Cybersecurity

Project 1 Hardening Summary and Checklist

OS Information

Customer	Baker Street Corporation
Hostname	**Baker_Street_Linux_Server**
OS Version	**Version="22.04.5 LTS (Jammy Jellyfish)"**
Memory information	**total used free shared buff/cache available** **Mem: 15Gi 1.4Gi 7.3Gi 184Mi 6.7Gi 13Gi** **Swap: 0B 0B 0B**
Uptime information	**Up 27 minutes**

Checklist

Completed	Activity	Script(s) used / Tasks completed / Screenshots
☑	OS backup	Screenshot (2).png Screenshot (4).png

☑	Auditing users and groups	First I identified the terminated staff as Lestrade, Irene, Mary, and Gregson. I then proceeded to remove all terminated staff along with all home directories and files by doing the following commands: *deluser –remove-home <terminated_staff_name>* Screenshot (9).png Second I Identified all staff members on temporary leave as Moriarty and Mrs_Hudson. I then proceeded to lock all user accounts on temporary leave by doing the following commands: *usermod -L <temporary_leave_staff_name>* To verify the accounts were indeed locked I used the the following commands: *passwd -S <temporary_leave_staff_name>* Screenshot (10).png Then I identified all employees that are currently listed as “employed.” The employed list consisted of Sherlock, Watson, Mycroft, Toby, and Adler. Then proceeded to verify if their accounts were locked/unlocked running the commands: *passwd -S <employed_staff_name>* From the Employed List only Toby and Adler accounts were locked. To unlock their accounts and give them a password I ran the commands: *sudo passwd <employed_staff_name>* Then verified that Toby and Adler accounts were unlocked by running the commands: *passwd -S <employed_staff_name>* Screenshot (12).png Next I looked at all the employees that were in the marketing group. After running the commands: *getent group marketing* Noticed the marketing group had no members. I then created the group “research” that did not previously exist by using the following commands: *addgroup research*

		Since the marketing group had no members there were no members to move in the research group. Lastly, I removed the marketing group by using: *delgroup marketing* Screenshot (11).png
☑	Updating and enforcing password policies	I first navigated to the common-password file by using the path */etc/pam.d/common-password*. I then ran the command *sudo nano /etc/pam.d/common-password* to edit the file. Next I added to the password requisite line to update password requirements for all users *password requisite pam_pwquality.so minlen=8 ocredit=1 retry=2 ucredit=-1* **minlen=8** Sets a minimum password length of 8 characters. **ocredit=-1** Requires at least one special character **retry=2** Allows 2 retries for incorrect passwords.

		ucredit=-1 Requires at least one uppercase letter. Screenshot (14).png
☑	Updating and enforcing sudo permissions	To update and enforce sudo permissions, I started with editing the sudoers file by running the command *sudo visudo* Sherlock is the only employee who should have full sudo privileges. Watson and Mycroft should only have sudo privileges to run the following script: */var/log/logcleanup.sh* And all employees in the **research** group should only have privileges to run the following script: */tmp/scripts/research_script.sh* Listed below are the edits that were made in the sudoers file: *sherlock ALL=(ALL:ALL) ALL* *watson ALL=(ALL) NOPASSWD: /var/log/logcleanup.sh* *mycroft ALL=(ALL) NOPASSWD: /var/log/logcleanup.sh* *%research ALL=(ALL) NOPASSWD: /tmp/scripts/research_script.sh* Screenshot (15).png Screenshot (16).png

