Computer and Information Security

(ECE560, Fall 2020, Duke Univ., Prof. Tyler Bletsch)

Homework 3

Name: _Yisong Zou_____

Duke NetID: __yz558____

Instructions - read all carefully:

- **DON'T SCREW UP:** Read each question carefully and be sure to answer all parts. Some questions are a mix of explanation and questions, so pay close attention to where you are being asked for something. It is recommended to answer the questions in the order they are asked, as they build on each other.
- **COMPUTERS YOU WILL NEED:**
  - The assignment will make use of the computers described below.
  - VMs you already have on the Duke VCM service:
    - Your **Linux VM** ("*ECE.560.01.F20 - Ubuntu20.04*" on VCM)
    - Your **Windows VM** ("*ECE.560.01.F20 - Win10*" on VCM)
    - A new **Kali VM** ("ECE.560.01.F20 - Kali 2002.2" on VCM)
  - Your own machine on Duke wifi: your **personal computer** (any OS). (If working remotely, VPN into the Duke network as needed.)
- **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.
- **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.

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 (8 points)

(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?

They are: Something the individual knows, Something the individual possesses, Something the individual is (static biometrics) and Something the individual is (static biometrics).

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

Popular password attack: Try few popular passwords on many accounts

Password guessing against single user: Do research on a single user and guess his password.

Exploiting multiple password use: Figure out one password for one platform and can get to know many other platform's passwords

Electronic monitoring: Sniffing network or use a keylogger to get the password

Offline dictionary attack: Cracking hashed passwords and use them to login.

c. What are three common techniques used to protect stored passwords?

They are:

Hash: Store the hash of the password in the database instead of the plaintext.

Salt: Add a random salt to the password before hashing.

Iteration count: Use more iteration counts when hashing a password.

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

Password complexity is more important because it will make the cracker impossible to figure out the password. However, if we use a weak password, even if it expires really fast, the cracker can find a way to crack it in no time.

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

MFA, aka multifactor authentication, refers to the authentication method that uses more than one means/factors.

f. List and briefly describe the principal characteristics used for biometric identification.

Facial characteristics, Fingerprints, Hand geometry, Retinal pattern, Iris, Signature, Voice. And the methods are based on pattern recognition.

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

Assume we have some authentication secret S and we do not want to send it. Instead, the server issues a challenge to the client that can only be answered if it has S, so in this way we can protect S and do not need to reveal it during the process.

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

Because the hash of the password might leak an under such circumstance, use a salted password with the nonce will solve this problem.

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 773.

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

The first three positions designate owner's permissions.

The second three positions designate permissions for the group.

The last three positions are for the world/anyone.

Files:

(r) read (view the file)

(w) write (create, edit or delete)

(x) execute (run a script/program)

Directories:

(r) read allows a user to view the directory's contents

(w) write allows a user to create new files or delete files in the directory.

(x) execute determines if the user can enter (cd) into the directory or run a program or script.

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

File 644:protection string is rw-r--r--, represents that for the file:

Owner can view the file and create, edit or delete the file.

Group can view the file.

Others can view the file.

Directory 750 :protection string is rwxr-x---, represents that for the directory:

Owner can view the directory's contents, create new files or delete files in the directory and enter (cd) into the directory or run a program or script.

Group can view the directory's contents and enter (cd) into the directory or run a program or script.

Others have no permission to the 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)

Non-owner non-group users cannot list the contents of the directory, but they can create new files.

```
yz558@vcm-17149:~$ getfacl permissionTest/
# file: permissionTest/
# owner: root
# group: root
user::rwx
group::rwx
other::-wx

yz558@vcm-17149:~$ cd permissionTest/
yz558@vcm-17149:~/permissionTest$ mkdir abc
yz558@vcm-17149:~/permissionTest$ ls
ls: cannot open directory '.': Permission denied
yz558@vcm-17149:~/permissionTest$
```

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)

-rw-r--r-- means that the file gosh.tar.gz can be read and write by the owner which is root, can be read only by the group root and other users.

The first root means that the owner of the file is root, the second root means that the group of the file is root.

1.6M is the size of the file and it is followed with the date of creation of the file.

The last is the file name.

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

chmod a+w <filename>

What command would you use to take full ownership from root to yourself? (1)

sudo chown <myUserName> <filename>

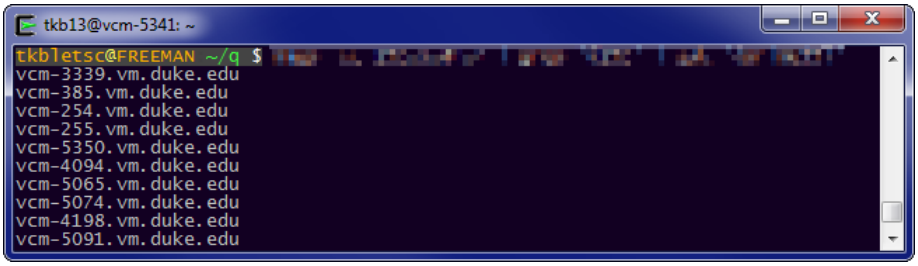
Question 3: Data processing (16 points)

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:



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

```
nmap -sn 152.3.64.* | grep 'Nmap scan report for vcm' | awk '{print $5}'
```

```
yz558@vcm-17149:~$ nmap -sn 152.3.64.* | grep 'Nmap scan report for vcm' | awk '{print $5}'
vcm-15220.vm.duke.edu
vcm-255.vm.duke.edu
vcm-15323.vm.duke.edu
vcm-15243.vm.duke.edu
vcm-15255.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 'Invalid user admin from' auth.log | awk '{print $10}' | sort -u

```
yz558@vcm-17149:~$ grep 'Invalid user admin from' auth.log | awk '{print $10}' | sort -u
103.78.150.164
106.14.185.125
106.51.226.154
110.185.106.47
111.241.208.92
113.161.32.65
113.173.152.101
113.190.53.212
114.67.38.1
115.146.127.201
115.248.207.78
115.84.91.115
116.96.215.181
116.97.207.57
```

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.

I firstly used

wget --recursive http://target.colab.duke.edu/app/

To get all the pages from the website.

