# DFRWS USA 2020

## Compromised Web Server!

You have been called to analyze a compromised Linux web server, find how the threat actor gained access to the system, what modifications were applied, and what persistent methods have been applied to the system (e.g. backdoors, users, sessions, etc).

**Deliverables:**

1. How the threat actor gained access to the system?
2. What privileges were obtained and how?
3. What modifications were applied to the system?
4. What persistent mechanisms are being used?
5. Could this system be cleaned/recovered?
6. Notes and recommendations

**Outcomes:**

At the end of this lab, you will have the required skills to deal with a compromised Linux system, were you will be capable of doing:

1. Listing the volumes and mounting a forensic case image
2. Searching through the FHS
3. Search in log files
4. Use TSK tools to list info of the image and deal with EXT4 fs
5. Use debugfs, EXT4 journal and ext4magic to recover deleted files
6. Generate and filter a super timeline

**Note(1):**

1. PLEASE ASK QUESTIONS. We will go over this case very slowly to cover not only the case but also the commands, tools used, etc.
2. Please use tables and screenshots to represent your results if needed. Like I usually say, "**Screenshot or it didn't happen!**".

## Notes and Settings:

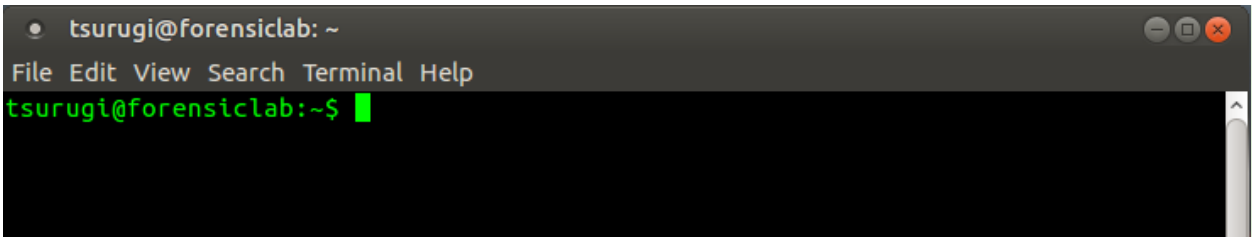
- I recommend you download and setup **Tsurugi Linux** distribution for this workshop, but if you prefer to use any other distro or know how to install the tools needed, then that's up to you. **Tsurugi Linux** could be downloaded from here: https://tsurugi-linux.org/downloads.php
- If you do not prefer to setup **Tsurugi Linux**, then you can use the OVA file which is for Virtual Machines. All you need to do, is download the file and then import it or open it using your hypervisor.
- The default username and password for **Tsurugi Linux** is: tsurugi
- Copy the forensic image for this workshop to your newly prepared **Tsurugi Linux**.
- Setup the language and the display as preferred.
- This workshop does not assume you have experience in Linux, but if you do, then that's an advantage.
- Some of the instructions that will be covered are for educational purposes, there are so many other ways to solve this case, but we are showing how to do that using very simple commands and trying our best not to make things complicated.
- Anything in this document referenced in red is a **command**, while those in blue are referencing either a **file** or **directory**.

## Workshop Files:

- All files for this workshop could be downloaded from here: https://mega.nz/folder/6r5AAI5T#3nAT3Vh371IpH57KOSuZ0w

## Task #0: Environment Preparation

From the top panel click on the red icon with a cross inside. Or from the menu **Applications→ System Tools → MATE Terminal**




Create a directory named "**cases**"

$ mkdir cases

Then change your working directory to the newly created directory, as follows:

$ cd cases

This will be where we will store all our cases, but for now let us create another directory for our web case as following:

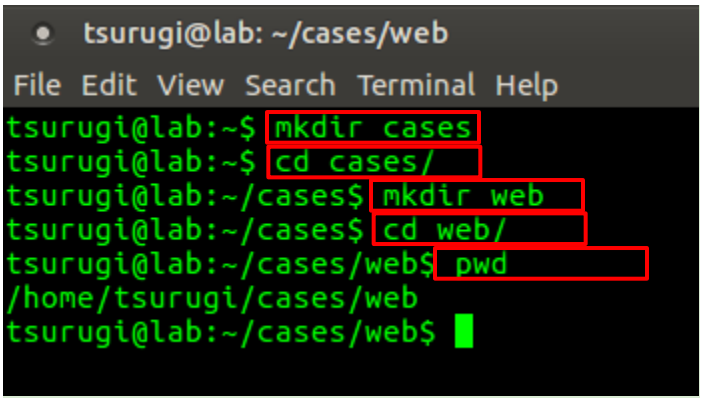
$ mkdir web

Again, change your working directory to the newly created directory, as follows:

