cron and Scheduled Jobs

Cybersecurity

5.2 Archiving and Logging Data Day 2

© 2021 Trilogy Education Services, a 2U, Inc. brand. All Rights Reserved.

Class Objectives

By the end of today's class, you will be able to:

- Schedule regular jobs for individual users with crontab.
- Create simple scripts for maintenance and security tasks.
- Use `cron` to automate the execution of security scripts to perform maintenance on a regular basis.

Scheduling Backups

Today, we'll learn how to create scripts and use a tool called `cron` to automate many of the tasks performed in the last class.

Archiving data

to ensure it remains available in the case of a natural disaster or cyber attack.

Scheduling backups

to ensure they're up to date and made at the appropriate frequency.

Monitoring log files

to prevent and detect suspicious activity and keep systems running efficiently.

Automating

Automating a series of tasks takes two forms:

01 Scripts

Files that contain multiple commands.

The commands are executed by calling the script name.

02 Scheduled jobs

Run commands or scripts at specific, designated times.

Using Scripts in a Professional Context

Sysadmins use scripts and scheduled jobs in their workflow.

Fix an issue.

Example:
Make a list of all users with old passwords, and force these users to update their credentials.

Create a script to automatically solve a problem.

Example:
Create the script `find_stale_users.sh` to fix the issue.

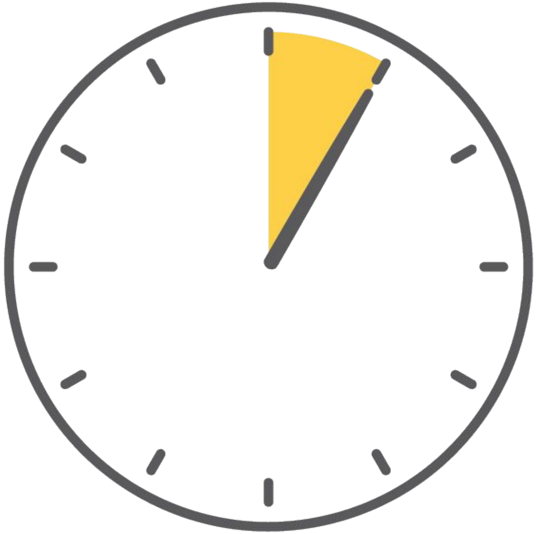
Run the script and email the results regularly, using `cron`.

Example:
Schedule `find_stale_users.sh` to run every Saturday at noon.

Overview of cron

Consider the Following Scenario:

Before leaving work, you spend **10 minutes** deleting your cache and your trash bin, and backing up your documents folder.



You also spend **one hour** per day installing software updates.

Introducing `cron`

Rather than spending time manually completing these repeatable tasks, we can automate them using `cron` jobs.

A **cron job** is a script or command designated to run at periodic, predetermined intervals.

Introducing `cron`

User-level

User-level `cron` jobs can automate the process of deleting cache, emptying trash, and backing up documents.

System-level

System-level `cron` jobs can automate daily software updates.

