Computer and Information Security

(ECE590-03, Fall 2019, Duke Univ., Prof. Tyler Bletsch)

Homework 3

Name: ___Nan Ni_____

Duke NetID: ____nn75_____

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:
    - The **Ubuntu 18.04** VM; which we'll call your **Linux VM**.
    - The **Windows 10** VM; which we'll call your **Windows VM**.
    - The **Kali Linux** VM; which we'll call your **Kali VM**.
  - A new **throw-away VM with Windows 10**, which we'll call your **throwaway VM.**
    NOTE: Do not do the malware analysis question on your long-term Windows VM!
    You must destroy any VM that you run the malware on as soon as you are done with it!
  - Your own machine on Duke wifi: your **personal computer** (any OS).
- **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.
- **PROGRAMMING PORTION DIRECTIONS:**
  - There is a small programming project in this assignment; **your code for this will be submitted as a separate file** via the **Sakai assignment facility**. See the question itself for details.
- **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 means of authenticating a user's identity?
(1) **Something the individual knows**: Examples include a password, a personal Identification number (PIN), or answers to a prearranged set of questions.
(2) **Something the individual possesses**: Examples include electronic keycards, smart cards, and physical keys. This type of authenticator is referred to as a token.
(3) **Something the individual is (static biometrics) and does (dynamic biometrics)**: Static biometrics examples include recognition by fingerprint, retina, and face. Dynamic biometrics examples include recognition by voice pattern, handwriting characteristics, and typing rhythm.

b. List and briefly describe five possible threats to the secrecy of passwords.
(1) **Offline dictionary attack**: The attacker bypasses access controls and gain access to the password file, then compares the password hashes against hashes of commonly used passwords. If a match is found, the attacker can gain access by that ID/password combination.
(2) **Specific account attack**: The attacker targets a specific account and submits password guesses until the correct password is discovered.
(3) **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.
(4) **Workstation hijacking**: The attacker waits until a logged-in workstation is unattended.
(5) **Exploiting user mistakes**: If the system assigns a password, then the user is more likely to write it down because it is difficult to remember. This situation creates the potential for an adversary to read the written password. A user may intentionally share a password, to enable a colleague to share files, for example. Also, attackers are frequently successful in obtaining passwords by using social engineering tactics that trick the user or an account manager into revealing a password. Many computer systems are shipped with preconfigured passwords for system administrators. Unless these preconfigured passwords are changed, they are easily guessed.

c. What are three common techniques used to protect stored passwords?
(1) Storing hashed password
(2) Storing salted hash of password
(3) Use PBKDF2, scrypt, bcrypt, etc.

d. Which is more important: password complexity requirements or password expiration requirements? Why?
(1) Password complexity requirements are more important.
(2) The password expiration is dying today, here are the reasons: a) The threat model is outdated. The attacker may need 90 days to crack the password they get in the past, but now it only takes literal second in the cloud, so the regular password change is useless. b) There is a huge cost to organizations for a password expiration program. c) Changing password increases risk. If you keep forcing people to change their passwords, they are going to come up with some system where it's easy for them to remember their new passwords, for example change the number 1 at the end of the password to 2.
Most of today's average or bad passwords can be quickly cracked. It is always important to meet the password complexity requirements.

e. What is MFA? What threat model does it deal with?
(1) MFA: Multifactor authentication combines two or more independent credentials: something the individual knows, something the individual possess and something the user is or does.
(2) The goal of MFA is to create a layered defense and make it more difficult for an unauthorized person to access a target. If one factor is compromised or broken, the attacker still has at least one more barrier to breach before successfully breaking into the target.

f. List and briefly describe the principal characteristics used for biometric identification.
(1) **Facial characteristics**: The most common approach is to define characteristics based on relative location and shape of key facial features, such as eyes, eyebrows, nose, lips, and chin shape. An alternative approach is to use an infrared camera to produce a face thermogram that correlates with the underlying vascular system in the human face.
(2) **Fingerprints**: Fingerprints have been used as a means of identification for centuries, and the process has been systematized and automated particularly for law enforcement purposes. A fingerprint is the pattern of ridges and furrows on the surface of the fingertip. Fingerprints are believed to be unique across the entire human population. In practice, automated fingerprint recognition and matching system extract a number of features from the fingerprint for storage as a numerical surrogate for the full fingerprint pattern.
(3) **Hand geometry**: Hand geometry systems identify features of the hand, including shape, and lengths and widths of fingers.
(4) **Retinal pattern**: The pattern formed by veins beneath the retinal surface is unique and therefore suitable for identification. A retinal biometric system obtains a digital image of the retinal pattern by projecting a low-intensity beam of visual or infrared light into the eye.
(5) **Iris**: Another unique physical characteristic is the detailed structure of the iris.
(6) **Signature**: Each individual has a unique style of handwriting and this is reflected especially in the signature, which is typically a frequently written sequence. However, multiple signature samples from a single individual will not be identical. This complicates the task of developing a computer representation of the signature that can be matched to future samples.
(7) **Voice**: Whereas the signature style of an individual reflects not only the unique physical attributes of the writer but also the writing habit that has developed, voice patterns are more closely tied to the physical and anatomical characteristics of the speaker. Nevertheless, there is still a

