Enhancing System Functionality

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

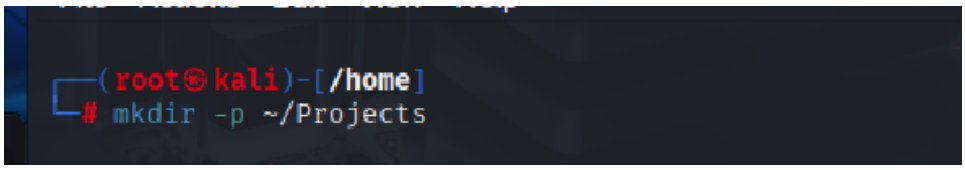
Prior to beginning the project you will need to download `TarDocs.tar` file

Goal: Use `tar` to extract archives and create backups while excluding specific directories.

1.1: Create the Project Directory and Move Tar File

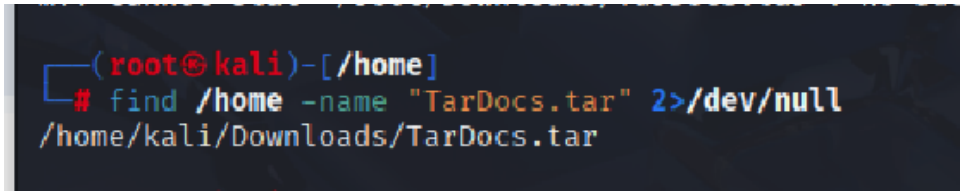
bash

```
mkdir -p ~/Projects
```

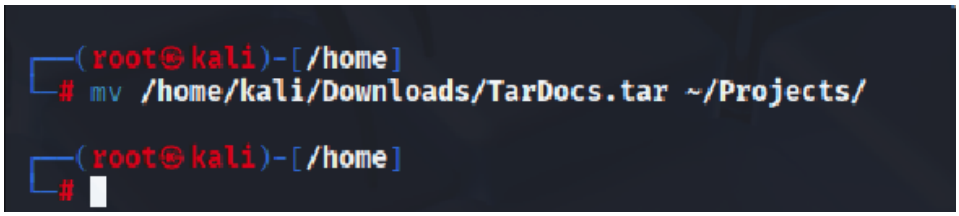


Locate `TarDocs.tar` file and ensure it is located in the Downloads folder

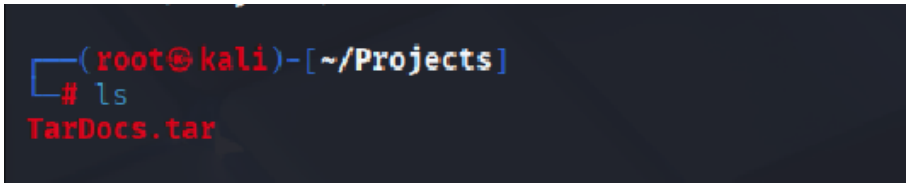
```
find /home -name "TarDocs.tar" 2>/dev/null
```




```
mv ~/Downloads/TarDocs.tar ~/Projects/
```




```
cd ~/Projects
```



Why: Organizing work in a dedicated `Projects` folder keeps your system clean and follows best practices for file management.

1.2: Extract the Archive

bash

```
tar -xvf TarDocs.tar
```

```
┌──(root㉿kali)-[~/Projects]
└─# tar -xvf TarDocs.tar
TarDocs/
TarDocs/Movies/
TarDocs/Movies/ZOE_0004.mp4
TarDocs/Movies/ZO_0001.mp4
TarDocs/Movies/ZOE_0003.mp4
TarDocs/Movies/ZOE_0002.mp4
```

Why: Extracts the contents of `TarDocs.tar` for inspection and modification.

1.3: Verify Subdirectory Exists

bash

```
ls -l ~/Projects/TarDocs/Documents/
```

```
┌──(root㉿kali)-[~/Projects]
└─# ls -l ~/Projects/TarDocs/Documents/
total 1512
-rwxr-xr-x 1 kali 1008 1365983 Aug 10  2012 c++interviewquestions.pdf
drwxr-xr-x 2 kali 1008    4096 Jan 12  2019 Design-Patterns
drwxr-xr-x 2 kali 1008    4096 Jan 12  2019 Google-Maps-Hacks
-rwxr-xr-x 1 kali 1008  161823 Oct  3  2015 IntelliJIDEA_ReferenceCard.pdf
drwxr-xr-x 5 kali 1008    4096 Jan 13  2019 Java
drwxr-xr-x 2 kali 1008    4096 Jan 12  2019 Music-Sheets
```

Why: Confirms that the `Java` subdirectory exists and is available for exclusion in the next step.

