Cybersecurity

Project 1 Hardening Summary and Checklist

OS Information

Customer	Baker Street Corporation
Hostname	**Baker_Street_Linux_Server**
OS Version	**Ubuntu 22.04.5 LTS (Jammy Jellyfish)**
Memory information	**15G**
Uptime information	**22:48:42 up 14 min**

Checklist

Completed	Activity	Script(s) used / Tasks completed / Screenshots
☑	OS backup	`root@Baker_Street_Linux_Server:/# sudo tar -cvpzf /baker_street_backup.tar.gz --exclude=/baker_street_backup.tar.gz --exclude=/proc --exclude=/tmp --exclude=/mnt --exclude=/sys --exclude=/dev --exclude=/run /` Executing this script back ups the Operating system into a .tar gz archive of the entire filesystem, except for the directories explicitly excluded.

☑	Auditing users and groups	`root@Baker_Street_Linux_Server:/# cat /etc/passwd`
		Use the above syntax to get a list of all users on the system, cross reference that list with the list provided in the guide.
		`root@Baker_Street_Linux_Server:/# sudo userdel -r lestrade`
		`root@Baker_Street_Linux_Server:/# sudo userdel -r irene`

```
root@Baker_Street_Linux_Server:/# sudo userdel -r mary
```

```
root@Baker_Street_Linux_Server:/# sudo userdel -r gregson
```

Used the above syntax to remove the terminated users, used userdel is the command to delete a user, -r deletes the home directory and files

```
File  Edit  View  Search  Terminal  Help
root@Baker_Street_Linux_Server:/# id mrs_hudson
uid=1006(mrs_hudson) gid=1006(mrs_hudson) groups=1006(mrs_hudson),1013(finance)
root@Baker_Street_Linux_Server:/# sudo passwd -l mrs_hudson
sudo: unable to resolve host Baker_Street_Linux_Server: Temporary failure in name resolution
passwd: password expiry information changed.
root@Baker_Street_Linux_Server:/# sudo passwrd -S mrs_hudson
sudo: unable to resolve host Baker_Street_Linux_Server: Temporary failure in name resolution
sudo: passwrd: command not found
root@Baker_Street_Linux_Server:/# sudo passwd -S mrs_hudson
sudo: unable to resolve host Baker_Street_Linux_Server: Temporary failure in name resolution
mrs_hudson L 12/12/2024 0 99999 7 -1
root@Baker_Street_Linux_Server:/#
```

```
uid=1010(toby) gid=1010(toby) groups=1010(toby)
root@Baker_Street_Linux_Server:/# id mary
uid=1007(mary) gid=1007(mary) groups=1007(mary),1013(finance)
root@Baker_Street_Linux_Server:/# id mrs_hudson
uid=1006(mrs_hudson) gid=1006(mrs_hudson) groups=1006(mrs_hudson),1013(finance)
root@Baker_Street_Linux_Server:/# passwd -l gregson
passwd: password expiry information changed.
root@Baker_Street_Linux_Server:/# id moriarty
uid=1002(moriarty) gid=1002(moriarty) groups=1002(moriarty),1012(engineering)
root@Baker_Street_Linux_Server:/# sudo passwd -l moriaty
^[[A^[[B^C;1
sudo: unable to resolve host Baker_Street_Linux_Server: Temporary failure in name resolution

root@Baker_Street_Linux_Server:/# ;1
bash: syntax error near unexpected token `;'
root@Baker_Street_Linux_Server:/# sudo passwd -l moriarty
sudo: unable to resolve host Baker_Street_Linux_Server: Temporary failure in name resolution
passwd: password expiry information changed.
root@Baker_Street_Linux_Server:/# id moriarty
uid=1002(moriarty) gid=1002(moriarty) groups=1002(moriarty),1012(engineering)
root@Baker_Street_Linux_Server:/# sudo grep 'moriarty' /etc/shadow
sudo: unable to resolve host Baker_Street_Linux_Server: Temporary failure in name resolution
moriarty:!$y$j9T$GzTsxAplRiUlh581K./ES/$3ZIj39Zd8PBaacD3ZtUaPx1hLb82y4vHpGvYt0mV2m0:20074:0:99999:7:::
root@Baker_Street_Linux_Server:/# sudo passwd -S moriarty
sudo: unable to resolve host Baker_Street_Linux_Server: Temporary failure in name resolution
moriarty L 12/17/2024 0 99999 7 -1
root@Baker_Street_Linux_Server:/#
```

Locking users that are on temporary leave: two users fit this description and were locked out using the following syntax. First we identify the user using the following syntax

"id username"

Then we lock out their password preventing any password based logins with the syntax

" passwd -l username"

Verify that the user is indeed locked out and the status of their password with the syntax

"passwd -S username"

The L in the highlighted portion of the screenshot indicates the password is locked

```
root@Baker_Street_Linux_Server:/# sudo passwd -S toby
sudo: unable to resolve host Baker_Street_Linux_Server: Temporary failure in name resolution
toby L 12/12/2024 0 99999 7 -1
root@Baker_Street_Linux_Server:/# sudo passwd -u toby
sudo: unable to resolve host Baker_Street_Linux_Server: Temporary failure in name resolution
passwd: unlocking the password would result in a passwordless account.
You should set a password with usermod -p to unlock the password of this account.
root@Baker_Street_Linux_Server:/# sudo nano /etc/hosts
sudo: unable to resolve host Baker_Street_Linux_Server: Temporary failure in name resolution
root@Baker_Street_Linux_Server:/# hostname
Baker_Street_Linux_Server
root@Baker_Street_Linux_Server:/# ping $(Baker_Street_Linux_Server)
bash: Baker_Street_Linux_Server: command not found
bash: ping: command not found
root@Baker_Street_Linux_Server:/# sudo passwd toby
New password:
Retype new password:
passwd: password updated successfully
root@Baker_Street_Linux_Server:/# sudo passwd -u toby
passwd: password expiry information changed.
root@Baker_Street_Linux_Server:/# sudo passwd -S toby
toby P 12/18/2024 0 99999 7 -1
root@Baker_Street_Linux_Server:/#
```

Used the syntax “**sudo passwd -S username**” with all the employed users to identify the users that were locked out.