variation from sample to sample over time from the same speaker, complicating the biometric recognition task.

g. Describe the general concept of a challenge-response protocol.
Challenge-response protocol is one kind of authentication protocols used with smart tokens. The computer system generates a challenge, such as a random string of numbers. The smart token generates a response based on the challenge (using hash function to the challenge combined with the user's password). For example, public-key cryptography could be used and the token could encrypt the challenge string with the token's private key.

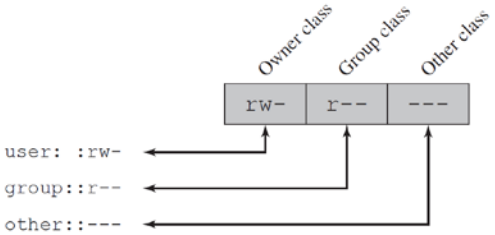
h. In a challenge-response password protocol, why is a random nonce used?
The use of random nonce which is randomly generated on each exchange (and where the response is different from the challenge) defends against replay attack, where a malicious intermediary simply records the exchanged data and retransmits it at a later time to fool one end into thinking it has authenticated a new connection attempt from the other.

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)

Nine of the protection bits specify read, write, and execute permission for the owner of the file or directories, other members of the group to which this file or directory belongs, and all other users.



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

Read permission (r) equals 4, write permission (w) equals 2, execute permission (x) equals 1.

(1) 644: rw-r--r--

Owner class is 6: 'rw-' Group class is 4: 'r--' Other class is 4: 'r--'.

(2) 750: rwxr-x---

Owner class is 7: 'rwx' Group class is 5: 'r-x' Other class is 0: '---'.

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)

I created a user hacker using “useradd hacker”, then switched user to hacker using “sudo su hacker”. And I made a directory named “hw3_testing” with permission 773.

The non-owner non-group user cannot list the contents of the directory, but such users can create new files in the directory.

```
nn75@vcm-11165:~/security/hw3$ ls -l
total 364
-rw-rw-r-- 1 nn75 nn75 126648 Oct  2 15:13 590_03_nn75_hw3.docx
-rw-rw-r-- 1 nn75 nn75 119920 Oct  2 14:48 homework3.jpeg
-rw-rw-r-- 1 nn75 nn75 119920 Oct  2 14:34 homework3.pdf
drwxrwx-wx 2 nn75 nn75      6 Oct 11 13:13 hw3_testing
nn75@vcm-11165:~/security/hw3$ sudo su hacker
$ ls
590_03_nn75_hw3.docx  homework3.jpeg  homework3.pdf  hw3_testing
$ ls hw3_testing
ls: cannot open directory 'hw3_testing': Permission denied
$ cd hw3_testing
$ ls
ls: cannot open directory '.': Permission denied
$ mkdir in_hw3_testing
$ ls
ls: cannot open directory '.': Permission denied
$ exit
nn75@vcm-11165:~/security/hw3$ ls
590_03_nn75_hw3.docx  homework3.jpeg  homework3.pdf  hw3_testing
nn75@vcm-11165:~/security/hw3$ ls hw3_testing/
in_hw3_testing
nn75@vcm-11165:~/security/hw3$
```

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	**2**	**3**	**4**	**5**	**6**	**7**
-rw-r--r--	1	root	root	1.6M	Sep 25 16:58	gosh.tar.gz
privileges for owner group and others	number of file links	owner name	group name	size(bytes) of file	last modified time	file name and type

