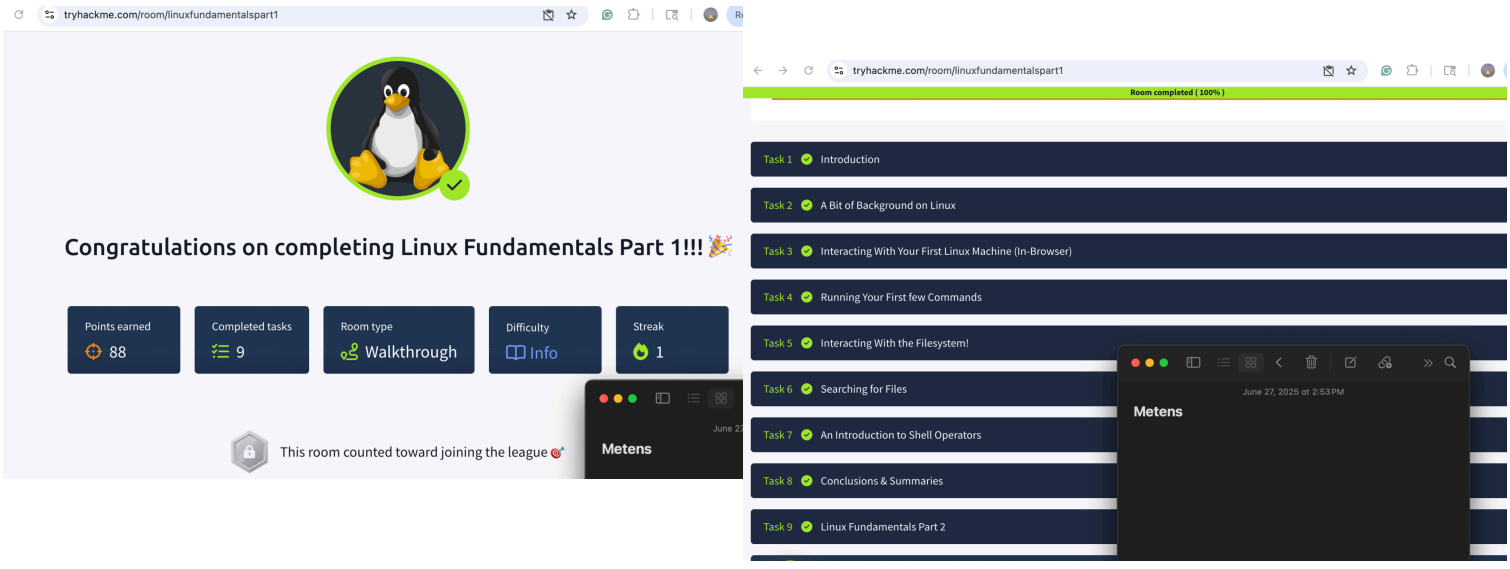
Lab 2.1: Linux File System

By: Nathan Metens (metens@pdx.edu)

Task 1: TryHackMe Linux Tutorial **1**
Task 2: | (the pipe character), head **2**
Task 3: grep **2**
Task 4: Matching multiple instances **3**
Task 5: Matching wildcards **4**
Task 6: find **6**
Task 7: Using -exec and xargs **7**
Task 8: Tampering with find **7**
Task 9: Polyglot files **8**
Task 10: file and polyglots **9**
Task 11: TryHackMe Linux file system forensics **10**

