File Carving

Clark Otte

February 25, 2022

CSC 5322 – Defensive Coding and Security

**Problem Description**

The goal of the project is to extract each instance of a jpg or pdf file from within a drive image. In order to do this, the bits of the drive image must be read and searched for instances of the headers and footers of each respective filetype. Then the bits must be written to a new file. The user must be able to specify the name of the image file and the type of file they want to search for using command line arguments.

**Problem Approach**

The approach taken to solve this problem was to first create the arguments and parse through them in order to get the information that was input from the user. Then using the information on the type of file they were looking for; the correct headers and footers were loaded into the program and the filename the user gave was loaded as well. Then the data from the file was searched through finding each instance of a header and footer and recording its bit position. Then the data between each starting bit and ending bit were written to a new file.

**Problem Approach for Carving Fragmented Files**

In order to carve files that are fragmented and not continuous, there would need to be more information found on the file. You would need to know the size of the file and would need to have some sort of knowledge on where to look for the next fragment of the file. Each fragment would need to have some sort of information like a header and footer to tell where each portion of the file starts and ends.

**False Positives**

The program did come across some false positives. There were some files that were not the right type of data and could not be opened as a normal jpg or pdf. There could be many reasons for these false positives. The headers and footers work off of the information in the bits of the image drive. If there was some data that had some bits that were similar or the same as the headers and footers of the file types we were searching for, that could lead to data being written that didn’t correspond to the same file type we were looking for.

**12 Byte vs 3 Byte headers**

If there were 12-byte headers instead of 3-byte headers in file carving, it would lead an increase in the number of bytes the program needs to read in order to fully search an image. There also might be less false positives in file carving due to the probability of 12 random bytes matching the 12 bytes of a header being less likely than 3 random bytes matching the 3 bytes of a header.