College of Computer Studies – Department of Computer Technology

NSSECU3 S12

Windows Forensics Practical: AppCompatCacheParser and AmcacheParser (Timeline Analysis)

Created by:

Jose Mari L. Del Castillo

Francisco Emmanuel T. Dumas

7/12/2024 – 0.3

I. Change Control

Version	Date	Comments
0.1	7/2/2024	Initial draft
0.2	7/3/2024	Second draft
0.3	7/12/2024	Final

Table of Contents

I. Change Control....2
II. Objectives....4
III. Abstract....5
IV. Introduction....5
V. Related Literature and Works....6
Evidence of Execution (AmcacheParser)....6
Amcache's Historical Data....6
Identifying Application compatibility (AppCompatCacheParser)....7
AppCompatCache's Historical Data....7
VI. Project Section....8
A. Code Flowchart....8
B. Summary of Project and Tool Execution Guide....8
C. Timeline, File Information, and File Combining....16
Timestamp in UTC+0 Format Column....16
File Destination Column....17
Time zone Adjustment....17
Merged Columns....18
D. Detailed Timeline Analysis....20
ShimCache timestamps....20
Amcache timestamps....24
E. Approval of Tools....28
VII. Conclusion....28
VIII. Appendix....29

A. ShimAmCacheParser.py 29
IX. References 32

Figure 1: Flowchart of the ShimAmCacheParser 8
Figure 2: GitHub.io download link for AmcacheParser 9
Figure 3: GitHub.io download link for AppCompatCacheParser 9
Figure 4: Needed tools located in the same directory 9
Figure 5: Command line with Admin Privileges 9
Figure 6: Change the directory to the folder with the tools and run ShimAmCacheParser .. 10
Figure 7: Error message when a tool is missing from the same directory as ShimAmCacheParser 10
Figure 8. Error message when the Amcache.hve is missing from the same directory as ShimAmCacheParser 10
Figure 9: Choosing to exit the tool 11
Figure 10: Error message when entering an Invalid Option 11
Figure 11: Running one tool individually and choosing to use the default directories 11
Figure 12: ShimAmCacheParser successfully using AmcacheParser to parse Amcache.hve 12
Figure 13: Output being saved into the default output directory 12
Figure 14: Contents of one of the CSV files from AmcacheParser 13
Figure 15: ShimAmCacheParser successfully combining all CSV files from AmcacheParser into the CSV file, amcache_combined_output 13
Figure 16: Running both tools simultaneously and choosing to set the directories manually 14
Figure 17: AppCompatCacheParser successfully running 14
Figure 18: AmcacheParser successfully running 14
Figure 19: Outputs from both tools successfully being printed and combined to the output directory 15
Figure 20: Output saved in the directory that was set manually 15
Figure 21: ShimAmCacheParser successfully combining all CSV files from both tools into the CSV file, both_combined_output 16
Figure 22. Code to convert CSV files to dataframes 16
Figure 23.Code to insert three columns of SourceFile, SourceDirectory, and Timestamp UTC+0 in every CSV file. 17
Figure 24. Code to get the file name and file destination 17
Figure 25. Combining and sorting of CSV files. 18
Figure 26. DriverName column of Amcache_DevicePnps 18

Figure 27:DriverName column of Amcache_DriveBinaries 19
Figure 28: Contents of both_combined_output.csv 19
Figure 29. Using RowCountChecker.py to check if there are any row duplicates 19
Figure 30: Extracting the exe file, ImHere, from the ZIP file 20
Figure 31: Checking the contents of the folder of ImHere without opening it in file explorer 20
Figure 32: ImHere.exe is not in the ShimCache after file extraction 21
Figure 33: Executing ImHere.exe for the first time 21
Figure 34: ImHere.exe not listed in the ShimCache, even after execution 22
Figure 35: Restarting the system after executing ImHere.exe 22
Figure 36: ImHere.exe is now inside the ShimCache 23
Figure 37: File properties of ImHere.exe 23
Figure 38: Extracting the exe file, ImHereAgain, from the ZIP file 24
Figure 39: Checking the contents of the folder of ImHereAgain without opening it in file explorer 25
Figure 40:ImHereAgain.exe is not in the Amcache after file extraction 25
Figure 41: Running Microsoft Compatibility Appraiser for the first time 26
Figure 42: ImHereAgain.exe is not in the Amcache after running Microsoft Compatibility appraiser 26
Figure 43: Running ImHereAgain.exe for the first time 27
Figure 44: Running Microsoft Compatibility Appraiser after running ImHereAgain.exe 27
Figure 45: ImHereAgain.exe is located inside the Amcache 27
Figure 46: File properties of ImHereAgain.exe 28
Figure 47: Proof of approved tools 28

II. Objectives

The objective of this Windows Forensics Practical is to integrate two (2) digital forensics tools into a two-in-one solution that can run both tools simultaneously and combine their outputs for an organized timeline with regards to their respective timestamps and conduct a detailed timeline analysis of their outputs, while achieving the following goals:

- Creating a tool that can run the tools either individually or simultaneously.
- Enhancing the tool execution process to be smoother and faster compared to running both tools individually in their respective folders or directories.
- Developing an interactive command-line interface with error handling that makes both tools easier to use.

- Producing an output that combines all generated CSV files of one of the tools or both tools into a single, consolidated file.
- Three additional columns will be generated for the output that can be used in creating a timeline:
    - File name
    - File destination
    - Timestamp in UTC+0 format.
- Fully analyzing and interpreting the timeline created by the combined digital forensics tools.
- Identifying the difference and similarities between the combined tools.
- Displaying what the group has learned throughout the course of NSSECU3.
- Overall, making the results easier to read and interpret for digital forensics.

III. Abstract

The group identified two digital forensics tools, AppCompatCacheParser and AmcacheParser by Eric Zimmerman, which were approved by the instructor. Using Python and various libraries, they aim to develop a new tool that enhances usability and execution speed. The new tool will be called ShimAmCacheParser and will introduce an interactive command line interface and produce the same results as the individual tools but will also combine all CSV output files of one or both tools into a single, consolidated file with three additional columns for file name, file destination, and timestamp in UTC+0 format for an organized timeline. This integrated tool will make the overall process easier to use and faster to execute. The group will also provide a detailed analysis of the generated timeline created by the tool. Additionally, we will give a background on AppCompatCacheParser and AmcacheParser, as well as their differences and similarities.

IV. Introduction

The paper focuses on combining two digital forensics tools: AmcacheParser and AppCompatCacheParser. AppCompatCacheParser is a command-line tool used for examining the ShellCache files called AppCompatCache in Windows [9]. Similarly, AmcacheParser is a command-line tool that parses Amcache.hve files [1]. Using Python and the subprocess library, our group will develop a tool that maximizes the efficiency of both tools by simplifying the process for the user. Instead of manually inputting the full command for each tool individually, the integrated tool will automatically set the parameters to the usual destinations. Additionally, the user will have the option as well to decide whether they wish to run each tool individually or run both at the same time. This allows the end user to simply start the tool and, if needed, change the input and

output destinations through an interactive command line interface. Additionally, we will use the pandas library to interact with the output of the tools, which are CSV files. With pandas, we can combine all outputs of one or both into a single CSV file and generate three additional columns for file name, file destination, and timestamp in UTC+0 format that can be used when creating a timeline. After developing the tool, the group will offer a comprehensive analysis of the timeline produced by ShimAmCacheParser In addition, we will provide an overview and comparison of AppCompatCacheParser and AmcacheParser, highlighting both their differences and similarities.

V. Related Literature and Works

Evidence of Execution (AmcacheParser)

Amcache Parser is a tool created by Eric Zimmerman and his team. The primary use of this tool is to parse the registry file, Amcache.hve. The tool can provide a detailed timeline of different executed programs, answering when and where the program was last executed. Additionally, it provides details such as the file path and the version of the program, giving us a wealth of information about what files have been executed on our system [2].

The Amcache.hve file was first introduced in Windows 8 as a replacement for RecentFilesCache.bcf on pre-Windows 8 machines. Amcache.hve is a small registry file that contains information about recently run applications and programs [3]. The file is in "<DRIVE>\Windows\AppCompat\Programs\Amcache.hve," and within this small hive is a folder called "Root" that contains four keys/folders. Out of these four, the "File" folder is the most important one for digital forensics. In this folder, files are grouped by their volume GUIDs, and under each GUID are several values holding details about those files, written in hexadecimal. In digital forensics, the Amcache hive can provide evidence for identifying malicious program executions by showing what programs were executed before an incident is detected [2, 3, 4].

Amcache's Historical Data

As for Amcache, to recap, the hive provides us with a timeline of executed programs, proving that a file was present or existed on a system. This is a great tool for digital forensics. However, the main purpose of Amcache for a Windows machine is to help maintain compatibility with various applications [7]. The information is collected and placed into the Amcache hive when the operating system determines if an application is compatible. An automated scheduled task is executed to gather information about files found within the Program Files, Program Files (x86), and Desktop directories, resulting in multiple records being recorded in the Amcache hive.

In rare cases, the timestamps may reflect when the file was scanned by this task rather than when the file was executed. Thus, a file may have never been executed but still maintained presence on the system and was scanned by this task. This makes the Amcache hive not an absolute way of confirming that an application was executed and at what time it was executed [7]. Additionally, the timestamps are by default printed in UTC time in the CSV outputs [8].

Identifying Application compatibility (AppCompatCacheParser)

AppCompatCache Parser is another tool created by Eric Zimmerman and his team. Its main use is to parse the AppCompatCache, also known as the ShimCache. ShimCache is a component of the application compatibility database used by the Windows operating system to identify application compatibility issues. This cache contains data related to the Windows feature that helps identify any application compatibility issues. With this, users can troubleshoot legacy functions or quickly search for any modules that require shimming for compatibility. The cache can be found in the directory "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatCache" [9, 10].

So, what is shimming? Shimming is a technique used by the Windows operating system to apply compatibility fixes, or "shims," to applications. A shim is a small library that transparently handles the application's interactions to provide support for older APIs in a newer environment or vice versa, allowing backward and forward compatibility for applications [9]. Another feature of the AppCompatCache Parser is that the command does not need the directory of the cache, as it automatically captures the live registry if no directory is given [11].

AppCompatCache's Historical Data

To summarize, the main purpose of AppCompatCache, or ShimCache, is to identify application compatibility issues. The timestamps in the cache can be useful as evidence of time stomping, but on Windows 10 and above, the cache is not reliable for determining program execution. This is because the timestamps are linked to when the file was last modified, not when it was executed. For example, if we see a ShimCache timestamp from 2015, it does not necessarily mean the file has not been accessed recently, as these are "last modified" timestamps [12].

VI. Project Section

A. Code Flowchart

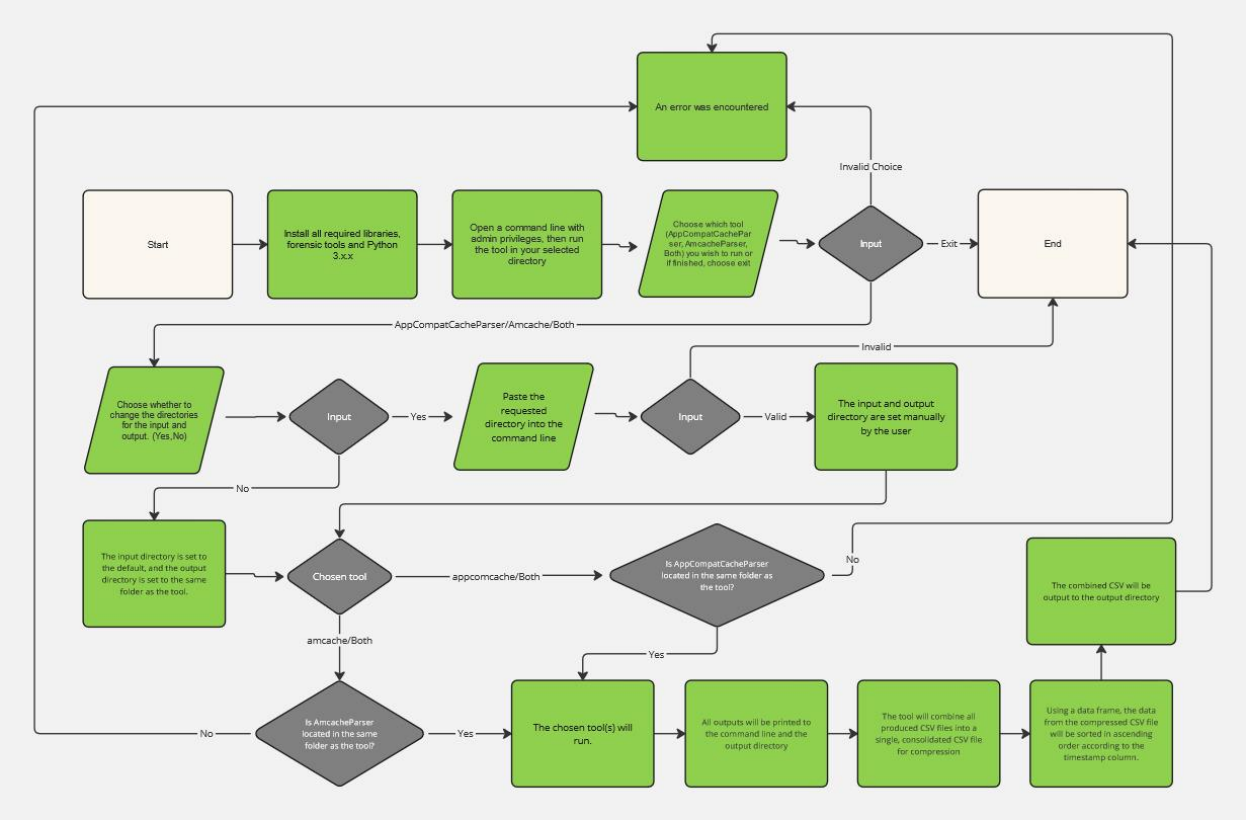

Figure 1: Flowchart of the ShimAmCacheParser

B. Summary of Project and Tool Execution Guide

The project was successfully made through Python and from functions located within two libraries, these libraries were *subprocess* and *pandas*. As seen in Figure 2, and Figure 3, the tools that the project combined were downloaded from Eric Zimmerman's Tools website located in GitHub.io [5] and are downloaded and extracted in the same folder as our created tool ShimAmCacheParser as seen in Figure 4. Note that .NET Framework 4 or 6 should be installed to make the respective tools work with regards to their respective versions as seen in Figure 2. Additionally, the outputs were added with three new columns for the file name, file destination, and timestamp in UTC+0 format.