1.4: Create a Tar Archive Excluding the `Java` Folder

bash

```
tar --exclude='Java' -cvf Javaless_Docs.tar TarDocs/Documents/
```

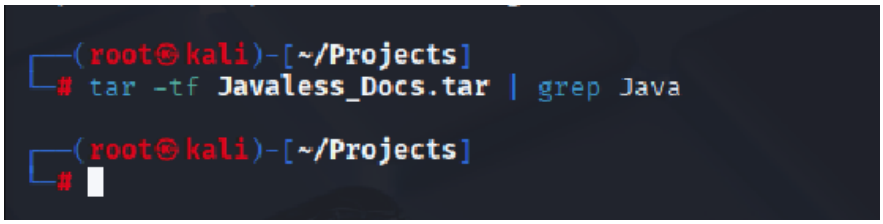
```
┌──(root㉿kali)-[~/Projects]
└─# tar --exclude='Java' -cvf Javaless_Docs.tar TarDocs/Documents/
TarDocs/Documents/
TarDocs/Documents/IntelliJIDEA_ReferenceCard.pdf
TarDocs/Documents/Google-Maps-Hacks/
TarDocs/Documents/Google-Maps-Hacks/googlemapshks-CHP-2.PDF
TarDocs/Documents/Google-Maps-Hacks/googlemapshks-CHP-6.PDF
TarDocs/Documents/Google-Maps-Hacks/googlemapshks-CHP-3.PDF
TarDocs/Documents/Google-Maps-Hacks/googlemapshks-CHP-5.PDF
TarDocs/Documents/Google-Maps-Hacks/googlemapshks-CHP-7.PDF
TarDocs/Documents/Google-Maps-Hacks/googlemapshks-CHP-1.PDF
TarDocs/Documents/Google-Maps-Hacks/googlemapshks-CHP-4.PDF
TarDocs/Documents/Design-Patterns/
TarDocs/Documents/Design-Patterns/DesignPatterns.pdf
TarDocs/Documents/Design-Patterns/Head_First_Design_Patterns__2008_.pdf
TarDocs/Documents/c++interviewquestions.pdf
TarDocs/Documents/Music-Sheets/
TarDocs/Documents/Music-Sheets/Stairway-to-heaven-guitar.pdf
TarDocs/Documents/Music-Sheets/Stairway-to-heaven-piano-guitar-A-minor.pdf
TarDocs/Documents/Music-Sheets/Stairway-to-heaven-bass-tab.pdf
TarDocs/Documents/Music-Sheets/Thumbs.db
```

Why: Backing up only relevant directories improves efficiency and conserves storage. `--exclude='Java'` ensures that this subdirectory is omitted.

1.5: Verify Java Folder is Not in Archive

bash

```
tar -tf Javaless_Docs.tar | grep Java
```

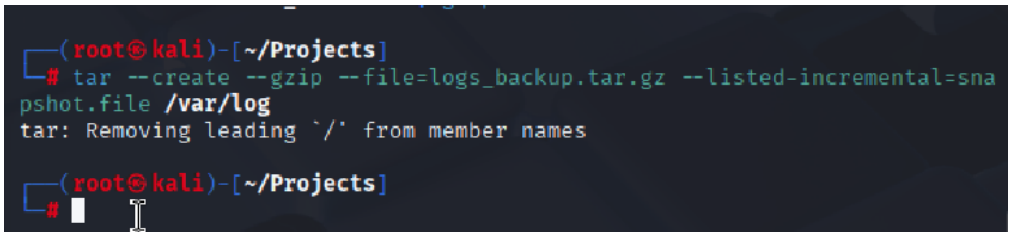


Why: Double-checks that `Java` was successfully excluded from the archive. If it works correctly and the `Java` directory was excluded as intended, the command should return **no output**.