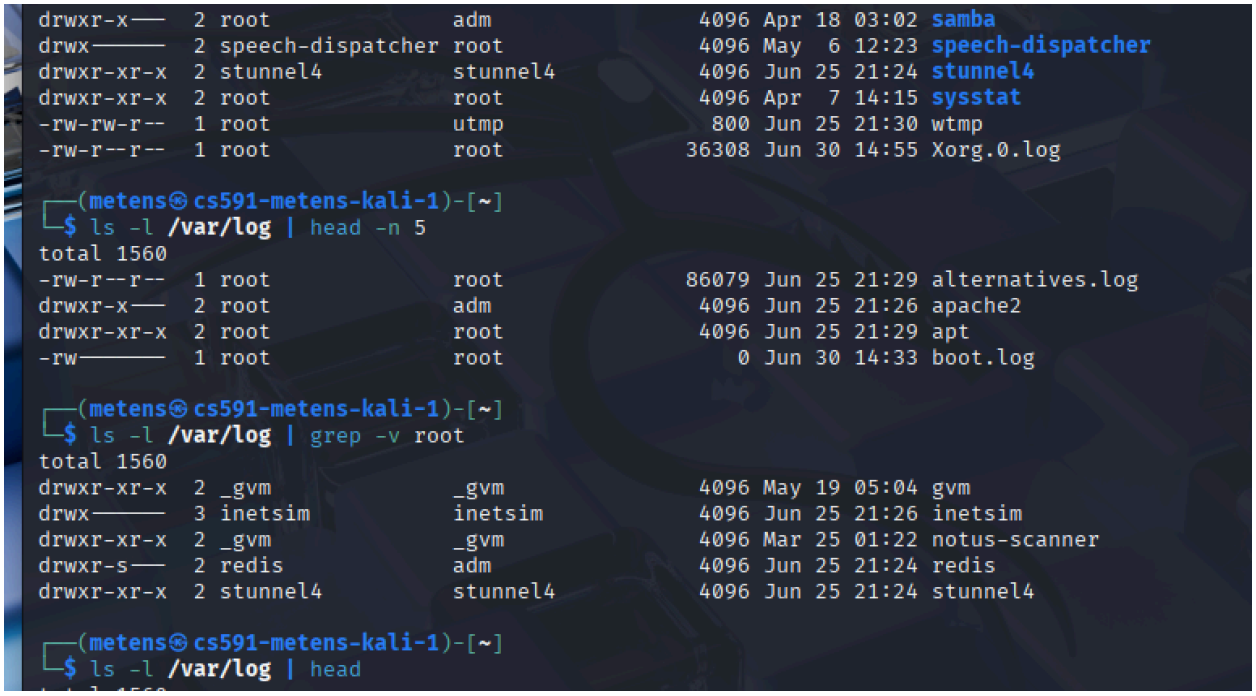
Task 1: TryHackMe Linux Tutorial

This TryHackMe room was mostly review, although I found out the reason why `find` never worked for me whenever I tried to use it in the past! I'd always do `find file.txt` and it would never find it, even though the file was right there in the directory. The problem was that I always forgot the -name flag. After testing on my Mac, it turns out that I also need to include the directory… So `find . -name file.txt` would work. Pretty cool.

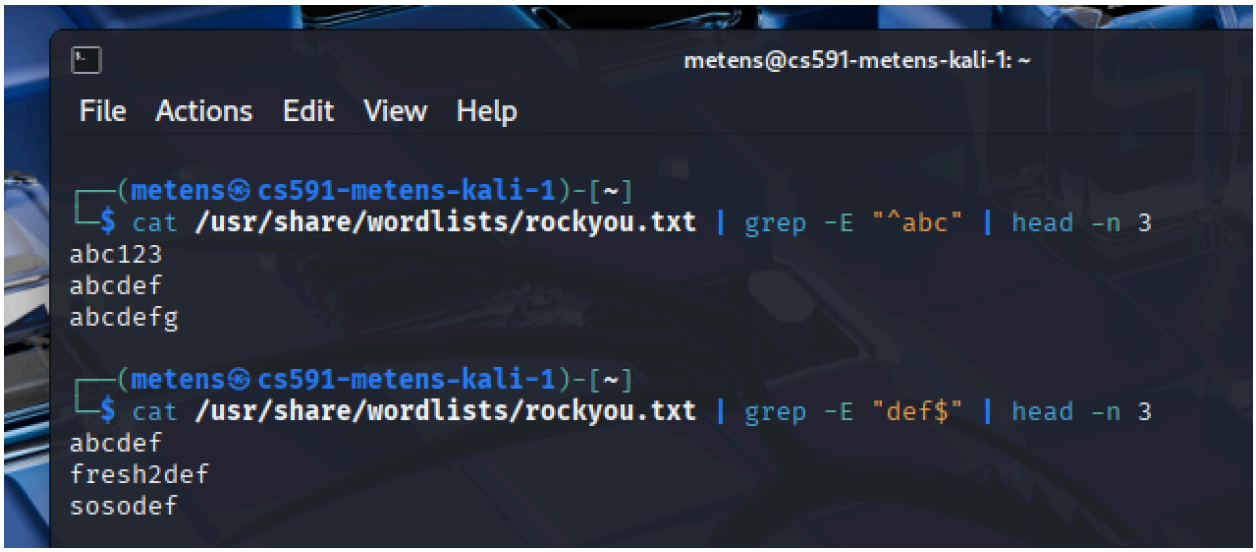


Task 2: | (the pipe character), head

`ls -l /var/log | grep auth` returned nothing in my Kali VM. So piping passes the output of one command as the input to another command. And then grep searches for a certain word or pattern and only outputs the lines containing this pattern or word. The -v flag for grep does everything except what comes after the -v part (inverting). With head, we can do it alone (first 10 lines) or we can specify a number with fewer lines using the -n flag.