```
yz558@vcm-17149:~/mockWebApp/target.colab.duke.edu/app$ ls
fd.jpg                                  X-5612da2213e7ca2799703a7246ccfe54.html  X-b2c0c817ed1fd5ba5085cb6ad920b5f2.html
index.html                              X-56807d98db5f5c26d9754e72d2e88b19.html  X-b2e1b85f7328778bb8d3fa48bf2bd602.html
X-00147160ccd25ac78482ed5fdc755ff9.html X-56daed52bb7683e8d4c879658bc75fcb.html  X-b2ea97439b43d1c70b75f1521464aea7.html
X-0037b5ad990f6a85732adc0171f7b399.html X-577d27510343540911799e10143a621f.html  X-b3b450a59eeb639d77246a4c7a744302.html
X-00eff2191fb569bb01579a8f8fbb90d4.html X-58428a4a0b23ef6e9bf799693fa88d46.html  X-b3b5163150Z0d8378ddab591b24f4aa6.html
```

Then I used

grep ‘<form>’ `ls`

to search all the htmls in the website in the current directory and get the two <forms> tag in the two files.

```
yz558@vcm-17149:~/mockWebApp/target.colab.duke.edu/app$ grep '<form>' `ls`
X-8dbb314e8ce2364be7117ca36eba64a4.html:<form><input type=text value='Sensitive vulnerable stuff'><input type=submit></form>
X-d1818c1ddd12f4455a79b7a74844e30d.html:<form><input type=text value='Sensitive vulnerable stuff'><input type=submit></form>
```

So the two pages are

X-8dbb314e8ce2364be7117ca36eba64a4.html

(http://target.colab.duke.edu/app/X-8dbb314e8ce2364be7117ca36eba64a4.html)

and

X-d1818c1ddd12f4455a79b7a74844e30d.html

(http://target.colab.duke.edu/app/X-d1818c1ddd12f4455a79b7a74844e30d.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.

1.Fat dog

For the fat dog, I firstly used

wget --recursive http://target.colab.duke.edu/app/

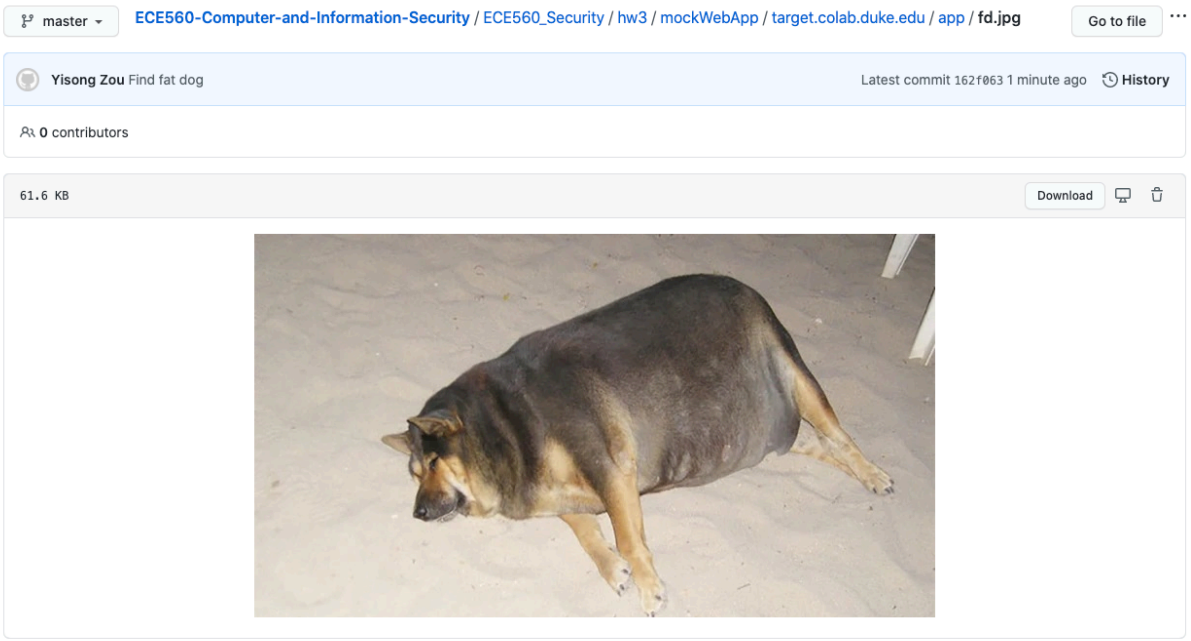
To get all the pages from the website.

Then when I use

ls

I can find a fd.jpg in the document, and I pushed it to my github and it is like the following:

```
yz558@vcm-17149:~/mockWebApp/target.colab.duke.edu/app$ ls
fd.jpg                                  X-5612da2213e7ca2799703a7246ccfe54.html  X-b2c0c817ed1fd5ba5085cb6ad920b5f2.html
index.html                              X-56807d98db5f5c26d9754e72d2e88b19.html  X-b2e1b85f7328778bb8d3fa48bf2bd602.html
X-00147160ccd25ac78482ed5fdc755ff9.html X-56daed52bb7683e8d4c879658bc75fcb.html  X-b2ea97439b43d1c70b75f1521464aea7.html
X-0037b5ad990f6a85732adc0171f7b399.html X-577d2751034354091179 9e10143a621f.html X-b3b450a59eeb639d77246a4c7a744302.html
X-00eff2191fb569bb01529a8f8fbb90d4.html X-58428a4a0b23ef6e9bf799693fa88d46.html X-b3b51631507 0d8378ddab591b24f4aa6.html
```

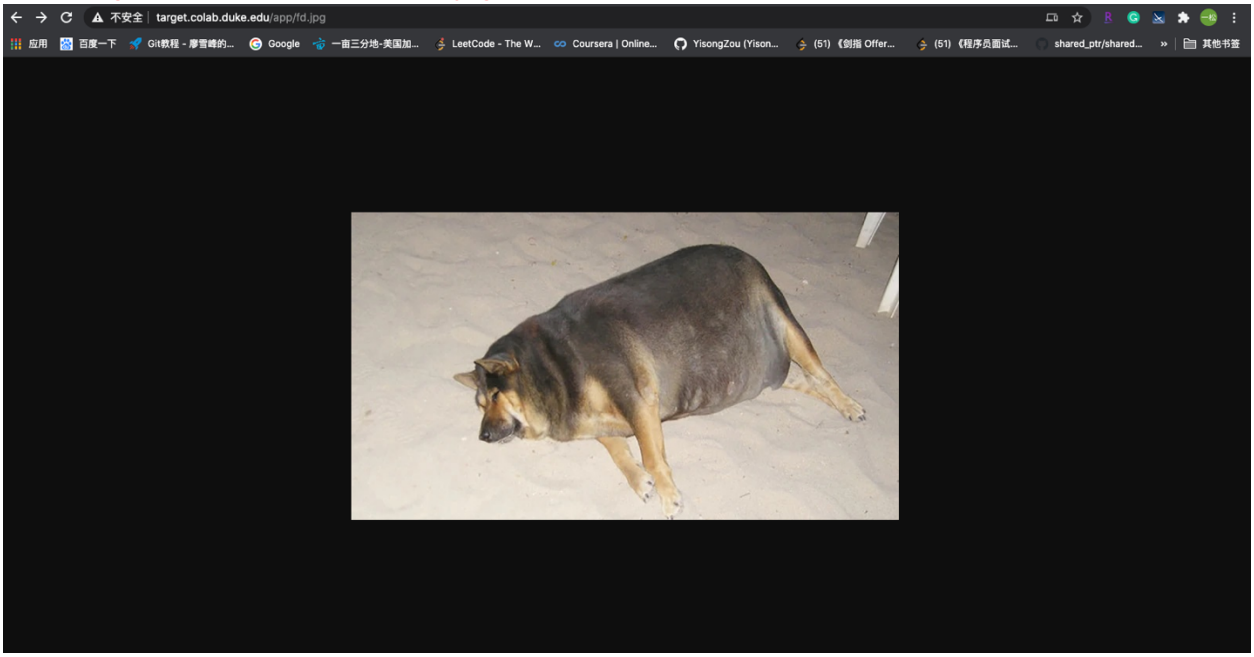


```
target.colab.duke.edu/app/X-8dbb314 100%[=================================================================>]  10.22K  --.-KB/s    in 0s

2020-10-02 18:03:52 (257 MB/s) - ‘target.colab.duke.edu/app/X-8dbb314e8ce2364be7117ca36eba64a4.html’ saved [10467/10467]

