passwd
```

This command will ensure that the `/etc/passwd` file is owned by the user `root` in the group `root`. This type of command also lends itself to automation and can easily be included in a shell script.

PART III — THE BIRTH OF H-SCRIPT

After the roster of Checklist items to automate was complete, the first version of h-script was born: this version of the script addressed only file permissions and ownership. The full list of commands contained in version 1.0 of h-script can be found in Appendix A. Note that all of the `chmod` commands use the `-R` switch. This guarantees that all directories in the stated path will have the command applied to them recursively, meaning that the command will be run on each subdirectory in turn until all subdirectories have been completed. Applying a recursive command to a file will correct that file and will not generate an error; essentially the command successfully completed a recursive action on zero subdirectories. (The `-R` switch was used universally simply for expediency.)

Recall that the SRR will parse the attributes for the file(s) in question and compare the value(s) to what it expects to see. The functionality of h-script is simpler than that: h-script ignores the existing settings and explicitly sets the attributes to acceptable values.

Because RHEL is an enterprise OS, it is possible (or even probable) that some file permissions and ownership attributes were set to DISA specifications by default and need no modification. However, in the interest of efficiency, h-script explicitly sets all parameters to the required values regardless of their current state: it would take longer to parse the file attributes, make a decision, and act on the decision than it would take simply to set the attributes correctly.

Version 1.0 of h-script ran flawlessly and was quite useful as a proof-of-concept, but there were more Checklist items to automate.

PART IV — H-SCRIPT GAINS FUNCTIONALITY

Any good security hardening process includes removing unnecessary access and maintaining careful control over user accounts. Version 2.0 of h-script removed a fairly long list of users, shown here:

lp	sync	shutdown
halt	news	gopher
operator	games	mail
uucp	ftp	netdump
adm	pcap	avahi-autoipd
sabayon		

Table 2. Users deleted by h-script v2.0

This portion of the script is included in full in Appendix B.

Version 2.0 of h-script also addressed another DISA requirement. DISA states that no unknown SUID and SGID files should exist on the system; any required SUID or SGID files must be approved and documented by the Information Assurance Officer (IAO). Some applications, such as Oracle 11G, include some SUID and SGID binaries so a blanket ban on such files is not always feasible. In any case, for security purposes it is desirable to be able to identify and locate any SGID/SUID files.

Here is the portion of the script which identifies and captures the listings of SUID files, SGID files, world-writeable files, and world-writeable directories:

```
#  
# The next four lines will find certain files and place the directory  
# listing in the files indicated. File names will contain the current  
# date in MM-DD-YYYY-HHMM format.  
find / -perm -4000 >> ./suid_files_`date +%m-%d-20%y-%H%M`  
find / -perm -2000 >> ./sgid_files_`date +%m-%d-20%y-%H%M`  
find / -type f -perm -002 >> ./world_writeable_files_`date +%m-%d-20%y-%H%M`  
find / -type d -perm -002 >> ./world_writeable_dirs_`date +%m-%d-20%y-%H%M`  
#
```

The *find* command in this case will start at the root filesystem and search every directory on the server for files that meet the stated criteria, e.g. permissions set to 4000. The redirects (>>) will cause the output of the *find* commands to be stored in text files instead of being displayed on the screen. All of the files created by the redirects will be placed into the same directory from which h-script was launched. Ideally these files will have sizes of zero bytes, meaning no SUID or SGID files were found, but as stated above this may not always be the case. The name of each backup file – as indicated in the script’s comments – will contain the date and time that the file was created. This allows h-script to be run many times without overwriting any of the previous files.

As above, instead of parsing the existing file content to gauge how “correct” it is, the more direct approach is simply to replace the default RHEL files with DISA-compliant versions.

A careful review of the DISA Unix Checklist indicated that quite a few configuration files would be evaluated by the SRR, and most of them were in the /etc directory. A number of target files were selected, copied to a backup directory location, and modified offline to include all settings required by DISA.

Version 3.0 of h-script replaced the following files:

```
/etc/motd  
/etc/bashrc  
/etc/hosts.allow  
/etc/hosts.deny  
/etc/sshd/sshd-config  
/etc/audit/auditd.conf  
/etc/audit/audit.rules  
/etc/login.defs  
/etc/shells  
/etc/pam.d/passwd  
/etc/pam.d/system-auth  
/etc/securetty  
/etc/syslog.conf  
/etc/security/opasswd  
/etc/sysctl.conf  
/etc/xinetd.conf  
/etc/inittab  
/etc/grub-menu.lst/etc/shadow
```

It is very important to note that version 3.0 and up of h-script should be run only on newly-installed systems: the tool will replace many configuration files and any previous customization will be overwritten.

Because the DISA requirements are very configuration-oriented, the changes in this version take place solely in the `/etc` directory. While the file-replacement portion of `h-script` is contained in Appendix C, here is a representative snippet:

```
#  
echo "Replacing /etc/securetty..."  
/bin/cp /etc/securetty /etc/securetty_`date +%m-%d-20%y-%H%M`  
/bin/cp ./etc-securetty /etc/securetty  
#
```

The `echo` statement prints a line of text to the screen so the administrator can see what the script is doing. The first `/bin/cp` line backs up the existing file to a dated and timestamped version in the same directory; this allows an administrator to undo a configuration change by copying the most recent backup file over the current file. Lastly, the second `/bin/cp` line copies the DISA-compliant version of the file into the correct location. Because `h-script` is designed to be run by root, the copy operations initiated by the script take place without interruption: the user will not be prompted to confirm the file overwrites.

PART VI – POST-H-SCRIPT

The system hardening process must also address the issue of unnecessary packages and unnecessary services. As with user accounts, installed software and running services should be kept to an absolute minimum and tightly controlled: anything not required for the operation of the system should be removed or disabled. The DISA SRR will also generate findings based on running services, so it behooves the administrator to shut off and remove as much as possible while still permitting the server to function.

There is a secondary tool called `post-h-script` that addresses the installation and removal of packages as well as the state of running services. The complete `post-h-script` is contained in Appendix D; here is a representative snippet:

```
echo " "  
echo "Removing Samba"  
chkconfig smb off  
yum -y remove smb  
echo " "  
echo "Removing TFTP Server"  
yum -y remove tftp-server  
echo " "
```

The `chkconfig` commands are used to shut down services on a RHEL machine. In this example it is used to shut down Samba, which is the SMB service used to share resources with Microsoft Windows machines. The `yum` command is used to manage software installation, allowing packages to be added or removed via the command line. Here we see that the `smb` and `tftp-server` packages are being removed by `post-h-script`. The `tftp` server was not running by default and did not need to be shut down prior to removing the `tftp-server` package. The `echo` commands provide updates to the administrator about what the script is doing. (They also print blank lines to make the screen output easier to read.)

The next section of `post-h-script` disables unnecessary services. As mentioned above, running services should be kept to an absolute minimum to avoid potential security issues.