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

chmod +w filename

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

chown -R username:groupname filename

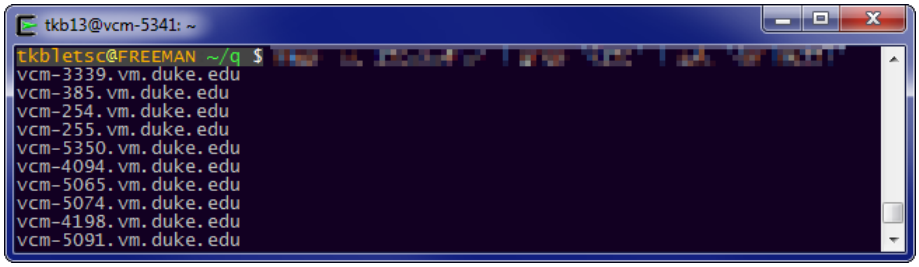
Question 3: Data processing (16 points)

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 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.

(1) nmap 152.3.64.* | grep -P 'Nmap scan report for vcm-.*' | awk '{print $5}'

(2) Screenshot

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.

(1) grep -P '.Invalid user admin from.*' auth.log | awk '{print $10}' | uniq

(2) Screenshot: (I didn't include all IPs in the screenshot)

```
nn75@vcm-11165:~/security/hw3/3_question$ grep -P ".Invalid user admin from.*" auth.log | awk '{print $10}' | 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.

(1) Download the root page and other pages recursively:

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

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

(2) Find which pages have the <form> tag:

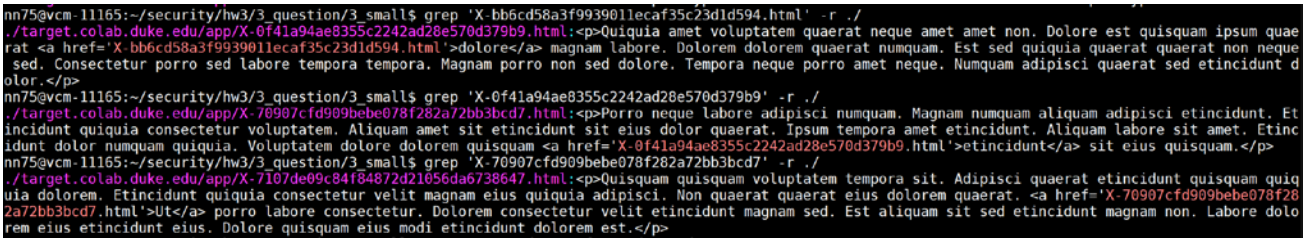
grep '<form>' -r ./

Take the first page I found as an example:

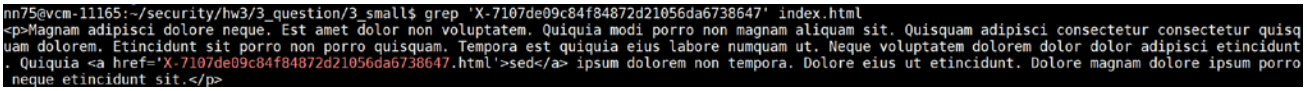
./target.colab.duke.edu/app/X-bb6cd58a3f9939011ecaf35c23d1d594.html

The next step is to find the page which has 'X-bb6cd58a3f9939011ecaf35c23d1d594':

grep 'X-bb6cd58a3f9939011ecaf35c23d1d594' -r ./



I repeated the same operation until the first HTML is found in the index.html (root page)



I got the name of the hyperlink from the search result.

The sequence of clicks to get to the first page is: sed, Ut, etincidunt, dolore (second)
The sequence of clicks to get to the first page is: ipsum, sit, numquam, est

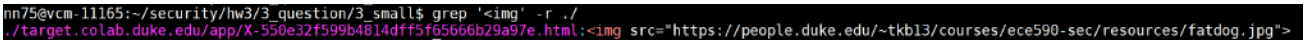
OPTIONAL: For up to 4 points of extra credit, find:

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.

A fat dog:

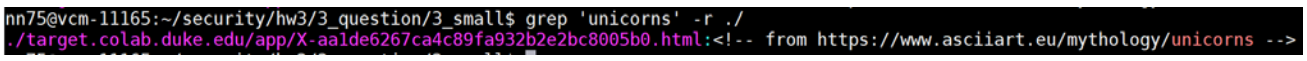
(1) grep '<img' -r ./



(2) The sequence of clicks: Velit, aliquam, porro, labore

A unicorn

(1) grep 'unicorn' -r ./



(2) The sequence of clicks: Velit, aliquam, consectetur, consectetur

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).
(1) $NetID = ~ /^[a-zA-Z]{1,3}\d+$/

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. Examples:

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

(1) $URL = ~ /^https:\/\/en.wikipedia.org\/wiki\/(.+)$/

Pi digits (3 points)

Develop a shell command to print the MD5 hash of pi (π) up to 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.

(1) I used Python and perl, here is my Python program:

```python
import re
import hashlib

f = open('./pi1000000.txt','r')
pi= f.read()
f.close()

m = re.match(r'^(3\.\d*?12345)', pi)

x = hashlib.md5(m.group(1).encode('utf-8')).hexdigest()
print('md5 hash result:' + x)
```

(2) The md5 hash result is c071744ca7b120fba43d71a219ae1402

```
nn75@vcm-11165:~/security/hw3/4_question$ python3 pi.py
md5 hash result:c071744ca7b120fba43d71a219ae1402
```

Question 5: GNU Screen (2 points)

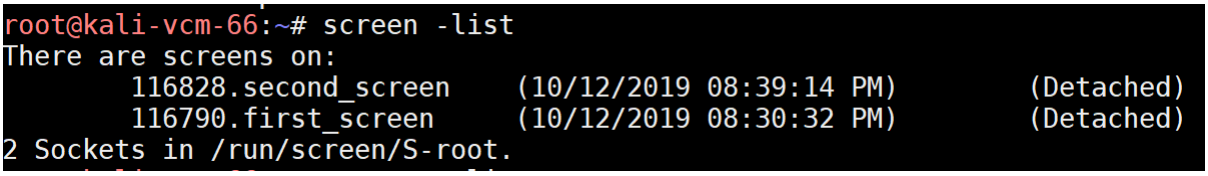
Screen is a full-screen window manager that multiplexes a physical terminal (or GUI terminal window) between several processes, typically interactive shells. When screen is called, it creates a single terminal with a shell in it (or the specified command) and then gets out of your way so that you can 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.



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

There is an FTP server called Serv-U. It is actually 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 has 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 -m 20 -a 3 test.hash -o test.out
(2) Screenshot of the output: (next page)

```
root@kali-vcm-66:~/Documents/hw3/6_question# hashcat --force -m 20 -a 3 test.hash -o test.out
hashcat (v5.1.0) starting...

OpenCL Platform #1: The pocl project
====================================
* Device #1: pthread-Intel(R) Xeon(R) CPU E5-2697 v3 @ 2.60GHz, 1024/2943 MB allocatable, 2MCU

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

Applicable optimizers:
* Zero-Byte
* Early-Skip
* Not-Iterated
* Single-Hash
* Single-Salt
* Brute-Force
* Raw-Hash

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

ATTENTION! Pure (unoptimized) OpenCL kernels selected.
This enables cracking passwords and salts > length 32 but for the price of drastically reduced perfor
If you want to switch to optimized OpenCL kernels, append -O to your commandline.

Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.

* Device #1: build_opts '-cl-std=CL1.2 -I OpenCL -I /usr/share/hashcat/OpenCL -D LOCAL_MEM_TYPE=2 -D
E=4 -D DEVICE_TYPE=2 -D DGST_R0=0 -D DGST_R1=3 -D DGST_R2=2 -D DGST_R3=1 -D DGST_ELEM=4 -D KERN_TYPE=
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.Type........: md5($salt.$pass)
Hash.Target......: 5cf2ff593419bcf3d22c62e65a82128a:ly
Time.Started.....: Sun Oct 13 01:02:43 2019 (1 sec)
Time.Estimated...: Sun Oct 13 01:02:44 2019 (0 secs)
Guess.Mask.......: ?1 [1]
Guess.Charset....: -1 ?l?d?u, -2 ?l?d, -3 ?l?d*!$@_, -4 Undefined
Guess.Queue......: 1/15 (6.67%)
Speed.#1.........:      284 H/s (0.02ms) @ Accel:1024 Loops:62 Thr:1 Vec:4
Recovered........: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.........: 62/62 (100.00%)
Rejected.........: 0/62 (0.00%)
```