$ cd web

Make sure you're inside the **web** directory. This could be done with the **'pwd'** command as follows:

$ pwd



Now, find a way to copy the case1web.E01 forensic image to your **'case1'** directory on your **Tsurugi Linux** system. This could be done by:

1. Dragging and dropping the file into that directory
2. Sharing it over a VM Shared directory
3. Having the file on a USB and then mounting the USB to your VM
4. Downloading it directly to your VM
5. Ask your instructor
6. Others?!

Your results should be similar to the following (my case was based on sharing a directory with my VMWare VM and copied the file over):


**NOTE(s):**

1. **Please do not move forward without doing the above steps exactly as they are…**
2. **You can copy and paste the commands literally as they are…**
3. **If you ever want to check why a command was used this way or what the options being used mean? Then use one of the following:**
    a. **man command-name**
    b. **command --help**
    c. **command -h**
        i. **Example:**
        **$ man kpartx**

## Task #1: Verification and mounting:

Before doing anything, let's verify the case image, which could be done:
$ ewfverify case1web.E01

Make sure you got the success message (this may take some time depending on the specs of your VM, so please be patient):

```
MD5 hash stored in file:                 03e4a40ebaf6071b346fb2bf217a9f3b
MD5 hash calculated over data:           03e4a40ebaf6071b346fb2bf217a9f3b

Additional hash values:
SHA1:    dd96a502f0978679191a0ea96548e5692fcb3a1c

ewfverify: SUCCESS
```

Now, let us check our drive and what volumes does it include:
$ mmls case1web.E01

```
tsurugi@lab:~/cases/web$ mmls case1web.E01
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

     Slot      Start        End          Length       Description
000:  Meta      0000000000   0000000000   0000000001   Primary Table (#0)
001:  -------   0000000000   0000002047   0000002048   Unallocated
002:  000:000   0000002048   0000499711   0000497664   Linux (0x83)
003:  -------   0000499712   0000501759   0000002048   Unallocated
004:  Meta      0000501758   0066064383   0065562626   DOS Extended (0x05)
005:  Meta      0000501758   0000501758   0000000001   Extended Table (#1)
006:  001:000   0000501760   0066064383   0065562624   Linux Logical Volume Manager (0x8e)
007:  -------   0066064384   0066064607   0000000224   Unallocated
tsurugi@lab:~/cases/web$
```

The volume we are interested is the one with index number 006. As you can see, this volume is part of a Logical Volume (*ask me about it please*), therefore we will be treating it a little bit differently than the normal way of mounting a volume in a E01 file.

First, let us mount the E01 file, which could be done as:
$ sudo ewfmount case1web.E01 /mnt/ewf1/

```
tsurugi@lab:~/cases/web$ sudo ewfmount case1web.E01 /mnt/ewf1
[sudo] password for tsurugi:
ewfmount 20180403

tsurugi@lab:~/cases/web$
```

If you list the contents of the ewf1 directory, you should see a file named "**ewf1**":
$ sudo ls -l /mnt/ewf1

```
tsurugi@lab:~/cases/web$ sudo ls -lh /mnt/ewf1
total 0
-r--r--r-- 1 root root 32G May 11 05:35 ewf1
tsurugi@lab:~/cases/web$
```

We will be using the kpartx tool to do the LVM mappings for the volumes we have. We can do a listing first, followed by the mapping itself, as seen below:
$ sudo kpartx -al ewf/ewf1

```
tsurugi@lab:~/cases/web$ sudo kpartx -al /mnt/ewf1/ewf1
loop0p1 : 0 497664 /dev/loop0 2048
loop0p2 : 0 2 /dev/loop0 501758
loop0p5 : 0 65562624 /dev/dm-1 2
loop deleted : /dev/loop0
tsurugi@lab:~/cases/web$
```

Then:
$ sudo kpartx -av ewf/ewf1

```
tsurugi@lab:~/cases/web$ sudo kpartx -av /mnt/ewf1/ewf1
add map loop0p1 (253:0): 0 497664 linear 7:0 2048
add map loop0p2 (253:1): 0 2 linear 7:0 501758
add map loop0p5 (253:2): 0 65562624 linear 7:0 501760
tsurugi@lab:~/cases/web$
```