(Optional) 1.6: Create an Incremental Backup Archive

bash

```
tar --create --gzip --file=logs_backup.tar.gz
--listed-incremental=snapshot.file /var/log
```



Why: Only changed files are archived, which speeds up backups and reduces storage usage—ideal for frequent backup schedules.

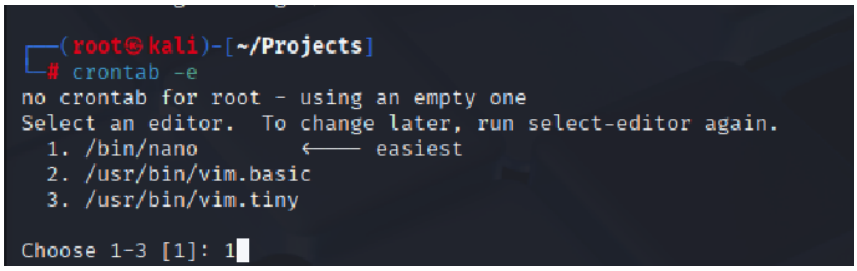
Step 2: Create, Manage, and Automate Cron Jobs

Goal: Schedule automated backups to protect critical system logs.

2.1: Open Crontab Editor

bash

```
crontab -e
```



Why: Opens the user's cron table where scheduled tasks are defined. Since this is the first time `crontab` is being used. You will be provided a selection to choose from. Nano is the easiest and most beginner-friendly editor. This will then open an empty crontab file.

```
GNU nano 8.3          /tmp/crontab.Wo6wTZ/crontab
# 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
```

2.2: Add Cron Job to Backup `/var/log/auth.log`

cron

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

```
#
# m h  dom mon dow   command
#
#0 6 * * 3 tar -cvf /auth_backup.tgz /var/log/auth.log
```

```
┌──(root㉿kali)-[~/Projects]
└─# crontab -e
no crontab for root - using an empty one
Select an editor.  To change later, run select-editor again.
  1. /bin/nano        <── easiest
  2. /usr/bin/vim.basic
  3. /usr/bin/vim.tiny

Choose 1-3 [1]: 1
crontab: installing new crontab
```

Why: Schedules a compressed archive backup every **Wednesday at 6 AM**. Protecting auth logs is crucial after incidents like ransomware.

Tip: Use crontab.guru to experiment and verify your cron syntax.

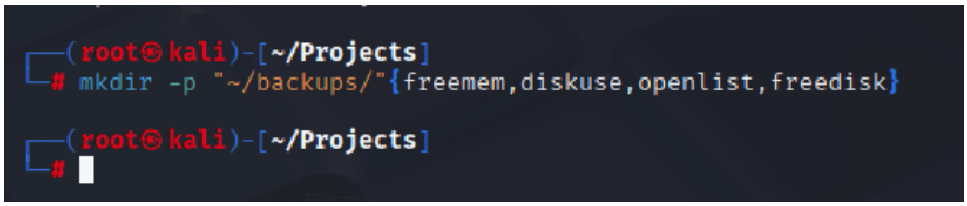
Step 3: Write Basic Bash Scripts

Goal: Automate system resource monitoring with a Bash script.

3.1: Create Backup Subdirectories

bash

```
mkdir -p "~/backups/"{freemem,diskuse,openlist,freedisk}
```

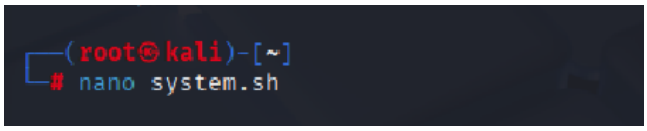


Why: Keeps each output type organized in its own folder for easier tracking and automation.