```
echo "Disabling Unnecessary Services"  
chkconfig anacron off  
chkconfig apmd off
```



chkconfig autofs off
chkconfig avahi-daemon off
chkconfig bluetooth off

Altogether post-h-script disables twenty-three separate services; the full list of these is in Appendix D. The advantage of this approach is that if a service has been disabled, the SRR a) will not flag any security issues or vulnerabilities related to the service and b) will not flag the service itself simply for running.

PART VII – RED HAT SATELLITE

The initial versions of h-script were developed on and applied to default, standalone RHEL systems that were installed manually from a DVD. As the script functionality grew, so did the need to test in an actual enterprise environment. This environment consists of Sun Microsystems x6270 server blades and chassis. The server blades are based on a set of core components, which allows the deployment of standardized system images using Red Hat Satellite. From <http://www.redhat.com>:

Red Hat Network (RHN) Satellite is a systems management platform that makes Linux deployable, scalable, manageable, and consistent. RHN Satellite provides administrators with the tools to efficiently manage their systems lowering per-system, deployment, and management costs. RHN Satellite offers superior security by having a single centralized tool, secure connection policies for remote administration, and secure content. Use RHN Satellite to ensure security fixes and configuration files are applied across your environment consistently.

The CSC team chose Red Hat Satellite as the deployment solution to allow fast, consistent, and remote system builds, OS patch and update installation, and – most importantly – custom security configurations on Red Hat servers.

A Red Hat Satellite OS deployment in the CSC environment follows this basic process:

- The operating system image and configuration files are stored on the deployment server.
- To install a new OS image, the administrator – via the blade management module – causes the blade server to boot from a small disk image provided by Satellite.
- The OS installation begins and is completed automatically.
- After the OS is installed, the administrator can initiate any post-processing.
- The Satellite administrator can re-deploy an image – or updates or configuration changes – at any time.

When the Operating System installation is complete, the Satellite server can replace any or all configuration files in the manner similar to that of h-script. One significant advantage of using Satellite over h-script is that Satellite supports multiple configurations. The CSC team created deployment and configuration paths for the following types of servers:

Email (SMTP)
Domain Name Service (DNS)
Lightweight Directory Access Protocol (LDAP)
Oracle 11G Database
Oracle WebLogic application server
Oracle Management Server (OMS)
Syslog
Base server image



Having several configuration paths may seem excessively complicated but it is desirable for one primary reason: **each server receives configuration files that are completely customized for each service.** For example, the SMTP server and the LDAP server will require the installation of different software packages and the use of different configuration files and settings, despite using identical hardware. Having all of the correct items and settings deployed automatically means that system administrators need to spend only a very small amount of time configuring the Operating System or service(s): as soon as the server is deployed it is nearly ready for production.

After the Operating System image has been deployed on the server, Satellite uses a tool called Cobbler to perform post-installation processing. It is at this point that the critical configuration files – previously replaced by h-script and post-h-script – will be installed on the target system. Once the configuration files have been updated, the server is now in compliance with the DISA Unix Security Checklist and is ready to be evaluated using the Security Readiness Review script.

Using Red Hat Satellite and custom scripts and configuration files developed in-house, the CSC team can deploy a fully hardened, operational, and DISA-compliant Red Hat Enterprise Linux server on standardized blade server hardware in under twenty (20) minutes.

FACTS AND FIGURES

Assuming a deployment of fifty (50) RHEL servers (and assuming the RHEL licenses have already been purchased):

- Using the scripted approach by itself will cost \$0 and will result in a DISA-compliant security posture. The COTS tool costs approximately \$360 per server: using the COTS tool will cost \$18000 and will result in a non-compliant security posture.
- Using the scripted approach in conjunction with Red Hat Satellite will cost \$9000. **This represents a CAPEX savings of 50% over using the COTS product.**
- Using the scripted approach in conjunction with Red Hat Satellite will take 12.5-16.7 labor hours; the manual process will take 300-400 labor hours. **This represents a labor savings of approximately 96% over using a manual process.**

The following table captures and compares the attributes of the various processes.

	Manual hardening	COTS tool	Automated hardening
Time/labor	6-8 hours (est.) per system	3-4 hours (est.) per system	15-20 minutes per system
CAPEX	\$0	\$360 per system	\$0
Effectiveness	Varies by individual	Large number of critical findings	<10 Total findings

Table 3. Comparison of Manual Hardening, COTS tool, and Automated Hardening

The results of the security assessment process were very encouraging. Here are the findings from a Domain Name Server which is placed in a Demilitarized Zone (DMZ):

Finding Counts – Security Readiness Review (SRR) output

CAT I = 1/177 (number of findings/number of items checked by SRR)

CAT II = 1/368

CAT III = 0/63



CAT IV = 0/0

Category I findings are the most critical. The single finding in this case was a false positive.

Category II findings are slightly less critical but are still very important. The single finding in this case was the result of the Linux system not being equipped with anti-virus software; the finding would be mitigated to a Category III and thus would not interfere with DIACAP accreditation.

Here are the findings from a typical email server, configured as the second node in a failover pair. The results are quite similar to those of the DNS server:

Finding Counts – Security Readiness Review (SRR) output

CAT I = 1/177

CAT II = 6/368

CAT III = 1/63

CAT IV = 0/0

The tools described in this paper – h-script, post-h-script, Red Hat Satellite, custom configuration files – were used successfully to prepare a newly-built enterprise environment for DIACAP accreditation. The entire environment, especially the Linux servers, passed the DIACAP assessment with flying colors.

CONCLUSION

The automated hardening process – in conjunction with Red Hat Satellite – is much less expensive, much faster, and much more effective than using a COTS security hardening product or performing all security hardening tasks manually.

The tools and processes described in this paper can be used to harden RHEL to meet Department of Defense and DISA standards in a consistent, automated, and highly effective manner.

FUTURE DEVELOPMENT PLANS

Currently an effort is underway to accomplish the same level of hardening without using Red Hat Satellite. The result so far is a very sizable script which replaces over fifty files and makes myriad other system changes. There are some technical impediments on this path but the CSC team is optimistic that these hurdles can be cleared.

FOR MORE INFORMATION

For more information, please contact Scott Zimmerman (szimmerm@csc.com).

APPENDIX A — FILE PERMISSIONS AND FILE OWNERSHIP

The initial version of h-script addresses file permissions and file ownership.