--2020-10-02 18:03:52--  http://target.colab.duke.edu/app/fd.jpg
Reusing existing connection to target.colab.duke.edu:80.
HTTP request sent, awaiting response... 200 OK
Length: 63056 (62K) [image/jpeg]
Saving to: ‘target.colab.duke.edu/app/fd.jpg’
```

From the above result, I got that we can also visit this photo using the link:
http://target.colab.duke.edu/app/fd.jpg

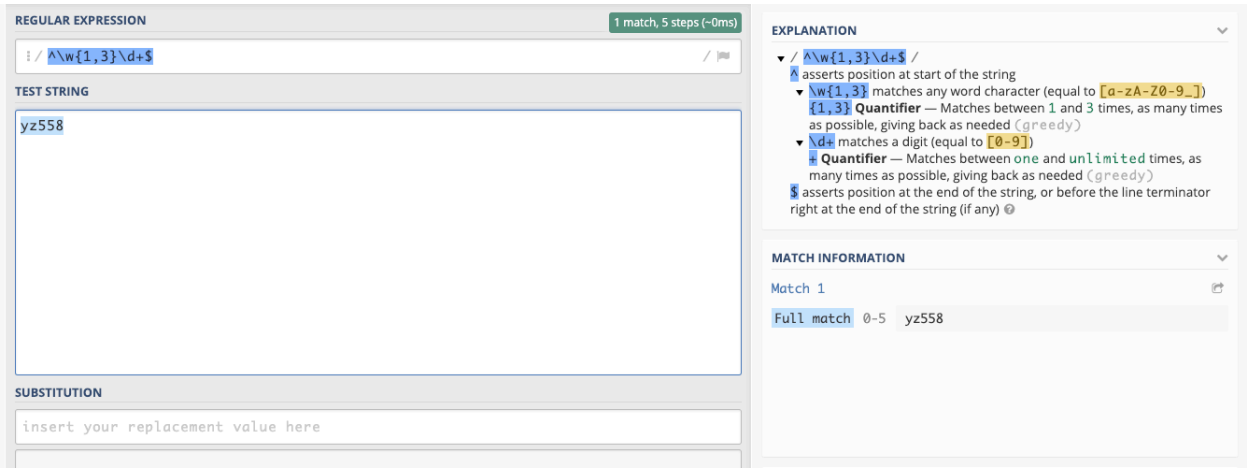


Question 4: Regular expressions (9 points)

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).

/^\w{1,3}\d+$/

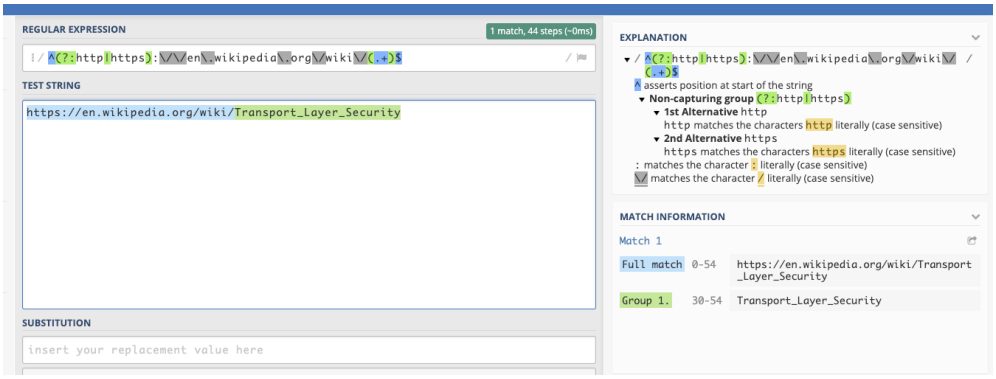


URL parser (3 points)

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	-

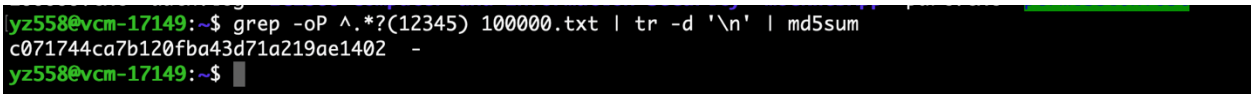
/^(?:http|https):\/\/en\.wikipedia\.org\/wiki\/(.+)$/



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 -E and other options, perl, Python, or other tools.

grep -oP ^.*?(12345) 100000.txt | tr -d '\n' | md5sum



Question 5: 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.

```
yz558@kali:~$ screen -list
There are screens on:
        114061.pts-0.kali       (10/03/20 16:22:59)     (Detached)
        114037.pts-0.kali       (10/03/20 16:18:10)     (Detached)
2 Sockets in /run/screen/S-yz558.
yz558@kali:~$ screen -ls
```

Question 6: Cracking MD5 hashes with hashcat (6 points)

There is an FTP server called Serv-U, a commercial program written by CATsoft. Hackers 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 hashing algorithm. 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)`**.

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.

(1) hashcat --force -a 3 -m 20 hashes

(2)

```
5cf2ff593419bcf3d22c62e65a82128a:ly:manito

Session..........: hashcat
Status...........: Cracked
Hash.Type........: md5($salt.$pass)
Hash.Target......: 5cf2ff593419bcf3d22c62e65a82128a:ly
Time.Started.....: Sat Oct  3 17:03:35 2020 (10 secs)
Time.Estimated...: Sat Oct  3 17:03:45 2020 (0 secs)
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.........: 31589.4 kH/s (7.96ms) @ Accel:512 Loops:256 Thr:1 Vec:4
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 322527232/3748902912 (8.60%)
Rejected.........: 0/322527232 (0.00%)
Restore.Point....: 144384/1679616 (8.60%)
Restore.Sub.#1...: Salt:0 Amplifier:0-256 Iteration:0-256
Candidates.#1....: sa4rto -> prltho