User “toby” was the only user locked out.

Used syntax “**sudo passwd toby**” to prompt a new password then after entering a new password use “sudo passwd -u toby” to unlock toby.

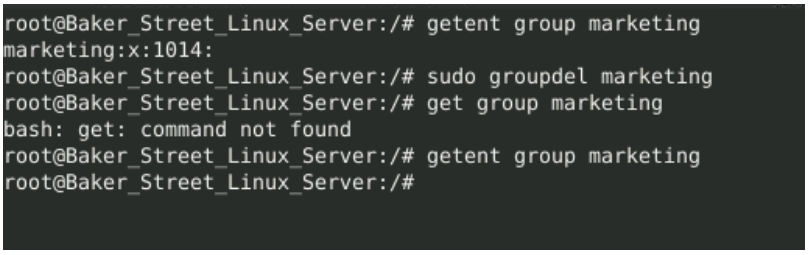
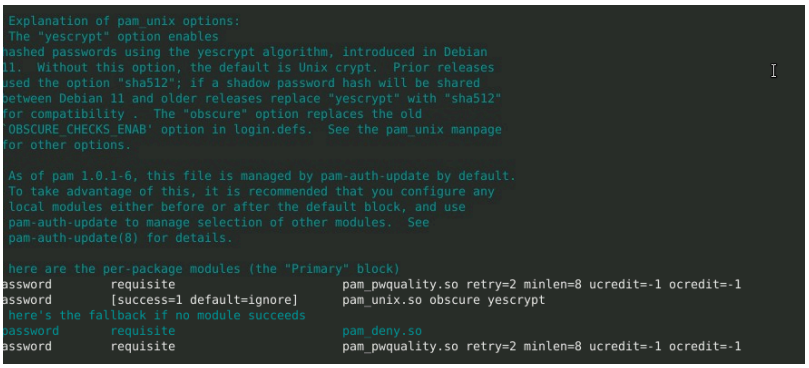
Use syntax “**sudo passwd -S toby**” and get the output
Toby P 12/18/2024 0 99999 7 -1

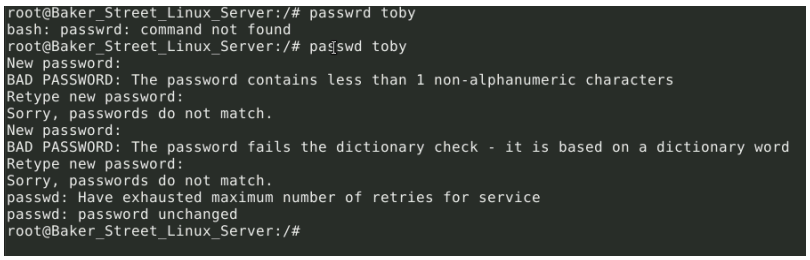
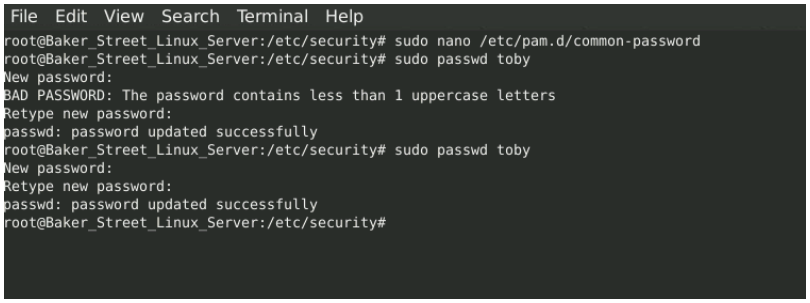
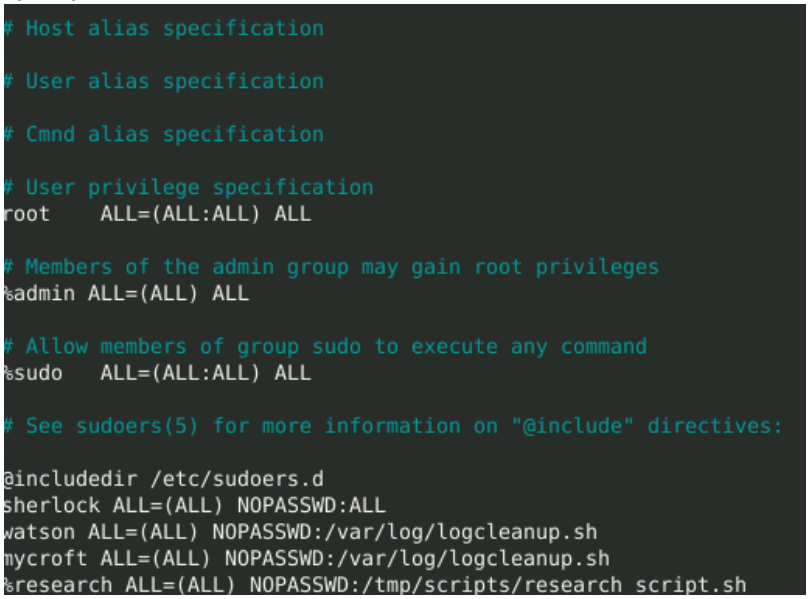
The “P” in that output indicates that the user is Unlocked

