Cybersecurity

Module 5 Challenge Submission File

Archiving and Logging Data

Make a copy of this document to work in, and then for each step, add the solution command below the prompt. Save and submit this completed file as your Challenge deliverable.

Step 1: Create, Extract, Compress, and Manage tar Backup Archives

1. Command to **extract** the `TarDocs.tar` archive to the current directory:

```
[tar -xvf TarDocs.tar]
```

2. Command to **create** the Javaless_Doc.tar archive from the TarDocs/ directory, while excluding the `TarDocs/Documents/Java` directory:

```
[tar -cvf Javaless_Docs.tar -exclude-tag0under=Java ~/Projects/TarDocs]
```

3. Command to ensure `Java/` is not in the new `Javaless_Docs.tar` archive:

```
[tar -tf Javaless_Docs.tar | grep -rw Java *]
```

Bonus

4. Command to create an incremental archive called `logs_backup_tar.gz` with only changed files to `snapshot.file` for the `/var/log` directory:

```
[sudo tar -listed-incremental=snapshot.file -cvzf logs_backup.tar.gz
/var/log]
```

Critical Analysis Question

5. Why wouldn't you use the options `-x` and `-c` at the same time with `tar`?

-x is to extract files from an archive where -c is used to create a new archive, so using them together would cancel eachother out.

Step 2: Create, Manage, and Automate Cron Jobs

1. Cron job for backing up the `/var/log/auth.log` file:

```
[0 6 * * 3 tar -czf /auth_backup.tgz /var/log/auth.log]
```

Step 3: Write Basic Bash Scripts

1. Brace expansion command to create the four subdirectories:

```
[mkdir -p ~/backups/{freemem,diskuse,openlist,freedisk]
```

2. Paste your `system.sh` script edits:

```
#!/bin/bash
[# Free memory output to a free_mem.txt file
free -mh > awk 'NR==2{printf "Memory Usage: %s/%sMB (%.2f%%)\n",
$3,$2,$3*100/$2 }' > ~/backups/freemem/free_mem.txt

# Disk usage output to a disk_usage.txt file
df -h | awk '$NF=="/"{printf "Disk Usage: %d/%dGB (%s)\n", $3,$2,$5}' >
~/backups/diskuse/disk_usage.txt

# List open files to a open_list.txt file
lsof > ~/backups/openlist/open_list.txt
```

```
# Free disk space to a free_disk.txt file
df -h >> ~/backups/freedisk/free_disk.txt]
```

3. Command to make the `system.sh` script executable:

```
[chmod +x system.sh]
```

Optional

4. Commands to test the script and confirm its execution:

```
[Enter answer here]
```

Bonus

5. Command to copy `system` to system-wide cron directory:

```
[sudo cp system.sh /etc/cron.weekly]
```

Step 4. Manage Log File Sizes

1. Run `sudo nano /etc/logrotate.conf` to edit the `logrotate` configuration file.

   Configure a log rotation scheme that backs up authentication messages to the `/var/log/auth.log`.

   a. Add your config file edits:

```
[/var/log/auth.log {
  Weekly
  rotate 7
  Notifempty
  Delaycompress
  Missingok
  Endscript  }]
```

Bonus: Check for Policy and File Violations

1. Command to verify `auditd` is active:

```
[sudo systemctl status auditd]
```

2. Command to set number of retained logs and maximum log file size:

```
[sudo nano /etc/audit/audit.conf
```

Add the edits made to the configuration file:

```
[max_log_file = 35
Num_logs = 7]
```

3. Command using `auditd` to set rules for `/etc/shadow`, `/etc/passwd`, and `/var/log/auth.log`:

```
[sudo nano /etc/audit/rules.d/audit.rules]
```

Add the edits made to the `rules` file below:

```
[-w /etc/shadow -p wra -k hashpass_audit]
[-w /etc/passwd -p wra -k userpass_audit]
[-w /var/log/auth.log -p wra -k authlog_audit]
```

4. Command to restart `auditd`:

```
[sudo systemctl restart auditd]
```

5. Command to list all `auditd` rules:

```
[sudo auditctl -l]
```

6. Command to produce an audit report:

```
[sudo aureport -au]
```

7. Create a user with `sudo useradd attacker` and produce an audit report that lists account modifications:

```
[sudo aureport -m]
```

8. Command to use auditd to watch `/var/log/cron`:

```
[sudo auditctl -w /var/log/cron]
```

9. Command to verify `auditd` rules:

```
[sudo auditctl -l]
```

Bonus (Research Activity): Perform Various Log Filtering Techniques

1. Command to return `journalctl` messages with priorities from emergency to error:

```
[sudo jounalctl -b emerg..err]
```

2. Command to check the disk usage of the system journal unit since the most recent boot:

```
[sudo journalctl -b -u systemd-journald | less]
```

3. Command to remove all archived journal files except the most recent two:

```
[sudo journalctl –vacuum-files=2]
```

4. Command to filter all log messages with priority levels between zero and two, and save output to `/home/sysadmin/Priority_High.txt`:

```
[journalctl -p 0..2 > /home/student/Priority_High.txt]
```

5. Command to automate the last command in a daily cron job. Add the edits made to the crontab file below:

```
@daily journalctl -p 0..2 > /home/student/Priority_High.txt]
```