Project 5: Forensics

This project is split into two parts, with the first checkpoint due on **Monday, November 18, 2019** at **6:00pm** and the second checkpoint due on **Friday, December 6, 2019** at **6:00pm**. The first checkpoint is worth 20 points, and the second checkpoint is worth 100 points.

This is a group project; you *SHOULD* work in **teams of two**. You *MUST* submit only one project per team. If you have trouble forming a team, post to Piazza's partner search forum.

Strict NO-leaks policy. In this project you play the role of a computer forensic analyst working to solve a murder case. Since you don't want to be fired for jeopardizing an ongoing criminal investigation, you need to follow a strict policy on collaboration. *You are bound by the Student Code not to communicate with anyone regarding any aspect of the case or your investigation (other than within your group or with course staff).* The number of pieces of evidence you find, the techniques you try, how successful said techniques are, the general process you follow, etc. are all considered part of your solution and must not be discussed with members of other groups.

Solutions *MUST* be submitted electronically in either one of the group member's github-dev repos, following filenames and solution formats specified in this handout. You *SHOULD* decide on which member's repo to use at the beginning of the project and ignore files distributed to the other partner's repository entirely.

"In general, computer forensics is rather ad hoc. Traditional rules of evidence are broken all the time. But this seems like a pretty egregious example."

– Bruce Schneier

Introduction

In this project, you will play the role of a digital forensic analyst and investigate a murder mystery.

A few days after Halloween 2015, a terrible crime occurred on the greens of Champaign Country Club in Urbana-Champaign. A hapless victim was found shot to death. The victim was last seen alive on November 4, 2015 early afternoon in Timpone's on Goodwin Avenue, and was discovered dead at approximately 11pm the same day.

In the crime scene, police officers noticed a whistle-blower complaint letter against a political entity, left open on the victim's computer. Upon further investigation, they shortlisted two suspects for the murder. They obtained a search warrant and were able to acquire hard disks of both suspects' computers. Investigators successfully created raw images of the disks for further investigation.

Your job is to conduct a digital forensic examination of the disk images and document any evidence related to the murder. If you find sufficient evidence, the suspect will be arrested and face trials.

Objectives

- Understand how computer use can leave persistent traces and why such evidence is often difficult to remove or conceal.
- Gain experience applying the security mindset to investigate computer misuse and intrusion.
- Learn how to retrieve information from a disk image without booting the operating system, and understand why this is necessary to preserve forensic integrity.

Guidelines

- You *SHOULD* work in a group of 2.
- Your answers may or may not be the same as your classmates'.
- We have generated files for you to submit your answers in. Please check your repository. You *MUST* submit your answers in the provided files.
- You *SHOULD* create a directory named `explanation/` and submit text files explaining how you got each answer and evidence file(s) that support your answers. This directory will not be graded but will be used to justify regrades and partial credit.
- Push your code to the master branch. We will only grade the latest commit on the master branch before the deadline.
- Each submission file contains an example of expected format. Failure to follow this format may result in 0 points for the section. You *MAY* delete the examples; they will be ignored (along with any other lines that start with #) when grading.

- Solution format is case-insensitive.
- General criteria for regrades and partial credit:
  - Decisions will be made solely based on explanations submitted to the `explanation/` directory by the checkpoint deadlines.
  - Failure to follow submission formats will result in 1 point deduction for the question.
  - Failure to provide a correct SHA256 hash value will result in 1 point deduction for the question, only if the correct evidence file is present in the `explanation/` directory. Otherwise, no points will be awarded.

Read this First

Collaboration: Strictly prohibited outside your group. As stated above, you are bound by the Student Code not to communicate with anyone regarding any aspect of the case or your investigation (other than within your group or with course staff). The number of pieces of evidence you find, the techniques you try, how successful said techniques are, the general process you follow, etc. are all considered part of your solution and must not be discussed with members of other groups. If anyone brings up the project, put your fingers in your ear, start yelling “LALALALA”, run around and refer them to your supervisor for an official spokesperson.

