ECE 560 Homework 4

Guanhua Chen

TOTAL POINTS

99.5 / 100

QUESTION 1

1 Question 1: Show your work from Question 0 2 / 2

✓ **- 0 pts** **Correct**

+ 5 pts Extra-credit

+ 3.5 pts Extra credit, but not showing code work for it

- 1 pts Click here to replace this description.

- 2 pts Click here to replace this description.

QUESTION 2

2 Question 2: Buffer Overflow 10 / 10

✓ **- 0 pts** **Correct**

- 2 pts Incorrect output or bad return

- 10 pts Did not attempt or incorrect attack (your buffer must print hax0red in addition to the "<name> is cool" message).

QUESTION 3

3 Question 3: Buffer Overflow Prevention 5 / 5

✓ **- 0 pts** **Correct**

- 5 pts Click here to replace this description.

QUESTION 4

4 Question 4: SQL Injection 6 / 6

✓ **- 0 pts** **Correct**

- 6 pts Incorrect

+ 5 pts Click here to replace this description.

+ 10 pts Click here to replace this description.

+ 8 pts Click here to replace this description.

QUESTION 5

5 Question 5: Backup with rsnapshot 8 / 8

✓ **- 0 pts** **Correct**

- 1 pts Didn't set SSH keys

QUESTION 6

6 Question 6: Run a honeypot and see what you get 8 / 8

✓ **- 0 pts** **Correct**

- 2 pts Error in Setup

- 4 pts No Result

- 3 pts Lack Analysis for Results

QUESTION 7

7 Question 7: Denial of Service attack using TorsHammer 5.5 / 6

- 0 pts Correct

- 1 pts Didn't explain why CPU usage did not increase much

- 1 pts Wrong time during attack

- 2 pts Wrong report on time

- 0.5 **Point adjustment**

The CPU usage does not increase much is that slow POST attack does not send computation demanding tasks. It just send charactars very slow. Apache may take CPU, in some cases.

QUESTION 8

8 Question 8: Shell Practice 9 / 9

✓ **- 0 pts** **Correct**

- 0.5 pts Didn't provide the command that downloads the image file.

- 0.5 pts Didn't deal with url pointing to non-existing vcms (ping return "unknown")

- 0.5 pts The "too compressable" message is not complete. The following ratio stuff would be prented, too.

- 1 pts Wrong magic number

QUESTION 9

9 Question 9: Analyze forensic server packet capture network logs 10 / 10

✓ **- 0 pts Correct**

- 2 pts Mistake on External Tool Download

- 10 pts No answer

- 1.5 pts The external tools are downloaded via FTP

QUESTION 10

10 Question 10: Analyze forensic server hacker package 8 / 8

✓ **- 0 pts Correct**

- 1 pts Click here to replace this description.

QUESTION 11

11 Question 11: Hardware level attacks 4 / 4

✓ **- 0 pts Correct**

- 1 pts Part A incorrect

- 1 pts Part B incorrect

- 1 pts Part C incorrect. Answer: Power consumption

- 1 pts Part D incorrect. Answer: Storing hashes of the password instead of the password itself

QUESTION 12

12 Question 12: A practical man-in-the-middle hardware attack 6 / 6

✓ **- 0 pts Correct**

- 1 pts a) Incorrect

- 1 pts b) Incorrect. Answer: It is used for detecting if the lap board is being tampered with while in operation

- 1 pts c) Incorrect

- 1 pts d) Incorrect

- 1 pts e) Incorrect

- 1 pts f) Incorrect. Answer: HTTPS is encrypted

QUESTION 13

13 Question 13: Malware Analysis 18 / 18

✓ **- 0 pts Correct**

- 1 pts Doesn't explain how malware hides itself

- 2 pts Doesn't explain that the hex strings are hashes

- 1 pts Doesn't explain how malware remains persistent

- 2 pts Doesn't (correctly) address how malware could have survived. Part 6, question 24 needs a more detailed answer.

- 1 pts Doesn't address whether analysis was worrying

- 1 pts Doesn't address feelings about 62/72 scanners detecting malware

- 2 pts Doesn't address what CurrentVersion\\Registry keys do

- 2 pts No answer for Part 5, question 18.

- 1 pts No answer Part 6, question 21

- 1 pts No description of actions taken. Part 6. Question 23

- 2 pts No screenshot indicating new malware path

- 2 pts No screenshot of process explorer. Part 3, question 7

- 1 pts No answer. Part 6, question 25

- 1 pts No answer. Part 6, question 26

- 18 pts No answer

- 16 pts One question answered

QUESTION 14

14 Late Penalty 0 / 0

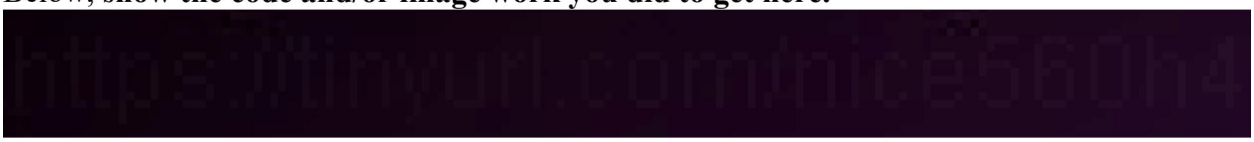
✓ **- 0 pts No penalty**

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

To get here, you found the URL via one of two steganography methods. Did you find this hint file along the way?

Question 1: Show your work from Question 0 (2 points)

Below, **show the code and/or image work you did to get here.**



watermark

OPTIONAL: For 5 points of extra credit, give the secret message included with the URL in the bitwise-encoded message. See the hints file for details.

Question 2: Buffer Overflow (10 points)

This question is highly flexible, so read carefully!

Here's how to disable ASLR. This article in the highly esteemed blog publication "m0skit0.org" explains how to turn off W^X (also known as NX support) at compile time (done by default by the Makefile in the starter kit linked below).

compile flag: `-fno-stack-protector -z execstack`

Here's the base problem:

- **Prepare VM:** In your Linux VM, install nasm ("sudo apt install nasm").
  - Note: If you want to build an attack for another OS, contact the instructor for permission, which I'll likely give, but you'll be on your own.
- **Get set up:** Download, extract, and test the **example exploit kit**.
  - The Makefile is set up to build the vulnerable program with W^X disabled.
  - Read through the vulnerable program "vuln1.c" and the attack script "attack.asm" closely.
  - **Watch this walkthrough video I recorded.**
    NOTE: This video was recorded for an earlier version of the problem, so there are slight differences, in particular, it claims that we need to avoid null bytes in the attack buffer, this is actually not true for a gets() exploit.
  - The included attack will make the program skip saying "Bye!" and exit with status code 5 instead of the usual 0. Recall that the exit status of a program can be checked by running "echo $?" after the program exits.

1 Question 1: Show your work from Question 0 **2 / 2**

✓ **- 0 pts** **Correct**

+ 5 pts Extra-credit

+ 3.5 pts Extra credit, but not showing code work for it

- 1 pts Click here to replace this description.

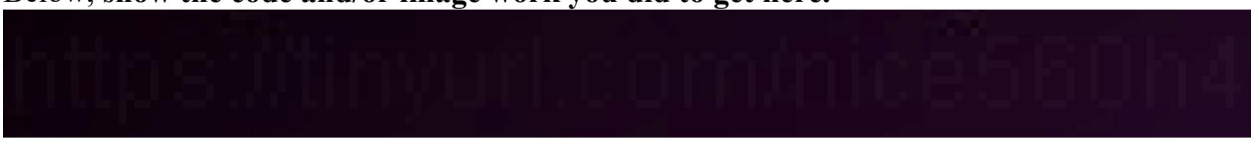
- 2 pts Click here to replace this description.

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

To get here, you found the URL via one of two steganography methods. Did you find this hint file along the way?

Question 1: Show your work from Question 0 (2 points)

Below, **show the code and/or image work you did to get here.**



watermark

OPTIONAL: For 5 points of extra credit, give the secret message included with the URL in the bitwise-encoded message. See the hints file for details.

Question 2: Buffer Overflow (10 points)

This question is highly flexible, so read carefully!

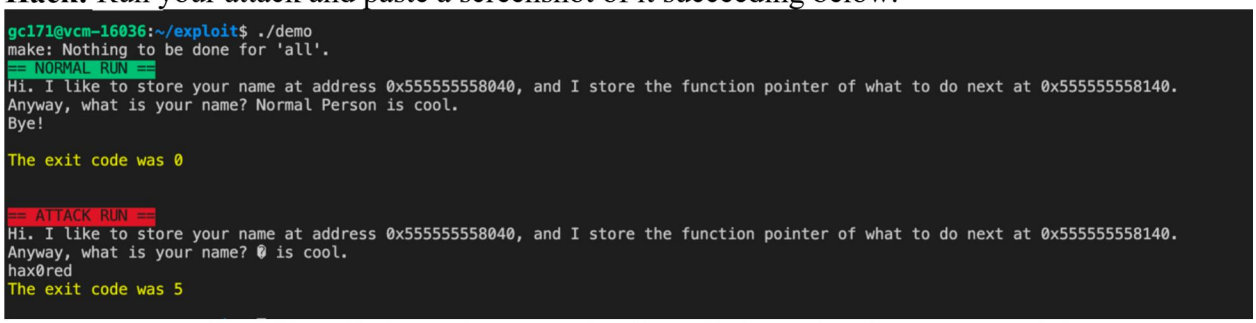
Here's how to disable ASLR. This article in the highly esteemed blog publication "m0skit0.org" explains how to turn off W^X (also known as NX support) at compile time (done by default by the Makefile in the starter kit linked below).

compile flag: `-fno-stack-protector -z execstack`

Here's the base problem:

- **Prepare VM:** In your Linux VM, install nasm ("sudo apt install nasm").
  - Note: If you want to build an attack for another OS, contact the instructor for permission, which I'll likely give, but you'll be on your own.
- **Get set up:** Download, extract, and test the **example exploit kit**.
  - The Makefile is set up to build the vulnerable program with W^X disabled.
  - Read through the vulnerable program "vuln1.c" and the attack script "attack.asm" closely.
  - **Watch this walkthrough video I recorded.**
    NOTE: This video was recorded for an earlier version of the problem, so there are slight differences, in particular, it claims that we need to avoid null bytes in the attack buffer, this is actually not true for a gets() exploit.
  - The included attack will make the program skip saying "Bye!" and exit with status code 5 instead of the usual 0. Recall that the exit status of a program can be checked by running "echo $?" after the program exits.

    - Assuming your VM's memory map looks like mine, you should be able to run the attack straight away by executing the included "demo" script. If it doesn't work, you may need to update the memory location constants in the attack script, re-build, and try again.
- **Make exploit:** Develop an attack buffer that makes the program print "`hax0red`".
    - Note: your *exploit* must cause it to say this -- merely having the word "`hax0red`" appear among the program's usual "so-and-so is cool" output does not count.
    - You can use any tool you wish to develop the attack buffer, though using the included NASM attack.asm as a starting point is probably easiest.
    - gdb is your friend.
    - If pursuing the extra credit (see below), you may target a program *other than* the provided vulnerable program.
- **Hack:** Run your attack and paste a screenshot of it succeeding below.



- **Document:** Make a file called "readme.txt" that lists all dependencies your attack has (packages needed to be installed, special steps to be taken, including the steps to disable ASLR and W^X)).
- **Submit:** Gather up *EVERYTHING* involved in the attack (readme.txt, the vulnerable program source and binary, the attack source code and attack binary file, and any Makefiles or helper scripts). Zip all of this up and submit it to Sakai as "<NetID>_attack.zip".

Here's some random tips to help you:

- GDB quick reference card.
- Syscall numbers. Alternate source with register indicators.
- Intel x86 assembly reference:
    - The giant Intel big book: the authoritative source.
    - The felixcloutier x86 instruction reference.
    - NASM manual, including instruction list.
    - Low-level conversions between hex bytes and CPU instructions: Numeric order, mnemonic order.
- You can use "`ndisasm -b64 <file>`" to do a raw disassembly of your attack buffer, with output in Intel (NASM) syntax.
- You can use "`hd <file>`" to get a hexdump of it, which is useful when looking for nulls or whitespace bytes that snuck into your attack buffer.

2 Question 2: Buffer Overflow **10 / 10**

✓ **- 0 pts Correct**

- 2 pts Incorrect output or bad return

- 10 pts Did not attempt or incorrect attack (your buffer must print hax0red in addition to the "<name> is cool" message).

- [ ] (+5) **Big math hack:** Same as above, but the first 100 Fibonacci numbers, and they must be to full precision. (Note: this exceeds a 64-bit register, so you'll have to get creative)
- [ ] (+5) **File IO:** If your payload causes the program to appear to the user to function normally in every way, except the message you're printing is saved to a file called "0wned.txt". (Note the zero in the filename -- this is used to indicate that we are very cool, mature people.)
- [ ] (+10) **Reverse shell:** If your payload connects out to a TCP host and, upon successful connection, establishes remote control of a /bin/bash shell.
- [ ] (+10) **Code reuse:** If your payload operates *entirely* on code reuse and does no code injection.
- [ ] (+15) **Dr. Bletsch's dissertation:** If you payload operates on code reuse and does no code injection, *and* does not exploit a single `ret` instruction.

Extra credit wildcard:

- [ ] (+$1000) **Overachiever**: If you unlock every extra credit achievement above, I will hand you a thousand dollars.
- [ ] (+?) **Wildcard:** If you're doing something else fancy with your attack that you feel is worth more points, pitch it to the instructor. You will need to document this additional feature in your readme file.

Question 3: Buffer Overflow Prevention (5 points)

Fix it: Take whatever program you used as your target for the previous problem and fix the vulnerability. Submit the fixed code as "<NetID>_fixed.zip" to Sakai.

use fgets

```
fgets(stuff.name, 256, stdin);
```

loop to remove \n

```
// remove \n
    for (int i = 0; stuff.name[i] != '\0'; i++)
    {
        if (stuff.name[i] == '\n')
        {
            stuff.name[i] = '\0';
            break;
        }
    }
```

Prove you actually fixed it: Paste a screenshot of the program NOT being exploited by the attack input from the previous question. Note: If a screenshot cannot show the improvement for your particular program, contact the instructor to discuss an alternative form of response.

```
== NORMAL RUN ==
Hi. I like to store your name at address 0x555555558040, and I store the function pointer of what to do next at 0x5555555
58140.
Anyway, what is your name? Normal Person is cool.
Bye!

The exit code was 0


== ATTACK RUN ==
Hi. I like to store your name at address 0x555555558040, and I store the function pointer of what to do next at 0x5555555
58140.
Anyway, what is your name? @ is cool.
hax0red
The exit code was 5

== fixed RUN ==
Hi. I like to store your name at address 0x555555558040, and I store the function pointer of what to do next at 0x5555555
58140.
Anyway, what is your name? @ is cool.
Bye!

The exit code was 0

rm vuln1 attack.bin fixed
```

Prove you didn't make it worse: Paste a screenshot of the program continuing to function correctly under normal input.

```
gc171@vcm-16036:~/exploit$ ./fixed
Hi. I like to store your name at address 0x555555558040, and I store the function pointer of what to do next at 0x5555555
58140.
Anyway, what is your name? gc171
gc171 is cool.
Bye!
gc171@vcm-16036:~/exploit$
```

3 Question 3: Buffer Overflow Prevention 5 / 5

✓ **- 0 pts** **Correct**

- 5 pts Click here to replace this description.

Question 4: SQL Injection (6 points)

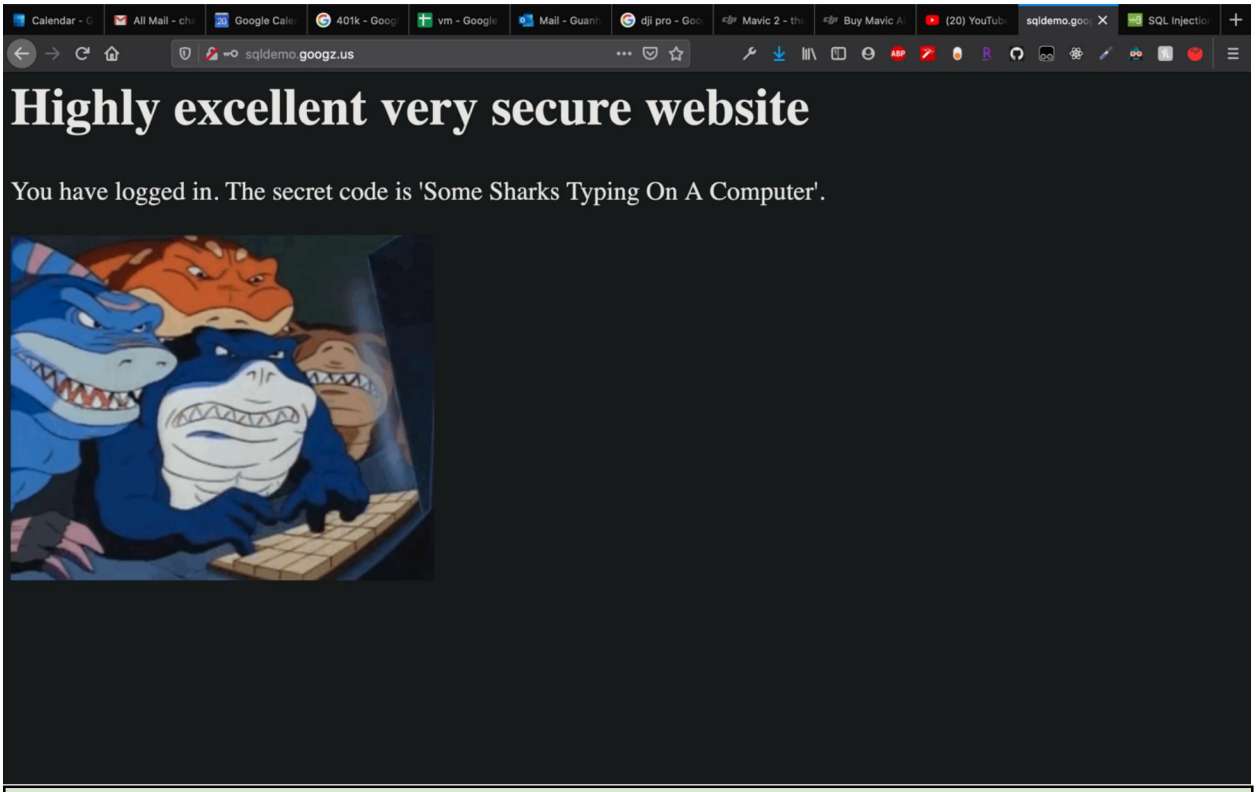
Use a SQL injection attack to login to http://sqldemo.googz.us/. This is a simple login page written in PHP and modeled after the many intro-to-PHP guides available online.

Note: do not modify the database in any way; just use the attack to fool the script into logging you in. Also, this is running on a production web host, so do not use brute force techniques in your attack.

Paste the username and password you used below.

' or 0=0;#
123

Upon successful login, the system will give you a secret code. Paste this code below.
Some Sharks Typing On A Computer



Extra credit +5: Find out jimmy's full name.

Extra credit +5: Find out the actual admin password.

Note: If you wish to use a tool or script to make successful requests in pursuit of the extra credit, email me first. If such a request is granted, you will need to throttle your attack to no more than one request every five (5) seconds. No tool is needed to get the main part of the question.

4 Question 4: SQL Injection 6 / 6

✓ **- 0 pts** **Correct**

- 6 pts Incorrect

+ 5 pts Click here to replace this description.

+ 10 pts Click here to replace this description.

+ 8 pts Click here to replace this description.

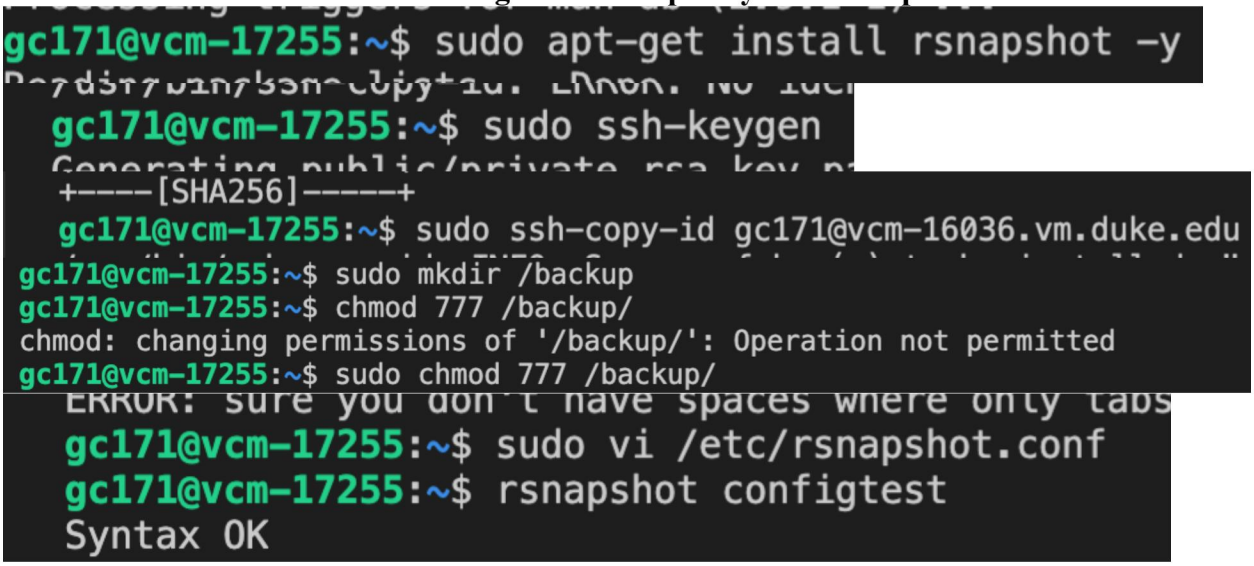
Question 5: Backups with rsnapshot (8 points)

*NOTE: This question is similar to one posed later in my Enterprise Storage Architecture course. If you have taken that course in a previous semester, you may simply write "I did this in Enterprise Storage Architecture in <semester+year>" for full credit (though you can do it again if you wish). If you are enrolled in that course now, it gets a little tricky, since you encounter it in group lab work. In that case, you must do it separately *yourself* for this class. This ensures you don't turn in a group member's contribution in that class for credit in this one. To be clear, in any case, you may not simply paste in a solution from the other course for this question.*

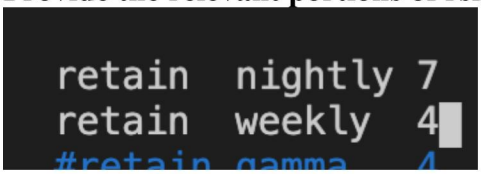
Let's build an automated backup system. Make an additional Linux VM in Duke VCM. This new Linux VM we'll refer to as the **Backup Server**. The thing we'll be backing up is your pre-existing Linux VM; this is the **Backup Client**, You can use this guide, and note the following:

- **Backup source**: We're going to back up your user home directory on the backup client machine. This is `/home/<NETID>`.
- **Backup destination**: As root, make a directory `/backup`.
- **SSH keys needed**: Because the source and destination are different machines, you'll need to set up SSH keys so that the backup server can *pull* data as needed.
- **Frequency**: Set up nightly and weekly intervals.
- **Retention**: You should retain seven nightly backups and four weekly backups.
- **Automation:** You do NOT need to do cron-based automation. Once you have rsnapshot configured, you can just run "rsnapshot nightly" to perform the backup that would happen nightly, and "rsnapshot weekly" to perform the backup that would happen weekly. You can run these commands as often as you wish; there's nothing in the software that actually needs the backups to be done nightly/weekly as opposed to every few dozen seconds (i.e., rsnapshot does not actually care that they're called "nightly" or "weekly").

Provide screenshots or a terminal log of these steps in your writeup.



Provide the relevant portions of rsnapshot.conf.



```
#
cmd_ssh /usr/bin/ssh

#backup_exec    /bin/date "+ backup of example.com started a
backup  gc171@vcm-16036.vm.duke.edu:/home/gc171 gc171/
#backup root@example.com:/etc/  example.com/    exclude=mtab
snapshot_root   /backup
```

Let's test it. Open one terminal window as root, and another terminal window as the non-root user. Do the following, **and for each step, show the process via screenshots or terminal logs. Label or otherwise identify each step you're doing in your screenshot/log.**

1. As the user on the backup client, add some content to the user home directory.

```
gc171@vcm-16036:~$ touch file1
gc171@vcm-16036:~$ touch file2
gc171@vcm-16036:~$ touch file3
gc171@vcm-16036:~$
```

2. As root on the backup server, do several nightly and weekly backups to populate your backups.

```
gc171@vcm-17255:~$ sudo rsnapshot nightly
gc171@vcm-17255:~$ sudo rsnapshot weekly
/backup/nightly.6 not present (yet), nothing to copy
gc171@vcm-17255:~$ sudo rsnapshot nightly
gc171@vcm-17255:~$ sudo rsnapshot nightly
gc171@vcm-17255:~$ sudo rsnapshot nightly
gc171@vcm-17255:~$ sudo rsnapshot nightly
gc171@vcm-17255:~$ sudo rsnapshot weekly
gc171@vcm-17255:~$
```

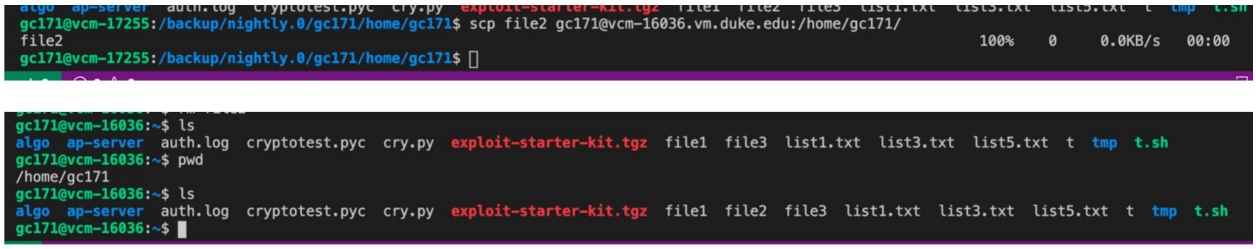
3. As the user on the backup client, "corrupt" an important file (overwrite or delete it).

```
gc171@vcm-16036:~$ rm file2
gc171@vcm-16036:~$
```

4. As root on the backup server, take another few nightly backups to simulate time passing.

```
gc171@vcm-17255:~$ sudo rsnapshot nightly
gc171@vcm-17255:~$ sudo rsnapshot nightly
gc171@vcm-17255:~$
```

5. Copy the appropriate backup from the backup server to the client's home directory, thus restoring the damaged file.



Note: If your SSH MFA is interfering with this, you may either make a separate user without MFA enabled, or disable MFA altogether.

Question 6: Run a honeypot and see what you get (8 points)

A honeypot is an intentionally-vulnerable system with monitoring. It is used to see what attackers do when they succeed in a given environment. There are many kinds of honeypots (Windows/RDP, Linux/SSH, etc.). They can be divided into "real environment" (i.e., the attacker is really breaking in, but the system they're gaining access to is one we don't care about) versus "simulated environment" (i.e., it looks like the real thing, but every operation is simulated and sandboxed).

Setup (4pts)

We'll be setting up a simulated-environment SSH sandbox called Cowrie.

Because Duke IT has a lot of existing automated defenses, for best results, you'll be installing to the Amazon cloud as an EC2 instance. As far as money goes, you can use the AWS Free Tier or the AWS Educate program, or just give them the ~$5 this experiment should cost. You may use another commercial cloud (Linode, Vultr, Digital Ocean, Azure, etc.) if you wish. You may want to play with what geographic area you place your VM, as this can affect what kind of attacks you see.

Set up your VM with Cowrie listening on port 22 and the real SSH daemon listening on port 60022. Be sure your cloud's network settings allow access on both ports.

Show the steps needed to set up this environment.

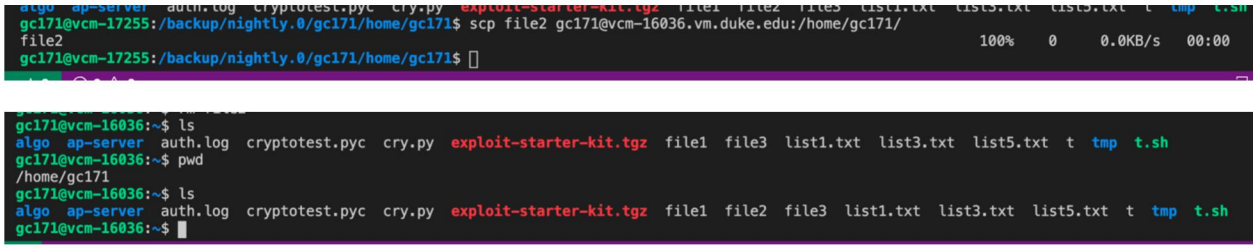
```
sudo vi /etc/ssh/sshd_config
sudo yum install git python-virtualenv libssl-dev libffi-dev build-essential
libpython3-dev python3 authbind virtualenv -y
git clone http://github.com/cowrie/cowrie
cd cowrie
vi etc/cowrie.cfg
pip install --upgrade pip
pip install --upgrade -r requirements.txt
bin/cowrie start
```

Login to your own honeypot and look around. Can you tell it's simulated?
no

5 Question 5: Backup with rsnapshot **8 / 8**

✓ **- 0 pts Correct**

- 1 pts Didn't set SSH keys



Note: If your SSH MFA is interfering with this, you may either make a separate user without MFA enabled, or disable MFA altogether.

Question 6: Run a honeypot and see what you get (8 points)

A honeypot is an intentionally-vulnerable system with monitoring. It is used to see what attackers do when they succeed in a given environment. There are many kinds of honeypots (Windows/RDP, Linux/SSH, etc.). They can be divided into "real environment" (i.e., the attacker is really breaking in, but the system they're gaining access to is one we don't care about) versus "simulated environment" (i.e., it looks like the real thing, but every operation is simulated and sandboxed).

Setup (4pts)

We'll be setting up a simulated-environment SSH sandbox called Cowrie.

Because Duke IT has a lot of existing automated defenses, for best results, you'll be installing to the Amazon cloud as an EC2 instance. As far as money goes, you can use the AWS Free Tier or the AWS Educate program, or just give them the ~$5 this experiment should cost. You may use another commercial cloud (Linode, Vultr, Digital Ocean, Azure, etc.) if you wish. You may want to play with what geographic area you place your VM, as this can affect what kind of attacks you see.

Set up your VM with Cowrie listening on port 22 and the real SSH daemon listening on port 60022. Be sure your cloud's network settings allow access on both ports.

Show the steps needed to set up this environment.

```
sudo vi /etc/ssh/sshd_config
sudo yum install git python-virtualenv libssl-dev libffi-dev build-essential
libpython3-dev python3 authbind virtualenv -y
git clone http://github.com/cowrie/cowrie
cd cowrie
vi etc/cowrie.cfg
pip install --upgrade pip
pip install --upgrade -r requirements.txt
bin/cowrie start
```

Login to your own honeypot and look around. Can you tell it's simulated?
no

```
root@18.222.254.81's password:

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@svr04:~# ls
root@svr04:~# mkdir test
root@svr04:~#
```

From outside the honeypot, show that you can find logs of your attempted "intrusion".

log

```
2020-10-18T21:09:49.234636Z [SSHService b'ssh-userauth' on HoneyPotSSHTransport,5,107.15.246.237] b'root' trying auth b'password'
2020-10-18T21:09:49.235065Z [SSHService b'ssh-userauth' on HoneyPotSSHTransport,5,107.15.246.237] Could not read etc/userdb.txt, default database
 activated
2020-10-18T21:09:49.235274Z [SSHService b'ssh-userauth' on HoneyPotSSHTransport,5,107.15.246.237] login attempt [b'root'/b'2323'] succeeded
2020-10-18T21:09:49.235956Z [SSHService b'ssh-userauth' on HoneyPotSSHTransport,5,107.15.246.237] Initialized emulated server as architecture: li
nux-x64-lsb
2020-10-18T21:09:49.236344Z [SSHService b'ssh-userauth' on HoneyPotSSHTransport,5,107.15.246.237] b'root' authenticated with b'password'
2020-10-18T21:09:49.236559Z [SSHService b'ssh-userauth' on HoneyPotSSHTransport,5,107.15.246.237] starting service b'ssh-connection'
2020-10-18T21:09:49.318824Z [SSHService b'ssh-connection' on HoneyPotSSHTransport,5,107.15.246.237] got channel b'session' request
2020-10-18T21:09:49.319200Z [SSHChannel session (0) on SSHService b'ssh-connection' on HoneyPotSSHTransport,5,107.15.246.237] channel open
2020-10-18T21:09:49.319385Z [SSHService b'ssh-connection' on HoneyPotSSHTransport,5,107.15.246.237] got global b'no-more-sessions@openssh.com' re
quest
2020-10-18T21:09:49.432600Z [SSHChannel session (0) on SSHService b'ssh-connection' on HoneyPotSSHTransport,5,107.15.246.237] pty request: b'xter
m-256color' (32, 145, 0, 0)
2020-10-18T21:09:49.432817Z [SSHChannel session (0) on SSHService b'ssh-connection' on HoneyPotSSHTransport,5,107.15.246.237] Terminal Size: 145
```

json


```
{"eventid":"cowrie.client.fingerprint","username":"root","fingerprint":"72:2a:0c:fe:3c:7e:1d:2d:68:49:bb:13:ee:2e:09:20","key":"ssh-rsa AAAAB3Nza
C1yc2EAAAADAQABAAABgQDmd3hs0RW7rXCX+XIFzhtfXaBAnFmXcWZ+/tIxwm0zR8YJ8FaFNurzAnXd7OSoBALugMSiTknp+ltCeAAeD+eX9Ssr7dajhiESUF0v97aT+sKTeTeHIXSmJRo9Rf
5v3HJL/plQGwmBOlj232xpZ08TMK+tDpd2euC3u9V73Tr/wKZujcbtAmkQZlmt0okA0yG7AM6hS9SGhSYobm9rC2LTFUPU/Y+f2god3Lr87Wb8YOE5o923fA0xhw9zRiIWoPMCYWWkky9t4IK
TEtH5YRcQw82oEaRujBRcezEyVLRbmendoHMhTxzlxEHGOHly88Y/erlIBxMrvLo9Y6APCnHlN4sQou0FwAkVuMrOmDv2xsHN7c5jJ4XMnx0F1Xmpy6KvKHRHmXMONa0DxZHO5nsm0Nq2eD65
+U976Lj3fV0fTuf5bqmdZBrbUnrJOfSEYB2D3fM/yBCFMdbR7Bsi7hGXlmCduAvU+gKVuEgYWvSN9cNW7mbGaHjEhV+o4eyyq/k=","type":"ssh-rsa","message":"public key atte
mpt for user root of type ssh-rsa with fingerprint 72:2a:0c:fe:3c:7e:1d:2d:68:49:bb:13:ee:2e:09:20","sensor":"ip-172-31-35-158.us-east-2.compute.
internal","timestamp":"2020-10-18T21:09:46.923924Z","src_ip":"107.15.246.237","session":"212a45316d84"}
{"eventid":"cowrie.login.success","username":"root","password":"2323","message":"login attempt [root/2323] succeeded","sensor":"ip-172-31-35-158.
us-east-2.compute.internal","timestamp":"2020-10-18T21:09:49.235274Z","src_ip":"107.15.246.237","session":"212a45316d84"}
{"eventid":"cowrie.client.size","width":145,"height":32,"message":"Terminal Size: 145 32","sensor":"ip-172-31-35-158.us-east-2.compute.internal",
"timestamp":"2020-10-18T21:09:49.432817Z","src_ip":"107.15.246.237","session":"212a45316d84"}
{"eventid":"cowrie.client.var","name":"LANG","value":"en_US.UTF-8","message":"request_env: LANG=en_US.UTF-8","sensor":"ip-172-31-35-158.us-east-2
.compute.internal","timestamp":"2020-10-18T21:09:49.433601Z","src_ip":"107.15.246.237","session":"212a45316d84"}
{"eventid":"cowrie.session.params","arch":"linux-x64-lsb","message":[],"sensor":"ip-172-31-35-158.us-east-2.compute.internal","timestamp":"2020-1
0-18T21:09:49.434955Z","src_ip":"107.15.246.237","session":"212a45316d84"}
{"eventid":"cowrie.command.input","input":"exit","message":"CMD: exit","sensor":"ip-172-31-35-158.us-east-2.compute.internal","timestamp":"2020-1
0-18T21:11:06.544156Z","src_ip":"107.15.246.237","session":"212a45316d84"}
{"eventid":"cowrie.log.closed","ttylog":"var/lib/cowrie/tty/2638f1c1c2018567a46a4cae049dd90db2d468e1538d60d328f2707d071f73c5","size":313,"shasum"
:"2638f1c1c2018567a46a4cae049dd90db2d468e1538d60d328f2707d071f73c5","duplicate":false,"duration":77.11108016967773,"message":"Closing TTY Log: va
r/lib/cowrie/tty/2638f1c1c2018567a46a4cae049dd90db2d468e1538d60d328f2707d071f73c5 after 77 seconds","sensor":"ip-172-31-35-158.us-east-2.compute.
internal","timestamp":"2020-10-18T21:11:06.545614Z","src_ip":"107.15.246.237","session":"212a45316d84"}
{"eventid":"cowrie.session.closed","duration":80.16746735572815,"message":"Connection lost after 80 seconds","sensor":"ip-172-31-35-158.us-east-2
.compute.internal","timestamp":"2020-10-18T21:11:06.587591Z","src_ip":"107.15.246.237","session":"212a45316d84"}
```


Once you're happy it's set up well, let it run for at least ~3 days or until you get several attacks logged!

Results (4pts)

Take a look at what attackers did. **How many attacks did you capture? What kinds of things did attackers tend to do?**

2

capturing bot and spreading

Pick a particular attack and describe:

- **The network origin (IP address organization and geolocation)**
  177.102.0.23, as27699 telefônica brasil s.a, Brazil
- **The steps carried out**
  checks for system info and download binary
  kill processes
  add user

searched ssh info

- **Does it appear to be automated or manual?**
  automated
- **What appears to have been the attacker's goal?**
  affect more machine, mining
- **If they downloaded file(s), analyze these**
  bash, crypto mining software XMRig

Question 7: Denial of Service attack using TorsHammer (6 points)

Let's do a small DOS attack from your Kali VM to your Linux VM using TorsHammer, a slow POST attack similar to the slowloris attack we discussed in class.

On your **Kali VM**, download the TorsHammer from https://sourceforge.net/projects/torshammer/

To launch an attack, use the torshammer.py file and pass the necessary parameters to it.

```
# ./torshammer.py
```

On your **Linux VM**, If you haven't already, install the Apache web server and ensure it's working on port 80.

From the Kali VM, attack your Linux VM. Use 512 threads. Show the command used.
./torshammer.py -t vcm-16036.vm.duke.edu -r 512

From your local machine's web browser, try to navigate to the Linux VM. Refresh the page a few times; what do you observe?
Browser wait for a long time before refresh

Show the output of "`netstat -t`" (list TCP connections) before versus during an attack.

6 Question 6: Run a honeypot and see what you get 8 / 8

✓ **- 0 pts** **Correct**

- 2 pts Error in Setup

- 4 pts No Result

- 3 pts Lack Analysis for Results

searched ssh info

- **Does it appear to be automated or manual?**
  automated
- **What appears to have been the attacker's goal?**
  affect more machine, mining
- **If they downloaded file(s), analyze these**
  bash, crypto mining software XMRig

Question 7: Denial of Service attack using TorsHammer (6 points)

Let's do a small DOS attack from your Kali VM to your Linux VM using TorsHammer, a slow POST attack similar to the slowloris attack we discussed in class.

On your **Kali VM**, download the TorsHammer from https://sourceforge.net/projects/torshammer/

To launch an attack, use the torshammer.py file and pass the necessary parameters to it.

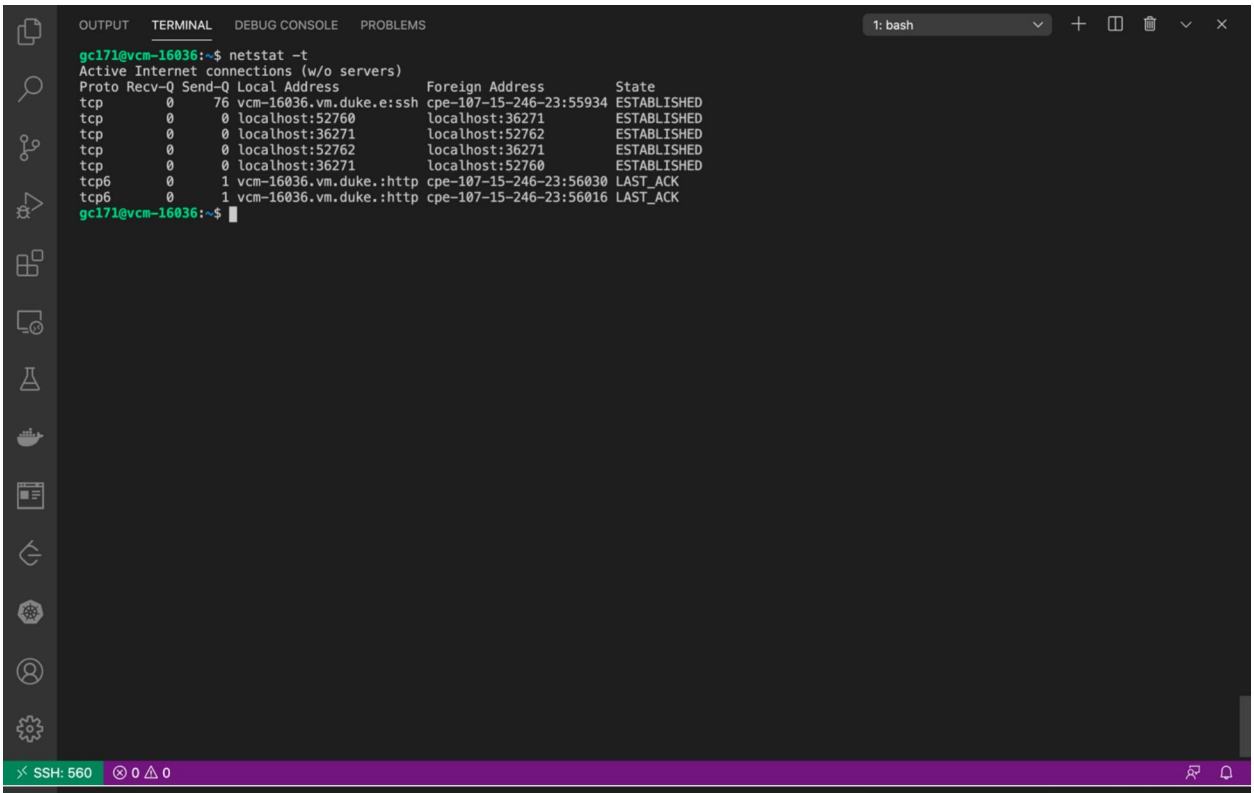
```
# ./torshammer.py
```

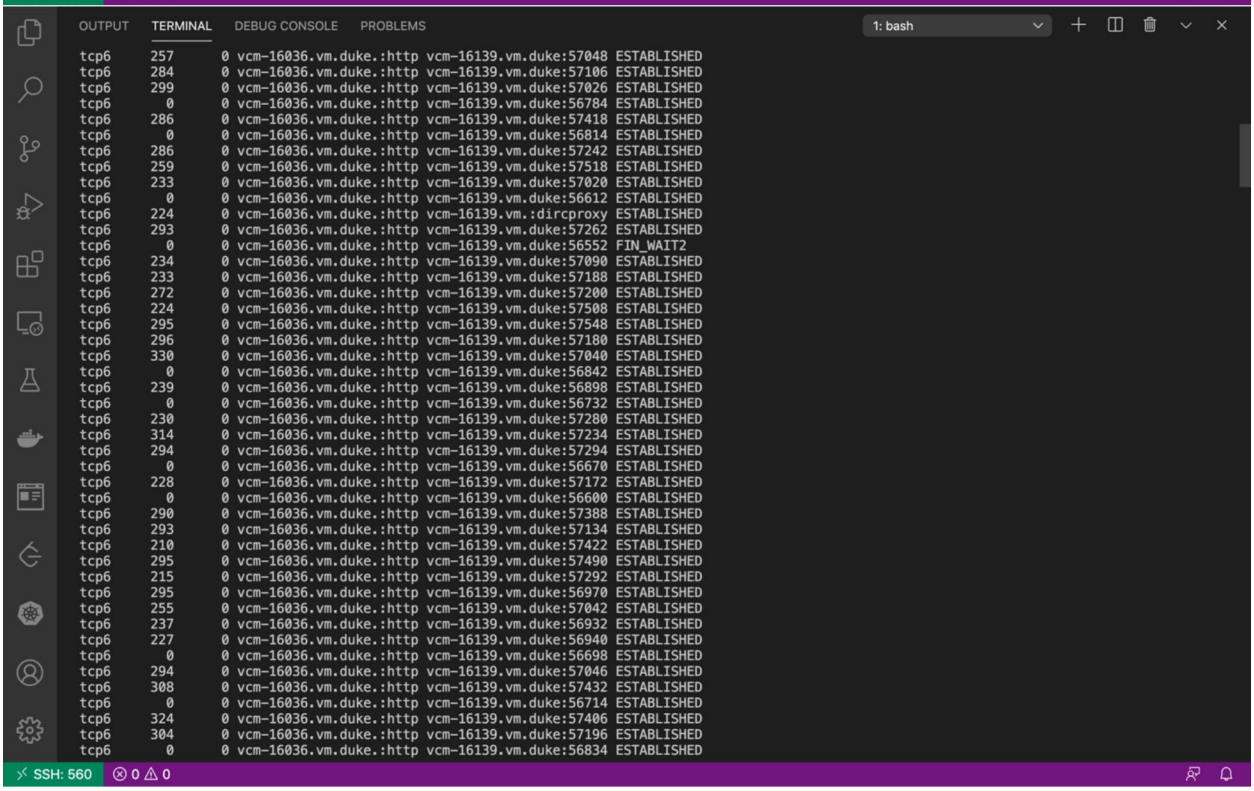
On your **Linux VM**, If you haven't already, install the Apache web server and ensure it's working on port 80.

From the Kali VM, attack your Linux VM. Use 512 threads. Show the command used.
./torshammer.py -t vcm-16036.vm.duke.edu -r 512

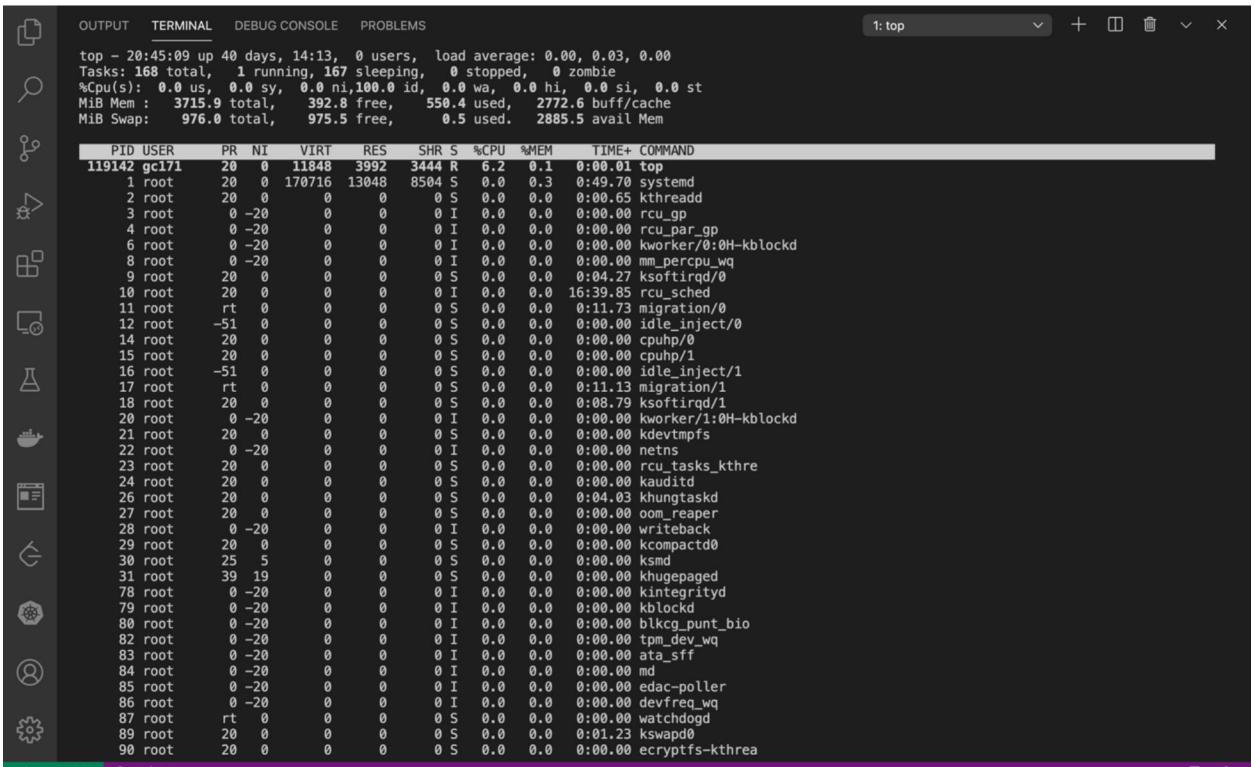
From your local machine's web browser, try to navigate to the Linux VM. Refresh the page a few times; what do you observe?
Browser wait for a long time before refresh

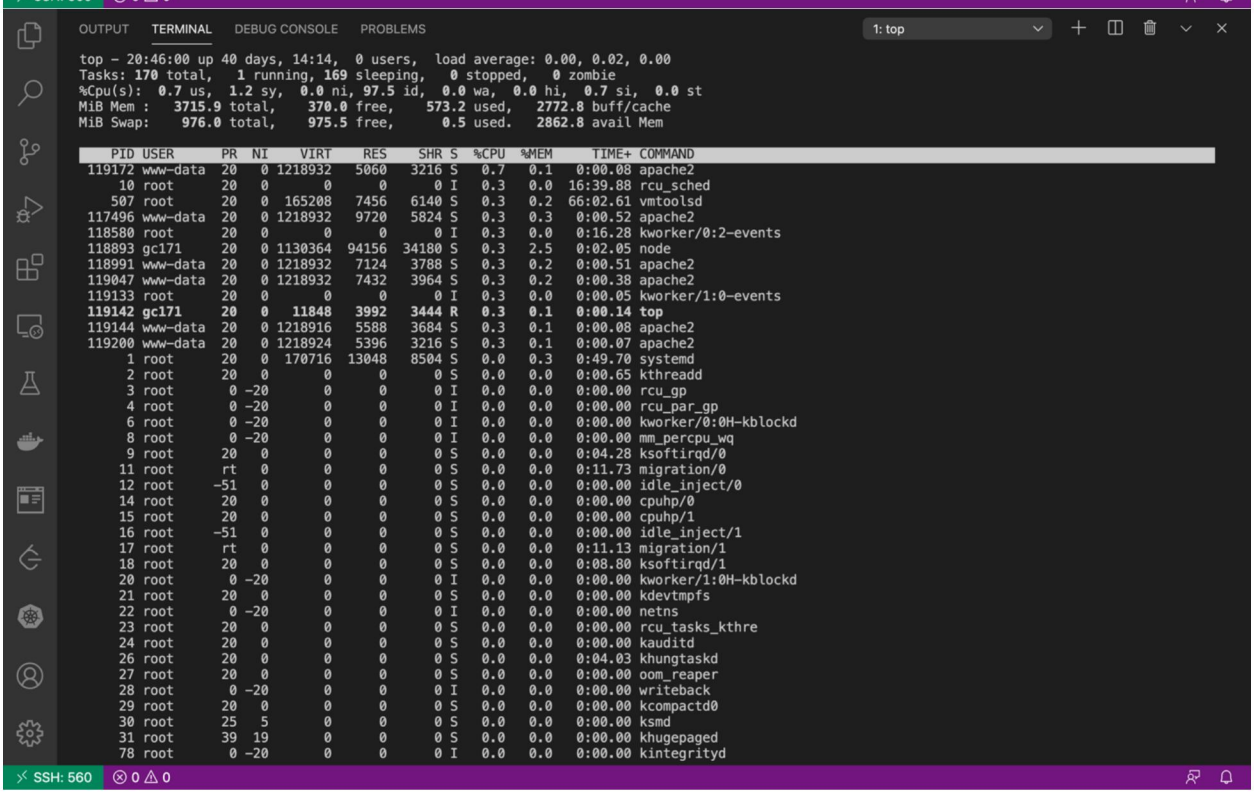
Show the output of "`netstat -t`" (list TCP connections) before versus during an attack.





Use "`top`" to assess CPU usage before versus during the attack. Does CPU usage increase significantly? Why or why not?





no.

I think: a) apache doesn't take up much cpu b) the cpu is good

In another terminal from the Kali VM (or another Linux machine), use the `time` and `curl` commands a few times to roughly gauge the latency of HTTP responses before, during, and after the attack.

before 0.021s
during 20.229s
after 0.022s

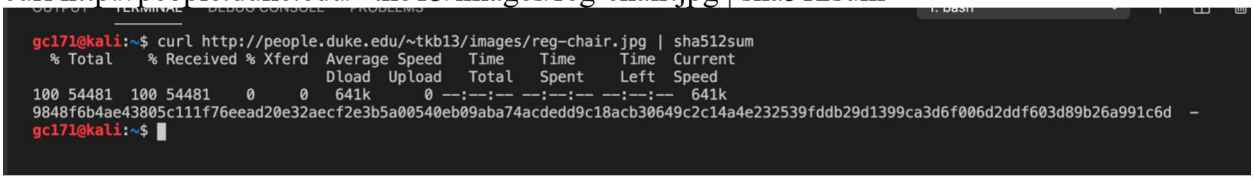
Question 8: Shell practice (9 points)

For each of the following questions, give the command(s) and output. If the output is more than a few lines, you may truncate it.

Web file hash (2 points)

Develop a shell command to find the SHA 512 hash of the JPEG picture of the instructor's dog, found on this page. To help check your work, the last byte is 0x6d. Show the command and its output. A properly fancy solution will avoid the need to write any files to disk.

curl http://people.duke.edu/~tkb13/images/reg-chair.jpg | sha512sum



Ping check (2 points)

Develop a shell command or shell script (max 5 lines) to **`ping`** several VCM VMs with a single packet with a timeout of 1 second with an output of simply "ok" or "down" for each host, one per line. The range of hosts to check is vcm-16000.vm.duke.edu through vcm-16100.vm.duke.edu (101 hosts total). Shorter solutions are preferred to longer ones. Example output (higher numbered hosts omitted for space):

7 Question 7: Denial of Service attack using TorsHammer 5.5 / 6

- **0 pts** Correct
- **1 pts** Didn't explain why CPU usage did not increase much
- **1 pts** Wrong time during attack
- **2 pts** Wrong report on time
- **0.5** **Point adjustment**

The CPU usage does not increase much is that slow POST attack does not send computation demanding tasks. It just send charactars very slow. Apache may take CPU, in some cases.

In another terminal from the Kali VM (or another Linux machine), use the `time` and `curl` commands a few times to roughly gauge the latency of HTTP responses before, during, and after the attack.

before 0.021s
during 20.229s
after 0.022s

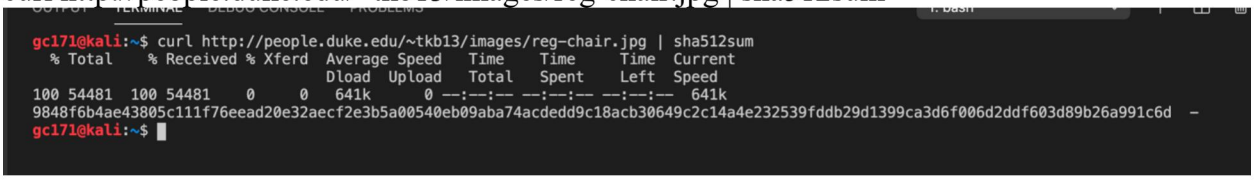
Question 8: Shell practice (9 points)

For each of the following questions, give the command(s) and output. If the output is more than a few lines, you may truncate it.

Web file hash (2 points)

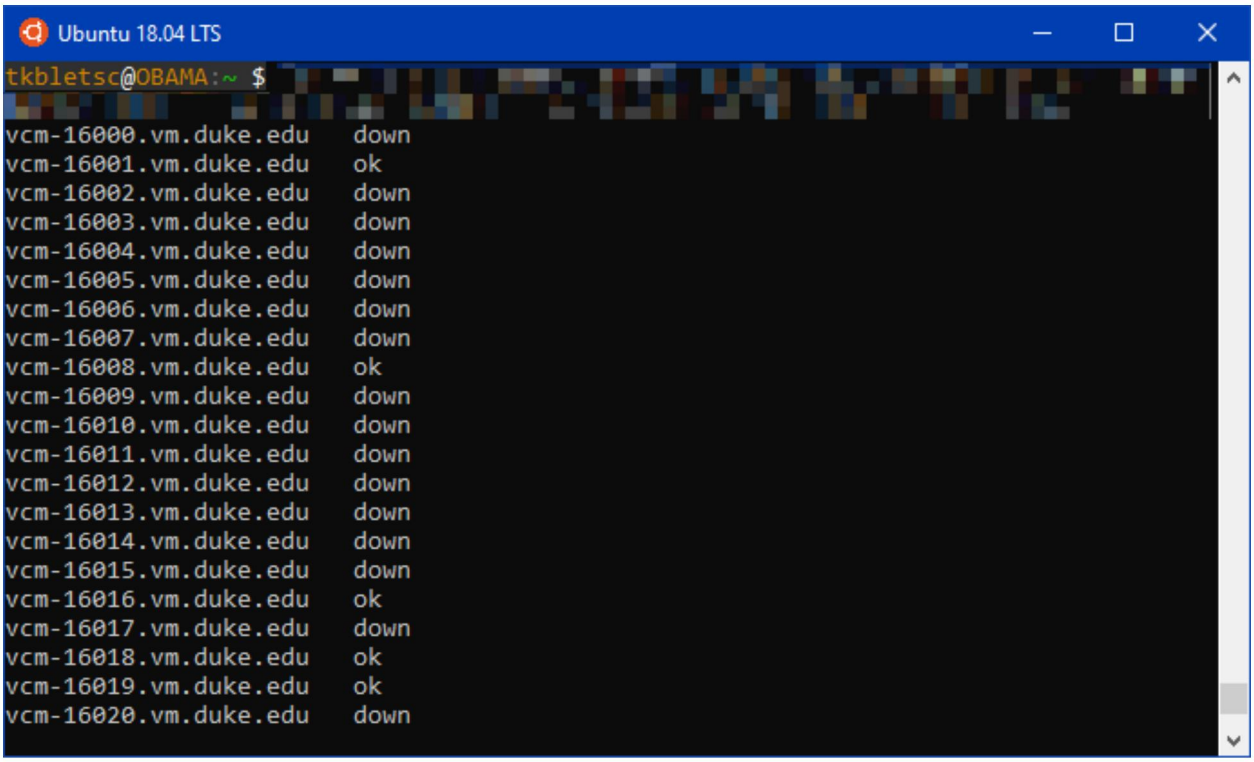
Develop a shell command to find the SHA 512 hash of the JPEG picture of the instructor's dog, found on this page. To help check your work, the last byte is 0x6d. Show the command and its output. A properly fancy solution will avoid the need to write any files to disk.

curl http://people.duke.edu/~tkb13/images/reg-chair.jpg | sha512sum



Ping check (2 points)

Develop a shell command or shell script (max 5 lines) to **`ping`** several VCM VMs with a single packet with a timeout of 1 second with an output of simply "ok" or "down" for each host, one per line. The range of hosts to check is vcm-16000.vm.duke.edu through vcm-16100.vm.duke.edu (101 hosts total). Shorter solutions are preferred to longer ones. Example output (higher numbered hosts omitted for space):


Ubuntu 18.04 LTS
tkbletsc@OBAMA:~ $
vcm-16000.vm.duke.edu down
vcm-16001.vm.duke.edu ok
vcm-16002.vm.duke.edu down
vcm-16003.vm.duke.edu down
vcm-16004.vm.duke.edu down
vcm-16005.vm.duke.edu down
vcm-16006.vm.duke.edu down
vcm-16007.vm.duke.edu down
vcm-16008.vm.duke.edu ok
vcm-16009.vm.duke.edu down
vcm-16010.vm.duke.edu down
vcm-16011.vm.duke.edu down
vcm-16012.vm.duke.edu down
vcm-16013.vm.duke.edu down
vcm-16014.vm.duke.edu down
vcm-16015.vm.duke.edu down
vcm-16016.vm.duke.edu ok
vcm-16017.vm.duke.edu down
vcm-16018.vm.duke.edu ok
vcm-16019.vm.duke.edu ok
vcm-16020.vm.duke.edu down

```
gc171@kali:~$ ./t.sh
vcm-16000.vm.duke.edu down
vcm-16001.vm.duke.edu ok
vcm-16002.vm.duke.edu down
vcm-16003.vm.duke.edu down
vcm-16004.vm.duke.edu ok
vcm-16005.vm.duke.edu down
vcm-16006.vm.duke.edu down
vcm-16007.vm.duke.edu down
vcm-16008.vm.duke.edu ok
vcm-16009.vm.duke.edu down
vcm-16010.vm.duke.edu down
vcm-16011.vm.duke.edu down
vcm-16012.vm.duke.edu down
vcm-16013.vm.duke.edu down
vcm-16014.vm.duke.edu down
vcm-16015.vm.duke.edu down
vcm-16016.vm.duke.edu down
vcm-16017.vm.duke.edu down
vcm-16018.vm.duke.edu ok
vcm-16019.vm.duke.edu ok
vcm-16020.vm.duke.edu down
vcm-16021.vm.duke.edu down
vcm-16022.vm.duke.edu down
vcm-16023.vm.duke.edu down
vcm-16024.vm.duke.edu ok
vcm-16025.vm.duke.edu ok
vcm-16026.vm.duke.edu down
vcm-16027.vm.duke.edu down
vcm-16028.vm.duke.edu down
vcm-16029.vm.duke.edu ok
vcm-16030.vm.duke.edu down
vcm-16031.vm.duke.edu down
vcm-16032.vm.duke.edu ok
vcm-16033.vm.duke.edu down
vcm-16034.vm.duke.edu down
vcm-16035.vm.duke.edu down
vcm-16036.vm.duke.edu ok
vcm-16037.vm.duke.edu down
```

```
for i in {16000..16100}; do
    ping vcm-$i.vm.duke.edu -w 1 -c 1 2>&1 | grep -qP "1 received" && echo "vcm-
$i.vm.duke.edu ok"
    ping vcm-$i.vm.duke.edu -w 1 -c 1 2>&1 | grep -qP "not known|0 received" && echo
"vcm-$i.vm.duke.edu down"
done
```

Binary file analysis (3 points)

Using **`strings`**, and **`hd`**, let's start analyzing **`cryptotest.pyc`** from the earlier "Simple Encryption Program" question for possible reverse engineering. Answer each of the following questions and show the command(s)/output that led you to your answer.

- What message will likely be printed if the tool is able to compress the cipher text too much?
  strings cryptotest.pyc

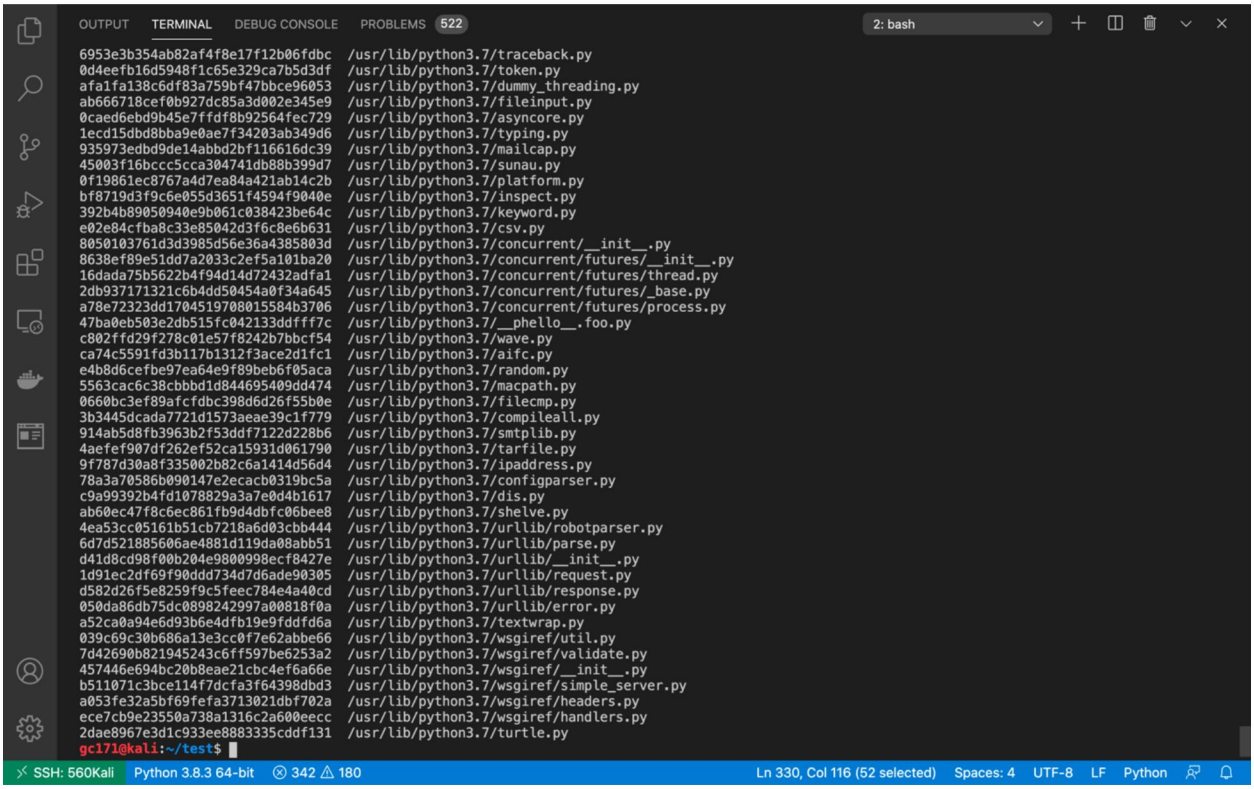
```
The cipher file is too compressable (Ratio: %.2f%%).
```

- What is the "magic number" of the file, as described here?
  hd cryptotest.pyc
  55 0d 0d 0a

Bulk hash (2 points)

On your Kali VM, develop a shell command that will print the MD5 hash of every `.py` file under `/usr/lib/python3.7`.

find /usr/lib/python3.7 -name "*.py" -exec md5sum '{}' \;



Question 9: Analyze forensic server packet capture network logs (10 points)

Below is a network capture from an attack on a real Linux server. The capture was created by setting up a new CentOS 5.9 Linux server, turning off the firewall, and setting the root password to root. The server was compromised within a few hours. Before it was put online, Wireshark was run and configured to capture the network event before, during, and after the compromise.

The network captures are located here:
http://people.duke.edu/~tkb13/fixed/iasg_capture_files.tgz

Download this file to your home directory and analyze it there.

8 Question 8: Shell Practice 9 / 9

✓ **- 0 pts** **Correct**

- 0.5 pts Didn't provide the command that downloads the image file.

- 0.5 pts Didn't deal with url pointing to non-existing vcms (ping return "unknown")

- 0.5 pts The "too compressable" message is not complete. The following ratio stuff would be prented, too.

- 1 pts Wrong magic number

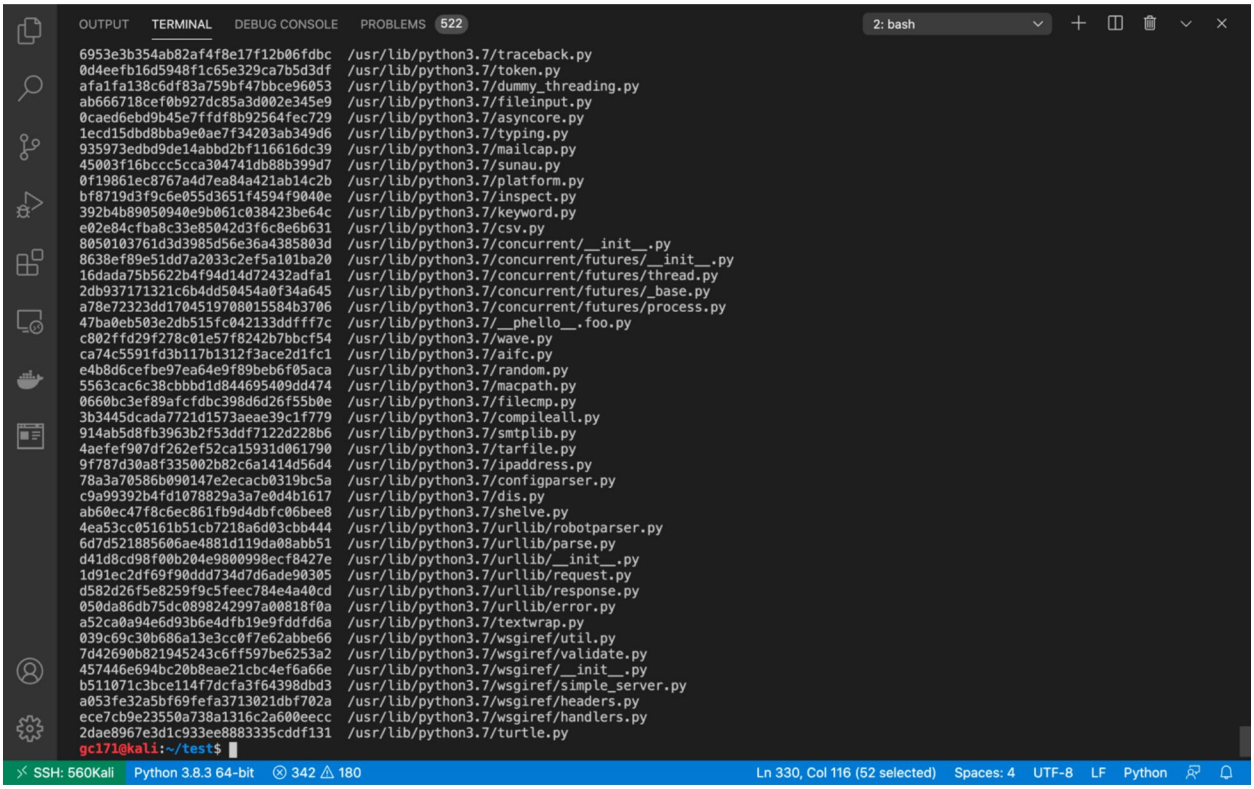
```
The cipher file is too compressable (Ratio: %.2f%%).
```

- What is the "magic number" of the file, as described here?
  hd cryptotest.pyc
  55 0d 0d 0a

Bulk hash (2 points)

On your Kali VM, develop a shell command that will print the MD5 hash of every `.py` file under `/usr/lib/python3.7`.

find /usr/lib/python3.7 -name "*.py" -exec md5sum '{}' \;



Question 9: Analyze forensic server packet capture network logs (10 points)

Below is a network capture from an attack on a real Linux server. The capture was created by setting up a new CentOS 5.9 Linux server, turning off the firewall, and setting the root password to root. The server was compromised within a few hours. Before it was put online, Wireshark was run and configured to capture the network event before, during, and after the compromise.

The network captures are located here:
http://people.duke.edu/~tkb13/fixed/iasg_capture_files.tgz

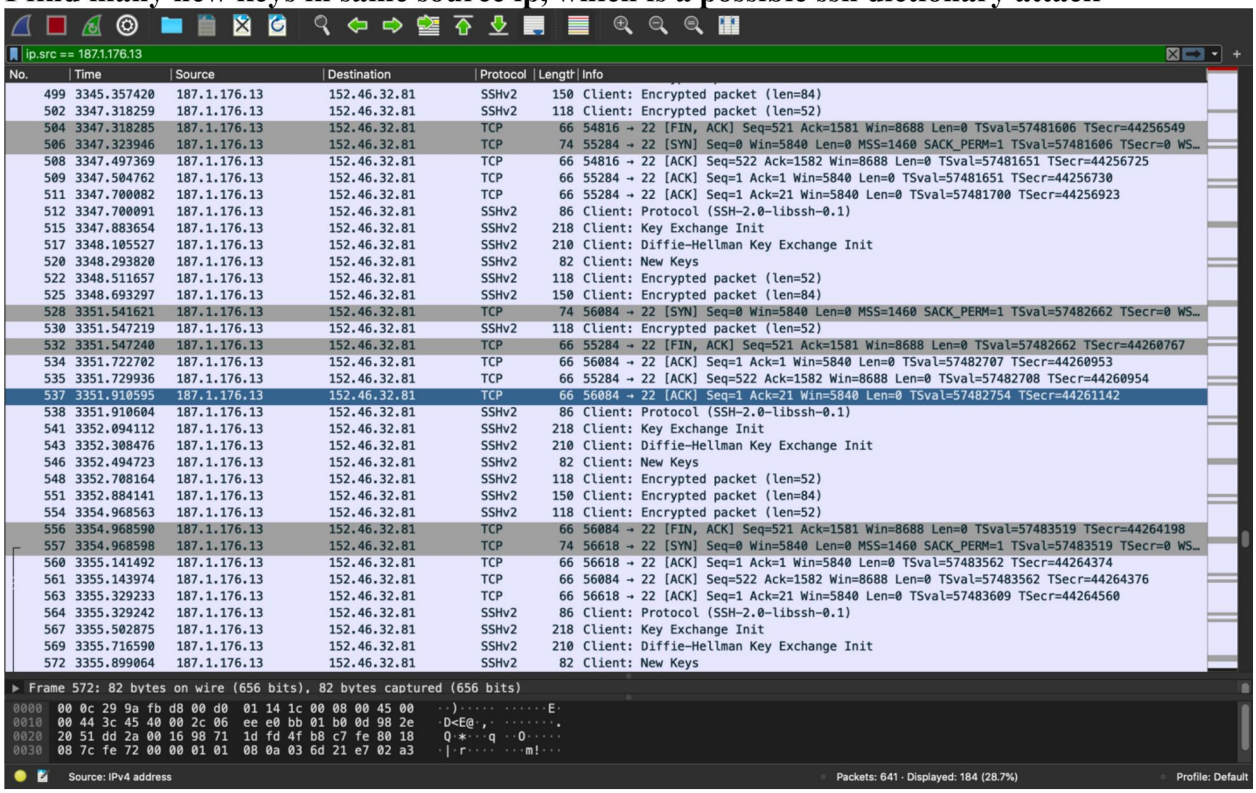
Download this file to your home directory and analyze it there.

Using tcpdump or Wireshark, analyze the packet capture files from that attack and describe/show the following activities or small samples:

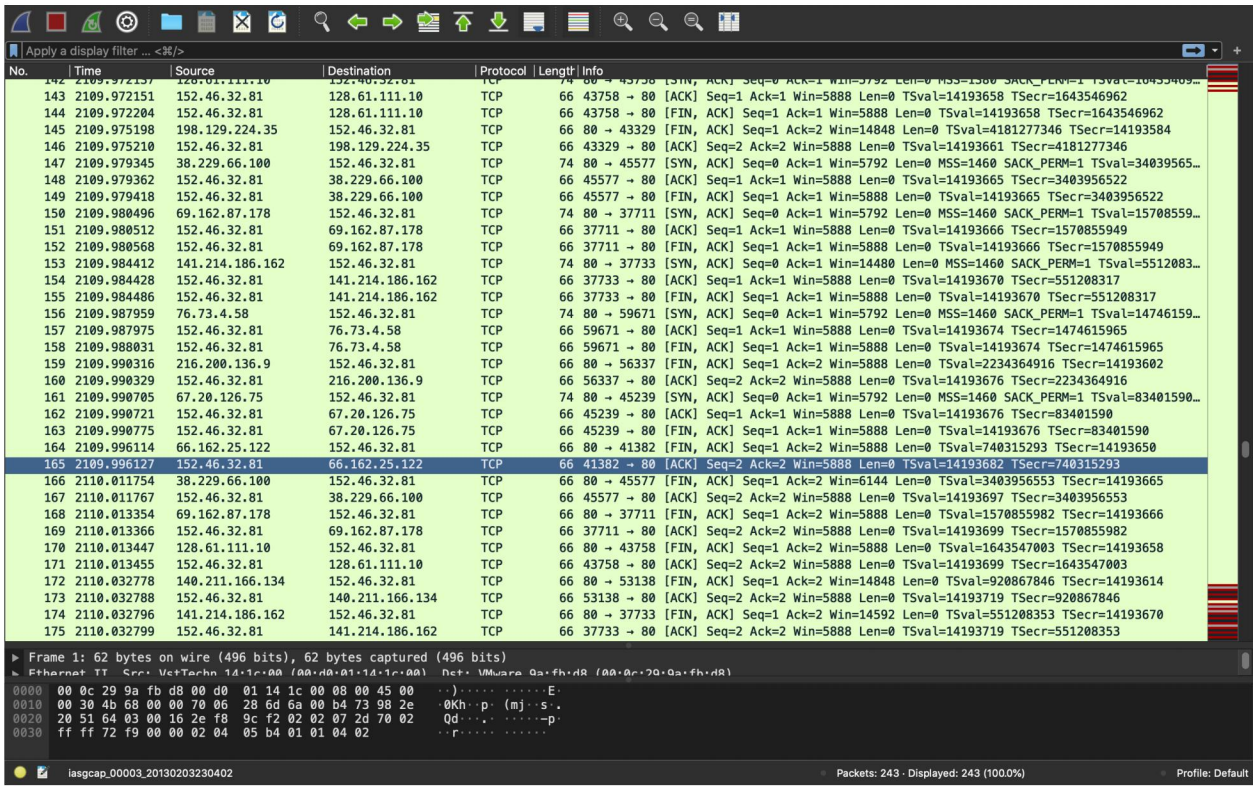
1. **Reconnaissance**

in file `iasgcap_00011_20130204070402`

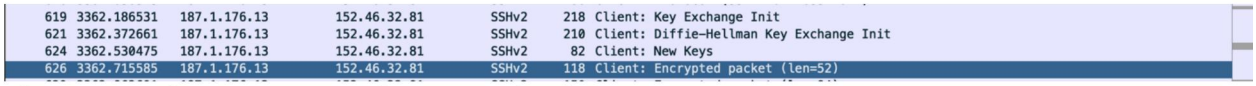
I find many new keys in same source ip, which is a possible ssh dictionary attack



previous log seems normal

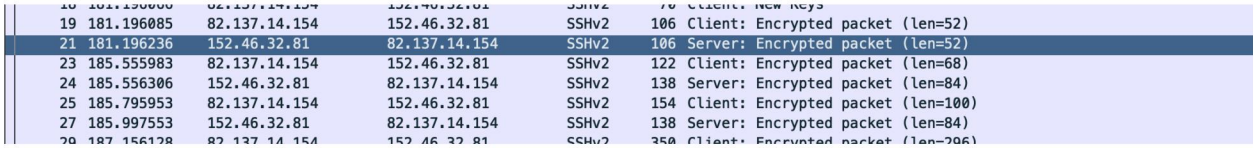


2. Actual SSH Attacks



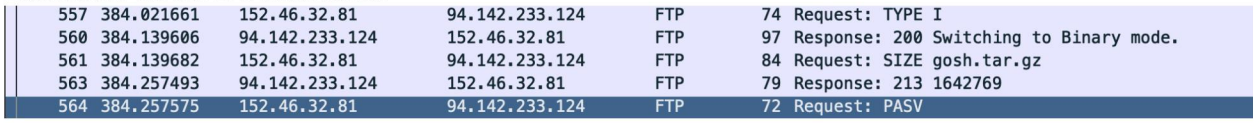
always new keys, no normal communication

3. Successful SSH authentication



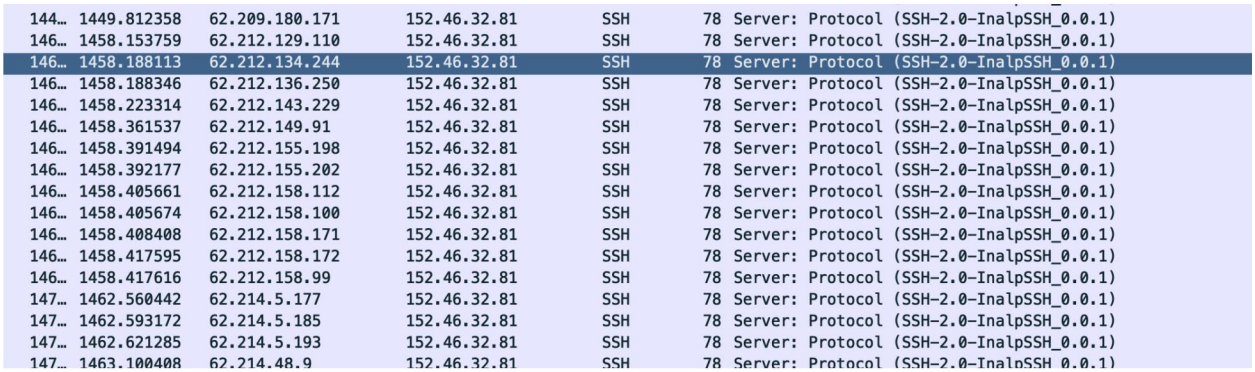
ssh normal communication

4. External Tools Downloaded



ftp

5. External Target attacks



144…	1449.812358	62.209.180.171	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
146…	1458.153759	62.212.129.110	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
146…	1458.188113	62.212.134.244	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
146…	1458.188346	62.212.136.250	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
146…	1458.223314	62.212.143.229	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
146…	1458.361537	62.212.149.91	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
146…	1458.391494	62.212.155.198	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
146…	1458.392177	62.212.155.202	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
146…	1458.405661	62.212.158.112	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
146…	1458.405674	62.212.158.100	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
146…	1458.408408	62.212.158.171	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
146…	1458.417595	62.212.158.172	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
146…	1458.417616	62.212.158.99	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
147…	1462.560442	62.214.5.177	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
147…	1462.593172	62.214.5.185	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
147…	1462.621285	62.214.5.193	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
147…	1463.100408	62.214.48.9	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)

The answers will not be obvious or clear-cut. You will need to use inferences and your own best judgment. For example, SSH is encrypted, so you won't see actual successful SSH logins take place. However, from volume and timing of traffic, you can make an informed guess with reasonable justification; that's what we're looking for.

Tip: Some of the packet capture files are very large and you will need to break them into smaller chunks before you analyze them. A good working size is around 100,000 packets per file.
Sample Pcap Split on Windows:

```
"c:\Program Files\Wireshark\editcap.exe" -c 100000 iasgcap_00012_20130204080402 test
```

Note: Credit for this dataset goes to Samuel Carter at NCSU.

Question 10: Analyze forensic server hacker package (8 points)

Analyze the **gosh.tar.gz** package downloaded as part of the server attack described above. The gosh.tar.gz file is located here:
https://people.duke.edu/~tkb13/courses/ece560/homework/hw4/gosh.tar.bz2

Just download a copy of it to your Linux VM to perform the analysis.

Explain what each file in the package is/does/used for and how they are related (if applicable).

Tips:

- Use **`file`** to figure out what kinds of files these are (text files, shell scripts, executables (ELF binaries), etc.).
- For shell scripts, read them without executing.
- For binary executables, your first step would usually be to set up a sandbox in a throwaway VM for initial analysis, but to save you some work: the binaries are safe to run without arguments. However, **don't run the executable files with any arguments or it will start attacking!** Running without arguments will actually give you a little bit of usage information.
- You can feed any file here to virustotal.com, which will scan it with every common malware scanner and give you a report. For some files, it may even have community info (postings by security researchers about the file).

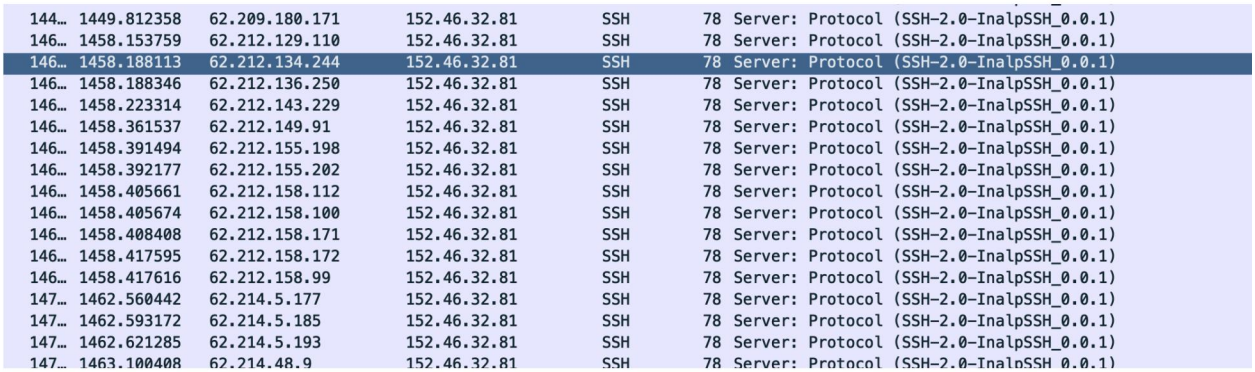
9 Question 9: Analyze forensic server packet capture network logs 10 / 10

✓ **- 0 pts** **Correct**

- 2 pts Mistake on External Tool Download

- 10 pts No answer

- 1.5 pts The external tools are downloaded via FTP



144…	1449.812358	62.209.180.171	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
146…	1458.153759	62.212.129.110	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
146…	1458.188113	62.212.134.244	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
146…	1458.188346	62.212.136.250	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
146…	1458.223314	62.212.143.229	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
146…	1458.361537	62.212.149.91	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
146…	1458.391494	62.212.155.198	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
146…	1458.392177	62.212.155.202	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
146…	1458.405661	62.212.158.112	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
146…	1458.405674	62.212.158.100	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
146…	1458.408408	62.212.158.171	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
146…	1458.417595	62.212.158.172	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
146…	1458.417616	62.212.158.99	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
147…	1462.560442	62.214.5.177	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
147…	1462.593172	62.214.5.185	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
147…	1462.621285	62.214.5.193	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)
147…	1463.100408	62.214.48.9	152.46.32.81	SSH	78 Server: Protocol (SSH-2.0-InalpSSH_0.0.1)

The answers will not be obvious or clear-cut. You will need to use inferences and your own best judgment. For example, SSH is encrypted, so you won't see actual successful SSH logins take place. However, from volume and timing of traffic, you can make an informed guess with reasonable justification; that's what we're looking for.

Tip: Some of the packet capture files are very large and you will need to break them into smaller chunks before you analyze them. A good working size is around 100,000 packets per file.
Sample Pcap Split on Windows:

```
"c:\Program Files\Wireshark\editcap.exe" -c 100000 iasgcap_00012_20130204080402 test
```

Note: Credit for this dataset goes to Samuel Carter at NCSU.

Question 10: Analyze forensic server hacker package (8 points)

Analyze the **gosh.tar.gz** package downloaded as part of the server attack described above. The gosh.tar.gz file is located here:
https://people.duke.edu/~tkb13/courses/ece560/homework/hw4/gosh.tar.bz2

Just download a copy of it to your Linux VM to perform the analysis.

Explain what each file in the package is/does/used for and how they are related (if applicable).

Tips:

- Use **`file`** to figure out what kinds of files these are (text files, shell scripts, executables (ELF binaries), etc.).
- For shell scripts, read them without executing.
- For binary executables, your first step would usually be to set up a sandbox in a throwaway VM for initial analysis, but to save you some work: the binaries are safe to run without arguments. However, **don't run the executable files with any arguments or it will start attacking!** Running without arguments will actually give you a little bit of usage information.
- You can feed any file here to virustotal.com, which will scan it with every common malware scanner and give you a report. For some files, it may even have community info (postings by security researchers about the file).

- On 64-bit Ubuntu-based VMs, you'll need to install some 32-bit support files to run the binaries; see here for info. Without this step, such binaries will give a cryptic "file not found" error on running.
- You may want to use Google Translate to understand some of the messages. Some of the language is Romanian and sometimes explicit.

  1: data. the possible user & password dictionary
  2: ASCII text. root & possible password dictionary
  3: ASCII text. the possible user & password dictionary
  4: ASCII text. the possible user & password dictionary
  5: ASCII text. root & possible password dictionary
  a: shell script scanning ssh using user pair in 1,2,3,4,5
  common: ASCII text. popular text
  gen-pass.sh: shell script. generate password file based on input file
  go.sh: shell script. run ss & ssh-scan
  mfu.txt: ASCII text. filtered ip
  pass_file: ASCII text. generated password file
  pscan2: binary. port scanning
  scam: shell script. run other scripts for scanning and attacking
  secure: shell script. checking user and modify folder
  ss: binary. scan ip
  ssh-scan: binary. dictionary attack on ssh
  vuln.txt: ASCII text. store victim machine ip

Question 11: Hardware level attacks (4 points)

Read PoC||GTFO 04:10 ("Forget Not the Humble Timing Attack").

a. Explain how the attack on the hard drive enclosure was able to reduce the search space from 1,000,000 attempts to 60.

   based on the time

   time is shorter when previous digit is wrong, longer when correct

b. How is the TinySafeBoot firmware "better" than the hard drive enclosure?

   not providing info for timing attack

   it will enter loop when password is wrong

c. A "side channel" attack is one where we use a normally-ignored side effect as useful information about a target. What is the side channel used to defeat the TinySafeBoot firmware despite the defense referred to (b) above?

   power consumption

10 Question 10: Analyze forensic server hacker package 8 / 8

✓ **- 0 pts** **Correct**

- 1 pts Click here to replace this description.

- On 64-bit Ubuntu-based VMs, you'll need to install some 32-bit support files to run the binaries; see here for info. Without this step, such binaries will give a cryptic "file not found" error on running.
- You may want to use Google Translate to understand some of the messages. Some of the language is Romanian and sometimes explicit.

  1: data. the possible user & password dictionary
  2: ASCII text. root & possible password dictionary
  3: ASCII text. the possible user & password dictionary
  4: ASCII text. the possible user & password dictionary
  5: ASCII text. root & possible password dictionary
  a: shell script scanning ssh using user pair in 1,2,3,4,5
  common: ASCII text. popular text
  gen-pass.sh: shell script. generate password file based on input file
  go.sh: shell script. run ss & ssh-scan
  mfu.txt: ASCII text. filtered ip
  pass_file: ASCII text. generated password file
  pscan2: binary. port scanning
  scam: shell script. run other scripts for scanning and attacking
  secure: shell script. checking user and modify folder
  ss: binary. scan ip
  ssh-scan: binary. dictionary attack on ssh
  vuln.txt: ASCII text. store victim machine ip

Question 11: Hardware level attacks (4 points)

Read PoC||GTFO 04:10 ("Forget Not the Humble Timing Attack").

a. Explain how the attack on the hard drive enclosure was able to reduce the search space from 1,000,000 attempts to 60.

   based on the time

   time is shorter when previous digit is wrong, longer when correct

b. How is the TinySafeBoot firmware "better" than the hard drive enclosure?

   not providing info for timing attack

   it will enter loop when password is wrong

c. A "side channel" attack is one where we use a normally-ignored side effect as useful information about a target. What is the side channel used to defeat the TinySafeBoot firmware despite the defense referred to (b) above?

   power consumption

it's different between correct & wrong password

d. What standard password-handling technique would have defeated the attacks described? Why wasn't the above technique deployed in these cases? Hint: what is the code storage capacity and the RAM size of the ATmega328P?

   storing hash

   embedded system doesn't have space for that. ATmega328P has 32 KB flash memory and 2 KB SRAM

Question 12: A practical man-in-the-middle hardware attack (6 points)

Consider the lens project by Zach Banks and Eric Van Albert, entertainingly presented in a presentation at Def Con 23. Watch the talk, then answer the questions below.

a. What is the importance of the punch-down method of connection as opposed to simply cutting and plugging in the conductors? How does this help the attacker?

   you don't need to break physical connection to attach cable
   data is still floating and victims are harder to find out

b. What is the accelerometer for?

   show tamper evident

c. What is the difference between "passive tap" and "active tap"? What does an active tap allow an attacker to do that would otherwise not be possible?

   passive: no physical separation between ports
   active: has physical separation between ports

   attacker gain full control to ethernet

d. To achieve the goal of looping camera footage, why can't they just record and replay the raw packets seen on the network?

   1 the seq number make it hard
   2 other traffic introduce noise

e. Briefly summarize the network layers and protocol involved in the final video looping demo.

Application	http
Session	Rtp
Transportation	Tcp,udp
Network	ip

11 Question 11: Hardware level attacks 4 / 4

✓ **- 0 pts** **Correct**

- 1 pts Part A incorrect

- 1 pts Part B incorrect

- 1 pts Part C incorrect. Answer: Power consumption

- 1 pts Part D incorrect. Answer: Storing hashes of the password instead of the password itself

it's different between correct & wrong password

d. What standard password-handling technique would have defeated the attacks described? Why wasn't the above technique deployed in these cases? Hint: what is the code storage capacity and the RAM size of the ATmega328P?

   storing hash

   embedded system doesn't have space for that. ATmega328P has 32 KB flash memory and 2 KB SRAM

Question 12: A practical man-in-the-middle hardware attack (6 points)

Consider the lens project by Zach Banks and Eric Van Albert, entertainingly presented in a presentation at Def Con 23. Watch the talk, then answer the questions below.

a. What is the importance of the punch-down method of connection as opposed to simply cutting and plugging in the conductors? How does this help the attacker?

   you don't need to break physical connection to attach cable
   data is still floating and victims are harder to find out

b. What is the accelerometer for?

   show tamper evident

c. What is the difference between "passive tap" and "active tap"? What does an active tap allow an attacker to do that would otherwise not be possible?

   passive: no physical separation between ports
   active: has physical separation between ports

   attacker gain full control to ethernet

d. To achieve the goal of looping camera footage, why can't they just record and replay the raw packets seen on the network?

   1 the seq number make it hard
   2 other traffic introduce noise

e. Briefly summarize the network layers and protocol involved in the final video looping demo.

Application	http
Session	Rtp
Transportation	Tcp,udp
Network	ip

f. They mention that they're "glossing over" the issue of HTTPS. How would HTTPS address this problem?
https is encrypted
and if you want to MITK in the beginning, that is costly for embedded system

Question 13: Malware Analysis (18 points)

This question will walk you through some rudimentary analysis of *REAL* malware. This is dynamic analysis (observing the malware's behavior in a VM); a deeper analysis would incorporate static code analysis (looking at the machine code).

NOTICE! Do *not* run this malware on your **Windows VM!** Follow the instructions to set up a scratch VM on your local system or a cloud service.

Further, if running the malware from a VM on a machine on the Duke network, please email the **Duke IT Security Office** first with the message below. Do this each separate day you are about to run the malware. The network requests made by the malware trigger alerts in systems monitored by the IT Security Office (ITSO), and they've requested this advance notice to avoid triggering needless intrusion response, especially after-hours.

This alert need not be sent if your throw-away Windows VM is hosted on a cloud service, because Duke IT won't see those malware request packets.

> To: security@duke.edu
> Subject: Running Emotet malware for ECE 560
>
> ITSO team: I am in Prof. Bletsch's Computer and Information Security class (ECE 560) and am studying malware analysis. I will be running an instance of Emotet malware on a temporary VM within the next few hours, and want to give you advance notice. This procedure has been approved by Duke ITSO, and the VM will be reverted to pre-infection state within 30 minutes. Please contact Tyler Bletsch (Tyler.Bletsch@duke.edu) and Anthony Miracle (Anthony.Miracle@duke.edu) if there are any issues. Thank you.

Note: This question includes a lot of steps, not all of which require a response from you. Steps that are asking for a response have whitespace after them, and the prompt is highlighted green.

Side note: The Windows Registry
If you are not familiar with what the Windows Registry is, research it before proceeding.

Part 1: Setup

1. We need a Windows VM, but because we're going to be intentionally executing malware on it, it cannot be hosted on Duke VCM. You have two choices:

12 Question 12: A practical man-in-the-middle hardware attack 6 / 6

✓ **- 0 pts Correct**

- 1 pts a) Incorrect

- 1 pts b) Incorrect. Answer: It is used for detecting if the lap board is being tampered with while in operation

- 1 pts c) Incorrect

- 1 pts d) Incorrect

- 1 pts e) Incorrect

- 1 pts f) Incorrect. Answer: HTTPS is encrypted

f. They mention that they're "glossing over" the issue of HTTPS. How would HTTPS address this problem?
https is encrypted
and if you want to MITK in the beginning, that is costly for embedded system

Question 13: Malware Analysis (18 points)

This question will walk you through some rudimentary analysis of *REAL* malware. This is dynamic analysis (observing the malware's behavior in a VM); a deeper analysis would incorporate static code analysis (looking at the machine code).

NOTICE! Do *not* run this malware on your **Windows VM!** Follow the instructions to set up a scratch VM on your local system or a cloud service.

Further, if running the malware from a VM on a machine on the Duke network, please email the **Duke IT Security Office** first with the message below. Do this each separate day you are about to run the malware. The network requests made by the malware trigger alerts in systems monitored by the IT Security Office (ITSO), and they've requested this advance notice to avoid triggering needless intrusion response, especially after-hours.

This alert need not be sent if your throw-away Windows VM is hosted on a cloud service, because Duke IT won't see those malware request packets.

> To: security@duke.edu
> Subject: Running Emotet malware for ECE 560
>
> ITSO team: I am in Prof. Bletsch's Computer and Information Security class (ECE 560) and am studying malware analysis. I will be running an instance of Emotet malware on a temporary VM within the next few hours, and want to give you advance notice. This procedure has been approved by Duke ITSO, and the VM will be reverted to pre-infection state within 30 minutes. Please contact Tyler Bletsch (Tyler.Bletsch@duke.edu) and Anthony Miracle (Anthony.Miracle@duke.edu) if there are any issues. Thank you.

Note: This question includes a lot of steps, not all of which require a response from you. Steps that are asking for a response have whitespace after them, and the prompt is highlighted green.

Side note: The Windows Registry
If you are not familiar with what the Windows Registry is, research it before proceeding.

Part 1: Setup

1. We need a Windows VM, but because we're going to be intentionally executing malware on it, it cannot be hosted on Duke VCM. You have two choices:

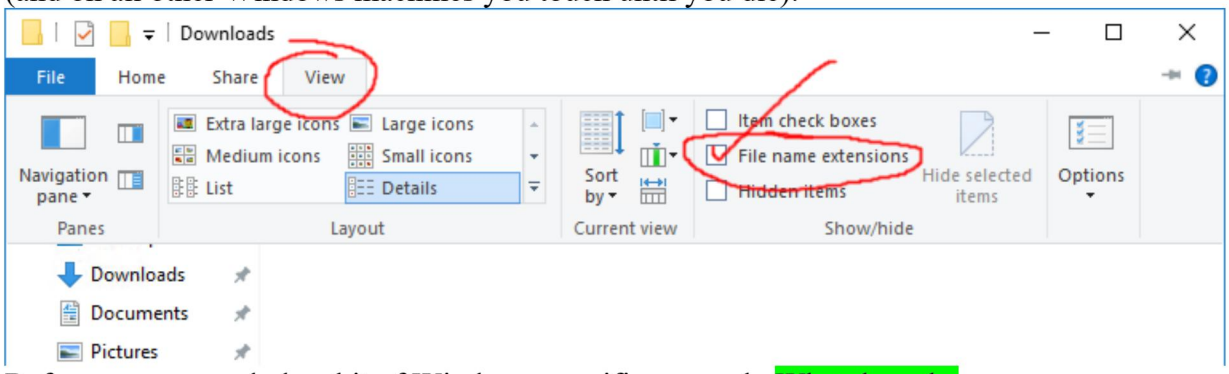
a. **CHOICE 1:** Install a hypervisor onto your own computer.
    i. For this, I recommend VirtualBox: it's free, supports snapshots, and works cross platform. If you have your own preferred hypervisor, you can use that, assuming it has support for VM snapshots.
    ii. You can get the install ISO for Windows 10 here. If you're on Windows and don't want to download Microsoft's "download tool" (a laudable goal), here are directions to get a direct ISO download link.
    iii. Just make a VM, boot the ISO, and next-next-next your way through the install, noting these facts:
        1. No need to provide an activation key, as our experiment will be over long before the activation grace period is up.
        2. If prompted, indicate that the version of Windows 10 you're installing is 64-bit Professional.
        3. Any time Microsoft gives you a privacy question, answer "no".
        4. When it comes to making a user, do NOT give it a Microsoft account! You don't want this machine to have any access to a persistent account. For dumb+bad reasons, Microsoft has increasingly hidden the ability to use a non-cloud account, but it's still there. Indicate that you want Windows to be part of a "domain", then it will let you make a local account. You may need to disable internet access during install to force the installer to let you make a local account.

            Note: After booting, Microsoft claims in the security panel that having a cloud account is "more secure"; if you see this, be sure to laugh out loud directly into the computer's microphone, because that is very wrong.
        5. Do not use a password you use anywhere else.
    iv. You should now be able to finish the install and boot into a fresh Windows VM.
    v. In your hypervisor of choice, enable clipboard sharing and drag-and-drop. This will make it easy to inject files into the VM and copy useful content out for analysis.
b. **CHOICE 2:** If you don't want a local VM, you can use a cloud service. Amazon EC2, Vultr, Digital Ocean, or Microsoft Azure should all be able to host a Windows VM easily. For Amazon, you have educational credits you can use to make the exercise free. The downside of this route is that cloud snapshot facilities are generally slower and more cumbersome than on a local hypervisor. For this, you'll be connecting to the VM via Remote Desktop.

2. Once the VM is running (either local or cloud), we'll permanently disable the built-in Windows malware protection "Windows Defender". Follow the "group policy" process described here.

    Side comment: You might think that malware isn't that dangerous if Defender can simply catch it. That's not because this malware isn't dangerous, but simply because it's old. Threats you face in the wild will often not be flagged by anti-malware software. We disable this protection here because we're learning on a piece of known malware.

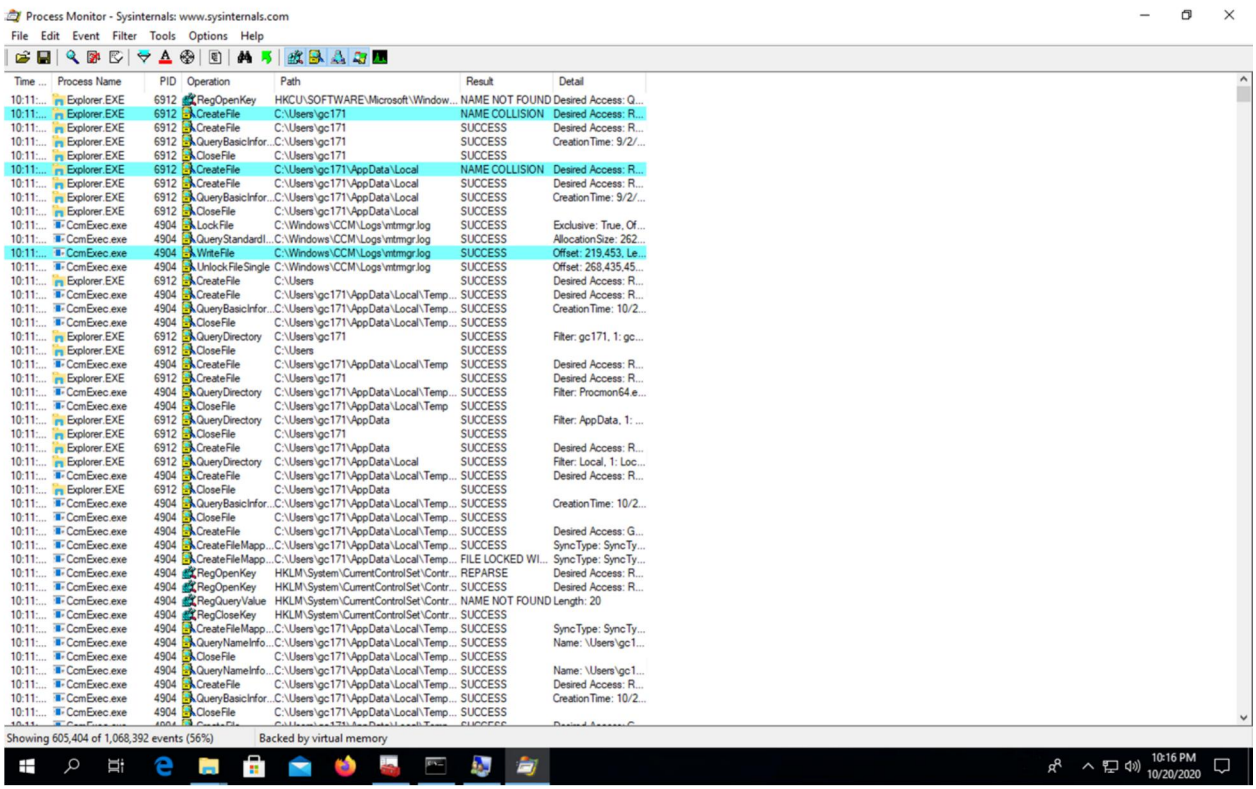
3. As described in Homework 1, be sure to enable file extensions in Explorer for this VM (and on all other Windows machines you touch until you die):



4. Before you proceed, do a bit of Windows-specific research. What does the "CurrentVersion\Run" family of registry keys do?
make programs run when user log in

Part 2: Process Monitor

1. Download and unzip Process Monitor from here. Process Monitor is like WireShark, except that instead of network traffic, it captures the traffic between user processes and the operating system, breaking these down into “File IO”, “Registry IO”, “Network IO”, and “Process and Thread Activity”.
2. Play around a bit with it.
3. Just like WireShark, Process Monitor captures events, displaying only those events that match the current filter. In preparation for the malware test we’ll be doing, let’s reduce how “noisy” the output is -- right click the process name for events that seems superfluous, such as “svchost.exe”, “services.exe”, “SearchIndexer.exe”, etc., and choose “Exclude <thing>”. Do NOT exclude “explorer.exe”. This implicitly updates the filters.
4. Process Monitor also has a highlighting filter -- events matching this filter are marked in cyan. Let’s highlight events that change something as opposed to simply reading -- add a highlight filter for “Category” set to “Write”. This filter is a high-level catch-all for all operations that “do something” rather than just passively read something (e.g. file writes, registry changes, etc.).
5. Paste a screenshot below of Process Monitor showing some events with a few “write” events highlighted.

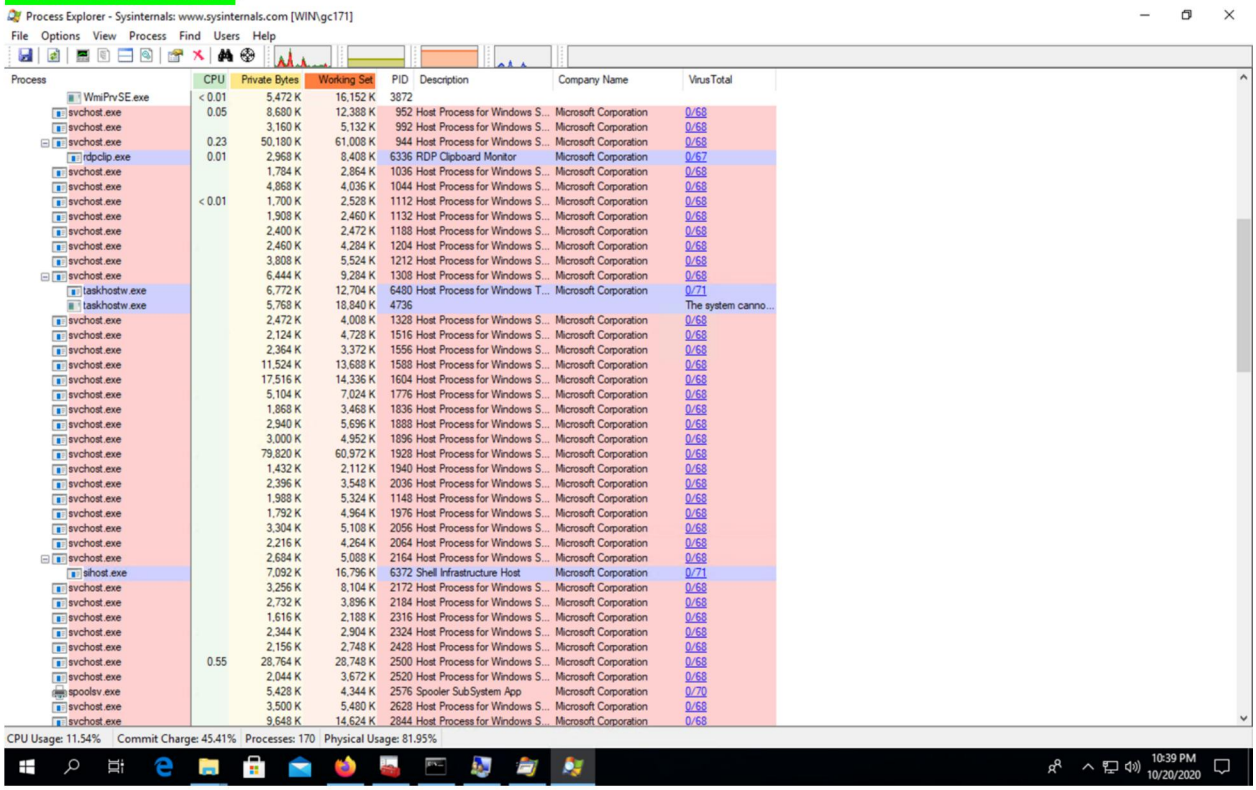


6. Stop the capture, clear the buffer, and leave Process Monitor running in preparation for later steps.

Part 3: Process Explorer

1. Download and unzip Process Explorer from here. Process Explorer is like a heavyweight version of the built-in Task Manager, showing all running programs, but it can do much more.
2. Play around a bit with it.
3. Try running the Skype that's included with Windows 10 (but do not login!). Find this program in Process Explorer. Right click the process, and view properties. Check the TCP/IP tab to see connections being made in realtime. At this time, turn off "resolve addresses".
4. One of the special abilities of Process Explorer is the ability to submit any running program to VirusTotal.com, a website that scans any file given to it with virtually every virus scanner on the market today. Right click the calculator process and choose "Check VirusTotal.com". The VirusTotal column will populate with a number, hopefully zero out of something (e.g., "0/67").
5. You can even check ALL running processes - do so from the menu: "Options" -> "VirusTotal.com" -> "Check VirusTotal.com". Wait for all the VirusTotal entries to populate. Did you get any results showing non-zero hits? If so, click the number to see details. If not, click on the Skype VirusTotal link to see details.
6. Now kill the calculator process using Process Explorer.

7. Paste a screenshot of Process Explorer showing all the running processes with their VirusTotal results.



Part 4: Wireshark

1. You know about Wireshark. Install it in your VM.

Part 5: The Malware

ULTRA-DANGER! Take EXTREME care in handling of the Windows malware linked below. Do not even *extract* it anywhere but the sandbox of the throwaway VM unless you seriously know what you're doing!

This is real malware recently analyzed by security researchers. It was the payload of a few spam email campaigns that included malicious javascript, macro-enabled Word document, etc. We're skipping the attack vector and analyzing the payload directly.

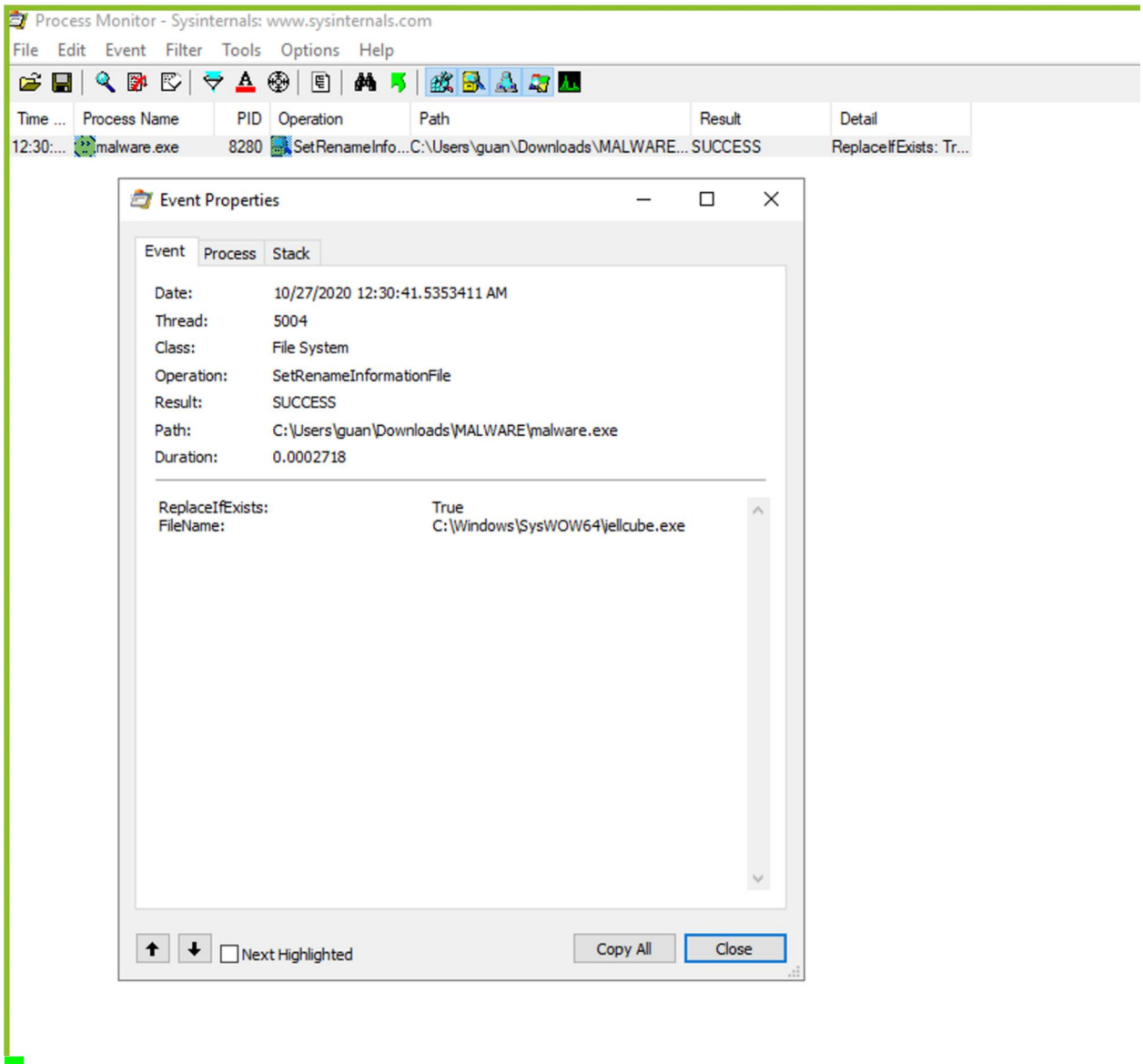
1. Download the malware to the VM:
   https://people.duke.edu/~tkb13/courses/ece560/homework/hw4/MALWARE.zip
2. Extract the malware. The ZIP password is:
   `infected`
3. At this time, take a snapshot of your VM (so if something goes wrong, you can restart at this point). You may need to shut down your VM to do so, depending on your hypervisor/cloud.

4. Ensure that Process Monitor, Process Explorer, and Wireshark are running.
5. In Wireshark, begin monitoring on the LAN connection. Leave it running for a few minutes to get a baseline for traffic on the system (as even idle Windows machines are notoriously chatty on the network). When you're satisfied, restart the capture to clear it.
6. In Process Monitor, clear the log and enable event capture.

This is the point of no return for the **first infection test**. Once you pass this point, you must complete this test fairly quickly or revert to the snapshot you took and start again. To help with this, read all the steps below before you start them. It's okay if you miss something or make a mistake, but you must revert the VM to the snapshot before you try again. The malware was fairly quiet in my analysis, but given enough time, it might wake up and start attacking other machines or taking unknown action at the behest of criminals. DO NOT LEAVE AN INFECTED VM LYING AROUND FOR MORE THAN 30 MINUTES.

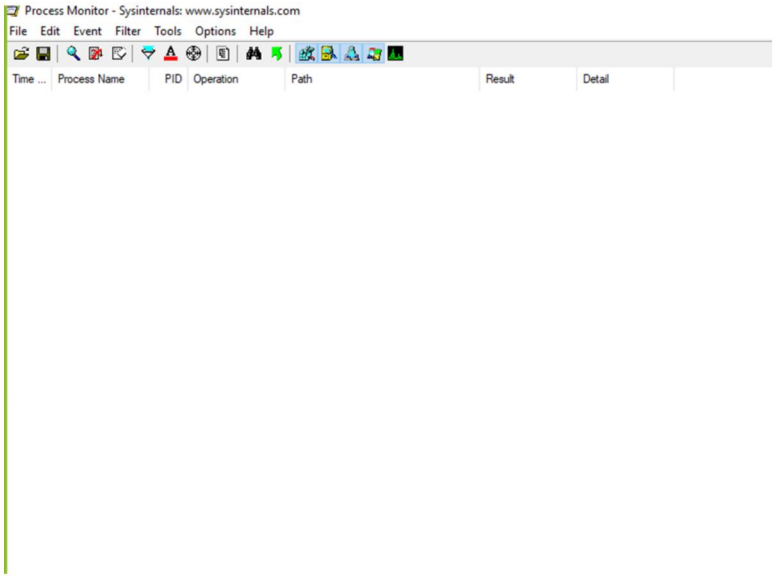
7. The malware is provided with no file extension to further reduce the risk of accidental execution. Rename the malware to `malware.exe`. The icon should change to an ambivalent face:

8. Execute the malware.
9. Find the malware's activity in Process Monitor. It does a LOT of registry reads, so you may want to filter some of that out.
10. Using Process Monitor, find out where the malware hides itself on the system.
    Hint: The malware uses a rename operation that shows up as SetRenameInformationFile in Process Monitor. What is the new path to the malware? Show a screenshot indicating this.

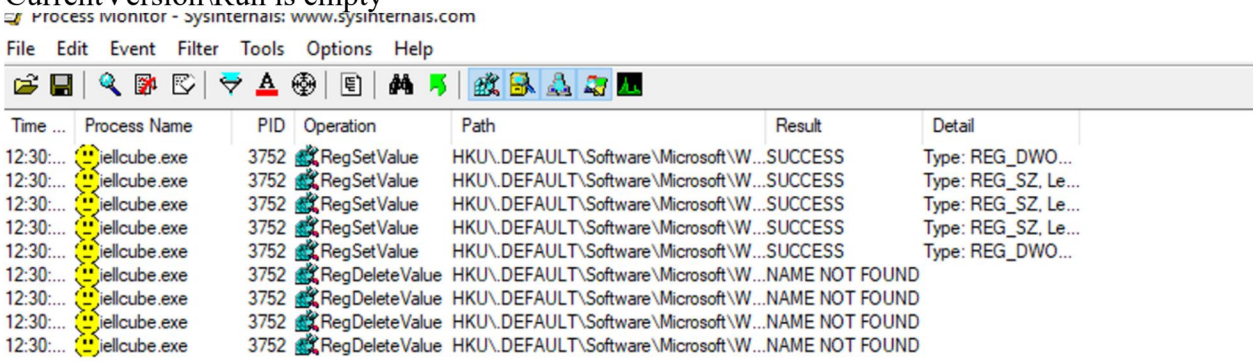


C:\Windows\SysWOW64\iellcube.exe

11. Now that we know the new filename of the malware, scroll further through Process Monitor to see activities under the new name.
12. After several seconds, the malware will *edit* (not just read) several registry settings. One of the registry entries contains "CurrentVersion\Run" (which you researched earlier in this problem). Based on this and your findings in step 10, how does this malware hide itself and remain persistent?



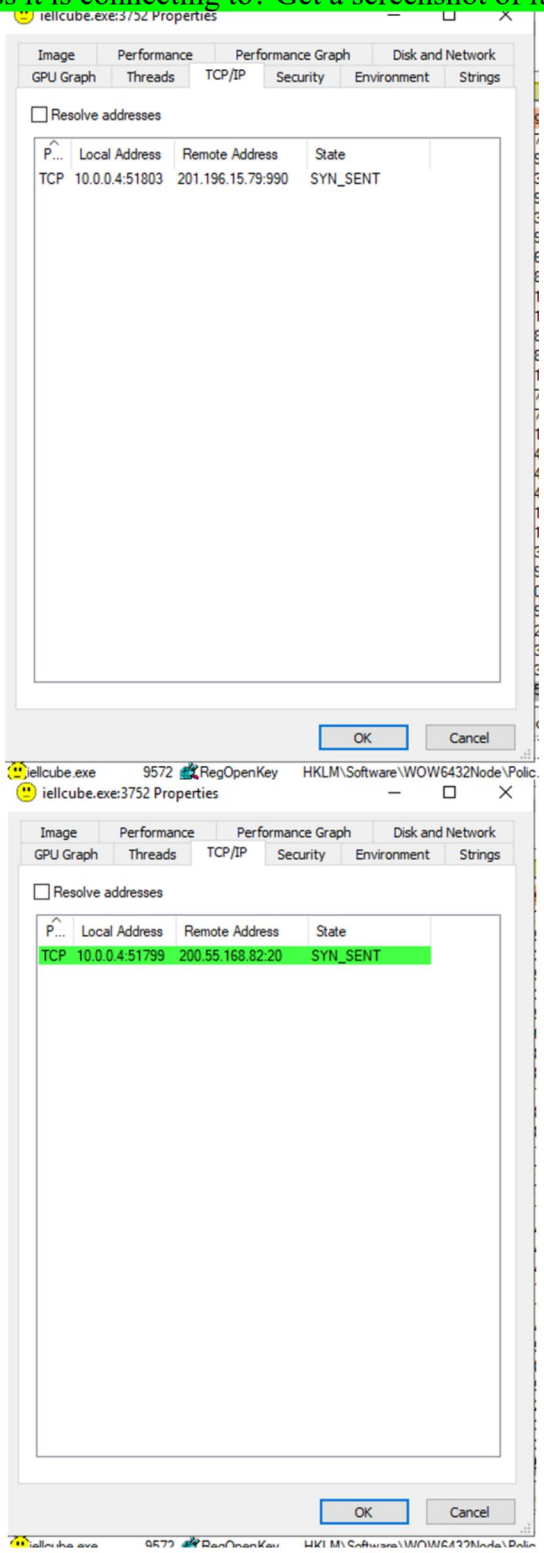
CurrentVersion\Run is empty



1 it change its name and location
2 it changes the register so every time when system boot it will auto load it

13. Using Process Explorer, identify the running malware resident in memory. Right click and open the properties, and view the TCP/IP tab. Ensure that “resolve addresses” is disabled, and monitor the tab for a few minutes. You should spot an outgoing connection. If you don’t, you probably missed the event -- revert to snapshot and try again, skipping the steps you’ve already done so you can catch the outgoing connection.

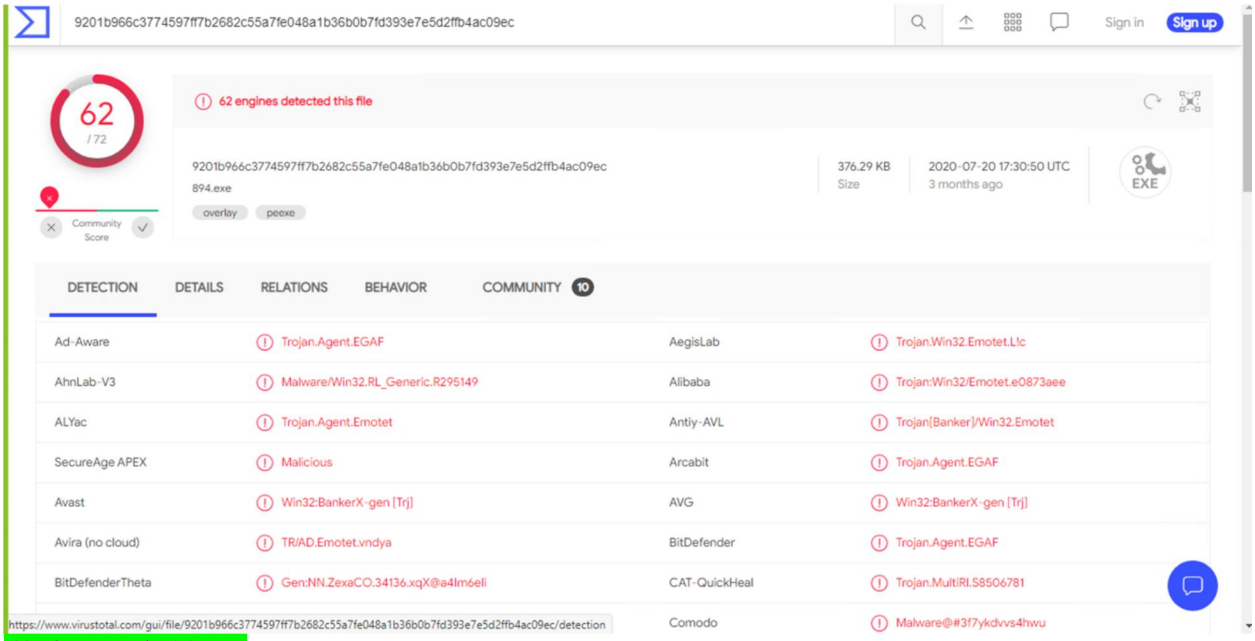
14. What is the IP address it is connecting to? Get a screenshot of it in Process Explorer.



it try to connect different ip

15. Use the VirusTotal option from Process Explorer to evaluate the malware. Click on the resulting number to see the detailed analysis. Paste a screenshot of the result. How many

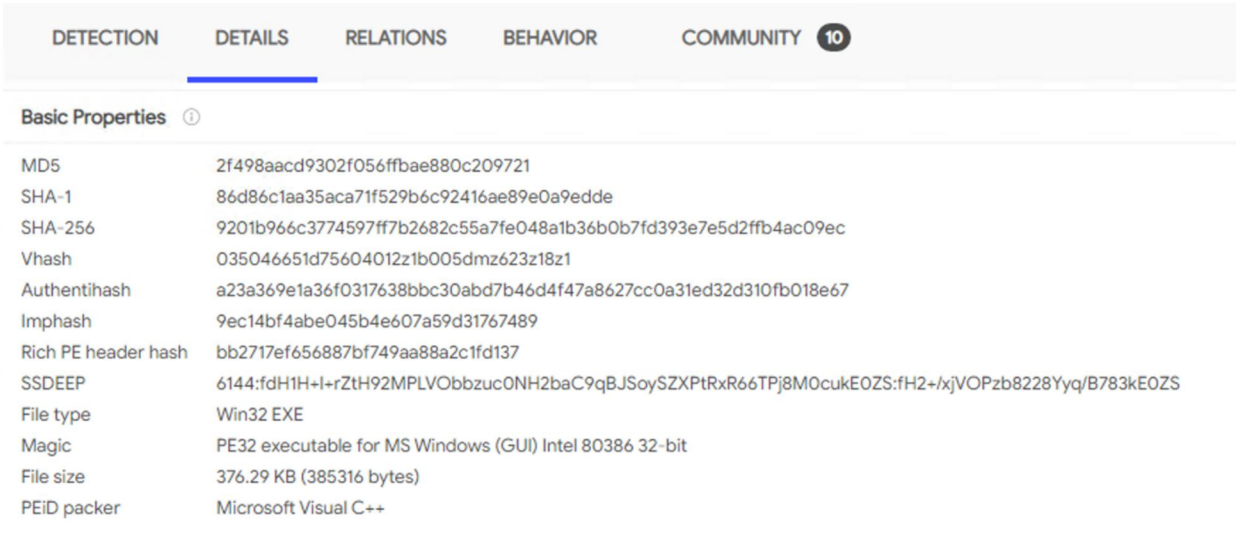
scanners detected it? How many did not? How does this make you feel?



62 detected. 10 not

it makes me think that there are still some room left for improvement for some security engine

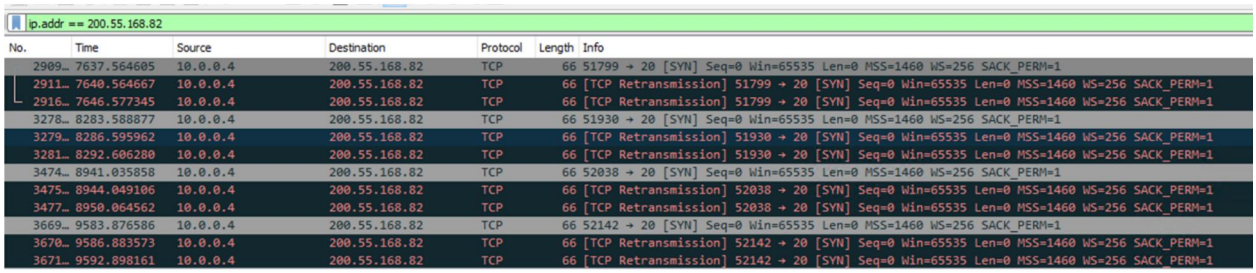
16. On VirusTotal.com, under "additional information", several long hex strings are listed -- what are these?



hash of malware

17. Now that we know the IP it connects to, set the Wireshark display filter to only show this IP address.

18. You'll find that most of the traffic is encrypted with HTTPS, but strangely, two requests are made on the HTTPS port (443), but the content is unencrypted HTTP. What are the URLs of these requests? Describe the content you see transmitted/received.

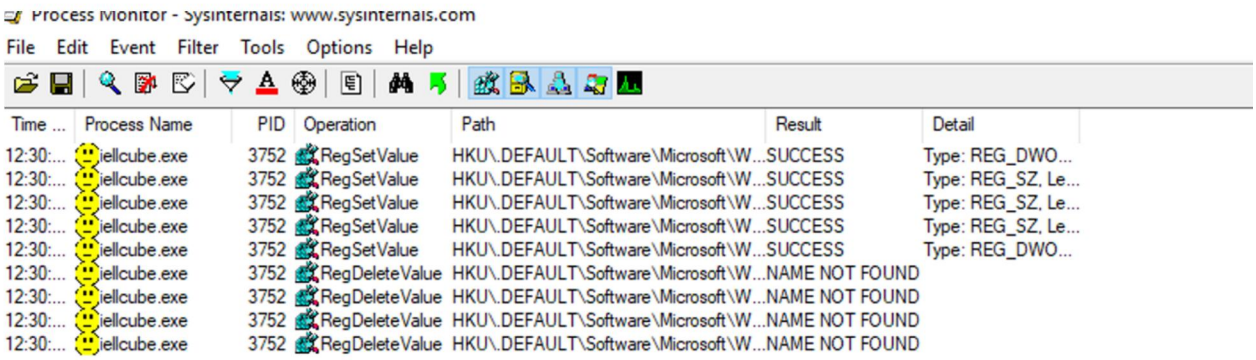


the malware doesn't establish consistent tcp connection to certain destination so I see nothing

19. Reboot the VM, and run Process Explorer. Confirm that the malware is again running.
20. Revert the VM to its pre-infection snapshot. Boot it and confirm the malware is not running. This concludes the **first infection test**.

Part 6: Developing a cure

21. Using the information you've gathered, what steps should you take to remove an instance of this malware from a real infected system (i.e., one where you don't have a known-good VM snapshot)?

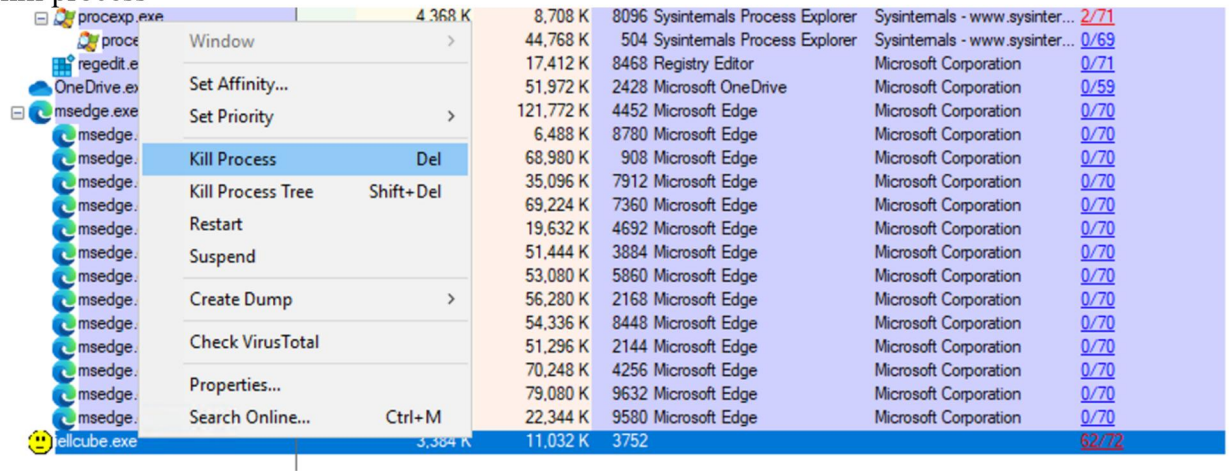


kill the process, delete binary file, trace all the registry change and delete them

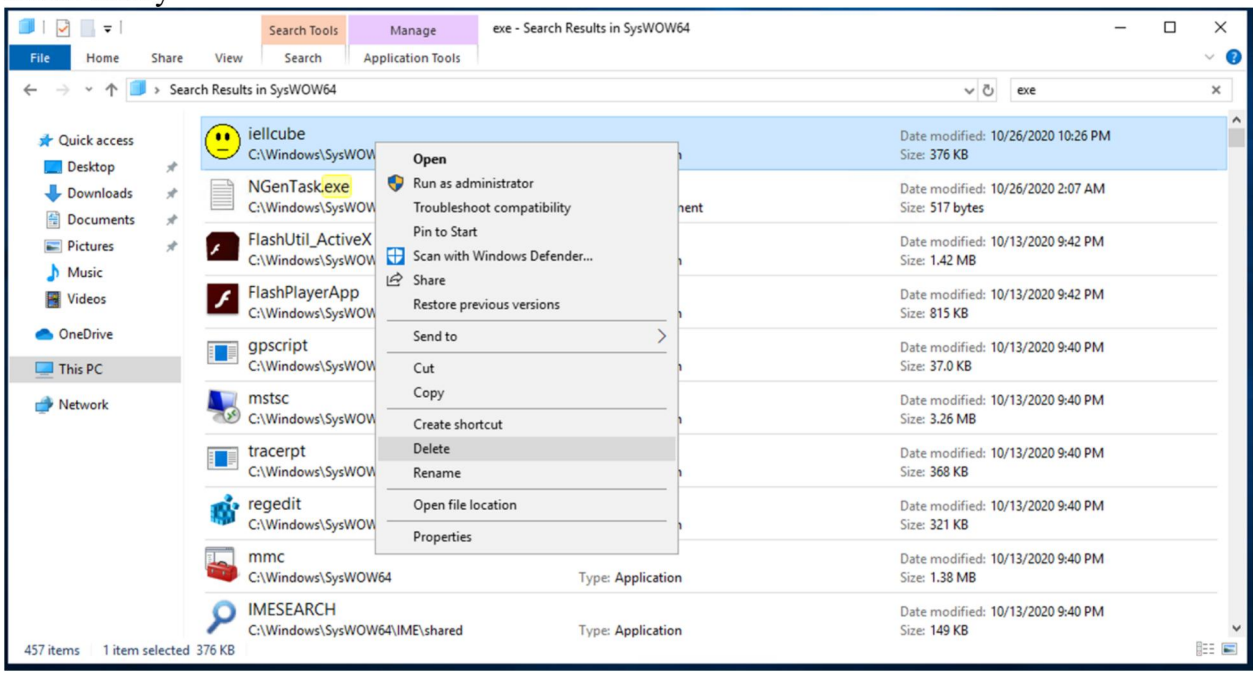
This is the point of no return for the **second infection test**. Again, once you pass this point, you must either succeed in clearing the infection or or revert to the snapshot you took and start again.

22. Run the malware on the clean VM again.
23. Perform your proposed procedure to remove the malware and reboot the VM. Using Process Explorer, verify the malware is gone. Did you get it? If it is not, try again until you successfully remove it. Note your actions and results as you go.

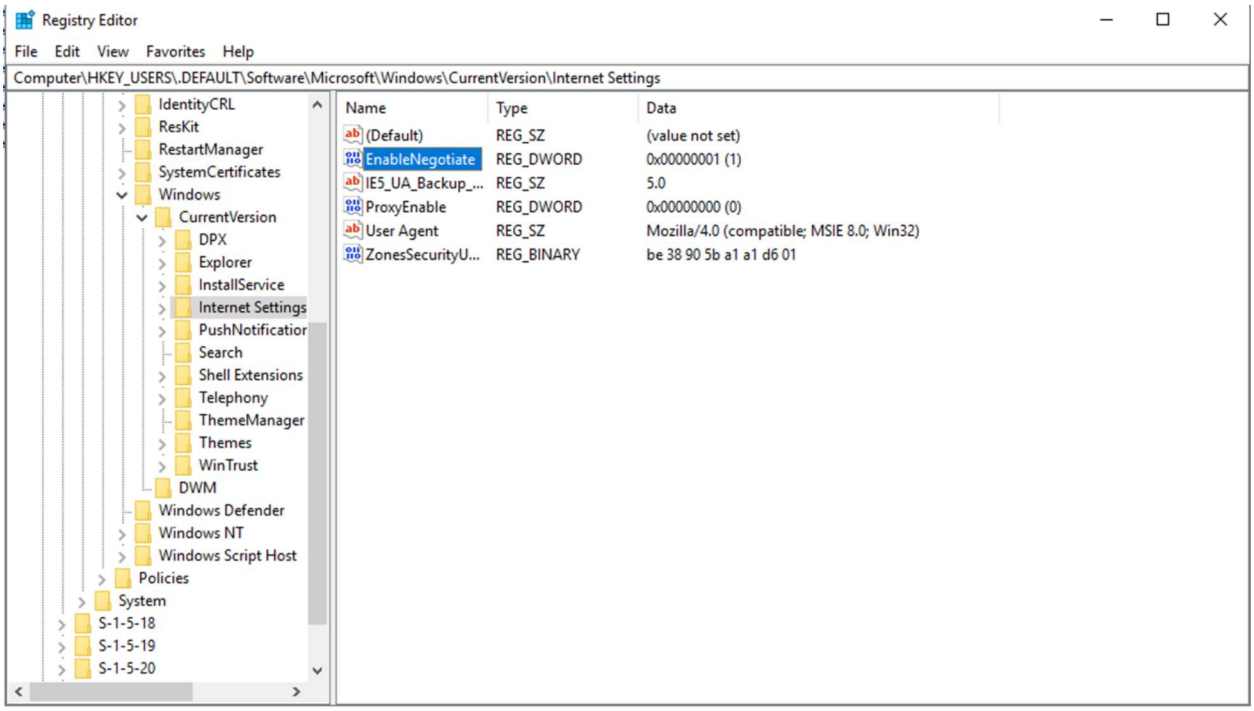
kill process

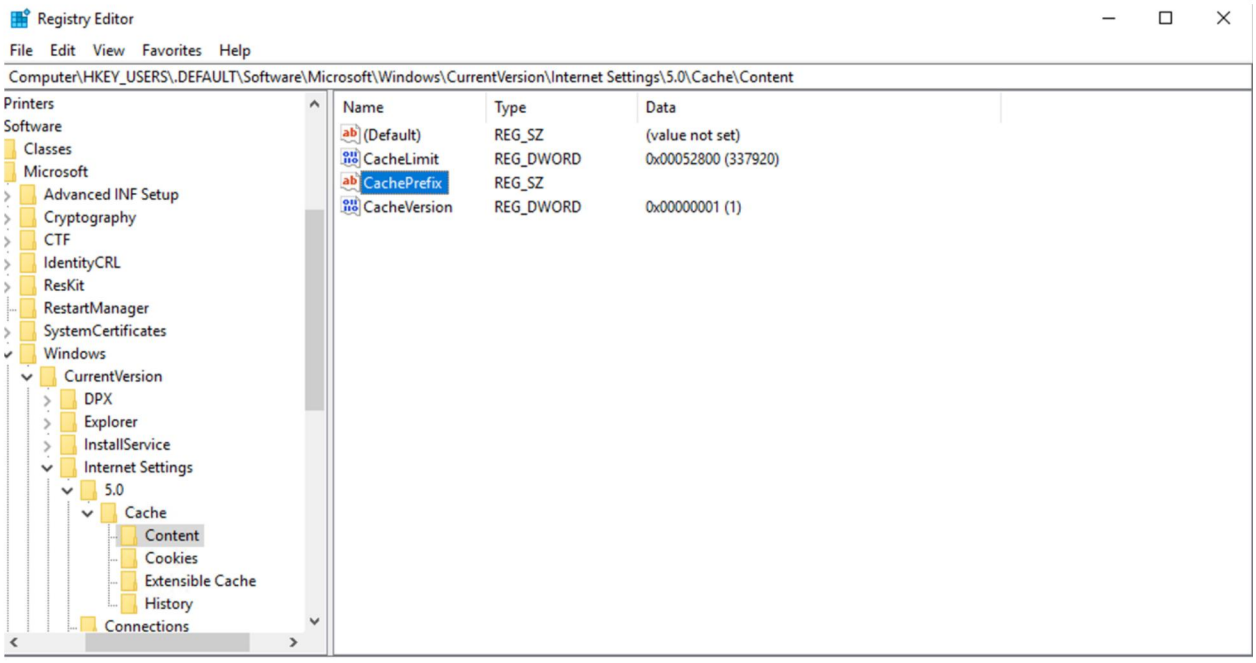


delete binary



delete reg





24. Once you are satisfied, revert to the snapshot again. This concludes the **second infection test**.

25. Now that the malware is apparently gone, how confident are you that you got it all? If you answered something like "100% confident", research the impact that hubris has had on human history (Icarus, Napoleon in Russia, the roaring '20s and great depression, the Vietnam War, the 2008 financial crisis, etc.), and revise your answer. How might some exceedingly clever malware have survived?

I think I get most of it. But it may still have some hidden copies I miss out.
Some clever malware may try to crypto itself and change it hash to a completely new binary to avoid detection. Then it may survive.

26. At any point, did our analysis uncover what this malware was designed to do? Does that worry you?

    it attempts to connect to some ip, that is an activity that malware try to steal user info. It will also try to attack other host as a source
    According to the documentation, it's a malware tries to steal user info & spamming. I think from its network behavior it makes sense

    it worries me that a lot of malware maybe trying to steal user info without user noticing it.

27. When you're done, shut down the VM.
28. To ensure the VM isn't accidentally used for something else later, after the homework deadline has passed, destroy the VM.

Background: The malware in question is a variant of Emotet, documented here. This specific variant is from here. It presumably is allowing remote control of infected machines for basically any purpose. This means that infected systems could easily be used as a botnet to conduct coordinated brute force login or DDOS attacks against internet sites, to retrieve any confidential files, to log user keystrokes, or to display any manner of advertising or malicious content to the users of the system.

~

13 Question 13: Malware Analysis 18 / 18

✓ **- 0 pts** **Correct**

- 1 pts Doesn't explain how malware hides itself

- 2 pts Doesn't explain that the hex strings are hashes

- 1 pts Doesn't explain how malware remains persistent

- 2 pts Doesn't (correctly) address how malware could have survived. Part 6, question 24 needs a more detailed answer.

- 1 pts Doesn't address whether analysis was worrying

- 1 pts Doesn't address feelings about 62/72 scanners detecting malware

- 2 pts Doesn't address what CurrentVersion\\Registry keys do

- 2 pts No answer for Part 5, question 18.

- 1 pts No answer Part 6, question 21

- 1 pts No description of actions taken. Part 6. Question 23

- 2 pts No screenshot indicating new malware path

- 2 pts No screenshot of process explorer. Part 3, question 7

- 1 pts No answer. Part 6, question 25

- 1 pts No answer. Part 6, question 26

- 18 pts No answer

- 16 pts One question answered

14 Late Penalty 0 / 0

✓ **- 0 pts** No penalty