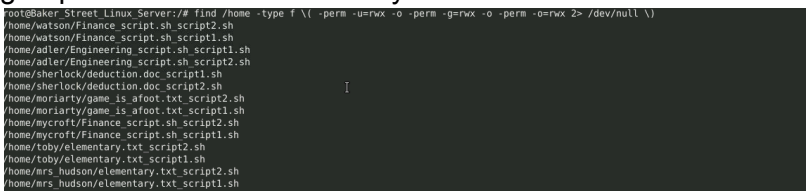
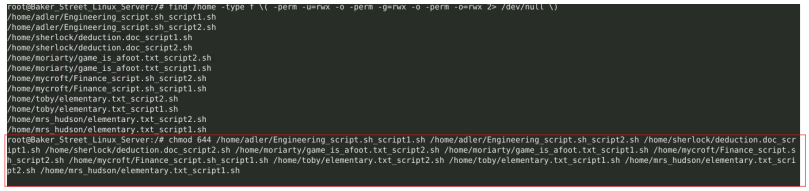
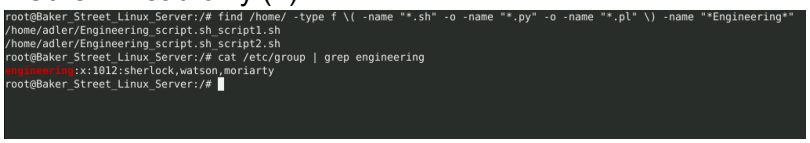
```
root@Baker_Street_Linux_Server:/# sudo groupadd research
root@Baker_Street_Linux_Server:/# getent group research
research:x:1014:
root@Baker_Street_Linux_Server:/# for user in toby adler mycroft; do sudo usermod -aG research $user; done
root@Baker_Street_Linux_Server:/# getent group research
research:x:1014:toby,adler,mycroft
root@Baker_Street_Linux_Server:/#
```

First create the research group using the syntax
“**Sudo groupadd research**”
Next syntax “getent group research” to insure the group exists
To move the three users that belong in the research use the syntax “**for user in toby adler mycroft; do sudo usermod -aG research $user; done**”
In short this syntax allowed me to add all three users in to the research group instead of entering them one by one, by using a loop in bash.

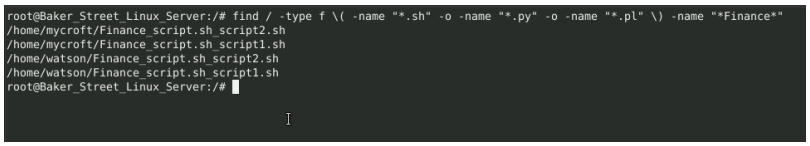
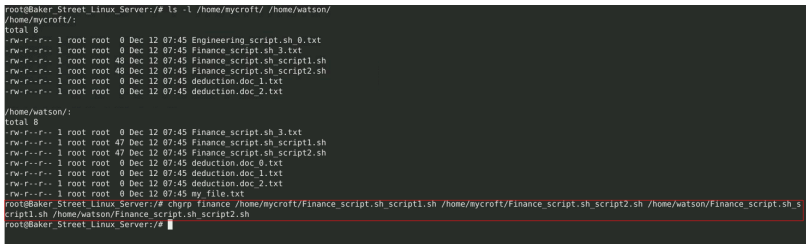
Next verify that the loop worked with the syntax **“gent group research”** this displayed all three users in the research group.

		 First identify the Marketing group with syntax **“getent group marketing”** this shows that the marketing group does in fact exist. Next use syntax “**sudo groupdel marketing**” to delete the marketing group. Finally verify that the group has been deleted by using syntax **“getent group marketing”** since the group was deleted there is no output.
☑	Updating and enforcing password policies	Us syntax **“nano /etc/pam.d/common-password”**  Under the line that states *# here are the per-package modules (the Primary” block)* First column: **password** Second column: **requisite** Third Column: **pam_pwquality.so retry=2 minlen=8 ucredit=-1 ocredit=-1** Exit and save

		  To verify that we have changed the password policies correctly pick a user and attempt to change their password but leave out one of the polices, then attempt to change the password utilizing all the polices **Example: pa$$w0rd vs Pa$$w0rd**
☑	Updating and enforcing sudo permissions	Typ syntax **“visudo”**  First ALL refers to the user, Second ALL refers to the host NOPASSWD tells sudo allows the user to runs specific commands without a password Third ALL refers to the commands the user can execute with sudo Sherlock gets full privileges and should be set to ALL=ALL

		NOPASSWD:ALL Watson and mycroft get ALL=(ALL) NOPASS: and script that set the limits Research (represented with a % before hte name indicating it is a group) gets ALL =(ALL) and then the scrip that sets the limits
☑	Validating and updating permissions on files and directories	`/# find /home -type f \( -perm -u=rwx -o -perm -g=rwx -o -perm -o=rwx 2> /dev/null \)` Get a list of all files with world permissions(read, write, or execute) using the the syntax above : **Find /home-type f \(-perm -u=rwx -o -perm -g+rwx -o -perm -o=rwx 2> /dev/null \)** Broken down this syntax allows you to find files in the the home directory utilizing **\(\)** groups the conditions and create logical groupings **-perm** to check specific permissions **rwx** matches files with Read, Write and Execute permissions **-o** match any file that meets this conditions **-g** refers to files group **o** refers to other or everyone else  Above shows the list of files with world permissions that need to be modified.  Using one syntax i modified all files with world permissions **“chmod 644 FILENAME”** Chmod allows you to modify a file 644 changes the permissions as follows 6 owner: Read (4) write (2) 4 Group: Read only (4) 4 Other : Read only (4)  To find all the scripts that contain the word engineering and who has access I used the syntax **find/home/ type f \(-name “*.sh” -o -name “*.py” -o -name “*.pl” \) -name “*Engineering”**

		Broken down this syntax using the command **find** isolate it to only the **/home** directory insures that the files are scripts (ending in a **.sh .py .pl**) and have the word **“engineering”** somewhere in the name by placing ***** on either side of the word .
		