☑	Validating and updating permissions on files and directories	To validate and update permissions on files and directories the following steps were taken: This command finds files that have world permissions and removes them: *find /home -type f -perm /o+rwx -exec chmod o-rwx {} +* This command finds and ensures directories in home folders have no world access: *find /home -type d -perm /o+rwx -exec chmod o-rwx {} +* The command that allows only **engineering** group to have access: *find /home -type f -iname '*engineering*' -exec chown :engineering {} + -exec chmod 770 {} +* The command that allows only **research** group to have access: *find /home -type f -iname '*research*' -exec chown :research {} + -exec chmod 770 {} +* The command that allows only **finance** group to have access: *find /home -type f -iname '*finance*' -exec chown :finance {} + -exec chmod 770 {} +* This command checks if any files still have world permissions: *find /home -type f -perm /o+rwx* Lastly this command will list the updated files so you can verify the changes: *find /home -type f \(-iname '*engineering*' -o -iname '*research*' -o -iname '*finance*' \) -ls* Screenshot (17).png I did not find any hidden files containing passwords after I ran the command: *find /home -type f -name ".*" -exec grep -Ei 'password|passwd|pwd|secret|key|token' {} + 2>/dev/null* Screenshot (65).png However, after doing a search on the home directory hidden files using the command *find /home -type f -iname ".*"*. I discovered several employees had hidden files containing passwords. I removed the files by the command: *rm -f /path/to/hidden/file* *Screenshot (67).png* *Screenshot (68).png* *Screenshot (69).png*

☐	Optional: Updating password hashing configuration	
☑	Auditing and securing SSH	Started with the command *sudo nano /etc/ssh/sshd*_config to start making edits. PermitEmptyPasswords yes, what changed to **PermitEmptyPasswrds no** PermitRootLogin yes, what changed to **PermitRootLogin no** Screenshot (24).png Removed all other Ports, then uncommented the line that contained Port 22. SSH with any other ports besides 22 Screenshot (25).png Removed all lines containing different Protocols beside Protocol 2 Screenshot (27).png Verified changes were correct the running the following commands: *sudo sshd -t* *sudo grep "^PermitEmptyPasswords" /etc/ssh/sshd_config* *sudo grep "^PermitRootLogin" /etc/ssh/sshd_config* *sudo grep "^Port" /etc/ssh/sshd_config* *sudo grep "^Protocol" /etc/ssh/sshd_config* Screenshot (26).png Then applied changes by running the command: *sudo service ssh restart*
☑	Reviewing and updating system packages	To Review and update system packages. I first ran the command *sudo apt update* to ensure I was working with the latest version of all packages. Next I ran *sudo apt upgrade -y* to update all installed packages to their latest versions. To view installed packages I ran the command *sudo apt list –installed.* Also created the file *package_list.txt*. To verify the file was created I ran Screenshot (28).png

		Screenshot (74).png Screenshot (30).png Screenshot (29).png Next I Identified the packages **telnet** and **rsh-client.** **telnet** and **rsh-client** are considered to be security risks. **Telnet** is considered a risk because it transmits data such as passwords in human-readable form (plaintext) making it insecure. **RSH (Remote Shell)** uses unencrypted communication exposing itself to attacks like MITM (man-in-the-middle attacks). Screenshot (33).png Then I installed the packages **ufw**, **lynis**, and **tripwire** by running the command: *sudo apt install -y ufw lynis tripwire* Too verify the packages were installed I used the command: *which <package_name>* To verify with more detail I used the command: *dpkg -l | grep <package_name>* Screenshot (34).png The hardening features these packages provide are as follows: **UFW (Uncomplicated Firewall)** ● Makes firewall management simple. ● Allows necessary connections while at the same time blocking unauthorized ones. ● Allows firewall rules to be easily configured **Lynis** ● Security auditing tool for Linux. ● Looks for vulnerabilities, misconfigurations, and gaps in security. ● Makes suggestions to improve system security **Tripwire** ● Is an intrusion detection system **(IDS**) ● Will monitor file integrity and alerts when files are suddenly modified. ● Can help detect unauthorized modifications to critical system files

☑	Disabling unnecessary services	To start the process of reviewing and disabling unnecessary services, I first ran the command *service --status-all > service_list.txt* to list all services then output them into the file call **service_list.txt.** Next I identified the running services were **mysql** and **samba** by using the command: *grep -E 'mysql|samba' service_list.txt.* Then I stopped the services by running the command: *sudo service <service_name> stop* Disabled mysql and samba by using the command: *sudo systemctl disable <service_name>* Next I removed the services running command: *sudo apt remove -y <system_name> <system_name>* Lastly I remove any unused dependencies by using: *sudo apt autoremove -y* Screenshot (35).png Screenshot (36).png Screenshot (37).png

