**Linux Command Synopsis & Bash Scripts**

**cron: Create, manage and automate various cron jobs.**

In response to a ransomware attack, you have been tasked with creating an archival and backup scheme to mitigate against CryptoLocker malware. This attack would encrypt the entire server’s hard disk and it can only be unlocked using a 256bit digital key after payment using bitcoin.

1. Create a cron job using the following specs:  
     
   * Create archive for the following directory: /var/log/auth.log
   * Save the archive to the following directory: /
   * Name the archive: auth\_Backup.tgz
   * Run every Wednesday at 6AM
   * Compress the archive using gzip

**mkdir /var/log/auth.log - Creates directory /var/log/auth.log  
crontab -e - opens crontab editor  
0 6 \* \* 3 /var/spool/cron/crontabs/crontab1.sh - 0 = minute; 6 = hour; \* = match any; \* = match any; 3 = Wednesday; /var/spool/cron/crontabs/crontab1.sh executes the crontab1 script located in the /var/spool/cron/crontabs/ directory.  
tar -czvf auth\_Backup.tgz /var/log/auth.log - creates tar backup auth\_Backup.tgz file  
% gzip auth\_Backup.tgz /var/log/auth.log - creates a gzip compressed file of the tar directory backup file auth\_Backup.tgz /var/log/auth.log**

**bash scripting: Write basic bash scripts.**

Portions of the Gramm-Leach-Bliley Act require organizations to maintain a regular backup regimen for the safe and secure storage of financial data.

1. Create an executable Bash script called system.sh using the following specifications:  
     
   * Using brace expansion, create the following four directories to save your work:
     + ~/home/backups/freemem
     + ~/home/backups/diskuse
     + ~/home/backups/openlist
     + ~/home/backups/freedisk

**nano system.sh - Opens the nano editor for a file named system.sh  
mkdir -p {~/home/backups/freemem}, {~/home/backups/diskuse}, {~/home/backups/openlist}, {~/home/backups/freedisk} - command to make the directories (mkdir) if they do not already exist (-p) {~/home/backups/freemem}, {~/home/backups/diskuse}, {~/home/backups/openlist}, {~/home/backups/freedisk}.  In this case, each bracket denotes the specific directory with each bracket separated by the comma (,) and the different directories are strung together using the commas in-between.**

**It is my understanding that it can also be written:**

**mkdir -p {~/home/backups/freemem, ~/home/backups/diskuse, ~/home/backup/openlist, ~/home/backup/freedisk}.  This command makes the directories (mkdir) if they don’t already exist (-p) and each of the directories is listed inside a single set of brackets ({}), separated by a comma (,)  I believe both of these commands will achieve the same results using different syntax.**

**#!/bin/bash  
echo “Let’s See if this Works!” - as a newbie to Linux this is my smartass tagline when trying a new script.  
mkdir -p mkdir -p {~/home/backups/freemem}, {~/home/backups/diskuse}, {~/home/backups/openlist}, {~/home/backups/freedisk}**

**OR**

**#!/bin/bash  
echo “Let’s See if this Works!”as a newbie to Linux this is my smartass tagline when trying a new script.  
mkdir -p mkdir -p {~/home/backups/freemem, ~/home/backups/diskuse, ~/home/backups/openlist, ~/home/backups/freedisk}**

**bash system.sh - executes bash script system.sh**

1. Create a script called system.sh that does the following:  
     
   * Prints the amount of free memory and saves it to ~/home/backups/freemem/free\_mem.txt directory.

Prints disk usage and saves it to ~/home/backups/diskuse/disk\_usage.txt directory.

* + Lists all open files and saves it to ~/home/backups/openlist/openlist.txt directory.
  + Prints file system disk space statistics in “human readable” format and saves it to ~/home/backups/freedisk/freedisk.txt directory.
  + **Note:** Ignore the warning lsof: WARNING: can't stat() fuse.gvfsd-fuse file system /run/user/1001/gvfs Output information may be incomplete.

3. Make the system.sh script executable.  
**Bonus**: Automate your script system.sh by adding it to the weekly system wide cron directory.