```
#  
# This section sets/resets permissions and ownership of critical files.  
#  
#  
chmod -R 755 /etc/init.d/*  
chmod -R 700 /root  
touch /etc/.login  
chmod -R 644 /etc/.login  
chmod -R 644 /etc/profile  
chmod -R 644 /etc/bashrc  
chmod -R 644 /etc/environment  
touch /etc/security/environ  
chmod -R 644 /etc/security/environ  
chmod -R 644 /etc/skel  
touch /dev/audio  
chmod -R 644 /dev/audio  
chmod -R 640 /var/log/*  
touch /etc/cron.allow  
chmod -R 600 /etc/cron.allow  
touch /etc/cron.deny  
chmod -R 700 /etc/cron.deny  
chmod -R 600 /var/spool/cron/  
chmod -R 600 /etc/cron.d/  
chmod -R 600 /etc/crontab  
chmod -R 700 /etc/cron.daily/  
chmod -R 700 /etc/cron.hourly/  
chmod -R 700 /etc/cron.monthly/  
chmod -R 700 /etc/cron.weekly/  
chmod -R 600 /var/log/cron  
touch /etc/at.allow  
chmod -R 700 /etc/at.allow  
chmod -R 755 /var/spool/at/spool  
chmod -R 700 /var/crash  
chmod -R 600 /etc/sysctl.conf  
chmod -R 755 /etc/xinetd.conf  
chmod -R 755 /etc/xinetd.d  
chmod -R 644 /etc/services  
chmod -R 700 /bin/traceroute  
chmod -R 640 /etc/syslog.conf  
chmod -R 600 /etc/grub.conf  
chmod -R 755 /usr/lib/*  
chmod -R 640 /etc/security/access.conf  
chmod -R 640 /etc/securetty  
chmod -R 644 /usr/share/man  
chmod -R 644 /usr/share/info
```



```
touch /usr/share/infopage
chmod -R 644 /usr/share/infopage
chmod -R 744 /selinux
chmod -R 744 /sys/class/scsi_host/*
#
#
# This section sets/resets file ownership.
#
#
chown root:root /etc/.login
chown root:root /etc/profile
chown root:root /etc/bashrc
chown root:root /etc/environment
chown root:root /etc/security/envIRON
chown root:root /dev/audio
chown root:root /var/spool/cron/
chown root:root /etc/cron.d/
chown root:root /etc/crontab
chown root:root /etc/cron.daily/
chown root:root /etc/cron.hourly/
chown root:root /etc/cron.monthly/
chown root:root /etc/cron.weekly/
chown root:root /var/spool/at/
chown root:root /etc/sysctl.conf
chown root:root /etc/xinetd.conf
chown root:root /etc/xinetd.d
chown root:root /etc/services
chown root:root /bin/traceroute
chown root:root /etc/syslog.conf
chown root:root /etc/security/access.conf
chown root:root /etc/securetty
#
#
```



APPENDIX B — REMOVAL OF UNNECESSARY USERS

The second version of h-script introduced the removal of extraneous users.

```
# This section removes unnecessary users.
# Thanks to Bill Bowers/ESI for much of this module.
#
userdel lp
userdel sync
userdel shutdown
userdel halt
userdel news
userdel gopher
userdel operator
userdel games
userdel mail
userdel uucp
userdel ftp
userdel netdump
# Remove a few more users - scz, 02Dec2009
userdel adm
userdel pcap
userdel avahi-autoipd
userdel sabayon
#
```

APPENDIX C — REPLACING SYSTEM CONFIGURATION FILES

The third version of h-script began to replace system configuration files.