Getting Started

In this project, you will be conducting a forensic examination of several disk images. You will be examining the victim's disk in Checkpoint 1 and the suspects' disks in Checkpoint 2. You will perform both live and dead analysis on each disk. The tools and techniques you use for your analysis are up to you, but here are some suggestions to help you get started. You *MUST* use your own computer (or virtual machines) to perform the analysis.

General Knowledge A general working knowledge of Linux is undoubtedly helpful for this project. If you don't have this yet, you may need to spend time Googling and/or experimenting to get up to speed. TAs will also answer general Linux questions as a last resort. For an excellent reference book, try *UNIX and Linux System Administration Handbook* by Nemeth, Snyder, Hein, and Whaley (you can easily find a PDF copy from Google search). A general knowledge of disk partitioning may be useful as well (see `https://en.wikipedia.org/wiki/Disk_partitioning`).

Mounting Disk Images Throughout the investigation, you may want to directly access filesystem(s) on disk images. To mount a disk image:

1. Create a directory where you will mount the image.
   ```
   $ mkdir ~/Desktop/victim_fs
   ```
2. Use `fdisk` or `parted` to understand the disk's partition table.
   ```
   $ fdisk -l victim.raw
   ```
3. Choose which partition you want to mount and check partition information (e.g. partition offset, filesystem type).
   - `offset` = `number-of-sectors` × `sector-size`
4. Mount the partition.
   ```
   $ sudo mount --types <fs-type> --options loop,ro,noexec,offset=<offset>
   victim.raw ~/Desktop/victim_fs
   ```

Password Cracking Password crackers may be helpful in trying to brute-force decrypt password-protected files. Here are some popular tools:

- John the Ripper (`https://www.openwall.com/john/`) is the canonical Unix password cracker.
- Hydra (`https://github.com/vanhauser-thc/thc-hydra`) is a tool used to brute force remote login passwords.
- fcrackzip (`https://github.com/hyc/fcrackzip`) is a ZIP password cracker
- pdfcrack (`https://sourceforge.net/projects/pdfcrack/`) is a PDF password cracker.

When using a password cracker, it is wise to make sure that the password is not susceptible to a dictionary attack and does not use a restricted character set (e.g., lowercase letters, letters only, letters and numbers only) before spending time on a full brute-force crack. It is also a good idea to crack a very vulnerable password first to make sure you are using the tool correctly.

Metadata Viewer There exist multiple metadata viewers, but for this project, we suggest using ExifTool by Phil Harvey. It can be downloaded at `https://owl.phy.queensu.ca/~phil/exiftool/` . Follow the instructions on the website to view the metadata of evidence.

Ensuring Evidence Integrity In a digital forensics investigation, a forensic investigator ensures evidence integrity by calculating hashes of digital evidence. You can calculate and verify SHA hashes of files using `shasum` command or other variants.
For example, to calculate the hash of a file:

```
$ shasum --algorithm 256 <filename>
```

If you are provided with a text file containing expected values, you can have the command verify results against the values. For example, to verify integrity of downloaded disk images with `SHA256SUMS-disks.txt`, which contains SHA256 hashes of the images:

```
$ shasum --algorithm 256 --check SHA256SUMS-disks.txt
```

Performing Live Analysis Live analysis is a forensic technique in which the investigator examines a running copy of the target system. In this project, you will boot up the system present on the provided disk images using VMs. You can import pre-configured VMs using OVA files shared on `https://uofi.box.com/v/cs461-forensics-live-analysis` . You *SHOULD* keep default settings when importing the VMs. We recommend making a snapshot before running them for the first time so that the original state can be easily restored when necessary.

