

**OS Information**

| Customer | Baker Street Corporation |
| --- | --- |
| Hostname | Baker\_Street\_Linux\_Server |
| OS Version | x86\_64-pc-linux-gnu |
| Memory information |  |
| Uptime information |  |

**Checklist**

| **Completed** | **Activity** | **Script(s) used / Tasks completed / Screenshots** |
| --- | --- | --- |
|  |  |  |
|  | OS backup | **Part 1**  The first part of server hardening is taking inventory on our server. We will need to collect data on the Host, OS, memory and uptime. After that we will back the whole thing up so that any changes we make can be rolled back if needed. In our backup we do not include the proc, tmp, mnt, sys, dev, and run folders. This is because those folders contain only temporary or runtime data that does not need to be backed up.  echo -e $MACHTYPE    hostname -s    free    uptime    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 / |
|  | Auditing users and groups | **Part 2**  Next we will audit our users and groups. In this scenario, we have 4 employees who are no longer with the company. We will delete their accounts with the deluser command. We use the --remove-home as a best practice to ensure the home directory and related files are removed. These files could contain sensitive data and another user with the correct permissions could take advantage of this. This is important in reducing the attack surface.  Next we have some users on leave. We do not want someone to be able to access their account or sensitive files while they are not in use. We will use passwd -l to lock the account of the user. Doing this sets appends a ! to the beginning of the password hash making the string invalid. For our active users and when the employees on leave return we will use passwd -u to ensure the account is not locked and the user can log in to complete their work. \*notably when unlocking user accounts toby and adler we receive a message indicating they do not have passwords set. We can set the password on their accounts to a temporary one and set it to expire so they are prompted to create a new one upon successful log in.  Finally, we were instructed to move everyone in marketing to research. Unfortunately, the project environment was not prepared correctly so we will use our imagination and assume these poor employees were the ones let go. However, to demonstrate my understanding we would have created a new group called research and used usermod to add the desired users into that group. We would then delete the marketing group.  deluser --remove-home irene    passwd -l moriarty    passwd -u sherlock    groups \*    addgroup research    For some reason my marketing group is empty. I would have used:  usermod -G research username  delgroup marketing |
|  | Updating and enforcing password policies | **Part 3**  Next we will enforce a stricter password policy for our users. For this scenario we will enforce a minimum of 8 characters with 1 special character, one uppercase, and 2 retries. However, best practices would likely have number requirements as well and in recent times a password length of 12 characters.  nano /etc/pam.d/common-password  password requisite pam\_pwquality.so minlen=8 ocredit=1 ucredit=1 retry=2 |
|  | Updating and enforcing sudo permissions | **Part 4**  Sudo permissions are very critical as they prevent users from using commands and files they should not have access to. Additionally, they allow users to have escalated privileges when necessary to complete tasks. I have displayed the before and after screenshots of the sudoers file. We have removed sudo permissions from everyone except sherlock. Sherlock was changed to still require a password. Additionally, watson and moriarty now only have permissions to run a specific script as sudo. |
|  | Validating and updating permissions on files and directories | Part 5  Permissions on files are also important. I will show the before and after of our file trees to show changes. We will first remove all world permissions. This reduces our attack surface by ensuring people without privileges cannot access any files. We are tasked with changing the files ownership to specific groups and to ensure that group is the only one able to read, write, or execute the scripts.      chmod /home/ -R o= \*      **Part 5 - Rationale**  The instructions are to update specifically the engineering and finance scripts so that only members of the group can access it. Therefore I changed the owner of specifically those files to the respective group as due to lab issues finance scripts were in directories belonging to engineers and vice versa. By executing the below commands I changed the owners of the scripts to the appropriate group and gave that group the appropriate permissions.  find -iname '\*engineering\*' -exec chown :engineering {} +  find -iname '\*engineering\*' -exec chmod g=wrx {} +  find -iname '\*finance\*' -exec chown :finance {} +  find -iname '\*finance\*' -exec chmod g=wrx {} +    I did a cat \* for each user's home directory and found no passwords or file content??? Seems like a productive group of employees at this company… Only including 1 screenshot as the rest were the same and redundant. This is due to a lab environment issue. In a larger org where using cat to view everyone's files is not feasible we could grep files for keywords like “password”, or we could directly search the plaintext password to see if it is stored anywhere. |
|  | Optional: Updating password hashing configuration | **Optional Task**  We are already using the yes script. After doing some research I was able to determine this is a strong hashing configuration so I reverted the changes made in the screenshot below. However, to demonstrate how you would update hashing I edited the line to use sha512 as the hashing algorithm. According to my research sha 512 is not as strong as yesscript. |
|  | Auditing and securing SSH | **Part 6**  To audit our ssh settings we will nano into sshd\_config. Within this file We need to uncomment certain lines and change their controls. We do not want root login, or empty passwords. We will only accept connections on port 22. We want to be using protocol 2. To finalize this we will restart the ssh server.  nano /etc/ssh/sshd\_config      We will uncomment port 22 and delete the cheeky ports at the bottom of the file        service ssh restart |