Figure 2: GitHub.io download link for AmcacheParser

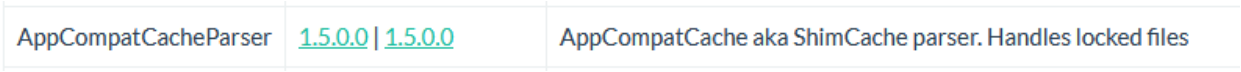

Figure 3: GitHub.io download link for AppCompatCacheParser

As seen in Figure 4 and Figure 5, before using ShimAmCacheParser, we first need to ensure that both AppCompatCacheParser (ShimCache) and AmcacheParser are extracted in the same folder and that the command line is run with admin privileges. To run the tool, first change to the same directory as ShimAmCacheParser just like in Figure 6.

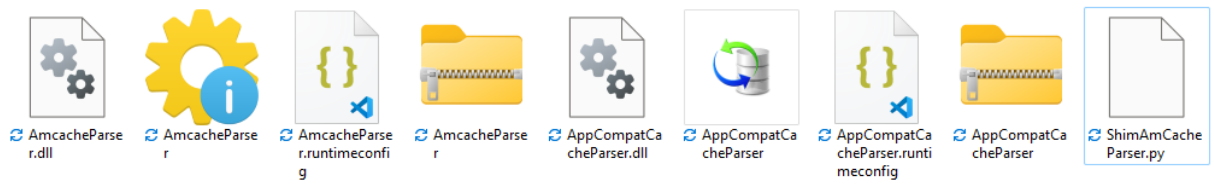

Figure 4: Needed tools located in the same directory

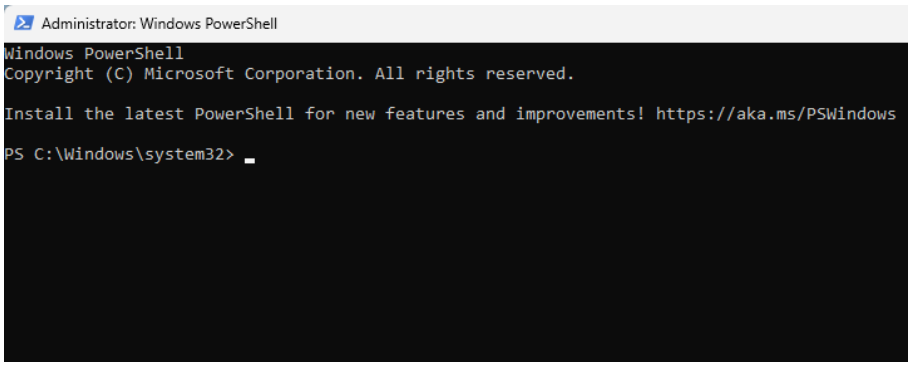

Figure 5: Command line with Admin Privileges

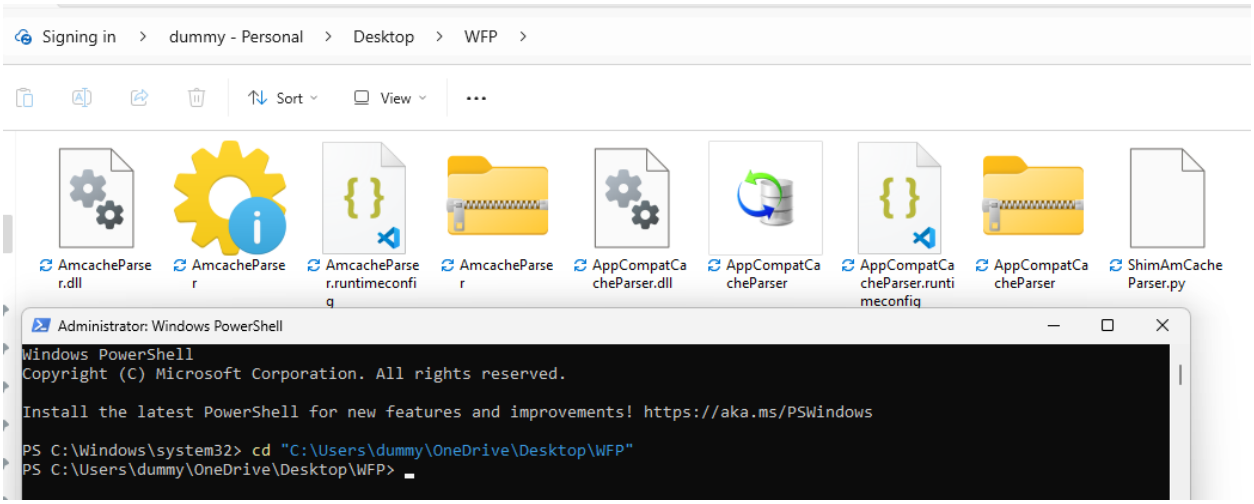

Figure 6: Change the directory to the folder with the tools and run ShimAmCacheParser

Different messages will be displayed to ensure that the user will understand the output of their input. In Figure 7, an error message will be displayed if the user failed to place the AppCompatCacheParser (ShimCache) and/or AmcacheParser in the right directory. In Figure 8, an error message will be displayed if the user failed to give the right path to the SYSTEM hive or Amcache.hve files for the AppCompatCacheParser and/or AmcacheParser tool to work, respectively. In Figure 9, we can see by typing *exit* the user can close the tool, and lastly in Figure 10, an error message will appear if the user chooses an option that is not listed.

```
PS C:\Windows\system32> cd "C:\Users\dummy\OneDrive\Desktop\WFP"
PS C:\Users\dummy\OneDrive\Desktop\WFP> python ShimAmCacheParser.py
Welcome!
Which tool do you want to use? (appcomcache/amcache/both/exit): appcomcache
You have chosen to use the appcomcache tool.
Do you want to edit the default directories? (yes/no): no
C:\Users\dummy\OneDrive\Desktop\WFP\AppCompatCacheParser.exe IS NOT FOUND. Exiting...
No CSV files generated.
CSV files have been saved to C:\Users\dummy\OneDrive\Desktop\WFP
Which tool do you want to use? (appcomcache/amcache/both/exit):
```

Figure 7: Error message when a tool is missing from the same directory as ShimAmCacheParser

```
PS C:\Users\dummy\OneDrive\Desktop\WFP> python ShimAmCacheParser.py
Welcome!
Which tool do you want to use? (appcomcache/amcache/both/exit): amcache
You have chosen to use the amcache tool.
Do you want to edit the default directories? (yes/no): yes
Enter the path to the Amcache database file (example: C:\Windows\appcompat\Programs\Amcache.hve): C:\Users\dummy\OneDrive\Desktop\WFP
Enter the output directory for CSV files: C:\Users\dummy\OneDrive\Desktop\WFP
ERROR: The database file 'C:\Users\dummy\OneDrive\Desktop\WFP' does not exist. Exiting...
No CSV files generated.
CSV files have been saved to C:\Users\dummy\OneDrive\Desktop\WFP
Which tool do you want to use? (appcomcache/amcache/both/exit):
```

Figure 8. Error message when the Amcache.hve is missing from the same directory as ShimAmCacheParser

```
Which tool do you want to use? (appcomcache/amcache/both/exit): exit
Exiting...
PS C:\Users\dummy\OneDrive\Desktop\WFP>
```

Figure 9: Choosing to exit the tool

```
PS C:\Users\dummy\OneDrive\Desktop\WFP> python ShimAmCacheParser.py
Welcome!
Which tool do you want to use? (appcomcache/amcache/both/exit): appcomcaches
Invalid choice.
Which tool do you want to use? (appcomcache/amcache/both/exit): amcaches
Invalid choice.
Which tool do you want to use? (appcomcache/amcache/both/exit): boths
Invalid choice.
Which tool do you want to use? (appcomcache/amcache/both/exit):
```

Figure 10: Error message when entering an Invalid Option

The user can decide to run one tool individually by typing what tool they wish to use. Additionally, the option to change the directories is available but in Figure 11 the user decides to type no, which sets the input directories to the default location of Amcache.hve for AmcacheParser, while in the case for AppCompatCacheParser it should always be the directory of the live SYSTEM hive, and the output directories will be the in same directory of where ShimAmCacheParser is located.

```
PS C:\Users\dummy\OneDrive\Desktop\WFP> python ShimAmCacheParser.py
Welcome!
Which tool do you want to use? (appcomcache/amcache/both/exit): amcache
You have chosen to use the amcache tool.
Do you want to edit the default directories? (yes/no): no
AmcacheParser version 1.5.1.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/AmcacheParser

Command line: -f C:\Windows\appcompat\Programs\Amcache.hve --csv C:\Users\dummy\OneDrive\Desktop\WFP --dt yyyy-MM-dd HH:mm:sszzz -i
```

Figure 11: Running one tool individually and choosing to use the default directories

As seen in Figure 12 and Figure 13, the default directories are being used to obtain the SYSTEM hive file and place the generated CSV files.

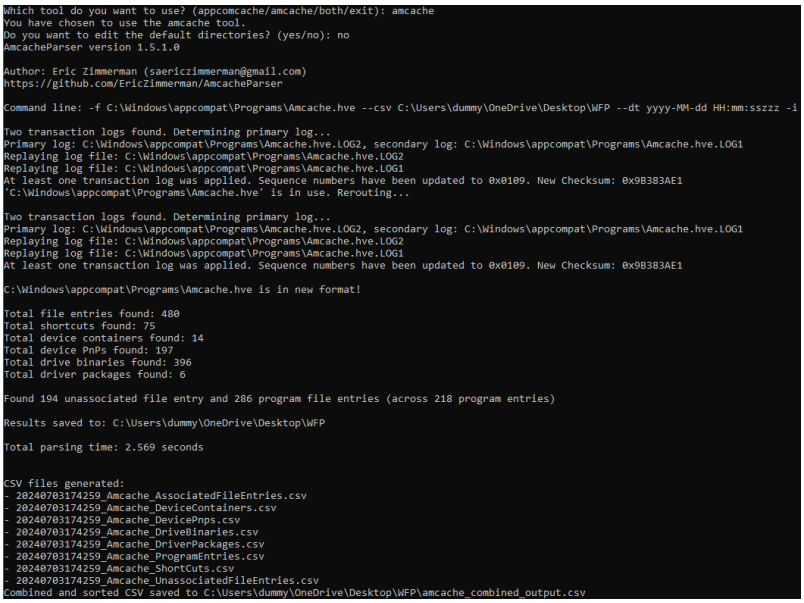

Figure 12: ShimAmCacheParser successfully using AmcacheParser to parse Amcache.hve

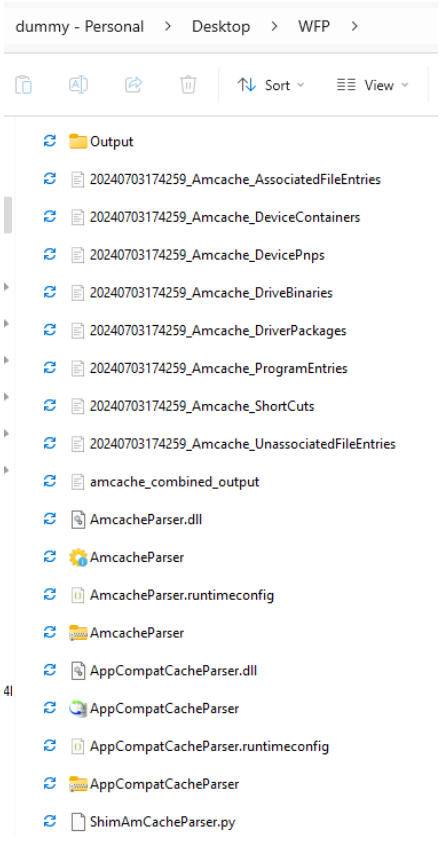

Figure 13: Output being saved into the default output directory

With CSViewer, we can view the generated CSV files. In Figure 14, we can see the DriveBinaries CSV file that was generated with three new columns that indicate the file name, file destination, and timestamp in UTC+0 format. Additionally, since multiple files are

created by AmCacheParser, ShimAmCacheParser combines all these files into a single CSV file called amcache_combined_output, which contains all the data from the generated CSV files as seen in Figure 15. This adding of columns and combining also works for AppCompatCacheParser also with ShimAmCacheParser but it would only add the three new columns. However, for every execution of tools using ShimAmCacheParser, the generated CSV files must be transferred somewhere else as the combined output will include the previous CSV files generated which can cause duplicate rows. Note that the three new columns added in each of the generated CSV files will make it seamless in combining the CSV files into one file and will be useful as the file destination is included in all rows so the CSV file on where the respective row came from can be determined.