Cron

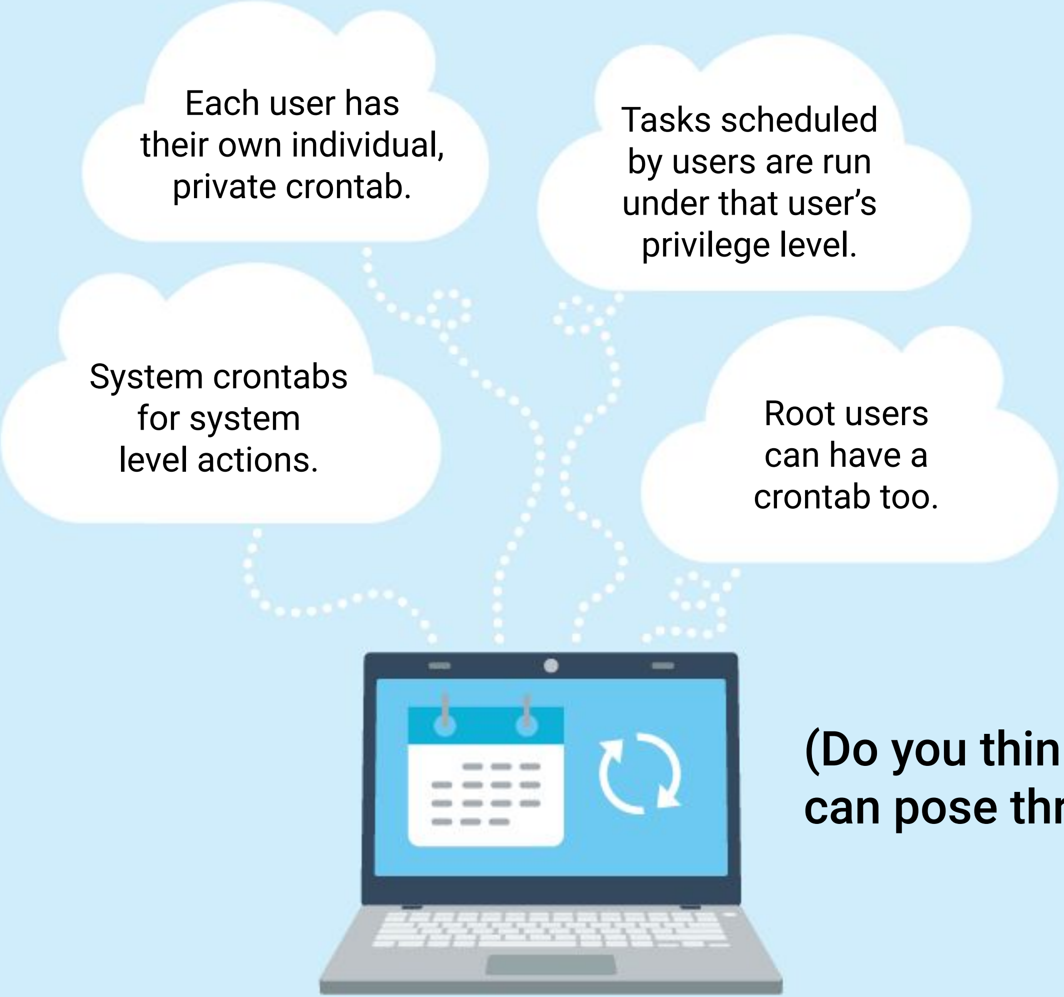
The *cron* in cron job refers to a system daemon that keeps track of when to run scheduled tasks.

A **daemon** is a computer program that runs as a background process, rather than being directly controlled by an interactive user.

`cron` is a robust task scheduler daemon that allows users to schedule repetitive tasks to run on a regular basis.

Daemons are often started at boot up. While other daemons respond to network requests and hardware activity, cron is initiated at designated time intervals.

Tasks are stored and scheduled in a file called a `crontab` (*cron table*).



Each user has their own individual, private crontab.

Tasks scheduled by users are run under that user's privilege level.

System crontabs for system level actions.