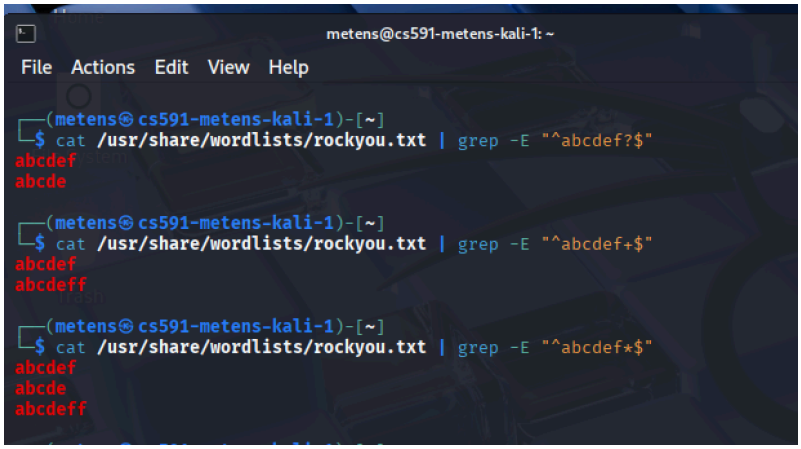


Task 3: grep

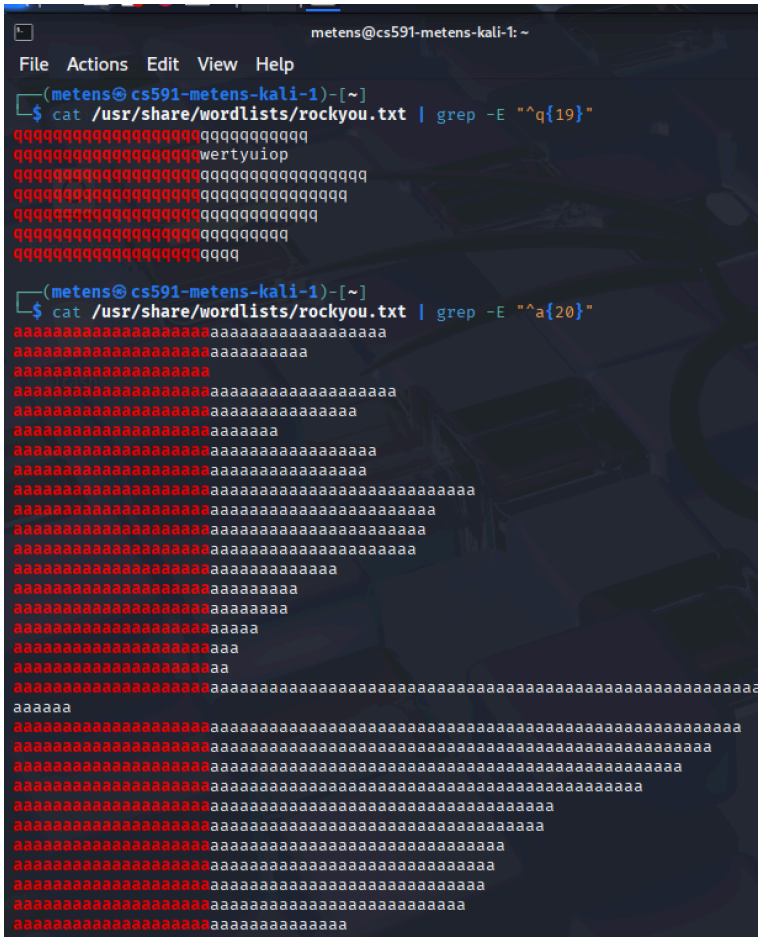


Since I wanted to check the entire rockyou file, I started with cat to output the entire file, then passed that output as input to the extended regex grep command, then output only the first 3 values from that entire list of passwords. The only thing changed from that was `def$` to see the first three passwords that ended in “def” for the second command. Super cool!

Task 4: Matching multiple instances

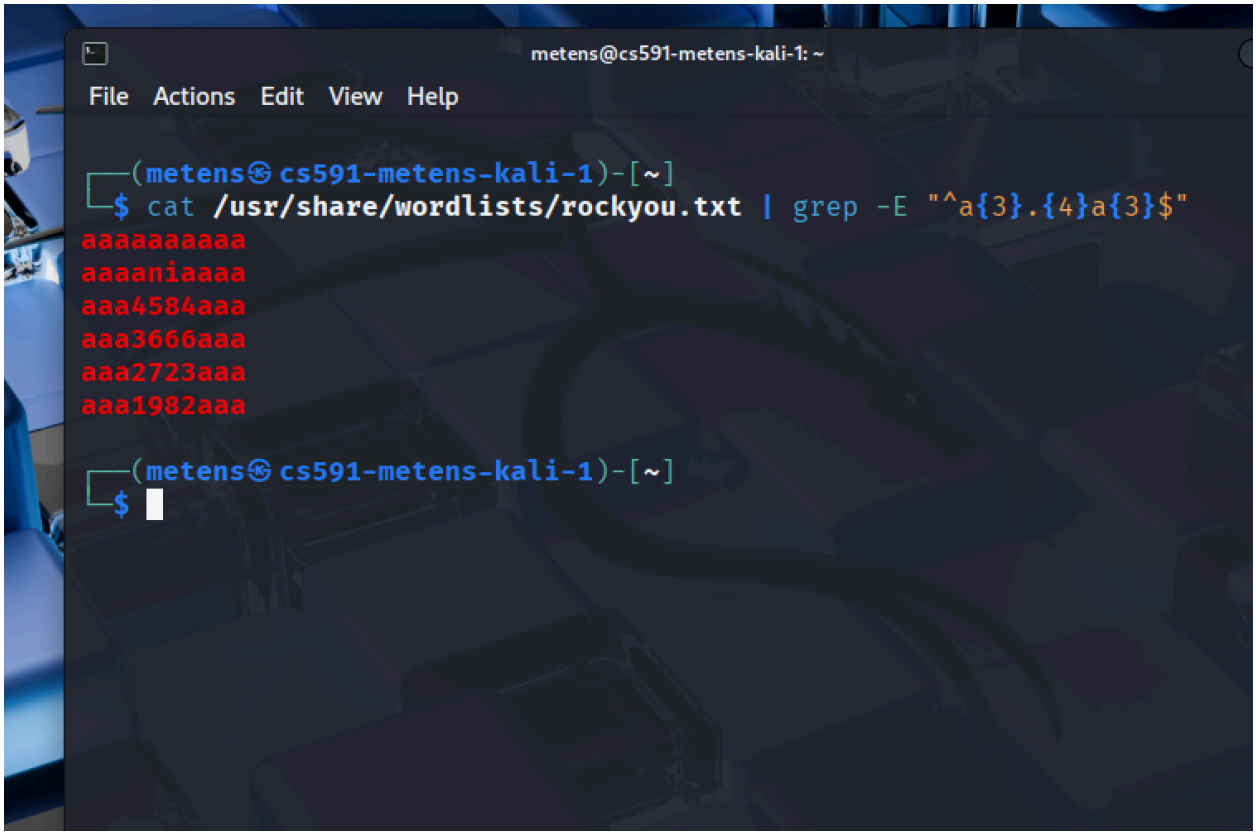


There are a few operators that determine where in the regular expression the characters are found and how many occurrences are requested. Having a `?` means zero or one occurrence, `+` means one or more, and `*` means zero or more. Then the braces can hold a certain number of characters requested to search. Handy dandy!

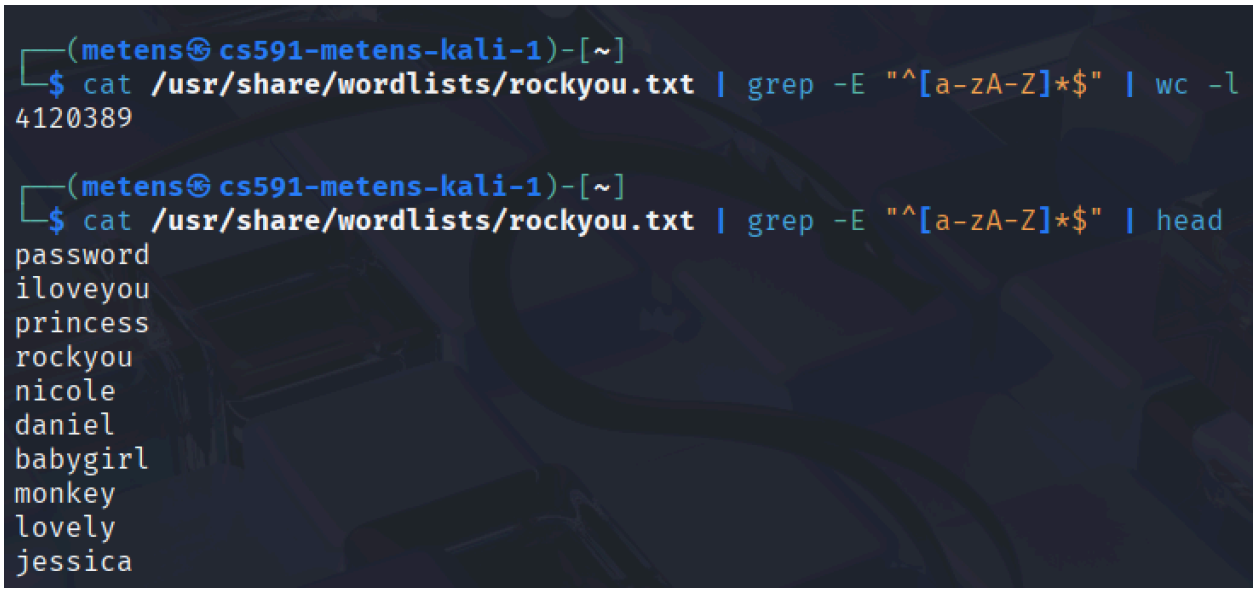


Task 5: Matching wildcards

I created a regex with 3 letters of a at the start, then any characters in the middle (only 4), and then 3 a's at the end…



The total number of passwords that have only alphabetical letters is more than 4,00,000:



The total number of passwords with exactly 6 digits can be found like this: start with any digit and end with any digit, and make sure to count only 6 total digits:

```
┌──(metens㉿cs591-metens-kali-1)-[~]
└─$ cat /usr/share/wordlists/rockyou.txt | grep -E "^[0-9]{6}$" | wc -l
390529

┌──(metens㉿cs591-metens-kali-1)-[~]
└─$ cat /usr/share/wordlists/rockyou.txt | grep -E "^[0-9]{6}$" | head
123456
654321
111111
000000
123123
666666
121212
112233
555555
789456
```

The non-alphanumeric passwords:

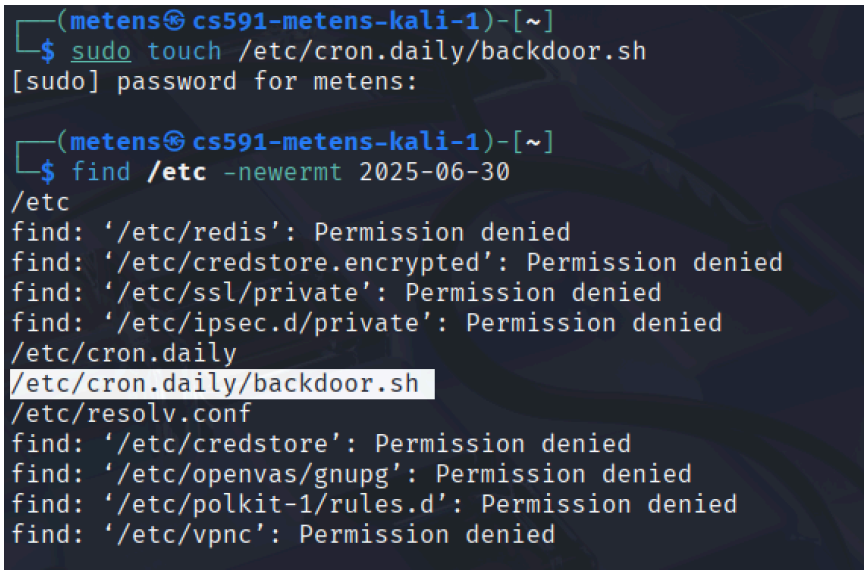
```
┌──(metens㉿cs591-metens-kali-1)-[~]
└─$ cat /usr/share/wordlists/rockyou.txt | grep -E "^[^a-zA-Z0-9]*$" | wc -l
5421

┌──(metens㉿cs591-metens-kali-1)-[~]
└─$ cat /usr/share/wordlists/rockyou.txt | grep -E "^[^a-zA-Z0-9]*$" | head
******
......

!@#$%^
*******
@@@@@@
********
$$$$$$
*********
*****
```

The password that starts with o, has a “>” in the middle, and ends with 4 numbers is:

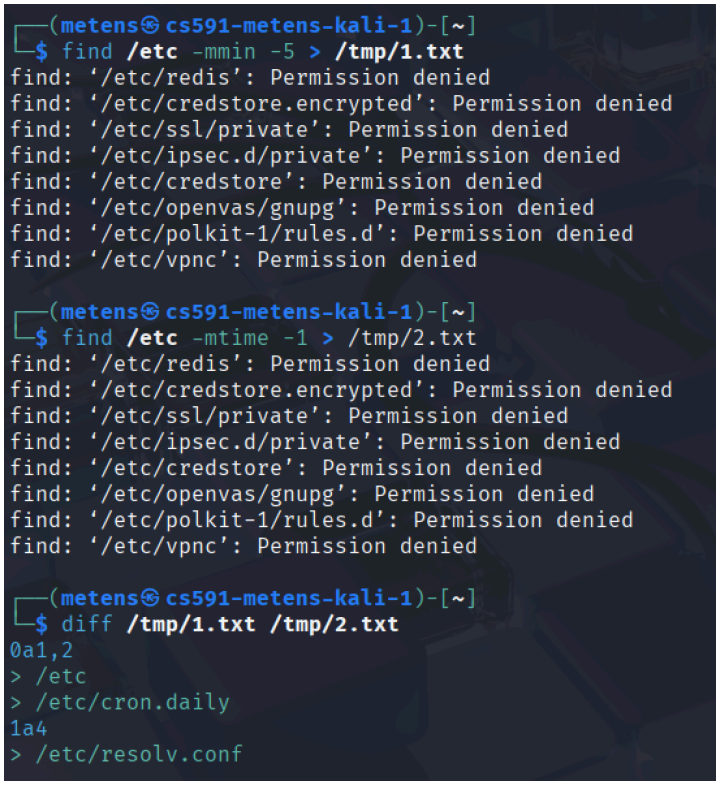
```
┌──(metens㉿cs591-metens-kali-1)-[~]
└─$ cat /usr/share/wordlists/rockyou.txt | grep -E "^o.*>.*[0-9]{4}$" | head
omi>_<0309
```

Task 6: find

Using find to find all files created after today's date:

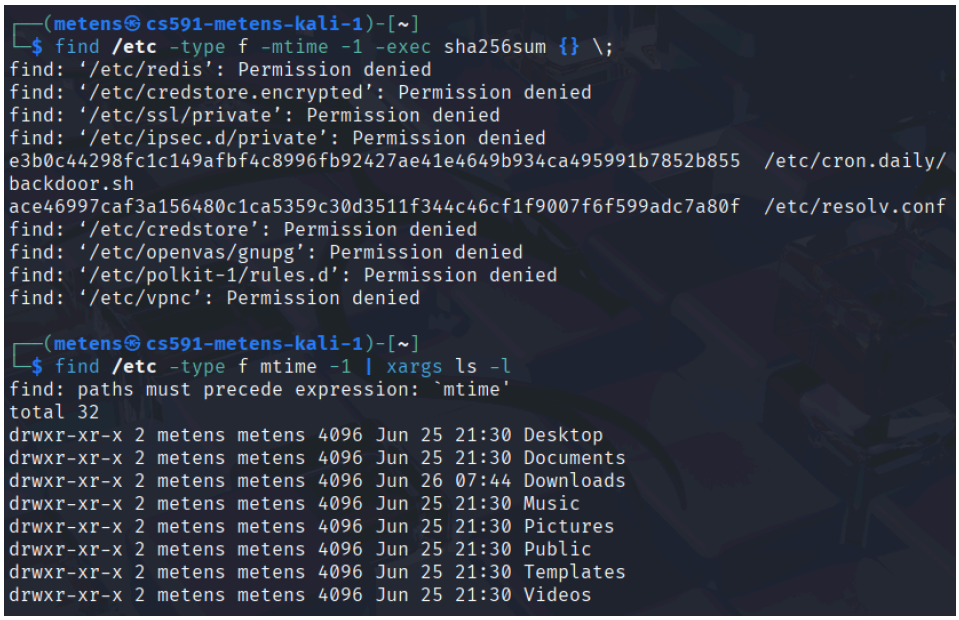


Seeing the difference between files changed within the last 5 minutes versus the last day:

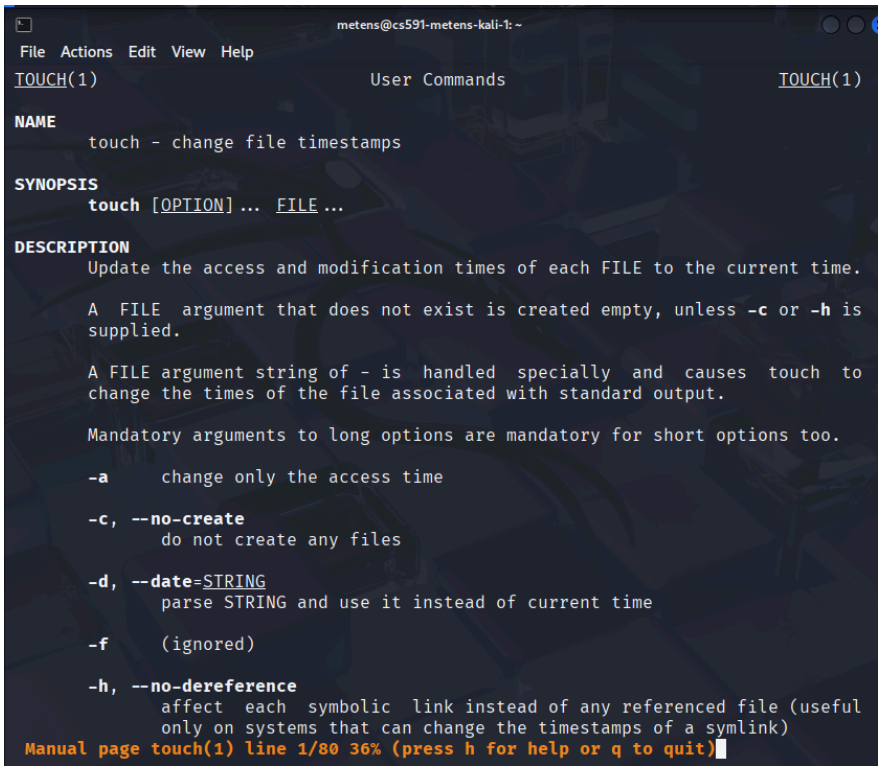


Task 7: Using -exec and xargs

Showing all the files that have been output within the last day using xargs to `ls -l`:



Task 8: Tampering with find



It looks like to change the access time of a file, the `-a` flag should be used.

To change the last modification time of the file, the `-m` flag should be used. The `–date=STRING` flag allows us to change the date and modify the times.

```
┌──(metens㉿cs591-metens-kali-1)-[~]
└─$ sudo touch --date="2024-06-30" /etc/cron.daily/backdoor.sh

┌──(metens㉿cs591-metens-kali-1)-[~]
└─$ ls -l /etc/cron.daily/backdoor.sh
-rw-r--r-- 1 root root 0 Jun 30  2024 /etc/cron.daily/backdoor.sh

┌──(metens㉿cs591-metens-kali-1)-[~]
└─$ find /etc -mmin -30 -exec ls -ld {} \;
find: ‘/etc/redis’: Permission denied
find: ‘/etc/credstore.encrypted’: Permission denied
find: ‘/etc/ssl/private’: Permission denied
find: ‘/etc/ipsec.d/private’: Permission denied
drwxr-xr-x 2 root root 4096 Jun 30 23:23 /etc/cron.daily
find: ‘/etc/credstore’: Permission denied
find: ‘/etc/openvas/gnupg’: Permission denied
find: ‘/etc/polkit-1/rules.d’: Permission denied
find: ‘/etc/vpnc’: Permission denied

┌──(metens㉿cs591-metens-kali-1)-[~]
└─$
```

Changed the timestamp of the backdoor.sh file to one year ago on this date.

Task 9: Polyglot files

The author of the two polyglot files:

- https://web.cecs.pdx.edu/~rchaney/Classes/cs491/polyglot/GIF+PDF.gif
- https://web.cecs.pdx.edu/~rchaney/Classes/cs491/polyglot/GIF+PDF.pdf

Jesse Chaney.

Now, running the `curl -IL` command on both the above files:

```
metens@cs591-metens-kali-1: ~
File  Actions  Edit  View  Help

┌──(metens㉿cs591-metens-kali-1)-[~]
└─$ curl -IL https://web.cecs.pdx.edu/~rchaney/Classes/cs491/polyglot/GIF+PDF.gif
HTTP/1.1 200 OK
Date: Tue, 01 Jul 2025 06:52:03 GMT
Server: Apache
Last-Modified: Sun, 29 Jun 2025 00:06:23 GMT
ETag: "ed96d-638aaae82c7f9"
Accept-Ranges: bytes
Content-Length: 973165
Connection: close
Content-Type: image/gif


┌──(metens㉿cs591-metens-kali-1)-[~]
└─$ curl -IL https://web.cecs.pdx.edu/~rchaney/Classes/cs491/polyglot/GIF+PDF.pdf
HTTP/1.1 200 OK
Date: Tue, 01 Jul 2025 06:52:10 GMT
Server: Apache
Last-Modified: Sun, 29 Jun 2025 00:06:23 GMT
ETag: "ed96d-638aaae82c7f9"
Accept-Ranges: bytes
Content-Length: 973165
Connection: close
Content-Type: application/pdf


┌──(metens㉿cs591-metens-kali-1)-[~]
└─$
```

Identical values from both files:

- **Server**
- **ETag**
- **Accept-Ranges**
- **Content-Length**
- **Connection**

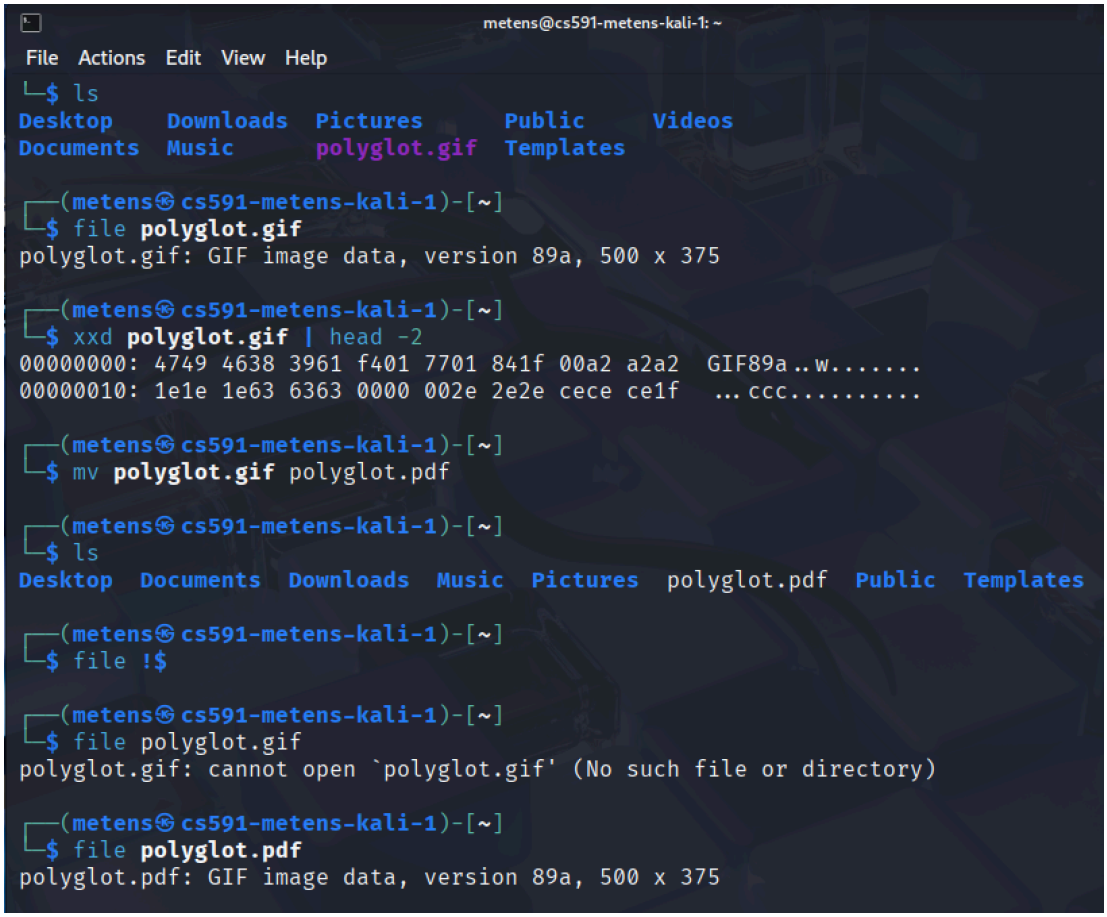
Difference between the files:

- **Content-Type. One is image/gif, the other is application/pdf**

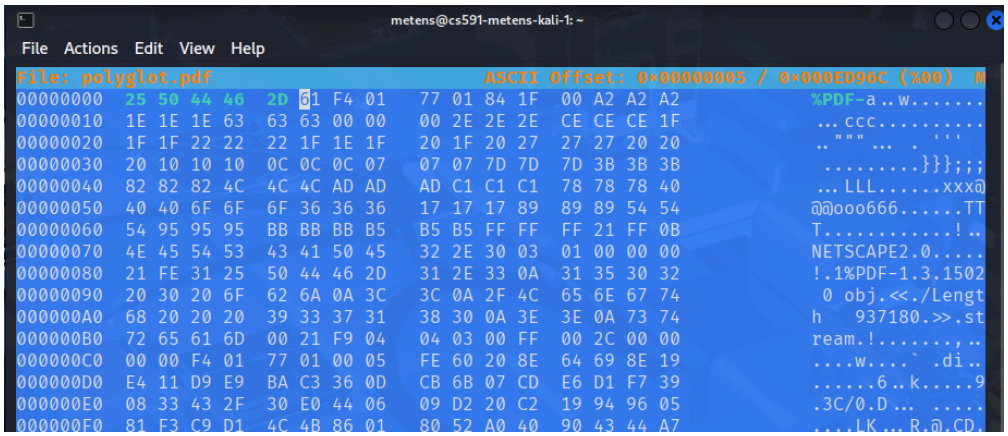
The web server uses a file extension heuristic to determine the content type of each file. The file that ends in PDF is considered a PDF, and the one with a GIF extension is considered an image by the server.

Task 10: file and polyglots

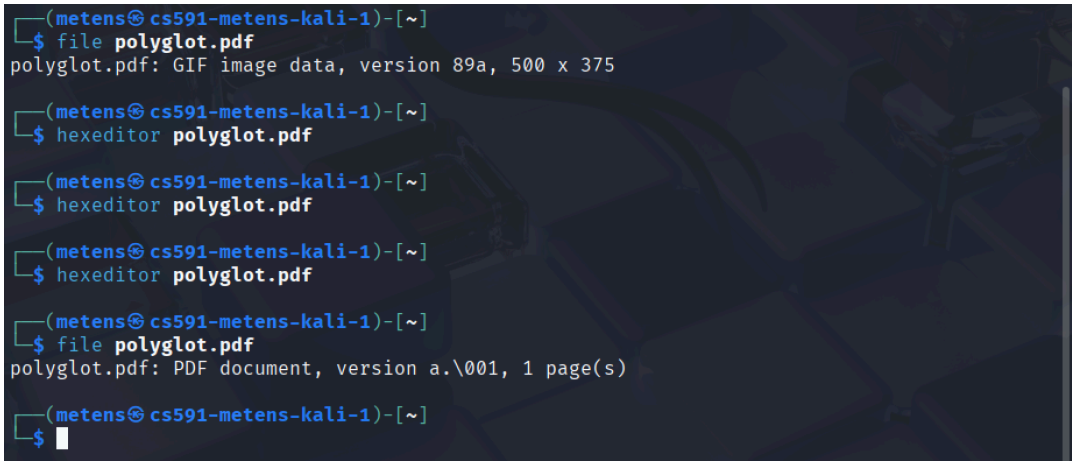
Used curl to grab the GIF image and save it to the Kali VM. Then used file xxd to dump the contents to see what type of file it is. Then, moved it to another name and used file to check the type, which uses content-based heuristics instead of file-extension heuristics to determine the file type, regardless of the file's name or extension.



Next, used hexeditor to change the filetype of the polyglot.gif into a PDF with the first 5 bytes of the hex data, which corresponds to the file type:



Then, after re-typing `file polyglot.pdf`, we can see that the type of the file has been changed:



Task 11: TryHackMe Linux file system forensics

Started by changing the path to a trusted USB source because an adversary could change commands to perform malicious behavior. Then, use find, exiftool, and stat to observe details about the attacker's steps. As well as learning about timestamps. This step took me about an hour; there was a lot of info, and it was quite cumbersome. The ‘Users and Groups’ part took me over 40 minutes to complete as well! I got stuck trying to figure out why sudo wasn’t working for me when trying to view Jane’s sudoers list. I re-read all the steps, tried searching the internet for help, navigating everywhere, finding Jane, and trying to sign in to her account. I even tried to restart the machine to see if there was a bug. I just didn’t know what the password was for the investigator account. But then I realized that since it had been over 2 hours since I had read the first step, I totally forgot that the password for the investigator account was right there: TryHackMe123!. So then I was finally able to finish the task and move on to the next one. The next step was about hidden files and investigating the hidden .ssh directory. I was able to find the flag in the hidden files by looking at their sizes. All of the hidden files had a size of zero,

except one of them, which contained the THM code to pass this step. Then I learned about rootkits, which are a way to get root privileges. To check for rootkits, we can use either rkhunter or chkrootkit.