3.2: Create and Edit the Bash Script

bash

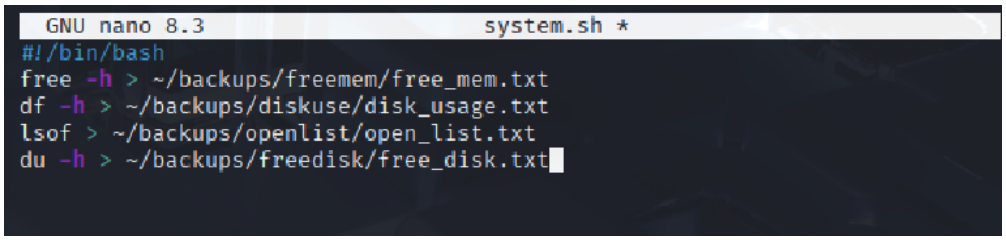
```
cd ~/
nano system.sh
```



Script Content:

bash

```
#!/bin/bash
free -h > ~/backups/freemem/free_mem.txt
df -h > ~/backups/diskuse/disk_usage.txt
lsof > ~/backups/openlist/open_list.txt
du -h > ~/backups/freedisk/free_disk.txt
```

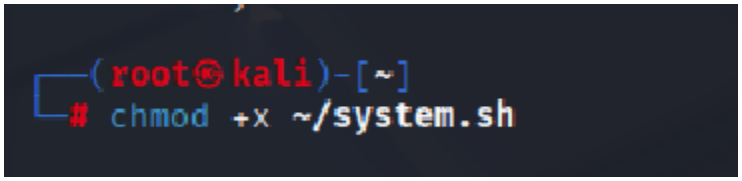


Why: This script captures memory usage, disk usage, open file handles, and disk space statistics — key metrics for system health monitoring.

3.3: Make Script Executable

bash

```
chmod +x ~/system.sh
```



Why: Marks the script as executable so it can be run like a program.

3.4: Run the Script

bash

```
sudo ./system.sh
```

```
┌──(root㉿kali)-[~]
└─# sudo ./system.sh
./system.sh: line 2: /root/backups/freemem/free_mem.txt: No such file or d
irectory
./system.sh: line 3: /root/backups/diskuse/disk_usage.txt: No such file or
 directory
./system.sh: line 4: /root/backups/openlist/open_list.txt: No such file or
 directory
./system.sh: line 5: /root/backups/freedisk/free_disk.txt: No such file or
 directory
```

Why: Executes the script with elevated permissions (required for `lsof` and accessing system data).

(Optional) 3.5: Automate the Script Weekly

bash

```
ln -s ~/system.sh /etc/cron.weekly/system-monitor
```

```
┌──(root㉿kali)-[~]
└─# ln -s ~/system.sh /etc/cron.weekly/system-monitor
```

Why: Ensures your resource logs are regularly updated, even if you forget to run the script manually.

Step 4 (Optional): Monitor Policy and File Violations with `auditd`

Goal: Use `auditd` to monitor sensitive file access and user account modifications.

4.1: Check if `auditd` is Active

bash

```
systemctl status auditd
```

```
┌──(root㉿kali)-[~]
└─# systemctl status audit
Unit audit.service could not be found.
```

Why: Confirms the audit daemon is running before setting up rules. `Unit audit.sevice could not be found` means that the `auditd` service is **not installed** on Kali Linux system.

To fix this, install `auditd` with:

Bash

```
sudo apt update
```

```
┌──(root㉿kali)-[~]
└─# apt update
Hit:1 http://http.kali.org/kali kali-rolling InRelease
1298 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

```
sudo apt install auditd -y
```

```
┌──(root㉿kali)-[~]
└─# apt install auditd -y
Upgrading:
  libaudit1

Installing:
  auditd

Installing dependencies:
  libauparse0t64

Suggested packages:
  audispd-plugins

Summary:
  Upgrading: 1, Installing: 2, Removing: 0, Not Upgrading: 1297
  Download size: 326 kB
  Space needed: 1,384 kB / 10.7 GB available
```

Then start and enable the service:

bash

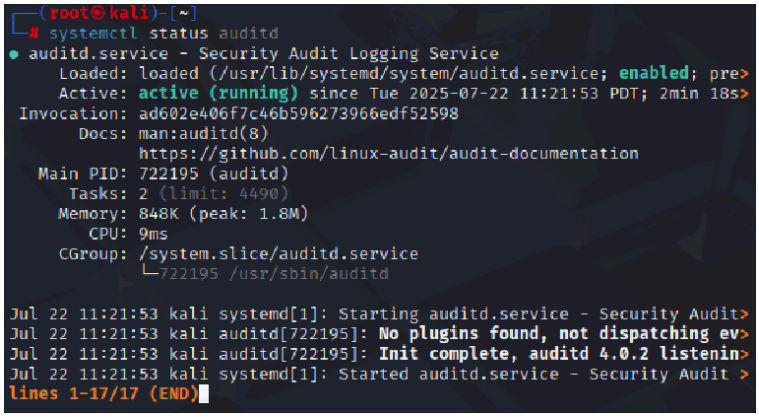
```
sudo systemctl start auditd
sudo systemctl enable auditd
```