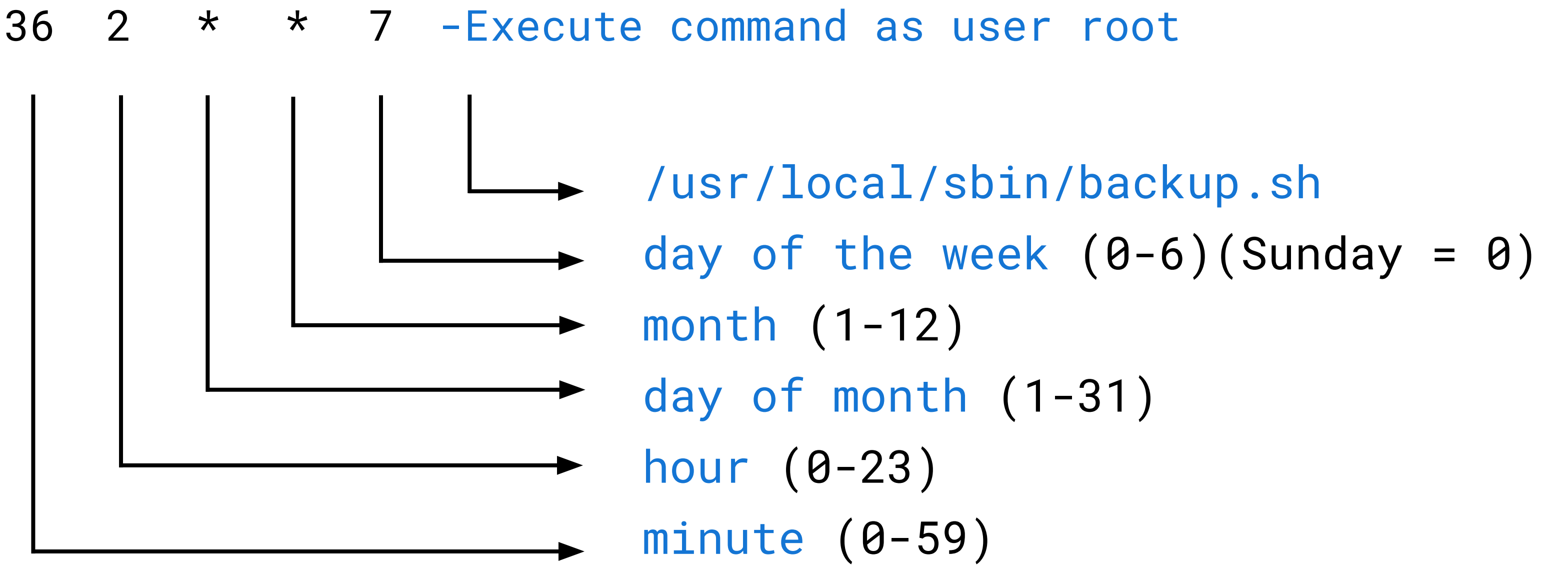
Root users can have a crontab too.

(Do you think this can pose threats?)

Let’s look at crontab on the command line. While they may seem intimidating, crontab rules can be learned relatively easily with some practice.

cron Syntax

```
# Execute backup .sh script every Sunday at 2:36 a.m.
36 2 * * 7 root/usr/local/sbin/backup.sh
```



36 2 * * 7 -Execute command as user root

/usr/local/sbin/backup.sh

day of the week (0-6)(Sunday = 0)

month (1-12)

day of month (1-31)

hour (0-23)

minute (0-59)

cron Syntax Walkthrough

In the upcoming demo, we will focus on the following:

01

General `cron` command line syntax.

02

Editing a crontab with `crontab -e`.

03

Listing the contents of the crontab using `crontab -l`.