```
Session..........: hashcat
Status...........: Cracked
Hash.Type........: md5($salt.$pass)
Hash.Target......: 5cf2ff593419bcf3d22c62e65a82128a:ly
Time.Started.....: Sun Oct 13 01:02:49 2019 (12 secs)
Time.Estimated...: Sun Oct 13 01:03:01 2019 (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.........: 26137.7 kH/s (9.53ms) @ 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: Sun Oct 13 01:02:41 2019
Stopped: Sun Oct 13 01:03:02 2019
root@kali-vcm-66:~/Documents/hw3/6_question# ls
test.hash  test.hash~  test.out
root@kali-vcm-66:~/Documents/hw3/6_question# cat test.out
5cf2ff593419bcf3d22c62e65a82128a:ly:manito
```

(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. Use straight attack mode crack the first password generated by function md5(salt+pass)
The link of wordlist (bt4-password.txt) :

https://github.com/danielmiessler/SecLists/raw/master/Passwords/bt4-password.txt

Instruction:

hashcat --force -a 0 -m 20 opa.hash bt4-password.txt -o opa.out

Result:

e39945197dbc380a5ac611fbbb7b5354:qj:baseball

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)**

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.

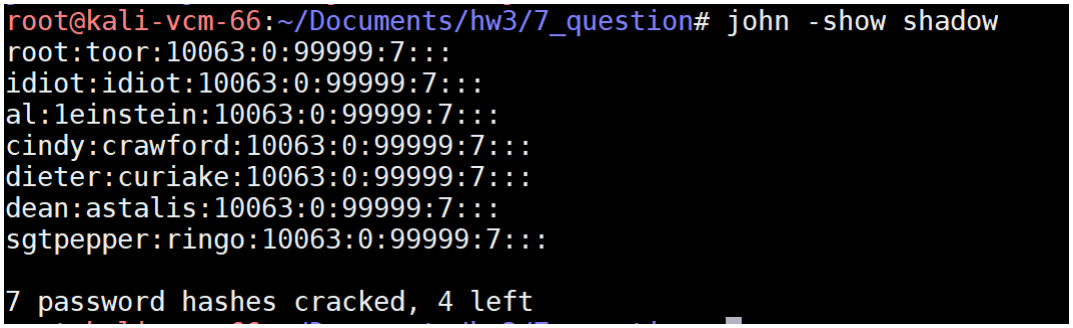
Paste the output your results with the following command "`john -show shadow`". (6 points)

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?
SMS-based two-factor authentication

b. What are two ways in which the attacker may have gained access to the second factor?
(1) SIM-swap
(2) mobile number port-out scams

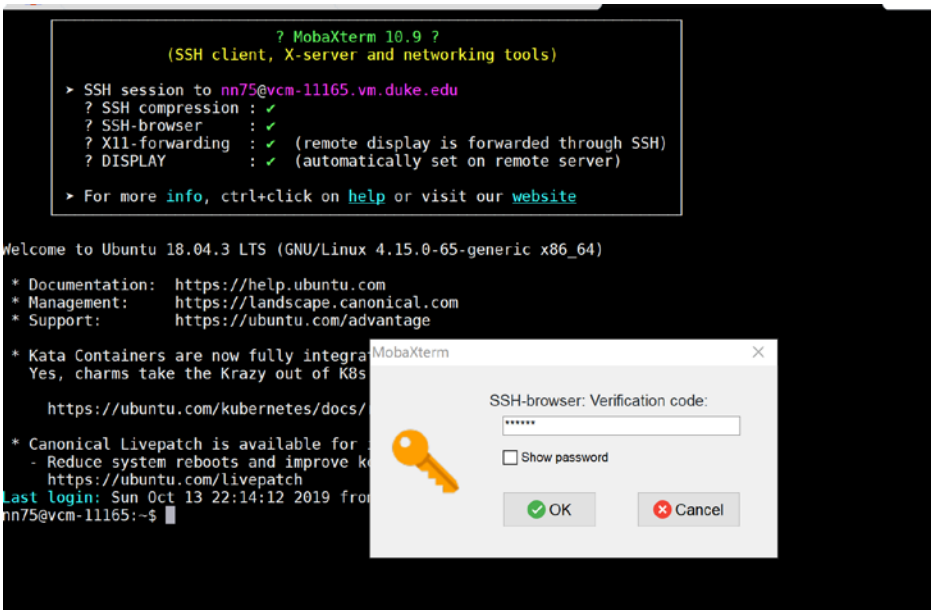
c. What are two alternative forms of MFA that could be deployed instead?
(1) mobile app: Google Authenticator or Authy
(2) Use of hardware-based security keys.

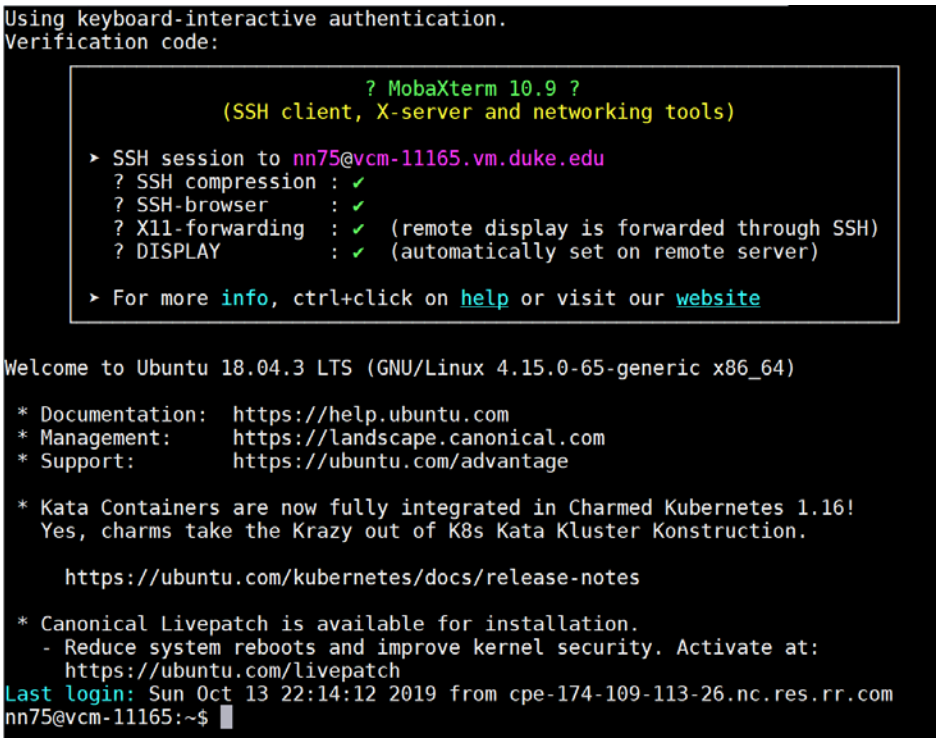
d. Identify another attack in the last 5 years in which SMS-based MFA was compromised.
Link: https://www.scmagazineuk.com/criminals-hit-metro-bank-multi-factor-authentication-bypass-ss7-attack/article/1524670
In this attack, criminals hit Metro Bank with multi-factor authentication bypass SS7 attack. UK-based Metro Bank has reportedly fallen victim to a 2FA bypass attack after cyber-criminals were able to compromise a known flaw in the SS7 protocol. Signalling System No. 7 (SS7) was developed in 1975 to perform essential operations on the public switched telephone network (PSTN). Among those are operations associated with transmitting Short Message Service (SMS) messages which are today often used in 2FA systems. During attack, a known telecommunications vulnerability being exploited to target bank accounts by intercepting SMS text messages used as two-factor authentication. We can say the SMS-based MFA was compromised in this attack.

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.

After successfully configuring Google Authenticator on my VM, I got the QR code and other verification information. When login the Linux VM, I first entered the verification code given by Authenticator app, and then entered the verification code generated by google authenticator.





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.
shypothetical
ssamplesberg
jexampleface
susanh
scotts
jacobe
susan
scott
jacob

Attack (4 points)

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

Give the hydra command you used below.

hydra -L /root/Documents/hw3/10_question/username.txt -P /usr/share/wordlists/metasploit/adobe_top100_pass.txt -t 8 target.colab.duke.edu ssh

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