Performing Dead Analysis In dead analysis, a forensic investigator examines data artifacts from a target system without running the system. In this project, you will be analyzing the provided disk image files. We suggest using The Sleuth Kit (TSK), a collection of open-source forensics tools. For more information on TSK:

`https://wiki.sleuthkit.org/index.php?title=Help_Documents`

To perform dead analysis, you can install tools in your own environment or use a pre-configured Ubuntu VM available at `https://uofi.app.box.com/v/cs461-forensics-vm` . Login name is *forensics* and password is *changeme*.
The VM has following tools installed:

- The Sleuth Kit v4.4.2
  - `mmls`: displays the layout of a disk, including the unallocated spaces
  - `ffind`: finds allocated and unallocated file names that point to a given inode number
  - `fls`: lists allocated and deleted file names in a directory
  - `ils`: lists inode numbers and their details in a pipe delimited format
  - `icat`: extracts content of a file, which is specified by its inode number
  - `istat`: displays details about a given inode number in an easy to read format
  - `mactime`: takes input from the fls and ils tools to create a timeline of file activity
  - For the full list, see `https://wiki.sleuthkit.org/index.php?title=TSK_Tool_Overview`

- File-carving/recovery tools
  - foremost v1.5.7
  - Photorec v7.0
  - extundelete v0.2.4
- Password cracking tools
  - John the Ripper v1.8.0
  - Hydra v8.6
  - fcrackzip v1.0
  - pdfcrack v0.16
- ExifTool v10.80

Note: Feel free to install other tools or make system changes on the VM if needed. To enable guest features, VirtualBox users can insert the GuestAdditions CD image and run `sudo install-vbox-guest-additions` . For VMware product users, VMware Tools is already installed.

Useful link(s):

- SANS Forensics Cheatsheets – `https://digital-forensics.sans.org/community/cheat-sheets`
- Ubuntu Data Recovery – `https://help.ubuntu.com/community/DataRecovery`

Disk Image Files The disk image files you will investigate are available at `https://uofi.box.com/v/cs461-forensics-disks` . If you cannot access the URL, please enable the U of I Box service on `https://cloud-dashboard.illinois.edu/` .
Note: If you plan on using the provided Ubuntu VM mentioned in **Performing Dead Analysis**, download these files inside the VM.

At the URL, you will find 4 files (*Note:* `victim.raw.gz` will be released on November 6, 2019 and the suspect disk images will be released the week after):

- `victim.raw.gz` – victim's disk image
  - You *MUST* analyze this disk image in Checkpoint 1.
  - You *MAY* analyze this disk image in Checkpoint 2.
- `suspect-l.raw.gz` – prime suspect's disk image
  - You *MUST* analyze this disk image in Checkpoint 2.
- `suspect-w.raw.gz` – second suspect's disk image
  - You *MUST* analyze this disk image in Checkpoint 2.
- `SHA256SUM-disks.txt` – SHA256 checksums of disk image files
  - You *SHOULD* use this file to check integrity of the disk image files.

To decompress the downloaded gz files, use `gzip` command. For example,

```
$ gzip --decompress --stdout victim.raw.gz > victim.raw
```

- If you omit `--stdout` and the output redirection, the original compressed file will be removed.

5.1 Checkpoint 1 (20 points)

The deliverables for this project are your answers to the questions below. Your answers should be *complete* but *concise* following the submission template.

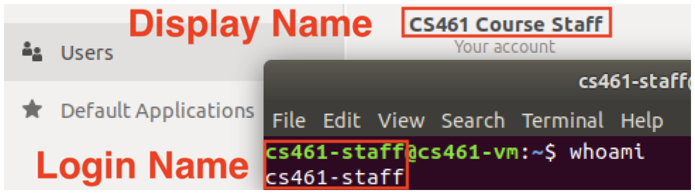
Exploring Victim's Traces

In this part of the project, you will be exploring traces left on the victim's hard disk.

5.1.1 Login Name (2 points) (*Difficulty: Easy*)

What is the victim's login name (or textual user ID) used on his computer?

Note: Login name vs. display name – `https://unix.stackexchange.com/questions/307452` An example is shown below.



```
cs461-staff@cs461-vm:~$ whoami
cs461-staff
```

What to submit: Submit a text file named `cp1.1.login.txt` that contains the victim's login name.

Example content of `cp1.1.login.txt`:

```
cs461-staff
```

5.1.2 Timezone (2 points) (*Difficulty: Easy*)

Which timezone is the victim's computer set with?
(a) AKST
(b) CST
(c) EST
(d) PST
(e) MST

What to submit: Submit a text file named `cp1.2.timezone.txt` that contains the letter of your choice.