```
┌──(root㉿kali)-[~]
└─# systemctl start auditd

┌──(root㉿kali)-[~]
└─# systemctl enable auditd
Synchronizing state of auditd.service with SysV service script with /usr/l
ib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable auditd
Created symlink '/etc/systemd/system/multi-user.target.wants/auditd.servic
e' → '/usr/lib/systemd/system/auditd.service'.
```

Bash (second attempt)

```
systemctl status auditd
```



Why: Finally confirmed the audit daemon is running before setting up rules.

4.2: Edit `auditd.conf` File

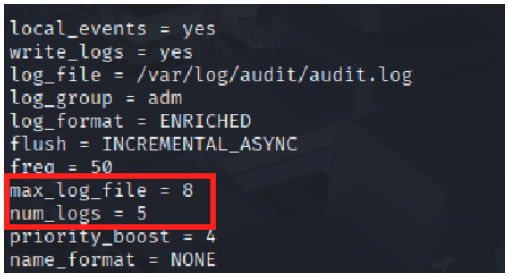
bash

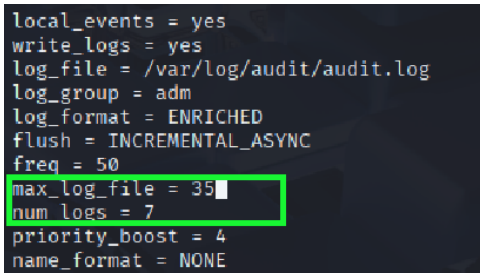
```
nano /etc/audit/auditd.conf
```

Edit the Following:

ini

```
num_logs = 7
max_log_file = 35
```





Why: Retains 7 logs at 35MB each — a reasonable setting for log rotation in most environments.

4.3: Add Audit Rules

bash

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

Add These Rules:

bash

```
-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
```

```
## Watch key files for write/read/attribute changes and tag logs
## with descriptive keys for filtering
-w /etc/passwd -p wra -k hashpass_audit
-w /etc/passwd -p wra -k userpass_audit
-w /var/log/auth.log -p wra -k authlog_audit
```

Why: Watches key files for write/read/attribute changes and tags logs with descriptive keys for filtering.

4.4: Restart and Check Rules

bash

```
sudo systemctl restart auditd
```

```
┌──(root㉿kali)-[~]
└─# systemctl restart auditd
```

```
sudo auditctl -l
```

```
┌──(root㉿kali)-[~]
└─# auditctl -l
-w /etc/passwd -p rwa -k hashpass_audit
-w /etc/passwd -p rwa -k userpass_audit
-w /var/log/auth.log -p rwa -k authlog_audit
```

Why: Applies changes and verifies that audit rules are active.

4.5: Generate and View Audit Reports

bash

```
sudo aureport -au       # Authentication events
```

```
┌──(root㉿kali)-[~]
└─# aureport -au

Authentication Report
============================================
# date time acct host term exe success event
============================================
<no events of interest were found>
```

```
sudo aureport -m        # Modification events
```

```
┌──(root㉿kali)-[~]
└─# aureport -m

Account Modifications Report
============================================
# date time auid addr term exe acct success event
============================================
<no events of interest were found>
```

4.6: Simulate User Creation

bash

```
sudo useradd attacker
```

```
┌──(root㉿kali)-[~]
└─# useradd attacker
```

```
sudo aureport -modification
```

```
┌──(root㉿kali)-[~]
└─# aureport --mods

Account Modifications Report
═══════════════════════════════════════════════════
# date time auid addr term exe acct success event
═══════════════════════════════════════════════════
1. 07/22/2025 11:47:21 1000 kali pts/0 /usr/sbin/useradd attacker yes 90
2. 07/22/2025 11:47:21 1000 kali pts/0 /usr/sbin/useradd attacker yes 94
```

Why: Creates a fake user to test if audit logging works as intended.

4.7: Monitor Cron Directory for Changes

bash