**Script**

**Nano system.sh  
#!/bin/bash  
Echo “Let’s See If This Works!”  
Free free | mkdir -p >> /~home/backups/diskuse/disk\_usage.txt  
free used | mkdir -p >> /~home/backups/diskuse/disk\_usage.txt  
lsof | mkdir -p >> /~home/backups/openlist/openlist.txt**

**Script Explanation and Rationale**

**Nano system.sh- opens nano editor and file system.sh  
#!/bin/bash - bash script  
Echo “Let’s See If This Works!” - Smartass comment  
Free free | mkdir -p >>  /~home/backups/diskuse/disk\_usage.txt - The free command (free) orders the amount of free memory (free) to be piped (|) into a new directory and file if it doesn’t exist (-p) ~/home/backups/freemem/free\_mem.txt  
free used | mkdir -p >>  /~home/backups/diskuse/disk\_usage.txt - The free command (free) orders the amount of used memory (used) to be piped (|) and appended to (>>) the file ~/home/backups/diskuse/disk\_usage.txt if the directory file does not already exist (mkdir -p).  
lsof | mkdir -p >> /~home/backups/openlist/openlist.txt - The lsof command data is piped (|) and appended (>>) into the directory file /~home/backups/openlist/openlist.txt if the directory file does not already exist (mkdir -p).  
Cntrl-X - save and exit Nano editor  
$ chmod +x system.sh - The chmod command set the system.sh file as executable (+x)  
Bash system.sh - executes bash script system.sh**

**Bonus:**

**$/etc/cron.weekly - directory file where the weekly cron scripts are scheduled  
crontab -e system.sh - opens file system.sh in the crontab editor  
0 6 \* \* 2 /var/spool/cron/crontabs/system.sh - sets the script to run at 6 AM every Monday from the /var/spool/cron/crontabs/system.sh file**

**journalctl: Perform various log filtering techniques.**

There was a suspicious login from a host on the network during the early morning hours when the office was closed. The Senior Security Manager tasked you with filtering through log files to determine if a system breach occurred.

1. Using journalctl, perform a log search that returns all boot log message priorities for alert and higher, against the current system boot.

**Journalctl -b -l -p “emerg”..”crit”**

1. Use journalctl to check the size of the system disk journal for the most recent boot.  
   **Journalctl --disk-usage and journalctl -b --disk-usage both give the same return.**
2. Use journalctl to remove all archived journals except the most recent two.

**Assuming journals are backed up daily, the journalctl command to remove all archived journals except the most recent two would be: sudo journalctl --vacuum-time=2d**

1. **Bonus**: Use journalctl to filter all log messages with priority levels between 0 and 2 and save the results to a file named Priority\_High.txt in /home/student/ directory.

**mkdir /home/student/directory - makes the directory /home/student/directory  
cd /home/student/directory - changes to directory /home/student/directory  
Nano priority\_filter.sh - opens the nano script editor with the filename priority\_filter.sh  
Journalctl -l -p “emerg”..”crit” | >>Priority\_High.txt - Journaclt command listing (-l) and filtering (-P) “emerg (0)”..”crit(2)” to the file (>>) Priority\_High.txt  
Save and Close - Save and Close  
Chmod +x priority\_filter.sh - sets file to execute  
Bash priority\_filter.sh - executes file**

1. **Bonus**: Automate the last task by creating a cron job that runs daily in the user crontab.

**crontab -e**  
**59 23 \* \* \* /var/spool/cron/crontabs/priority\_filter.sh - this crontab is set to run at the 59th minute of the 23rd hour on every day - A.K.A. 11:59 PM everyday.**

#### rsyslog: Priority based log file creation.

Your organization is constantly bombarded with spam messages, a form of **social engineering** attacks. To address this, you’ve decided to implement a priority based log filtering system to monitor access to the mail daemon.

1. Configure rsyslog to record maillog message for all priorities “EXCEPT” debug to the /var/log/mail.log directory.

**Image 1-2 rsyslog.conf prior to modification; Image 3 rsyslog.conf with modifications.**

1. **Bonus**: Configure rsyslog to record boot log message for priorities except info and debug to the /var/log/boot.log directory.  
    **Boot.notice /var/log/boot.log  
   Boot.warn /var/log/boot.log  
   Boot.err /var/log/boot.log  
   Boot.crit /var/log/boot.log  
   Boot.alert /var/log/boot.log  
   Boot.emerg /var/log/boot.log**

#### logrotate: Manage log file sizes.

Because of the spam messages, you realized that the size of the log files are becoming unmanageable.

You’ve decided to implement log rotation in order to preserve log entries and keep log file sizes more manageable. You’ve also chosen to compress logs during rotation to preserve disk space and lower costs.

1. Configure a log rotation scheme that backs up authentication messages to the /var/log/auth.log directory using the following settings:  
     
   * Rotates weekly.
   * Keep the most recent 7 logs only.
   * Does not rotate empty logs.
   * Delays compression.
   * Skips error messages for missing logs, then continues to next log.