Started: Sat Oct  3 17:03:13 2020
Stopped: Sat Oct  3 17:03:46 2020
yz558@kali:~$ ls
ECE560-Computer-and-Information-Security  certs  hashes
yz558@kali:~$
```

(3) The password is: manito

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

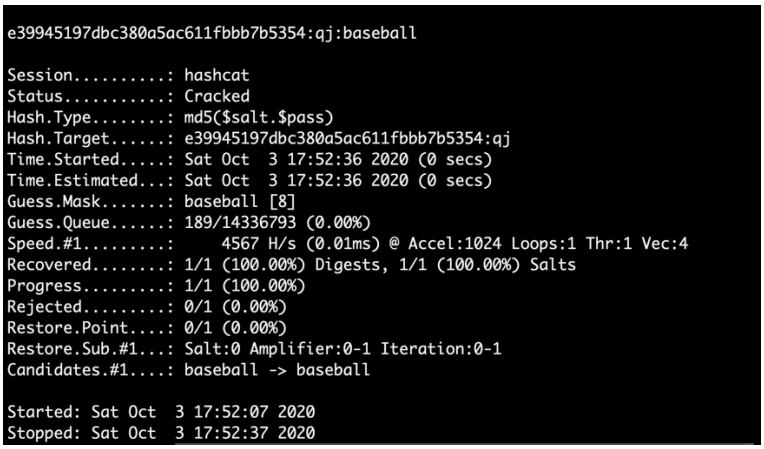
You may need to use more time, wordlists, and possibly even a combination wordlist/brute-force attack.

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

a.

The password is: baseball

hashcat --force -a 3 -m 20 a /usr/share/wordlists/rockyou.txt


e39945197dbc380a5ac611fbbb7b5354:qj:baseball
Session..........: hashcat
Status...........: Cracked
Hash.Type........: md5($salt.$pass)
Hash.Target......: e39945197dbc380a5ac611fbbb7b5354:qj
Time.Started.....: Sat Oct 3 17:52:36 2020 (0 secs)
Time.Estimated...: Sat Oct 3 17:52:36 2020 (0 secs)
Guess.Mask.......: baseball [8]
Guess.Queue......: 189/14336793 (0.00%)
Speed.#1.........:   4567 H/s (0.01ms) @ Accel:1024 Loops:1 Thr:1 Vec:4
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 1/1 (100.00%)
Rejected.........: 0/1 (0.00%)
Restore.Point....: 0/1 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1....: baseball -> baseball
Started: Sat Oct 3 17:52:07 2020
Stopped: Sat Oct 3 17:52:37 2020

Question 7: John the Ripper (10 points)

To start, do some research, and **describe in detail (purpose and content) the /etc/passwd and /etc/shadow files on a Linux system. (4 points)**

(source: https://tldp.org/LDP/lame/LAME/linux-admin-made-easy/shadow-file-formats.html#:~:text=Traditional%20Unix%20systems%20keep%20user,%60%60%2Fetc%2Fpasswd".&text=A%20second%20file%2C%20called%20%60%60,or%20password%20expiration%20values%2C%20etc.)

/etc/password:

Unix systems keep user account information, including one-way encrypted passwords, in a text file called /etc/passwd. This file is used by many tools (such as ls) to display file ownerships. By matching user id #'s with the user's names, the file needs to be world-readable. Consequently, this can be somewhat of a security risk.

/etc/shadow:

Another way to store passwords is using the password shadow format. A second file, called /etc/shadow, contains encrypted password as well as other information such as account or password expiration values, etc. The /etc/shadow file is readable only by the root account and is therefore less of a security risk.

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/ece590-sec/homework/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.

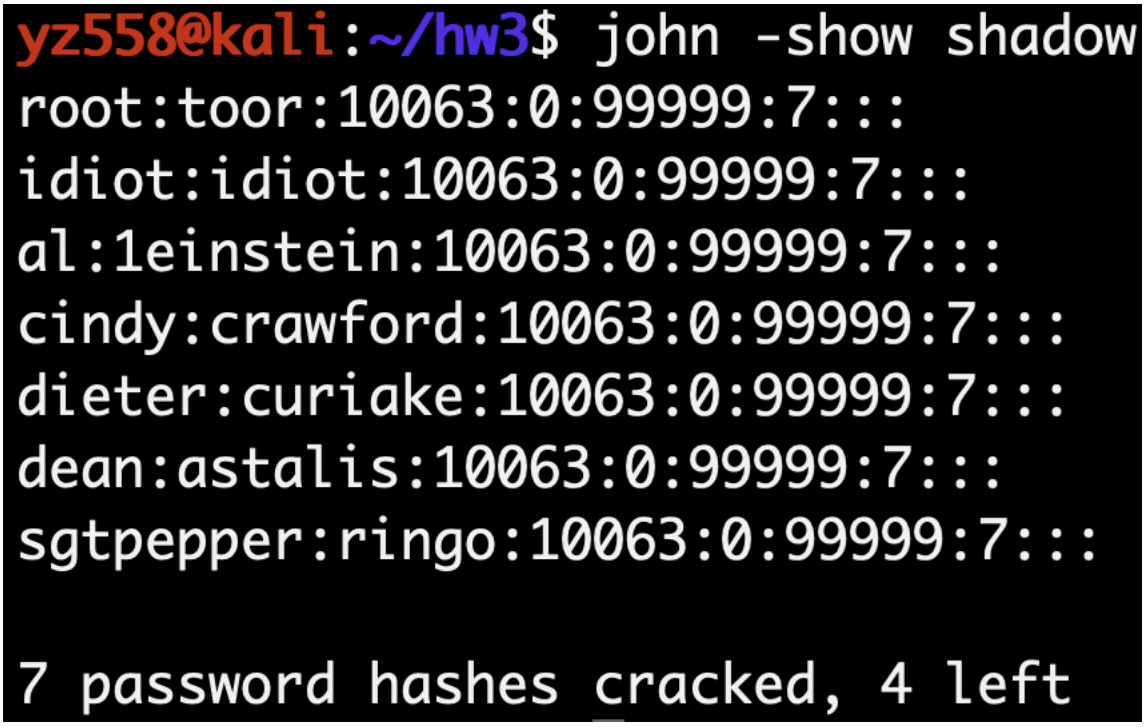
Paste the output your results with the following command "`john -show shadow`". (6 points)

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 8: Evaluating MFA approaches (8 points)

Review this article on a data theft attack against content aggregator site Reddit.

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

Reddit uses mobile text messages (SMS).

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

One is SIM-swap, the attacker masquerading as the target tricks the target's mobile provider into tying the customer's service to a new SIM card that the bad guys control.

Another is mobile number port-out scams, wherein the attacker impersonates a customer and requests that the customer's mobile number be transferred to another mobile network provider.

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

One is using a mobile app. Such as Google Authenticator or Authy, to generate the one-time code that needs to be entered in addition to a password.

Another is hardware-based security keys. These inexpensive USB-based devices allow users to complete the login process simply by inserting the device and pressing a button.

d. Identify another attack in the last 5 years in which SMS-based MFA was compromised.

Source: https://dcid.me/notes/2013-apr-19.html

A good example showing this insecurity happened to CloudFlare:

" AT&T was tricked into redirecting my voicemail to a fraudulent voicemail box; Google's account recovery process was tricked by the fraudulent voicemail box and left an account recovery PIN code that allowed my personal Gmail account to be reset; "

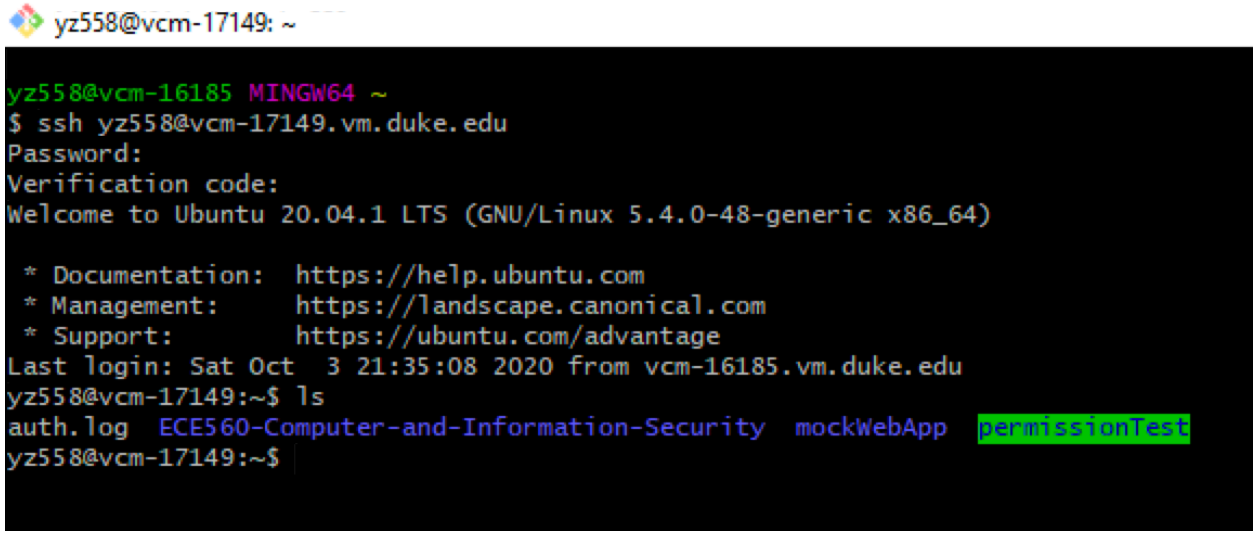
Question 9: SSH with MFA (5 points)

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.

Source: https://blog.devolutions.net/2017/05/how-to-secure-ssh-with-google-two-factor-authentication



Question 10: 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 senior leadership, which includes Susan Hypothetical, Scott Samplesberg, and Jacob Exampleface. Based on this, produce a small text file of likely usernames.

Paste your username list below.

hyp
hypothetical
sht
ssusan
ssan
hypo
hpt
sh

susan
SH
scott
sample
samples
Berg
sscott
sssb
SS
ss
ssb
sct
sb
Berg
ssamples
jjacob
jcb
ex
jacob
jef
example
face
Jacob

Attack (4 points)

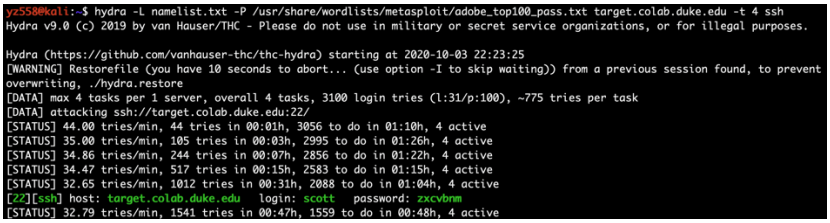
Run hydra against target.colab.duke.edu using the resources gathered.

Give the hydra command you used below.

(source: https://linuxconfig.org/ssh-password-testing-with-hydra-on-kali-linux)

hydra -L namelist.txt -P /usr/share/wordlists/metasploit/adobe_top100_pass.txt target.colab.duke.edu -t 4 ssh

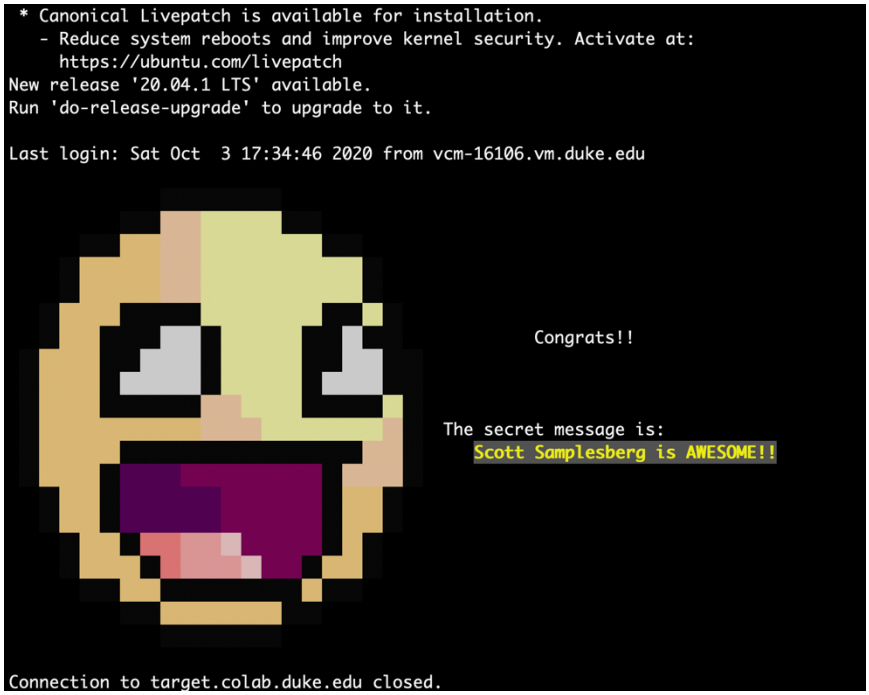
Display the <u>successful</u> results of your Hydra Attack below. If you don't get any results, try more (and simpler!) usernames.



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?

Scott Samplesberg is AWESOME!!

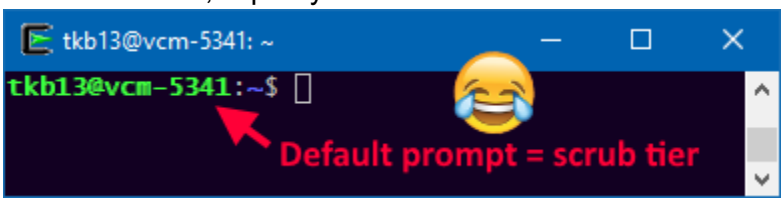


OPTIONAL: For up to 5 points of extra credit, get a shell on target.colab.duke.edu.

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.**

Question 11: 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.



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.

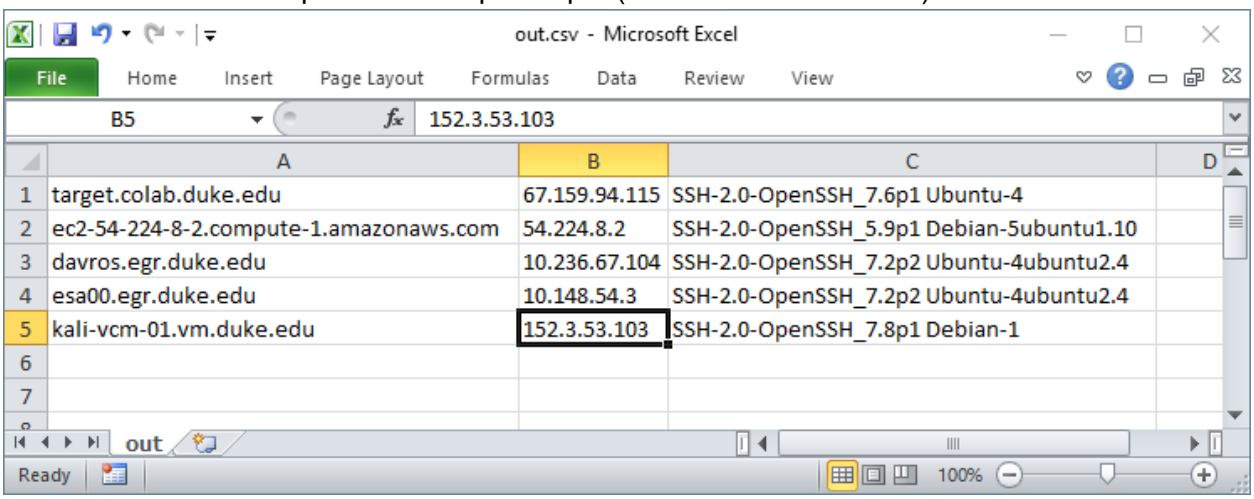
export PS1=export PS1="\e[40;33m[\u@\h \W \\@]\$ \e[m "

```
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
export PS1="\e[40;33m[\u@\h \W \\@]\$ \e[m "
```

```
[yz558@kali ~ 11:11 PM]$ ls
ECE560-Computer-and-Information-Security  certs  hydra.restore  namelist.txt  namelist.txt~
[yz558@kali ~ 11:11 PM]$
```

Question 12: 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):



Apply this to the following hosts:

```
target.colab.duke.edu
96.30.196.58
139.180.157.201
davros.egr.duke.edu
esa06.egr.duke.edu
storemaster.egr.duke.edu
t-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.)

Give your script, code, or other artifacts inline in the answer PDF along with a screenshot of it in use. Your solution should be as small and simple as possible!

Here I used a shell script to do the task. The content of the file script.sh

```
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
echo -n > out.csv
while read NAME;do
    if [[ $NAME =~ ^[0-9]+ ]];then
        host $NAME | awk '{print $5}' | sed 's/.$//' | tr -d '\n' >> out.csv
        echo -n , >> out.csv
        echo -n $NAME, >> out.csv
        sudo scanssh -s ssh $NAME | awk '{print $2 " "$3}' >> out.csv
    else
        echo -n $NAME, >> out.csv
        nslookup $NAME | grep ^Name -A1 | grep ^Address | awk '{print $2}' | tr -d '\n' >> out.csv
        echo -n , >> out.csv
        sudo scanssh -s ssh $NAME | awk '{print $2 " "$3}' >> out.csv
    fi
done < hostList.txt
```

The content of the file script.sh

```
#!/bin/bash

echo -n > out.csv
while read NAME;do
    if [[ $NAME =~ ^[0-9]+ ]];then
        host $NAME | awk '{print $5}' | sed 's/.$//' | tr -d '\n' >> out.csv
        echo -n , >> out.csv
        echo -n $NAME, >> out.csv
        sudo scanssh -s ssh $NAME | awk '{print $2 " "$3}' >> out.csv
    else
        echo -n $NAME, >> out.csv
        nslookup $NAME | grep ^Name -A1 | grep ^Address | awk '{print $2}' | tr -
d '\n' >> out.csv
        echo -n , >> out.csv
        sudo scanssh -s ssh $NAME | awk '{print $2 " "$3}' >> out.csv
    fi
done < hostList.txt
```