Let us check the Logical Volume details:
$ sudo lvdisplay

```
tsurugi@lab:~/cases/web$ sudo lvdisplay
  --- Logical volume ---
  LV Path                /dev/VulnOSv2-vg/root
  LV Name                root
  VG Name                VulnOSv2-vg
  LV UUID                cEA4A3-qwNJ-U3Sj-oYW9-mK9i-1rwE-bE6f2t
  LV Write Access        read/write (activated read only)
  LV Creation host, time VulnOSv2, 2016-04-03 17:05:44 +0100
  LV Status              available
  # open                 0
  LV Size                30.51 GiB
  Current LE             7811
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:3

  --- Logical volume ---
  LV Path                /dev/VulnOSv2-vg/swap_1
  LV Name                swap_1
  VG Name                VulnOSv2-vg
  LV UUID                Q7X8aN-kcP5-SVUY-PS35-y3dz-pvRO-uAcg1f
  LV Write Access        read/write (activated read only)
  LV Creation host, time VulnOSv2, 2016-04-03 17:05:44 +0100
  LV Status              available
  # open                 0
  LV Size                768.00 MiB
  Current LE             192
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:4

tsurugi@lab:~/cases/web$
```

We are interested in:
LV Name = root ← this is the name of the logical volume
VG Name = VulnOSv2-vg ← this is the name of the Volume Group

Checking the physical volume:
$ sudo pvs

```
tsurugi@lab:~/cases/web$ sudo pvs
  PV                  VG          Fmt  Attr PSize  PFree
  /dev/mapper/loop0p5 VulnOSv2-vg lvm2 a--  31.26g    0
tsurugi@lab:~/cases/web$
```

Won't hurt to check the Logical Volume details, which could be done using:
$ sudo dmsetup info

```
tsurugi@lab:~/cases/web$ sudo dmsetup info
Name:              VulnOSv2--vg-root
State:             ACTIVE (READ-ONLY)
Read Ahead:        256
Tables present:    LIVE
Open count:        0
Event number:      0
Major, minor:      253, 3
Number of targets: 1
UUID: LVM-RJRcoEWgWPCS2S5Gfpv1ZF31vJTT8SFQcEA4A3qwNJU3SjoYW9mK9i1rwEbE6f2t

Name:              VulnOSv2--vg-swap_1
State:             ACTIVE (READ-ONLY)
Read Ahead:        256
Tables present:    LIVE
Open count:        0
Event number:      0
Major, minor:      253, 4
Number of targets: 1
UUID: LVM-RJRcoEWgWPCS2S5Gfpv1ZF31vJTT8SFQQ7X8aNkcP5SVUYPS35y3dzpvROuAcg1f

Name:              loop0p5
State:             ACTIVE (READ-ONLY)
Read Ahead:        256
Tables present:    LIVE
Open count:        2
Event number:      0
Major, minor:      253, 2
Number of targets: 1
UUID: part5-loop0

Name:              loop0p2
State:             ACTIVE (READ-ONLY)
Read Ahead:        256
Tables present:    LIVE
Open count:        0
Event number:      0
Major, minor:      253, 1
Number of targets: 1
UUID: part2-loop0

Name:              loop0p1
State:             ACTIVE (READ-ONLY)
Read Ahead:        256
Tables present:    LIVE
Open count:        0
Event number:      0
Major, minor:      253, 0
Number of targets: 1
UUID: part1-loop0
```

For more info. as usual check the man pages 😉

WOW, that was a long number of steps, now let us get into business!

Make a directory named “case1” in the same directory you are currently in, which should be “**/home/tsurugi/cases/web**”, you can double check your current location using **pwd** 😉

So, we need to mount the forensic image, which could be done in many different ways, but we are going to do it this way:
$ sudo mount -o ro,noatime,noexec /dev/VulnOSv2-vg/root case1/

Q1: What was the error that you got? Why?