#### auditd: Check for policy and file violations.

In an effort to help mitigate against future attacks, you have decided to create an event monitoring system that specifically generates reports whenever new accounts are created or modified and when any modifications are made to authorization logs.

Verify the auditd service is active using the systemctl command.

1. Using auditd, edit /etc/audit/rules.d/audit.rules and create a rule that watches the following paths:  
     
   * For /etc/shadow, set wra for the permissions to monitor and set the keyname for this rule to hashpass\_audit.

**auditctl -w /etc/shadow -p wra -k hashpass\_audit**

* + For /etc/passwd, set wra for the permissions to monitor and set the keyname for this rule to userpass\_audit.

**auditctl -w /etc/passwd -p wra -k userpass\_audit**

* + For /var/log/auth.log, set wra for the permissions to monitor and set the keyname for this rule to authlog\_audit.

**auditctl -w /var/log/auth.log -p wra -k authlog\_audit**

* + Restart the auditd daemon.
  + Perform a listing that reveals all existing auditd rules.

1. Produce an audit report that returns results for all user authentications.

* **Note:** You will need to log out and back in to populate the report.

1. Time to shift into hacker mode: Create a user sudo useradd attacker then produce an audit report that lists account modifications.
2. Add another rule using auditctl that watches the /var/log/cron directory.  
     
   * Perform a listing that reveals changes to the auditd rules took affect

**## Week 4 Homework Submission File: Linux Systems Administration**

**### Step 1: Ensure Permissions on Sensitive Files**

​

**The following code was used for numbers 1-4 below:**

* **0 = ---**
* **1 = --x**
* **2 = -w-**
* **3 = -wx**
* **4 = r-**
* **5 = r-x**
* **6 = rw-**
* **7 = rwx**

1. Permissions on /etc/shadow should allow only root read and write access.

   - Command to inspect permissions:

**/etc/shadow/ ls -l**

   - Command to set permissions (if needed):

**As we are working in sensitive directories it is assumed that only the network managers/system administrators are owners of files with root access.  If the objective is to allow the users with access to this directory only read and write access displaying -rw- --- --- the command would be:**

**chmod 600 /etc/shadow/ - 6=rw- owner; 0=--- user; 0=--- others = 600**

​

2. Permissions on /etc/gshadow should allow only root read and write access.

   - Command to inspect permissions:

**/etc/gshadow/ ls -l**

   - Command to set permissions (if needed):

**As we are working in sensitive directories it is assumed that only the network managers/system administrators are owners of files with root access.  If the objective is to allow the users with access to this directory only read and write access displaying -rw- --- --- the command would be:**

**chmod 600 /etc/shadow/ - 6=rw- owner; 0=--- user; 0=--- others = 600**

​

​

3. Permissions on /etc/group should allow root read and write access, and allow everyone else read access only.

  - Command to inspect permissions:

**/etc/group/ ls -l**

 - Command to set permissions (if needed):

**As we are working in sensitive directories it is assumed that only the network managers/system administrators are owners of files with root access.  If the objective is to allow the root read and write access to this directory while all others have read access displaying ls -l printout of -rw- r-- r-- the command would be:**

**chmod 644 /etc/group/ - 6=rw- owner; 4=r- user; 4=r- others = 644**

​

​

4. Permissions on /etc/passwd should allow root read and write access, and allow everyone else read access only.

   - Command to inspect permissions:

**/etc/passwd/ ls -l**

   - Command to set permissions (if needed):

**As we are working in sensitive directories it is assumed that only the network managers/system administrators are owners of files with root access.  If the objective is to allow the root read and write access to this directory while all others have read access displaying ls -l printout of -rw- r-- r-- the command would be:**

**chmod 644 /etc/passwd/ - 6=rw- owner; 4=r- user; 4=r- others = 644**

​

**### Step 2: Create User Accounts**

​

1. Add user accounts for sam, joe, amy, sara, and admin.

   - Command to add each user account (include all five users):

**I was unable to run these commands, so the answers below are the best guess based on research.**

​**A useradd command without specified options follows the default settings specified in /etc/default/useradd file.  This command will create entries in the /etc/passwd, /etc/shadow, and etc/gshadow files.  As adding a user is an administrative function, it will not operate without a sudo command:**

**Sudo useradd -m sam**

**Sudo useradd -m joe**

**Sudo useradd -m amy**

**Sudo useradd -m sara**

**Sudo useradd -m admin**

2. Force users to create 16-character passwords incorporating numbers and symbols.

   - Command to edit pwquality.conf file:

**nano /etc/security/pwquality.conf**

   - Updates to configuration file:

**Change the original setting in /etc/security/pwquality.conf from “#minlen = 8 to “minlen = 16” making sure to remove the # so as this line is executed in the script.**

**Change the original setting in /etc/security/pwquality.conf from #dcredit = 0 (number of digits); #ucredit = 0 (number of UPPERCASE); #lcredit = 0 (number of lowercase); and #ocredit = 0 (number of other characters: Examples: 0123456789!@#$%^&\*()) to -> dcredit = 16, ucredit = 0, lcredit = 0, 0credit = 2.  This change would specify a password with a minimum length of sixteen characters for a password, indifferent to the number of capitals or lowercase characters, but requires a minimum of two numbers or symbols.  In this case, it needs to be specified to the new account user that at least one number and one symbol must be incorporated into the password.**

3. Force passwords to expire every 90 days.

   - Command to to set each new user’s password to expire in 90 days (include

   all five users):

**/etc/security/ chage --lastday 90 sam**

**/etc/security/ chage --lastday 90 joe**

**/etc/security/ chage --lastday 90 amy**

**/etc/security/ chage --lastday 90 sarah**

**/etc/security/ chage --lastday 90 admin**

​

4. Ensure that only the admin has general sudo access.

   - Command to add admin to the sudo group:

**sudo visudo to open the sudoers.tmp file in an editor.  Select %sudo line and make the following changes:  admin=(admin:admin) admin.**

**### Step 3: Create User Group and Collaborative Folder**

​

1. Add an engineers group to the system.

   - Command to add group:

**Groupadd engineers**

​

2. Add users sam, joe, amy, and sara to the managed group.

   - Command to add users to engineers group (include all four users):

**sudo usermod -a -G Engineers sam**

**sudo usermod -a -G Engineers joe**

**sudo usermod -a -G Engineers amy**

**sudo usermod -a -G Engineers sara**

**This command gives super user (sudo) privilege to the user moderator (usermod) to append (-a) to the Engineers group (-G) sam, joe, amy, and sara.**

​

3. Create a shared folder for this group at /home/engineers.

   - Command to create the shared folder:

**​mkdir -p /home/engineers**

4. Change ownership of the new engineers’ shared folder to the engineers group.

   - Command to change ownership of engineer’s shared folder to engineer group:

**To set the ownership of the engineers shared folder to the engineer’s group where owners possessed, read, write, and execute permission over their files; group members possessed read and write permission over files of the group; and others possessed read permission, the following would occur:**

**sudo chgrp -R /engineers/**

**sudo chmod -R 764 /engineers/  owner = 7 = wrx; group = 6 = rw-; others = 4 = r-**

​

5. Add the SGID bit and the sticky bit to allow collaboration between engineers in this directory.

   - Command to set SGID and sticky bit to shared folder:

**If original permissions read -rwx rwx r--; a setting with the SGID bit and sticky bit to allow collaboration among engineers in this director would read: -rwxrwsrwt following a command of chmod +t /engineers.  If original settings are not defined the setting would read -rwxrwSrwT with a command of chmod +T /engineers or command chmod 2770.**

**### Step 4: Lynis Auditing**

​

1. Command to install Lynis:

**sudo apt -get install lynis**

2. Command to see documentation and instructions:

​

**show help; --help; -h**

3. Command to run an audit:

**​**

**sudo lynis audit system**

4. Provide a report from the Lynis output on what can be done to harden the system.

   - Screenshot of report output:

​

​

**### Bonus**

1. Command to install chkrootkit:

**apt install chkrootkit -y**

​

2. Command to see documentation and instructions:

**​ ?**

3. Command to run expert mode:

**​?**

4. Provide a report from the chrootkit output on what can be done to harden the system.

Create a secret user named sysd. Make sure this user doesn't have a home folder created.

* **Sudo su useradd sysd**

Give your secret user a password.

* **Sudo passwd sysd**
* **P@ssw0rd!**

Give your secret user a system UID < 1000.

* **Usermod -u 233 sysd**

Give your secret user the same GID

* **Usermod -g 233 sysd**

Give your secret user full sudo access without the need for a password.

* **sudo visudo sysd ALL=(ALL) NOPASSWD: ALL**

Test that sudo access works without your password

Your BASH commands go here

**#!/bin/bash  
sudo su useradd sysd  
sudo passwd sysd  
usermod -u 233 sysd  
usermod -g 233 sysd  
Sudo visudo sysd ALL=(ALL) NOPASSWD: ALL**