A close up of a sign

Description automatically generated

Project 2 Report

Team Member Name(s)

Arju Reza

Date

11/5/2022

# Executive Summary

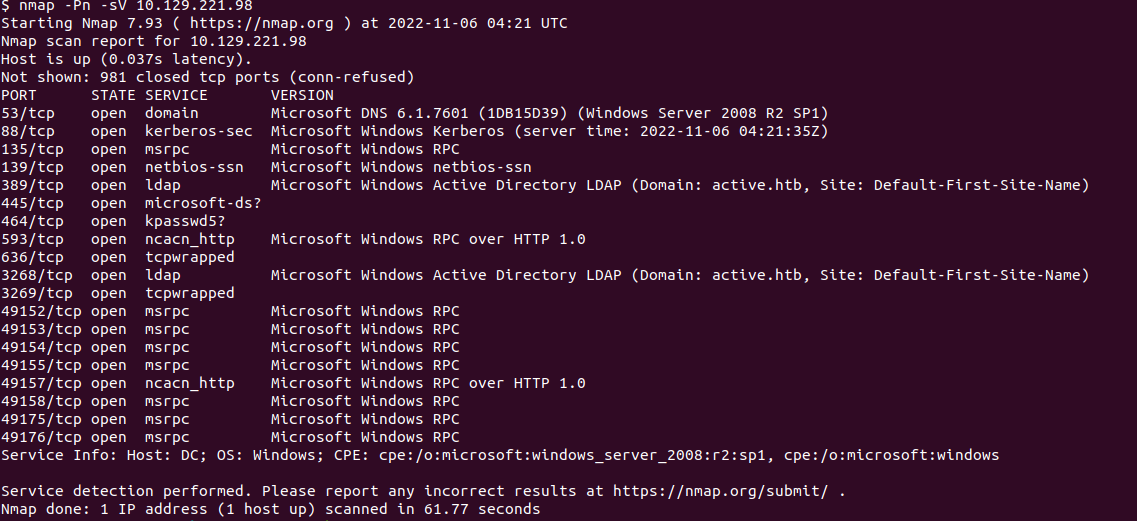
## Part 1

For this project, I was given a disk image file named ‘Project2.dd’. I was assigned the task of recovering all of the files present on the disk image file like in Project 1. For this project, however, I was not going to manual go through the hex dump of the .dd file, but instead, I was going to use knowledge of file signatures to find file details like file beginning, ending, and size. With the necessary information, I then ran the dd command in the SIFT workspace in Linux to recover all of the files.

In this project, there were many things to consider, especially since it is still dealing with parsing through the hex dump of the given .dd file. First, I needed to research the file signatures, specifically the header and footer information (if it existed). In some cases, I also needed parse into the file signatures to figure out details like file size. I also had to deal with problems related to false positives and multiple files of the same type being next to each other. The disk image ended up containing 13 files.

## Part 2

For the second part of this project, I was tasked with getting network information from the Active lab in HTB.



Domain service Microsoft DNS 6.1.7601 (Port 53): Domain service is used to store centralized directory information and allows for communication between users and domains

Kerberos Protocol (Port 88): Computer-network authentication protocol that uses tickets that allows for communication between nodes over a network which allows for identity exchange securely

Microsoft Remote Procedure Call Protocol (Port 135, 49152-49158, 49175-49176): Uses the client-server model to enable a program request service from another computer’s program without having to need that computer’s network details.

NetBIOS SSN (Port 139): Provides session layer services in the OSI model that allows for applications on separate computers to communicate on a local area network.

Light Directory Access Protocol (Port 389, 3268): Application protocol that is used for accessing and maintaining distributed directory information over an IP network.

ncacn\_http (Port 593): Identifies the Microsoft Internet Information Server which allows for client-server communication over applications.

Transmission Control Protocol: Allows applications and computing devices to exchange messages over a network

# Problem description

## Part 1

Given a disk image by the name of “Project2.dd”, use file signature information to gather enough information to successfully recover all the files that exist on the disk image.

## Part 2

Conduct network and application scans of a network host to identify port number and protocol information of the Active lab in HTB. Using network forensics techniques, collect information about a network host and describe how each protocol works.

# Description of analysis techniques utilized

## File Attributes

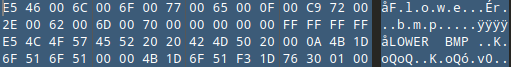
Table 1: File Headers

|  |  |
| --- | --- |
| **File Type** | **Header** |
| mpg | \x00\x00\x01\xB3\x14 |
| pdf | \x25\x50\x44\x46 |
| bmp | \x42\x4D\x76\x30\x01 |
| gif | \x47\x49\x46\x38\x39\x61 |
| jpg | \xFF\xD8\xFF\xE0 |
| docx | \x50\x4B\x03\x04\x14\x00\x06\x00 |
| avi | \x52\x49\x46\x46 |
| png | \x89\x50\x4E\x47\x0D\x0A\x1A\x0A |

Table 2: File Footers

|  |  |
| --- | --- |
| **File Type** | **Footer** |
| mpg | \x00\x00\x01\xB7 |
| pdf | \x25\x50\x44\x46 |
| gif | \x00\x3B |
| jpg | \xFF\xD9 |
| docx | \x50\x4B\x05\x06 |
| png | \x49\x45\x4E\x44\xAE\x42\x60\x82 |

As seen through the tables, all of the file types have a header and footer except for .bmp and .avi file types. The main tool used to give out all of this information was found in <https://www.garykessler.net/library/file_sigs.html>. The only file signature that did not come exactly from this table is the file header for file type bmp. Since this file type does not have a file footer and the file header is only 2 bytes long, there are many instances that exists throughout the disk image. However, bytes 2-5 contain the file size in little-endian order. So, I went into the hex for Project2.dd and found the file size using 101 Editor in the SIFT workspace to get the file size for the .bmp file.



## Methodology

The main method used to recover the files was iterating through all of the file types and locating the file(s) of that type that exists in the disk image using the file signatures. The main method first located the file header and recognized that as the start offset of the file. Next, the main method files the file footer and adds footer length to that to figure out the ending offset. Finally, the size of the file is figured out by subtracting the location of the ending offset from the start offset.

The first exception to this main methodology was for .pdf files. When I first initially used my Python script to recover the files, the first of two pdf files that were in the disk image ended up being corrupted. I assumed that this error was being caused because of interference from the second pdf corrupting the file recovery of the first pdf. To work around this, I separated figured out where the second pdf started and separated them so that I could focus only on the hex dump of the first pdf when recovering it.

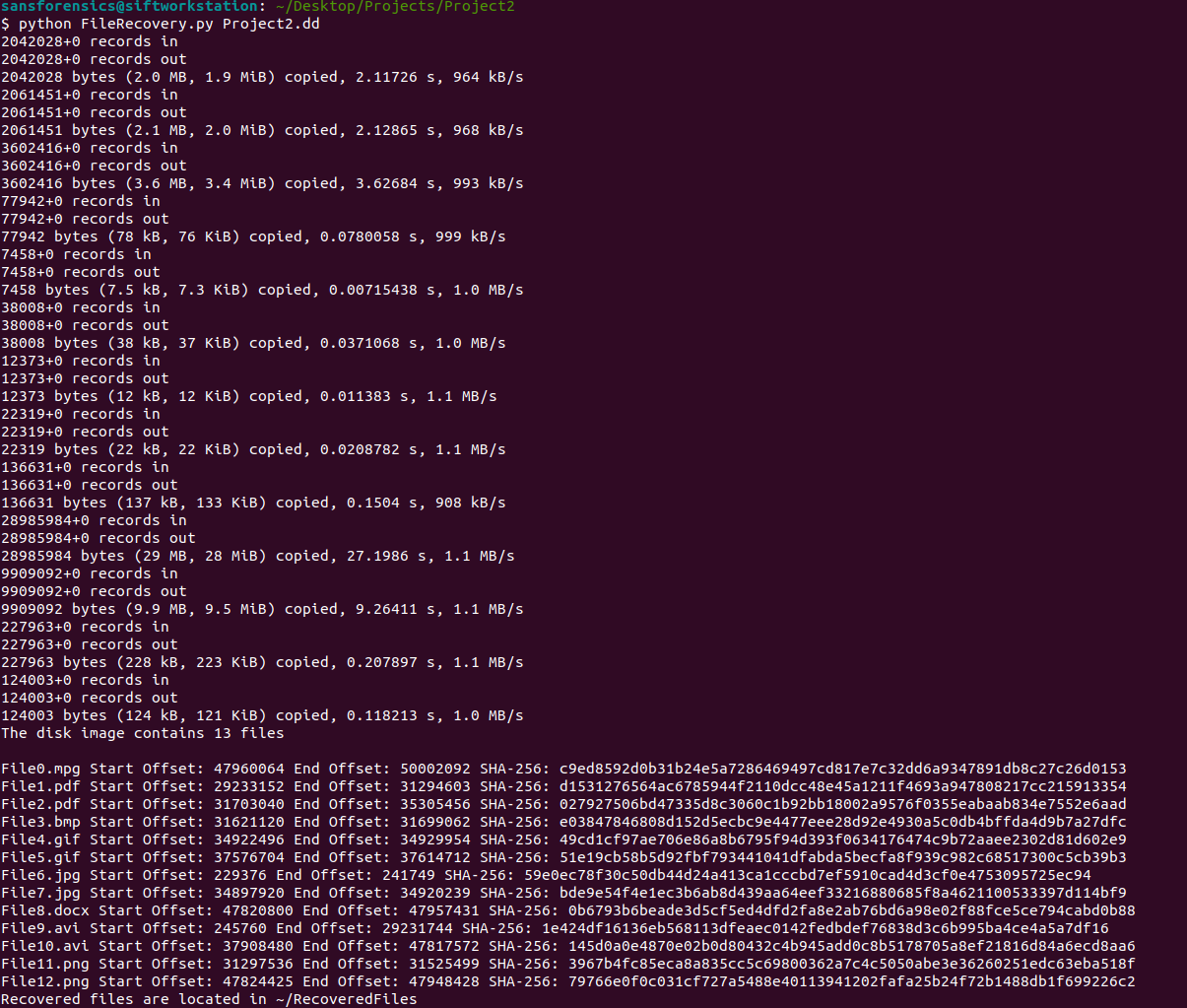
The second exception is for .bmp files. The description applies from the ‘File Signature’ section. Additionally, I figured out from the file description for bmp files that bytes 6-9 of the file header are reserved so I checked in my script to make sure that bytes 6-9 has all zeros.

The third exception is .avi files. For avi files, I found the size using the knowledge that bytes 4-7 contain the file size in little-endian order.

The last exception is for .docx files. There are 18 additional bytes after the file footer, so I added 18 to the ending offset after normal calculations.

After gathering necessary information, I organized all the information needed into a two-dimensional tuple. I then used this tuple to construct the dd command that would recover each file and then finished by output all the necessary information about the files including the SHA256 hash.

Final results:



# Conclusions and Recommendations

The given disk image ‘Project2.dd’ contains 13 files that are located in the RecoveredFiles directory.

Network analysis on Active lab in HTB complete.