☑	Enabling and configuring logging	Started by opening the journald.conf file for editing using *sudo nano /etc/systemd/journald.conf*. Found and updated the following lines: **Storage=persistent** **SystemMaxUse=300** **Storage=persisten**t saves logs permanently to your local machine instead of using RAM **SystemMaxUse=300** Restricts the log storage usage to 300MB to prevent excessive disk space usage Next I edited the logrotate.conf file by running the command *sudo nano /etc/logrotate.conf* I then found and update the following lines: # rotate log files weekly Daily # keep 4 weeks worth of backlogs Rotate 7 **Daily** Ensures logs are rotated every day instead of weekly. **Rotate 7** Keeps logs for **7 days** before deleting old ones. Lastly I verified and tested the log rotation changes by running the command: *sudo logrotate -d /etc/logrotate.conf* Screenshot (38).png Screenshot (39).png Screenshot (41).png Screenshot (42).png Screenshot (43).png Screenshot (44).png Screenshot (45).png Screenshot (46).png Screenshot (47).png

☑	Scripts created	I started by creating an output file named ***bcs_project1.txt.*** This is where the output of the **hardening_script1.sh** will be saved. Then created a backup of the **hardening_script1.sh** by running the following command: *sudo cp hardening_script1.sh /var/backups/hardening_script1.sh* Then I proceeded to edit the script by running: *sudo nano hardening_script1.sh* After completing and saving the script I then made the script executable by running the command: *chmod +x hardening_script1.sh* Then I ran the script using the command: *sudo ./hardening_script1.sh* Screenshot (48).png Copy of hardening_script_1.sh Screenshot (49).png For the second script I started by creating another output file named ***bcs2_project1.txt.*** This is where the output of the **hardening_script2.sh** will be saved. Screenshot (52).png Screenshot (50).png Then created a backup of the **hardening_script2.sh** by running the following command: *sudo cp hardening_script1.sh /var/backups/hardening_script2.sh* Then I proceeded to edit the script by running: *sudo nano hardening_script2.sh* After completing and saving the script I then made the script executable by running the command: *chmod +x hardening_script2.sh* Then I ran the script using the command: *sudo ./hardening_script2.sh* Copy of hardening_script_2.sh Screenshot (53).png

		Screenshot (54).png Screenshot (55).png Screenshot (56).png Screenshot (57).png Screenshot (58).png Screenshot (59).png Screenshot (60).png Screenshot (61).png
☑	Scripts scheduled with cron	To schedule scripts with cron I used the following command to start editing the crontab: *crontab -e* To schedule hardening_script1.sh to run Once a month on the first of the month I inputted: ***0 0 1 * * /hardening_script1.sh*** To schedule hardening_script2.sh to run Once a week every Monday, I inputted: ***0 0 * * 1 /hardening_script2.sh*** Screenshot (62).png Screenshot (63).png

Project 1 Technical Brief Summary Report

Prepared by: Chontele Coleman

The Baker Street Corporation (BSC)

As a security professional I was contacted by **The Baker Street Corporation (BSC)** to harden a Linux server they owned. The BSC Linux server contains sensitive and highly confidential data that they would like to keep secure. Over a three day period I was given the tasks to confirm that their system was indeed properly configured to protect them from security breaches, as well as determine any security issues I came across along the way. If any security issues did arise I was given the authority to make the necessary updates.

Before getting started with my security analyst of the BSC Linux server. It is good practice to record vital information pertaining to the system I will be working on. Such as, host name of the server, OS version, memory info, and uptime information. Last and certainly not least I performed an OS backup. Performing an OS backup can help recover from hardware failure, accidental deletions, virus attacks, or other occurrences.

Day 1: The main takeaway of day 1 is the **principle of least privilege (PoLP)**. According to Wikipedia the definition of **PoLP, requires that in a particular abstraction layer of a computing environment, every module must be able to access only the information and resources that are necessary for its legitimate purpose.** Because employees are your first line of defense and often the most vulnerable, It is important to limit employee access to unnecessary information in case their user accounts are compromised by an attack.