		Next we check who is in the engineering group by using **“cat /etc/group | grep engineering”**
		
		Using **cat** and the path to the group directory then piping it with a **grep** command for the engineering group shows the members of the engineering group.
		
		Adler has access to the script but noone else in the engineering group does
		
		`root@Baker_Street_Linux_Server:/# chgrp engineering /home/adler/Engineering_script.sh_script1.sh /home/adler/Engineering_script.sh_script2.sh`
		`root@Baker_Street_Linux_Server:/# cd /home/adler`
		`root@Baker_Street_Linux_Server:/home/adler# ls -l`
		`total 8`
		`-rw-r--r-- 1 root root 0 Dec 12 07:45 Engineering_script.sh_0.txt`
		`-rw-r--r-- 1 root root 0 Dec 12 07:45 Engineering_script.sh_3.txt`
		`-rw-r--r-- 1 root engineering 46 Dec 12 07:45 Engineering_script.sh_script1.sh`
		`-rw-r--r-- 1 root engineering 46 Dec 12 07:45 Engineering_script.sh_script2.sh`
		`-rw-r--r-- 1 root root 0 Dec 12 07:45 deduction.doc_2.txt`
		`-rw-r--r-- 1 root root 0 Dec 12 07:45 game_is_afoot.txt_1.txt`
		`root@Baker_Street_Linux_Server:/home/adler#`
		
		Used the syntax **chgrp engineering SCRIPTNAMES**
		This moved the two scripts from just adler to the engineering group
		To verify us syntax **cd /home/adler** and **ls -l**
		To display the two scrips and see that they are now accessible by the engineering group
		`root@Baker_Street_Linux_Server:/# find / -type f \( -name "*.sh" -o -name "*.py" -o -name "*.pl" \) -name "*research*"`
		`/tmp/scripts/research_script.sh`
		`root@Baker_Street_Linux_Server:/#`
		
		Similar to the first syntax used to find the engineering scripts use the following to find the research scripts
		find / -type f \(-name "*.sh" -o -name "*.py" -o -name "*.pl" \) -name "*research*"
		Note that the change to this syntax is that it is looking from the root directory by using just **find /**
		
		`root@Baker_Street_Linux_Server:/tmp/scripts# chgrp research research_script.sh`
		`root@Baker_Street_Linux_Server:/tmp/scripts# ls -l`
		`total 0`
		`-rwxr-xr-x 1 root research 0 Dec 18 17:42 research_script.sh`
		`root@Baker_Street_Linux_Server:/tmp/scripts#`
		
		Navigate to the scripts file **cd /tmp/scripts**
		
		use syntax **chgrp research research_script.sh** to make it so that only the research group has access
		To verify this use command **ls -l** in the script directory this will display that **research_script.sh** is in the research group.

		 Using the same syntax for research but substituting “*Research*" with "*Finance*" **find / -type f \(-name "*.sh" -o -name "*.py" -o -name "*.pl" \) -name "*Finance*"** Will list all the finance scripts  Change the group for all files to the fiance group by using the syntax **“chgrp finance FULLPATH TO FILE”** Separate each filepath with a space to change multiple files in one syntax
☑	Optional: Updating password hashing configuration	After searching through every employee's file, no files with hidden passwords were discovered.
☑	Auditing and securing SSH	 Access the primary configuration file for the OpenSSH Server Using the following syntax: **“nano /etc/ssh/sshd_config”**

```
#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

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

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
KbdInteractiveAuthentication no

# Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
```

Disable the ability to SSH with an empty password with **“PermitEmptyPassword no”**

```
# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be disregarded by default in future.
#AuthorizedKeysFile     .ssh/authorized_keys .ssh/authorized_keys2

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody
```

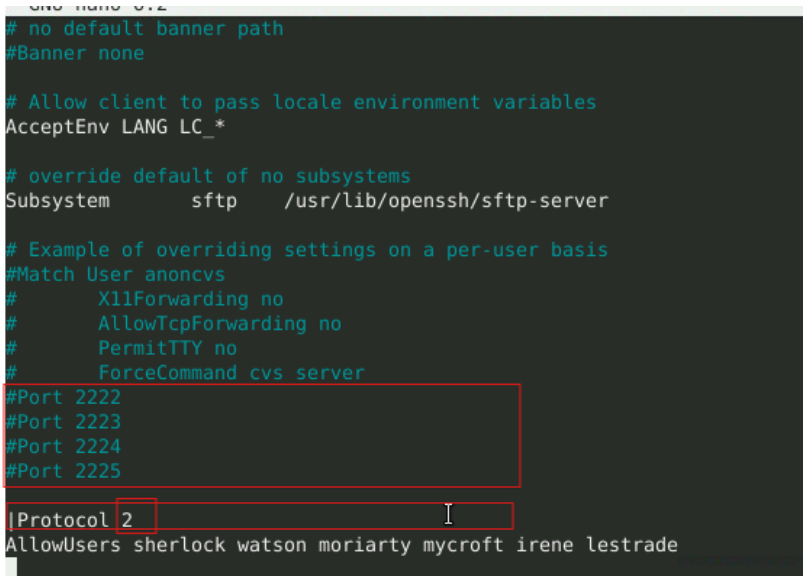
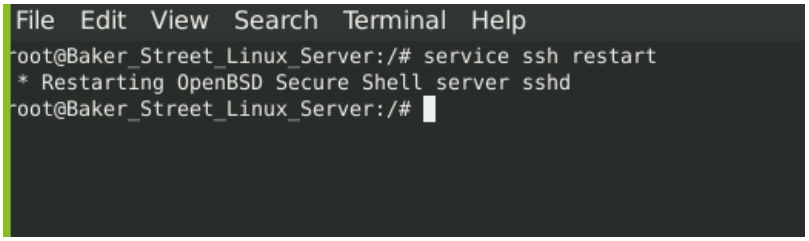
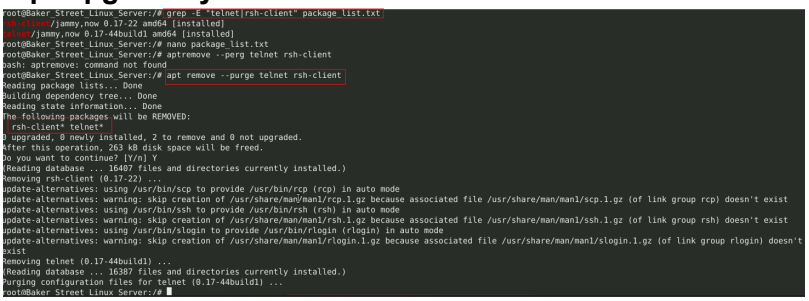
Disable the ability to SSH with the root user with **“PermitRootLogin no”**