Let us do some checking first using **fsstat** to see why that happened:
$ sudo fsstat /dev/VulnOSv2-vg/root | head -n 20

```
tsurugi@lab:~/cases/web$ sudo fsstat /dev/VulnOSv2-vg/root | head -n 20
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: Ext4
Volume Name:
Volume ID: 46c34db340bee5aa35423fd055183259

Last Written at: 2019-10-05 10:41:50 (BST)
Last Checked at: 2016-04-03 17:05:48 (BST)

Last Mounted at: 2019-10-05 10:41:50 (BST)
Unmounted properly
Last mounted on: /

Source OS: Linux
Dynamic Structure
Compat Features: Journal, Ext Attributes, Resize Inode, Dir Index
InCompat Features: Filetype, Needs Recovery, Extents, Flexible Block Groups,
Read Only Compat Features: Sparse Super, Large File, Huge File, Extra Inode Si

Journal ID: 00
tsurugi@lab:~/cases/web$
```

Please read all the details, they are important, but for being as brief as possible here, check the line under “Last Mounted at”. It says that it was not unmounted properly, and this might happen when the system was not shutdown properly. Therefore, this could mean that, there is some data in the journal that was not written to volume, which will usually happen once the volume comes back online.

Okay, enough talking, let’s adjust our command with the noload/norecovery option:
$ sudo mount -o ro,noatime,noexec,noload /dev/VulnOSv2-vg/root case1/

```
tsurugi@lab:~/cases/web$ sudo mount -o ro,noatime,noexec,noload /dev/VulnOSv2-vg/root case1/
tsurugi@lab:~/cases/web$
```

Super! No errors this time!

To check mounted volumes, we can use the mount command, so it would not hurt to check the mount status:

$ mount | grep case1

```
tsurugi@lab:~/cases/web$ mount | grep case1
/dev/mapper/VulnOSv2--vg-root on /home/tsurugi/cases/web/case1 type ext4 (ro,noexec,noatime,norecovery)
tsurugi@lab:~/cases/web$
```

Now let's see what we have now inside the "**case1**" directory. I'm going to use "tree" this time to do that, but feel free to use other stuff, such as "ls":

$ tree -L 1 case1/

```
tsurugi@lab:~/cases/web$ tree -L 1 case1
case1
├── bin
├── boot
├── dev
├── etc
├── home
├── initrd.img -> boot/initrd.img-3.13.0-24-generic
├── lib
├── lost+found
├── media
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin
├── srv
├── sys
├── tmp
├── usr
├── var
└── vmlinuz -> boot/vmlinuz-3.13.0-24-generic

19 directories, 2 files
tsurugi@lab:~/cases/web$
```

## Task #2: System Navigation and Timezone

PLEASE (all uppercase) take some time to navigate and understand the file hierarchy standard (FHS) before proceeding. Understanding the hierarchy of the system is very important, especially if you are new to Linux systems. It is an excellent time to ask yourself questions about the directories under root and what could be found under each one of them.

An important part of an investigation is also verifying the timezone used on the system. I know some might say that this is already provided to us. You're correct about that BUT "trust, but verify" is how we should go with investigations…

Let's check the timezone of the system:
$ cat case1/etc/timezone

Q1: What did you find?

We are going to use this information later, especially when generating our timeline, so make sure you document it on your technical notes document.

PLEASE READ ME: not all of the workshop has screenshots for a reason, which is I need you to pay attention and ask questions and not just copy and paste commands. Therefore, I will purposely leave questions and commands unanswered, but they will be together, so enjoy the ride 😉

## Task #3: Checking System Logs

Let us start by checking the logs. The logs under Linux are found under /var/log directory:

- Important logs to check are syslog (named messages on other systems, auth.log, wtmp, btmp, etc)
- Others depending on what systems/applications are installed on the system

First log we are going to check is the wtmp and btmp

First let us do it this way:
$ last -f case1/var/log/wtmp

Now again but with head (hope you notice the difference):
$ last -f case1/var/log/wtmp | head -n 10

Q1: Who was the last user to login to the system?

Q2: From where did the login happen?

Now the btmp file (failed login attempts). Do the same as before without using head and with sudo powers 😊:
$ sudo last -f case1/var/log/btmp | head -n 20

Q3: Why are there so many failed login attempts, what do you think is happening? Explain your answer.

Check the auth.log file and explain what happened:
$ sudo cat case1/var/log/auth.log

Q4: does it match the activity that you saw in the previous log (btmp)?

Q5: Was the user successful in obtaining access using this method? Explain with proof.

Please carefully go through this log file, it is very important.
Q6: Did you find any user account access that was successful? Which user was it?