Example content of `cp1.2.timezone.txt`:

```
k
```

5.1.3 Conversations (4 points) (*Difficulty: Easy*)

Whom did the victim have conversation(s) with? List each chat username in separate line, in order of occurrences. If the victim had multiple conversations with the same person in different times, list the username as many times as there were distinct conversations.

For example, based on following chat history, *bob*, *eve*, *eve* should be listed in the solution file.

```
<victim's chat history>
(with bob@chat.org) @ Apr 2 7:00pm
(with eve@chat.org) @ Apr 3 10:00am
(with eve@chat.org) @ Apr 3 2:00pm
```

What to submit: Submit a text file named `cp1.3.usernames.txt` that contains the list of chat **usernames** (not display names) the victim had one or more conversations with in chronological order.

Example content of `cp1.3.usernames.txt`:

```
bob
eve
eve
```

5.1.4 Evidence File (4 points) (*Difficulty: Easy*)

A file found on the computer that suggests a possible reason for the murder.

1. (2 pts) What is the name of the file? Submit the name and SHA256 checksum of the file.
2. (2 pts) When was this file last modified? Submit the date time in *MMddhhmm* 24-hr format (*MM*=Month, *dd*=day, *hh*=hour, *mm*=minute), disregarding timezone information.

What to submit:

1. Submit a text file named `cp1.4.filename.txt` that contains the name and SHA256 checksum of the file.
2. Submit a text file named `cp1.4.modtime.txt` that contains the last modified time of the file.

Example content of `cp1.4.filename.txt`:

```
# filename in first line
# SHA256 checksum in second line
filename.ext
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Example content of `cp1.4.modtime.txt`:

```
# September 30th 23:59
09302359
```

5.1.5 Attack (8 points) (*Difficulty: Medium*)

Logs on the victim's computer suggest that someone had attacked and might have tampered with the system.

1. (1 pts) When did the attacker make the first contact to the victim's computer? A contact can be either successful login or failed attempt. Submit the date time in *MMddhhmm* 24-hr format (*MM*=Month, *dd*=day, *hh*=hour, *mm*=minute), disregarding the timezone information.

2. (4 pts) List all remote IPv4 addresses the attack originated from. If there are multiple attacker IP addresses, include one IP address per each line.

3. (3 pts) What was the IPv4 address assigned to the victim's computer during the attack?

What to submit:

1. Submit a text file named `cp1.5.attacktime.txt` that contains the first attack time.
2. Submit a text file named `cp1.5.attackerip.txt` that contains a list of IPv4 addresses the attack came from. Each line in the file contains one IP address.
3. Submit a text file named `cp1.5.victimip.txt` that contains the IPv4 address assigned to the victim's computer during the attack.

Example content of `cp1.5.attacktime.txt`:

```
# September 30th 23:59
09302359
```

Example content of `cp1.5.attackerip.txt`:

```
1.2.3.4
5.6.7.8
```

Example content of `cp1.5.victimip.txt`:

```
127.0.0.1
```

Checkpoint 1: Submission Checklist

Inside `Forensics` directory in your github-dev repo, you will have the auto-generated files named as below. Make sure that your answers for all tasks up to this point are submitted in the following files before **Monday, November 18, 2019** at **6:00pm**:

- `partners.txt` [one netid per line]
- `cp1.1.login.txt`
- `cp1.2.timezone.txt`
- `cp1.3.usernames.txt`
- `cp1.4.filename.txt`
- `cp1.4.modtime.txt`
- `cp1.5.attacktime.txt`
- `cp1.5.attackerip.txt`
- `cp1.5.victimip.txt`

5.2 Checkpoint 2 (100 points)

The deliverables for this project are your answers to the questions below. Your answers should be *complete* but *concise* following the submission template.

Investigating Suspect's Traces

In this part of the project, you will be exploring traces left by the suspects. You *MAY* need to revisit the victim's disk to help your investigation. Your answers *MUST* be based on evidence found from the disk image(s) specified in each section.

- 5.2.1 – 5.2.8: prime suspect (`suspect-l.raw`)
- 5.2.9: second suspect (`suspect-w.raw`)
- 5.2.10: victim and both suspects

5.2.1 [`suspect-l`] Live Analysis (11 points) (*Difficulty: Medium*)

When needed, you *MAY* perform dead analysis on the disk.

1. (3 pts) Try booting the prime suspect's computer. What operating system does it boot by default? Submit the distribution name and release (or version) number of the default OS in separate lines.

2. (5 pts) What potentially dangerous behavior(s) does the default OS present? Find a script file that executes the specific behavior. Submit the name and SHA256 checksum of the file in separate lines.

3. (3 pts) The hard disk contains another operating system that was primarily used. Submit the distribution name and release (or version) number of the primary OS in separate lines.

Note: The OS distribution release number is not equivalent to the kernel number.
See `https://whatsmyos.com/` and `https://unix.stackexchange.com/questions/6345` .

What to submit:

1. Submit a text file named `cp2.1.default.txt` that contains the default OS information.
2. Submit a text file named `cp2.1.script.txt` that contains the name and SHA256 checksum of the script file.
3. Submit a text file named `cp2.1.primary.txt` that contains the primary OS information.

Example content of `cp2.1.default.txt`:

```
centos
7.6
```

Example content of `cp2.1.script.txt`:

```
# filename in first line
# SHA256 checksum in second line
filename.ext
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Example content of `cp2.1.primary.txt`:

```
ubuntu
18.04
```

5.2.2 [`suspect-1`] Login Name (2 points) (*Difficulty: Easy*)

What is the prime suspect's login name (or textual user ID) on his/her computer?

What to submit: Submit a text file named `cp2.2.login.txt` that contains the prime suspect's login name.

Example content of `cp2.2.login.txt`:

```
cs461-staff
```

5.2.3 [`suspect-1`] Conversations (5 points) (*Difficulty: Easy*)

You have investigated the victim's chat history in 5.1.3. Now, you will do the same for the prime suspect.
Whom did the prime suspect have conversation(s) with? List each chat username in separate line, in the order of occurrences. If the suspect had multiple conversations with the same user at different times, list the username as many times as there were distinct conversations. This does not equal to the message count transferred back and forth for single conversation. *Note:* Consider only the chat history similar to Checkpoint 1.
What to submit: Submit a text file named `cp2.3.usernames.txt` that contains the list of usernames the prime suspect had conversations with.

Example content of `cp2.3.usernames.txt`:

```
bob
eve
eve
```

5.2.4 [`suspect-1`] Search History (6 points) (*Difficulty: Easy*)

Investigate the prime suspect's web history.

1. (5 pts) Is there any indication that the prime suspect could have been involved in attacking the victim's computer or the murder? Find any website that is potentially related to the case. List at least 5 full URLs, one URL per line. **Each of the 5 URLs *MUST* be from a different domain.** For example, two different Google search result pages count as 1.

2. (1 pts) What did the prime suspect plan to use as a weapon to murder the victim? *Note:* you may consider any toy evidence found in this part lethal and dangerous.
   (a) Cyanide
   (b) Infinity Gauntlet
   (c) Lightsaber
   (d) Nerf gun
   (e) Toxic waste candies

What to submit:

1. Submit a text file named `cp2.4.websites.txt` that contains a list of visited websites that are related to the murder.
2. Submit a text file named `cp2.4.weapon.txt` that contains the letter of your choice.

Example content of `cp2.4.websites.txt`:

```
https://www.google.com/maps
https://www.cnet.com/news/
```

Example content of `cp2.4.weapon.txt`:

```
k
```

5.2.5 `[suspect-1]` Encrypted File (9 points) (*Difficulty: Medium*)

The prime suspect left an encrypted zip file on his/her computer.

1. (2 pts) Find this file and calculate its SHA256 checksum.
2. (5 pts) What is the password used to encrypt the zip file?
3. (2 pts) Decrypt/extract the file and calculate the content's SHA256 checksum.

What to submit:

1. Submit a text file named `cp2.5.encrypted.txt` that contains the name and SHA256 checksum of the encrypted file.
2. Submit a text file named `cp2.5.zippassword.txt` that contains the password of the encrypted file.
3. Submit a text file named `cp2.5.decrypted.txt` that contains the name and SHA256 checksum of the decrypted content.