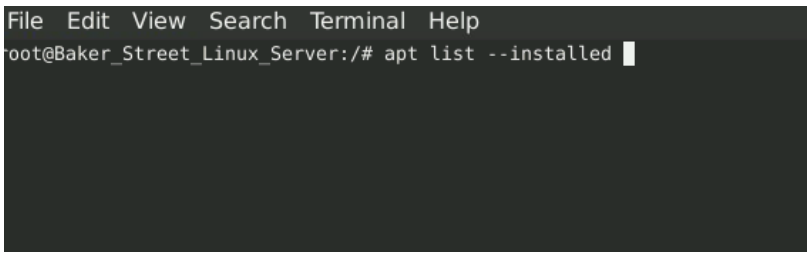
```
# no default banner path
#Banner none

# Allow client to pass locale environment variables
AcceptEnv LANG LC_*

# override default of no subsystems
Subsystem       sftp    /usr/lib/openssh/sftp-server

# Example of overriding settings on a per-user basis
#Match User anoncvs
#       X11Forwarding no
#       AllowTcpForwarding no
#       PermitTTY no
#       ForceCommand cvs server
#Port 2222
#Port 2223
#Port 2224
#Port 2225

Protocol 2
AllowUsers sherlock watson moriarty mycroft irene lestrade
```

Hash ports 2222, 2223, 2224, 2225 Disabling the ability to SSH from those ports

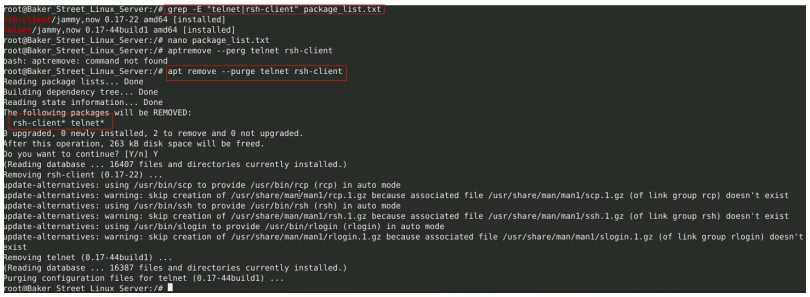
```
# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.

Include /etc/ssh/sshd_config.d/*.conf

Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
```

Unhash port 22 allowing SSH to port 22

		 Enable Protocol 2  Restarting SSH with syntax **Service ssh restart**
☑	Reviewing and updating system packages	Run the following two syntax: **“apt update”** **“apt upgrade -y”**  Create the Package List using the syntax **“nano package_list.txt”**



Run syntax **“apt list –installed”**



Identify if telnet or rsh-client with syntax
grep -E “telnet | rsh-client” package_list.txt
Remove both from package_list.txt with the following syntax
Apt remove –perge telnet rsh-client

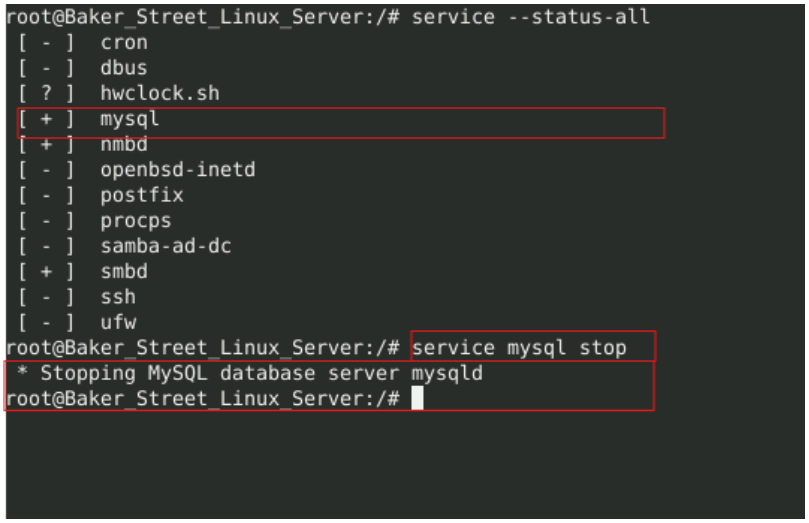
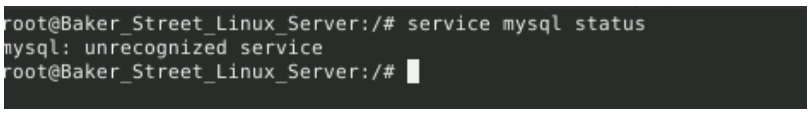
Why does **telnet** and **rsh-client** have potential security risks?