```
sudo auditctl -w /var/log/cron -p wra -k cron_watch
```

```
┌──(root㉿kali)-[~]
└─# auditctl -w /var/log/cron -p wra -k cron_watch
Old style watch rules are slower
```

Why: Keeps an eye on cron activity, which attackers often tamper with to maintain persistence.'old style watch rules are slower' is just a warning, not an error. It means you're using the older syntax for audit rules (like -w /path -p rwxa -k key), which still works, but newer rule formats are preferred for better performance.

4.8: Verify the Rule is Loaded

Bash

```
sudo auditctl -l | grep cron_watch
```

```
┌──(root㉿kali)-[~]
└─# auditctl -l | grep cron_watch
-w /var/log/cron -p rwa -k cron_watch
```

5.0: Trigger an Event

Edit the crontab to create a detectable change:

bash

```
crontab -e
```

Just add a harmless comment like:

```
shell
```

```
# test audit trigger
```

```
┌──(root㉿kali)-[~]
└─# crontab -e
crontab: installing new crontab
```

- Save and exit.

This should **modify** `/var/log/cron`, triggering the audit rule.

5.1. Search Audit Logs for the Key

Now check if the event was captured using the key you assigned:
bash
```
sudo ausearch -k cron_watch
```

If working properly, you'll see log entries showing access or modification events related to `/var/log/cron`.

Optional: Generate a Report

You can generate a report filtered by key:
bash
```
sudo aureport --file | grep cron
```

```
┌──(root㉿kali)-[~]
└─# ausearch -k cron_watch
----
time->Tue Jul 22 11:51:47 2025
type=PROCTITLE msg=audit(1753210307.841:101): proctitle=617564697463746C00
2D77002F7661722F6C6F672F63726F6E002D7000777261002D6B0063726F6E5F7761746368
type=PATH msg=audit(1753210307.841:101): item=0 name="/var/log/" inode=524
448 dev=fe:03 mode=040755 ouid=0 ogid=0 rdev=00:00 nametype=PARENT cap_fp=
0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=0
type=CWD msg=audit(1753210307.841:101): cwd="/root"
type=SOCKADDR msg=audit(1753210307.841:101): saddr=100000000000000000000000
0
type=SYSCALL msg=audit(1753210307.841:101): arch=c00000b7 syscall=206 succ
ess=yes exit=1080 a0=4 a1=ffffd8f351f0 a2=438 a3=0 items=1 ppid=364787 pid
=738064 auid=1000 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0
tty=pts0 ses=2 comm="auditctl" exe="/usr/sbin/auditctl" subj=unconfined ke
y=(null)
type=CONFIG_CHANGE msg=audit(1753210307.841:101): auid=1000 ses=2 subj=unc
onfined op=add_rule key="cron_watch" list=4 res=1
----
time->Tue Jul 22 11:58:36 2025
type=PROCTITLE msg=audit(1753210716.327:123): proctitle=617564697463746C00
2D6100657869742C616C77617973002D460070617468002F7661722F6C6F672F63726F6E00
2D46007065726D00777261002D46006B65793D63726F6E5F7761746368
type=SOCKADDR msg=audit(1753210716.327:123): saddr=100000000000000000000000
0
type=SYSCALL msg=audit(1753210716.327:123): arch=c00000b7 syscall=206 succ
ess=yes exit=1080 a0=4 a1=fffff332c2c0 a2=438 a3=0 items=0 ppid=364787 pid
=741736 auid=1000 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0
tty=pts0 ses=2 comm="auditctl" exe="/usr/sbin/auditctl" subj=unconfined ke
y=(null)
type=CONFIG_CHANGE msg=audit(1753210716.327:123): auid=1000 ses=2 subj=unc
onfined op=add_rule key="cron_watch" list=4 res=0
```

Or see the most recent relevant logs:
bash

```
sudo ausearch -k cron_watch --start recent
```

```
┌──(root㉿kali)-[~]
└─# ausearch -k cron_watch --start recent
<no matches>
```

Summary

If `ausearch -k cron_watch` shows **any results**, then the audit rule is **successfully capturing events**.