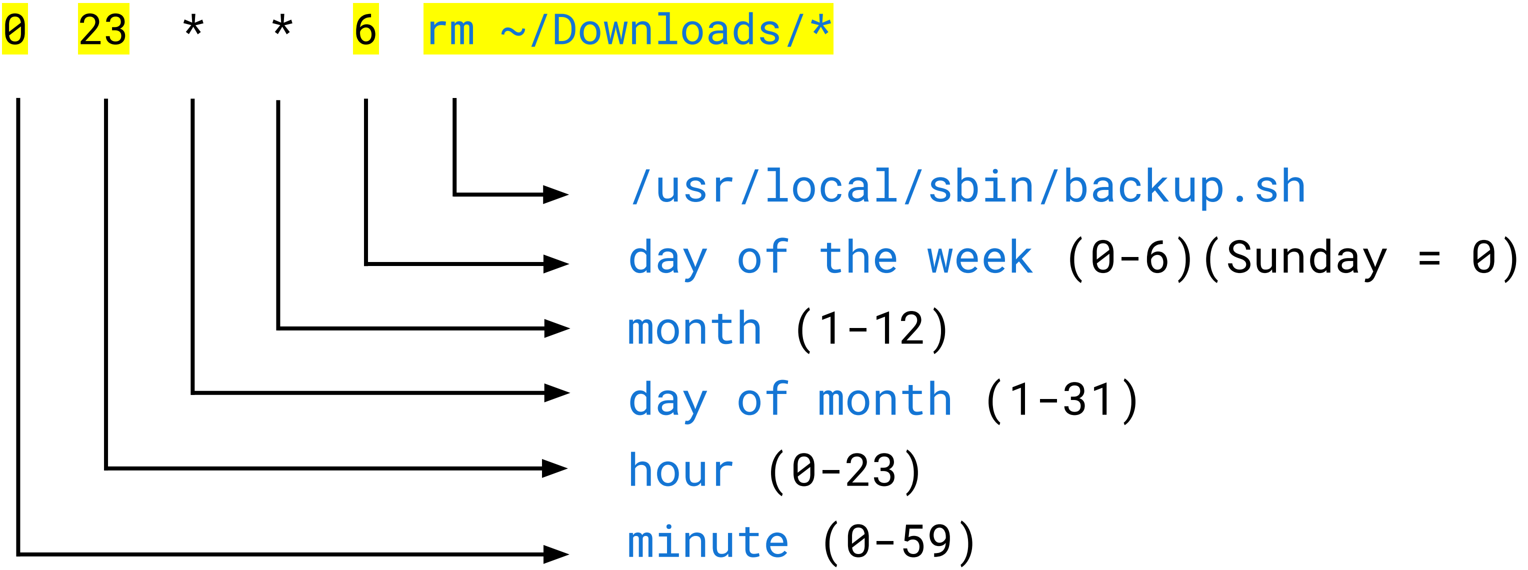
(Note: this is the lower case letter L, not the number 1.)

Instructor Demonstration

cron syntax

cron jobs

```
# Removes all files in the ~/Downloads directory at 11pm, every Saturday, every month.
0 23 * * 6 root/usr/local/rm ~/Downloads/*
```



```
0  23  *  *  6  rm ~/Downloads/*
```

- /usr/local/sbin/backup.sh
- day of the week (0-6)(Sunday = 0)
- month (1-12)
- day of month (1-31)
- hour (0-23)
- minute (0-59)

Instructor Demonstration

`Crontab generator, crontab  -e`

crontab

Cron jobs run under the same permissions as the user who creates them. A cron job created by user `root` will run with `root` privileges.

This introduces security risks. Anyone capable of privilege escalation can add a malicious cron job to the root crontab.

- We'll revisit the risks of using `cron` with root privileges later in the unit.
- Best practice is to avoid using the root crontab.

Inspecting the root crontab for unauthorized or malicious entries is a critical step in ensuring the integrity of any system that uses it. **Let's explore.**

Instructor Demonstration

```
crontab   -1
```

Today’s Activity Scenario

In today’s activities, we’ll act the role of a junior administrator at the company Rezifp Pharma Inc.

- There has been a wave of recent ransomware attacks. You will be responsible for using `cron` to automate tasks that backup E-Prescription Treatment database.
- Rezifp maintains a large number of files related to patients, doctors, and treatments.
- Administrators at various clinics often create files that contain Personal Identifiable Information or (PII) such as email addresses, passwords, biometric records, etc.

Activity: Simple cron Jobs

In this activity you will create a simple cron job to ensure the integrity and availability of data and help ensure compliance with HIPAA regulations.

Suggested Time:

25 Minutes

Time’s Up! Let’s Review.