```
root@kali-vcm-66:~/Documents/hw3/10_question# hydra -L /root/Documents/hw3/10_question/username.txt -P /usr/share/wordlists/metasploit/adobe_top100_pass.t
xt -t 8 target.colab.duke.edu ssh
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2019-10-17 16:56:19
[DATA] max 8 tasks per 1 server, overall 8 tasks, 300 login tries (l:3/p:100), ~38 tries per task
[DATA] attacking ssh://target.colab.duke.edu:22/
[STATUS] 88.00 tries/min, 88 tries in 00:01h, 212 to do in 00:03h, 8 active
[22][ssh] host: target.colab.duke.edu   login: scott   password: zxcvbnm
[STATUS] 83.33 tries/min, 250 tries in 00:03h, 50 to do in 00:01h, 8 active
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2019-10-17 17:00:05
```

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.**

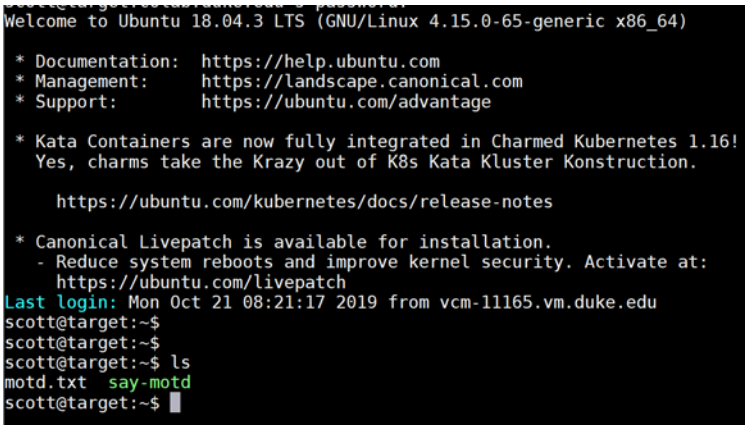
(1) Command:

ssh scott@target.colab.duke.edu bash -i

or

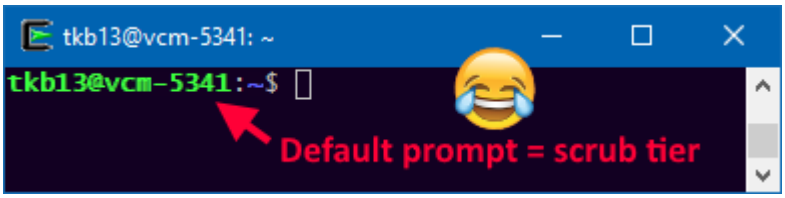
ssh -t scott@target.colab.duke.edu "rm ~/.bash_profile"

(2) Screenshot:



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.



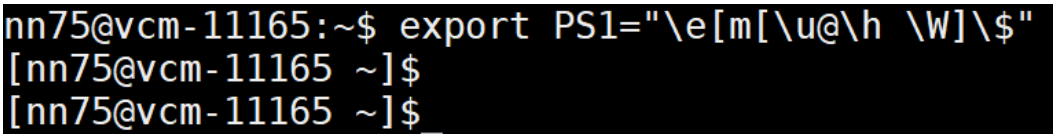
Using your understanding of shell color codes, develop a custom prompt by modifying the PS1 environment variable. You'll need to mark the escape codes for the shell using `\[` and `\]` indicators (this the shell needs to know what characters are printed vs. interpreted for cursor control purposes); [details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

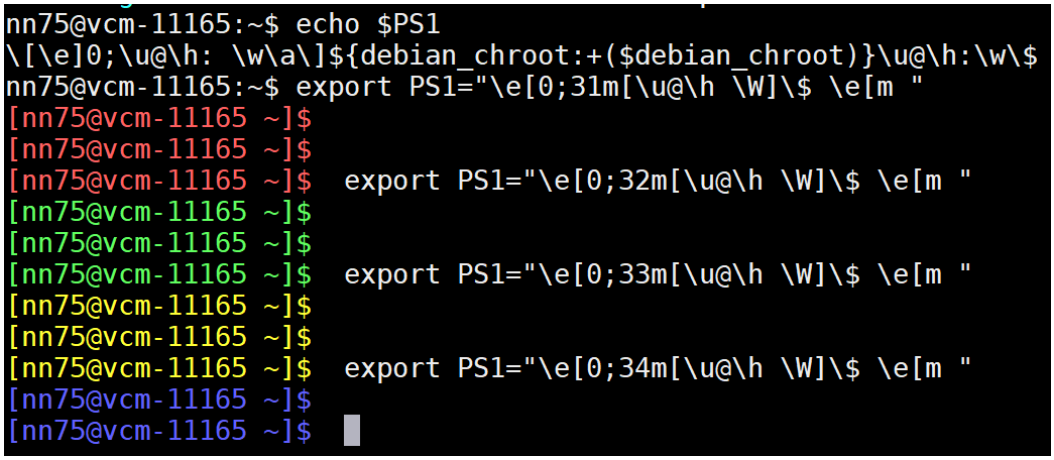
Paste your PS1 environment variable command & a screenshot of the resulting prompt.

(1) Command:

```
export PS1="\e[m[\u@\h \W]\$"
```

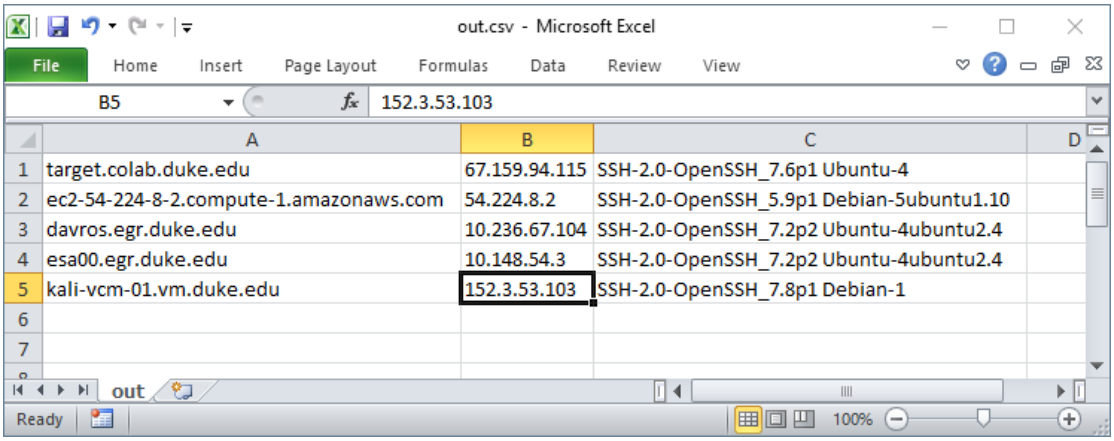
(2) Screenshot:





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
144.202.30.145
155.138.231.109
155.138.195.55
davros.egr.duke.edu
storemaster.egr.duke.edu
kali-vcm-41.vm.duke.edu
```

(Do not scan machines other than the above. All of the above are machines are 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!

```
#! /bin/bash

filename="hostlist"
while read line; do

    var1=$(nmap  -sV -p 22 $line | perl -ne '/^Nmap scan report for (.*) \((.*)\)/ and print "$1\n"')
    var2=$(nmap  -sV -p 22 $line | perl -ne '/^Nmap scan report for (.*) \((.*)\)/ and print "$2\n"')
    var3=$(sudo scanssh -s ssh $line | perl -ne '/(SSH-.*)/ and print "$1\n"')

    echo -e "$var1:$var2:$var3" >> out.csv

done < "$filename"
```

When loading the out.csv into Excel, choose "colon" as separator:

Column1	Column2	Column3
target.colab.duke.edu	152.3.69.23	SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3
144.202.30.145.vultr.com	144.202.30.145	SSH-2.0-OpenSSH_7.5 FreeBSD-20170903
155.138.231.109.vultr.com	155.138.231.109	SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3
155.138.195.55.vultr.com	155.138.195.55	SSH-2.0-OpenSSH_7.9
davros.egr.duke.edu	10.236.67.104	SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.8
storemaster.egr.duke.edu	10.148.54.108	SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.8
kali-vcm-41.vm.duke.edu	10.138.16.119	SSH-2.0-OpenSSH_8.0p1 Debian-4

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)**
- **Identify which aspects of the CIA triad are at play and how. (2)**
- **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. (3)

(1) **Summary**: The XMRig Monero mining malware infected airport. This malware skated by antivirus solutions on the endpoints by adding a new tweak. The modification was really simple: the MD5 was modified, however, the attacker kept the use of the original tools and even the original file names. Its business impact quality of service and service interruptions, as well as a significant increase in power consumption throughout the airport. The remote user stealthily accessed the network by launch an application named player.exe and then injected malware.
Link: https://threatpost.com/major-airport-malware-attack-ot-security/149330/

(2) **Availability**: The mining malware impacts quality of service and service interruptions, as well as a significant increase in power consumption.
Integrity: The MD5 was modified.
Confidentiality: The attacker kept the use of the original tools and even the original file names.

(3) **Assets at risk:** the system of airport

Vulnerability at play: the airport network can be accessed by launching an evil application at the remote, hard for detection products to detect this malware.
Attacker's capabilities and knowledge: access network stealthily at the remote, modify MD5, impact quality of service.

(4) Describe a defense: Detect the illegal launch of the application and prevent the launch, so that the attackers are unable to inject the malware. Also detect and clean the malware in the infected system.

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 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.

(1) Compress file:

```
zip nn75.zip fat_dog.jpg nn75_ssh_public_key
```

(2) Create PDF+ZIP polyglot:

```
cat hw3.pdf nn75.zip > 590_03_nn75_hw3.pdf
```

(3) Unzip the polyglot:

```
zip -A 590_03_nn75_hw3.pdf
```