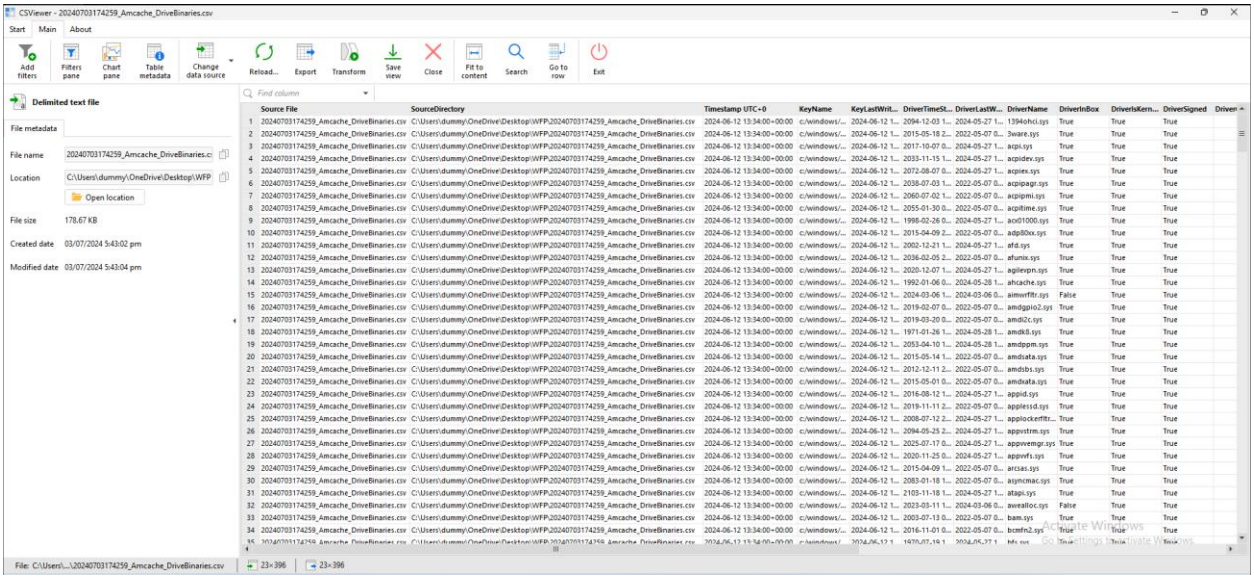

Figure 14: Contents of one of the CSV files from AmcacheParser

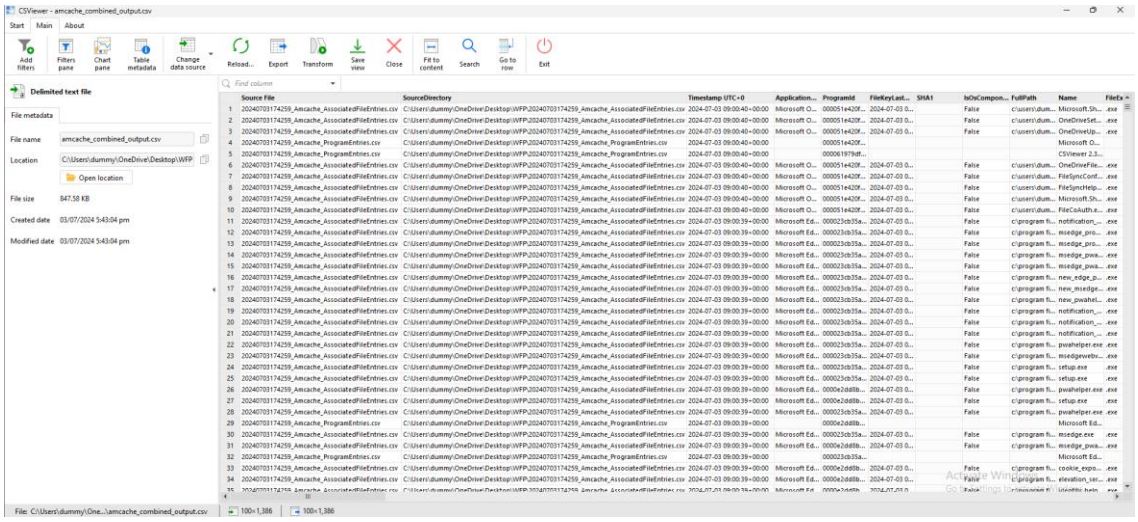

Figure 15: ShimAmCacheParser successfully combining all CSV files from AmcacheParser into the CSV file, amcache_combined_output

In Figure 16, the user can choose to use both tools and set the input directory for AmcacheParser only and set the output directories manually. Figures 17, 18, 19, and 20 show both tools being successfully run, with status messages being printed from their respective tools and the list of generated CSV files.

```
PS C:\Users\dummy\OneDrive\Desktop\WFP> python ShimAmCacheParser.py
Welcome!
Which tool do you want to use? (appcomcache/amcache/both/exit): both
You have chosen to use the both tool.
Do you want to edit the default directories? (yes/no): yes
Enter the path to the Amcache database file (example: C:\Windows\appcompat\Programs\Amcache.hve): C:\Windows\appcompat\Programs\Amcache.hve
Enter the output directory for CSV files: C:\Users\dummy\OneDrive\Desktop\WFP\Output
```

Figure 16: Running both tools simultaneously and choosing to set the directories manually

```
Welcome!
Which tool do you want to use? (appcomcache/amcache/both/exit): both
You have chosen to use the both tool.
Do you want to edit the default directories? (yes/no): yes
Enter the path to the Amcache database file (example: C:\Windows\appcompat\Programs\Amcache.hve): C:\Windows\appcompat\Programs\Amcache.hve
Enter the output directory for CSV files: C:\Users\dummy\OneDrive\Desktop\WFP\Output
AppCompatCache Parser version 1.5.0.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/AppCompatCacheParser

Command line: --csv C:\Users\dummy\OneDrive\Desktop\WFP\Output --dt yyyy-MM-dd HH:mm:sszzz

Processing hive 'Live Registry'

Found 790 cache entries for Windows10C_11 in ControlSet001

Results saved to 'C:\Users\dummy\OneDrive\Desktop\WFP\Output\20240703182120_Windows10C_11_PRINCE_AppCompatCache.csv'


AmcacheParser version 1.5.1.0
```

Figure 17: AppCompatCacheParser successfully running

```
AmcacheParser version 1.5.1.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/AmcacheParser

Command line: -f C:\Windows\appcompat\Programs\Amcache.hve --csv C:\Users\dummy\OneDrive\Desktop\WFP\Output --dt yyyy-MM-dd HH:mm:sszzz -i

Two transaction logs found. Determining primary log...
Primary log: C:\Windows\appcompat\Programs\Amcache.hve.LOG1, secondary log: C:\Windows\appcompat\Programs\Amcache.hve.LOG2
Replaying log file: C:\Windows\appcompat\Programs\Amcache.hve.LOG1
Replaying log file: C:\Windows\appcompat\Programs\Amcache.hve.LOG2
At least one transaction log was applied. Sequence numbers have been updated to 0x010D. New Checksum: 0x9B383AE1
'C:\Windows\appcompat\Programs\Amcache.hve' is in use. Rerouting...

Two transaction logs found. Determining primary log...
Primary log: C:\Windows\appcompat\Programs\Amcache.hve.LOG1, secondary log: C:\Windows\appcompat\Programs\Amcache.hve.LOG2
Replaying log file: C:\Windows\appcompat\Programs\Amcache.hve.LOG1
Replaying log file: C:\Windows\appcompat\Programs\Amcache.hve.LOG2
At least one transaction log was applied. Sequence numbers have been updated to 0x010D. New Checksum: 0x9B383AE1

C:\Windows\appcompat\Programs\Amcache.hve is in new format!

Total file entries found: 480
Total shortcuts found: 75
Total device containers found: 14
Total device PnPs found: 197
Total drive binaries found: 396
Total driver packages found: 6

Found 194 unassociated file entry and 286 program file entries (across 218 program entries)
```

Figure 18: AmcacheParser successfully running

```
C:\Windows\appcompat\Programs\Amcache.hve is in new format!

Total file entries found: 480
Total shortcuts found: 75
Total device containers found: 14
Total device PnPs found: 197
Total drive binaries found: 396
Total driver packages found: 6

Found 194 unassociated file entry and 286 program file entries (across 218 program entries)

Results saved to: C:\Users\dummy\OneDrive\Desktop\WFP\Output

Total parsing time: 0.562 seconds


CSV files generated:
- 20240703182120_Amcache_AssociatedFileEntries.csv
- 20240703182120_Amcache_DeviceContainers.csv
- 20240703182120_Amcache_DevicePnps.csv
- 20240703182120_Amcache_DriveBinaries.csv
- 20240703182120_Amcache_DriverPackages.csv
- 20240703182120_Amcache_ProgramEntries.csv
- 20240703182120_Amcache_ShortCuts.csv
- 20240703182120_Amcache_UnassociatedFileEntries.csv
- 20240703182120_Windows10C_11_PRINCE_AppCompatCache.csv
Combined and sorted CSV saved to C:\Users\dummy\OneDrive\Desktop\WFP\Output\both_combined_output.csv
CSV files have been saved to C:\Users\dummy\OneDrive\Desktop\WFP\Output
Which tool do you want to use? (appcomcache/amcache/both/exit):
```

Figure 19: Outputs from both tools successfully being printed and combined to the output directory

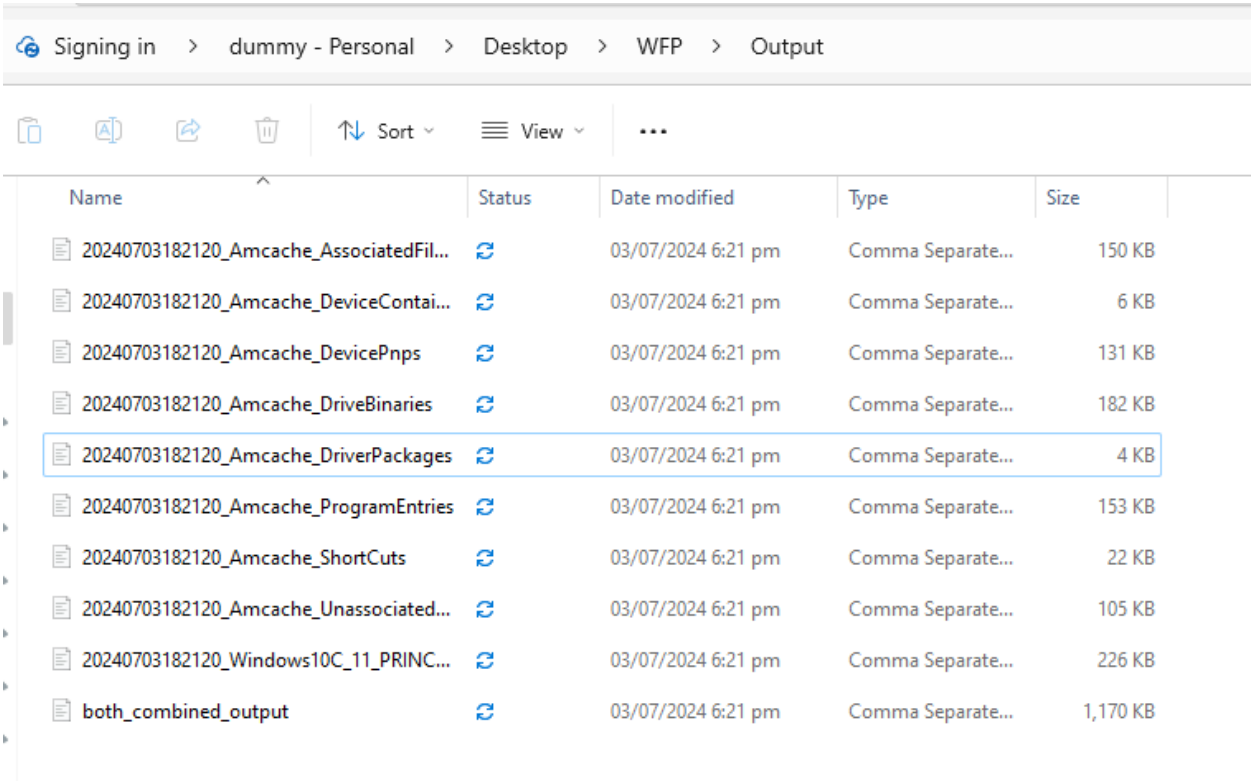

Figure 20: Output saved in the directory that was set manually

In Figure 21, we can see data from both AppCompatCacheParser and AmcacheParser being combined into one singular CSV file called both_combined_output for combining of output.

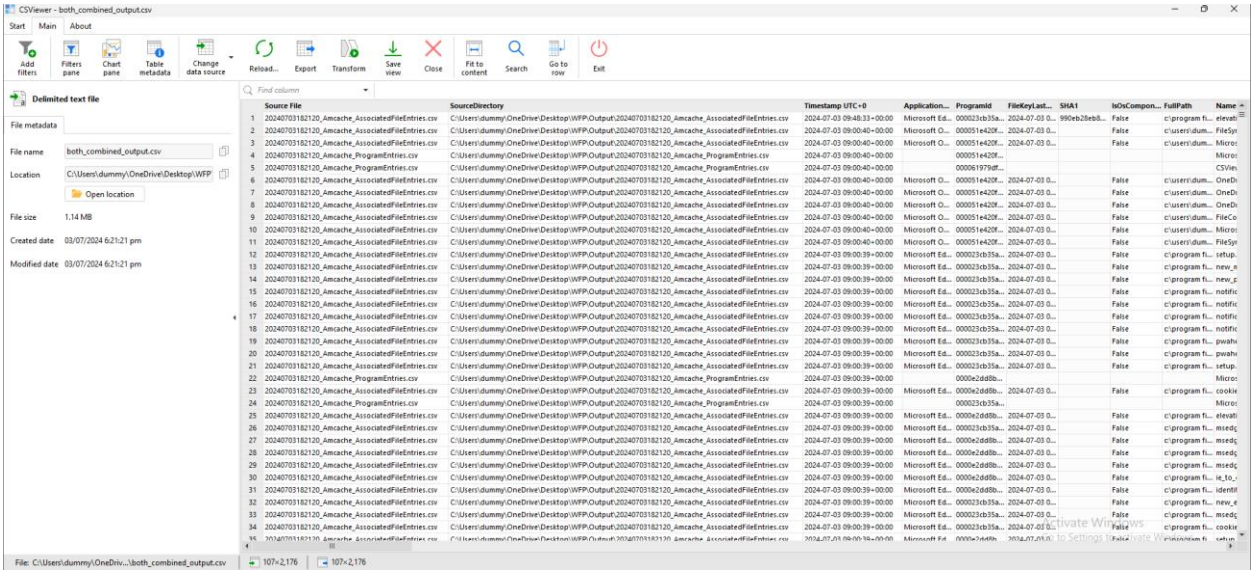

Figure 21: ShimAmCacheParser successfully combining all CSV files from both tools into the CSV file, both_combined_output

C. Timeline, File Information, and File Combining

Timestamp in UTC+0 Format Column

In Eric Zimmerman's tools, it can be observed that the AppCompatCacheParser and AmcacheParser's output time stamp is already in UTC+0 format when running both tools individually with the '--dt "yyyy-MM-ddTHH:mm:ss zzz"' command which would add the hours and minutes offset from UTC in the timestamp (+00:00). Note that all CSV files will be converted into data frames as seen in line 43 in Figure 22 for all these changes to be made.

```
24  def combine_and_sort_csv_files(csv_files, input_folder, output_file):
25      AppCompatCacheCSVFilesFields = {
26          'AppCompatCache': 'LastModifiedTimeUTC',
27      }
28      AmcacheCSVFilesFields = {
29          "Amcache_AssociatedFileEntries":"FileKeyLastWriteTimestamp",
30          "Amcache_DeviceContainers": "KeyLastWriteTimestamp",
31          "Amcache_DevicePnps": "KeyLastWriteTimestamp",
32          "Amcache_DriveBinaries": "KeyLastWriteTimestamp",
33          "Amcache_DriverPackages": "KeyLastWriteTimestamp",
34          "Amcache_ShortCuts": "KeyLastWriteTimestamp",
35          "Amcache_UnassociatedFileEntries": "FileKeyLastWriteTimestamp",
36          "Amcache_ProgramEntries":"KeyLastWriteTimestamp"
37      }
38
39      df_list = []
40
41      for file in csv_files:
42          file_path = os.path.join(input_folder, file)
43          df = pd.read_csv(file_path)
44          sf = file
45          sd = file_path
```

Figure 22. Code to convert CSV files to dataframes

With the AppCompatCacheCSVFilesFields and AmcacheCSVFilesFields list in line 25 to 37 in Figure 22 that indicates the target file name, and the respective field to be used as the timestamp for the 'Timestamp UTC+0' column which will be inserted in the CSV files for both outputs of tools as seen in line 51 for the AppCompatCacheParer CSV files in and line 60 for the AmcacheParser CSV files in Figure 23, respectively. Note that the timestamp will be converted into a datetime object for it to be sorted properly in the combining of the CSV files. Note that the AmcacheParser and AppCompatCacheParser CSV files are already in the proper date/time format.

```
47      for identifier, timestamp_column in AppCompatCacheCSVFilesFields .items():
48          if identifier in file:
49              df.insert(0, 'Source File', sf)
50              df.insert(1, 'SourceDirectory', sd)
51              df.insert(2, 'Timestamp UTC+0', pd.to_datetime(df[timestamp_column], errors='coerce'))
52
53              df.to_csv(file_path, index=False)
54              break
55
56      for identifier, timestamp_column in AmcacheCSVFilesFields.items():
57          if identifier in file:
58              df.insert(0, 'Source File', sf)
59              df.insert(1, 'SourceDirectory', sd)
60              df.insert(2, 'Timestamp UTC+0', pd.to_datetime(df[timestamp_column], errors='coerce'))
61
62              df.to_csv(file_path, index=False)
63              break
64
65      df_list.append(df)
```

Figure 23.Code to insert three columns of SourceFile, SourceDirectory, and Timestamp UTC+0 in every CSV file.

File Destination Column

To indicate the file name and file destination, the 'SourceFile' and 'SourceDirectory' column will be seen in all the CSV files as both columns will be inserted as seen in line 49, 50, 58, and 59 in Figure 23. The file name of the respective CSV file will be sf in line 44 being the 'SourceFile' column and the file destination will be sd being the 'SourceDirectory' column in line 45 as seen in Figure 24.

```
41      for file in csv_files:
42          file_path = os.path.join(input_folder, file)
43          df = pd.read_csv(file_path)
44          sf = file
45          sd = file_path
```

Figure 24. Code to get the file name and file destination

Time zone Adjustment

Note that there will be no time zone adjustments needed to be made with regards to the timestamps as they are already in UTC+0 format.

Merged Columns

In merging the columns of all the CSV files, the loop in line 41 in Figure 24 will combine every CSV file as seen in line 65 as the files are converted to data frames before. Each CSV file's data frame form will be combined with the df_list or date frame list after all the changes and inserts are done in each respective data frame. As seen in line 65 to 72 in Figure 25, the df_list will be initialized as the combined data frame combined_csv and will be sorted according to the 'Timestamp UTC+0' column in ascending order and it will be converted into CSV file for the combined output as seen in line 72.

```
65          df_list.append(df)
66
67      combined_csv = pd.concat(df_list, ignore_index=True)
68
69      if 'Timestamp UTC+0' in combined_csv.columns:
70          combined_csv = combined_csv.sort_values(by='Timestamp UTC+0', ascending=False)
71
72          combined_csv.to_csv(output_file, index=False)
73          print(f"Combined and sorted CSV saved to {output_file}")
```

Figure 25. Combining and sorting of CSV files.

Due to the three new columns of the file name, file destination, and timestamp in UTC+0 format which are added in all the CSV files, the combining of all the outputs will be seamless as seen in Figure 28. There are no row duplicates also as it is confirmed by the RowCountChecker as seen in Figure 39. Note that there will not be any column duplicates also. For example, in Figure 26 and 27, the column 'DriverName' appeared in different CSV files, but their respective DriverName data are in the same column located in the combined output.

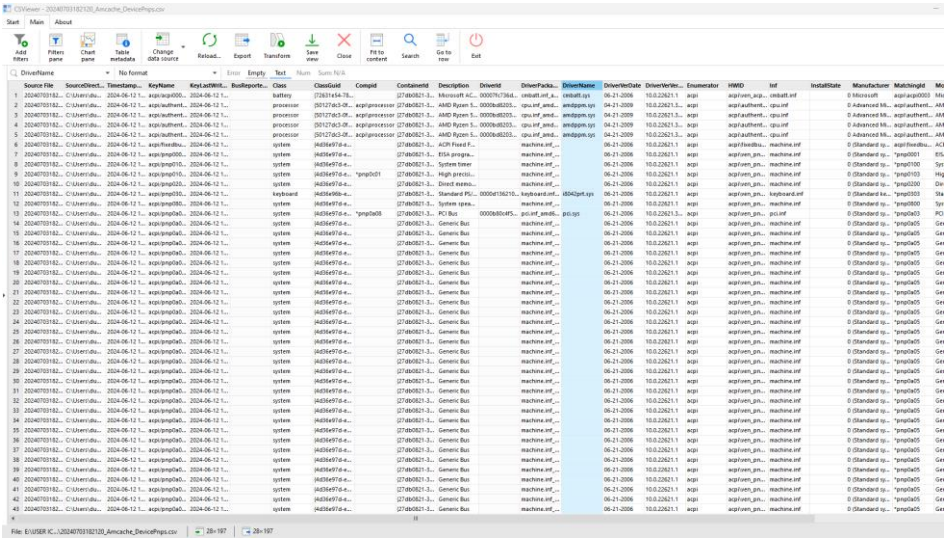

Figure 26. DriverName column of Amcache_DevicePnps

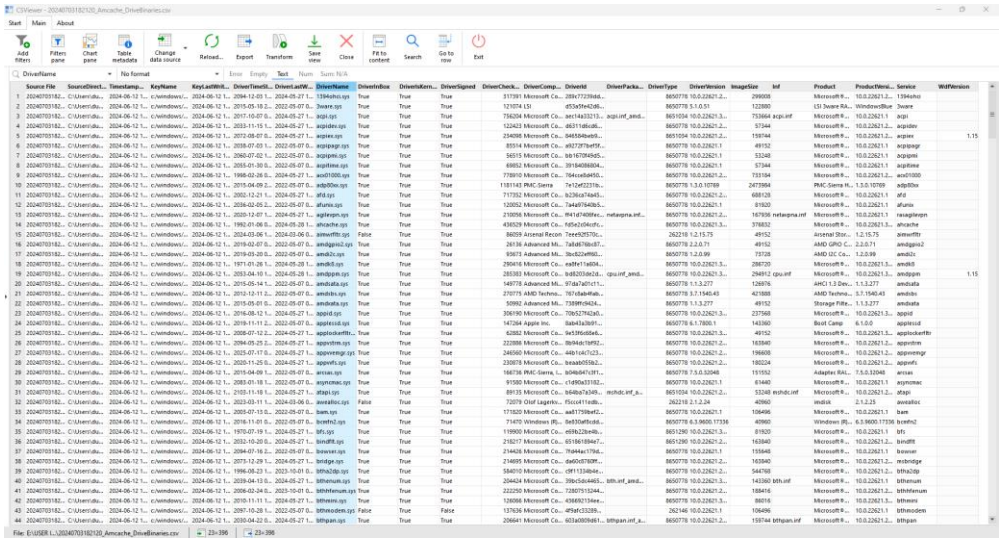

Figure 27:DriverName column of Amcache_DriveBinaries

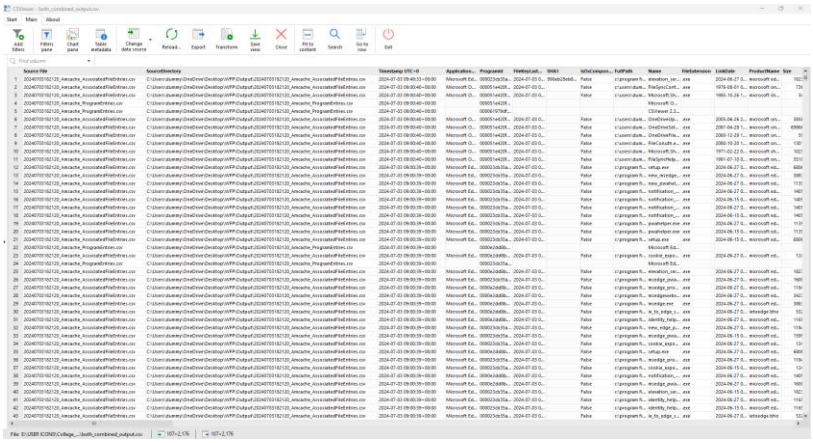

Figure 28: Contents of both_combined_output.csv

```
PS E:\USER ICONS\College_Work\Third Year (Third Term)\NSSECU3\WFP\OUTPUT> python RowCountChecker.py
20240703182120_Amcache_AssociatedFileEntries.csv: 286 rows
20240703182120_Amcache_DeviceContainers.csv: 14 rows
20240703182120_Amcache_DevicePnps.csv: 197 rows
20240703182120_Amcache_DriveBinaries.csv: 396 rows
20240703182120_Amcache_DriverPackages.csv: 6 rows
20240703182120_Amcache_ProgramEntries.csv: 218 rows
20240703182120_Amcache_ShortCuts.csv: 75 rows
20240703182120_Amcache_UnassociatedFileEntries.csv: 194 rows
20240703182120_Windows10C_11_PRINCE_AppCompatCache.csv: 790 rows
both_combined_output.csv: 2176 rows
Total: 4352 rows
PS E:\USER ICONS\College_Work\Third Year (Third Term)\NSSECU3\WFP\OUTPUT>
```

Figure 29. Using RowCountChecker.py to check if there are any row duplicates

D. Detailed Timeline Analysis

In this analysis, the group will present the different timestamps parsed from the ShimCache and Amcache hive, and we will demonstrate how executable files are flushed into the ShimCache and Amcache by providing figures that show what actions can influence the timestamps.

ShimCache timestamps

As seen in Figure 30, the group has extracted the sample exe file, called ImHere.exe, currently it located in the folder with the same file name. With the sample exe file, we will be demonstrating how executable files are not flushed immediately into ShimCache. In Figure 31, we used the command '*dir/w'* to view the contents of the file without viewing it in File explorer, this is done to present that we have not executed the file yet.

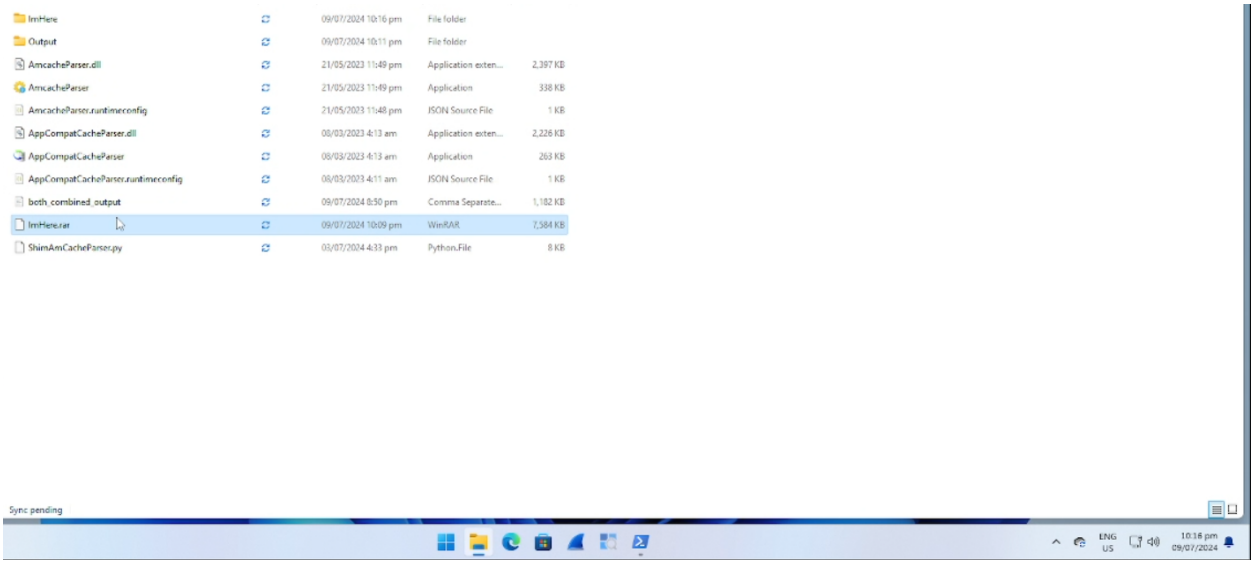

Figure 30: Extracting the exe file, ImHere, from the ZIP file

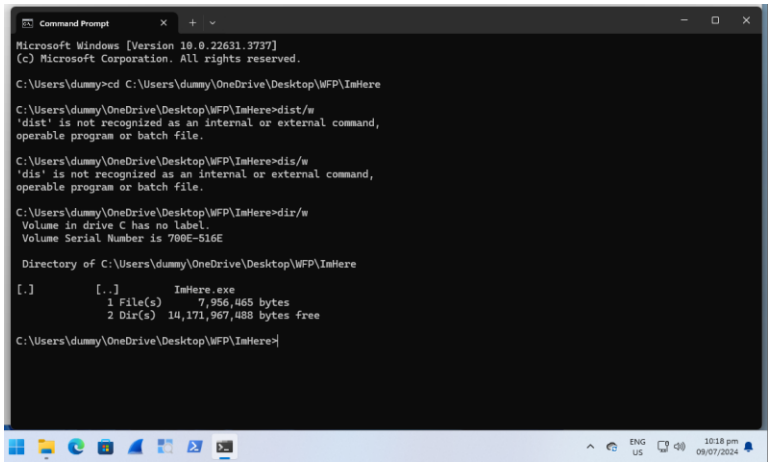

Figure 31: Checking the contents of the folder of ImHere without opening it in file explorer

Now as seen in Figure 32, we have parsed the ShimCache with the created tool, ShimAmCacheParser. As shown, no records of the exe file ImHere, has been registered.

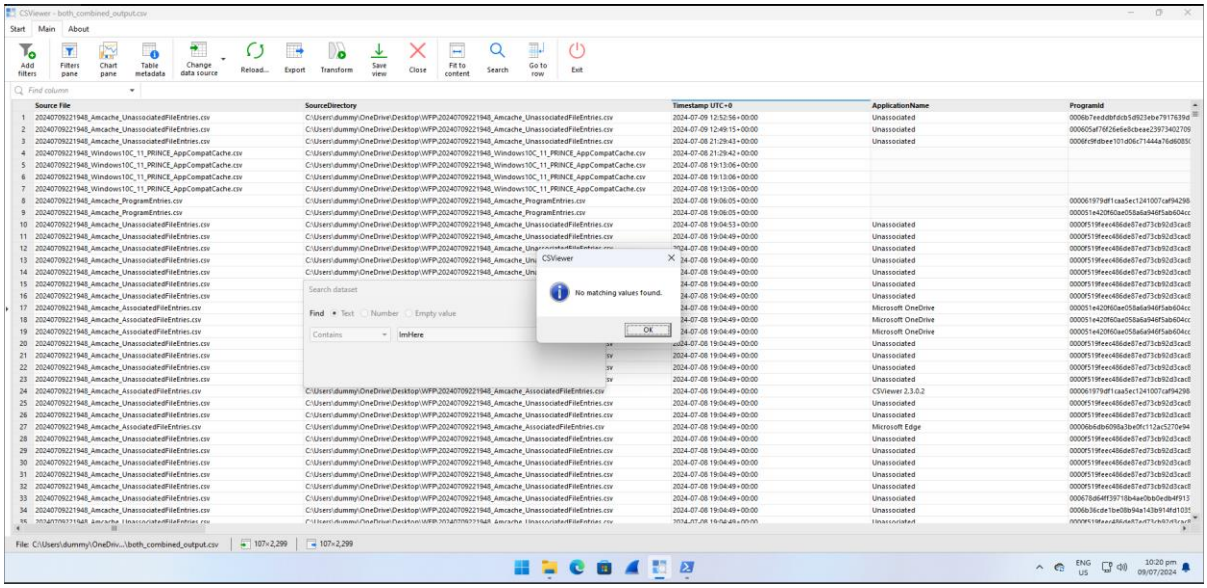

Figure 32: ImHere.exe is not in the ShimCache after file extraction

In Figure 33, we ran the executable file for the first time. ImHere.exe simply prints out a statement, which is why the command line closes instantly. After running ImHere.exe, the groups parse the ShimCache for the second time, and as seen in Figure 34, the executable is not yet registered into the ShimCache. Do note the time of execution is July 9, 2024, 10:21:55 PM UTC+8.

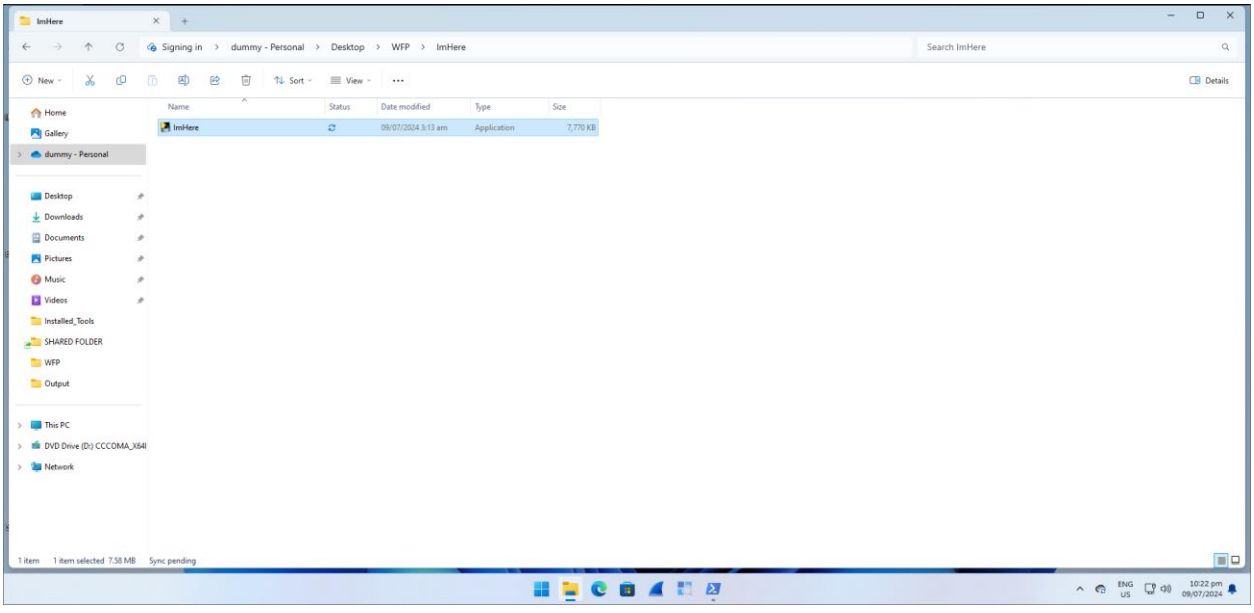

Figure 33: Executing ImHere.exe for the first time

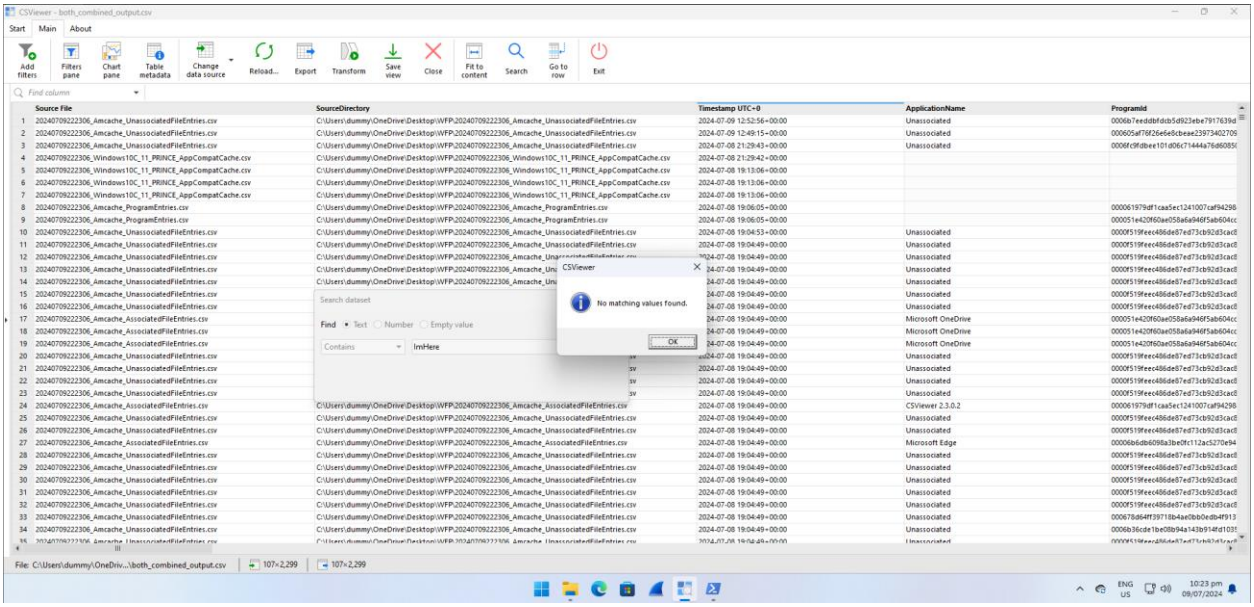

Figure 34: ImHere.exe not listed in the ShimCache, even after execution

As seen in Figure 35, we restarted the system as the ShimCache only contains the information prior to the system's last startup, as current entries are stored only in memory [13].

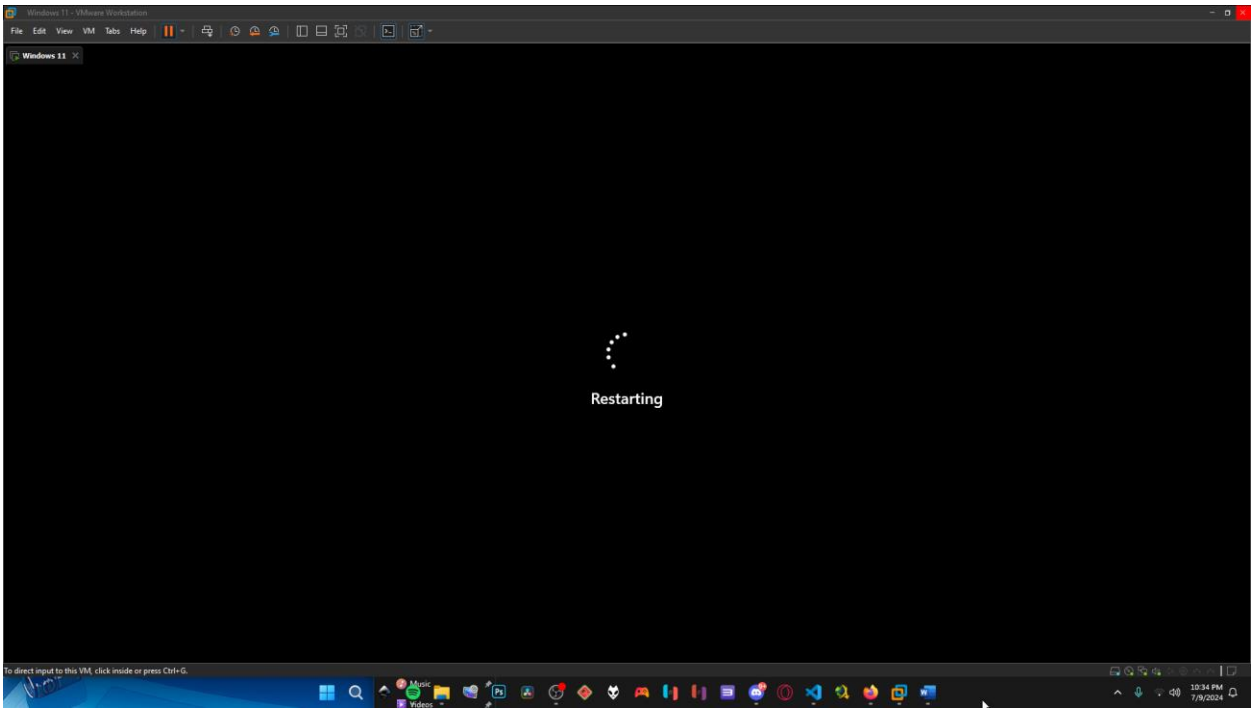

Figure 35: Restarting the system after executing ImHere.exe

After successfully restarting the system, the group parses the ShimCache again, and now the ImHere.exe file is finally registered in the ShimCache, as seen in Figure 36. As explained before, the timestamp listed in the ShimCache is when the file was last modified, as seen in Figure 37, where the time is adjusted to UTC+8. This is why the timestamp is 2024-07-08 19:13:06+00:00 and not 2024-07-09 14:21:55+00:00, which is when we ran ImHere.exe.

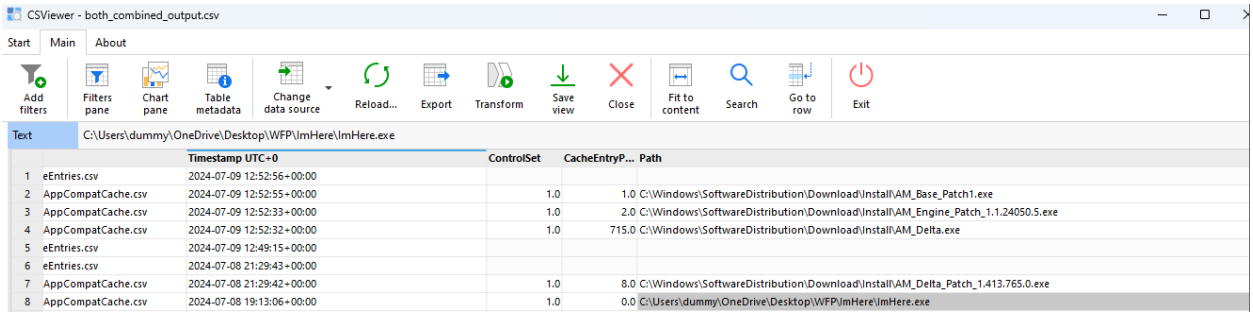

Figure 36: ImHere.exe is now inside the ShimCache

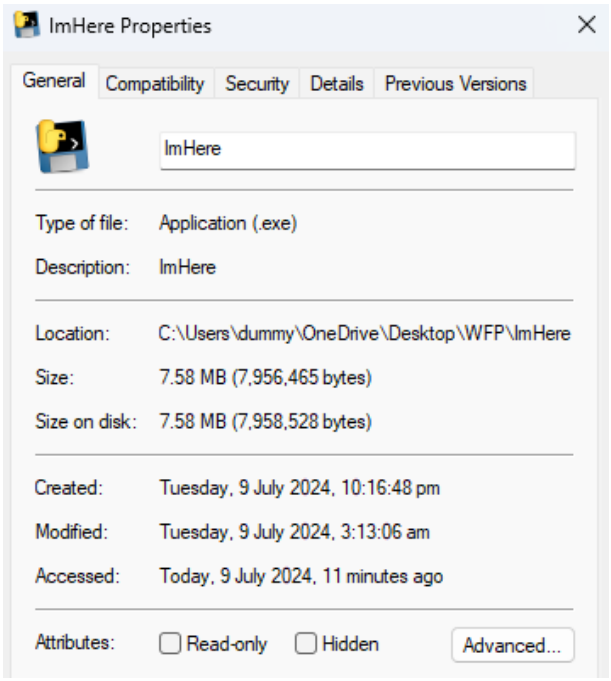

Figure 37: File properties of ImHere.exe

To recap, ShimCache is a valuable artifact found on Windows systems that can provide forensic investigators with insights into the execution history of applications. It provides metadata such as the full file path, file size, last modified time, and last updated time of the file [13]. However, it is important to note that ShimCache is not a reliable source for proving execution. Despite its ability to show proof of execution, the system needs to be restarted first to flush the current entries out of memory, as ShimCache only contains information prior to the system's last startup [13]. Additionally, the timestamp does not

indicate the exact execution time; it only shows the last modified time. As illustrated in Figure 36, the execution time and the listed timestamp can be significantly different. ShimCache is better used as evidence of existence, as the system registers an executable to the ShimCache without needing to be executed; it just needs to be viewed in any file explorer [14]. Therefore, it is more appropriate to use ShimCache as supporting evidence alongside more reliable proof of execution, such as the Amcache.

Amcache timestamps

As seen in Figure 38, the group has extracted the sample exe file, called ImHereAgain.exe. With the sample exe file, we will be demonstrating how executable files are flushed immediately into AmCache after the operating system runs Microsoft Compatibility Appraiser. In Figure 39 we used the command '*dir/w*' to view the contents of the file without viewing it in File explorer, this is done to present that we have not executed the file yet.

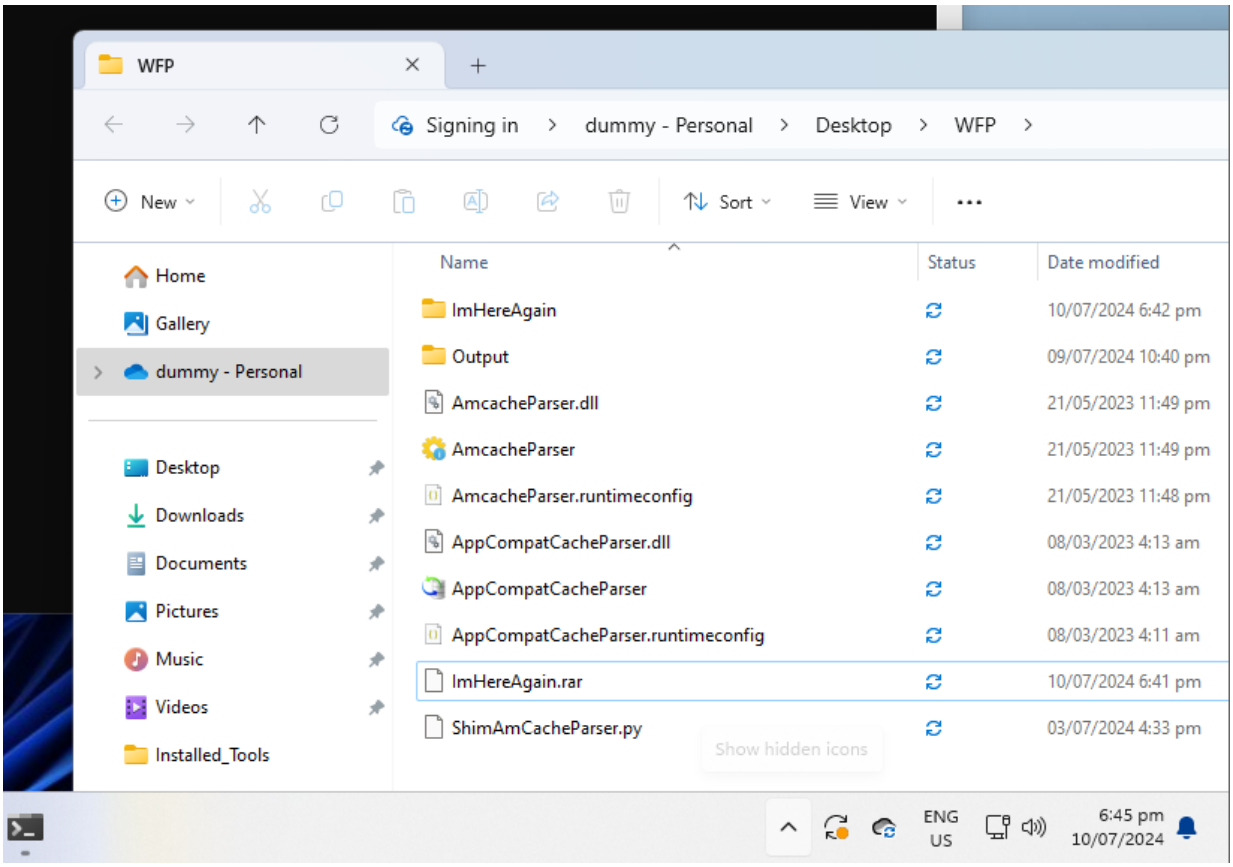

Figure 38: Extracting the exe file, ImHereAgain, from the ZIP file

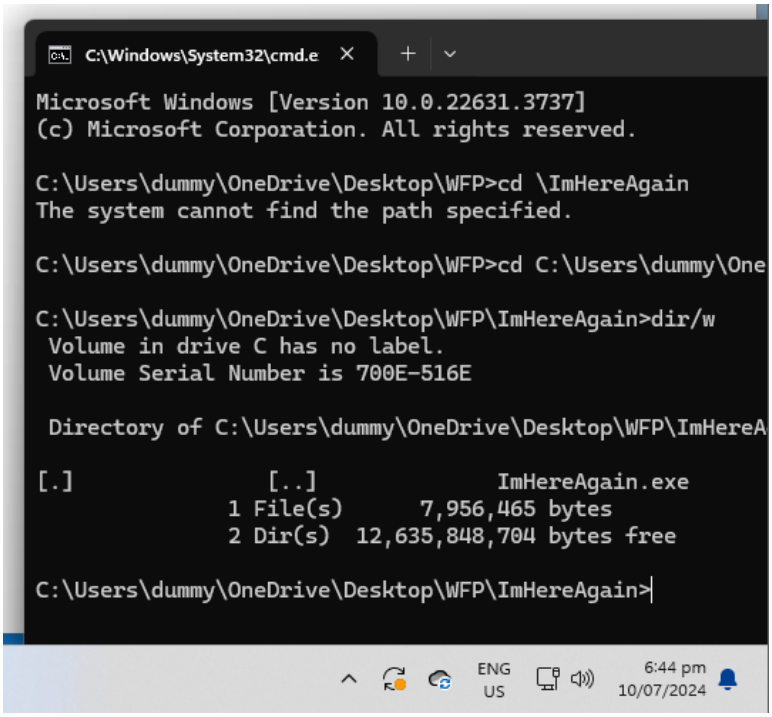

Figure 39: Checking the contents of the folder of ImHereAgain without opening it in file explorer

Now as seen in Figure 40, we have parsed the AmCache with the created tool, ShimAmCacheParser. As shown, no records of the exe file ImHereAgain, has been registered.

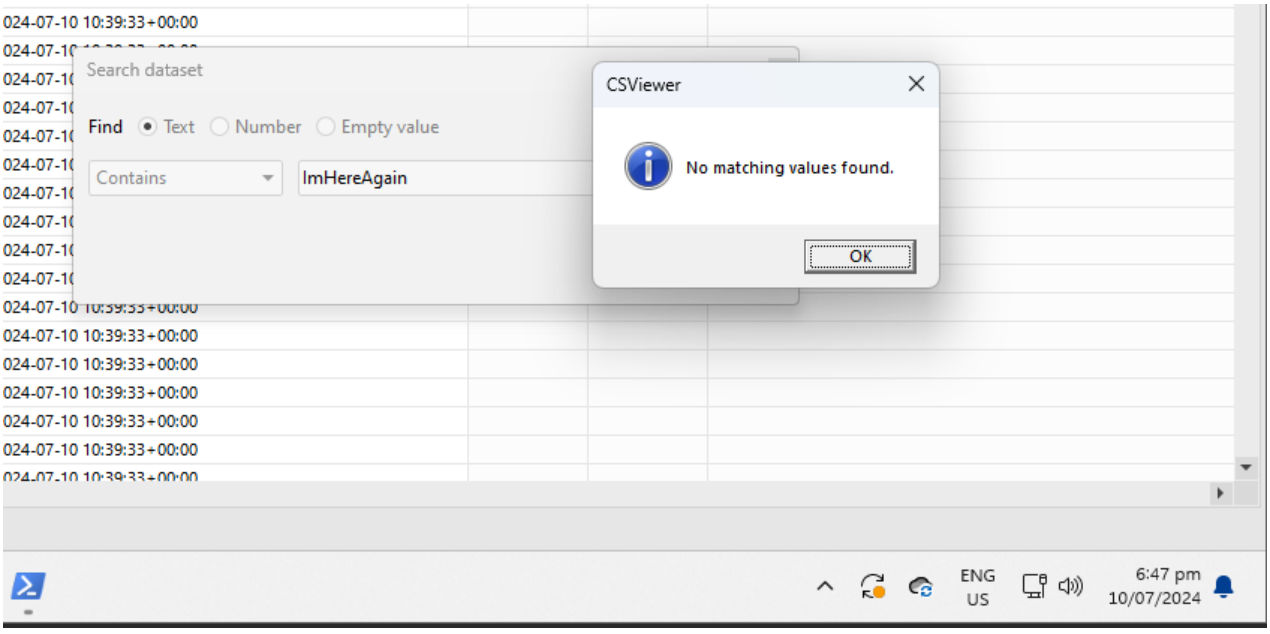

Figure 40:ImHereAgain.exe is not in the Amcache after file extraction

As noted previously, an automated scheduled task called the 'Microsoft Compatibility Appraiser' is executed to gather information about files found within the system storage. When this system tool runs, it flushes the gathered information into the AmCache. This tool usually runs automatically once per day, but as seen in Figure 41, the group can run it manually to populate the AmCache, as seen the group runs the tool at 6:49pm [7].

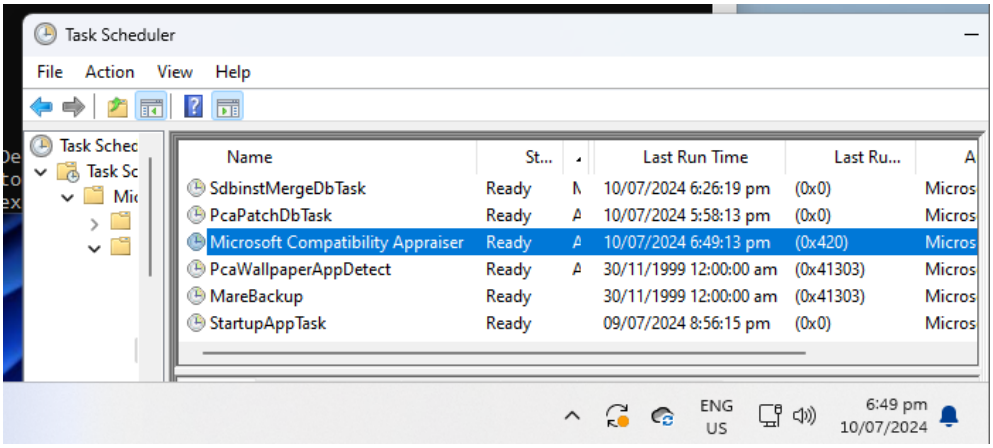

Figure 41: Running Microsoft Compatibility Appraiser for the first time

After running the Microsoft Compatibility Appraiser, the ImHereAgain.exe file is still not inside the AmCache, as seen in Figure 42. The group have not yet executed the file, which is why it has not been recorded yet. In Figure 43, the group will now finally run the ImHereAgain.exe file for the first time at 6:54pm.

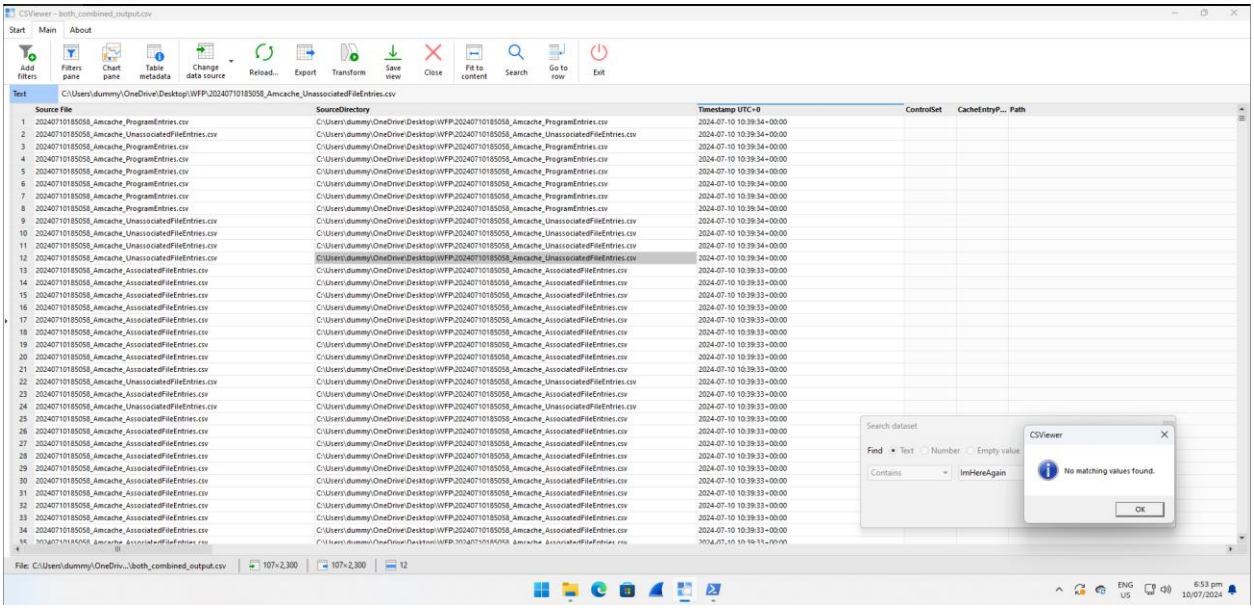

Figure 42: ImHereAgain.exe is not in the Amcache after running Microsoft Compatibility appraiser

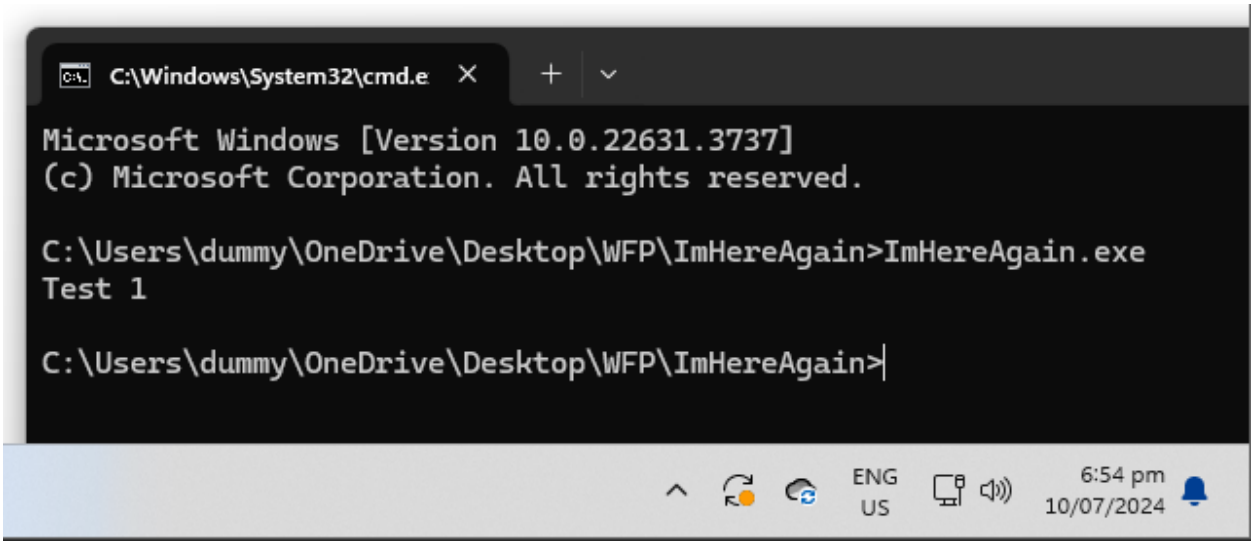

Figure 43: Running ImHereAgain.exe for the first time

In Figure 44, after running the ImHereAgain.exe file for the first time, we run Microsoft Compatibility Appraiser again for the second time at 6:57pm.

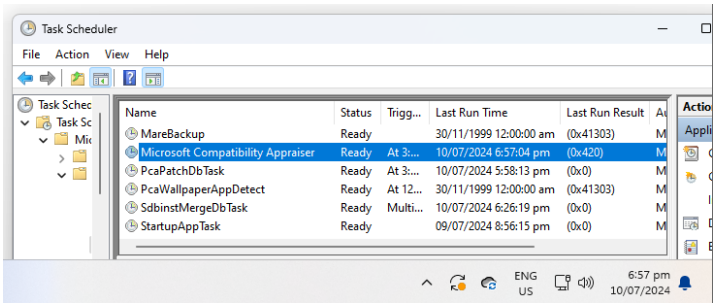

Figure 44: Running Microsoft Compatibility Appraiser after running ImHereAgain.exe

The ImHereAgain.exe file has successfully been flushed into the AmCache after being executed, as seen in Figure 45. Note that the group did not have to reset the system, unlike with ShimCache. Additionally, the timestamp in AmCache matches when the executable file was last accessed, as seen in Figure 46. We ran the ImHereAgain.exe file at 6:54 PM, and the timestamp in the AmCache is 10:54:33+00:00. When converted to UTC+8 and 12-hour format, we get 6:54 PM as well.

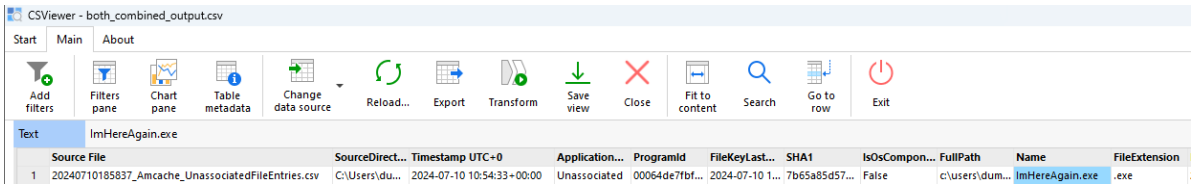

Figure 45: ImHereAgain.exe is located inside the Amcache

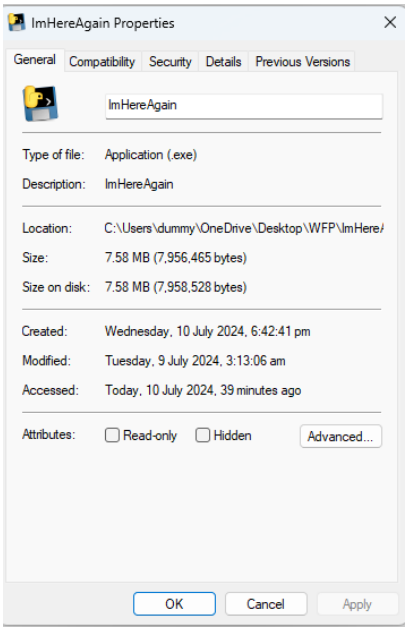

Figure 46: File properties of ImHereAgain.exe

In conclusion, the AmCache serves as a better artifact in providing proof that an application has been executed. It contains a list of executed programs and metadata such as the file name, the name of the software product to which the executable belongs (product name), product version, file path, file size, and when the program was last executed. This provides more detailed information regarding the executable file. Additionally, the user does not need to restart the system; instead, they can run the system tool, Microsoft Compatibility Appraiser, to scan the system for recently executed tools. Since AmCache provides a timestamp that correlates with when the program was last executed, it becomes easier to identify the timeline of what tools were run before a specific event on the system [12].

E. Approval of Tools

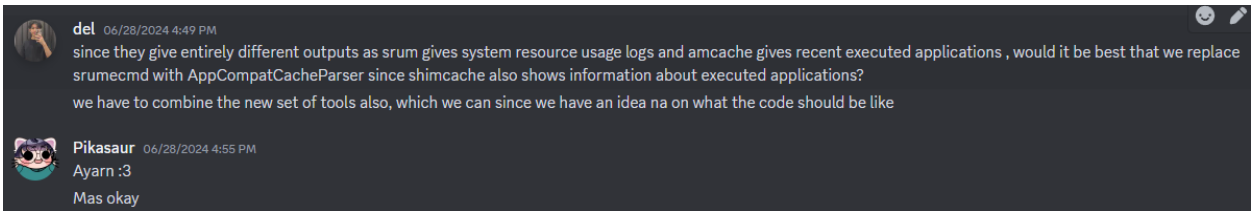

Figure 47: Proof of approved tools

VII. Conclusion

The group has successfully created a tool called “ShimAmCacheParser” that combines the two forensic tools, AppCompatCacheParser and AmcacheParser by Eric Zimmerman.

Using Python along with the Subprocess and Pandas libraries, we were able to achieve our goals and objectives, such as providing the user the option to either run one tool individually or both tools simultaneously and producing three additional columns for file name, file destination, and timestamp in UTC+0 format that can be used when creating a timeline. Ideally, the ShimAmCacheParser should run by using AppCompatCacheParser and AmcacheParser one at a time (single tool mode) since the combined outputs are easier to navigate as they have lesser columns and better simplified also. Additionally, ShimAmCacheParser is user-friendly, offering an interactive command line that allows the user to choose which tool to use and to change the input or output directory if needed. The group has successfully analyzed the differences between AmCache and ShimCache in regard to digital forensics. The group found that AmCache is much more suitable to use as evidence when showing proof of execution, as the timestamp in AmCache actually represents the last run time of the executable. In contrast, ShimCache displays when the file was last modified. Additionally, there is no need to restart the system to update AmCache; the user simply needs to run the system tool, Microsoft Compatibility Appraiser. ShimCache, on the other hand, is better suited to be used as supporting evidence, as it still proves that an executable was run on the system.

VIII. Appendix

A. ShimAmCacheParser.py

```
import os
import subprocess
import pandas as pd

def run_tool(tool_path, databasefile, outputfolder):
    if not os.path.isfile(tool_path):
        print(f"{tool_path} IS NOT FOUND. Exiting...")
        return False
    try:
        default_output_folder = os.path.dirname(os.path.abspath(__file__))
        Amcache = os.path.join(default_output_folder,"AmcacheParser.exe")
        if tool_path == Amcache:
            result = subprocess.run([tool_path, '-f', databasefile, '--csv', outputfolder, '--dt',
'yyyy-MM-dd HH:mm:sszzz', '-i'], check=True, capture_output=True, text=True)
            print(result.stdout)
        else:
            result = subprocess.run([tool_path, '--csv', outputfolder, '--dt', 'yyyy-MM-dd
HH:mm:sszzz'], check=True, capture_output=True, text=True)
            print(result.stdout)
        return True
    except subprocess.CalledProcessError as e:
        print(f"Error running tool: {e}")
        print(f"Output: {e.output}")
        return False

def combine_and_sort_csv_files(csv_files, input_folder, output_file):
    AppCompatCacheCSVFilesFields = {
        'AppCompatCache': 'LastModifiedTimeUTC',
    }
```

```
    AmcacheFilesFields = {
        "Amcache_AssociatedFileEntries":"FileKeyLastWriteTimestamp",
        "Amcache_DeviceContainers": "KeyLastWriteTimestamp",
        "Amcache_DevicePnps": "KeyLastWriteTimestamp",
        "Amcache_DriveBinaries": "KeyLastWriteTimestamp",
        "Amcache_DriverPackages": "KeyLastWriteTimestamp",
        "Amcache_ShortCuts": "KeyLastWriteTimestamp",
        "Amcache_UnassociatedFileEntries": "FileKeyLastWriteTimestamp",
        "Amcache_ProgramEntries":"KeyLastWriteTimestamp"
    }

    df_list = []

    for file in csv_files:
        file_path = os.path.join(input_folder, file)
        df = pd.read_csv(file_path)
        sf = file
        sd = file_path

        for identifier, timestamp_column in AppCompatCacheCSVFilesFields .items():
            if identifier in file:
                df.insert(0, 'Source File', sf)
                df.insert(1, 'SourceDirectory', sd)
                df.insert(2, 'Timestamp UTC+0', pd.to_datetime(df[timestamp_column],
errors='coerce'))

                df.to_csv(file_path, index=False)
                break

        for identifier, timestamp_column in AmcacheFilesFields.items():
            if identifier in file:
                df.insert(0, 'Source File', sf)
                df.insert(1, 'SourceDirectory', sd)
                df.insert(2, 'Timestamp UTC+0', pd.to_datetime(df[timestamp_column],
errors='coerce'))

                df.to_csv(file_path, index=False)
                break

        df_list.append(df)

    combined_csv = pd.concat(df_list, ignore_index=True)

    if 'Timestamp UTC+0' in combined_csv.columns:
        combined_csv = combined_csv.sort_values(by='Timestamp UTC+0', ascending=False)

        combined_csv.to_csv(output_file, index=False)
        print(f"Combined and sorted CSV saved to {output_file}")

def get_default_input_directories(tool_name):
    if tool_name == 'amcache':
        default_database_file = "C:\\Windows\\appcompat\\Programs\\Amcache.hve"
        default_output_folder = os.path.dirname(os.path.abspath(__file__))
    elif tool_name == 'appcomcache':
        default_database_file = ""
        default_output_folder = os.path.dirname(os.path.abspath(__file__))
    else:
        default_database_file = ""
        default_output_folder = ""
    return default_database_file, default_output_folder

if __name__ == "__main__":
    print("Welcome!")

    while True:
```

```
        tool_choice = input("Which tool do you want to use? (appcomcache/amcache/both/exit):
").strip().lower()

        if tool_choice == 'exit':
            print("Exiting...")
            break
        elif tool_choice not in ['appcomcache', 'amcache', 'both']:
            print("Invalid choice.")
            continue

        print(f"You have chosen to use the {tool_choice} tool.")

        edit_directories = input("Do you want to edit the default directories? (yes/no):
").strip().lower()

        if edit_directories not in ['yes', 'no']:
            print("Invalid choice.")
            continue

        if edit_directories == 'yes' and tool_choice in ['appcomcache']:
            outputfolder = input("Enter the output directory for CSV files: ").strip()
        elif edit_directories == 'yes' and tool_choice in ['amcache']:
            databasefile_amcache = input("Enter the path to the Amcache database file (example:
C:\\Windows\\appcompat\\Programs\\Amcache.hve): ").strip()
            outputfolder = input("Enter the output directory for CSV files: ").strip()
        elif edit_directories == 'yes' and tool_choice in ['both']:
            databasefile_amcache = input("Enter the path to the Amcache database file (example:
C:\\Windows\\appcompat\\Programs\\Amcache.hve): ").strip()
            outputfolder = input("Enter the output directory for CSV files: ").strip()
        else:
            databasefile_amcache, _ = get_default_input_directories('amcache')
            _, outputfolder = get_default_input_directories('appcomcache')

        csv_files = []
        if tool_choice in ['appcomcache']:
            appcompatcache_parser_path = os.path.abspath("AppCompatCacheParser.exe")
            if run_tool(appcompatcache_parser_path, None, outputfolder):
                csv_files.extend([file for file in os.listdir(outputfolder) if
file.endswith('.csv')])

        elif tool_choice in ['amcache']:
            if not os.path.isfile(databasefile_amcache):
                print(f"ERROR: The database file '{databasefile_amcache}' does not exist.
Exiting...")
            else:
                amcache_parser_path = os.path.abspath("AmcacheParser.exe")
                if run_tool(amcache_parser_path, databasefile_amcache, outputfolder):
                    csv_files.extend([file for file in os.listdir(outputfolder) if
file.endswith('.csv')])
        elif tool_choice in ['both']:
            if not os.path.isfile(databasefile_amcache):
                print(f"ERROR: The database file '{databasefile_amcache}' does not exist.
Exiting...")
            else:
                appcompatcache_parser_path = os.path.abspath("AppCompatCacheParser.exe")
                checkappcompatcache = run_tool(appcompatcache_parser_path, None, outputfolder)
                amcache_parser_path = os.path.abspath("AmcacheParser.exe")
                checkamcache = run_tool(amcache_parser_path, databasefile_amcache, outputfolder)

                if checkappcompatcache or checkamcache:
                    csv_files.extend([file for file in os.listdir(outputfolder) if
file.endswith('.csv')])

        if csv_files:
            print("CSV files generated:")
```

```
            for csv_file in csv_files:
                print(f"- {csv_file}")
            combine_and_sort_csv_files(csv_files, outputfolder, os.path.join(outputfolder,
tool_choice + '_combined_output.csv'))
        else:
            print("No CSV files generated.")

        print(f"CSV files have been saved to {outputfolder}")

        if os.name == 'nt':
            os.startfile(outputfolder)
        else:
            print(f"Please open the following directory to view the CSV files: {outputfolder}")
```

IX. References

[1] "Investigating AmCache," ArtiFast, [Online]. Available: https://forensafe.com/blogs/AmCache.html. [Accessed 17 June 2024].

[2] eyehatemalwares, "Incident Response with EZTools - Evidence of Execution," eyehatemalwares, [Online]. Available: https://www.eyehatemalwares.com/incident-response/eztools/amcacheparser/. [Accessed 18 June 2024].

[3] Muldwych, "DFIR tools AmcacheParser what is it how to use," thesecuritynoob, 9 December 2022. [Online]. Available: https://thesecuritynoob.com/dfir-tools/dfir-tools-amcacheparser-what-is-it-how-to-use/. [Accessed 18 June 2024].

[4] Y. Khatri, "Amcache.hve in Windows 8 - Goldmine for malware hunters," swiftforensics, 4 12 2013. [Online]. Available: http://www.swiftforensics.com/2013/12/amcachehve-in-windows-8-goldmine-for.html. [Accessed 18 June 2024].

[5] E. Zimmerman, "Eric Zimmerman's tools," [Online]. Available: https://ericzimmerman.github.io/#!index.md. [Accessed 18 June 2024].

[6] M. Learn, "Custom date and time format strings," 4 December 2022. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/standard/base-types/custom-date-and-time-format-strings. [Accessed 30 June 2024].

[7] thedfirspot, "Evidence of Program Existence - Amcache," thedfirspot, [Online]. Available: https://www.thedfirspot.com/post/evidence-of-program-existence-amcache. [Accessed 2 July 2024].

[8] svch0st, "Battle of the Shims," Medium, 5 Dec 2018. [Online]. Available: https://svch0st.medium.com/battle-of-the-shims-60fdae38264e. [Accessed 2 July 2024].

[9] T. S. Noob, "[DFIR TOOLS] AppCompatCacheParser, what is it & how to use !," thesecuritynoob, 23 September 2022. [Online]. Available: https://thesecuritynoob.com/dfir-tools/dfir-tools-amcacheparser-what-is-it-how-to-use-2/. [Accessed 3 July 2024].

[10] eyehatemalwares, "Incident Response with EZTools - Evidence of Execution," eyehatemalwares, [Online]. Available: https://www.eyehatemalwares.com/incident-response/eztools/appcompatcacheparser/. [Accessed 3 July 2024].

[11] M. B, "Tooling Thursday: AppCompatCacheParser," medium, 16 Dec 2016. [Online]. Available: https://bromiley.medium.com/tooling-thursday-appcompatcacheparser-ccf5f6bf0b0d. [Accessed 3 July 2024].

[12] thedfirspot, "Evidence of Program Existence - Shimcache," thedfirspot, [Online]. Available: https://www.thedfirspot.com/post/evidence-of-program-existence-shimcache. [Accessed 4 July 2024].

[13] A. Fortuna, "Amcache and Shimcache in forensic analysis," Andrea Fortuna, 16 Oct 2017. [Online]. Available: https://andreafortuna.org/2017/10/16/amcache-and-shimcache-in-forensic-analysis/. [Accessed 10 July 2024].

[14] 13Cubed, "Let's Talk About Shimcache - The Most Misunderstood Artifact," 13Cybed, 19 July 2021. [Online]. Available: https://www.youtube.com/watch?v=7byz1dR_CLg&t=531s. [Accessed 11 July 2024].