Search for the lines that show the following (README):
Oct 5 12:52:52 VulnOSv2 sshd[2370]: Connection closed by 192.168.210.131 [preauth]
Oct 5 12:52:52 VulnOSv2 sshd[2372]: Connection closed by 192.168.210.131 [preauth]

Q7: What happened after that and at what time? Explain your answer.

Let us also check the lastlog file using the "strings" command:
$ strings case1/var/log/lastlog

Q8: What IP Address did you find?

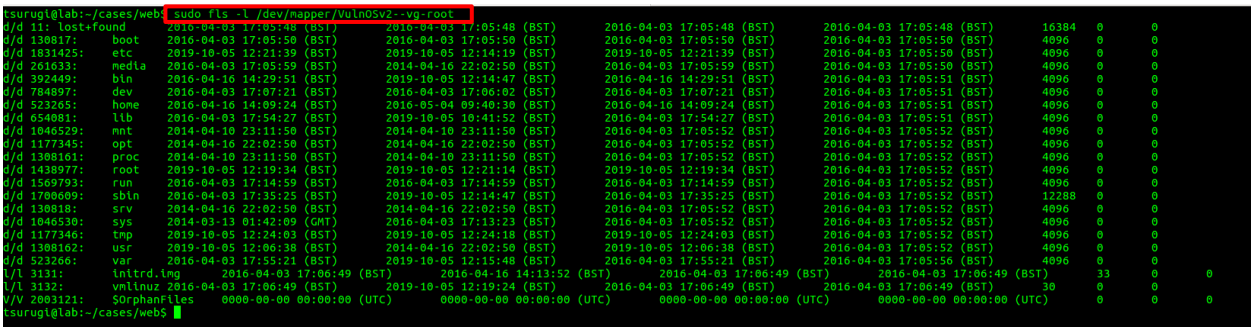
Q9: Does it match the IP Address you saw in any of the previous log files? Please document all files that you found that IP address in, you will need them for your final report.

## Task #4: Working with The Sleuth Kit (TSK)

The idea of this task is to get familiar with basic TSK tools available. Let us start by listing the contents of the logs directory. To do that, we first need to find the log directory's inode number and then use it to list the contents. But before you start, TSK deals with volumes/disks not with volumes that are mounted! In other words, we cannot use TSK directly with files in our case1/ directory.

$ sudo fls -l /dev/mapper/VulnOSv2-vg-root

The inode number, is the number you will find in the 2nd column, like the following:




The inode number for the "**lost+found**" directory is **11** and for the **boot** directory is **130817**, and so on and so forth.

Pipe the results and grep "var" to focus on the var directory only:

$ sudo fls -l /dev/mapper/VulnOSv2-vg-root | grep var

Now use the inode value found from the previous step and add it to the end of your command to be something like this (replace inode# with the number you found):

$ sudo fls -l /dev/mapper/VulnOSv2-vg-root inode#

Q1: What was the inode number for the log directory?

Now, you might be thinking, how can we check if that is true? Well, the good thing is, that TSK comes with a command named "ffind" where we can use the inode number to find which file it is pointing to. Let's check the inode no of the logs directory:

$ sudo ffind /dev/mapper/VulnOSv2-vg-root inode#

Now I want you to go back and look at the contents of the logs directory but focus on lastlog file. Use the | grep lastlog to help narrow down our search

$ sudo fls -l /dev/VulnOSv2-vg/root inode# | grep lastlog

Q2: How many files did you find that have that name and why? Could you explain?

Q3: What does **relloc** mean here (README)?

Use the inode# of that file and search for what file does it belong to, using the same approach above.

We can extract a file from the volume using the TSK's icat command:
$ sudo icat /dev/mapper/VulnOSv2-vg-root inode# | tee lastlog

**Note:** the command above will concatenate the output of the file to standard output (stdout) and using the tee command we can also copy the output to a file of our name, which was lastlog in the command above.

Now check the file's type and content with both the file and strings commands:
$ file lastlog

Then
$ strings lastlog

Please use the same methods mentioned above to do other experiments and make sure you're comfortable with using the TSK commands we've covered until now. Do not move forward without understanding everything before this message. At the end, our goal is to learn, not just to run commands and find the answer to the questions!

## Task #5: Users, Groups, Permissions, etc

From what we saw in the logs, it seems a user was added to the system and another user **'mail'** was being used, which is very weird!

