CSC153: Independent Project - Anti-Forensics

Curtis Botonis, Ryan Kozak, Rongguang ou

SACRAMENTO STATE

2019-12-04

Project Goal

The goal of our project was to explore and develop various anti-forensic methodologies. In other words, we were looking to apply our knowledge of computer forensics to obfuscate data and destroy data, such that other forensic analysts could not discover or recover said data during an investigation. This project was to simulate a scenario in which a suspect had advanced knowledge of computer forensics, anti-forensic tools, and how to use them.

Tools Methodologies

There are many tools that exist for encrypting data, as well as destroying data. For this project we chose to explore each of the tools listed below. After using each tool we examined the results using our own knowledge of forensic investigation tools and techniques. Through each iteration we were able to validate the claims each tool makes about encryption or destruction of data, as well as the proper ways to use them.

Anti-Forensic Tools

- *Veracrypt* (Encryption)
- *sfdisk* (Data Hiding)
- *Eraser* (Data Destruction)
- *CCleaner* (Data Destruction)

Other Tools Utilized

- *dd* (Forensic Analysis)
- *dcfldd* (Forensic Analysis)
- *OSForensics* (Forensic Analysis)
- *Autopsy* (Forensic Analysis)

Data Obfuscation

In this project we, we've broken down data hiding techniques into two categories, Stenography, and Cryptography. The difference between them is well explained in *Module 7 Slide 12*, quoted below.

- Cryptography

  - Does not hide the communication.
  - Encodes the data to prevent eavesdroppers from understanding the content.
  - Presence of encrypted data may cause suspicions.

- Stenography
  - Hides the communication.
  - The data may or not be encrypted.
  - If they don't know about it, how can they be suspicious?

Data Destruction

We were seeking ways in which to destroy data on a drive such that it cannot be recovered by a forensic analyst.

Scenario

Overview

As a demonstration of our research, we've developed a scenario in which a suspect has used the tools and techniques described in the previous sections to hide sensitive information from investigators. We will cover in detail what the suspect has done, and then play the role of an investigator trying to find the sensitive information.

Sensitive Information

The sensitive information hidden by the suspect consists of a public/private key pair used to SSH into a server determined by investigators to be used for illegal activity. There is also a secret text file, which may contain additional information about their operation. The three files are listed below.

- id_rsa
- id_rsa.pub
- TopSecret.txt

Encryption of Data

To encrypt the sensitive information on the drive, we've used *Veracrypt*. While this tools provides a mechanism for encrypting an entire disk, we've chosen to use an encrypted container. This is an encrypted file, that remains static in size after its creation. This container, because it is handled as a regular file, is extremely portable. However, it can be mounted and used as a typical drive using Veracrypt and the correct decryption keys.

Hidden Volume

As an additional measure of protection, the sensitive information stored in the encrypted container will reside on a hidden volume. The hidden volume is explained in *Veracrypt's Documentation*, quoted below.

> It may happen that you are forced by somebody to reveal the password to an encrypted volume. There are many situations where you cannot refuse to reveal the password (for example, due to extortion). Using a so-called hidden volume allows you to solve such situations without revealing the password to your volume.

As you can see in Figure 1, a hidden volume uses the fact that encrypted data always appears as random data, regardless of whether or not it contains any actual information. *The difference between free space and files can only be determined once the container is decrypted.*

