SeeShells: An Optimized Solution for Utilizing Shellbags in a Digital Forensic Investigation

Edward L. Amoruso
Department of Computer Science
University of Central Florida
Orlando, FL 32816 US
eamoruso@knights.ucf.edu

Richard Leinecker
Department of Computer Science
University of Central Florida
Orlando, FL 32816 US
Richard.Leinecker@ucf.edu

Cliff C. Zou
Department of Computer Science
University of Central Florida
Orlando, FL 32816 US
changchun.zou@ucf.edu

***Abstract*— The Windows registry is a treasure trove for digital forensics investigators. Shellbags, an important element in registry, can assist investigators with detailed timeline evidence. Several existing applications provide access to Shellbags, but they lack a complete and effective interface for searching and reporting event timelines. In this paper, we develop an optimized and configurable application called "SeeShells" to query Shellbags to build history of criteria-based events and efficiently display them in a rich user interface to facilitate forensic investigation. Our application provides analysis capabilities to flag suspicious events in an easy-to-view frequency map with corresponding event labels. Our frequency map, also known as a heat map, will show density plots in a range of colors to identify the intensity of activities satisfying a query. In addition, our application can export parsed timeline event information into various commonly used file formats to compliment an investigator's digital forensic report.**

Keywords*— *Digital forensic investigation, Shellbags, registry datatype, event timelines, event frequency

I. Introduction

Windows operating systems have a special hierarchical database that holds important, and many times mission critical information. It is known as the Windows Registry. According to the Microsoft Computer Dictionary, the registry contains user profiles for each user, installed application, document types, property sheet settings for folders and application icons, detected hardware, and ports in use during the system's operation which is continually referenced [1]. This data is so crucial that system restore points always contain a copy of the registry so that in case of catastrophic system failure, an older copy of the registry can be restored. It includes system and hardware configuration information, user data, installed software data, and much more. The registry is organized into sections that contain similar information. Each of these maps to one of seven specific physical files. There are four major classes of registry hives: system configuration, current user, local machine, and users. The local machine grouping is further subdivided into security accounts, security, software, and system. It is interesting to note that when a user logs in, the current user data is taken from their own personal hive. This is how each user has their own desktop, among other different configurations.

The Windows registry contains a plethora of information that a digital forensics investigator can leverage for evidence in various types of cases. The registry data type that this paper focuses on are known as Shellbags. Introduced in Windows XP, Shellbags are utilized by the operating system to help record views, sizes, and positions of a folder when accessed by the current user. In Windows XP, Shellbags information is stored in the NTUSER.dat file located in the following folder path "%UserProfile%\LocalSettings\ApplicationData\Microsoft\Windows". But in later releases of Windows, such as Windows Vista thru Windows 11, another file called USRCLASS.dat located in the following folder path "%UserProfile%\AppData\Local\Microsoft\Windows" was added having predominance of Shellbags information [2]. They contain information about user activity such as folder interaction, folder locations, folder existence even after deletion, timestamps and more. While this may not be the smoking gun that an investigator is looking for, it assists in developing a timeline. And timelines in forensic investigations are extremely important since they lead to crime solutions. The mechanism for developing timelines based on the shellbag evidence is using the time and date stamp of each registry entry. For instance, if a folder is interacted with at a certain time and date, then that represents a single user action and a shellbag record is created or updated. With many such events that have associated times and dates, the timeline becomes more complete. Table I shows what information you can find from Shellbags, and what information you cannot find from Shellbags.

TABLE I. INFORMATION FOUND AND NOT FOUND IN SHELLBAGS

Contains	Does Not Contain
Folders last interacted with	Folders created with "md" or
Location of folders	WSL (Windows Subsystem for Linux) "mkdir" command
Evidence of previously existing folder	Evidence of program execution
Folder on external devices or network resources	
Folders that have existed on the desktop	
Timestamp of last interaction time	
Timestamp of when a folder was selected	

The contributions of this paper are:

- An optimized and ready-to-use software application that can be utilized by digital forensics investigators.
- A graphical heat map of events, color coded by type, captured by Shellbags to help with anomaly detection.
- Facilitate the capability to easily filter global events by attributes such as type, path, user, registry hive, begin date, and end date.
- Provide extensive and flexible reporting module to supplement reports with corresponding graphical heat maps.

The rest of this paper is organized as follows. Section II covers an overview of other related works and how our paper differs. In Section III, we introduce our proposed approach. Section IV we discuss our implementation. Section V will go over an evaluation based on a case study. Finally, section VI provides our conclusion.

II. Related Work

There are numerous proposed methods of shellbag analysis techniques [3] [4] [5]. Also, there are several applications available that have been developed by digital forensic organizations and other entities. In the process of evaluating several of these programs, they all shared a common theme that required the user to manually search and extract the shellbag's information in a spreadsheet-like representation.

During our research for free shellbag analysis tools, we were presented with many applications, both Shellbags specific and general artifact recovery. Most of the applications found performed many other features besides searching for shellbag artifacts, making them lack in efficient and visually focused shellbag analysis [6] [7].

The most commonly available and free tools designed specifically for shellbag analysis consist of the following:

- Shellbags Explorer [8]
- ShellBagger [9]
- Shellbagsview [10]

After evaluating the three above-mentioned applications, we found Shellbags Explorer, created by Eric Zimmerman, to be the most feature shellbag artifact analysis tool available [6]. This tool provides a visual representation of Shellbags information in a directory structure layout with features to sort, filter, and examine shellbag entries obtained either from an active registry or offline hive. However, compared to our solution, Eric Zimmerman's "Shellbags Explorer" lacks in features such as frequency analysis with heat map, global events filtering, and advanced reporting.

III. Our Proposed Approach

In our proposed approach, we provide the digital forensic investigator a tool, called SeeShells, which will help to leverage the Windows registry in gathering evidential events from the registry keys, also referred to as Shellbags. Unlike other tools, SeeShells will be able to create the big picture and allow the investigator to zoom in or out of timelines with the assistance of our frequency mapping feature. Other basic criteria for SeeShells are as follows:

- Can be used on multiple versions of Windows Operating Systems.
- Can run on both live systems and offline hives.
- Requires no installation to ensure that limited artifacts are left on live systems during an investigation.
- Provide an interface that is intuitive and efficient for the forensic investigator.
- Provide features that will help the investigator quickly find and report on his findings.

SeeShells can support Windows 7 through Windows 11, which is presently the most current version of the Windows Operating System. The events captured on these machines will include logging in/out, powering on/off, deletions, downloads, folder access to various document types, and insertion or removal of USB Drives. This information, referred to as shellbag artifacts, can be extracted either on a running Windows system (e.g., live analysis) or the user's specific registry file taken from the machine in question. Once the shellbag artifacts are extracted, each user event can be displayed in a rich Graphical User Interface (GUI). These events can also be filtered and sorted by the event's date, name, type, and user by employing intuitive controls. Finally, a report of the findings can be exported in the following formats, comma-separated values (CSV), hypertext transfer protocol (HTTP), and portable document format (PDF).

In our application design, it was essential that the program executes from a single standalone file. This provides a forensic investigator the flexibility to perform a live analysis on the suspect's computer without the need of installing any software. In this scenario, the forensic investigator must only copy this executable file onto his or her removable media (e.g., USB Stick, External Hard or Flash Drive). Once the file is on the removable media, it can be connected to the suspects computer, and directly executed. Although this approach may create additional digital artifacts, they can be excluded from the investigation by properly documenting the process. These scenarios typically occur in situations when a machine cannot be shutdown, imaged, or logged off because of the threat of losing any evidence is predominant. On the flip side, if the investigator is working with a post-mortem scenario, he or she can execute the same program on their local machine and then open the suspect's registry file containing the Shellbags.

The first step when using SeeShells is to acquire shellbag data. This can be done from the active registry or from an offline registry hive as shown in Fig 1.

Fig. 1. Initial program's screen which allows the user to select either from the machine's active registry or from acquired registry file.

With the acquisition complete, users see the following program window called the SeeShells Inspector and is shown in Fig 2. The SeeShells Inspector will consist of seven windowpanes, also referred to as widgets, and have the following functions:

- Shell Inspector – full description of selected event's time, user, location, and path
- Hex Viewer – examine the raw constituents of an item selected to help provide insight
- User Actions Frequency – graph to help visualize the frequency hot spots of activities
- User Actions Frequency Selector – help to select a window for the user actions frequency pane
- Item Event List – event time and description
- Registry – presents all the data retrieved in a hierarchical view to help explore file system
- Global Event Filters – filter events on type, path, user, registry hive, begin and end date
- Exporting – allows users to export Shellbags in a report formation.

Fig. 2. SeeShells Inspector.

Additionally, the toolbar menu will provide the investigator the option to Import, Export, or Reset. The import feature allows the investigator to retrieve registry hive file or live registry. This comes in handy when the investigator is working with multiple user registry files. When the analysis is finished and the investigator has identified the information needed for the case, he or she can simply select the export option on the toolbar of the application to create a highly customizable report. The following reports modules can be used by the investigator:

- Captioned HeatMap - inserts a heatmap with a text editor to the side allowing user to add personalized text or notes
- Captioned Histogram - inserts a histogram with a text editor to the side
- HeatMap and Histogram - inserts a side-by-side image of a heatmap and a histogram
- Header - inserts a textbox that includes a default header
- HeatMap - inserts a heatmap
- Overview - inserts a pie graph with shellbag event types as percentages
- TextBox - inserts a textbox that includes a default header
- ShellEvent Table - inserts a shellbag event table filled with Shellbags
- Timeline Histogram - inserts a timeline

All these views can be interacted with from this menu, to only show the specifics that the user wants to display. Events can be pre-filtered before exporting as a report. Finally, the report can then be saved as a PDF or XPS, sent directly to printer, or sent to OneNote.

IV. Implementation

A. *Finding Shellbags Data*

There are two locations within the Windows registry where shellbag data can be found. The keys are HKEY_CURRENT_USER\SOFTWARE\Classes\Local Settings\Software\Microsoft\Windows\Shell and HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows \Shell. Within these each of these two subkeys are two more subkeys named Bags and BagMRU. The BagMRU subkey contains the name of folders and their respective paths. The base BagMRU represents the desktop. The folders under BagMRU represent the disk hierarchy. If one examines the shellbag data, most of it is binary data. For this reason, simple examination of Shellbags is difficult and need software to convert to usable information. Three such programs are Shellbags Explorer, SeeShells, and Shellbags. The authors of this paper were directly involved with the development of SeeShells, so this tool will be examined.

B. *Gathering Shellbags*

The technique of gathering all Shellbags is recursive, like traversing a disk hierarchy. Table II shows pseudocode for gathering Shellbags from a subkey, either Bags or BagMRU.

TABLE II. Pseudocode for Gathering Shellbags Information Shellbags

```
gatherShellbagData(currentSubkey)
  currentList = (SystemCall)GetSubkeysAndValues(currentSubkey)
  foreach datapoint in currentList
```

```
if dataPoint is value then store in list
if dataPoint is subkey then gatherShellbagData(dataPoint)
```

C. Parsing Shellbags Data

Each shellbag contains standalone data that describes aspects of the system such as the system drive. There are two things that make parsing Shellbags difficult. The first is that they are binary, so to access data at certain offsets requires programming techniques such as pointers in C or "BitConverter" in C#. Many non-shellbag registry entries are text or numeric, and thus much easier to read. The second difficulty is that, except for the first two bytes, the data structures are different for each of the shellbag types. There is little uniformity in the data structure for each shellbag type.

As stated, the first two bytes contain the same data for all types. This is the only consistency in shellbag data. These values are a big-endian word that represents the size of the shellbag. For some Shellbags, the third byte indicates the shellbag type. For other Shellbags, bytes 6 through 9 contain a signature identifying the shellbag type. In either case, there is a way to get the shellbag type from the binary data.

It should be noted that all Shellbags have associated date and time date values. Since these registry entries have a date/time stamp that indicates when the registry entry was last written, all Shellbags inherit such data from the registry itself. When collecting shellbag data this information should be saved because it might contain valuable and needed information. Within the shellbag data there is at least one additional date and time. What internal date/times represent depends on the shellbag type. For instance, for a URI (Uniform Resource Identifier) shellbag there is a date/time value the indicates a connection date/time and can be found at offset 14 within the binary data.

Some Shellbags have text data, usually in Unicode format, that represent information such as paths of URIs. When visually examining shellbag data these strings can be read. As a note of detail, they are all null terminated. That is, the string continues until a zero is encountered. And if there is a string in the binary data, somewhere there will be a collection of bit values indicating specifications such as whether a shellbag uses Unicode strings or not.

To explain how parsing works in a more cohesive way, an example illustrating shellbag parsing follows. The process starts with a block of binary shellbag data. Start by retrieving the shellbag size from the first two bytes of the binary data. Next, examine the third byte. Suppose the process encounters a shellbag with the hex value 0x10 in the first byte of the binary data, that is a recognized type. Note that here can be other bits in this first byte, but the determining bit is 0x10. To filter other bits that might have meaning later in the parsing process, the following operation is used:

- if thirdByte & 0x70 == 0x10 then type = "Root Folder"

Now the data size has been retrieved and the shellbag type identified. The next 16 bytes contain the Globally Unique Identifier (GUID), which concretely identifies this shellbag. These GUIDs are well known and can identify the subtype as belonging to a Network, Program File, Document, and many other types. The program described later in this paper has a built-in table with the know GUIDs and their subtypes, so matching a GUID to a subtype is easy.

Near the end of the data another GUID exists indicating a shellbag extension. This may or may not exist, but if so can be read and recorded so that the process knows that there is an extension to this shellbag. Recording the shellbag last write date and slot modified data should be done regardless of the shellbag type. Other types may have a path or file name information. Desktop folders will have location data.

V. Evaluation Based on a Case Study

To demonstrate how SeeShells can provide an effectively rich interface for finding shellbag information, a situation, analysis, and results are presented below (a case study is provided on our GitHub page with the following URL https://github.com/eamoruso/SeeShells).

A. Situation

Assuming the present day is 03/15/2021, while working as a Digital Forensics and Incident Response (DFIR) analyst, you are investigating an insider threat of Intellectual Property (IP) theft case. The company, Tehsla, said their own Cyber Threat Intelligence department found that a person or group was selling a folder on the dark web with intellectual property inside the folder. The forum post selling the information was posted at 9:34 PM on 03/08/2021. The Threat Intelligence team couldn't verify what exactly was being sold inside the folder, but they believe the claim is legitimate and only people working within the company could have accessed any confidential company information. Therefore, the company's security department believes they have identified a suspect. However, the company does not have definitive proof that this employee was the one who did it, so they hired you to help. They were able to get the suspected employee's computer and registry information - are you able to find any solid evidence and gather information on what exactly was stolen?

B. Analysis

Using SeeShells and opening the registry file provided by Tehsla security department, one of the first things that can be seen is the large timeline of the events spanning from 2019 to March 2021 shown in Fig 3.

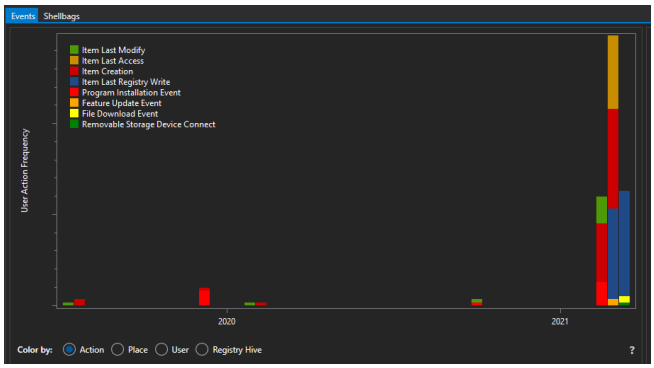

Fig. 3. SeeShells showing one of the first events in 2019.

One thing that can be done to reduce the number of events shown, is to filter out some events using the SeeShells Global Events Filter. One of the key details about the investigation is the timeline of incidents. The company's Cyber Threat Intelligence team said the post was put up on 03/08/2021. Showing activity from a week before the incident date could show a list of events that led up to it.

Within SeeShells, you can edit the Start Date and End Date fields to only show events within that time frame. For this, I set the Start Date on 03/01/2021 and End Date on 03/08/2021 as shown in Fig 4.

Fig. 4. SeeShells using a Start and End Date in the Global Events Filter.

From the situation description, the company was not able to figure out what specific confidential information is found, so currently it is not possible to filter by event name. Looking around at the folder names could show what could potentially be intellectual property (IP). IP is any information, property, or asset that the company owns which is prohibited from outside use or distribution.

From the directory names, we can figure out the company, industry, and potential IP items. The following are directory names that were found that are indicative of the industry:

- Self_Driving_Code
- 2020CarDesigns
- SelfDrivingCompCode
- ElectricMotorBlueprint

We see that the employee had access to those files and was able to modify them, shown in Fig 5. Though so far there's no evidence that the employee took them from his or her work computer.

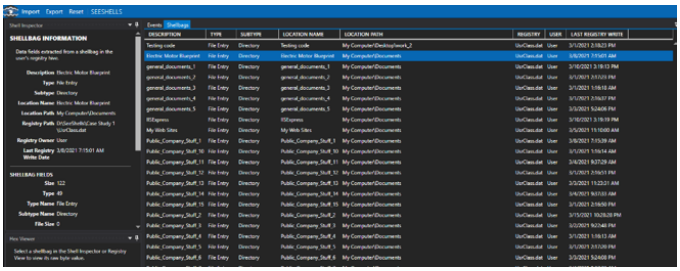

Fig. 5. Finding one of the folders that can potentially be IP.

This company is an electric vehicle company that also has specialization in self driving technology which is highly valued. Though it is worth noting that there are other directories within the environment and not all are suspicious files, such as general folders like Tehsla_Documents_1. Also, since the suspect was an internal employee, he or she is allowed legitimate access to those internal documents and so far, there's no evidence they have taken anything outside the company's work environment. By continuing to walk up the dates there is interesting activity found the day before the IP is posted online for sale on the deep web (03/08/2021) as shown in Fig 6.

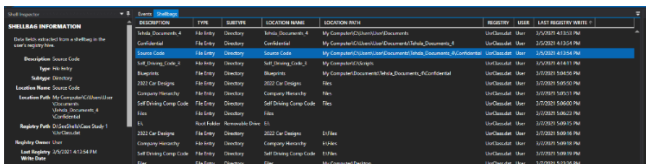

Fig. 6. Showing another interesting directory within the timeframe.

On March 7th, it is observed that the employee viewed several directories within the folder labeled Confidential, created another folder Files, and copied directories under that Confidential folder into the new folder called "Files". This is highlighted and shown in Fig 7.

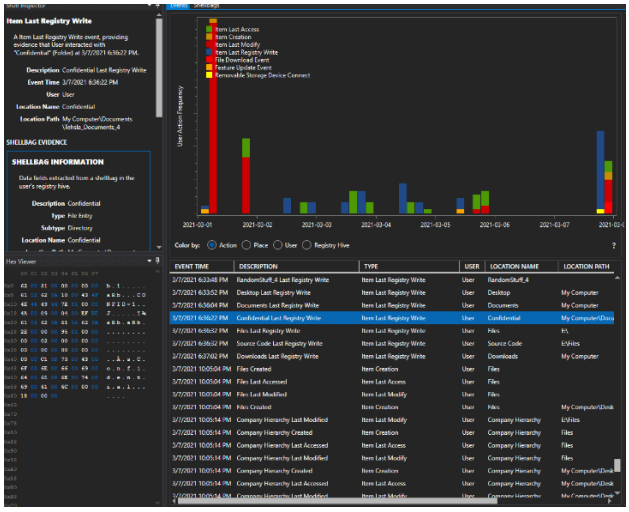

Fig. 7. Creation of a folder named Files.

Furthermore, filtering out the types of events to Removable Storage Device Connect by clicking on it, will grey out everything and show that a drive named "E:\" was connected. Clicking on it will show that it is a removable storage device that was connected in the interested timeframe and is shown in Fig 8.

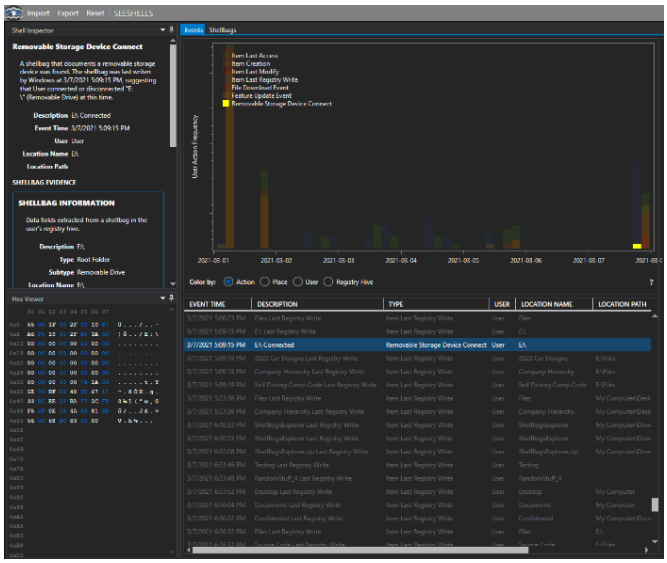

Fig. 8. Getting proof some type of external device was last plugged in before the post date with confidential files within them.

On the top right-hand side of the SeeShells explorer, the registry view will show what the filesystem looked like. We can expand "Drives" and see both the C drive (the main computer) and the E drive (the external device). We can expand on the E drive which shows a few folders, one of which is the same Files folder we found earlier. Expanding on that we see the following folders as shown in Fig 9.

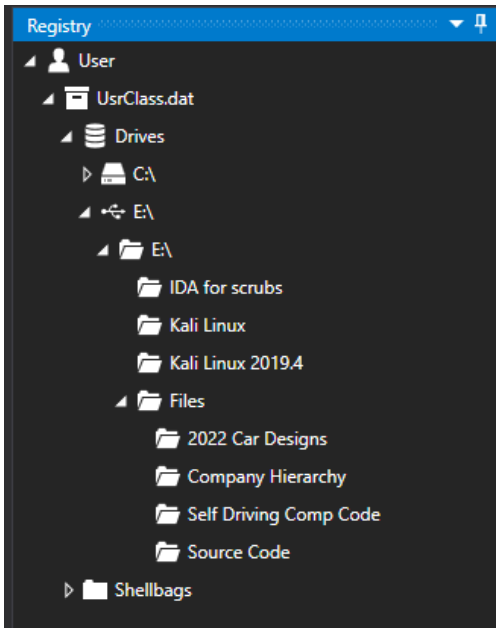

Fig. 9. The same files in that external hard drive can be found under the Confidential Folder.

C. Analysis Conclusion

The data analyzed from Shellbags in the Windows Registry clearly indicates that the employee copied several confidential files from their work computer onto some type of external storage device (E:) on 03/07/2021 at 22:09. In our case study, using SeeShells we were able to quickly find evidence that several files such as the 2022 Car Designs, Corporate Hierarchy, Self-Driving Computer Code, and Source Code were all taken from the company's computer. Even though that external device is no longer attached to the system, the Windows Registry (more specifically Shellbags) was able to log information regarding folders that had previously existed on this device. This data should be a lead to other artifact files or data within the OS to corroborate this assertion. Examples may include, but are not limited to, link or shortcut files, file system artifacts, event logs, etc.

VI. Conclusion

In this paper we demonstrated that by using our developed application, SeeShells, a digital forensic investigator can find timeline evidence more efficiently in the Windows registry, specifically Shellbags. Unlike the other applications mentioned earlier, where the investigator must search manually to piece together the evidence, SeeShells helps resolve this by providing an analyst the ability to identify anomalies and other suspicious burst of activities with our frequency data chart containing a heat map representation. Furthermore, by adding the global events filtering option, SeeShells provides the ability to screen out unwanted events reducing clutter and confusion when examining specific evidence. Finally, we used a case study to illustrate how to conduct a digital forensic investigation using our developed SeeShells application and demonstrate the easiness and effectiveness of our solution.

References

[1] Microsoft Press. 2002. *Microsoft Computer Dictionary*, Fifth Edition (5th. ed.). Microsoft Press, USA.

[2] Carvey, H., 2011. Windows registry forensics: Advanced digital forensic analysis of the windows registry. Elsevier.

[3] 1. Zhu, Yuandong, Pavel Gladyshev, and Joshua James. "Using shellbag information to reconstruct user activities." *digital investigation* 6 (2009): S69-S77.

[4] Mize, Ryan. *Behavior of Shellbags in windows 10*. Diss. Utica College, 2018

[5] Đuranec, A., et al. "Investigating file use and knowledge with Windows 10 artifacts." 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE, 2019.

[6] "ArtiFast Windows", ARTIFAST, 2022. [Online]. Available: https://www.forensafe.com/free.html. [Accessed: Jan. 14, 2022]

[7] "Autopsy", AUTOPSY DIGITAL FORENSICS, 2022. [Online]. Available: https://www.autopsy.com/download/ [Accessed: Jan. 14, 2022]

[8] Eric Zimmerman, "Shellbags Explorer", SANS, 2022. [Online]. Available: https://www.sans.org/tools/Shellbags-explorer/. [Accessed: Jan. 18, 2022]

[9] Chad Gough, "ShellBagger", 4Discovery, 2022. [Online]. Available: https://4discovery.com/our-tools/shellbagger/. [Accessed: Jan. 18, 2022]

[10] Nir Sofer, "ShellbagsView", NirSoft 2022. [Online]. Available: https://4discovery.com/our-tools/shellbagger/. [Accessed: Jan. 18, 2022]