On a Linux system, the user information is found in the /etc/passwd file. What are you waiting for? Check it out using the same methods we used in Task #4 or you can even use basic Linux commands such as:

$ cat case1/etc/passwd

```
tsurugi@lab:~/cases/web$ cat case1/etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/bin/bash
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
php:x:999:999::/usr/php:/bin/bash
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
libuuid:x:100:101::/var/lib/libuuid:
syslog:x:101:104::/home/syslog:/bin/false
messagebus:x:102:106::/var/run/dbus:/bin/false
landscape:x:103:109::/var/lib/landscape:/bin/false
vulnosadmin:x:1000:1000:vulnosadmin,,,:/home/vulnosadmin:/bin/bash
mysql:x:104:113:MySQL Server,,,:/nonexistent:/bin/false
webmin:x:1001:1001::/home/webmin:
sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin
postfix:x:106:114::/var/spool/postfix:/bin/false
postgres:x:107:116:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
tsurugi@lab:~/cases/web$
```

Please take some time to read the contents of the file and if you search online for how the passwd file should look like or what it should contain, that would be great, since it will help you understand the content. After you've finished going through the file, check the questions below.

Q1: What did you find that is suspicious, or in other words, which entries do you think are suspicious and why? Please explain your answer.

Let's check if they have passwords to login. This can be found in the **shadow** file, which is found under the /etc directory
$ cat case1/etc/shadow

In the **shadow** file, the second column (: is the separator), if you find a ***** it means no password is there, but if you find a long string? Then a password exists for that user.

Q2: List who has a password?

We need to check the group info that these users belong to. The group info is found in **/etc/group** file.

Q3: Any suspicious entries found?

Q4: Who has sudo access, or in other words, is in the sudo group?

Q5: What is the home directory of the user that was added to the system?

You can use the same methods explained previously to find the user's home directory and then check its directory content, using either TSK or basic Linux commands under case1 directory.

Explore the directory and its contents.
Q6: Did you find anything useful?

Move on to investigate the mail user.
Q7: Did you find anything in mail's home directory?

Check the contents of the **.bash_history** (a file that is used to store a history of all the commands used on a Linux system). The **dot** at the beginning of the file, denotes that this is a hidden file.

Create directories for each user for organization purposes and then add the "| tee users/mail/bash-history" at the end of your command, to list and copy out the file. You know how to do this, so get moving...

From the bash history, it seems the user has logged into both root and php. Therefore, let us check the root's directory, since we already did for php.

Q8: What did you find?

poweroff
whoami
…..
….
….

cat .cache/motd.legal-displayed
logout

*Note: not all the contents are listed above.*

Use the history to dig deeper with your investigation, since they could be a good map to what to check and where to look for activity. After that move to the next task.

## Task #6: Data Recovery / File Carving

In this task we want to recover the files that have been deleted, especially the file that was deleted based on the commands we found being used in the **.bash_history** file. Unfortunately, on an EXT4 file system, once the file is deleted, the metadata that points to the file is zeroed out and there are no longer any pointers pointing back to the volume.

Now, there is some good news! Let us assume that you manage to get your hands on the system before any of the deleted files metadata was overwritten, then we might be able to recover that data with the help of the file system's journal. If this method does not work, let's say because you arrived to the crime scene late or this was an operation that happened a couple of months ago, then we still could probably apply file carving techniques to extract the deleted files, as long as they have not been overwritten.

Therefore, let us go with option (a) and use the journal to help us recover the files. To extract the journal, we will be using debugfs and asking it to dump the file with the inode **#8**, which is the inode number for the file system's journal. This can be done as:
$ sudo debugfs -R 'dump <8> ./journal' /dev/VulnOSv2-vg/root

You should end up with a 128MB file (size of the EXT4 journal) as seen below:

```
tsurugi@lab:~/cases/web$ ll
total 1.3G
drwxr-xr-x 21 root    root    4.0K Apr  3  2016 case1/
-rw-rw-r--  1 tsurugi tsurugi 1.1G May 11 05:11 case1web.E01
-rw-r--r--  1 root    root    128M May 11 06:40 journal
-rw-rw-r--  1 tsurugi tsurugi 286K May 11 06:26 lastlog
drwxrwxr-x  5 tsurugi tsurugi 4.0K May 11 06:39 users/
tsurugi@lab:~/cases/web$
```

Now we want to search for files that were deleted between the Oct. 5th and 8th, based on the case brief that was given to us. Therefore, let us define a variable with that value:
**AFTER**=$(date -d"2019-10-05 00:00:00" +%s)

**BEFORE**=$(date -d"2019-10-08 00:00:00" +%s)

Before attempting the recovery step, I would like you to check or list what files actually we can recover with the help of the journal for example from the /tmp directory. This can be done using the command below (please adjust the values AFTER & BEFORE with the corresponding numbers from the commands above):
$ sudo ext4magic /dev/VulnOSv2-vg/root -a $**AFTER** -b $**BEFORE** -f tmp -j journal -l

Q1: What was the “tmp” word used in the options above for? (hint: man ext4magic)

Please check the man page for ext4mage on how to set a time range for your search.
Q2: What do you think? Not much?