```
#
echo "Replacing /boot/grub/menu.lst..."
/bin/cp /boot/grub/menu.lst /boot/grub/menu.lst_`date +%m-%d-20%y-%H%M`
cp ./boot-grub-menu.lst /boot/grub/menu.lst
#
echo "Replacing /etc/audit/auditd.conf..."
/bin/cp /etc/audit/auditd.conf /etc/audit/auditd.conf_`date +%m-%d-20%y-%H%M`
cp ./etc-audit-auditd.conf /etc/audit/auditd.conf
#
echo "Replacing /etc/audit/audit.rules..."
/bin/cp /etc/audit/audit.rules /etc/audit/audit.rules_`date +%m-%d-20%y-%H%M`
cp ./etc-audit-audit.rules /etc/audit/audit.rules
#
echo "Replacing /etc/bashrc..."
/bin/cp /etc/bashrc /etc/bashrc_`date +%m-%d-20%y-%H%M`
cp ./etc-bashrc /etc/bashrc
#
echo "Replacing /etc/grub.conf..."
/bin/cp /etc/grub.conf /etc/grub.conf_`date +%m-%d-20%y-%H%M`
cp ./etc-grub.conf /etc/grub.conf
#
echo "Replacing /etc/hosts.allow..."
touch /etc/hosts.allow
/bin/cp /etc/hosts.allow /etc/hosts.allow_`date +%m-%d-20%y-%H%M`
cp ./etc-hosts.allow /etc/hosts.allow
#
echo "Replacing /etc/hosts.deny..."
touch /etc/hosts.deny
/bin/cp /etc/hosts.deny /etc/hosts.deny_`date +%m-%d-20%y-%H%M`
cp ./etc-hosts.deny /etc/hosts.deny
#
echo "Replacing /etc/inittab..."
/bin/cp /etc/inittab /etc/inittab_`date +%m-%d-20%y-%H%M`
cp ./etc-inittab /etc/inittab
#
echo "Replacing /etc/login.defs..."
/bin/cp /etc/login.defs /etc/login.defs_`date +%m-%d-20%y-%H%M`
cp ./etc-login.defs /etc/login.defs
#
echo "Replacing /etc/motd/..."
/bin/cp /etc/motd /etc/motd_`date +%m-%d-20%y-%H%M`
cp ./etc-motd /etc/motd
#
echo "Replacing /etc/pam.d/passwd..."
/bin/cp /etc/pam.d/passwd /etc/pam.d/passwd_`date +%m-%d-20%y-%H%M`
```

```
cp ./etc-pam.d-passwd /etc/pam.d/passwd
#
echo "Replacing /etc/pam.d/system-auth..."
/bin/cp /etc/pam.d/system-auth /etc/pam.d/system-auth_`date +%m-%d-20%y-%H%M`
cp ./etc-pam.d-system-auth /etc/pam.d/system-auth
#
echo "Replacing /etc/securetty..."
/bin/cp /etc/securetty /etc/securetty_`date +%m-%d-20%y-%H%M`
cp ./etc-securetty /etc/securetty
#
echo "Replacing /etc/security/opasswd"
/bin/cp /etc/security/opasswd /etc/security/opasswd_`date +%m-%d-20%y-%H%M`
cp ./etc-security-opasswd /etc/security/opasswd
#
echo "Replacing /etc/shadow..."
/bin/cp /etc/shadow /etc/shadow_`date +%m-%d-20%y-%H%M`
echo "WARNING - THIS WILL RESET THE ROOT PASSWD "
cp ./etc-shadow /etc/shadow
#
echo "Replacing /etc/shells..."
/bin/cp /etc/shells /etc/shells_`date +%m-%d-20%y-%H%M`
cp ./etc-shells /etc/shells
#
echo "Replacing /etc/ssh/sshd_config..."
/bin/cp /etc/ssh/sshd_config /etc/ssh/sshd_config_`date +%m-%d-20%y-%H%M`
cp ./etc-ssh-sshd_config /etc/ssh/sshd_config
#
echo "Replacing /etc/sysctl.conf..."
/bin/cp /etc/sysctl.conf /etc/sysctl.conf_`date +%m-%d-20%y-%H%M`
cp ./etc-sysctl.conf /etc/sysctl.conf
#
echo "Replacing /etc/syslog.conf..."
/bin/cp /etc/syslog.conf /etc/syslog.conf_`date +%m-%d-20%y-%H%M`
cp ./etc-syslog.conf /etc/syslog.conf
#
echo "Replacing /etc/xinetd.conf..."
/bin/cp /etc/xinetd.conf /etc/xinetd.conf_`date +%m-%d-20%y-%H%M`
cp ./etc-xinetd.conf /etc/xinetd.conf
#
```



APPENDIX D — ADDITIONAL FUNCTIONALITY IN POST-H-SCRIPT

Here is the post-h-script script:

```
#!/bin/bash
#
# Welcome to post-hardening-script! post-h-script is a bash script designed
# to complete the hardening of Red Hat Enterprise Linux 5.3/5.4 system to
# meet the DISA Security Readiness Review (SRR) requirements.
#
echo " "
echo "post-h-script v1.1"
echo " "
echo "Running new job at `date +%m-%d-20%y-%H%M`"
echo " "
echo " "
#
echo " "
echo "Removing VNC"
yum -y remove vnc vnc-server
echo " "
echo "Removing Samba"
chkconfig smb off
yum -y remove smb
echo " "
echo "Removing TFTP Server"
yum -y remove tftp-server
echo " "
echo "Removing Telnet"
yum -y remove telnet telnet-server krb5-workstation
echo " "
echo "Removing MINICOM"
yum -y remove minicom
echo " "
echo "Removing RSH"
yum -y remove rsh rsh-server
echo " "
echo "Removing NIS"
chkconfig ypbind off
yum -y remove ypserv
echo " "
echo "Removing DHCP Server"
chkconfig dhcpd off
yum -y remove dhcp
echo " "
echo "Disabling FTP Server"
chkconfig vsftpd off
echo " "
echo "Install Console Screen Lock"
```

```
yum -y install vlock
echo " "
echo "Removing IP6Tables"
chkconfig ip6tables off
yum -y remove iptables
#
#
#
echo " "
echo "Disabling Unnecessary Services"ch
chkconfig anacron off
chkconfig apmd off
chkconfig autofs off
chkconfig avahi-daemon offy
chkconfig bluetooth off
chkconfig cups off
chkconfig firstboot off
chkconfig gpm off
chkconfig haldaemon off
chkconfig hidd off
chkconfig hplip off
chkconfig isdn off
chkconfig kdump off
chkconfig kudzu off
chkconfig mcstrans off
chkconfig mdmonitor off
chkconfig messagebus off
chkconfig microcode_ctl off
chkconfig pcscd off
chkconfig readahead_early off
chkconfig readahead_later off
chkconfig setroubleshoot off
chkconfig xfs off
#
MODPROBE=/etc/modprobe.conf
if [ -f ${MODPROBE} ]; then
    if [ $(grep -c "install[[:space:]]cransfs[[:space:]]/bin/true" ${MODPROBE}) -lt 1 ]; then
        echo "install cramfs /bin/true" >> ${MODPROBE}
    fi
    if [ $(grep -c "install[[:space:]]freevxfs[[:space:]]/bin/true" ${MODPROBE}) -lt 1 ]; then
        echo "install freevxfs /bin/true" >> ${MODPROBE}
    fi
    if [ $(grep -c "install[[:space:]]jffs2[[:space:]]/bin/true" ${MODPROBE}) -lt 1 ]; then
        echo "install jffs2 /bin/true" >> ${MODPROBE}
    fi
    if [ $(grep -c "install[[:space:]]hfs[[:space:]]/bin/true" ${MODPROBE}) -lt 1 ]; then
        echo "install hfs /bin/true" >> ${MODPROBE}
    fi
fi
```

```
if [ $(grep -c "install[:space:]hfsplus[:space:]/bin/true" ${MODPROBE}) -lt 1 ]; then
    echo "install hfsplus /bin/true" >> ${MODPROBE}
fi
if [ $(grep -c "install[:space:]squashfs[:space:]/bin/true" ${MODPROBE}) -lt 1 ]; then
    echo "install squashfs /bin/true" >> ${MODPROBE}
fi
if [ $(grep -c "install[:space:]udf[:space:]/bin/true" ${MODPROBE}) -lt 1 ]; then
    echo "install udf /bin/true" >> ${MODPROBE}
fi
fi