I was given a list of employees that detailed their current status with BSC and performed and executed the following tasks:

- Removed all staff that had been terminated including all home directories and files associated with those staff members.
- Locked all user accounts of staff on temporary leave.
- Unlocked staff members whose status was listed as currently employed.
- Moved all the employees who were in the marketing department to a new group called research. Research group did not previously exist so it had to be created.
- Removed the marketing group because that department closed this year.
- Updated password requirements for all users based on the criteria given by BCS.
- Gave full sudo privileges to the employee specified by BSC and removed all other full privileged employees
- Gave sudo privileges to specified employees to run scripts in a designated location
- Employees in the research group were given sudo privileges only to run a designated script.
- Removed all **read, write, execute (RWX)** permissions from every user's files in their home directory.
- Located employee files with hidden passwords. Removed those files because storing passwords on the server is not only prohibited but poses a security risk to the corporation and the employee.

Day 2: On this day we audited, reviewed, updated packages, disabled unnecessary services, as well as enabled and configured logging. These practices are key to the security and stability of the system. They can also assist and help reduce unrealized exposures confirming only necessary services and packages are running on the system. Keeping

these this in good standing greatly reduces the attack surface of the Baker Street Linux Server.

- Configured SSH to not allow the ability to: SSH with empty passwords, SSH with the root user, SSH with any other ports besides Port 22
- Enabled Protocol 2
- Restart the SSH service to set my updates
- Updated the package manager to ensure I was working with the latest version of all packages
- Upgraded all packages to ensure I was working with the latest versions.
- Created a file that contains all installed packages
- Identified telnet and rsh-client as high security risk packages for the following reasons:
    - **telnet** and **rsh-client** are considered to be security risks. **Telnet** is considered a risk because it transmits data such as passwords in human-readable form (plaintext) making it insecure. **RSH (Remote Shell)** uses unencrypted communication exposing itself to attacks like **MITM (man-in-the-middle attacks).**
- Because of the above mentioned concerns telnet and rsh-client were removed along with all unnecessary dependencies of those packages.
- Added the **ufw, lynis, tripwire.** The benefits of these packages are listed below:
  UFW (Uncomplicated Firewall)

    - Makes firewall management simple.
    - Allows necessary connections while at the same time blocking unauthorized ones.
    - Allows firewall rules to be easily configured

  Lynis

    - Security auditing tool for Linux.
    - Looks for vulnerabilities, misconfigurations, and gaps in security.
    - Makes suggestions to improve system security

  Tripwire

    - Is an intrusion detection system **(IDS)**
    - Will monitor file integrity and alerts when files are suddenly modified.
    - Can help detect unauthorized modifications to critical system files
- Ran command to list out all services and created an output file
- Identified that the services **mysql** and **samba** were running. Stopped, Disabled, and removed those services
- Accessed the journald.conf and proceeded to make edits:

        - **storage=persistent** this setting will save logs locally on the machine.
        - **systemMaxUse300M** configures the maximum disk space the logs can utilize.
    - To Prevent logs from taking up too much space it was essential to edit the logrotate.conf file.
        - Changed log rotation from weekly to daily
        - Rotated out logs after 7 days

Day 3: Built out 2 scripts to automate the tasks completed over the last 2 days. The first script will cover day one's tasks. The second script will cover day 2 tasks. The automation of tedious task like backing up files, monitoring system resources, and overseeing user accounts can increase efficiency along with an increase in accuracy and help to simplify tasks.

- After completing the scripts and checking them for errors they were ready to be scheduled to perform at specific times using cron:
    - Script 1 is scheduled to run Once a month on the first of the month.
    - Script 2 is scheduled to run Once a week every Monday.

In conclusion, we learned how to greatly reduce the attack surface of the Baker Street Corporation Linux server by following the model of the **CIA triad (Confidentiality, Integrity, Availability)**. **Confidentiality** safeguards personal and company information. **Integrity** confirms verifiability and defends against unauthorized changes to the system. **Availability** makes sure access to the system information can be done as timely and reliably as possible.

My final thought is all the changes that were made in my project to the BSC Linux Server. There is no such thing as 100% protection from a malicious attack. Recommendations I can give is to educate your staff about what to do with personal information as well as handling sensitive company data. Examples of this are not leaving an unattended laptop open signed into a BSC user account at a coffee shop or airport. Clicking on links inside of emails that look suspicious or from unknown senders. Clicking on links from legitimate looking emails with misspellings. My last recommendation to the BSC is to have a contingency plan in case a cyberattack or data breach occurs.