Example content of `cp2.5.encrypted.txt`:

```
# filename in first line
# SHA256 checksum in second line
filename.ext
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Example content of `cp2.5.zippassword.txt`:

```
p4ssw0rd
```

Example content of `cp2.5.decrypted.txt`:

```
# filename in first line
# SHA256 checksum in second line
filename.ext
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

5.2.6 [`suspect-1`] Attack (24 points) (*Difficulty: Medium*)

The prime suspect attempted to remotely access the victim's computer via SSH.

1. (4 pts) What tools helped the prime suspect to gain remote access to the victim's computer? Consider what network services could have been involved to gain the access. List the command name of each tool in separate lines, in order of occurrence.
2. (4 pts) List IPv4 address(es) assigned to the prime suspect's computer during the attack.
3. (4 pts) What was the public key presented by the victim's computer when the prime suspect established SSH connections? Find the public key file and calculate its SHA256 checksum.
4. (4 pts) What was the private key used by the victim's computer when the prime suspect established SSH connections? Find the private key file and calculate its SHA256 checksum.
5. (4 pts) Which user on the victim's computer did the prime suspect remotely log in as? If the suspect tried more than one login name, submit the last one he/she tried to access.
6. (4 pts) What is the password the prime suspect used to access the victim's computer?

What to submit:

1. Submit a text file named `cp2.6.tools.txt` that contains a list of tools used to gain the remote access.
2. Submit a text file named `cp2.6.ip.txt` that contains a list of IPv4 address(es) assigned to the prime suspect's computer during the attack.
3. Submit a text file named `cp2.6.publickey.txt` that contains the name and SHA256 checksum of the public key file.
4. Submit a text file named `cp2.6.privatekey.txt` that contains the name and SHA256 checksum of the private key file.
5. Submit a text file named `cp2.6.remotelogin.txt` that contains the login name used by the prime suspect to access the victim's computer.
6. Submit a text file named `cp2.6.victimpassword.txt` that contains the password used by the prime suspect to access the victim's computer.

Example content of `cp2.6.tools.txt`:

```
zip
john
```

Example content of `cp2.6.ip.txt`:

```
1.2.3.4
5.6.7.8
```

Example content of `cp2.6.publickey.txt`:

```
# filename in first line
# SHA256 checksum in second line
filename.ext
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Example content of `cp2.6.privatekey.txt`:

```
# filename in first line
# SHA256 checksum in second line
filename.ext
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Example content of `cp2.6.remotelogin.txt`:

```
cs461-staff
```

Example content of `cp2.6.victimpassword.txt`:

```
p4ssw0rd
```

5.2.7 [`suspect-1`] Escape Plan (13 points) (*Difficulty: Medium*)

The prime suspect may have planned an after-murder escape scenario.

1. (3 pts) The prime suspect contacted someone else through email to plan his/her escape. Find the email address he/she contacted.
2. (4 pts) Where does the escape take place? Submit the GPS coordinates, latitude and longitude in separate lines. Trim the number to the 3rd decimal digit (e.g. -1.2345 -> -1.234).
   - Signs for four directions in GPS coordinates: North (+), South (-), East (+), West (-).
3. (3 pts) What time did the prime suspect originally plan to escape? Submit the time in *hhmm* 24-hr format (hh=hour, mm=minute).
4. (3 pts) What time did the prime suspect decide to escape in the end? Submit the time in *hhmm* 24-hr format (hh=hour, mm=minute).

What to submit:

1. Submit a text file named `cp2.7.email.txt` that contains the email address the prime suspect contacted.

2. Submit a text file named `cp2.7.location.txt` that contains the GPS coordinates of the escape location, latitude and longitude in separate lines.
3. Submit a text file named `cp2.7.originaltime.txt` that contains the original escape time.
4. Submit a text file named `cp2.7.finaltime.txt` that contains the final escape time.