Let’s Review

In the previous section, we learned:

Crontabs come in two varieties:

- **User-level:** runs for a specific user under their privilege level.
- **System-level:** runs for the system as a whole under root privileges.

User-level crontabs are often used for “personal” tasks, such as organizing files.

The general syntax for a crontab:

```
minute hour day-of-month month day-of-week command
```

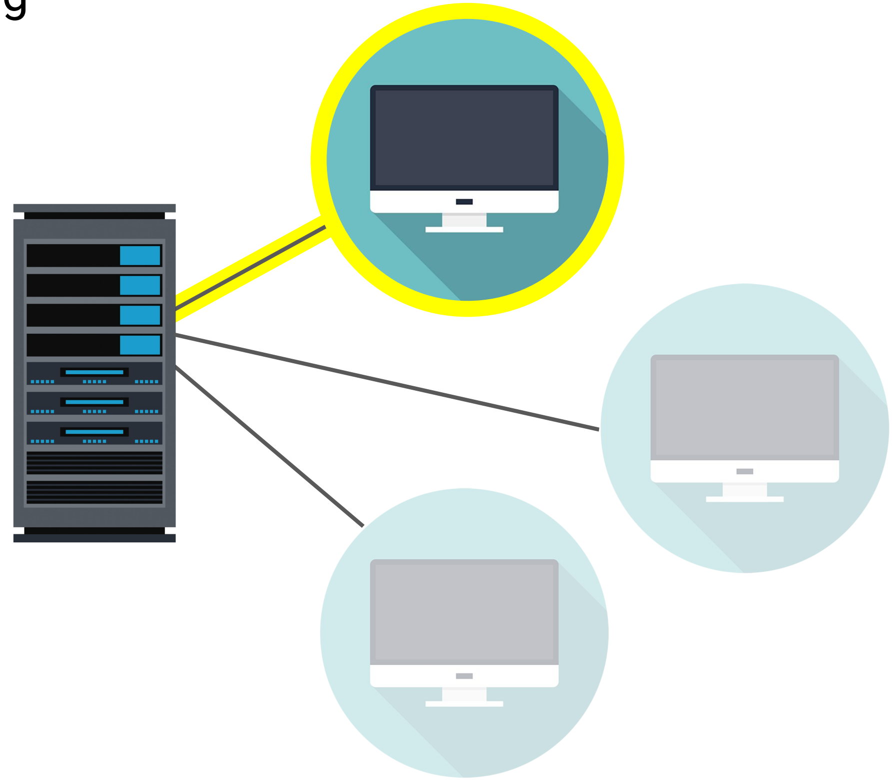
Tools like crontab.guru can verify the syntax of cron jobs before they run.

Scripting

Why Scripts?

`cron` is useful for single tasks, like backing up a single user's directory, but not for backing up several users' directories.

- Scripts allow us to complete complex tasks, like creating backups or cleaning up multiple directories, by executing a single script.
- This results in cleaner crontabs with fewer lines of code, and allows us to schedule complex jobs that can't be expressed by a single command.



Next, we’ll use `cron` to run a script that executes several commands simultaneously that run once per day.

Instructor Demonstration

Writing a Script

Demo Summary

In the previous guided tour, we did the following:

01

Saved several commands into a single script called `~/customs_scripts/cleanup_downloads`.

02

Verified that this script behaves as expected.

03

Updated the crontab to run the script instead of running three separate commands.

Activity: Scriptings

In this activity, you will assume the role of a junior administrator tasked with creating a shell script that keeps the system clean, up-to-date, and ensures backups remain current and uncorrupted.

Suggested Time:

30 Minutes

Time’s Up! Let’s Review.

Questions?