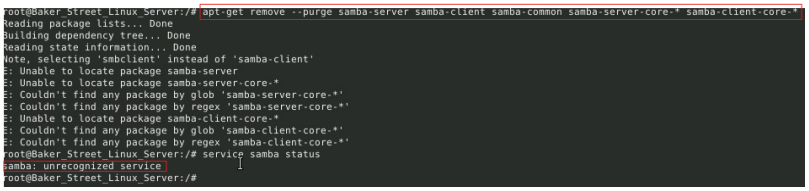
Telnet:

- Unencrypted Communication: Telnet sends data, including sensitive information such as passwords, in plaintext. This means that anyone intercepting the communication could easily capture and read it.
- Lack of Authentication: Unlike SSH, Telnet does not provide strong authentication mechanisms, making it easier for attackers to impersonate legitimate users.
- Deprecated Protocol: Due to these risks, Telnet is considered outdated and is generally replaced by more secure protocols like SSH (Secure Shell).

rsh-client:

- **Unencrypted Communication:** Like Telnet, the Remote Shell (RSH) protocol sends data, including passwords, in plaintext over the network, making it vulnerable to eavesdropping.
- **No Strong Authentication:** RSH does not have robust authentication mechanisms, and it relies on

		trust relationships between systems, which can be exploited. • **Security Vulnerabilities:** Historically, RSH has been found to have various vulnerabilities that could be exploited by attackers to gain unauthorized access.
☑	Disabling unnecessary services	 List all services wit the following syntax **Service –status-all** Identify that mysql is running Stop it from running with syntax **Service mysql stop**  Further remove, disable and move mysql completely with syntax **Apt-get remove –purge mysql-client mysql-common mysql-server-core-* mysql-client-core-***  Verify that mysql is completely removed with syntax **Service mysql status** Prompt is: unrecognizable service verifying that mysql is gone

		 remove, disable and move samba completely with syntax **Apt-get remove –purge samba-server samba-client samba-common samba-server-core* samba-client-core-*** Verify that samba is completely removed with syntax **Service samba status** Prompt is: unrecognizable service verifying that samba is gone  Double check the status of each service with syntax **service –status-all**
☑	Enabling and configuring logging	 Access journald.conf with syntax **Nano /etc/systemd/journald.conf**

```
#  This file is part of systemd.
#
#  systemd is free software; you can redistribute it and/or modify it under the
#  terms of the GNU Lesser General Public License as published by the Free
#  Software Foundation; either version 2.1 of the License, or (at your option)
#  any later version.
#
# Entries in this file show the compile time defaults. Local configuration
# should be created by either modifying this file, or by creating "drop-ins" in
# the journald.conf.d/ subdirectory. The latter is generally recommended.
# Defaults can be restored by simply deleting this file and all drop-ins.
#
# Use 'systemd-analyze cat-config systemd/journald.conf' to display the full config.
#
# See journald.conf(5) for details.

[Journal]
Storage=persistent
#Compress=yes
#Seal=yes
#SplitMode=uid
#SyncIntervalSec=5m
#RateLimitIntervalSec=30s
#RateLimitBurst=10000
SystemMaxUse=300M
#SystemKeepFree=
#SystemMaxFileSize=
#SystemMaxFiles=100
#RuntimeMaxUse=
```

Set “**storage=persistent**”
Set “**systemMaxUse=300M**”

```
root@Baker_Street_Linux_Server:/# nano /etc/logrotate.conf
```

Navigate to the logrotate.conf file with syntax
Nano /etc/logrotate.conf

```
# see "man logrotate" for details

# global options do not affect preceding include directives

# rotate log files daily
daily

# use the adm group by default, since this is the owning group
# of /var/log/syslog.
su root adm

# keep 1 weeks worth of backlogs
rotate 1

# create new (empty) log files after rotating old ones
create

# use date as a suffix of the rotated file
#dateext

# uncomment this if you want your log files compressed
#compress

# packages drop log rotation information into this directory
include /etc/logrotate.d

# system-specific logs may also be configured here.
```

Changed the log rotation from weekly to daily.
Rotate out the logs after 7 days. By changing it to 1 rotation