NETWORK_SYS=/etc/sysconfig/network
if [ -f ${MODPROBE} ]; then
    if [ $(grep -c "NETWORKING_IPV6=no" ${NETWORK_SYS}) -lt 1 ]; then
        echo "NETWORKING_IPV6=no" >> ${NETWORK_SYS}
    fi
    if [ $(grep -c "IPV6INIT=no" ${NETWORK_SYS}) -lt 1 ]; then
        echo "IPV6INIT=no" >> ${NETWORK_SYS}
    fi
    if [ $(grep -c "IPV6_AUTOCONF=no" ${NETWORK_SYS}) -lt 1 ]; then
        echo "IPV6_AUTOCONF=no" >> ${NETWORK_SYS}
    fi
fi

LIMITS=/etc/security/limits.conf
if [ -f ${LIMITS} ]; then
    if [ $(grep -c "[[:space:]]hard[:space:]core[:space:] 0" ${LIMITS}) -lt 1 ]; then
        echo "** hard core 0" >> ${LIMITS}
    fi
fi
fi
#
echo " "
echo " "
echo "Reboot System after script completes"
```

APPENDIX E — THE FULL VERSION OF H-SCRIPT V3.0 (COMPRESSED ARCHIVE)

Embedded in this Appendix is the full content of h-script version 3.0, which was highlighted in Appendix C. It does not include the post-h-script processing but it should suffice to demonstrate the concepts included in this paper. Including the contents of all text and configuration files in this document would be impractical.



Double-click on the green box to open the archive file using WinZip or a similar application. Extract the contents to a folder called 'h-script' and view the README file.



ACKNOWLEDGMENTS

The author would like to thank the following for their significant and substantive contributions to the Automated Hardening effort:

Kristopher Maxson– for adapting and expanding the original version of h-script, for creating post-h-script, and for tireless attention to resolving SRR findings and other issues

Johnathon Trumble) – for spearheading the Red Hat Satellite initiative and for streamlining the remote Operating System deployment and post-installation processing

The National Security Agency (<http://www.nsa.gov>) and the Defense Information Systems Agency (<http://www.disa.mil>) – during the development of tools and configurations, the CSC team made extensive use of the technical security materials produced and distributed by these organizations



CSC

266 Second Avenue
Waltham, Massachusetts 02451
United States
+1.800.272.0018

Worldwide CSC Headquarters

The Americas

3170 Fairview Park Drive
Falls Church, Virginia 22042
United States
+1.703.876.1000

Europe, Middle East, Africa

Royal Pavilion
Wellesley Road
Aldershot, Hampshire GU11 1PZ
United Kingdom
+44(0)1252.534000

Australia

26 Talavera Road
Macquarie Park, NSW 2113
Australia
+61(0)29034.3000

Asia

139 Cecil Street
#06-00 Cecil House
Singapore 069539
Republic of Singapore
+65.6221.9095

About CSC

The mission of CSC is to be a global leader in providing technology enabled business solutions and services.

With the broadest range of capabilities, CSC offers clients the solutions they need to manage complexity, focus on core businesses, collaborate with partners and clients, and improve operations.

CSC makes a special point of understanding its clients and provides experts with real-world experience to work with them. CSC is vendor-independent, delivering solutions that best meet each client's unique requirements.



For 50 years, clients in industries and governments worldwide have trusted CSC with their business process and information systems outsourcing, systems integration and consulting needs.

The company trades on the New York Stock Exchange under the symbol "CSC."

DISCLAIMER

The information, views and opinions expressed in this paper constitute solely the authors' views and opinions and do not represent in any way CSC's official corporate views and opinions. The authors have made every attempt to ensure that the information contained in this paper has been obtained from reliable sources. CSC is not responsible for any errors or omissions or for the results obtained from the use of this information. All information in this paper is provided "as is," with no guarantee by CSC of completeness, accuracy, timeliness or the results obtained from the use of this information, and without warranty of any kind, express or implied, including but not limited to warranties of performance, merchantability and fitness for a particular purpose. In no event will CSC, its related partnerships or corporations, or the partners, agents or employees thereof be liable to you or anyone else for any decision made or action taken in reliance on the information in this paper or for any consequential, special or similar damages, even if advised of the possibility of such damages.

Copyright © 2010 Computer Sciences Corporation. All rights