Now, let us perform the recovery step itself instead of just listing the files that are recoverable, which could be done as seen below:
$ sudo ext4magic /dev/VulnOSv2-vg/root -a $**AFTER** -b $**BEFORE** -f tmp -j journal -r -d output1/

If you check the output directory and didn’t find much, then let’s attempt another approach. This could also be done using ext4magic, but by providing the **-m** option to try and recover all the deleted files on the volume, as seen below:
$ sudo ext4magic /dev/VulnOSv2-vg/root -a $**AFTER** -b $**BEFORE** -f tmp -j journal -m -d output2/

Use:
$ sudo tree -L 1 output2/

Please check the man pages for the ext4magic tool, this is truly an excellent tool with so many more features/capabilities, so what are you waiting for? Go check them out!

Search through the files recovered and find the Kernel exploit used. As a hint on how to do that, you can Google the name of the dot c file that you found to find what it is and what its contents looks like, then use that in your search. The Linux find command is very good for this, learn how to use it.

## Task #7: Finding how the threat actor gained access

Since, we know the threat actor did not gain access using Brute force, then how did the threat actor get access to the system? We still have not found the answer to that question. Therefore, we need to check other methods that were probably used.

Now, since this is a web server that is hosting the company's website, the attack might have been successful through that. So, let us gather some information about our system and move forward. Let us first check what "web application server" is being used on the system, which could be found under /etc/ directory.

Q1: What "web application" was being used? (**Hint:** /var/www/html)

Q2: Try to identify the version of the web application and do you think it was vulnerable?

**Note:** this requires some research and it is not straight forward, plus experience with web applications would definitely help!

Q3: Search through the web application server's error and access logs. Did you find anything? (**Hint:** you should find weird POST requests 😉)

Do some research and understand what these PHP functions are used for:

- passthru
- eval
- base64_decode

Check the long string in the HTTP POST request.

Q4: What type of encoding is being used?

Now, you might be asking, how can we decode the string we found? Well the answer is very easy in this case. We can either echo the string to base64 command or save it in a file and cat to base64 as we can see here (blob saved in post.txt file):

```
$ cat post.txt | base64 -d
```

Clean your results and try to understand the code found.

Q6: What was the final result of the code you just decoded, and did you find new php functions being used? Please explain your answer.

## Task #8: Generating a super timeline and filtering it

This task could be done at the beginning or the end, it depends on how you approach your cases. I am not going to say which is good and which is bad, just use the approach you feel more comfortable with. Now, to generate a super timeline for our case, we will be using the log2timeline.py framework. This could be seen in the command below (**Note:** replace timezoneValue with the value that was found in Task #2 question #1):

```
$ sudo log2timeline.py -z timezoneValue -t / --parse linux,apache_access,apt_history case1.timeline case1/
```

Q1: What does the linux parser used in the command above search for?

Q2: Why did we add the apache_access parser?

Finally, let's filter our timeline and sort it using the psort.py tool, which can be done as:

```
$ sudo psort.py -z timezoneValue -o L2tcsv -w webserver.csv case1.timeline "date > '2019-10-05 00:00:00' AND date < '2019-10-08 00:00:00'"
```

We will go over this together during the session…

**Note(s):**

1. Spaces were added to the command above to help you understand it, otherwise it is not needed.
2. Use whatever tool or spreadsheet application to go through your timeline. For quick checks, I usually use Eric Zimmerman's Timeline Explorer, but it's up to you.

## Deliverables:

1. How the threat actor gained access to the system?
   ANSWER:

2. What privileges were obtained and how?
   ANSWER:

3. What modifications were applied to the system?
   ANSWER:

4. What persistent mechanisms are being used?
   ANSWER:

5. Could this system be cleaned/recovered?
   ANSWER:

6. Notes and recommendations
   ANSWER: