Computer and Information Security

(ECE560, Fall 2024, Duke Univ., Prof. Tyler Bletsch)

Homework 3

Name: xxx   Duke NetID: xxx

Instructions - read all carefully:

- **COMPUTERS YOU WILL NEED:**
  - From the Duke VCM service:
    - VCM "**ECE 560 F23 Ubuntu 22.04**": this is your **Linux VM**.
    - VCM "**ECE 560 F23 Win10**": this is your **Windows VM**.
    - *New* VCM "**ECE 560 F23 Kali**": this is your **Kali Linux VM**.
  - Your own machine on Duke wifi: your **personal computer** (any OS). (If working remotely, VPN into the Duke network as needed.)
- **WRITTEN PORTION DIRECTIONS:**
  - This assignment is designed to be copied into a new document so you can answer questions inline (either as a Google doc or in a local word processor).
  - This assignment should be submitted as a **PDF through Gradescope**. Other formats or methods of submission will not be accepted.
  - When you submit, the tool will ask you to mark which pages contain which questions. This is easiest if you avoid having two questions on one page and keep the large question headers intact. Be sure to mark your answer pages appropriately. Staff reserves the right to penalize submissions that flagrantly fail to do this.
  - Follow directions carefully! The actions or items you need to submit are marked in cyan.
- **CITE YOUR SOURCES:** Make sure you document any resources you may use when answering the questions, including classmates and the textbook. Please use authoritative sources like RFCs, ISOs, NIST SPs, man pages, etc. for your references. Written answers should be in your own words. Long quotes or copy/paste from a source are not accepted because I want to hear from *you*.

This assignment is adapted from material by Samuel Carter (NCSU).

Question 0: Accessing the Homework (0 points, but necessary)

The Homework 3 pointer was a JPG/PDF polyglot, but if you're reading this, you've already figured this out. You can read more in PoC||GTFO 03:03 ("This PDF is a JPEG; or, This Proof of Concept is a Picture of Cats"). The final question of this assignment will require you to apply a similar technique to transform your submitted PDF.

Question 1: Chapter 3 - User Authentication (7 points)

Answer the questions below. (Based in part on review questions from the textbook)

a. In general terms, what are the three (or four) categories of ways to authenticate a user's identity?

   Something the individual knows
   Something the individual possesses
   Something the individual is (static biometrics)
   Something the individual does (dynamic biometrics)

b. List and briefly describe five possible threats to the secrecy of passwords.

   1. **Offline dictionary attack**: The attacker obtains the system password file and compares the password hashes against hashes of commonly used passwords. If a match is found, the attacker can gain access by that password combination.
   2. **Specific account attack**: The attacker targets a specific account and submits password guesses until the correct password is discovered.
   3. **Popular password attack**: Since user tends to choose a password that is easily remembered, this unfortunately makes the password easy to guess.
   4. **Password guessing against single user**: The attacker attempts to gain knowledge about the account holder and system password policies and uses that knowledge to guess the password.
   5. **Workstation hijacking**: The attacker waits until a logged-in workstation is unattended.

c. What are two enhancements *beyond* basic hashing to protect stored passwords?

   1. **Salting**: add a fixed-length, random and unique value to each password before hashing, and thus even if two users have the same password, the salt ensures that the resulting hash is different.
   2. **Key Stretching**: This involves applying the hashing function multiple times or using algorithms designed to be computationally intensive, such as **bcrypt**, **PBKDF2** that increase the time and effort required for an attacker to brute-force a password.

d. Which is more important: password complexity requirements or password expiration requirements? Why?

Password complexity requirements is more important. Complex passwords that are long and use a variety of characters are much more resistant to brute force attacks and dictionary attacks. In contrast, password expiration policies will lead to frequent change of the password and users tend to create weaker passwords or make only minor modifications to existing one [**NIST Password Guidelines 2024**]. If the modification is too weak, cybercriminals typically use compromised credentials immediately, so periodic password changes do little to prevent unauthorized access in most cases.

e. What is MFA? What threat model does it deal with?

   Multi-Factor Authentication (MFA) is a security method that requires users to provide two or more different authentication factors to verify their identity. Authentication factors include 1) something you know 2) something you have 3) something you are 4) something you do. The threat model deal with the threat model where 1) Asset: user's secrecy password, 2) Vulnerability: password might be cracked or obtained by the attacker and 3) Attacker's capabilities and knowledges: knows how to crack password.

f. Describe the general concept of a challenge-response protocol.

   In general, a challenge-response protocol is a function that when a user attempts to logon to a server, the server issues some sort of challenge that the user must respond to in order to be authenticated.

g. In a challenge-response password protocol, why is a random nonce used?

   The random nonce ensures that each authentication attempt is unique, even if the same password is used. In other words, even if a user authenticates multiple times with the same password, the challenges and responses will be different each time. This prevents an attacker from simply replaying a previously captured valid response to authenticate.

Question 2: File permissions (7 points)

From Problem 4.5: UNIX treats file directories in the same fashion as files; that is, both are defined by the same type of data structure, called an inode. As with files, directories include a nine-bit protection string. If care is not taken, this can create access control problems. For example, consider a file with the protection mode 644 (octal) contained in a directory with protection mode 750.

Explain the interpretation of the 9-bit protection string, for both files and directories. (1)

The 9-bit protection string for files: The first group of three bits refers to the **owner** of the file. The second group of three bits refers to the **group** that the file belongs to. The third group of three bits refers to **others** (everyone else). The three bits in each group represent
- r: read, this bit allows reading the contents of the file.

- w: write, this bit allows writing the contents of the file.
- x: execute, this bit allows running the file as a program.

For directories, the group is the same as the file. However, the representation of the three bits is slightly different
- r: read, this bit allows listing the files in the directory.
- w: write, this bit allows creating, deleting, and renaming files within the directory.
- x: execute, this bit allows accessing files within the directory.

Explain the meaning of the octal protection string of 644 for a file, and 750 for a directory. (2)

644: 6(110) 4(100) 4(100) means that the file can be read, written by the owner of the file, can be read by the group (of user) that the file belongs to, can be read by others.
750: 7(111) 5(101) 0(000) means that the owner user can list files in the director, can create, delete and renaming files within the directory and accessing files within the directory, Group user that the directory belongs to can list files in the directory, and access the files within the directory. Others do not have any permission of this directory.

Do an experiment: Make a directory with permissions 773. Can non-owner non-group users list the contents of the directory? Can such users create new files in the directory? (1)

As per the experiments shows that without the read (r) permission, non-owner, non-group users cannot list the directory contents. However, they can create new files in the directory as they have w (write) permission.

Consider the `ls` output below.

```
 -rw-r--r--  1 root     root    1.6M Sep 25 16:58 gosh.tar.gz
```

Explain in detail what each element this line means. (1)
The `-rw-r--r—` is file permission, which is divided into four sections:
1) first dash (`-`) means that this is a file
2) following nine characters are the permission bits, first set of three characters (`rw-`) means that the owner of the file has read, write permission of the file, second set of three characters (`r--`) means that the group has read permission of the file, third set of three characters (`r--`) means that others has read permission of the file.
3) `1` is the number of links, it tells you how many hard links point to this file, in this case, only one.
4) first `root` indicates that the file’s owner is root.
5) second `root` indicates that the file’s group is also root.
6) `1.6M` is the file size in bytes.

7) `Sep 25 16:58` means that the file is modified at September 25, at 16:58.
8) `gosh.tar.gz` is the name of the file.

What command would you use to change the access rights to allow any user to edit the file? (1)

```
chmod a+w gosh.tar.gz
```

What command would you use to take ownership of this file from root to yourself? (1)

```
sudo chown MYUSERNAME gosh.tar.gz
```

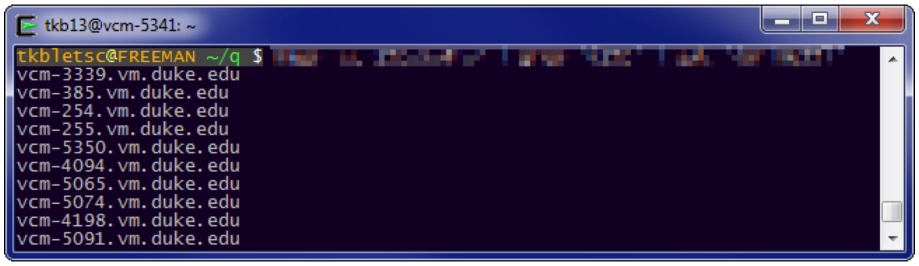
Question 3: Data processing (16 points)

In this question, you'll be manipulating some text data. Beyond the tools covered in class, you may want to look into:

- AWK, a general purpose computer language that is designed for processing text-based data, either in files or data streams. The name AWK is derived from the surnames of its authors Alfred V. Aho, Peter J. Weinberger, and Brian W. Kernighan.
- Perl is also a general purpose computer language focused on text-based data developed by Larry Wall.

Task 1: Filter nmap output (4 points)

Using any combination of the tools described in class or listed above, on your Linux VM, write a *single command* with pipes that will parse out only the registered hostnames from an Nmap List scan of the 152.3.64.* subnet. Your output should not include those hosts without a hostname or any other extraneous output. Give the full command and a small sample (5 hosts) of output. **Do not use -sT or -sS in the Nmap Scan**. A sample screenshot (with command blurred out of course) is shown below:



```
tkbletsc@FREEMAN ~/q $ [redacted]
vcm-3339.vm.duke.edu
vcm-385.vm.duke.edu
vcm-254.vm.duke.edu
vcm-255.vm.duke.edu
vcm-5350.vm.duke.edu
vcm-4094.vm.duke.edu
vcm-5065.vm.duke.edu
vcm-5074.vm.duke.edu
vcm-4198.vm.duke.edu
vcm-5091.vm.duke.edu
```

An attacker might use output like the above to better understand a target environment or to feed to another attack tool, especially after they've gained a foothold to an internal network.

```
Command: nmap -sL 152.3.64.* | grep -Eo 'Nmap scan report for [a-zA-
Z0-9.-]+\.vm\.duke\.edu' | awk '{print $5}'
```

```
Output:
vcm-27651.vm.duke.edu
vcm-27655.vm.duke.edu
vcm-27659.vm.duke.edu
vcm-26936.vm.duke.edu
vcm-22701.vm.duke.edu
vcm-21311.vm.duke.edu
```

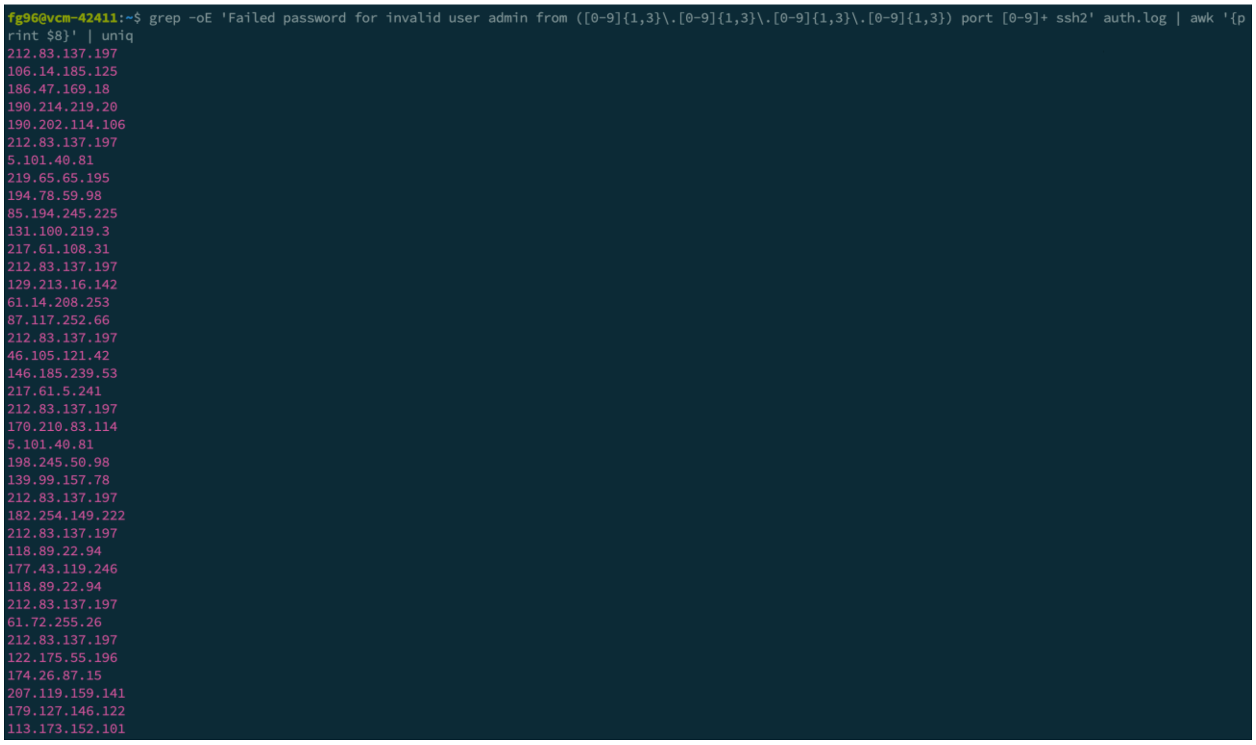
Task 2: Log analysis (4 points)

In Homework 1's question 2, we reviewed the output of Logwatch. That tool analyzes system logs such as the authorization log **auth.log**. Let's do a bit of similar analysis ourselves.

Using wget, obtain this auth.log from a real production server on the internet.

Write a single command that identifies all the *unique* IP addresses that tried and failed to login using the invalid user 'admin'. Post a screenshot.

```
grep -oE 'Failed password for invalid user admin from ([0-9]{1,3}\.[0-
9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}) port [0-9]+ ssh2' auth.log | awk
'{print $8}' | uniq
```



```
fg96@vcm-42411:~$ grep -oE 'Failed password for invalid user admin from ([0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}) port [0-9]+ ssh2' auth.log | awk '{p
rint $8}' | uniq
212.83.137.197
106.14.185.125
186.47.169.18
190.214.219.20
190.202.114.106
212.83.137.197
5.101.40.81
219.65.65.195
194.78.59.98
85.194.245.225
131.100.219.3
217.61.108.31
212.83.137.197
129.213.16.142
61.14.208.253
87.117.252.66
212.83.137.197
46.105.121.42
146.185.239.53
217.61.5.241
212.83.137.197
170.210.83.114
5.101.40.81
198.245.50.98
139.99.157.78
212.83.137.197
182.254.149.222
212.83.137.197
118.89.22.94
177.43.119.246
118.89.22.94
212.83.137.197
61.72.255.26
212.83.137.197
122.175.55.196
174.26.87.15
207.119.159.141
179.127.146.122
113.173.152.101
```

Task 3: Web app needle in a haystack (8 points)

A mock web application has been set up here:

http://target.colab.duke.edu/app/

The term “web application” here is a misnomer: for safety reasons, there’s no actual server-side code. Rather, there’s “lorem ipsum” content with hyperlinks that branch out to much of the same, with a few hidden things mixed in.

Using shell-based tools, identify the two pages that have an HTML <form> tag and give the sequence of clicks needed to find these pages from the start. Show your work.

```
base_url="http://target.colab.duke.edu/app"
base_url_https="https://target.colab.duke.edu/app"

declare -A visited

check_page() {
      local url=$1
      local path=$2

      # If the URL has been visited, skip it
      if [[ ${visited[$(echo -n "$url" | md5sum | awk '{print $1}')]} ]];
then
            return
      fi

      # Mark the URL as visited
      visited[$(echo -n "$url" | md5sum | awk '{print $1}')]=1

      local content=$(curl -s "$url")

      if echo "$content" | grep -q '<form'; then
            echo "<form> found in: $url, click sequence: $path"
      fi

      local links=$(echo "$content" | awk -F 'href=["'\'']' '{for (i=2;
i<=NF; i++) print $i}' | awk -F "'" '{print $1}' | awk -F '"' '{print
$1}')

      for link in $links; do
            if [[ $link != http* ]]; then
                  link="$base_url_https/$link"
            fi
            new_path="$path -> $link"
            check_page "$link" "$new_path"
      done
}

check_page "$base_url" "$base_url"
```

Result:

```
<form> found in: https://target.colab.duke.edu/app/X-
cad7214067aecc83edc49594504b0d9d.html, click sequence:
http://target.colab.duke.edu/app -> https://target.colab.duke.edu/app ->
https://target.colab.duke.edu/app/ -> https://target.colab.duke.edu/app/X-
fe4c7ea59925ac1d3d7fdf7554133457.html ->
https://target.colab.duke.edu/app/X-9d8b379a0cefec5dbd3cfbec570f99a1.html
-> https://target.colab.duke.edu/app/X-
0d13549d240d8d8ac49969b56659ce6d.html ->
https://target.colab.duke.edu/app/X-cad7214067aecc83edc49594504b0d9d.html
<form> found in: https://target.colab.duke.edu/app/X-
c54d004868ab19be1d57069415376c73.html, click sequence:
http://target.colab.duke.edu/app -> https://target.colab.duke.edu/app ->
https://target.colab.duke.edu/app/ -> https://target.colab.duke.edu/app/X-
4991b6c1b0037eab5e60c3284154fe1e.html ->
https://target.colab.duke.edu/app/X-c8acc24f331e1aae2dd00acd1cc838b4.html
-> https://target.colab.duke.edu/app/X-
c54d004868ab19be1d57069415376c73.html
```

OPTIONAL: For up to 4 points of extra credit, find:

1. A fat dog
2. A unicorn

As with the main question, give the page with the content and the click sequence needed to get there from the start. Show your work.

I change my script, the “search for keyword part” of the script

```
if echo "$content" | grep -q '<form'; then
           echo "<form> found in: $url, click sequence: $path"
fi
```

has been changed to search for image label in html, which is

```
if echo "$content" | grep -q '<img'; then
           echo "<img> found in: $url, click sequence: $path"
fi
```

Then I found the fat dog image in the following clicking sequence:

```
<img found in: https://target.colab.duke.edu/app/X-
14952159a0705846bc3e6d743175ca09.html, click sequence:
```

```
http://target.colab.duke.edu/app/ ->
https://target.colab.duke.edu/app/ ->
https://target.colab.duke.edu/app/X-
bfec97a36abea3f01d5d337bd89e54a6.html ->
https://target.colab.duke.edu/app/X-
8fa7c99f64643001a07ffc88c6eade0e.html ->
https://target.colab.duke.edu/app/X-
aac1657ce3b1a80de585f7f7f5e5acf1.html ->
https://target.colab.duke.edu/app/X-
14952159a0705846bc3e6d743175ca09.html
```

The second unicorn, I use ‘|’ as the search and found the drawing in the following clicks:

```
https://target.colab.duke.edu/app/X-
ad8868b064073b5e1c0e4fffdc029bab.html, click sequence:
http://target.colab.duke.edu/app/ ->
https://target.colab.duke.edu/app/ ->
https://target.colab.duke.edu/app/X-
909bd645fa5c8ca40b55f68d7291a139.html ->
https://target.colab.duke.edu/app/X-
0d1b79f2c19e940c370b59a629e93d37.html ->
https://target.colab.duke.edu/app/X-
1cbeffaab4af0249131327f1f7d75fe8.html ->
https://target.colab.duke.edu/app/X-
ad8868b064073b5e1c0e4fffdc029bab.html
```

```
                                                 ,/
                                                //
                                              ,//
                                  ___   /|   |//
                              `__/\_ --(/|___/-/
                           \|\_-\___ __-_`- /-/ \.
                          |\_-___,-\_____--/_)' ) \
                           \ -_ /     __ \( `( __`\|
                           `\__|      |\)\ ) /(/|
   ,._____.,            ',--//-|      \  |  '   /
  /     __. \,          / /,---|       \       /
 / /    _. \  \        `/`_/ _,'        |     |
|  | ( (  \   |      ,/\'__/'/          |     |
|  \  \`--, `_/_------______/           \(   )/
| | \  \_. \,                            \___/\
| |  \_   \  \                                 \
\ \    \_ \   \   /                             \
 \ \  \._  \__ \_|       |                       \
  \ \___  \      \       |                        \
   \__ \__ \  \_ |       \                         |
   |  \_____ \  ____      |                        |
   | \  \__ ---' .__\     |        |               |
   \  \__ ---   /   )     |        \              /
    \   \____/ / ()(      \          `---_       /|
     \__________/(,--__    \_________.    |    ./ |
       |     \ \  `---_\--,           \   \_,./   |
       |      \  \_ ` \    /`---_______-\   \\    /
        \      \.___,`|   /              \   \\   \
         \     |  \_ \|   \              (   |:    |
          \    \      \    |             /  / |    ;
           \    \      \    \          ( `_'   \  |
            \.    \      \.   \          `__/   |  |
              \   \       \.  \                |  |
               \   \        \  \               (  )
                \   |        \  |              |  |
                 |  \         \ \              I  `
                 ( __;        ( _;            ('-_';
                 |___\        \___:            \___:
```

Question 4: Regular expressions (9 points)

NetID validator (3 points)

Develop a regular expression that matches if and only if the input is a valid NetID (1-3 letters, 1 or more digits).

```
^[a-zA-Z]{1,3}\d+$
```

URL parser (3 points)

Develop a regular expression that, for a given URL, matches if and only if it is a Wikipedia page, and it captures just the article part of the URL. It should work for HTTP and HTTPS. Examples:

URL	Match?	Match group 1
http://en.wikipedia.org/wiki/Computer_security	yes	Computer_security
https://en.wikipedia.org/wiki/Transport_Layer_Security	yes	Transport_Layer_Security
https://duke.edu/	no	-

```
^https?:\/\/en\.wikipedia\.org\/wiki\/([A-Za-z0-9_]+)$
```

Pi digits (3 points)

Develop a shell command to print the MD5 hash of pi (π) up to and including the first appearance of decimal digits "12345". To help check your work, the last byte of the answer is 0x02. You can download the value of pi, no need to compute it. For actually employing the regex, you can use grep with -P and other options, perl, Python, or other tools. Note: newlines or other whitespace should be included in your hash.

```
curl -s https://assets.angio.net/100000.txt | grep -oE '.*12345' | tr
-d '\n' | md5sum
```

Question 5: Manual user management (6 points)

In order to better understand the raw underpinnings of user authentication, we're going to manipulate the Linux authentication system manually without the aid of the usual tools. One reason an attacker might do this is to reduce the probability of detection, as pre-fabricated tools like `adduser` might trigger log events and review by admins.

WARNING: we're going to be manually editing low level authentication files, and if you make a mistake, it can render your Linux VM inaccessible. Ensure you have no files you need on there, as you'll have to blow away and re-create the VM in this case.

First, research the format of the /etc/passwd and /etc/shadow files.

What's the purpose of `/etc/passwd`? What fields are stored in it?

The purpose of `/etc/passwd` is that it is a system file that contains essential information about the user accounts in a Linux or Unix system. It lists all the user accounts on the system, along with some basic details about each user.

From left to right, the fields are:
Each line in /etc/passwd is formatted as a colon-separated list. It involves:
1. **Username**: the username of the user
2. **Encrypted Password**: An "x" is typically placed here, meaning that the password is encrypted.
3. **User ID (UID)**: The user's numeric ID. Each user has a unique UID.
4. **Group ID (GID)**: The numeric ID of the user's primary group. Group information is found in /etc/group.
5. **User Information (Optional)**: An optional field used to store additional information about the user (like their full name, phone number, etc.).
6. **Home Directory**: The path to the user's home directory, where their personal files and configurations are stored (e.g., /home/user1).
7. **Shell**: The user's default login shell, e.g. /bin/bash. It determines which command-line interpreter is started when the user logs in.

What's the purpose of `/etc/shadow`? What fields are stored in it?

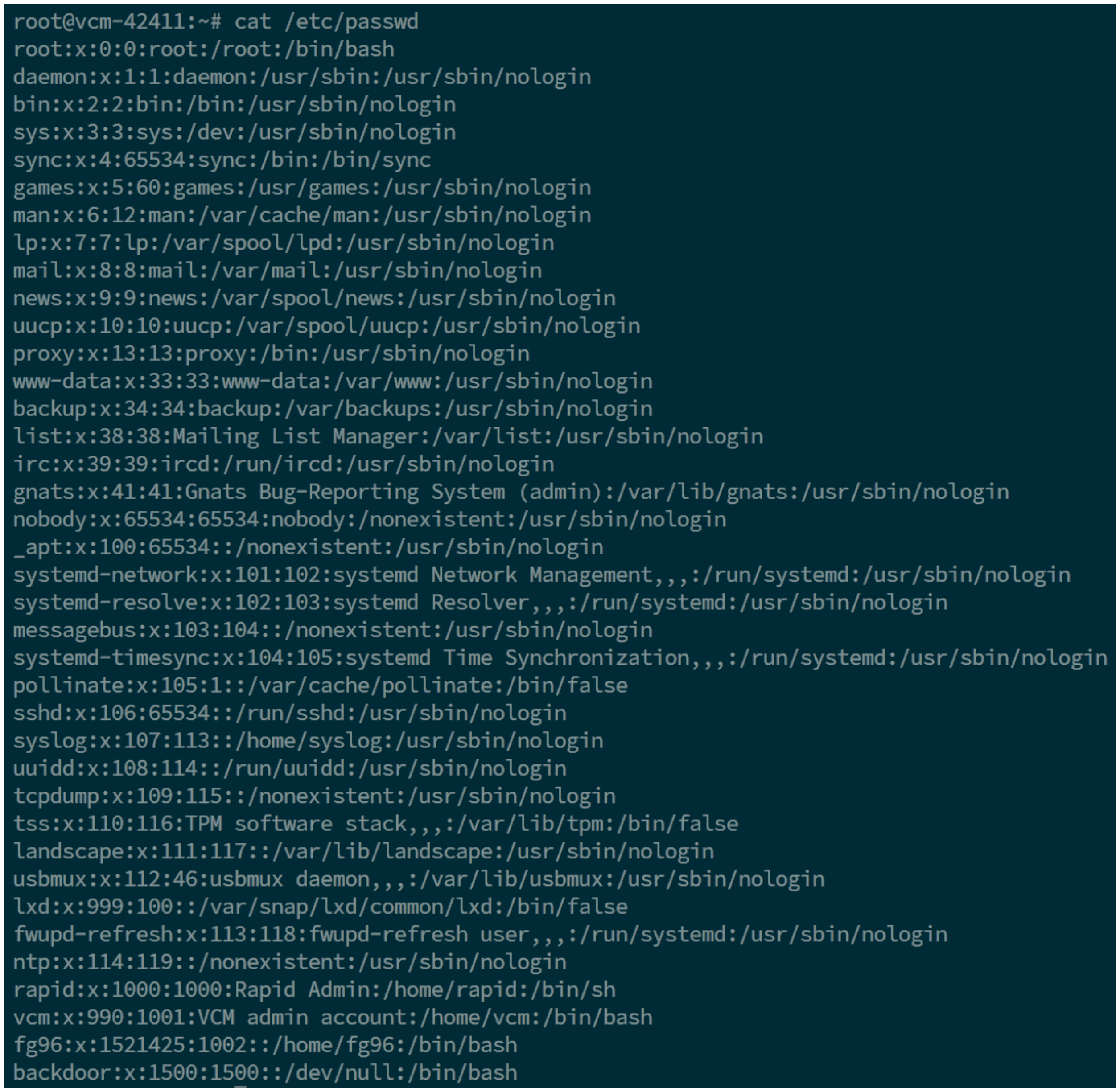
The /etc/shadow file stores encrypted password hashes and related information about user account passwords.
From left to right, the fields are:
1. **Username**: The login name of the user, matching the name in /etc/passwd.
2. **Encrypted Password**: This field contains the hashed password. If this field contains a placeholder (like ! or *), it indicates that the account is locked or doesn't have a password.
3. **Last Password Change**: The date of the last password change, stored as the number of days since Unix epoch.
4. **Minimum Days**: The minimum number of days required between password changes. If set to 0, the user can change their password at any time.
5. **Maximum Days**: The maximum number of days a password is valid before the user is forced to change it.
6. **Warning Days**: The number of days before the password expiration when the user will start receiving warnings to change their password.
7. **Inactive Days**: The number of days of inactivity after the password expires before the account is locked.

8. **Expiration Date**: The date when the account will be disabled, stored as the number of days since the Unix epoch. If this field is empty, the account never expires.
9. **Reserved Field**: Reserved for future use.

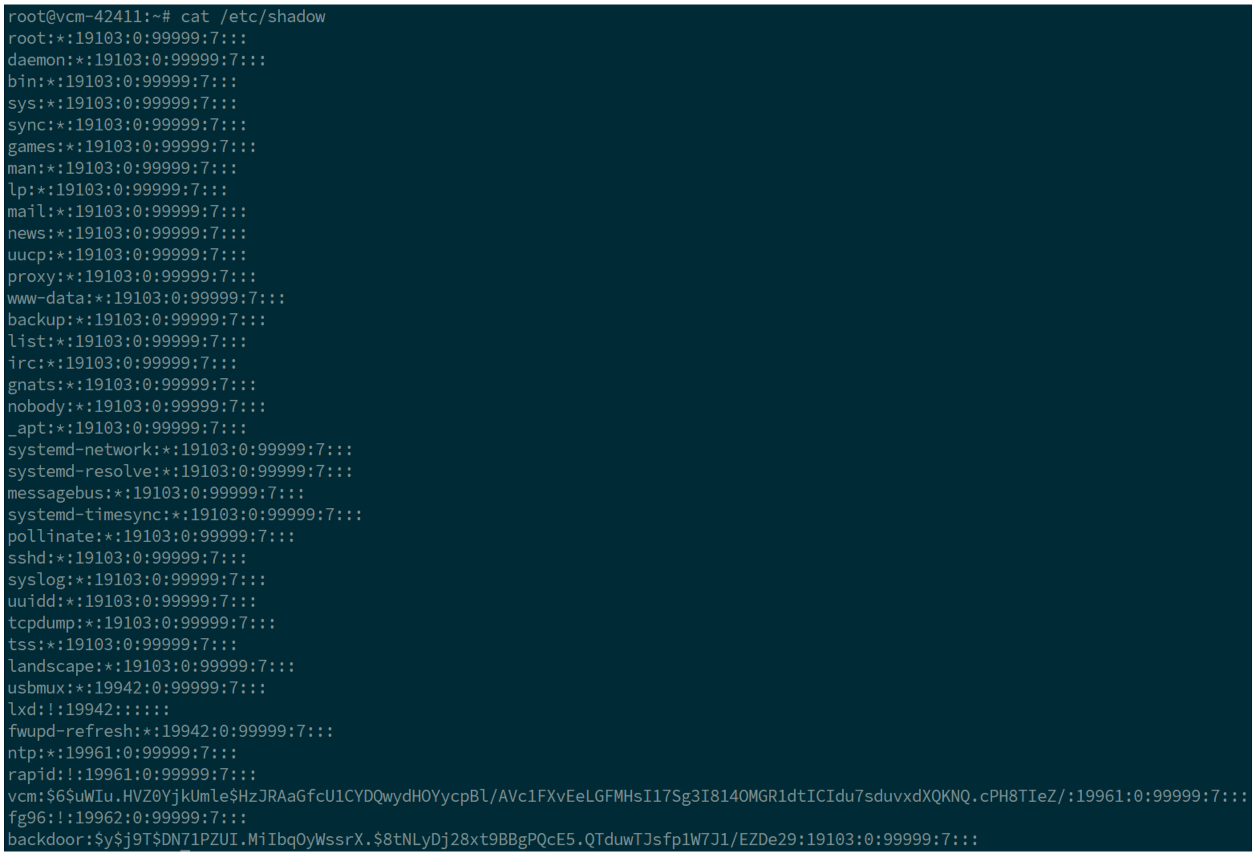
Second, let's add a user by hand. Use sudo to become root (e.g. `sudo -i`). For all steps that say "as root", you'll use this session. Note: it's wise to keep a spare terminal window logged in as root in the background, so if you screw something up with logins, you can fix it without trashing the VM.

In order to add a user, we'll need a password hash suitable for use on Linux. On your *Kali VM*, the tool `mkpasswd` is pre-installed. Run it, and type in a reasonably strong unique password you can remember. The output will be a suitable salted hash.

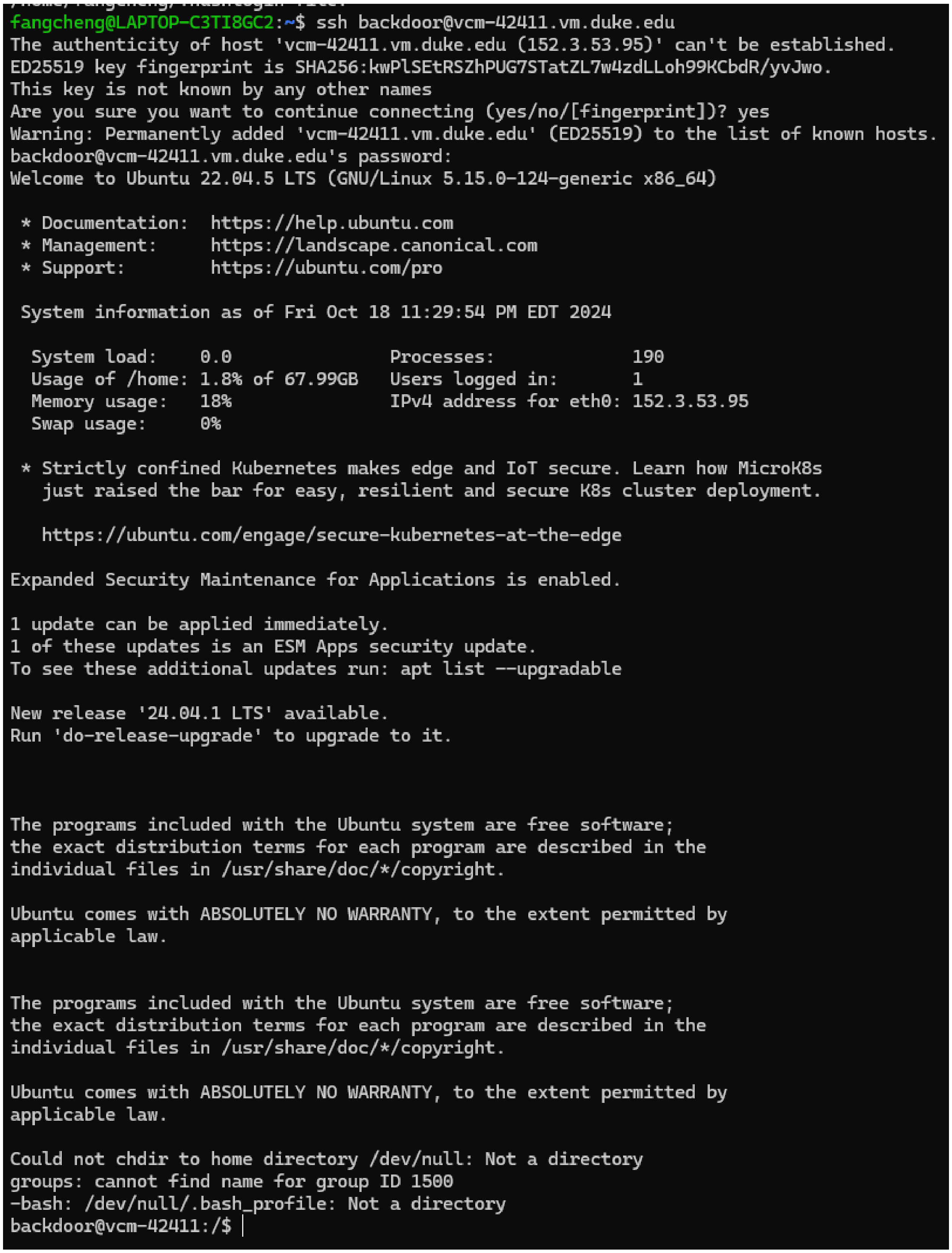
As root on your Linux VM, edit `/etc/passwd` and `/etc/shadow` to add a new user called `backdoor`. For UID, pick any unused number above 1000, and pick reasonable values for the other fields (experimenting as needed). For the password hash, use the one generated above. You should now be able to SSH into your VM as the `backdoor` user with the chosen password. Show your changes to `/etc/passwd` and `/etc/shadow` as well as the successful login.



```
root@vcm-42411:~# cat /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:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
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:/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
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-network:x:101:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:104::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:104:105:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
pollinate:x:105:1::/var/cache/pollinate:/bin/false
sshd:x:106:65534::/run/sshd:/usr/sbin/nologin
syslog:x:107:113::/home/syslog:/usr/sbin/nologin
uuidd:x:108:114::/run/uuidd:/usr/sbin/nologin
tcpdump:x:109:115::/nonexistent:/usr/sbin/nologin
tss:x:110:116:TPM software stack,,,:/var/lib/tpm:/bin/false
landscape:x:111:117::/var/lib/landscape:/usr/sbin/nologin
usbmux:x:112:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
lxd:x:999:100::/var/snap/lxd/common/lxd:/bin/false
fwupd-refresh:x:113:118:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
ntp:x:114:119::/nonexistent:/usr/sbin/nologin
rapid:x:1000:1000:Rapid Admin:/home/rapid:/bin/sh
vcm:x:990:1001:VCM admin account:/home/vcm:/bin/bash
fg96:x:1521425:1002::/home/fg96:/bin/bash
backdoor:x:1500:1500::/dev/null:/bin/bash
```



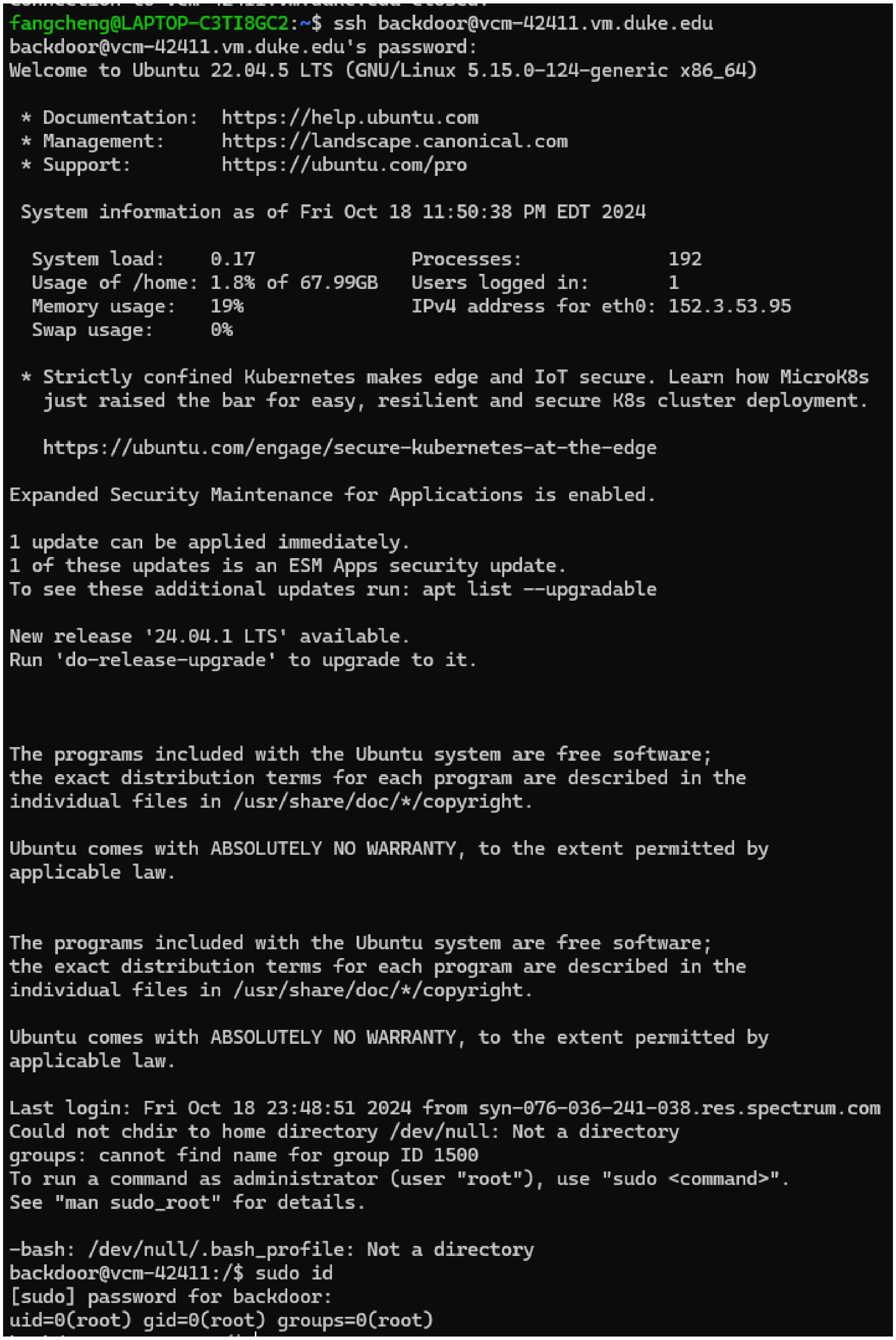
```
root@vcm-42411:~# cat /etc/shadow
root:*:19103:0:99999:7:::
daemon:*:19103:0:99999:7:::
bin:*:19103:0:99999:7:::
sys:*:19103:0:99999:7:::
sync:*:19103:0:99999:7:::
games:*:19103:0:99999:7:::
man:*:19103:0:99999:7:::
lp:*:19103:0:99999:7:::
mail:*:19103:0:99999:7:::
news:*:19103:0:99999:7:::
uucp:*:19103:0:99999:7:::
proxy:*:19103:0:99999:7:::
www-data:*:19103:0:99999:7:::
backup:*:19103:0:99999:7:::
list:*:19103:0:99999:7:::
irc:*:19103:0:99999:7:::
gnats:*:19103:0:99999:7:::
nobody:*:19103:0:99999:7:::
_apt:*:19103:0:99999:7:::
systemd-network:*:19103:0:99999:7:::
systemd-resolve:*:19103:0:99999:7:::
messagebus:*:19103:0:99999:7:::
systemd-timesync:*:19103:0:99999:7:::
pollinate:*:19103:0:99999:7:::
sshd:*:19103:0:99999:7:::
syslog:*:19103:0:99999:7:::
uuidd:*:19103:0:99999:7:::
tcpdump:*:19103:0:99999:7:::
tss:*:19103:0:99999:7:::
landscape:*:19103:0:99999:7:::
usbmux:*:19942:0:99999:7:::
lxd:!:19942::::::
fwupd-refresh:*:19942:0:99999:7:::
ntp:*:19961:0:99999:7:::
rapid:!:19961:0:99999:7:::
vcm:$6$uWIu.HVZ0YjkUmle$HzJRAaGfcU1CYDQwydHOYycpBl/AVc1FXvEeLGFMHsI17Sg3I814OMGR1dtICIdu7sduvxdXQKNQ.cPH8TIeZ/:19961:0:99999:7:::
fg96:!:19962:0:99999:7:::
backdoor:$y$j9T$DN71PZUI.MiIbqOyWssrX.$8tNLyDj28xt9BBgPQcE5.QTduwTJsfp1W7J1/EZDe29:19103:0:99999:7:::
```



```
fangcheng@LAPTOP-C3TI8GC2:~$ ssh backdoor@vcm-42411.vm.duke.edu
The authenticity of host 'vcm-42411.vm.duke.edu (152.3.53.95)' can't be established.
ED25519 key fingerprint is SHA256:kwPlSEtRSZhPUG7STatZL7w4zdLLoh99KCbdR/yvJwo.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'vcm-42411.vm.duke.edu' (ED25519) to the list of known hosts.
backdoor@vcm-42411.vm.duke.edu's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-124-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Fri Oct 18 11:29:54 PM EDT 2024

  System load:    0.0               Processes:             190
  Usage of /home: 1.8% of 67.99GB   Users logged in:       1
  Memory usage:   18%               IPv4 address for eth0: 152.3.53.95
  Swap usage:     0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is enabled.

1 update can be applied immediately.
1 of these updates is an ESM Apps security update.
To see these additional updates run: apt list --upgradable

New release '24.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Could not chdir to home directory /dev/null: Not a directory
groups: cannot find name for group ID 1500
-bash: /dev/null/.bash_profile: Not a directory
backdoor@vcm-42411:/$
```

You’ve now added a new user account. It doesn’t have a real home directory, but that doesn’t matter to an attacker who’s more interested in sustained access than storing documents. Currently, however, this user is unprivileged, unable to use sudo or editing much of anything.

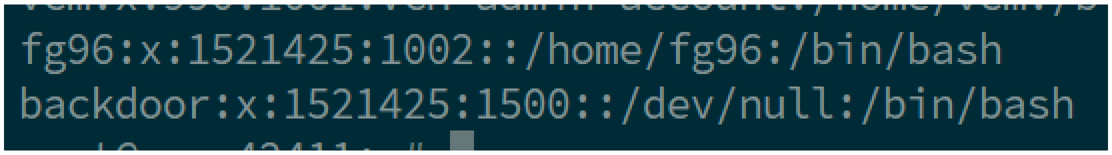
We’re going to take two separate approaches to adding privileges to the `backdoor` user. As root on your Linux VM, let’s add the user to the sudo group, which gives it sudo permissions (so the `backdoor` can become root). This is quite easy -- using your original root termal, edit `/etc/group`, and add `backdoor` at the end of the “`sudo:`” line. Open a new SSH connection as `backdoor`, and verify you can use sudo with “sudo id”.



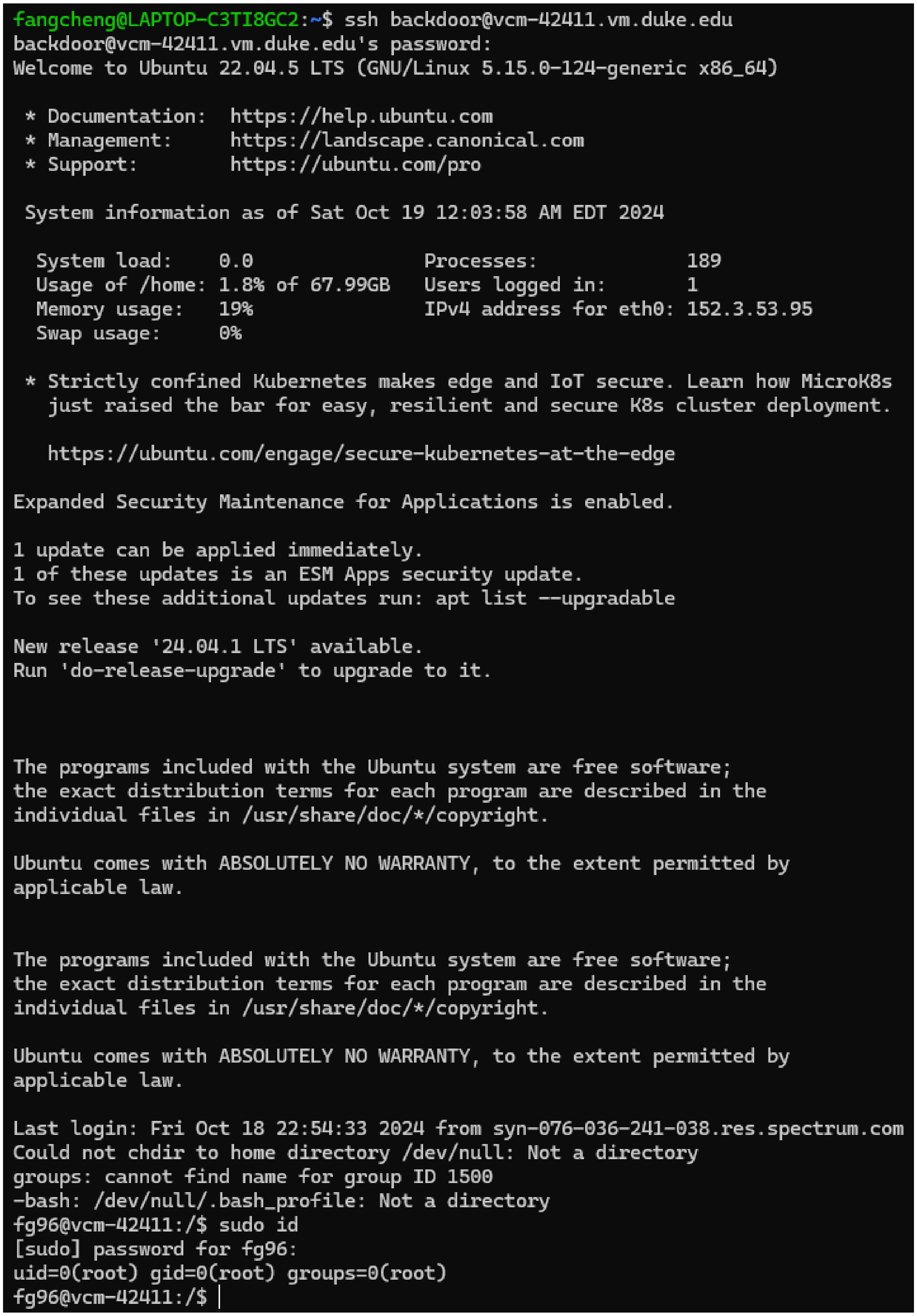
```
fangcheng@LAPTOP-C3TI8GC2:~$ ssh backdoor@vcm-42411.vm.duke.edu
backdoor@vcm-42411.vm.duke.edu's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-124-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Fri Oct 18 11:50:38 PM EDT 2024

  System load:    0.17              Processes:             192
  Usage of /home: 1.8% of 67.99GB   Users logged in:       1
  Memory usage:   19%               IPv4 address for eth0: 152.3.53.95
  Swap usage:     0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is enabled.

1 update can be applied immediately.
1 of these updates is an ESM Apps security update.
To see these additional updates run: apt list --upgradable

New release '24.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Last login: Fri Oct 18 23:48:51 2024 from syn-076-036-241-038.res.spectrum.com
Could not chdir to home directory /dev/null: Not a directory
groups: cannot find name for group ID 1500
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

-bash: /dev/null/.bash_profile: Not a directory
backdoor@vcm-42411:/$ sudo id
[sudo] password for backdoor:
uid=0(root) gid=0(root) groups=0(root)
```

There's a more subtle way to gain access, however. To test this, first undo the above by removing `backdoor` from the sudo group, and verify that `backdoor` no longer has sudo rights.

Normally, there's a 1:1 mapping between usernames and UID numbers, but we're going to intentionally create a UID collision. As root on your Linux VM, edit `/etc/passwd`, and change the UID of the `backdoor` user to the same number as your NetID (also shown in the passwd file). Ensure that the line for the `backdoor` user appears *after* your NetID in the passwd file. Then, in a new terminal, SSH into the Linux VM as `backdoor` again. If you've done it right, you should find yourself suddenly logged in under your NetID account! Show the edited passwd file (with both your NetID and `backdoor` entries), as well as a successful SSH connection where you logged in as `backdoor` but became your NetID. Verify that you can use sudo just as your normal NetID login can.



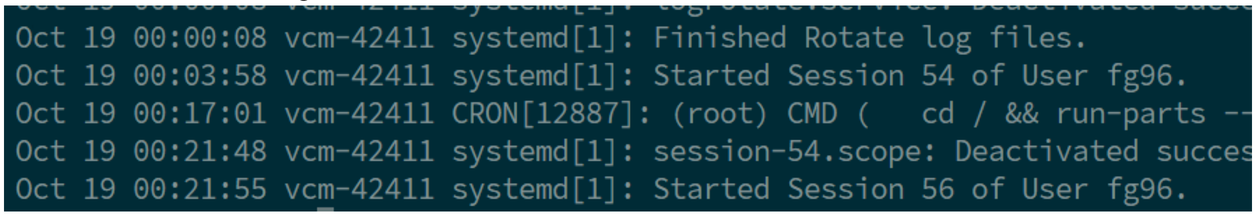
```
fg96:x:1521425:1002::/home/fg96:/bin/bash
backdoor:x:1521425:1500::/dev/null:/bin/bash
```



```
fangcheng@LAPTOP-C3TI8GC2:~$ ssh backdoor@vcm-42411.vm.duke.edu
backdoor@vcm-42411.vm.duke.edu's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-124-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sat Oct 19 12:03:58 AM EDT 2024

  System load:    0.0               Processes:             189
  Usage of /home: 1.8% of 67.99GB   Users logged in:       1
  Memory usage:   19%               IPv4 address for eth0: 152.3.53.95
  Swap usage:     0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is enabled.

1 update can be applied immediately.
1 of these updates is an ESM Apps security update.
To see these additional updates run: apt list --upgradable

New release '24.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Last login: Fri Oct 18 22:54:33 2024 from syn-076-036-241-038.res.spectrum.com
Could not chdir to home directory /dev/null: Not a directory
groups: cannot find name for group ID 1500
-bash: /dev/null/.bash_profile: Not a directory
fg96@vcm-42411:/$ sudo id
[sudo] password for fg96:
uid=0(root) gid=0(root) groups=0(root)
fg96@vcm-42411:/$
```

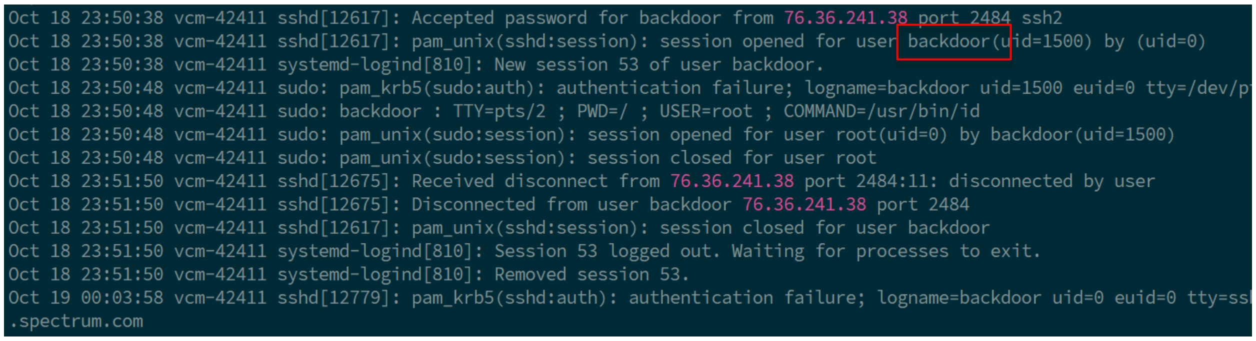
Now we’ve set up an alternate means of logging into the VM as you, but without your password, and we did it without triggering any log file entries about the change. Let’s see what log entries are created with regard to login. As root on your Linux VM, observe the content of /var/log/syslog and /var/log/auth.log during an SSH login as `backdoor`. If an administrator only looked at syslog, could they tell that a `backdoor` user was involved in the login?

The syslog will only tell the UID of the user login, while the auth.log will provide the more info about the username login



```
Oct 19 00:00:08 vcm-42411 systemd[1]: Finished Rotate log files.
Oct 19 00:03:58 vcm-42411 systemd[1]: Started Session 54 of User fg96.
Oct 19 00:17:01 vcm-42411 CRON[12887]: (root) CMD (   cd / && run-parts --
Oct 19 00:21:48 vcm-42411 systemd[1]: session-54.scope: Deactivated succes
Oct 19 00:21:55 vcm-42411 systemd[1]: Started Session 56 of User fg96.
```

/var/log/syslog



```
Oct 18 23:50:38 vcm-42411 sshd[12617]: Accepted password for backdoor from 76.36.241.38 port 2484 ssh2
Oct 18 23:50:38 vcm-42411 sshd[12617]: pam_unix(sshd:session): session opened for user backdoor(uid=1500) by (uid=0)
Oct 18 23:50:38 vcm-42411 systemd-logind[810]: New session 53 of user backdoor.
Oct 18 23:50:48 vcm-42411 sudo: pam_krb5(sudo:auth): authentication failure; logname=backdoor uid=1500 euid=0 tty=/dev/p
Oct 18 23:50:48 vcm-42411 sudo: backdoor : TTY=pts/2 ; PWD=/ ; USER=root ; COMMAND=/usr/bin/id
Oct 18 23:50:48 vcm-42411 sudo: pam_unix(sudo:session): session opened for user root(uid=0) by backdoor(uid=1500)
Oct 18 23:50:48 vcm-42411 sudo: pam_unix(sudo:session): session closed for user root
Oct 18 23:51:50 vcm-42411 sshd[12675]: Received disconnect from 76.36.241.38 port 2484:11: disconnected by user
Oct 18 23:51:50 vcm-42411 sshd[12675]: Disconnected from user backdoor 76.36.241.38 port 2484
Oct 18 23:51:50 vcm-42411 sshd[12617]: pam_unix(sshd:session): session closed for user backdoor
Oct 18 23:51:50 vcm-42411 systemd-logind[810]: Session 53 logged out. Waiting for processes to exit.
Oct 18 23:51:50 vcm-42411 systemd-logind[810]: Removed session 53.
Oct 19 00:03:58 vcm-42411 sshd[12779]: pam_krb5(sshd:auth): authentication failure; logname=backdoor uid=0 euid=0 tty=ssl
.spectrum.com
```

/var/log/auth.log

For safety, undo all changes to `/etc/passwd` and `/etc/shadow` when done.

Question 6: GNU Screen (2 points)

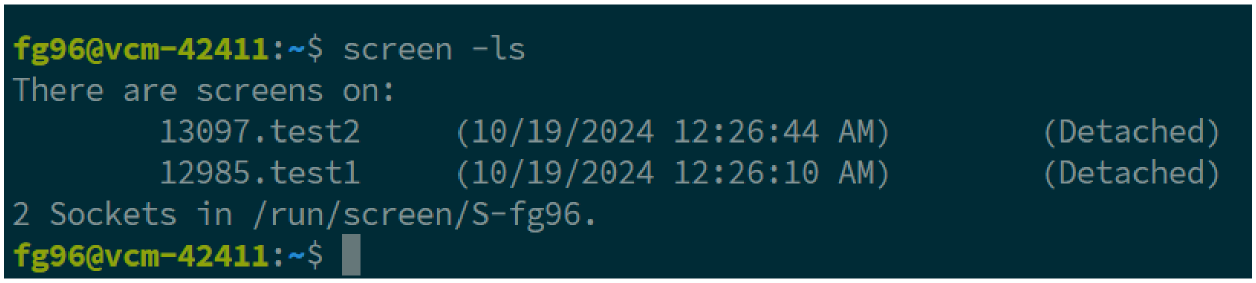
Screen is a full-screen window manager that multiplexes a physical terminal (or GUI terminal window) between several processes, typically interactive shells. When screen is called, it creates a single terminal with a shell in it (or the specified command) and then gets out of your way so that you can use the program as you normally would. Then, at any time, you can create new (full-screen) terminals with other programs in them (including more shells), kill the current window, view a list of the active windows, turn output logging on and off, view the scrollback history, switch between terminals, etc. All terminals run their programs completely independent of each other. Especially useful for us is the fact you can *detach* a running screen session from your console and let all the associated terminals keep running in the background, then later *reattach* it to another console. Programs continue to run even when screen is detached.

Review this very brief video intro to screen. Here is a screen quick reference.

This might be useful for the MD5 cracking question below, and it's essential for the problem "John the Ripper" after that. Start a new screen session on your Kali VM. Prove to yourself that you can detach, reattach, and are generally comfortable with screen.

Note: If you prefer another terminal virtualizer, such as tmux, you may use that instead.

Paste a screenshot below of `screen --list` two separate screen sessions running.



```
fg96@vcm-42411:~$ screen -ls
There are screens on:
        13097.test2     (10/19/2024 12:26:44 AM)        (Detached)
        12985.test1     (10/19/2024 12:26:10 AM)        (Detached)
2 Sockets in /run/screen/S-fg96.
fg96@vcm-42411:~$
```

Question 7: Cracking MD5 hashes with hashcat (5 points)

There is an old FTP server called Serv-U[1], a commercial program originally written by CATsoft. In the past, hackers would sometimes install pirated copies on computers they compromise. This allows an attacker to access the file system and run commands on the compromised computer. The FTP server usernames and passwords (used by the attackers) are stored in a configuration file. The passwords are hashed with MD5. Here is a sample file from a compromised machine:

```
[USER=admin|1]
Password=ly5cf2ff593419bcf3d22c62e65a82128a
HomeDir=c:\
AlwaysAllowLogin=1
TimeOut=600
Maintenance=System
Access1=C:\|RWAMELCDP
```

The MD5 hash of the password is on the line that starts with 'Password='. The two letters after the equal sign are the salt as ASCII text (shown in orange above) followed by the MD5 hash in hex (shown in blue above). This salt is prepended, i.e. the Serv-U password hash function is **`md5(salt+pass)`**.

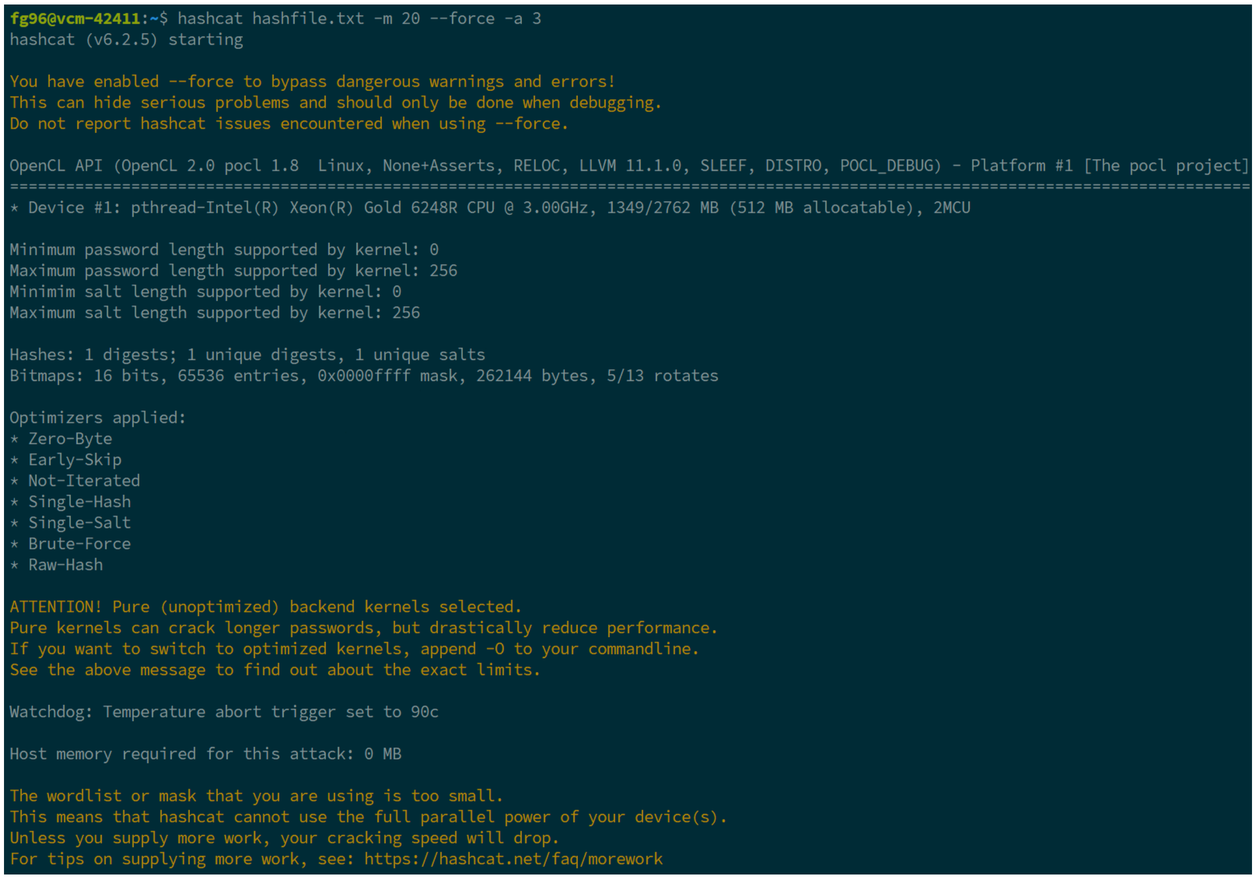
[1] A bit of trivia: the Serv-U FTP server recently and unexpectedly came back into the news. It turns out it got bought out by IT management firm SolarWinds years ago, and used as part of their full solution. A vulnerability with severity score 10.0 (the highest possible) was reported, and was used as part of a major compromise of SolarWinds and their customers. All of this is totally unrelated to this question, but it is insane to me that it's still around and still being involved in system compromises. Amazing.

Your Kali VM comes with a hash cracker called **hashcat**. Read up on how to use it, and have it crack the salted hash above. Tips:

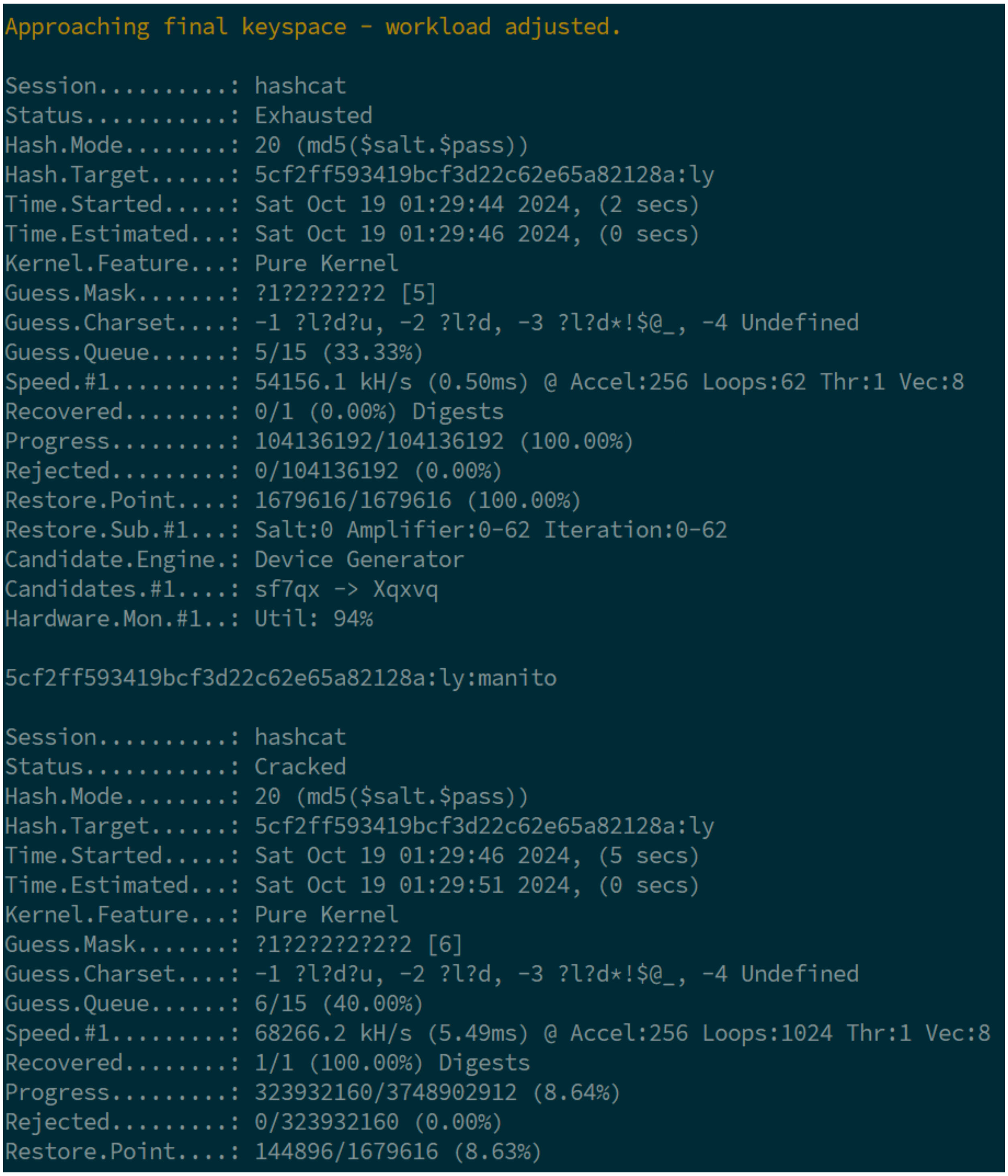
- Hashcat is optimized for GPUs, but your VM doesn't have one. Override the resulting error message using the `--force` flag.
- You will need to create a "hash file" as input; the format of this file is "**`<hash>:<salt>`**".
- The default brute force mode will be fine, and should take around 30 seconds on the Kali VM.
- Found passwords are echoed in **`<hash>:<salt>:<pass>`** format and also saved to **`~/.hashcat/hashcat.potfile`**.

Paste (1) the command executed, (2) a screenshot of the output, and (3) the password itself below.

Command: `hashcat hashfile.txt -m 20 --force -a 3`



```
fg96@vcm-42411:~$ hashcat hashfile.txt -m 20 --force -a 3
hashcat (v6.2.5) starting

You have enabled --force to bypass dangerous warnings and errors!
This can hide serious problems and should only be done when debugging.
Do not report hashcat issues encountered when using --force.

OpenCL API (OpenCL 2.0 pocl 1.8  Linux, None+Asserts, RELOC, LLVM 11.1.0, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
=====================================================================================================================================
* Device #1: pthread-Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz, 1349/2762 MB (512 MB allocatable), 2MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256
Minimim salt length supported by kernel: 0
Maximum salt length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates

Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Iterated
* Single-Hash
* Single-Salt
* Brute-Force
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 0 MB

The wordlist or mask that you are using is too small.
This means that hashcat cannot use the full parallel power of your device(s).
Unless you supply more work, your cracking speed will drop.
For tips on supplying more work, see: https://hashcat.net/faq/morework
```



```
Approaching final keyspace - workload adjusted.

Session..........: hashcat
Status...........: Exhausted
Hash.Mode........: 20 (md5($salt.$pass))
Hash.Target......: 5cf2ff593419bcf3d22c62e65a82128a:ly
Time.Started.....: Sat Oct 19 01:29:44 2024, (2 secs)
Time.Estimated...: Sat Oct 19 01:29:46 2024, (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: ?1?2?2?2?2 [5]
Guess.Charset....: -1 ?l?d?u, -2 ?l?d, -3 ?l?d*!$@_, -4 Undefined
Guess.Queue......: 5/15 (33.33%)
Speed.#1.........: 54156.1 kH/s (0.50ms) @ Accel:256 Loops:62 Thr:1 Vec:8
Recovered........: 0/1 (0.00%) Digests
Progress.........: 104136192/104136192 (100.00%)
Rejected.........: 0/104136192 (0.00%)
Restore.Point....: 1679616/1679616 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-62 Iteration:0-62
Candidate.Engine.: Device Generator
Candidates.#1....: sf7qx -> Xqxvq
Hardware.Mon.#1..: Util: 94%

5cf2ff593419bcf3d22c62e65a82128a:ly:manito

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 20 (md5($salt.$pass))
Hash.Target......: 5cf2ff593419bcf3d22c62e65a82128a:ly
Time.Started.....: Sat Oct 19 01:29:46 2024, (5 secs)
Time.Estimated...: Sat Oct 19 01:29:51 2024, (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: ?1?2?2?2?2?2 [6]
Guess.Charset....: -1 ?l?d?u, -2 ?l?d, -3 ?l?d*!$@_, -4 Undefined
Guess.Queue......: 6/15 (40.00%)
Speed.#1.........: 68266.2 kH/s (5.49ms) @ Accel:256 Loops:1024 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests
Progress.........: 323932160/3748902912 (8.64%)
Rejected.........: 0/323932160 (0.00%)
Restore.Point....: 144896/1679616 (8.63%)
```

Password: manito

OPTIONAL: For up to 6 points of extra credit, crack each of the following Serv-U hashes.

You may need to use more time, wordlists, and possibly even a combination wordlist/brute-force attack.

a. **`qje39945197dbc380a5ac611fbbb7b5354`**
b. **`csa2eb704ec429728c1fe15814c006a4cc`**
c. **`qj847d6d5952baef48fa101d84a18b8de8`**

a. baseball
b. csh4xoRzz

Question 8: John the Ripper (6 points)

Note: before doing this question, you should understand the `/etc/passwd` and `/etc/shadow` files, as introduced in Question 5.

For this exercise, we will use John the Ripper, which is already installed on your Kali VM. I'll be supplying the shadow file for the exercise.

We will be using the shadow file here:
http://people.duke.edu/~tkb13/courses/ece560/homework/hw3/shadow

Download this shadow file to your Kali VM. Unlike hashcat, john has pretty smart defaults and wordlists, so you can just run it against the shadow file. To improve performance, you can specify **`--fork=<N>`** to run *N* instances in parallel; set *N* to the number of CPU cores on your system (you can check with `top`, `cat /proc/cpuinfo`, etc.).

Run `john` on the provided shadow file in a `screen` session (so you can detach, disconnect, then reattach later). Let it run for at least 24 hours. It is not expected to be able to crack all the passwords, but you should get at least 6.

Paste the output of your results with the following command "`john -show shadow`". (6 points)

```
root:toor:10063:0:99999:7:::
idiot:idiot:10063:0:99999:7:::
al:1einstein:10063:0:99999:7:::
cindy:crawford:10063:0:99999:7:::
dieter:curiake:10063:0:99999:7:::
dean:astalis:10063:0:99999:7:::
sgtpepper:ringo:10063:0:99999:7:::

7 password hashes cracked, 4 left
```

It should look something like the example below:

```
$ john -show shadow
abcuser:apple1:15358:0:99999:7:::
defuser:orange:15364:0:99999:7:::
ghiuser:gpgtest:15373:0:99999:7:::

3 password hashes cracked, 25 left
```

Note: Here, “abcuser” is the username and “apple1” is the password.

Question 9: Evaluating MFA approaches (8 points)

Review this article on a data theft attack against content aggregator site Reddit and answer the following.

a. What form of Multi-Factor Authentication (MFA) was deployed at Reddit?

   Reddit uses SMS-based MFA as their second factor of authentication for their employees.

b. What are two ways in which the attacker may have gained access to the second factor?

   An attacker might gain access to the second factor by
   1. **SIM-swapping** where attacker make the victim’s phone carrier to tie the customer’s service to attacker’s sim card and thus receive the SMS message with the MFA code.
   2. **SS7 protocol attacks** where vulnerabilities in the global telecom network are exploited to intercept SMS messages.

c. What are two alternative forms of MFA that could be deployed instead?

   1. **App-based authentication**, such as Google Authenticator or Microsoft Authenticator, which generates time-based one-time passwords (TOTP) on the user’s device.
   2. **Hardware security keys** such as YubiKey, which use the FIDO2 standard and are phishing-resistant.

d. Identify another attack in the last 5 years in which SMS-based MFA was compromised. Describe what happened, and provide one or more citations of where you found this info.

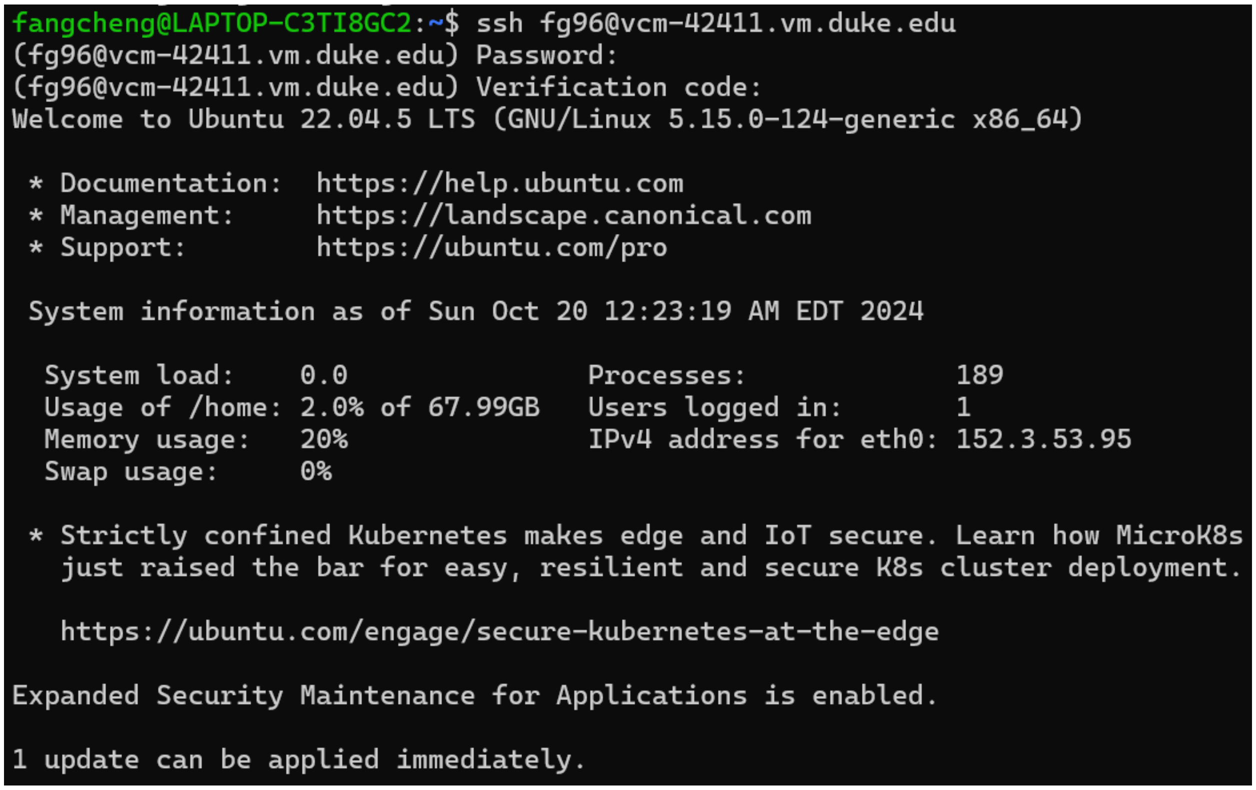
   One attack in the last 5 years where SMS-based MFA was compromised occurred in April 2024, involving Cisco Duo, a major multi-factor authentication provider. The attack took place on April 1, 2024. Hackers obtained employee credentials of the telephony provider through a phishing attack. The intruders downloaded SMS and VoIP MFA message logs associated with specific Duo accounts. Since Duo uses to send multifactor authentication (MFA) messages via SMS and VOIP to its customers, the breach of the log will help attackers to launch targeted phishing attacks to gain access to sensitive information, such as corporate credentials [Cisco Duo warns third-party data breach exposed SMS MFA logs].

Question 10: SSH with MFA (5 points)

On your Linux VM, you are going to add some extra protections to the SSH server by adding Multi-Factor Authentication (MFA) to your account.

Some research will tell you how to do this using Google Authenticator or the desktop app Authy.

Show a screenshot of your SSH login with MFA code.



```
fangcheng@LAPTOP-C3TI8GC2:~$ ssh fg96@vcm-42411.vm.duke.edu
(fg96@vcm-42411.vm.duke.edu) Password:
(fg96@vcm-42411.vm.duke.edu) Verification code:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-124-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sun Oct 20 12:23:19 AM EDT 2024

  System load:    0.0               Processes:             189
  Usage of /home: 2.0% of 67.99GB   Users logged in:       1
  Memory usage:   20%               IPv4 address for eth0: 152.3.53.95
  Swap usage:     0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is enabled.

1 update can be applied immediately.
```

When done, you may remove this MFA support if you wish.

Question 11: Hydra (Online SSH Dictionary Attack) (5 points)

Hydra is an online network-based dictionary attack tool for SSH and many other protocols. For this example, the tool will take 4 input values: username file, a password file, target list, and a service (ssh). With that information, it attempts to perform a network-based authentication to the remote machine(s) on port 22. If it successfully authenticates with the remote machine, it shows the results in the standard output with the username and passwords that worked.

For this exercise, you are going to use Hydra to perform an SSH dictionary attack against a machine specially prepared for this purpose, **target.colab.duke.edu**. Do not use the tool on any other targets!

Implementation note: Because I've intentionally put a guessable username/password combo on this machine, I've used the ufw software firewall (based on Linux iptables) to restrict SSH access to campus IP ranges.

On your Kali VM, do the following.

Acquire a password list

A password list is a text file of likely passwords. There are *many* password lists out there, and Kali Linux ships with many of them pre-installed. For our purposes, we will use the Adobe "top 100" password list, which was derived from a 2013 breach of 150 million Adobe user accounts. You can find this password list in
`/usr/share/wordlists/metasploit/adobe_top100_pass.txt`

Build user list (1 point)

If an attacker is doing a general scan, they will typically use a common username list, such as one of these. If an attacker is focused on a known server or organization, they will do research using public information to create a list of likely usernames.

In this scenario, let's assume we are targeting an organization's founders, which includes Susan Hypothetical, Scott Samplesberg, and Joe Exampleface. Based on this, produce a small text file of likely usernames.

Paste your username list below.
susan
hypothetical
susan.hypothetical
s.hypothetical
susan.h
sh
s.h
scott
samplesberg
scott.samplesberg
s.samplesberg
scott.s
ss
s.s
joe
exampleface

joe.exampleface
j.exampleface
joe.e
je
j.e

Attack (4 points)

Run hydra against target.colab.duke.edu using the resources gathered.

Give the hydra command you used below.

```
hydra ssh://target.colab.duke.edu -P /usr/share/wordlists/metasploit/adobe_top100_pass.txt -L pass.txt
```

Display the successful results of your Hydra Attack below. If you don't get any results, try more (and simpler!) usernames.

```
[22][ssh] host: target.colab.duke.edu   login: joe   password: zxcvbnm
```

Note: It is very likely that you will only be able to find one set of working credentials.

When you log into target.colab.duke.edu using the credentials found, a secret message is displayed. What is it?

Joe Exampleface is AWESOME!!

OPTIONAL: For up to 5 points of extra credit, get a shell on target.colab.duke.edu.

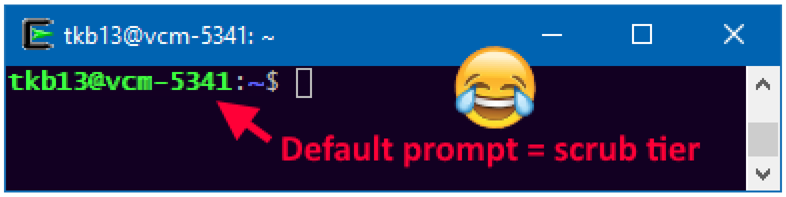
The vulnerable account on the server is configured to print a secret bit of text and exit without providing a prompt. **Show how you can get an interactive shell using these credentials.**

Command:

```
ssh -t joe@target.colab.duke.edu /bin/bash
```

Question 12: Craft your prompt (3 points)

A rite of passage of UNIX users is the customization of the bash prompt. The default, even on modern Ubuntu, is pretty lame.



```
tkb13@vcm-5341: ~
tkb13@vcm-5341:~$ 
```

Using your understanding of shell color codes, develop a custom prompt by modifying the PS1 environment variable. You'll need to mark the escape codes for the shell using `\[` and `\]` indicators (this the shell needs to know what characters are printed vs. interpreted for cursor control purposes); details are here.

Edit the file `.bashrc` (sourced by the bash shell every time it is launched) in order to apply your prompt permanently.

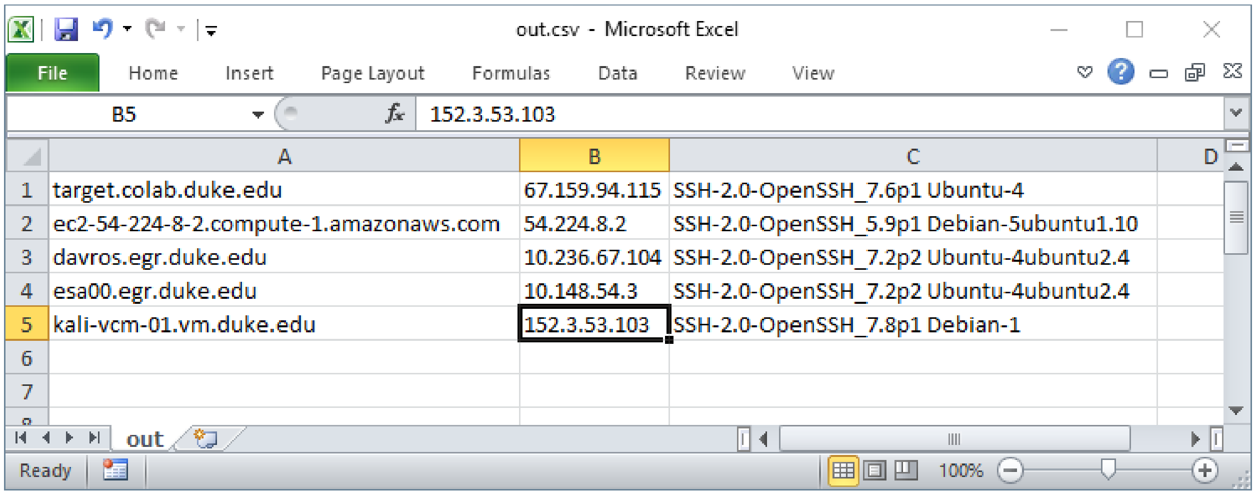
Paste your PS1 environment variable command from `.bashrc` & a screenshot of the resulting prompt.

```
export
PS1="\[\e[36m\][\[\e[m\]\[\e[31m\]\u\[\e[m\]\[\e[33m\]@\[\e[m\]\[\e[32m\]\h\[\e[m\]:\[\e[32m\]\w\[\e[
m\]\[\e[36m\]]\[\e[m\]\\$ \[\e[m\]"
cat<<'EOF'
                 ,--,                                                         ,----,.
               ,---.'|                                                      ,'   ,' |
   ,---,       |   | :                                 ,---,.  ,----..     ,---,.  ,'   .'  |              ,----..
,`--.' |       :   : |                               ,'  .' | /   /   \  ,'  .' | ,----.'   .'    ,---.     /   /   \
|   :  :       |   ' :     ,---.                   ,---.'   ||   :     :,---.'   | |   |  .'     /     \   /   .     :
:   |  '       ;   ; '     '   ,'\     .---.       |   |   .'.   |  ;. /|   |   .' :   :  |--,   /    / '  .   /   ;.  \
|   :  |       '   | |__  /   /   |   /.  ./|  ,---.     :   :  |-,.   ; /--` :   :  |-, :   |  ;.' \ .    ' /   .   ;   /  ` ;
'   '  ;       |   | :.'|.  ; ,. : .-' . ' | /     \    :   |  ;/|;   | ;     :   |  ;/| |   |     |'   / ;    ;   |  ; \ ; |
|   |  |       '   :    ;'  | |: :/___/ \: |/    /  |    |   :   .'|   : |    |   :   .' `----'.'\   ;|   :  \    |   :  | ; | '
'   :  ;       |   |  ./ '  | .; :.   \  ' .    ' / |    |   |  |-,.   | '___ |   |  |-,   ___ \  .  |;   |   ``. .   | ' ' ' :
|   |  '       ;   : ;   |   :    | \   \   '   ;   /|    '   :  ;/|'   ; : .'|'   :  ;/| /   /\/  /  :'   ;      \'   ;  \; /  |
'   :  |       |   ,/     \   \  /   \   \  '   |  / |    |   |    \'   | '/  :|   |    \/ ,,/  ',-   .'   |  .\  |\   \  ',  /
;   |.'        '---'       `----'     \   \ |   :    |    |   :   .'|   :    / |   :   .'\ ''\        ;|   :  ';  :  ;   :    /
'---'                                 '---" \   \  /     |   | ,'   \   \ .'  |   | ,'  \   \   .'  \   \    /   \   \ .'
                                            `----'      `----'      `---`    `----'     `--`-,-'     `---`--`     `---`
EOF
```

```
Last login: Sun Oct 20 01:50:40 2024 from syn-076-036-241-038.res.spectrum.com

[fg96@vcm-42411:~]$
```

Question 13: Attack recon (6 points)

One of the first things an attacker will do when focusing on a particular target organization is conduct reconnaissance. Using your choice of tools we've learned, develop a command or script that will produce a CSV file containing the hostname, IP address, and SSH version of all the hosts listed in an input file. Example output (once loaded into Excel):



out.csv - Microsoft Excel

B5 152.3.53.103

	A	B	C
1	target.colab.duke.edu	67.159.94.115	SSH-2.0-OpenSSH_7.6p1 Ubuntu-4
2	ec2-54-224-8-2.compute-1.amazonaws.com	54.224.8.2	SSH-2.0-OpenSSH_5.9p1 Debian-5ubuntu1.10
3	davros.egr.duke.edu	10.236.67.104	SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.4
4	esa00.egr.duke.edu	10.148.54.3	SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.4
5	kali-vcm-01.vm.duke.edu	152.3.53.103	SSH-2.0-OpenSSH_7.8p1 Debian-1

Apply this to the following hosts:

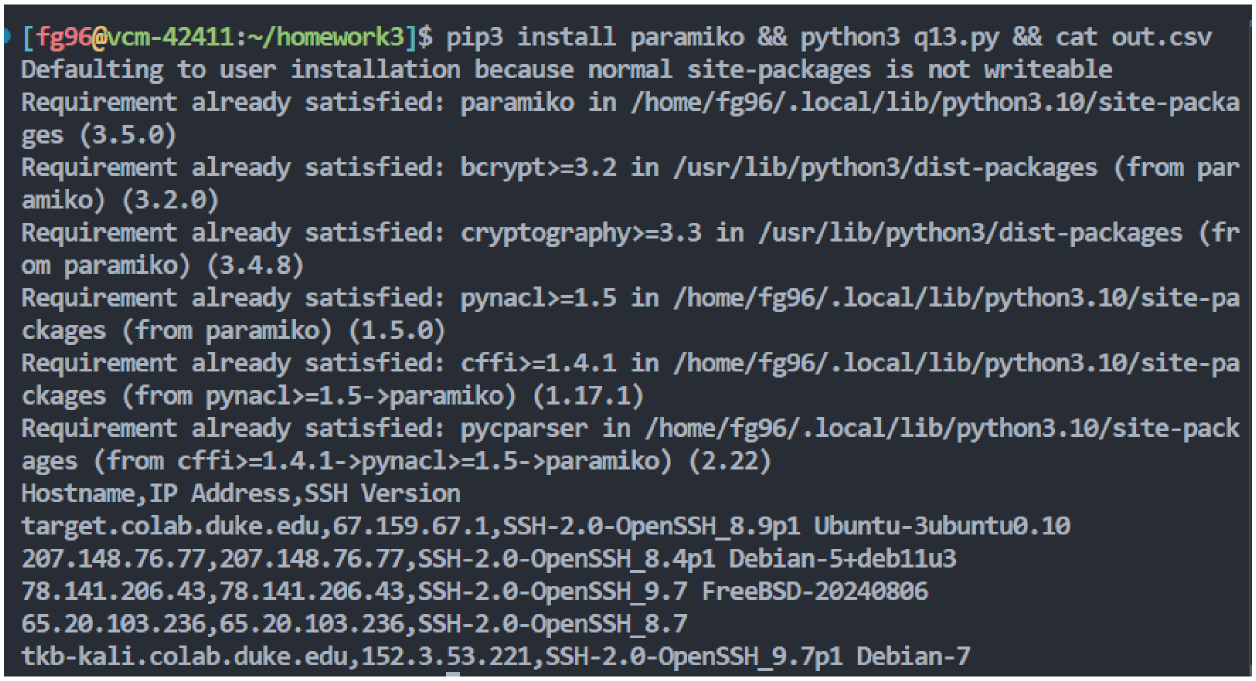
```
target.colab.duke.edu
207.148.76.77
78.141.206.43
65.20.103.236
tkb-kali.colab.duke.edu
```

(Do not scan machines other than the above. All of the above are machines administered by me either at Duke or at a cloud hosting provider.)

Your goal is to write a tiny script as quickly and easily as possible. You may include code from the internet with citation.

Provide:
(1) the command or script as copyable text (so we can try it out)
(2) a screenshot of your command plus its output

```python
import csv
import paramiko
import socket

hosts = [
    "target.colab.duke.edu",
    "207.148.76.77",
    "78.141.206.43",
    "65.20.103.236",
    "tkb-kali.colab.duke.edu"
]

def get_ip_address(hostname):
    try:
        return socket.gethostbyname(hostname)
    except socket.error as e:
        return None

def get_ssh_version(ip):
    try:
        sock = socket.create_connection((ip, 22), timeout=3)
        transport = paramiko.Transport(sock)
        transport.start_client()
        return transport.remote_version
    except Exception as e:
        return None

if __name__ == "__main__":
    with open('out.csv', mode='w', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(["Hostname", "IP Address", "SSH Version"])
        for host in hosts:
            ip_address = get_ip_address(host)
            ssh_version = get_ssh_version(ip_address)
            writer.writerow([host, ip_address, ssh_version])
```


```
[fg96@vcm-42411:~/homework3]$ pip3 install paramiko && python3 q13.py && cat out.csv
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: paramiko in /home/fg96/.local/lib/python3.10/site-packa
ges (3.5.0)
Requirement already satisfied: bcrypt>=3.2 in /usr/lib/python3/dist-packages (from par
amiko) (3.2.0)
Requirement already satisfied: cryptography>=3.3 in /usr/lib/python3/dist-packages (fr
om paramiko) (3.4.8)
Requirement already satisfied: pynacl>=1.5 in /home/fg96/.local/lib/python3.10/site-pa
ckages (from paramiko) (1.5.0)
Requirement already satisfied: cffi>=1.4.1 in /home/fg96/.local/lib/python3.10/site-pa
ckages (from pynacl>=1.5->paramiko) (1.17.1)
Requirement already satisfied: pycparser in /home/fg96/.local/lib/python3.10/site-pack
ages (from cffi>=1.4.1->pynacl>=1.5->paramiko) (2.22)
Hostname,IP Address,SSH Version
target.colab.duke.edu,67.159.67.1,SSH-2.0-OpenSSH_8.9p1 Ubuntu-3ubuntu0.10
207.148.76.77,207.148.76.77,SSH-2.0-OpenSSH_8.4p1 Debian-5+deb11u3
78.141.206.43,78.141.206.43,SSH-2.0-OpenSSH_9.7 FreeBSD-20240806
65.20.103.236,65.20.103.236,SSH-2.0-OpenSSH_8.7
tkb-kali.colab.duke.edu,152.3.53.221,SSH-2.0-OpenSSH_9.7p1 Debian-7
```

Question 14: Keeping up with the news (10 points)

One absolutely essential attribute to the field of information security is that **it changes**. While most of the base theory we're talking about will remain true long into the future, attackers and defenders are locked into a continuous arms race of ever increasing sophistication. As such, a core skill for any security practitioner is to keep up to date with developments in the field.

Below are five IT security news sources:

- Security Week
- ThreatPost
- Reddit /r/netsec
- The Record
- Risky Business News

Browse each, and pick out an article from the last 30 days related to one of the following topics: networking, cryptography, user authentication, malware, or denial of service attacks.

Based on this article, do the following:

- Provide a brief summary, including a link to the article. (3)
- Identify which aspects of the CIA triad are at play and how. (1)
- Explain the threat model(s) at play (assets at risk, vulnerability at play, attacker's capabilities and knowledge). (2)
- If your article is explaining an attack, describe a defense that would have mitigated the attack and explain why it would be effective. Use the threat model in your analysis.

  -or-

If your article is explaining a defense, describe an attack that the defense would mitigate and explain why it would be effective. Use the threat model in your analysis. (4)

Article name: Casio Confirms Data Breach as Ransomware Group Leaks Files
Article date: Oct 14, 2024
Article link: https://www.securityweek.com/casio-confirms-data-breach-as-ransomware-group-leaks-files/

Summary: The article describes a recent data breach involving Casio, a well-known Japanese electronics company. The incident resulted in a system failure and some service disruptions. The ransomware group "Underground" claimed responsibility for the attack and leaked sensitive data.

CIA Triad Analysis:
1. Confidentiality: Underground cybercriminals have gained unauthorized access to and leakage of sensitive information compromises data privacy and security.
2. Availability: The breach has resulted in a system failure and some service disruptions

Threat Model:
Assets at risk: Sensitive corporate information, potentially including employee records, customer data, and internal communications.
Vulnerability at play: The breach points to weaknesses in network security, potentially involving outdated software or inadequate access controls.
Attacker's Capabilities: The ransomware group likely had the knowledge and capability to exploit Casio's internal network vulnerabilities, gain administrative access, and exfiltrate data.

Defense Analysis:
1. Encrypting sensitive files both at rest and in transit would have prevented the attackers from easily leaking readable data, mitigating the confidentiality breach (Even if the ransomgroup has the capability to gain unauthorized access to the network, they will not be able to decrypt the info).
2. Upgrade software (with patches) to the latest version to prevent security vulnerabilities (so that the attacker will not use their knowledge to exploit the internal network).
3. Perform network segmentation, the critical info will be stored on different server locations, so even if attackers breached one segment, they would have faced additional barriers in accessing valuable data, which is a defense at attacker's knowledge to exploit only one buggy network.

Question 15: Manipulating binary file formats (5 points)

Question 0 of this assignment involved detecting and decoding a polyglot PDF that was also a valid JPEG. **To receive credit for this question, the answers PDF you submit must also be a polyglot, but rather than a PDF+JPEG polyglot, it should be a PDF+ZIP polyglot. The ZIP aspect should contain (1) your public SSH key (the one submitted earlier to Encrypted Thing Giver in homework 2) and (2) a picture of a fat dog (<100kB).**

The construction of a PDF+ZIP polyglot is easier than you may guess because of the peculiar format of ZIP files. You may consult the ZIP file format article on Wikipedia, the napkin drawings by Julia Wolf in PoC||GTFO 01:05 (“This ZIP is also a PDF”), or this way-way-too-detailed presentation deck also by Julia Wolf.

HINTS: Does a zip file have a header? Where is its central directory?