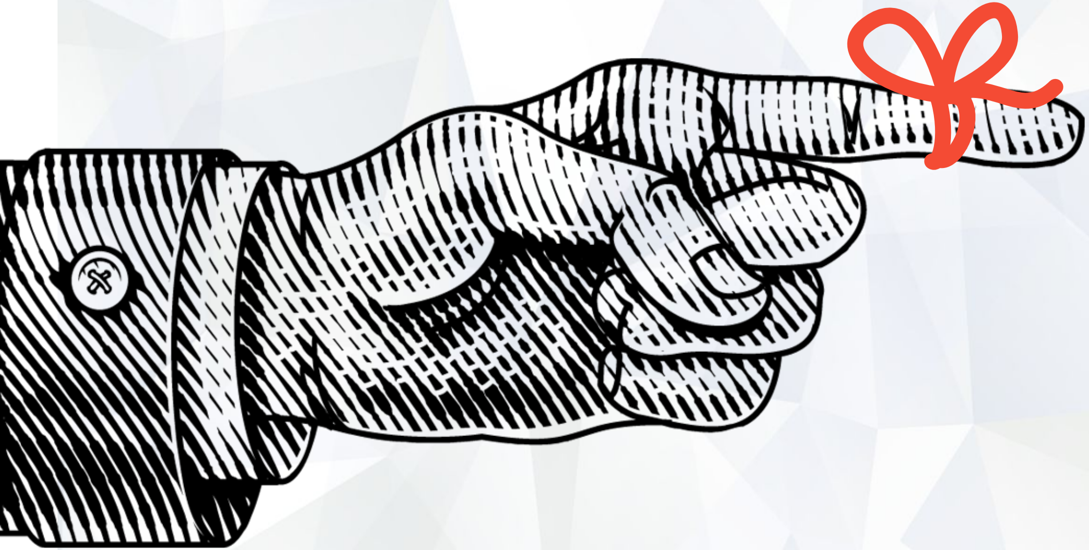
Any Questions?

Countdown timer

15:00

(with alarm)

Scheduling Backups, Cleanups, and Security Checks



Remember,

User `cron` jobs are not very useful for system-level maintenance.

- User-level crontabs belong to non-root users, so they are unable to access most system files.
- Now we will use **system crontabs**, which runs with root privileges, to run system-wide cron tasks.

System-Wide Cron Directories

System-wide cron directories are located at:

- **`/etc/cron/d`**
- **`/etc/cron.daily`**
- **`/etc/cron.weekly`**
- **`/etc/cron.monthly`**

Each directory contains several scripts that run at the dedicated time intervals.

- For example, scripts placed in **`/etc/cron.weekly`** will run once per week.

Next, we’ll examine the `etc/cron` daily, weekly, and monthly directories.