☑	Scripts created	 Script 1 Hardening_script1.sh This scripts backs up the operating system, audits users and groups and removes all world permissions starting at the home directory ensure that members of the of groups can only view edit and execute the scripts intended for their respected groups

```
root@Baker_Street_Linux_Server:/etc/cron.monthly# ./hardening_script1.sh
Gathering hostname...
Gathering OS version...
Gathering memory information...
Gathering uptime information...
Backing up the OS...
```

```
/home/sherlock/my_file.txt
/home/sherlock/deduction.doc_3.txt
/home/sherlock/game_is_afoot.txt_2.txt
/home/sherlock/elementary.txt_0.txt
/home/sherlock/game_is_afoot.txt_1.txt
/home/moriarty/
/home/moriarty/.bashrc
/home/moriarty/.bash_logout
/home/moriarty/.profile
/home/moriarty/game_is_afoot.txt_script2.sh
/home/moriarty/game_is_afoot.txt_script1.sh
/home/moriarty/Finance_script.sh_2.txt
/home/moriarty/my_file.txt
/home/moriarty/elementary.txt_1.txt
/home/moriarty/game_is_afoot.txt_3.txt
/home/moriarty/Finance_script.sh_0.txt
/home/mycroft/
/home/mycroft/.bashrc
/home/mycroft/.bash_logout
/home/mycroft/.profile
/home/mycroft/Finance_script.sh_script2.sh
/home/mycroft/Finance_script.sh_script1.sh
/home/mycroft/Finance_script.sh_3.txt
/home/mycroft/Engineering_script.sh_0.txt
/home/mycroft/deduction.doc_1.txt
/home/mycroft/deduction.doc_2.txt
/home/toby/
/home/toby/.bashrc
/home/toby/.bash_logout
/home/toby/.profile
/home/toby/.secret.txt
/home/toby/elementary.txt_script2.sh
/home/toby/elementary.txt_script1.sh
/home/toby/elementary.txt_3.txt
/home/toby/elementary.txt_0.txt
/home/toby/Engineering_script.sh_2.txt
/home/toby/deduction.doc_1.txt
/home/watson/
/home/watson/.bashrc
/home/watson/.bash_logout
/home/watson/.profile
/home/watson/.bash_history
/home/watson/Finance_script.sh_script2.sh
/home/watson/Finance_script.sh_script1.sh
/home/watson/Finance_script.sh_3.txt
/home/watson/my_file.txt
/home/watson/deduction.doc_1.txt
/home/watson/deduction.doc_2.txt
/home/watson/deduction.doc_0.txt
/home/mrs_hudson/
/home/mrs_hudson/.bashrc
/home/mrs_hudson/.bash_logout
/home/mrs_hudson/.profile
/home/mrs_hudson/elementary.txt_script2.sh
/home/mrs_hudson/elementary.txt_script1.sh
/home/mrs_hudson/elementary.txt_3.txt
/home/mrs_hudson/Engineering_script.sh_1.txt
/home/mrs_hudson/deduction.doc_2.txt
/home/mrs_hudson/deduction.doc_0.txt
/home/sysadmin/
/home/sysadmin/.bashrc
/home/sysadmin/.bash_logout
/home/sysadmin/.profile
/var/
/var/mail/
/var/opt/
/var/lib/
/var/lib/shells.state
/var/lib/misc/
/var/lib/apt/
/var/lib/apt/extended_states
/var/lib/apt/periodic/
/var/lib/apt/lists/
/var/lib/apt/lists/security.ubuntu.com_ubuntu_dists_jammy-security_main_binary-amd64_Packages.lz4
/var/lib/apt/lists/security.ubuntu.com_ubuntu_dists_jammy-security_multiverse_binary-amd64_Packages.lz4
/var/lib/apt/lists/archive.ubuntu.com_ubuntu_dists_jammy-updates_multiverse_binary-amd64_Packages.lz4
/var/lib/apt/lists/archive.ubuntu.com_ubuntu_dists_jammy-updates_InRelease
/var/lib/apt/lists/archive.ubuntu.com_ubuntu_dists_jammy-updates_universe_binary-amd64_Packages.lz4
/var/lib/apt/lists/archive.ubuntu.com_ubuntu_dists_jammy_universe_binary-amd64_Packages.lz4
```

Output of script 1 hardening_script1

```
 GNU nano 6.2                                  hardening_script2.sh
#!/bin/bash

# Variable for the report output file, choose a NEW output file name

REPORT_FILE_2="/report_file_2.txt"

# Ensure the script has write permission to the output file

touch $REPORT_FILE_2 2>/dev/null || { echo "Error: Cannot write to $REPORT_FILE_2. Check permissions."; exit 1; }

# Output the sshd configuration file

echo "Gathering details from sshd configuration file"

echo "sshd configuration file:" >> $REPORT_FILE_2

cat /etc/ssh/sshd_config >> $REPORT_FILE_2

printf "\n" >> $REPORT_FILE_2

# Update packages and services

echo "Updating packages and services"

echo "Updating package list..." >> $REPORT_FILE_2

sudo apt-get update >> $REPORT_FILE_2 2>&1

echo "Upgrading packages..." >> $REPORT_FILE_2

sudo apt-get upgrade -y >> $REPORT_FILE_2 2>&1

echo "Packages have been updated and upgraded" >> $REPORT_FILE_2

printf "\n" >> $REPORT_FILE_2

# List all installed packages

echo "Listing all installed packages..."

echo "Installed Packages:" >> $REPORT_FILE_2

dpkg -l >> $REPORT_FILE_2

printf "\n" >> $REPORT_FILE_2

# Printing out logging configuration data

echo "Printing out logging configuration data"

echo "journald.conf file data:" >> $REPORT_FILE_2

cat /etc/systemd/journald.conf >> $REPORT_FILE_2

printf "\n" >> $REPORT_FILE_2

# Display logrotate configuration

echo "logrotate.conf file data:" >> $REPORT_FILE_2

cat /etc/logrotate.conf >> $REPORT_FILE_2

printf "\n" >> $REPORT_FILE_2

# Final message

echo "Script execution completed. Check $REPORT_FILE_2 for details."
```

Hardening script 2
This script updates your packages and output the log files into a file called report_file_2.txt

```
root@Baker_Street_Linux_Server:/etc/cron.weekly# ./hardening_script2.sh
Gathering details from sshd configuration file
Updating packages and services
Listing all installed packages...
Printing out logging configuration data
Script execution completed. Check /report_file_2.txt for details.
root@Baker_Street_Linux_Server:/etc/cron.weekly#
```