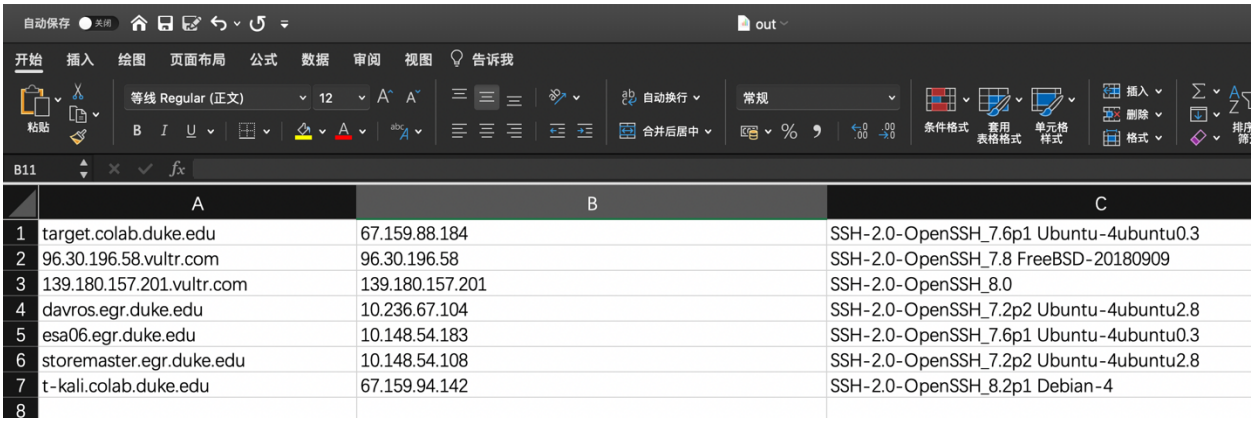
Usage: ./ script.sh and it will produce an out.csv file

```
[yz558@kali hw3 02:38 AM]$  ./script.sh
sudo: unable to resolve host kali: Name or service not known
Effective host scan rate: 2.69 hosts/s
sudo: unable to resolve host kali: Name or service not known
Effective host scan rate: 2.03 hosts/s
sudo: unable to resolve host kali: Name or service not known
Effective host scan rate: 2.34 hosts/s
sudo: unable to resolve host kali: Name or service not known
Effective host scan rate: 4.39 hosts/s
sudo: unable to resolve host kali: Name or service not known
Effective host scan rate: 3.18 hosts/s
sudo: unable to resolve host kali: Name or service not known
Effective host scan rate: 18.52 hosts/s
sudo: unable to resolve host kali: Name or service not known
Effective host scan rate: 2.92 hosts/s
[yz558@kali hw3 02:39 AM]$  cat out.csv
target.colab.duke.edu,67.159.88.184,SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3
96.30.196.58.vultr.com,96.30.196.58,SSH-2.0-OpenSSH_7.8 FreeBSD-20180909
139.180.157.201.vultr.com,139.180.157.201,SSH-2.0-OpenSSH_8.0
davros.egr.duke.edu,10.236.67.104,SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.8
esa06.egr.duke.edu,10.148.54.183,SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3
storemaster.egr.duke.edu,10.148.54.108,SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.8
t-kali.colab.duke.edu,67.159.94.142,SSH-2.0-OpenSSH_8.2p1 Debian-4
[yz558@kali hw3 02:39 AM]$  ls
hostList.txt  hydra.restore  namelist.txt  out.csv  script.sh
[yz558@kali hw3 02:39 AM]$
```

It takes an input file which is the hostList.txt, the content in hostList.txt is:

```
[yz558@kali hw3 02:39 AM]$  cat hostList.txt
target.colab.duke.edu
96.30.196.58
139.180.157.201
davros.egr.duke.edu
esa06.egr.duke.edu
storemaster.egr.duke.edu
t-kali.colab.duke.edu
[yz558@kali hw3 02:40 AM]$
```

And when load the out.csv to excel:



	A	B	C
1	target.colab.duke.edu	67.159.88.184	SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3
2	96.30.196.58.vultr.com	96.30.196.58	SSH-2.0-OpenSSH_7.8 FreeBSD-20180909
3	139.180.157.201.vultr.com	139.180.157.201	SSH-2.0-OpenSSH_8.0
4	davros.egr.duke.edu	10.236.67.104	SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.8
5	esa06.egr.duke.edu	10.148.54.183	SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3
6	storemaster.egr.duke.edu	10.148.54.108	SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.8
7	t-kali.colab.duke.edu	67.159.94.142	SSH-2.0-OpenSSH_8.2p1 Debian-4
8			

Question 13: 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 four IT security news sources:

- SANS Information Security News
- Security Week
- ThreatPost
- Reddit /r/netsec

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.

From this article:

- **Provide a brief summary, including a link to the article. (3)**
  https://threatpost.com/malware-pastebin-like-service/159838/
  The article talks about the trend that malwares are started to use legitimate web-services like pastebin or paste.nrecom for malware infrastructure.

- **Identify which aspects of the CIA triad are at play and how. (2)**
  Those malwares include ransomware and phishing emails.
  Ransomware will destroy user machines' availability and phishing emails will destroy confidentiality as it gets the users' private information.

- **Explain the threat model(s) at play (assets at risk, vulnerability at play, attacker's capabilities and knowledge). (2)**
  As the article mentions the spear-phishing attacks, the model is:

  Assets: The personal or other private information of the users.

  Vulnerability: The user may input their personal information into the spear-phishing email which is purposely built.

  Attacker's capabilities and knowledge: Attackers do research on the user and purposely build a pear-phishing email for the victim user.

- **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. (3)

My article is describing spear-phishing attack.

To solve this problem, we need to educate the users to teach them how to determine what emails are phishing emails because those attacks uses social engineering to let the users downloads or input their information. Moreover, at the same time we should develop the email system to auto block those emails with phishing risks.

Question 14: 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.