

Case #1: 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:

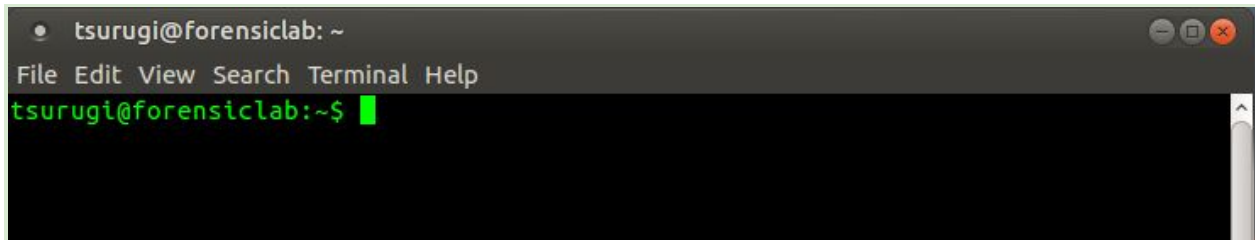
1. Listing the volumes and mounting a forensic case image
2. Searching through the FHS
3. Search in log files
4. Use TSK tools to list info of the image and deal with EXT4 fs
5. Use debugfs, EXT4 journal and ext4magic to recover deleted files
6. Generate and filter a super timeline

Note: please use tables and screenshots to represent your results if needed. Like I usually say **“Screenshot or it didn’t happen!”**.

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 “osdfcon”

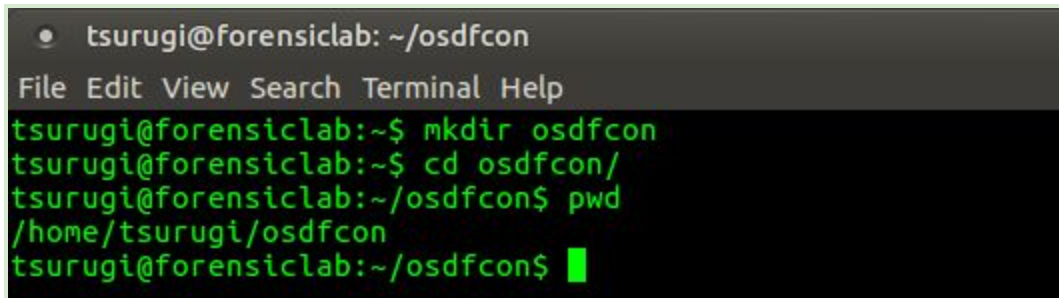
```
$ mkdir osdfcon
```

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

```
$ cd osdfcon
```

Make sure you’re inside the osdfcon directory. This could be done with the ‘pwd’ command as follows:

```
$ pwd
```



Now, find a way to copy the Webserver.E01 forensic image to your ‘osdfcon’ 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. Others?!

Your results should be similar to the following:

```
• tsurugi@forensiclab: ~/osdfcon
File Edit View Search Terminal Help
tsurugi@forensiclab:~/osdfcon$ ll
total 1,1G
-rwxrwxrwx 1 tsurugi tsurugi 1,1G oct.  6 20:04 Webserver.E01*
tsurugi@forensiclab:~/osdfcon$
```

Now create a directory to mount our case and E01 file

```
$ mkdir case1 ewf
```

```
tsurugi@forensiclab:~/osdfcon$ mkdir case1 ewf
tsurugi@forensiclab:~/osdfcon$
```

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 Webserver.E01
```

Make sure you got the success message:

```
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 Webserver.E01
```

```
tsurugi@forensiclab:~/osdfcon$ mmls Webserver.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@forensiclab:~/osdfcon$
```

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

First, let's mount the E01 file, which could be done as:

```
$ sudo ewfmount Webserver.E01 ewf/
```

```
tsurugi@forensiclab:~/osdfcon$ sudo ewfmount Webserver.E01 ewf/
[sudo] password for tsurugi:
ewfmount 20180403
```

If you list the contents of the ewf directory, you should see a file named “ewf1”:

```
$ sudo ls -l ewf
```

```
tsurugi@forensiclab:~/osdfcon$ sudo ls -l ewf
total 0
-r--r--r-- 1 root root 33825079296 oct. 18 03:24 ewf1
tsurugi@forensiclab:~/osdfcon$
```

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
```

Then:

```
$ sudo kpartx -av ewf/ewf1
```

```
tsurugi@forensiclab:~/osdfcon$ sudo kpartx -al ewf/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@forensiclab:~/osdfcon$ sudo kpartx -av ewf/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@forensiclab:~/osdfcon$
```

Let's check the Logical Volume details:

```
$ sudo lvdisplay
```

```
tsurugi@forensiclab:~/osdfcon$ 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 18:05:44 +0200
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
```

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@forensiclab:~/osdfcon$ sudo pvs
PV          VG          Fmt  Attr  PSize  PFree
 /dev/mapper/loop0p5  VulnOSv2-vg  lvm2  a--   31,26g    0
tsurugi@forensiclab:~/osdfcon$
```

```
$ sudo dmsetup info
```

Whoo, that was a long number of steps, now let's get into business!

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?

