Secure Deletion Worksheet

The goal of this lab is to familiarize students with some issues related to file deletion.

Task 1: Create an EXT2 "virtual disk"

Create a "virtual disk", aka a file that is formatted like a disk that can be mounted like it was physical media.

1. To create a virtual disk on Linux, one can use the `dd` command, specifying an input file (`if`), an output file (`of`), a block size (`bs`) and the number of blocks. Execute the following `dd` command to create a 1MB virtual disk that contains nothing but zeros (provided by the "zero" device, `/dev/zero`):

```
dd if=/dev/zero of=myfs.img bs=1k count=1k
```

The above command creates a file filled with zeros called `myfs.img` that has 1024, 1024-byte blocks (1024 * 1024 = 1MB). You will be using this file to emulate a very small hard disk.

2. Use the `ls` command to verify the existence of the `myfs.img` file.

3. It is not enough to have a blank hard disk because a useable file system requires data structures to manage the stored files. These data structures (called *inodes* in Unix/Linux) need to be laid out on the disk, which is typically referred to as *formatting the drive*.

   Format the disk image by typing the following command:

   ```
   mkfs.ext2 -F myfs.img
   ```

4. In Unix, file systems can only be mounted on directories, so create a mount point for your file system by creating the `mnt` directory, as shown below:

   ```
   mkdir mnt
   ```

5. Now that you have a mount point, you can mount your virtual disk. We would typically need sudo permissions to use the `mount` command, but we can do a less privileged version with the following:

   ```
   guestmount -a myfs.img -m /dev/sda mnt
   ```

   Your file system is now mounted under the `mnt` directory.

Task 2: Create some Files

Now that you have created your virtual disk and mounted it, you are going to add some files in this "disk" to experiment with file deletion.

1. Create **three** text files of different sizes, using the below commands:

```
echo "First file created" > mnt/file1
echo "Second file created" > mnt/file2
echo "Third file" > mnt/file3
```

2. Use the command '`ll mnt/`' to display the size (in bytes) of the files under the directory `mnt`. The file size is displayed before the Month.

Question #1: ***What are the sizes of the files you created? Fill out the below table with this data.***

File name	File content	File size (bytes)
file1	First file created	19
file2	Second file created	20
file3	Third file	11

3. Unmount the "disk" by doing the following:

```
guestunmount mnt
```

Task 3: Deleting a File on Unix

1. Display the contents of `myfs.img` as raw data in hexadecimal notation, using the command:

```
hexdump -C myfs.img
```

The output of this command is in three columns: the raw data (in hex), an ASCII representation of the same data, and the offset in the disk image where the data is located.

offset	data (hex)	data (ASCII)
00000000	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	|................|
*		
00000400	80 00 00 00 00 04 00 00 33 00 00 00 d7 03 00 00	|........3.......|

Notice: the '`*`' replaces repeated data. For example, in the above output, the data at offsets `00000000`, `00000010`, … `000003f0` are all the same: 16 bytes, where each byte has the value `0x00`. Thus, the output is abbreviated with the asterisk in place of repeated data. Further, this data does not have an ASCII representation, so the third column displays the data as dots. Also in the example, at offset 0x408 is hex value 0x33, which is ASCII value '3'.

2. If we are interested only in the ASCII data in the file, we can extract this using the `strings` command, to display the ASCII data and the offset of that data in `myfs.img`:

```
strings –tx myfs.img
```

Question #2: ***Fill out the below table with any output the `strings` command provides. (Some of the table is filled out for you.)***

Offset	String
6034	file1
6044	file2
6054	file3
9800	First file created
9c00	Second file created
a000	Third file

3. Re-mount your file system:

   ```
   guestmount -a myfs.img -m /dev/sda mnt
   ```

4. Use the below command to delete `file2`:

```
rm mnt/file2
```

5. Use the '`ls mnt`' command to verify that the deleted file is no longer present.

6. Once again, unmount your file system:

   ```
   guestunmount mnt
   ```

7. Again, display all the ASCII text in the "disk" by entering the following:

```
strings –tx myfs.img
```

Question #3: ***Fill out the below table with the data found in the disk image, with the offset of each piece of data.***

Offset	String
6020	Lost+found
6034	File1
6044	File2
6054	File3

9800	First file created
9c00	Second file created
a000	Third file

Question #4: ***What do you observe? Is this what you expected to see? What security problems, if any, are implied by your observations?***

It's exactly the same as the last table before the deletion. I did not expect this because I thought deleting file2 would removes its strings as well. This is a problem because it looks like file2 could still be accessed after removing it.

Task 4: Undeleting a File on Unix

In this task you will attempt to undelete the file you deleted earlier. In Unix this can be a tricky and difficult task, reserved for knowledgeable system administrators. Even then, when attempting to delete a file there should not be anyone on the system creating new files, or the data may be lost permanently anyway.

Later, we will undelete files using tools that know how to interpret the file system layout. These are especially useful as the file systems grow large and when the formats are complex. For this task, however, we will undelete files "manually," looking at the raw bytes of the drive and without the assistance of any special file recovery tools.

1. Find the decimal representation of the location on the disk for the beginning of the deleted file `file2`, using the following command:

```
strings –td myfs.img | grep "Second file created"
```

Notice, this should simply be the decimal representation of the same hexadecimal offset you recorded in the table for Question #2.

Question #5: ***What is the decimal offset into the disk, to the point where*** `file2` ***begins?***

39936

2. Use the `dd` command (shown below) to copy the data from the location on the disk to a new file (`rfile2`) that will hold the recovered data. Below, replace SKIPNUMBER with the offset of the beginning of `file2` (the number you recorded in Question 5) and replace FILESIZE with the size of `file2` (from the table in Question 1):

```
dd if=myfs.img bs=1 skip=SKIPNUMBER count=FILESIZE of=rfile2
```

This command yanked the data out of the virtual disk into another file, called rfile2. This is a very simply type of file carving.

3. Display the contents of the file you just recovered:

```
cat rfile2
```

Question #6: ***Were you able to restore the file?***

Yes, in rfile2 it shows "Second file created"

Task 5: Securely Deleting a File on Unix

For some, it is comforting to know that it may be possible to undelete data. For others, it is frightening to know that something that was deleted may still be there. For the latter group, this task will show one way to securely delete a file on Ubuntu. You will be using a command called `shred`, which may not be installed on all Linux distributions; although, most operating systems give you *some* utility or operation that will allow you to securely delete files.

1. Use the `strings` command to verify the existence of `file3` and its data:

```
strings myfs.img
```

2. Re-mount your file system:

```
guestmount -a myfs.img -m /dev/sda mnt
```

3. View the files, securely delete file3, and confirm the deletion using the commands:

```
ls mnt
shred –uxz mnt/file3
ls mnt
```

4. Once again, unmount your file system:

```
guestunmount mnt
```

5. Repeat the use of the `strings` command:

```
strings –tx myfs.img
```

Question #7: ***What do you observe? Fill out the below table with the output of the strings command.***

Offset	String
6020	Lost+found
6034	File1
6044	0000
6054	00003
9800	First file created
9c00	Second file created

Task 6: NTFS Virtual Disk

NTFS is the file system used by versions of Windows. Because of the way NTFS manages files, it is much easier to undelete them, as long as new files have not erased the details of the metadata or the deleted data on the disk.

1. Create a new virtual disk:

```
dd if=/dev/zero bs=1024 count=2048 of=ntfs.img
```

2. Format the virtual disk to have an NTFS file system:

```
mkntfs -F ntfs.img
```

3. Mount the virtual disk:

```
guestmount -a ntfs.img -m /dev/sda mnt
```

4. Create file1, file2, file3 just like in Task 2, and fill out the below table:

File name	File content	File size (bytes)
file1	First file created	19
file2	Second file created	20
file3	Third file	11

5. Delete file1 and securely delete file3, using the commands:

```
rm mnt/file1
shred –uxz mnt/file3
```

6. Unmount the “disk” by doing the following:

```
guestunmount mnt
```

7. Verify that the data still exists on the virtual disk by entering the following command:

```
strings –tx ntfs.img | grep file
```

Question #8: ***What do you observe? Fill out the below table with the output of the above command.***

Offset	String
14168	First **file** created
14568	Second **file** created

8. Use the `ntfsundelete` command to find information about deleted files, as shown below:

```
ntfsundelete –p 100 ntfs.img
```

Question #9: ***Which inode number was associated with file1? Which inode number was associated with file3?***

File 1 Inode = 64
File 3 Inode = 66

9. We will undelete file1, using the following command (replacing INODE with the number of the recoverable file):

```
ntfsundelete --undelete --inodes INODE --output rfile1 ntfs.img
```

10. Use `ll` to list the contents of the current directory. You should see the deleted file. Once again, this utility yanks the file out of the file system.

11. Use the `cat` command to display the content of `rfile1`.

Question #10*: Try to recover file3 and compare with the results from your attempt to recover file1.*

When recovering file 1 it recovers it fine and has the contents “First file created” but when trying to recover file 3 you need to paste to a different rfile, and when you cat that given rfile the contents will only contain “0” so it looks like shred destroys the data, but the memory location still exists nonetheless.

Question #10*: Go back to some steps in this lab and experiment. Try to do something differently, or otherwise explore. Explain what you did here:*

I tried to hexdump the ntfs.img file and it insists on being mounted on to something.