Output for hardening_script2.sh

		root@Baker_Street_Linux_Server:/# cat report_file_2.txt sshd configuration file: # This is the sshd server system-wide configuration file. See # sshd_config(5) for more information. # This sshd was compiled with PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games # The strategy used for options in the default sshd_config shipped with # OpenSSH is to specify options with their default value where # possible, but leave them commented. Uncommented options override the # default value. Include /etc/ssh/sshd_config.d/*.conf Port 22 #AddressFamily any #ListenAddress 0.0.0.0 #ListenAddress :: logrotate.conf file data: # see "man logrotate" for details # global options do not affect preceding include directives # rotate log files daily daily # use the adm group by default, since this is the owning group # of /var/log/syslog. su root adm # keep 1 weeks worth of backlogs rotate 1 # create new (empty) log files after rotating old ones create # use date as a suffix of the rotated file #dateext # uncomment this if you want your log files compressed #compress # packages drop log rotation information into this directory include /etc/logrotate.d # system-specific logs may also be configured here. Summary of the contents of file_report.txt
☑	Scripts scheduled with cron	# Edit this file to introduce tasks to be run by cron. # # Each task to run has to be defined through a single line # indicating with different fields when the task will be run # and what command to run for the task # # To define the time you can provide concrete values for # minute (m), hour (h), day of month (dom), month (mon), # and day of week (dow) or use '*' in these fields (for 'any'). # # Notice that tasks will be started based on the cron's system # daemon's notion of time and timezones. # # Output of the crontab jobs (including errors) is sent through # email to the user the crontab file belongs to (unless redirected). # # For example, you can run a backup of all your user accounts # at 5 a.m every week with: # 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/ # # For more information see the manual pages of crontab(5) and cron(8) # # m h dom mon dow command # this is run monthly 0 0 1 * * /etc/cron.monthly/hardening_script1.sh #this runs weekly 0 0 * * 1 /etc/cron.weekly/hardening_script2.sh

		In the cron file it is listed that **hardening_script1.sh** runs monthly and **hardening_script2.sh** is running weekly

Project Summary

This document provides a detailed overview of the measures taken to enhance the security and stability of the **Linux server** deployed at Baker Street Corporation. Below is an expanded summary of the key details and activities described in the document.

1. Operating System Backup

- The entire filesystem was backed up into a compressed .tar.gz archive.
- Specific directories, such as those containing temporary files or logs, were excluded from the backup to save storage space and avoid redundant data.
- This ensures a comprehensive disaster recovery strategy, allowing quick restoration of the system in the event of failure or compromise.

2. User and Group Management

- **User Auditing**: A complete audit identified all users on the system. Terminated employees were promptly removed to eliminate potential security risks.
- **Account Locking**: Temporary leaves were addressed by locking accounts rather than removing them, preventing unauthorized access while preserving data integrity.
- **Password Management**: Locked accounts were reset and reactivated securely using strong, unique passwords.
- **Group Structuring**:
    - New groups were created for specific departments (e.g., "research") to streamline permissions.
    - Redundant groups, such as "marketing," were deleted to reduce administrative complexity.
    - Members were assigned to groups using efficient scripting, ensuring that permissions align with job responsibilities.

---

3. Password Policy Enforcement

- Password complexity policies were strengthened to include requirements such as:
    - A minimum length of 8 characters.

    - Mandatory use of at least one uppercase letter, lowercase letter, number, and special character.
- Changes were implemented in the Pluggable Authentication Module (PAM) configuration file and tested rigorously.
- By enforcing these standards, weak passwords were eliminated, reducing the risk of brute-force attacks.

---

4. Sudo Permissions

- Sudo privileges were reconfigured to ensure that:
    - Administrators, such as Sherlock, have full access to all commands without requiring repeated authentication.
    - Users in roles like Watson and Mycroft were granted limited privileges for specific administrative tasks.
    - Group-based permissions (e.g., for "research") were implemented to centralize and simplify permission management.
- This minimizes the potential for privilege escalation by restricting unnecessary administrative access.

5. File and Directory Permissions

- A thorough scan identified files with inappropriate world-readable, writable, or executable permissions.
- Permissions were revised using standardized practices (e.g., chmod 644) to ensure:
    - Owners have appropriate control over files.
    - Groups and others have limited access to sensitive data.
- Scripts critical to organizational functions were reassigned to the appropriate groups (e.g., "research" or "engineering") for controlled access.

6. SSH Configuration

- Security enhancements for SSH included:
    - Disabling root login to prevent unauthorized administrative access.
    - Prohibiting empty password logins to enforce user-specific credentials.
    - Restricting SSH access to secure port 22 and disabling less commonly used ports.
    - Enabling Protocol 2, a more secure alternative to Protocol 1, to safeguard communication.
- After making changes, the SSH service was restarted to apply these configurations immediately.

7. Package and Service Management

- Regular updates (apt update and apt upgrade) were performed to ensure the latest security patches were applied.
- Insecure and deprecated services, such as Telnet and Remote Shell (RSH), were removed due to their plaintext communication and lack of robust authentication.
- Unnecessary services, including MySQL and Samba, were identified, disabled, and completely removed to minimize attack surfaces.

8. Logging and Monitoring

- Logging configurations were updated to improve system auditing:
    - Logs were set to persist across reboots by enabling storage=persistent in the journaled configuration.
    - A log size limit of 300 MB was defined to prevent excessive disk usage.
    - Log rotation frequency was increased from weekly to daily, with retention reduced to seven days, ensuring timely and efficient log management.

9. Automation

- Two hardening scripts were developed and scheduled using cron jobs:
    - **hardening_script1.sh**: Executes monthly to back up the OS and manage user/group permissions.
    - **hardening_script2.sh**: Runs weekly to handle additional hardening tasks, such as auditing files and verifying configurations.
- Automating these tasks reduces manual intervention and ensures consistent adherence to security policies.

Conclusion

This report provides a clear and thorough overview of how the Linux server at Baker Street Corporation was secured. The hardening process addressed multiple key areas, such as system configurations, user and group management, file permissions, and service optimization. By combining well-planned manual adjustments with automated scripts, the system is now more resilient to potential threats, offering improved security, stable performance, and strong data protection.