Example content of `cp2.7.email.txt`:

```
cs461-staff@illinois.edu
```

Example content of `cp2.7.location.txt`:

```
# latitude in first line
# longitude in second line
1.234
5.678
```

Example content of `cp2.7.originaltime.txt`:

```
2330
```

Example content of `cp2.7.finaltime.txt`:

```
2330
```

5.2.8 [`suspect-l`] Suspicious activity (5 points) (*Difficulty: Easy*)

The prime suspect deleted some files from his/her computer. Find one that seems the most incriminating and answer following question.
What is the name of the deleted file?
What to submit: Submit a text file named `cp2.8.filename.txt` that contains the name of the file.

Example content of `cp2.8.filename.txt`:

```
filename.ext
```

5.2.9 [`suspect-w`] Second Suspect's Plan (20 points) (*Difficulty: Medium*)

In this part, you will be investigating the second suspect's disk (`suspect-w.raw`). The second suspect deleted a file containing his/her murder plan from the computer. Recover the file and answer following questions.

1. (5 pts) What is the name of the deleted file that contains the second suspect's plan?
2. (5 pts) When was the file deleted from the second suspect's computer? Submit the date time in *MMddhhmm* 24-hr format (*MM*=Month, *dd*=day, *hh*=hour, *mm*=minute).
3. (5 pts) Recover the deleted file and calculate its SHA256 checksum.
4. (5 pts) What did the second suspect plan to use as a weapon to murder the victim? *Note:* you may consider any toy evidence found in this part lethal and dangerous.
   (a) Cyanide

(b) Infinity Gauntlet
(c) Lightsaber
(d) Nerf gun
(e) Toxic waste candies

What to submit:

1. Submit a text file named `cp2.9.planfilename.txt` that contains the name of the file.
2. Submit a text file named `cp2.9.plandeletedtime.txt` that contains the file deletion time.
3. Submit a text file named `cp2.9.planhash.txt` that contains the file's SHA256 checksum.
4. Submit a text file named `cp2.9.planweapon.txt` that contains the letter of your choice.

Example content of `cp2.9.planfilename.txt`:

```
filename.ext
```

Example content of `cp2.9.plandeletedtime.txt`:

```
# September 30th 23:59
09302359
```

Example content of `cp2.9.planhash.txt`:

```
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Example content of `cp2.9.planweapon.txt`:

```
k
```

5.2.10 [`all`] Final Decision (5 points) (*Difficulty: Easy*)

Based on the evidence you have gathered, which suspect do you think murdered the victim? You can provide additional explanations in the `explanation/` directory to support your argument.

(a) Prime suspect (owner of `suspect-l.raw`)
(b) Second suspect (owner of `suspect-w.raw`)

What to submit: Submit a text file named `cp2.10.murderer.txt` that contains the letter of your choice.

Example content of `cp2.10.murderer.txt`:

```
k
```

Checkpoint 2: Submission Checklist

Inside `Forensics` directory in your github-dev repo, you will have the auto-generated files named as below. Make sure that your answers for all tasks up to this point are submitted in the following files before **Friday, December 6, 2019** at **6:00pm**:

- `partners.txt` [one netid per line]
- `cp2.1.default.txt`
- `cp2.1.script.txt`
- `cp2.1.primary.txt`
- `cp2.2.login.txt`
- `cp2.3.usernames.txt`
- `cp2.4.websites.txt`
- `cp2.4.weapon.txt`
- `cp2.5.encrypted.txt`
- `cp2.5.zippassword.txt`
- `cp2.5.decrypted.txt`
- `cp2.6.tools.txt`
- `cp2.6.ip.txt`
- `cp2.6.publickey.txt`
- `cp2.6.privatekey.txt`
- `cp2.6.remotelogin.txt`
- `cp2.6.victimpassword.txt`
- `cp2.7.email.txt`
- `cp2.7.location.txt`
- `cp2.7.originaltime.txt`
- `cp2.7.finaltime.txt`
- `cp2.8.filename.txt`

- `cp2.9.planfilename.txt`
- `cp2.9.plandeletedtime.txt`
- `cp2.9.planhash.txt`
- `cp2.9.planweapon.txt`
- `cp2.10.murderer.txt`