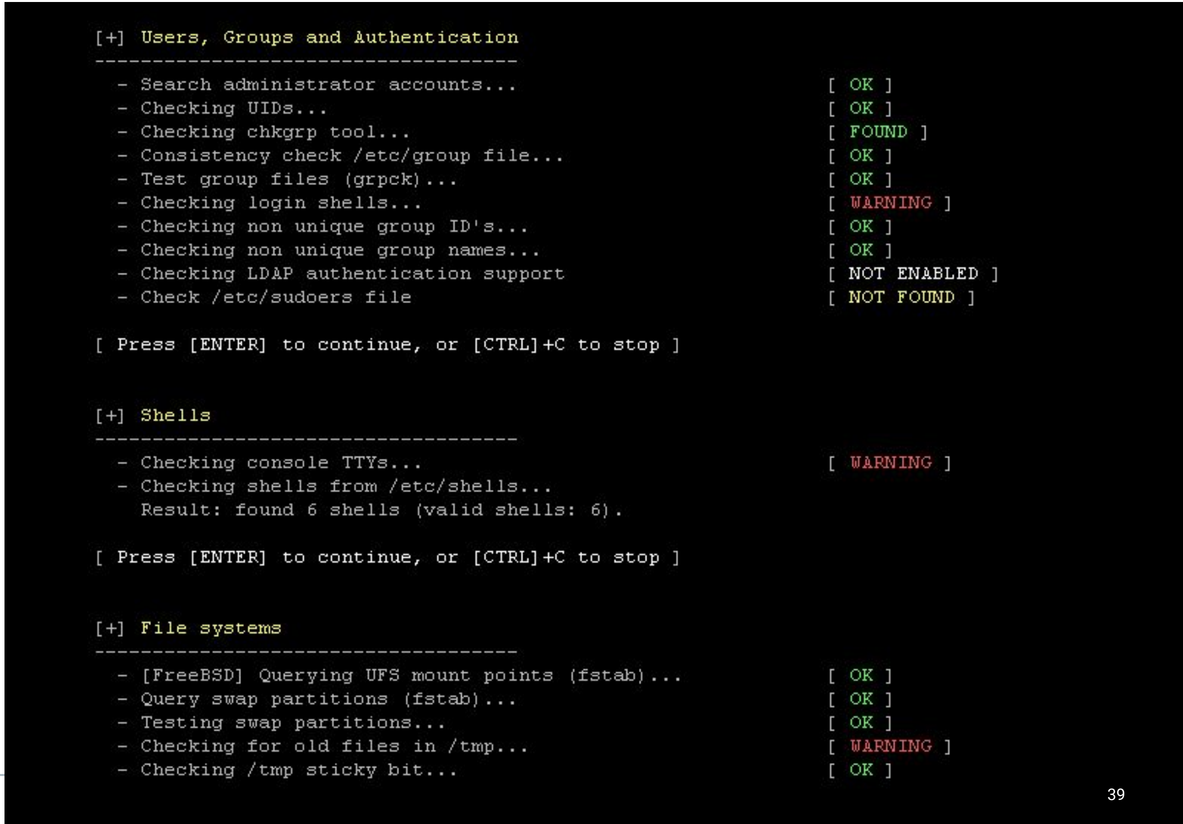
Instructor Demonstration

System-Wide cron Directories

Lynis Scanner

Lynis is a security scanner used to check a machine for vulnerabilities.

- Generates and saves reports of its findings for administrators to review.
- Offers numerous scan types.
- Today, we’ll experiment with a few different ones...



```
[+] Users, Groups and Authentication
------------------------------------
  - Search administrator accounts...                              [ OK ]
  - Checking UIDs...                                              [ OK ]
  - Checking chkgrp tool...                                       [ FOUND ]
  - Consistency check /etc/group file...                          [ OK ]
  - Test group files (grpck)...                                   [ OK ]
  - Checking login shells...                                      [ WARNING ]
  - Checking non unique group ID's...                             [ OK ]
  - Checking non unique group names...                            [ OK ]
  - Checking LDAP authentication support                          [ NOT ENABLED ]
  - Check /etc/sudoers file                                       [ NOT FOUND ]

[ Press [ENTER] to continue, or [CTRL]+C to stop ]


[+] Shells
------------------------------------
  - Checking console TTYs...                                      [ WARNING ]
  - Checking shells from /etc/shells...
    Result: found 6 shells (valid shells: 6).

[ Press [ENTER] to continue, or [CTRL]+C to stop ]


[+] File systems
------------------------------------
  - [FreeBSD] Querying UFS mount points (fstab)...                [ OK ]
  - Query swap partitions (fstab)...                              [ OK ]
  - Testing swap partitions...                                    [ OK ]
  - Checking for old files in /tmp...                             [ WARNING ]
  - Checking /tmp sticky bit...                                   [ OK ]
```

https://cisofy.com

Instructor Demonstration

Lynis Scanner

Activity: Scheduling Backups and Cleanups

In this activity, you will assume the role of a junior administrator tasked with creating system-wide cron jobs to schedule your previously made scripts.

Activity file sent via Instructor.

Suggested Time:

25 Minutes

Time’s Up! Let’s Review.

Questions?

Any Questions?

The End