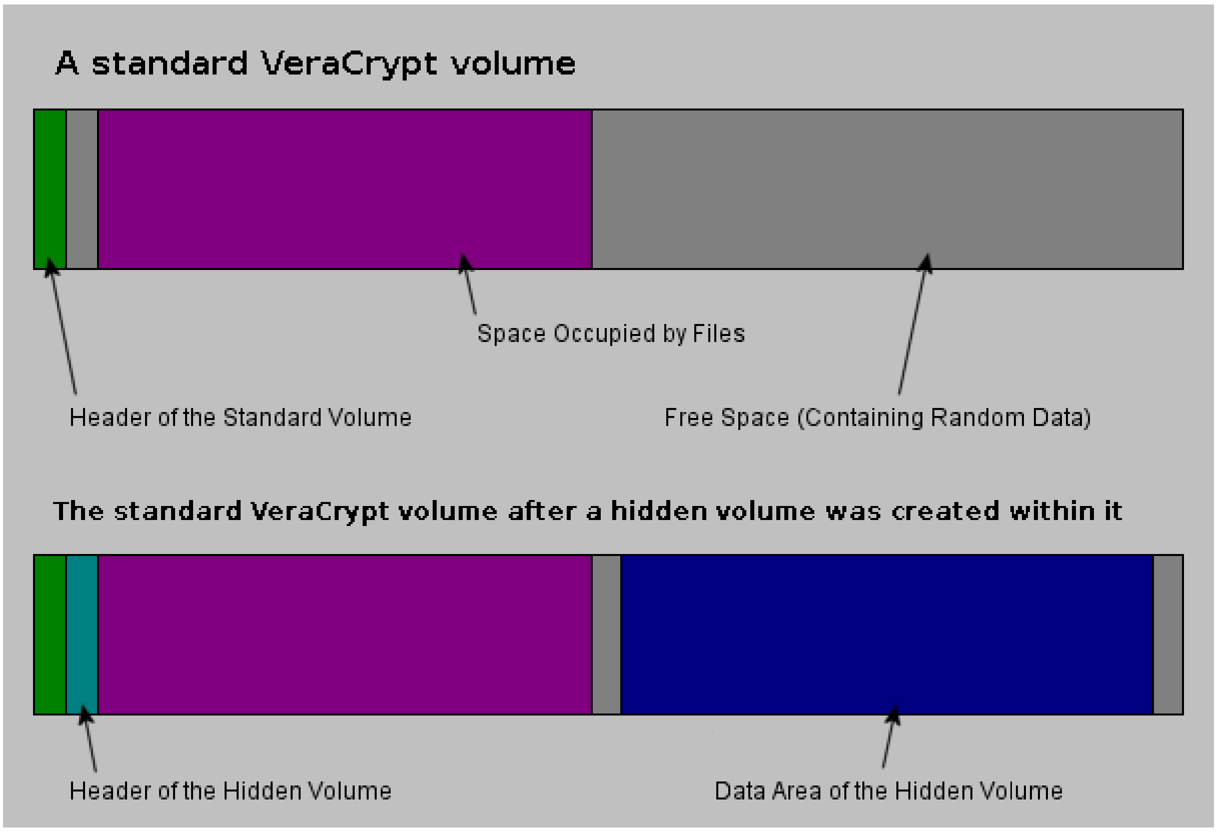
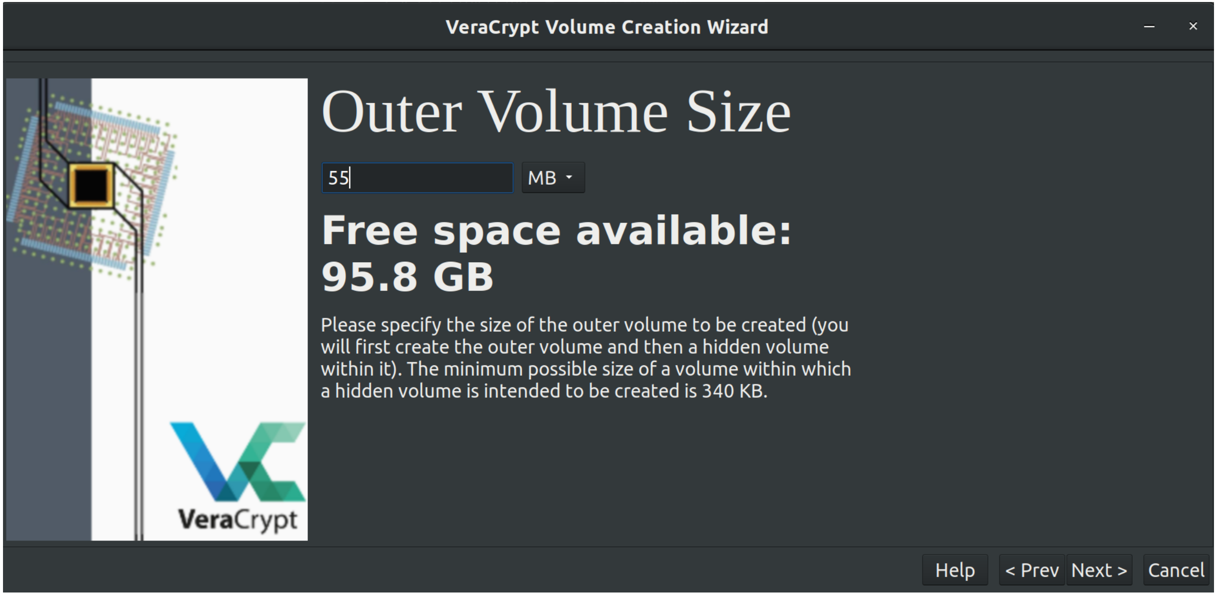

A standard VeraCrypt volume
Space Occupied by Files
Header of the Standard Volume
Free Space (Containing Random Data)
The standard VeraCrypt volume after a hidden volume was created within it
Header of the Hidden Volume
Data Area of the Hidden Volume

Figure 1: Hidden container before and after hidden volume creation, *source*.

The encrypted container we created was 55`MB` in size, and named `emirc`.


VeraCrypt Volume Creation Wizard

Outer Volume Size

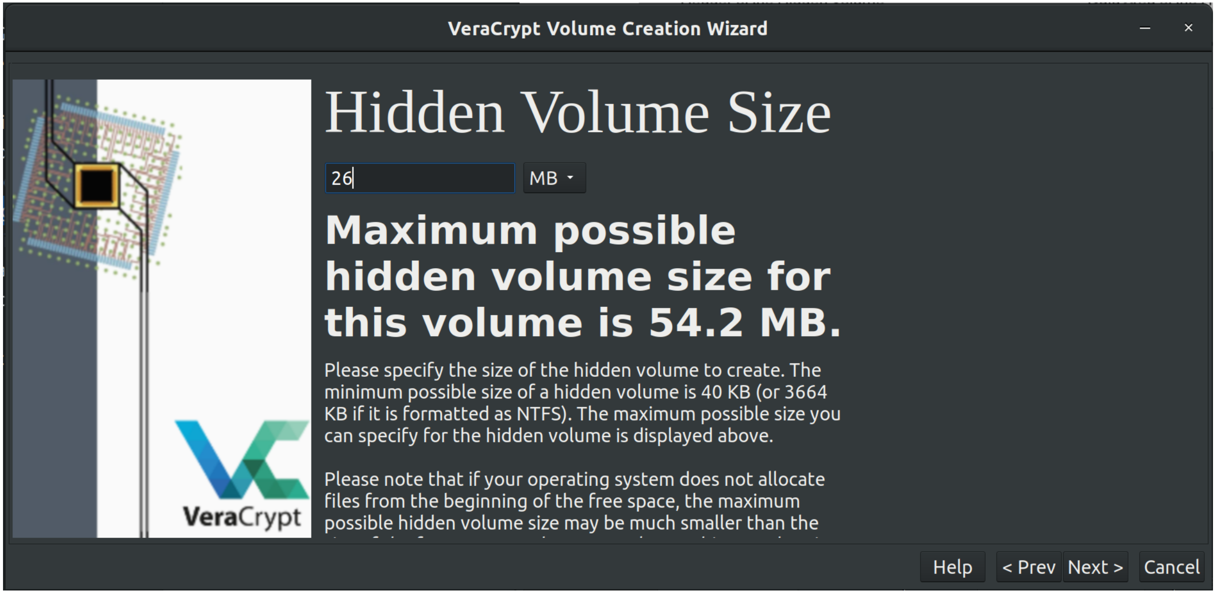
55 MB

Free space available: 95.8 GB

Please specify the size of the outer volume to be created (you will first create the outer volume and then a hidden volume within it). The minimum possible size of a volume within which a hidden volume is intended to be created is 340 KB.

Figure 2: Creating outer volume of encrypted container in VeraCrypt.

Our hidden volume inside of the encrypted container was 26`MB`.


VeraCrypt Volume Creation Wizard

Hidden Volume Size

26 MB

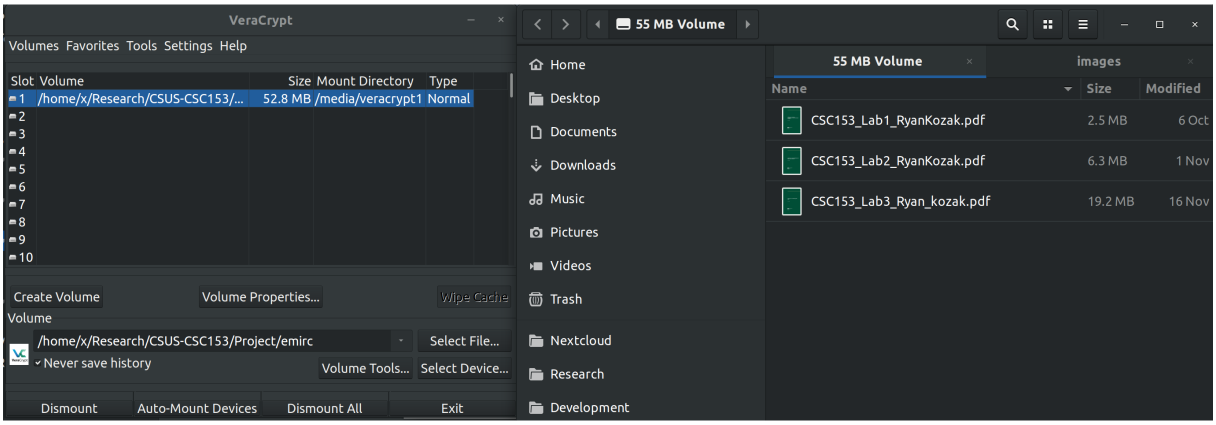
Maximum possible hidden volume size for this volume is 54.2 MB.

Please specify the size of the hidden volume to create. The minimum possible size of a hidden volume is 40 KB (or 3664 KB if it is formatted as NTFS). The maximum possible size you can specify for the hidden volume is displayed above.

Please note that if your operating system does not allocate files from the beginning of the free space, the maximum possible hidden volume size may be much smaller than the

Figure 3: Creating inner (hidden) volume of encrypted container in VeraCrypt.

After the creation of our hidden container, we placed decoy files inside of the outer volume. These consisted of homework pdf's from CSC153. If this encrypted container were to be discovered, the decoy key would be provided and these files would be all that is decrypted.



Slot	Volume	Size	Mount Directory	Type
1	/home/x/Research/CSUS-CSC153/...	52.8 MB	/media/veracrypt1	Normal
2				
3				
4				
5				
6				
7				
8				
9				
10				

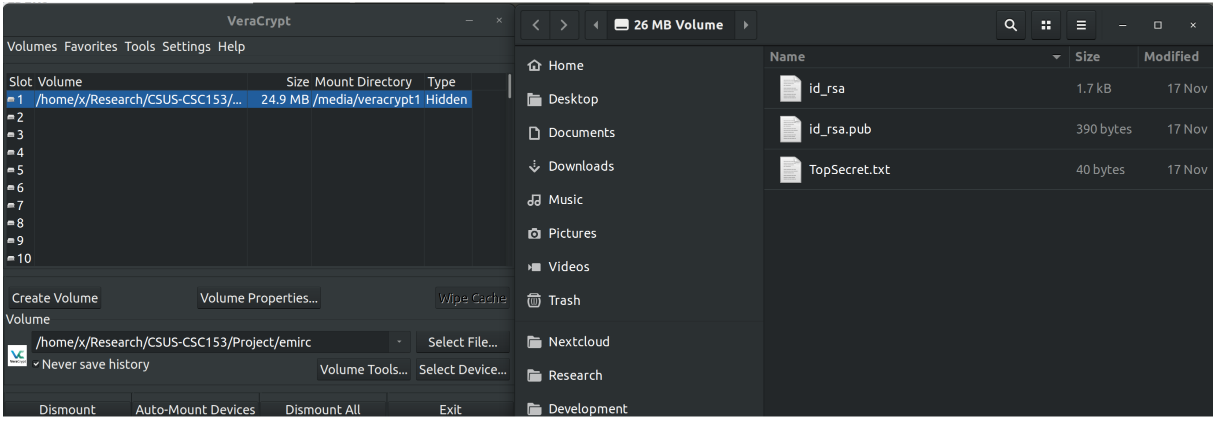
Volume: /home/x/Research/CSUS-CSC153/Project/emirc

✓ Never save history

Name	Size	Modified
CSC153_Lab1_RyanKozak.pdf	2.5 MB	6 Oct
CSC153_Lab2_RyanKozak.pdf	6.3 MB	1 Nov
CSC153_Lab3_Ryan_kozak.pdf	19.2 MB	16 Nov

Figure 4: Decoy pdf files on outer volume of encrypted container.

The true sensitive information was placed on the inner (hidden) volume of the encrypted container, as seen in Figure 5 below.



Slot	Volume	Size	Mount Directory	Type
1	/home/x/Research/CSUS-CSC153/...	24.9 MB	/media/veracrypt1	Hidden
2				
3				
4				
5				
6				
7				
8				
9				
10				

Volume: /home/x/Research/CSUS-CSC153/Project/emirc

✓ Never save history

Name	Size	Modified
id_rsa	1.7 kB	17 Nov
id_rsa.pub	390 bytes	17 Nov
TopSecret.txt	40 bytes	17 Nov

Figure 5: The truly sensitive information is on the inner (hidden) volume of the encrypted container.

Stenography

After the creation of our encrypted container, it was hidden on a 1GB flash drive inside of bad blocks. The goal of this was to avoid the suspicion aroused by the presence of an encrypted file. By hiding the encrypted container we've added another layer of protection, as now the flash drive appears as though it contains only harmless data.

Hiding Data in Bad Blocks

Conceptually hiding our encrypted container inside of bad blocks is simple. We create a small partition in the middle of our drive, large enough to fit the container, and create a file system. The encrypted container is placed on that partition. We then partition the entire drive, but before writing a file system to the outer partition, we tell the file system that the blocks containing our inner partition are bad blocks. This will prevent the outer partition from seeing or using the sectors that contain the inner partition.

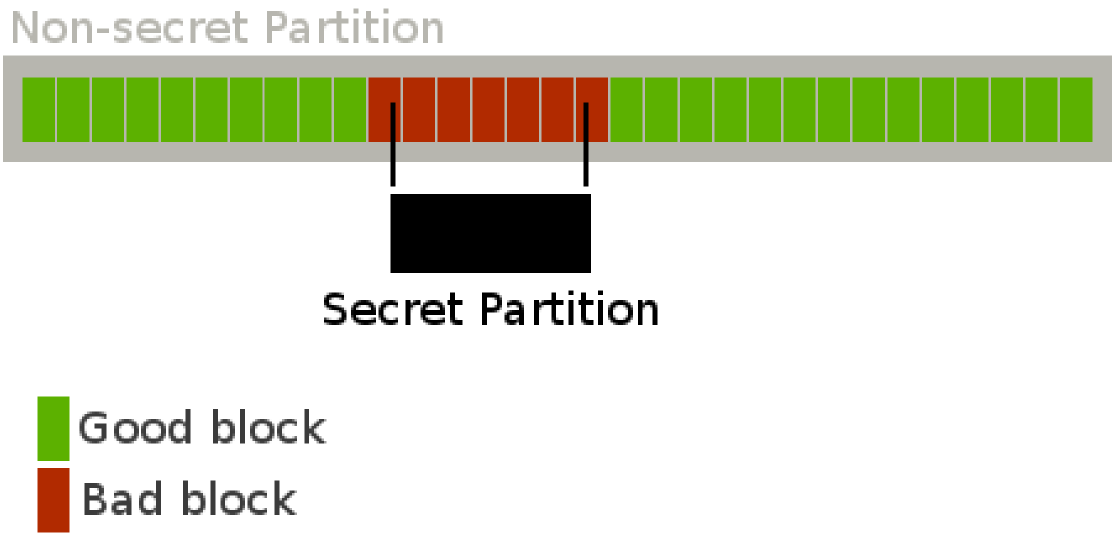


Figure 6: Hiding an inner partition in bad blocks *source*.

The steps taken to accomplish this can be found below, and will assume the drive is `/dev/sdc`.

0. Zero out the drive `sudo dd if=/dev/zero of=/dev/sdc status=progress`.
1. Create our inner partition of `56.32MB`. The starting point is sector `2630`, and it is `110000` sectors, all `512B` each.

```
sudo sfdisk /dev/sdc << EOF
2630,110000,6
EOF
```

2. Make a `FAT 16` file system for the inner partition.

```
sudo mkfs.vfat -F 16 /dev/sdc1
```

3. Place our encrypted container inside this partition.
4. Unmount the drive via `sudo umount/dev/sdc1`.

5. Create an outer partition that takes up all the space on the drive. This will consume the entire inner partition.

```
sudo sfdisk /dev/sdc << EOF
,,6
EOF
```

6. Build a file of bad blocks, to set when creating the file system for the outer partition. In this case we will start at block `288` and go to block `58000`. Each block is `1kb`, so the size of this comes out to be about `57.7KB`. The reason that this is slightly larger than our inner partition is because we're padding the size a bit to account for any potential miscalculations.

```
seq 288 58000 > /tmp/badblocks
```

7. Make a `FAT 16` file system for the outer partition, marking the blocks of the inner partition as bad. This will prevent the outer partition from using these blocks.

```
sudo mkfs.vfat -F 16 -l /tmp/badblocks /dev/sdc1
```

8. Fill the outer partition with harmless data.

Further Information

Switching between partitions to access data is simple. The hidden partition can be accessed again by unmounting the drive and using the command from step 1. To toggle to the outer partition, unmount and enter the command from step 5. Repeat those two processes as necessary.

Analysis of Encrypted Data Hidden in Bad Blocks

We've taken an image of the drive with the outer partition mounted for a forensic analysis.

```
dcfldd if=/dev/sdc1 of=/home/x/Research/CSUS-CSC153/Project/crime.dd
    conv=noerror,sync hash=md5 hashwindow=0 hashlog=/home/x/Research/
    CSUS-CSC153/Project/crime.md5.txt
```

Destroying Data

Data Destruction with dd

See dis ddKillDisk.sh

Video

In the interest of keeping this paper under the 20 page limit, screen shots of every individual step were not included. Our project's video contains a step by step guide to accomplish the creation of a hidden volume, how to hide data in bad blocks, and how to effectively destroy data.

See Our Video **Here**

References

1. Veracrypt Documentation
2. davidverhasselt.com