|  | Reviewing and updating system packages | **Part 7**  System packages need to be kept up to date to reduce known vulnerabilities. We will use update and upgrade to make sure we are running the newest version of all packages. Then we will list all of these packages to submit in our report. Finally, we will remove packages that impose a security risk and add some packages that will help harden the server.  apt update    apt upgrade -y    touch package\_list.txt  apt list --installed > package\_list.txt  cat package\_list.txt    grep 'telnet\|rsh-client' package\_list.txt    apt remove rsh-client  apt remove telnet    apt autoremove -y    Telnet and rsh-client transmit data and passwords in plaintext which in today's age is a massive security issue.  apt -y install ufw lynis tripwire    UFW is an uncomplicated firewall that simplifies configuring the firewall. This helps ensure the firewall is configured properly  Linus can run scans and provide feedback on your server's hardening. It can be used to address vulnerabilities.  Tripwire is a host based intrusion detection system that is used to alert admin to any unauthorized changes. |
|  | Disabling unnecessary services | **Part 8**  Here we will compile our services into a text document for reporting purposes. Mysql, and samba need to be removed. If these services are not needed then they should be removed to reduce the attack surface and free up resources.  service –status-all  hwclock.sh status was not able to be determined. So it does not get written to the file. |
|  | Enabling and configuring logging | **Part 9**  Here we want to configure our logs to be created daily, be persistent and not take up too much space on the system. To do this we will set limitations to size as well as rotate the logs on a weekly basis.  nano /etc/systemd/journald.conf    nano /etc/logrotate.conf |
|  | Scripts created | **Part 10**  Automating tasks can greatly impact productivity and allow analysts to focus on more important tasks. Here I will be automating all of the tasks we completed prior in two separate scripts. I have pasted the content of the scripts and included some screenshots of the output of bot the echo lines showing the script is working and the report files that were created to hold our system information.  #!/bin/bash  # Variable for the report output file, choose an output file name  REPORT\_FILE="report1.txt"  # Output the hostname  echo "Gathering hostname..."  # Placeholder for command to get the hostname  echo "Hostname: $(hostname -s)" >> $REPORT\_FILE  printf "\n" >> $REPORT\_FILE  # Output the OS version  echo "Gathering OS version..."  # Placeholder for command to get the OS version  echo "OS Version: $(uname -o)" >> $REPORT\_FILE  printf "\n" >> $REPORT\_FILE  # Output memory information  echo "Gathering memory information..."  # Placeholder for command to get memory info  echo "Memory Information: $(free -h)" >> $REPORT\_FILE  printf "\n" >> $REPORT\_FILE  # Output uptime information  echo "Gathering uptime information..."  # Placeholder for command to get uptime info  echo "Uptime Information: $(uptime)" >> $REPORT\_FILE  printf "\n" >> $REPORT\_FILE  # Backup the OS  echo "Backing up the OS..."  # Placeholder for command to back up the OS  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 /  echo "OS backup completed." >> $REPORT\_FILE  printf "\n" >> $REPORT\_FILE  # Output the sudoers file to the report  echo "Gathering sudoers file..."  # Placeholder for command to output sudoers file  echo "Sudoers file:$(visudo -V)" >> $REPORT\_FILE  printf "\n" >> $REPORT\_FILE  # Script to check for files with world permissions and update them  echo "Checking for files with world permissions..."  chmod -R o= \*  # Placeholder for command to find and update files with world permissions  echo "World permissions have been removed from any files found." >> $REPORT\_FILE  printf "\n" >> $REPORT\_FILE  # Find specific files and update their permissions  echo "Updating permissions for specific scripts..."  # Engineering scripts - Only members of the engineering group  echo "Updating permissions for Engineering scripts."  # Placeholder for command to update permissions  find -iname '\*engineering\*' -exec chown :engineering {} +  find -iname '\*engineering\*' -exec chmod g=wrx {} +  echo "Permissions updated for Engineering scripts." >> $REPORT\_FILE  printf "\n" >> $REPORT\_FILE  # Research scripts - Only members of the research group  #echo "Updating permissions for Research scripts..."  # Placeholder for command to update permissions  #Place command here to only allow members of ^`^|research ^`^} group to view, edit, and execute all research scripts. See above script for syntax.  #echo "Permissions updated for Research scripts" >> $REPORT\_FILE  #printf "\n" >> $REPORT\_FILE      report1.txt      #!/bin/bash  # Variable for the report output file, choose a NEW output file name  REPORT\_FILE="report2.txt"  # Output the sshd configuration file  echo "Gathering details from sshd configuration file"  # Placeholder for command to get the sshd configuration file  echo "sshd configuration file:$(cat /etc/ssh/sshd\_config)" >> $REPORT\_FILE  printf "\n" >> $REPORT\_FILE  # Update packages and services  echo "Updating packages and services"  # Placeholder for command to update packages  apt update  # Placeholder for command to upgrade packages  apt upgrade -y  echo "Packages have been updated and upgraded" >> $REPORT\_FILE  printf "\n" >> $REPORT\_FILE  # Placeholder for command to list all installed packages  echo "Installed Packages:$(apt list --installed)" >> $REPORT\_FILE  printf "\n" >> $REPORT\_FILE  echo ^`^|Printing out logging configuration data ^`^}  # Placeholder for command to display logging data  echo "journald.conf file data: $(cat /etc/systemd/journald.conf)" >> $REPORT\_FILE  printf "\n" >> $REPORT\_FILE  # Placeholder for command to display logrotate data  echo "logrotate.conf file data:$(cat /etc/logrotate.conf)" >> $REPORT\_FILE  printf "\n" >> $REPORT\_FILE  echo "Script execution completed. Check $REPORT\_FILE for details." |
|  | Scripts scheduled with cron | **Part 13**  To further automation so that we do not need to run scripts daily we will use crontabs to ensure the scripts are run routinely. Doing this ensures the task is always completed which is necessary if 3rd party auditing demands it. I have placed this cron schedule in the root users crontab. This means the root user will be the one running the scripts. Alternatively, we could have placed these crons in the cron directories so that they would be system wide.  crontab -e  0 0 1 \* \* /hardening\_script1  0 0 \* \* 1 /hardening\_script2 |