```
tsurugi@forensiclab:~/osdfcon$ sudo mount -o ro,noatime,noexec /dev/VulnOSv2-vg/root case1
mount: /dev/mapper/VulnOSv2--vg-root: can't read superblock
tsurugi@forensiclab:~/osdfcon$
```

Let's do some checking first using **fsstat** to see why that happened:

```
$ sudo fsstat /dev/VulnOSv2-vg/root | head -n 20
```

```
tsurugi@forensiclab:~/osdfcon$ 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 11:41:50 (CEST)
Last Checked at: 2016-04-03 18:05:48 (CEST)

Last Mounted at: 2019-10-05 11:41:50 (CEST)
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 Size

Journal ID: 00
tsurugi@forensiclab:~/osdfcon$
```

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 noexec/norecovery option:

```
$ sudo mount -o ro,noatime,noexec,noload /dev/VulnOSv2-vg/root case1/
```

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

Super!

Would not hurt to check the mount status:

```
$ mount | grep case1
```

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

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@forensiclab:~/osdfcon$ 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@forensiclab:~/osdfcon$
```

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.

Task #3: Checking System Logs

Let's 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`

```
$ 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):

```
$ 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:

```
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's 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 list all files where you found that IP address.

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:

```
tsurugi@forensiclab:~/osdfcon$ sudo fls -l /dev/VulnOSv2-vg/root
d/d 11: lost+found      2016-04-03 18:05:48 (CEST)    2016-04-03 18:05:48 (CEST)    2016-04-03 18:05:48 (CEST)2
016-04-03 18:05:48 (CEST)    16384    0    0
d/d 130817: boot        2016-04-03 18:05:50 (CEST)    2016-04-03 18:05:50 (CEST)    2016-04-03 18:05:50 (CEST)2
016-04-03 18:05:50 (CEST)    4096    0    0
d/d 1831425: etc         2019-10-05 13:21:39 (CEST)    2019-10-05 13:14:19 (CEST)    2019-10-05 13:21:39 (CEST)2
016-04-03 18:05:50 (CEST)    4096    0    0
d/d 261633: media        2016-04-03 18:05:59 (CEST)    2014-04-16 23:02:50 (CEST)    2016-04-03 18:05:59 (CEST)2
016-04-03 18:05:50 (CEST)    4096    0    0
```

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**”:

```
$ sudo fls -l /dev/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/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/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 **reloc** mean here?

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/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 case.

Task #5: Users, Groups, Permissions, etc

So, 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@forensiclab:~/osdfcon$ 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@forensiclab:~/osdfcon$
```

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 php 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?

Extract 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.

Use "`tee 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's 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 of the contents is listed above.

Use the history to dig deeper with your investigation, then 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 is no longer any pointers pointing back to the volume.

Now, there is some good news! Let's 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's 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@forensiclab:~/osdfcon$ sudo ls -lh
total 1,3G
drwxr-xr-x 21 root    root    4,0K avril  3  2016 case1
drwxr-xr-x  2 root    root         0 janv.  1  1970 ewf
-rw-r--r--  1 root    root   128M oct.   18 07:19 journal
-rw-rw-r--  1 tsurugi tsurugi 286K oct.   18 06:54 lastlog
-rwxrwxrwx  1 tsurugi tsurugi 1,1G oct.   6 20:04 Webserver.E01
tsurugi@forensiclab:~/osdfcon$
```

Now we want to search for files that were deleted after 10 days ago, based on the case brief that was given to us. Therefore, let's define a variable with that value:

```
$ after=$(date -d"-20days" +%s)
```

Before attempting the recovery step, I would like you to check/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 following:

```
$ sudo ext4magic /dev/VulnOSv2-vg/root -a $after -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's 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 -f tmp -j journal -l -r -d output/
```

If you check the output directory, we won't find much! Therefore, 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 -f tmp -j journal -m -d output2/
```

```
tsurugi@forensiclab:~/osdfcon$ sudo tree -L 1 output/
output/
├── bin
├── dev
├── etc
├── home
├── initrd.img -> boot/initrd.img-3.13.0-24-generic
├── lib
├── MAGIC-2
├── MAGIC-3
├── root
├── sbin
├── tmp
├── usr
├── var
└── vmlinuz -> boot/vmlinuz-3.13.0-24-generic

12 directories, 2 files
tsurugi@forensiclab:~/osdfcon$
```

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 S/He get access to the system? We still haven't 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's gather some information about our system and move forward. Let's 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?

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 your decode 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'm 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  
timeline.case1 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 timeline.case1  
"date > '2019-10-05 01:01:01' AND date < '2019-10-08 01:01:01'"
```

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: