PRIVACY AND SECURITY OF THE WINDOWS REGISTRY

by

EDWARD L. AMORUSO
B.S. Computer Engineering, University of Central Florida, 1995
M.S. Computer Engineering, University of Central Florida, 2020

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Electrical and Computer Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2024

Major Professor: Cliff Zou

ABSTRACT

The Windows registry serves as a valuable resource for both digital forensics experts and security researchers. This information is invaluable for reconstructing a user's activity timeline, aiding forensic investigations, and revealing other sensitive information. Furthermore, this data abundance in the Windows registry can be effortlessly tapped into and compiled to form a comprehensive digital profile of the user. Within this dissertation, we've developed specialized applications to streamline the retrieval and presentation of user activities, culminating in the creation of their digital profile. The first application, named "SeeShells," using the Windows registry shellbags, offers investigators an accessible tool for scrutinizing and generating event timelines based on specific criteria like file access patterns and system navigations. It boasts analytical features that can identify potentially suspicious events through a heat mapping system. In the context of our research, we've also crafted another application designed to collect and deduce a user's extensive activities by solely accessing the Windows registry. This program effectively sidesteps security software by utilizing native Windows application programming interface (API) to interact with the registry, granting unrestricted access to valuable information. This trove of data, often referred to as the user's digital footprint, holds the potential to either investigate or compromise both the user's privacy and security. Finally, we propose a custom-developed application that utilizes both software-based encryption and advanced hooking techniques to protect users' personal data within the registry. Our program is designed to create a more secure and discreet environment for users, effectively fortifying it against privacy and security threats while maintaining accessibility to legitimate users and applications.

Dedicated to my beloved wife, Maria, whose unwavering support, motivation, and encouragement have been my guiding light throughout this journey. Your love has been the fuel for my perseverance.

To my precious children, Paul, Lauren, and Amelia, who bring joy and purpose to my life. Your existence inspires me to make this world a better place for all.

In loving memory of my father, Angelo, whose wisdom, and guidance have been my compass in life's obstacles and feats. His memory will forever be a beacon of wisdom, guiding me even from beyond.

ACKNOWLEDGMENTS

I would like to express my heartfelt gratitude to the following individuals and organizations who have been instrumental in the completion of this dissertation: My Advisor, Dr. Cliff Zou: Your guidance, support, and patience have been invaluable throughout this research journey. Your mentorship has been a constant source of inspiration, and I am deeply thankful for your expertise and dedication. My Committee Members, Dr. David Mohaisen, Dr. Fan Yao, Dr. Ronald Demara, Dr. Yao Li: I am grateful for your valuable insights and constructive feedback that have enriched the quality of this work. Your collective wisdom and diverse perspectives have been indispensable. In addition, the financial support provided by U.S. National Science Foundation (NSF) under Grant DGE-2325452 and University of Central Florida for making this research possible. Your investment in my work is deeply appreciated. I also want to recognize all those whose names may not appear here but have contributed in various ways to this research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Definition
ACL	Access Control List
AI	Artificial Intelligence
API	Application Programming Interface
BIOS	Basic Input Output System
DDoS	Distributed Denial of Service
DLL	Dynamic Link Library
DP	Data Protection
CAD	Computer Aided Design
CPU	Central Processing Unit
CSV	Comma-Separated Values
IP	Internet Protocol or Intellectual Property
ML	Machine Learning
MRU	Most Recently Used
SID	Security Identifier
OS	Operating System
UAC	User Account Control

CHAPTER 1: INTRODUCTION

The cybersecurity threat landscape is rapidly evolving, with web and email phishing campaigns posing significant risks to organizations. Phishing attacks, a form of social engineering, involves a malicious actor sending seemingly legitimate messages (e.g., emails, texts, or voice messages) to mislead recipients into revealing sensitive information, such as login credentials, or clicking on malicious web links to run nefarious applications [34]. Phishing attacks can be highly effective in deceiving individuals and organizations, leading to financial losses and reputational damage. According to Statista, a global intelligence platform, phishing attacks have seen a significant surge in recent years, solidifying their position as the most prevalent cybercrime [35]. This underscores the growing demand for digital forensic investigators to combat cybercrimes and highlights the increased likelihood of users encountering various types of system breaches.

However, another area that has received relatively little attention is the Windows registry. The sheer amount of information the registry contains, including details on installed software, recently opened files, timestamps, and system configurations, makes it an invaluable tool for reconstructing a user's activity timeline and uncovering sensitive information. Unfortunately, there are currently limited tools available to assist digital forensic investigators in optimizing their use of the registry, resulting in a lack of visibility into this critical facet. Furthermore, there is a lack of research addressing the privacy and security implications of the Windows registry, which has the potential to divulge significant amounts of information, encompassing both cyber-related and non-cyber aspects. For instance, a specially designed methodology can effortlessly collect sufficient registry data, allowing for the inference of a user's complete set of activities, commonly referred to as their digital footprint.

Another notable aspect of the Windows registry is that it allows most of its stored information to be readily accessible to both users and running applications. This accessibility is a result of its original

design as an open, centralized database intended for developers to utilize in storing their application configuration settings. The registry's default lack of encryption exposes its contents either in plain text or binary format, simplifying the task of reading and modifying registry entries for users, applications, and potentially malicious actors. Additionally, the operating system heavily relies on the registry to store all its settings and preferences, and it provides developers with standard APIs that facilitate convenient read and write access to the registry. While the ease of access to the Windows registry has its advantages for legitimate users and developers, it can also be a potential security risk if misused by unauthorized users or malicious applications.

In response to the escalating cybersecurity challenges, our research is driven by the objective of enhancing the digital forensic community's capacity to efficiently examine registry data. Our efforts involve devising methods for rapid traversal of the Windows registry and the extraction of both personal and security-related data. Furthermore, we present a methodology aimed at introducing a novel approach to safeguarding sensitive information stored within the Windows registry. Through our approaches and custom-built software, we empower the analysis of user behavior, the exposure of an individual's digital footprint, and the protection of sensitive information residing within the Windows Registry.

In Chapter 2, we present "SeeShells," an innovative application that harnesses the power of the Windows registry to offer investigators a robust toolkit for analyzing and constructing event timelines. This tool is particularly adept at pinpointing user behavior, including file access patterns and system navigation, through customizable criteria. Additionally, SeeShells boasts advanced analytical features that employ a heat mapping system, enabling the detection of potentially suspicious events. Moreover, our application can export parsed timeline event information into various commonly used file formats, enhancing the value of an investigator's digital forensic report. SeeShells proves to be an indispensable

resource for digital forensics experts and security researchers, furnishing them with invaluable insights into user activities within the Windows system.

Chapter 3 delves into a meticulous methodology and a custom developed application tailored to uncover a user's extensive digital profile, also known as a digital footprint. This is solely made possible through direct access to the Windows registry. By leveraging native Windows APIs for registry interaction, we easily sidestep security software, thereby gaining unrestricted access to a wealth of valuable registry data. This information includes user credentials, installed software, recently accessed documents, as well as resources like local, remote, and removable devices. Our custom application can store this data locally or securely upload it to an external website, offering a glut of information that can either unveil user activities for investigative purposes or, conversely, pose a threat to their privacy and security.

In Chapter 4, we present our custom-developed application designed to protect sensitive user information in the Windows registry. Our program safeguards registry data by applying software-based encryption, effectively fortifying it against privacy and security threats while creating a more secure environment for users. This is achieved by leveraging the Windows native Data Protection API (DPAPI) for both encryption and decryption functionality. We follow the necessary steps to access and maintain encrypted registry data due to proprietary structure, which is managed by the operating system. Additionally, it's crucial to authenticate access to this data prior to decryption using our custom-developed application and DPAPI functions, ensuring the security of user information.

CHAPTER 2: SEESHELLS: AN OPTIMIZED SOLUTION FOR UTILIZING SHELLBAGS IN A DIGITAL FORENSIC INVESTIGATION

2.1 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 1 shows what information you can find from Shellbags, and what information you cannot find from Shellbags.

Table 1 Information found and not found in the shellbags.

Contains	Does Not Contain
Folders last interacted with	Folders created with "md" or
Location of folders	WSL (Windows Subsystem for Linux) e.g., "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 chapter 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.

2.2 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 [8]. 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.

2.3 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 Figure 1.

Figure 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 Figure 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.

Figure 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.

2.4 Implementation

2.4.1 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.

2.4.2 Gathering Shellbags

The technique of gathering all Shellbags is recursive, like traversing a disk hierarchy. Table 2 shows pseudocode for gathering Shellbags from a subkey, either Bags or BagMRU.

Table 2 Pseudocode for Gathering Shellbags Information Shellbags.

Pseudocode
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)

2.4.3 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.

2.4.4 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).

2.4.4.1 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?

2.4.4.2 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 Figure 3.

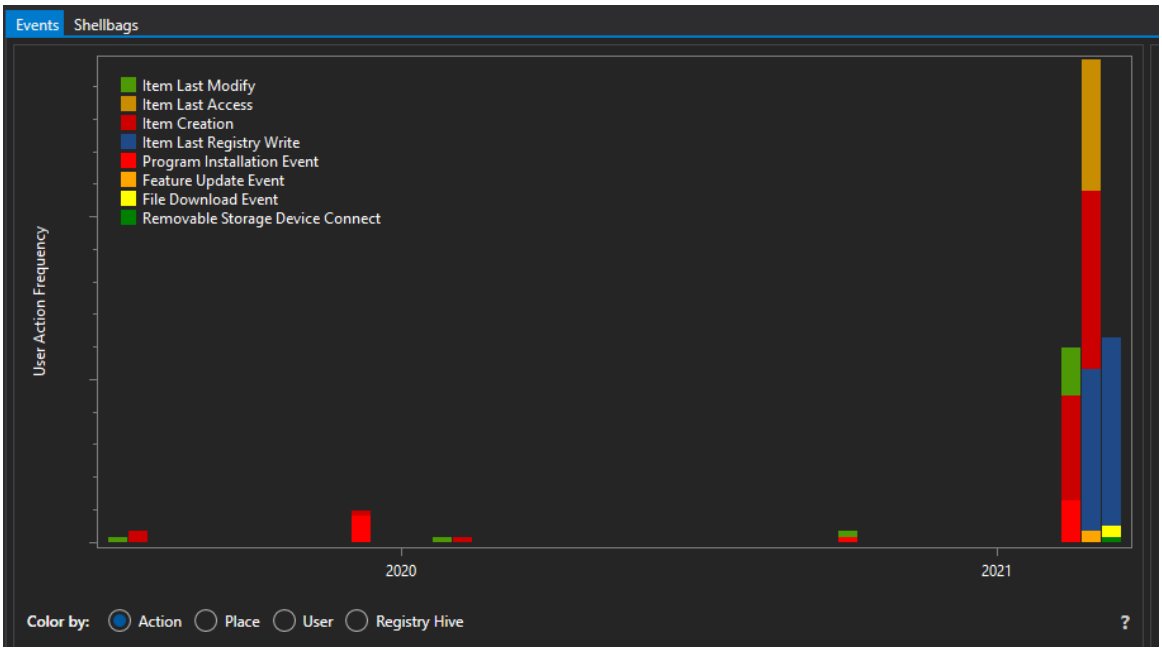


Figure 3 SeeShells showing one of the first events in 2019.

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 Figure 4.

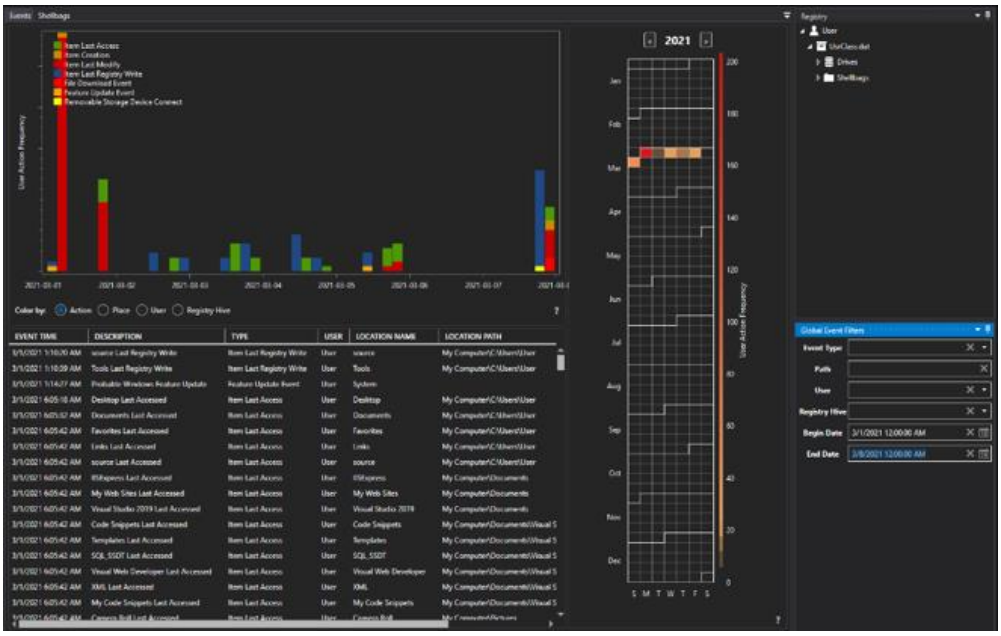


Figure 4 SeeShells using a Start and End Date in the Global Events Filter.

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 Figure 5. Though so far there's no evidence that the employee took them from his or her work computer.

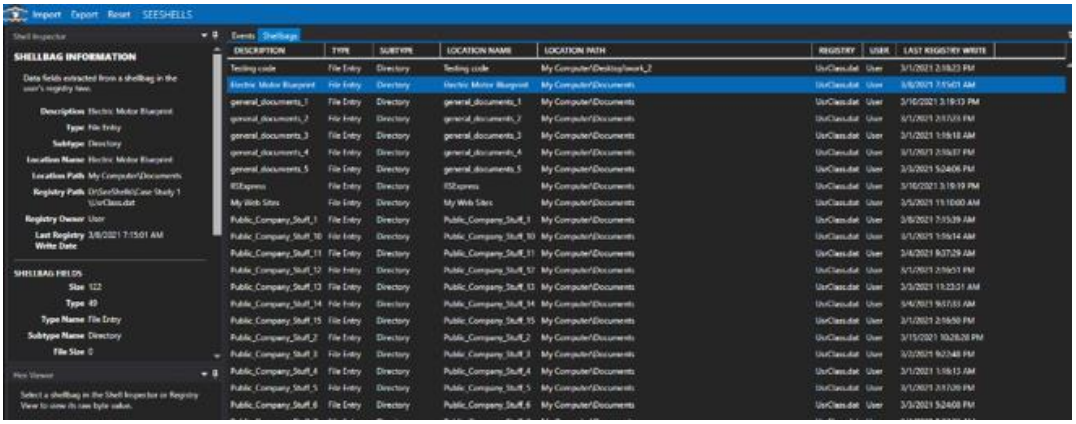


Figure 5 Finding one of the folders that can potentially be IP.

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 Figure 6.

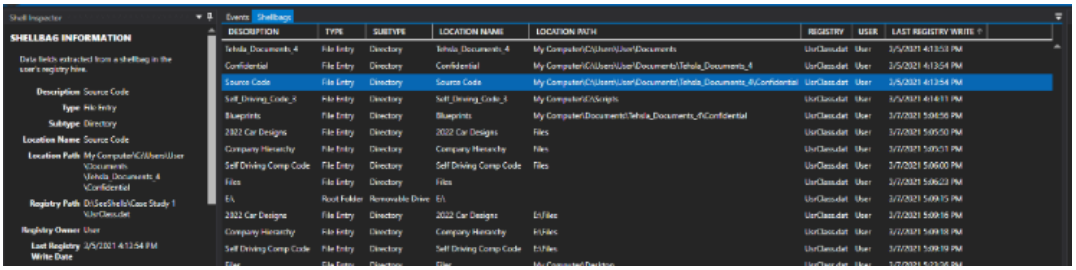


Figure 6 Showing another interesting directory within the timeframe.

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 Figure 7.

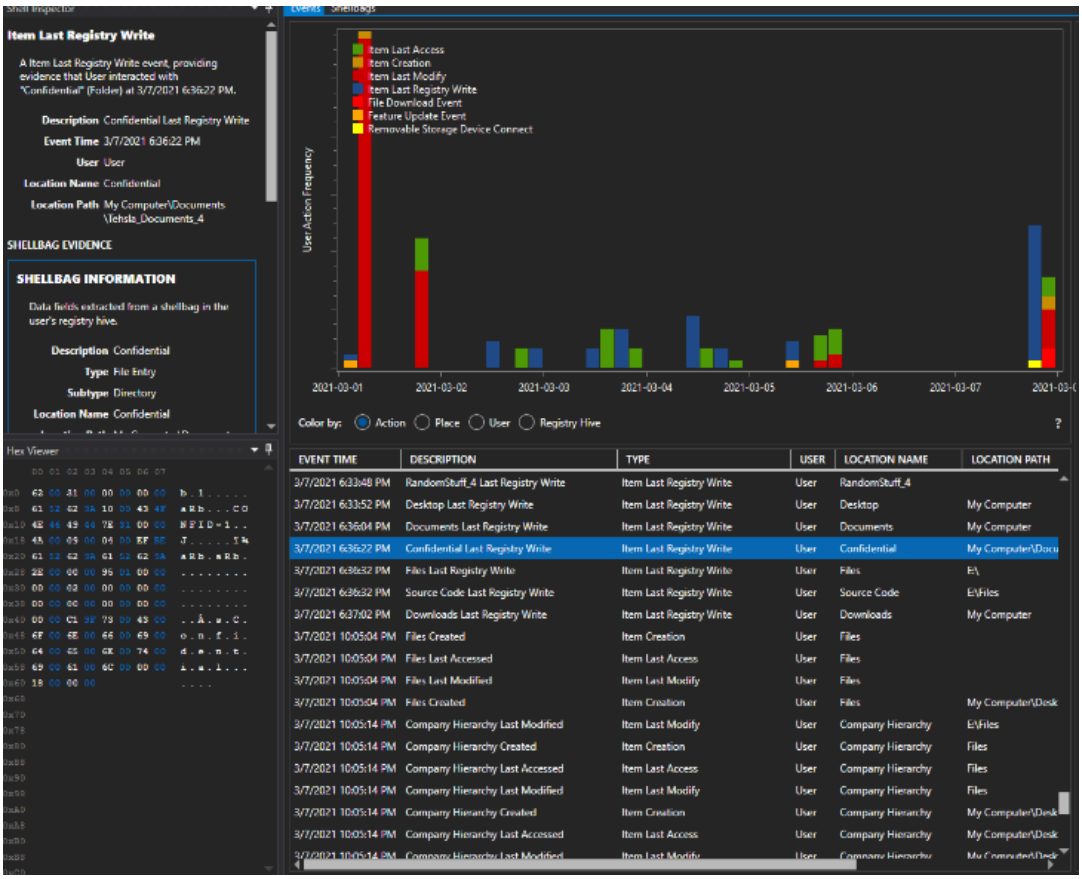


Figure 7 Creation of a folder named Files.

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 Figure 8.

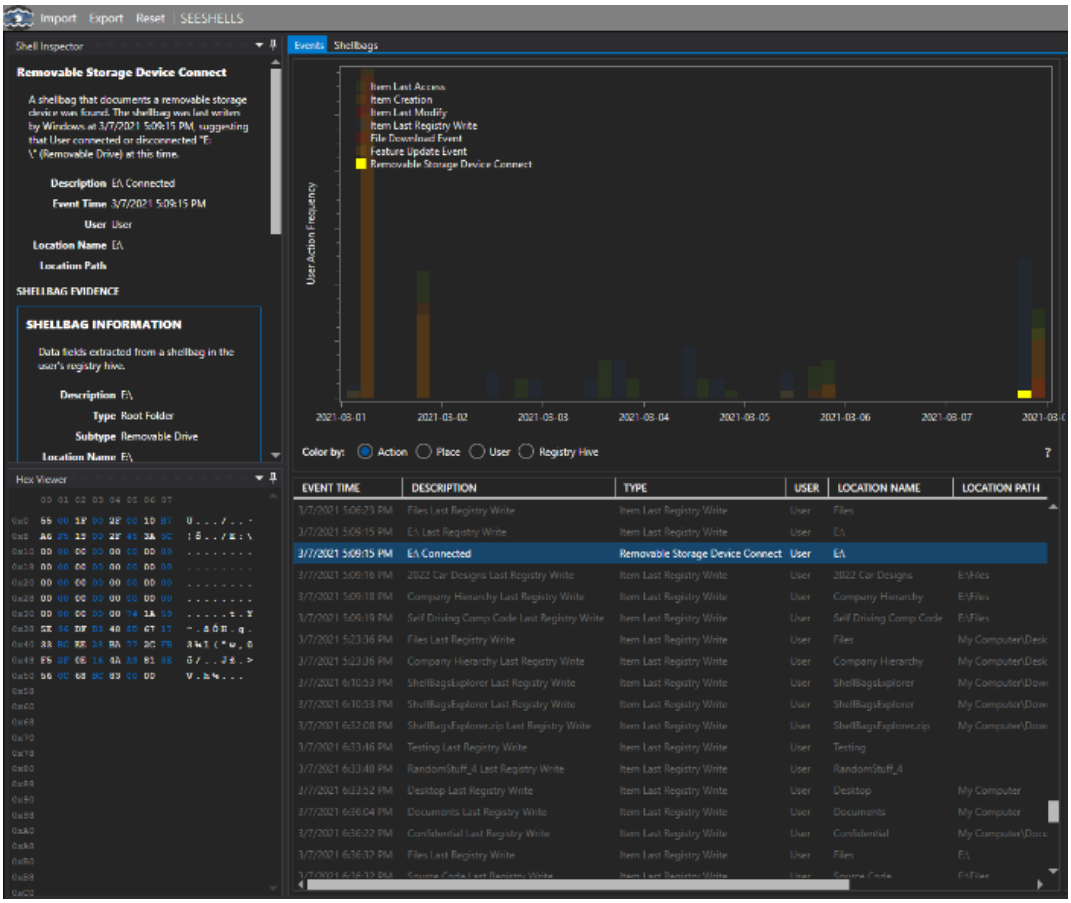


Figure 8 Getting proof some type of external device was last plugged in before the post date with confidential files within them.

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 Figure 9.

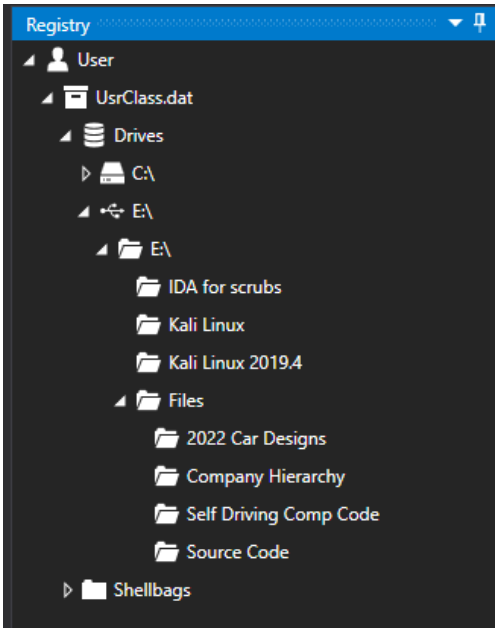


Figure 9 The same files in that external hard drive can be found under the Confidential Folder.

2.4.4.3 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.

2.5 Conclusion

In this chapter 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.

CHAPTER 3: USER PROFILING ATTACK USING WINDOWS REGISTRY DATA

3.1 Introduction

The Windows registry, which will be referred to as the registry for the remaining of this chapter, holds an overabundance of information. The registry is used by various applications, including the Windows operating system (OS) with the intended use to keep configuration data. This data is used for systemwide and per-user settings, but a fair amount of it contains system and user identifiable information. For example, items such as:

- Type of computer and hardware
- Windows operating system and version number
- Computer system's current time zone
- Applications installed and available to users
- History of recently accessed files
- Available network resources
- Network settings, such as IP address and gateway

Furthermore, many entries in the registry contain date and time information, making it possible to infer many user activities. Even non-cyber activities can be inferred such as user's working performance (e.g., is she or he actively working on company assigned tasks), user's job type (e.g., is she

an accountant or programmer based on her modified file names). Also, other insight can be deduced such as typical workday hours.

When a user executes an application, he or she will generally leave behind some type of data in the registry. For example, if opening a folder, the date and time and source location will be recorded. Having date and time information can help create a timeline of activities performed by that user. In other scenarios, such as opening a document, the action will also create an entry of the file name in the registry. This information will be referred to as the user's digital footprint. Subsequently, the term digital footprint has a broad definition, in our research, we will use the expression to identify user and system information acquired throughout the registry for building a user profile.

The registry can be accessed by various techniques and applications. For example, an attacker or common user can run the Window's built-in command-line utility called "reg.exe" to extract data from the registry. Similarly, a more powerful tool, the Window's PowerShell, a cross-platform task automation solution, can be used to access registry information [11]. Both mentioned programs evade security applications since they are included and available to the OS. According to Mandiant, the PowerShell is predominately used among malicious actors because it is a trusted environment allowing it to be less suspicious to identify for nefarious activities [12]. Alarmingly, as stated by RANGEFORCE, PowerShell has been deployed by top cybercriminals such as APT1, Duqu, and APT10 to gather critical intelligence to assist in sophisticated cyberattacks [13]. There are also several third-party applications available on the internet with various functionality to extract registry information [14] [15] [16] [17]. Some of these programs are typically written and intended for digital forensic investigations. Others, used by penetration testers, such as PowerShell Empire and Covenant, can also exfiltrate registry information [18] [19].

Addressing security access is a pivotal consideration during the retrieval of data from the registry. Certain registry keys necessitate elevated permissions, typically in the form of administrative access, for successful retrieval. Moreover, the Windows Operating System incorporates a built-in security mechanism known as "User Account Control" (UAC). This feature operates by default and functions to mitigate the potential impact of malicious software. It achieves this by mandating user approval through prompts whenever a program necessitates privileged system access. The primary intent behind this notification is to notify the user, thereby prompting a sense of wariness and preempting potential cyberattacks.

Nonetheless, our research underscores a remarkable finding: the acquisition of all requisite information was accomplished without setting off UAC prompts or demanding heightened privileges. This revelation bears notable significance, given that malevolent actors frequently exploit compromised accounts stemming from phishing endeavors. Typically, these ill-gained accounts lack administrative privileges, especially within environments that prioritize stringent security measures.

This chapter focuses on the collection of comprehensive information from the system registry. This is achieved by utilizing both the operating system's native application programming interface (API) and a custom-developed program. The primary aim is to construct a detailed digital footprint of a user. Leveraging the native API allows us to streamline development by building upon an existing foundation, thereby reducing the amount of code and time required for development. Additionally, employing native APIs provides an added advantage of circumventing security applications such as Windows Defender. The objective of this research is to uncover a significant real-world threat. We achieve this by meticulously extracting ample registry data, all while operating within standard privileges. The outcome of this process results in a potentially hazardous vulnerability, posing risks to both privacy and security.

The contributions of this chapter are:

- A detailed approach in gathering enough registry information to produce a user's digital footprint that can expose the user to privacy and security breaches.
- Provide a ready-to-use application capable of seamless execution, bypassing any potential conflicts with the operating system or security software to easily create the digital footprint of the user's account on the target Windows system.
- Conduct experiments using real-world registry data collected from multiple businesses, illustrating the extent to which a user's digital footprint can be derived solely from registry data.

3.2 Related Work

Presently, there are many solutions available to help acquire a user's information from his or her computer. Some are automated scripts or programs developed to extract various data [20] [21] [22], others are manual techniques using built-in OS applications [23] [24]. In our research, we aim to leverage a technique and data repository to easily acquire a user's digital footprint. With this in mind, we focus on a single source, Windows registry, for gathering a user's profile. Furthermore, we explore methods for which the information can be automatically gathered with no resistance from security software.

Throughout our assessment, we examined many tools and practices documented by educational and other entities. Most provided a process to acquire evidence from the registry for digital forensic analysis [23] [24]. Also, several resources offered techniques and cheat sheets to extract registry information manually [25] [26]. Other research use machine learning (ML) and artificial intelligence (AI)

in creating a user's digital footprint, associated to performance analysis and anomalous activity detection [27].

While the previously mentioned approaches show promise, their primary limitation lies in neglecting the automated extraction of registry data to establish a user's digital footprint, thereby making them only feasible in theory but too complicated and time-consuming to be utilized by most people. Taking our research one step further, we unveil the actual vulnerability, highlighting how attackers can effortlessly amass a user's digital footprint from their computer by simply accessing registry data. This process is made even smoother by utilizing established Windows APIs, sidestepping potential obstacles posed by system constraints and security software.

3.3 Our Proposed Approach

In this section we describe how our custom application extracts registry data, without alerting security software, to construct a user's digital footprint. Our goal is to rapidly extract the necessary values (e.g., usernames, applications, hardware, accessed resources, time and date stamps) through predefined registry key locations. This eliminates the need to traverse the entire registry, improving performance and execution time. For our prototype, all results needed for a user's digital footprint are saved to a locally created folder. Each file contains information essential for either compromising or inferring the following:

- User's computer environment, whether it is virtual or bare metal machine, and a high-performance device.

- User’s computer usage patterns, such as access to most recently used (MRU) files (e.g., Word, Excel, PowerPoint).
- Repetitions on folders and devices accessed by the user.
- A timeline of a user’s cyber activities, including program names and access time, file names and access/modification time, etc. Such a comprehensive timeline of cyber activities would enable us to infer user’s private and non-cyber activities, such as work hours, lunch breaks, job type (e.g., an accountant or programmer) or time-taken-off by the user.
- Sensitive data, files identified by their corresponding names and locations (e.g., payroll, prototype design, sale commissions).
- Additional data extracted, like the user's device type, computer name, and installed software, can be utilized to infer the individual's occupational category. To illustrate, consider a user managing a system referred to as "sys-CAD-cpu01," which includes drafting applications; this scenario might indicate that the user holds a profession related to computer-aided design (CAD).
- The security posture of the user's system can be assessed by categorizing both startup and installed programs. This classification could potentially aid an attacker in recognizing security applications, allowing them to either bypass or deactivate security software.

3.3.1 Digital Footprint Privacy

In this research, we are not concerned about discovering the user’s identity, instead we compile a profile to understand his or her responsibilities and activities, as they relate closely to privacy. The

captured behaviors in turn will help assemble a timeline, exposing a user's cyber and non-cyber routines and processes. This also includes other information such as frequently accessed resources (e.g., hard drive, network, external devices), to help infer the user's responsibilities.

3.3.2 Digital Footprint Security

We establish that the digital traces generated through registry data have the potential to compromise the security of the user's system and the privacy of the user. The information collected includes the locations of files, usernames, hardware specifications, and installed software. This type of information exposure can make the user susceptible to additional attacks or vulnerabilities. Consider the following scenario: having access to information about the programs within a system could potentially become a weak point for malicious actors to exploit. This entails identifying vulnerabilities within these applications, which could then be used to infiltrate the system even more profoundly.

To illustrate further, let's take an example involving intellectual property (IP). A criminal could strategically focus on stealing a user's documents if they possess pieces of information of the IP. Lastly, disclosing hardware specifics to a malicious actor could reveal whether the device is powerful enough to execute a desired assault, such as a distributed denial of service attack (DDoS).

3.3.3 Retrieving Registry Key Data

Numerous methods exist for extracting data from the registry. For someone new to computer usage, the integrated Windows command-line tools, such as reg.exe, can be employed to extract either

individual or multiple values. This method necessitates that the user is familiar with each key necessary for constructing a user's profile. Consequently, they must undertake the laborious task of manually gathering and parsing data to piece together the user's digital footprint. In essence, current approaches demand that an attacker possesses extensive knowledge of the registry and engages in intensive operations to profile users on a targeted Windows machine.

In our approach, using C++ programming language and Windows built-in APIs, we automate the parsing of specific registry key data and assemble them into meaningful information. This is possible with extensive examination and research of the registry; we collect the requisites for building a user's digital footprint. Other insight on the registry was obtained from several digital forensic cheat sheets [25] [26].

The purpose of our application is to efficiently arrange extracted registry details, facilitating coherent data access and identification. Once our specialized program processes the information, it generates a designated folder containing appropriately categorized files based on their types. In our initial demonstration, we opted to store these files directly on the targeted device. However, it's worth noting that this functionality could readily be adapted to enable uploading to a website or an internet server if desired.

3.3.4 Storing Captured Registry Data

In our prototype, we generate six distinct files, each encompassing particular pieces of information that we've collected from the registry. To enable seamless integration into external applications such as Excel, Google Sheets, and Calc, we structure the data in each file using the comma-

separated values (CSV) format. This formatting choice offers numerous advantages, primarily facilitating the transfer of data from one application to another. One particularly noteworthy benefit, as highlighted in subsequent sections of this paper, is its capacity to facilitate the creation of visual representations like timeline graphs through tools like Excel.

Of the six files, three of the files contain different Office applications' information to help build a list of most recently used (MRU) documents, spreadsheets, and presentations by the user. The fourth file is created to store general information such as system hardware, network settings, installed software, and user credentials. Finally, the last two files store date and time stamps of user activities and all user accounts on the target system.

3.3.5 Avoiding Security Software

To evade security software, we use built-in Windows APIs to retrieve our data from registry keys. These registry keys require no additional privileges for admittance, making it seamless for access by any regular user account. Equally, using native APIs, we can avoid system monitoring software, a concept referred to as living off the land. According to [28], this is considered a tactic used by malicious actors to evade detection and blend in as normal activities. Both methods allow us to gather information unnoticed and efficiently from the registry.

3.4 Implementation

The Windows registry is a specialized database that the Windows operating system maintains. For instance, if you install Microsoft Office, the registry contains data paths, plugins, language information, and anything else the program needs to function with. While the amount of information that an application such as Office saves to the registry is large, most other applications at the same time also install their own configuration data. Interestingly, the registry has become a "De Facto" for application developers to use to store their program settings. To illustrate the importance of registry data, if the registry gets corrupted, the Windows operating system will not function correctly.

This paper is intended to show how user information can be harvested to create a user's digital footprint based on registry only. A great deal of useful information can be found in the registry. For the purposes of this paper, we will categorize what we examine as basic user data, general hardware and software information, the file usage history for users, and recent document handling. To research this, we create an application that examines the Windows registry and extracts the relevant artifacts. In the next several sections these will be explained. Our developed application code is publicly available on GitHub at "https://github.com/eamoruso/UserProfileAttack" link.

When our custom-developed application runs, a folder named **mm-dd-yyyy** is created, which is named with the current date. For example, if we run our program on February 11, 2023, the folder will be called "02-11-2023" within the same directory as our executing application. Report CSV files are saved into this folder.

3.4.1 User's Basic Information

The first step to gather basic user information is to enumerate the registry key called "**ProfileList"** located in "**SOFTWARE\Microsoft\Windows NT\CurrentVersion\"**. Each key represents a user's security ID. The data contained within each key has relevant information such as ProfileImagePath, Flags, and Security Identifier (SID). Together, this information can positively identify a user to the operating system. For the purposes of this paper, the ProfileImagePath and SID are saved to a CSV within the newly created folder named Basic.CSV.

One other important detail that can be gathered from here is a list of software that each user operates. This can narrow down what program is used by each user and when he or she installed it. In many cases we can infer with high confidence what type of job the user has in the business. For example, if several accounting applications (e.g., QuickBooks, Sage, NetSuite) are listed, one can infer that the user is responsible for bookkeeping or a finance related job function. Figure 10 shows the contents of an example Basic.CSV file.

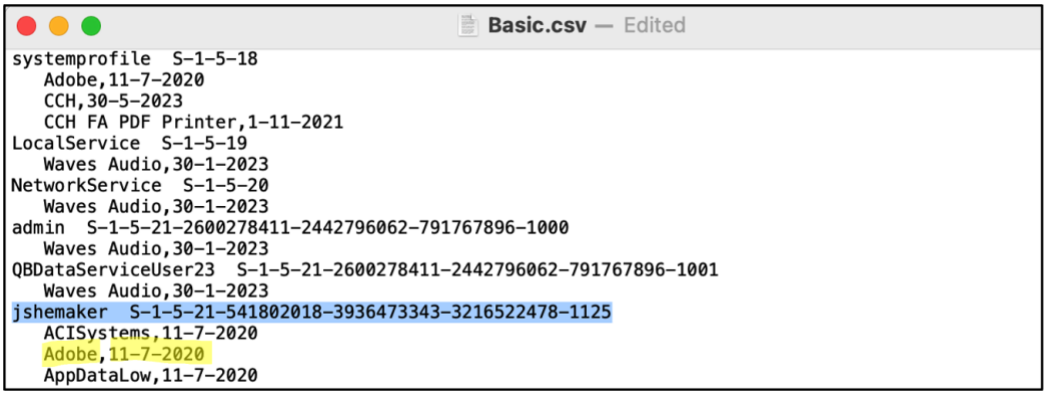


Figure 10 Basic User Information (ProfileList) file contains all available users on the system. For example, "jshemaker" is the username followed by the Security Identifier (SID), which starts with "S" and ends with numbers.

3.4.2 Identifying and Retrieving Personal Data

The next resource that can be dug to extract personal information is the list of the documents, spreadsheets, and presentations that a targeted user has edited. For this paper we will gather all documents, spreadsheets, and presentations created or modified with Microsoft Word, Excel, and PowerPoint. Although, there are other such programs (e.g., LibreOffice, Apache OpenOffice) capable of also creating these types of files, we only focus on Microsoft Office in this paper, which can be easily expanded to cover other types of documents and programs in the future. There are several factors that help solidify our choice to use Microsoft. Office applications are predominant, according to Enlyft, Microsoft Office has 45.46% Market Share [29]. Also, all the private entities willing to participate in our testing only used the Office suite. Finally, Office applications store all their most recently used (MRU) files in the registry, shown in Figure 11. Other applications, such as OpenOffice, store few settings, but nothing related to the opening of any documents, spreadsheets, or presentations.

It is important to note that our custom designed application does not open all users' MRUs, only the targeted user's MRU. In other words, the application only extracts information from the user that is currently logged into the system. The target user's MRU consists of a list of file names, each can be opened and read to gain significant information about the active user. These files could also be downloaded, possibly with an automated retrieval program if the program has this target user's account access.

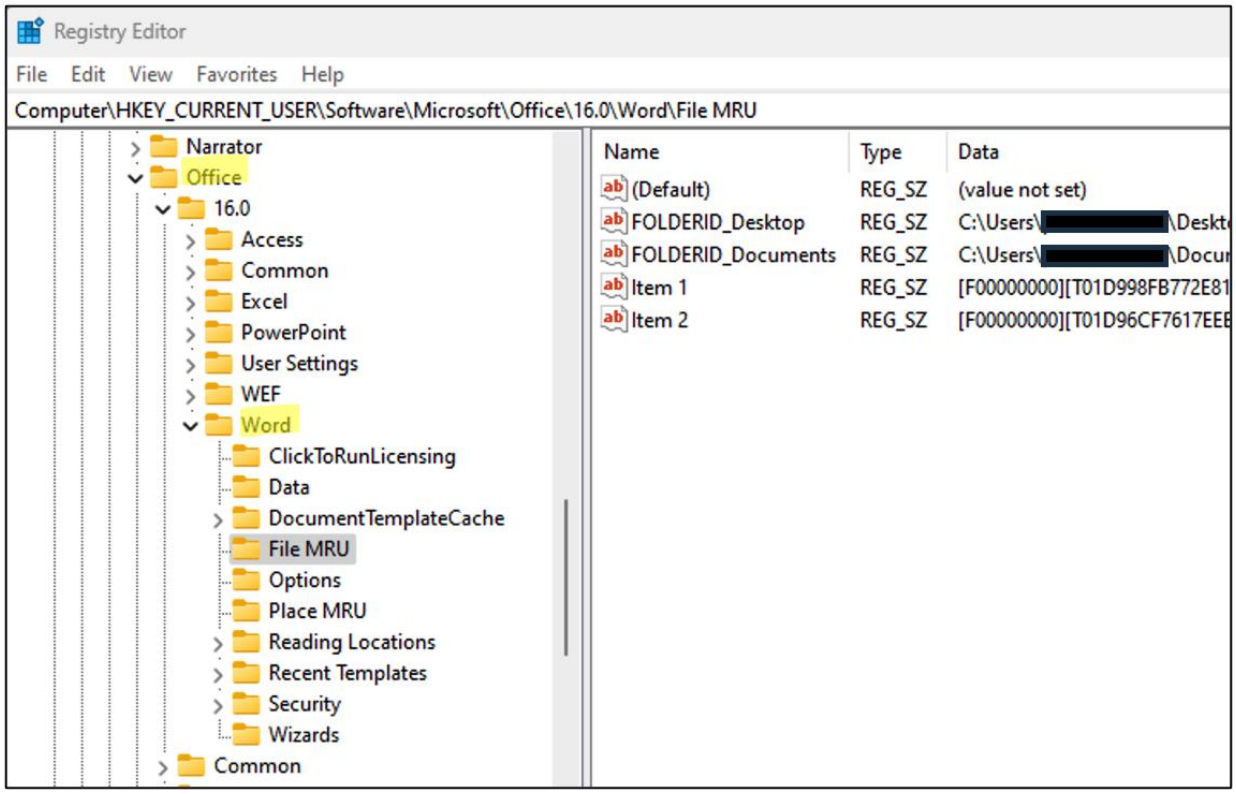


Figure 11 Excerpt of the Microsoft Office Word File MRU containing most recently used document file(s).

In gathering the user's access history to all Word and Excel files, we start with the most recently used (MRU) entries found in the registry. The key is **SOFTWARE\Microsoft\Office** followed by the version of Office that is installed. For our development system, we use the most recent available version **16.0** (e.g., *SOFTWARE**Microsoft**Office**16.0***)** which covers Office 365, 2019, and 2016 [30]. Each application's subkey is shown in Table 3. Within that key are subkeys for each of the Office applications. For this paper we limit the search to Word, Excel, and PowerPoint.

Within each of the Office and Word subkeys are further subkeys organized into two categories: files and places; represented by **\User MRU\File MRU** and **\User MRU\Place MRU**. The files' subkey lists the documents that have been edited while the places' subkey lists the locations of those files.

Table 3 Office Version 2016 thru 365 MRU Entries.

Program	Registry Subkey Location
Word	HKCU\SOFTWARE\Microsoft\Office\16.0\Word
Excel	HKCU\SOFTWARE\Microsoft\Office\16.0\Excel
Access	HKCU\SOFTWARE\Microsoft\Office\16.0\Access
Outlook	HKCU\SOFTWARE\Microsoft\Office\16.0\Outlook
PowerPoint	HKCU\SOFTWARE\Microsoft\Office\16.0\PowerPoint

The results are then saved into three files named Word.CSV, Excel.CSV, and Powerpoint.CSV. To demonstrate this, Figure 12, Figure 13, and Figure 14 show the Word, Excel, and PowerPoint CSV files with redacted, modified private information since they are extracted from real business computers.

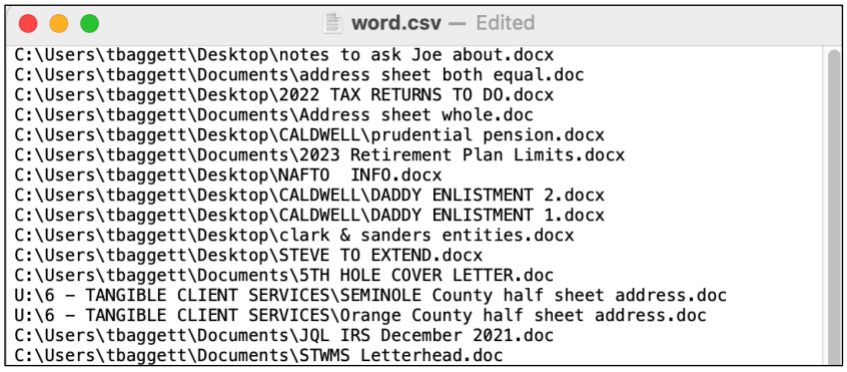


Figure 12 The Word CSV File shows a list of all most recently used Word file(s) accessed by the target user.

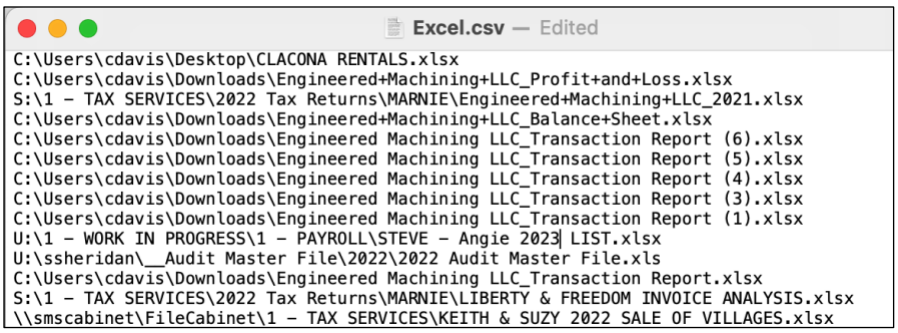


Figure 13 The Excel CSV File shows a list of all the most recently used Excel spreadsheet file(s) by the target user.

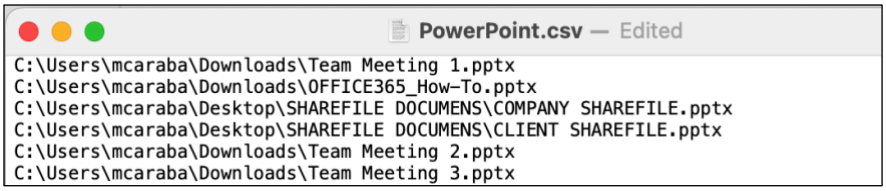


Figure 14 The PowerPoint CSV File shows a list of all recently used PowerPoint presentation file(s) accessed by the target user.

Note that Access and Outlook all have their own MRU sections. The software written for this paper can be easily amended to query these. For the program as it exists, this method is called three times with “Word,” “PowerPoint,” and “Excel” parameters. We can easily add “Access” and “Outlook” parameters to create additional extracted information files for these applications.

3.4.3 Building Timeline of Activities

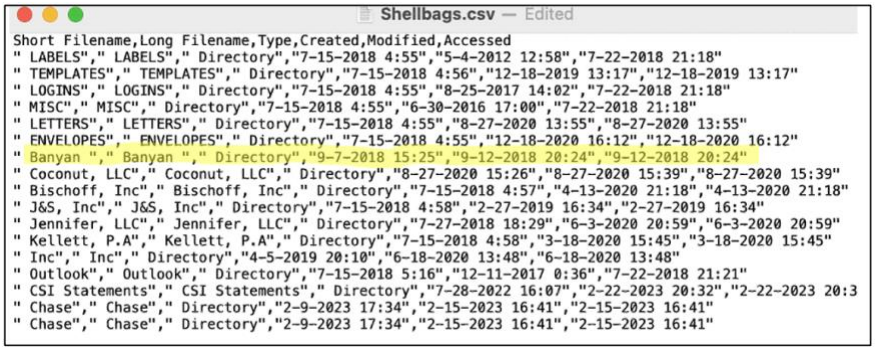
There are two registry subkeys that contain information called Shellbags. These subkeys record all folder operations that a user performs [23]. For instance, if someone resizes a window, that action will

be recorded as Shellbag information. If a folder is opened by an application, then that application name will be recorded. The valuable information contained in Shellbags is that a timeline for user navigation and usage can be tracked. The different types of information from the two registry subkeys that contain Shellbag data are both "**USRCLASS.DAT\Local Settings\Software\ Microsoft\Windows\Shell\BagMRU**" and "**USRCLASS.DAT\Local Settings\Software\Microsoft\ Windows\Shell\Bags,**" described and shown in Table 4.

Table 4 Shellbag Data.

Type of Information	Registry Subkey Location
Stores folder names and records the folder paths	USRCLASS.DAT\Local Settings\ Software\Microsoft\Windows\Shell\ BagMRU
Stores the view preferences such as the window size, location, and view mode	USRCLASS.DAT\Local Settings\ Software\Microsoft\Windows\Shell\Bags

There is an immense amount of information in these Shellbags. To produce a manageable subset for this paper, we simply examine a few months of data, since Windows may store such data for several years. An example of this file can be seen in Figure 15.


Shellbags.csv — Edited

```
Short Filename,Long Filename,Type,Created,Modified,Accessed
" LABELS"," LABELS"," Directory","7-15-2018 4:55","5-4-2012 12:58","7-22-2018 21:18"
" TEMPLATES"," TEMPLATES"," Directory","7-15-2018 4:56","12-18-2019 13:17","12-18-2019 13:17"
" LOGINS"," LOGINS"," Directory","7-15-2018 4:55","8-25-2017 14:02","7-22-2018 21:18"
" MISC"," MISC"," Directory","7-15-2018 4:55","6-30-2016 17:00","7-22-2018 21:18"
" LETTERS"," LETTERS"," Directory","7-15-2018 4:55","8-27-2020 13:55","8-27-2020 13:55"
" ENVELOPES"," ENVELOPES"," Directory","7-15-2018 4:55","12-18-2020 16:12","12-18-2020 16:12"
" Banyan "," Banyan "," Directory","9-7-2018 15:25","9-12-2018 20:24","9-12-2018 20:24"
" Coconut, LLC"," Coconut, LLC"," Directory","8-27-2020 15:26","8-27-2020 15:39","8-27-2020 15:39"
" Bischoff, Inc"," Bischoff, Inc"," Directory","7-15-2018 4:57","4-13-2020 21:18","4-13-2020 21:18"
" J&S, Inc"," J&S, Inc"," Directory","7-15-2018 4:58","2-27-2019 16:34","2-27-2019 16:34"
" Jennifer, LLC"," Jennifer, LLC"," Directory","7-27-2018 18:29","6-3-2020 20:59","6-3-2020 20:59"
" Kellett, P.A"," Kellett, P.A"," Directory","7-15-2018 4:58","3-18-2020 15:45","3-18-2020 15:45"
" Inc"," Inc"," Directory","4-5-2019 20:10","6-18-2020 13:48","6-18-2020 13:48"
" Outlook"," Outlook"," Directory","7-15-2018 5:16","12-11-2017 0:36","7-22-2018 21:21"
" CSI Statements"," CSI Statements"," Directory","7-28-2022 16:07","2-22-2023 20:32","2-22-2023 20:3
" Chase"," Chase"," Directory","2-9-2023 17:34","2-15-2023 16:41","2-15-2023 16:41"
" Chase"," Chase"," Directory","2-9-2023 17:34","2-15-2023 16:41","2-15-2023 16:41"
```

Figure 15 The Shellbags CSV File shows all folders and files accessed by a user with date and time stamps, for example, "Banyan" was last accessed December 9th, 2018, at 20:24 UTC. Also, Banyan is identified as a folder name.

3.4.4 Extracting General Information

Finally, we can also extract general information containing hardware, software, and network settings from the registry. To obtain this information, we create two separate functions, to facilitate extracting from the different registry subkeys. For example, some subkeys have specific value name/data, other subkeys just have all value name/data in the same subkey.

During the process of collecting all the data, our application stores this information to a text (TXT) file named GeneralInfo.TXT. An example of this file is shown in Figure 7.

Information such as system's manufacturer, product name, and basic input/output system (BIOS) details can all be found under the "**HARDWARE\DESCRIPTION\System\"** subkey in the registry**.** An interesting fact about this subkey is that the OS creates it during system boot and is stored entirely in memory. Although this subkey also contains a list of processors, for coding simplicity, we use the value data from "PROCESSOR_IDENTIFIER" under "**SYSTEM\CurrentControlSet\Control\SessionManager\Environment"** subkey. The processor count and architecture are also recorded under this subkey.

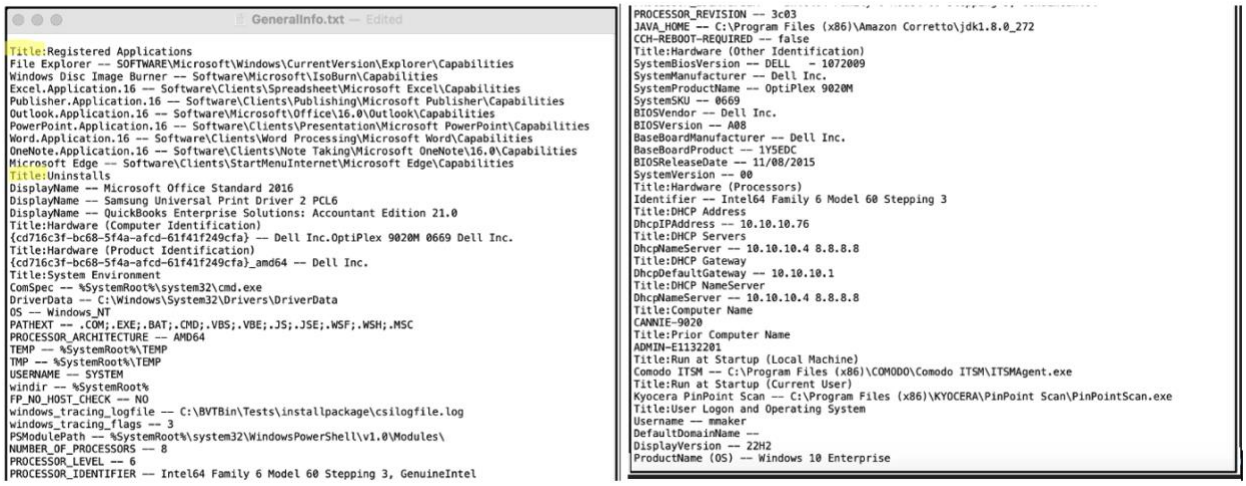


Figure 16 General Information (Hardware, Network & Software settings), this is an excerpt with many redundant values removed. The highlighted “Title” serves as a separator for each retrieved category of information.

The remaining registry data obtained for our general information file is:

- Operating System and version information
- Username login credential
- User registered applications
- All Installed programs and patches (uninstall)
- Internet Protocol (IP) addresses
- Assigned networking information
- Computer name and previous name
- Programs scheduled to launch at startup

All this information is retrieved from specific subkey locations and value names as shown in Table 5.

Table 5 Subkey and Value Name for General Information.

Registry Subkey Location	Value Name(s)
HARDWARE\DESCRIPTION\System\BIOS	SystemManufacturer SystemProductName BIOSReleaseDate BIOSVendor BIOSVersion
SYSTEM\CurrentControlSet\Control\Session Manager\Environment	NUMBER_OF_PROCESSORS PROCESSOR_ARCHITECTURE PROCESSOR_IDENTIFIER
SOFTWARE\Microsoft\Windows NT\CurrentVersion	ProductName DisplayVersion
SOFTWARE\RegisteredApplications	{All Value Names in this Subkey} (e.g., File Explorer, Paint, Notepad)
SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall	DisplayName
SOFTWARE\Microsoft\Windows\CurrentVersion\Run	{All Value Names in this Subkey} (e.g., SecurityHealth)
SYSTEM\CurrentControlSet\Control\ComputerName\ComputerName	ComputerName
SOFTWARE\Microsoft\Windows\CurrentVersion\Authentication\LogonUI	LastLoggedOnSAMUser LastLoggedOnUser IdleTime
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Interfaces\	DhcpIPAddress DhcpSubnetMask DhcpDefaultGateway DhcpServer DhcpNameServer

3.5 Evaluation based on Real-World Data

In our evaluation, we demonstrate how our custom developed application effectively creates a user's digital footprint by exclusively retrieving registry data from target systems. To achieve realistic results for our assessment, private data is extracted from actual user accounts in several real-world collaborated companies. To corroborate our evaluation, diverse organizations in the fields of accounting, law, and engineering were sampled and utilized for our test cases. Of course, all results shown in this paper are redacted to erase personal or private information from those real-world data.

For our test cases, we executed our application on three Windows 10 pro workstation (we also tested the application on Windows 11 Pro, and it worked as well). Also, to verify that our custom application evaded other security products, we uploaded and analyzed the executable on VirusTotal.com, a service that uses over fifty malware and breach detection engines (e.g., Avast, AVG, Sophos, Kaspersky, McAfee). The results from the VirusTotal's detection and sandbox analysis reported no threats found.

Initiating our assessment, we run our specialized software, sourced from our GitHub repository "https://github.com/eamoruso/UserProfileAttack", on each of the systems targeted for analysis, to collect data and generate the necessary CSV files. In a real-world context, a malicious actor could potentially employ our program on a Windows-based target machine, leveraging avenues like a successful phishing attack [31] where the user is manipulated into running our specialized application on their system.

Once the files have been generated, we transfer them to our OneDrive account for further handling. In a malicious context, these files would instead be sent to the perpetrator's computer or cloud storage, where they could be used for unauthorized or nefarious purposes.

In the following we will introduce three test cases, each from different organizations and serving diverse industries. Each test case will show what was collected, processed, and concluded from the captured information. The names of individuals and company names will be redacted, to avoid breach of confidentiality, for each test case. All three test cases had enterprise security software running on their systems. For security and confidentiality, the security product's name is not mentioned.

3.5.1 Test Case One

To reach the intended system, we established a remote desktop session while utilizing the user's provided login credentials. Within the scope of this experiment, the necessary access information was furnished to us. With the session successfully initiated, we launched the user's web browser and obtained our application from the GitHub repository. Subsequently, the application was executed via a command prompt, enabling us to retrieve our required data from the system's registry. This can be seen in our screen shots shown in Figure 17 and Figure 18.

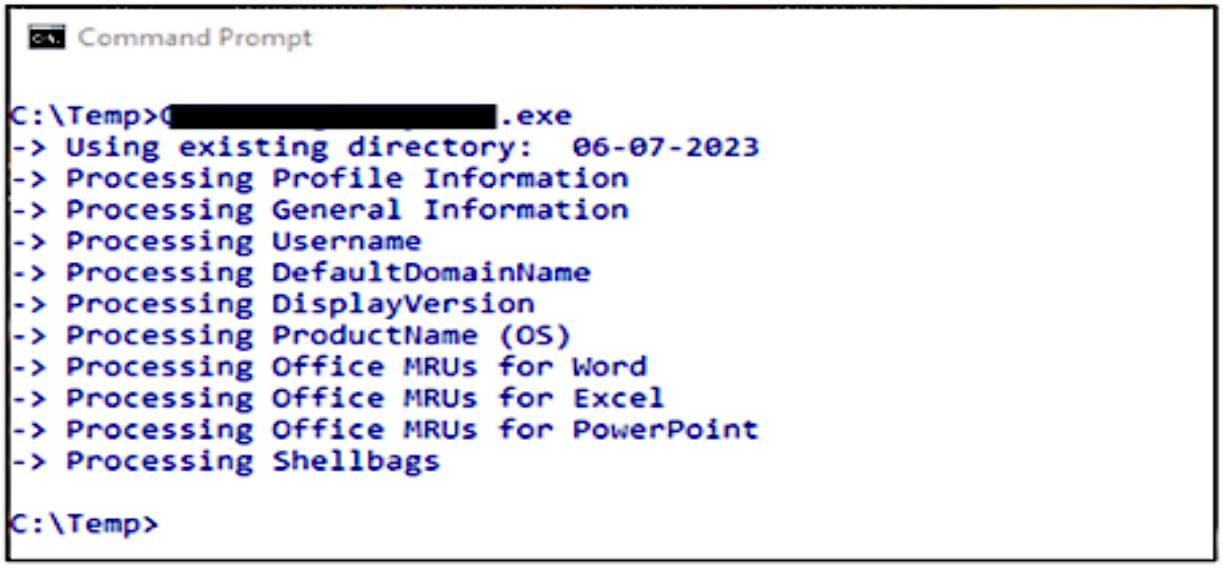


Figure 17 Running our Custom Developed Application will display processing status as it collects each registry's key data. Private information was redacted.

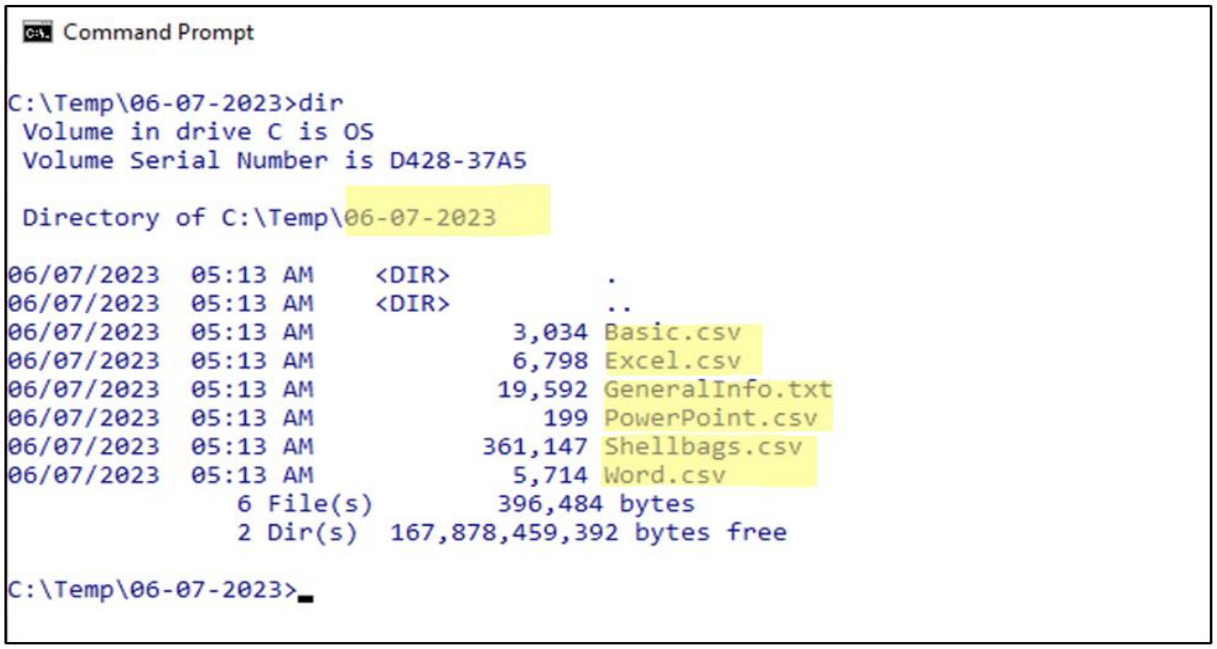
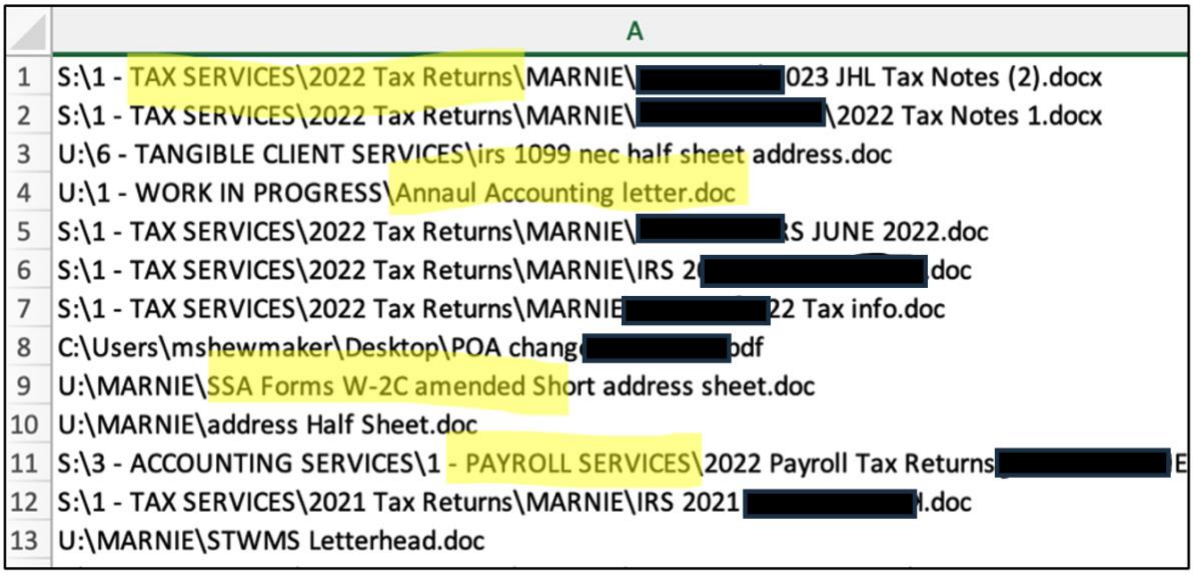


Figure 18 Example of the directory and files created during program execution.

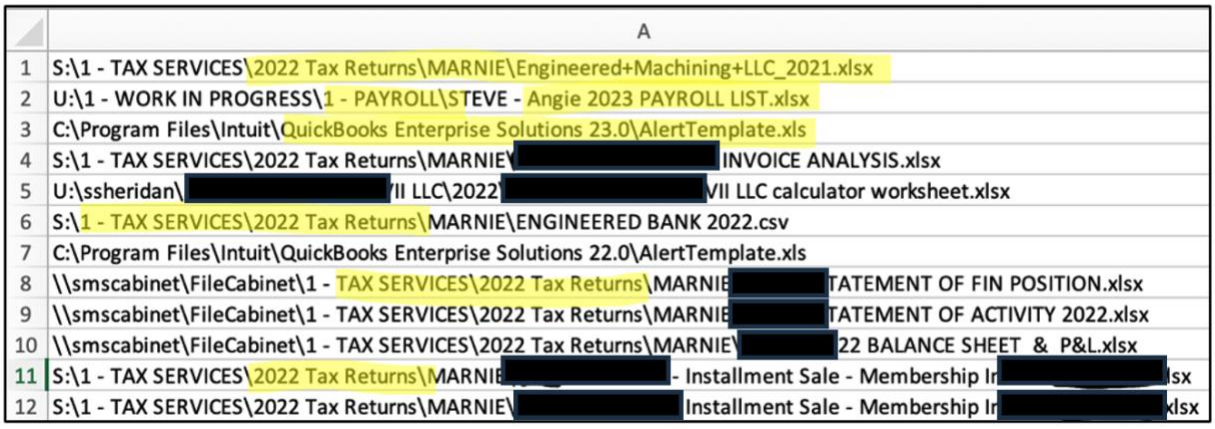
For this evaluation, we highlight and reveal the information that helps us infer the user's job responsibility, average work hours, and possible position with the organization. To accomplish this, we import the data into Excel, on a local workstation. This can also be accomplished with any other program that supports the CSV file format and provides data manipulation and graphing. With Excel, we can sort and graph our captured information from all the CSV files.

After reviewing both Excel and Word MRUs, we infer the user is either an accountant or bookkeeper since he or she is working mostly with tax return and payroll information, as shown highlighted in Figure 19 and Figure 20.



	A
1	S:\1 - TAX SERVICES\2022 Tax Returns\MARNIE\[redacted]023 JHL Tax Notes (2).docx
2	S:\1 - TAX SERVICES\2022 Tax Returns\MARNIE\[redacted]\2022 Tax Notes 1.docx
3	U:\6 - TANGIBLE CLIENT SERVICES\irs 1099 nec half sheet address.doc
4	U:\1 - WORK IN PROGRESS\Annaul Accounting letter.doc
5	S:\1 - TAX SERVICES\2022 Tax Returns\MARNIE\[redacted]S JUNE 2022.doc
6	S:\1 - TAX SERVICES\2022 Tax Returns\MARNIE\IRS 2[redacted]doc
7	S:\1 - TAX SERVICES\2022 Tax Returns\MARNIE[redacted]22 Tax info.doc
8	C:\Users\mshewmaker\Desktop\POA chang[redacted]pdf
9	U:\MARNIE\SSA Forms W-2C amended Short address sheet.doc
10	U:\MARNIE\address Half Sheet.doc
11	S:\3 - ACCOUNTING SERVICES\1 - PAYROLL SERVICES\2022 Payroll Tax Returns[redacted]E
12	S:\1 - TAX SERVICES\2021 Tax Returns\MARNIE\IRS 2021[redacted].doc
13	U:\MARNIE\STWMS Letterhead.doc

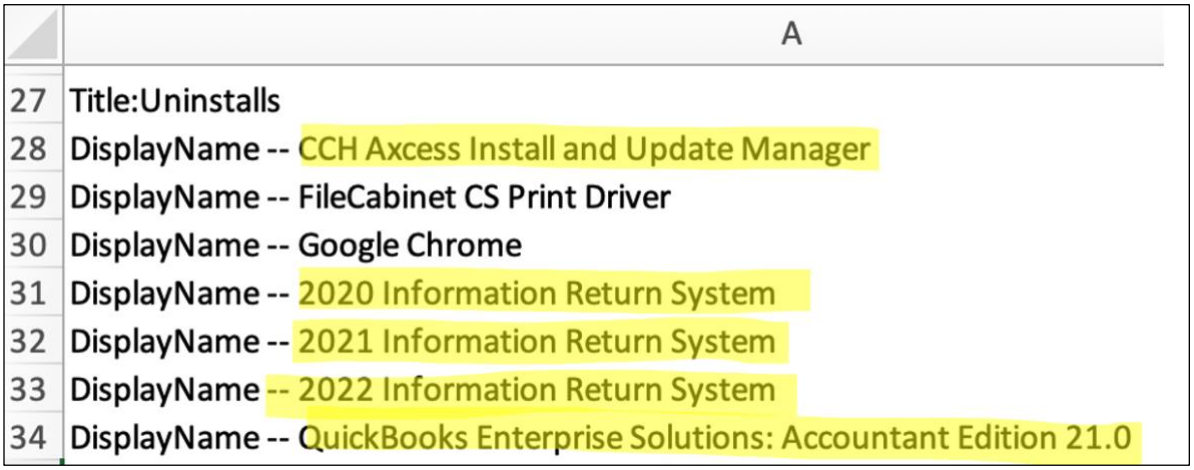
Figure 19 The Word.CSV file shows a list of most recently used Word documents. Upon reviewing, it becomes apparent from the filenames "Annaul Accounting letter" and "PAYROLL SERVICES" that the user may have an accounting job.



	A
1	S:\1 - TAX SERVICES\2022 Tax Returns\MARNIE\Engineered+Machining+LLC_2021.xlsx
2	U:\1 - WORK IN PROGRESS\1 - PAYROLL\STEVE - Angie 2023 PAYROLL LIST.xlsx
3	C:\Program Files\Intuit\QuickBooks Enterprise Solutions 23.0\AlertTemplate.xls
4	S:\1 - TAX SERVICES\2022 Tax Returns\MARNIE\[redacted]INVOICE ANALYSIS.xlsx
5	U:\ssheridan\[redacted]VII LLC\2022\[redacted]VII LLC calculator worksheet.xlsx
6	S:\1 - TAX SERVICES\2022 Tax Returns\MARNIE\ENGINEERED BANK 2022.csv
7	C:\Program Files\Intuit\QuickBooks Enterprise Solutions 22.0\AlertTemplate.xls
8	\\smscabinet\FileCabinet\1 - TAX SERVICES\2022 Tax Returns\MARNIE[redacted]TATEMENT OF FIN POSITION.xlsx
9	\\smscabinet\FileCabinet\1 - TAX SERVICES\2022 Tax Returns\MARNIE[redacted]TATEMENT OF ACTIVITY 2022.xlsx
10	\\smscabinet\FileCabinet\1 - TAX SERVICES\2022 Tax Returns\MARNIE\[redacted]22 BALANCE SHEET & P&L.xlsx
11	S:\1 - TAX SERVICES\2022 Tax Returns\MARNIE[redacted]- Installment Sale - Membership Ir[redacted]lsx
12	S:\1 - TAX SERVICES\2022 Tax Returns\MARNIE\[redacted]Installment Sale - Membership Ir[redacted]xlsx

Figure 20 Examining the list of entries in the Excel.CSV, we can reveal tax and accounting related filenames, for example “2022 Tax Returns.”

To help reinforce our claim that the user’s job function is most likely an accountant, we look at the General.txt file and identify tax and accounting software installed on this system, highlighted, and shown in Figure 21.



	A
27	Title:Uninstalls
28	DisplayName -- CCH Axcess Install and Update Manager
29	DisplayName -- FileCabinet CS Print Driver
30	DisplayName -- Google Chrome
31	DisplayName -- 2020 Information Return System
32	DisplayName -- 2021 Information Return System
33	DisplayName -- 2022 Information Return System
34	DisplayName -- QuickBooks Enterprise Solutions: Accountant Edition 21.0

Figure 21 The General.TXT file reveals accounting programs “CCH Axcess” and “QuickBooks Enterprise Solutions” available to the target user. The title “Uninstalls” means the application is available for uninstalling, meaning it is currently installed.

Using the shellbags information shown in Figure 22, we can aggregate all the time stamp information and create a frequency graph using Excel. Showing the user's most active time on the system was done by averaging every day's activity and plotting on a twenty-four-hour x-axis. Examining this graph, the user starts work on average between 6am and 8am. It can also be inferred that the user, on average, leaves work between 5pm and 6pm, with occasional after hours, is shown on Figure 22.

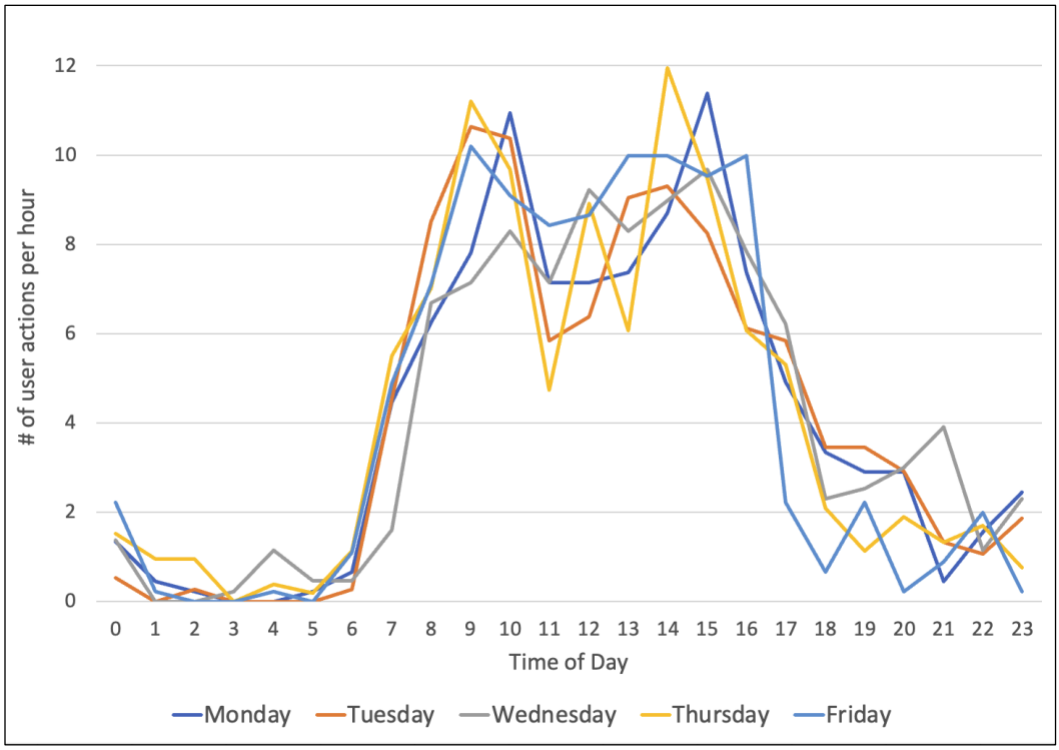


Figure 22 Daily Computer Usage Activity extracted and saved in the Shellbags.CSV file. [User is an Accountant who still does some work regularly in nighttime].

3.5.2 Test Case Two

In this test case, we perform the same process as in the first case. After reviewing Word.CSV, shown in Figure 23, we were able, with high confidence, infer the user may be a legal professional. Most of the file names contained legal words and are seen in Figure 23 with keywords such as “Order of Dismissal” and “Memorandum of Law” in the file names.

	A
1	\\Clfserver\cpshare2\CPWin\History\230307_0001\[104084] Letter re Agreed Order of Dismissal re_[redacted].doc
2	\\Clfserver\cpshare2\CPWin\History\230307_0001\[103875] Objection to Notice of Production from Non-Parties.doc
3	S:\[redacted]Lett[redacted]06.02.23_AND.DLD.doc
4	\\Clfserver\cpshare2\CPWin\History\230307_0001\[104058] Letter to Employer - [redacted]terference.docx
5	\\Clfserver\cpshare2\CPWin\History\230307_0001\[104017] Notice of Taking Deposition[redacted]doc
6	\\Clfserver\cpshare2\CPWin\History\230307_0001\[103967] Notice of Taking Deposition[redacted]ende.doc
7	\\Clfserver\cpshare2\CPWin\History\230307_0001\[103324] Memorandum of Law In Opp. to [redacted]ismiss.doc
8	S:\Wholesome[redacted]LC\WT[redacted]NTERFERENCE LETTER -[redacted]docx
9	\\Clfserver\cpshare2\CPWin\History\221004_0001\[100698] Notice of Taking Dep[redacted].doc
10	\\Clfserver\cpshare2\CPWin\History\230307_0001\[103676] Order.Writ of Bodily Attachment.doc
11	\\Clfserver\cpshare2\CPWin\History\230307_0001\[103982] Letter re Order Directing Clerk to Issue Writ of B.docx
12	\\Clfserver\cpshare2\CPWin\History\230307_0001\[103962] Motion for Clerk's Default.Eviction.doc
13	\\Clfserver\cpshare2\CPWin\History\230307_0001\[103977] Notice of Filing Returns of Service-Defendants.doc
14	\\Clfserver\cpshare2\CPWin\History\230307_0001\[103538] Summons (5-day).doc

Figure 23 Word.CSV file shows the list of last modified Word documents by the target user. This information can help identify the user’s job responsibilities and important documents on the computer or network.

To help validate if this user was working as a legal professional, we examined the General.TXT file, looking for law related software. The results highlighted in Figure 24 are programs used by legal professionals to capture billing and case information.

24	Title:Uninstalls
25	DisplayName -- Adobe Creative Cloud
26	DisplayName -- Adobe Genuine Service
27	DisplayName -- Google Chrome
28	DisplayName -- Juris Application
29	DisplayName -- Office 16 Click-to-Run Extensibility Component
30	DisplayName -- Adobe Refresh Manager
31	DisplayName -- Adobe Acrobat
32	DisplayName -- Aderant Total Office Workstation Client

Figure 24 General.TXT file shows a list of available applications, two items, ”Juris Application” and “Aderant Total Office” gives insight this user has law firm software, helping to further complement our MRUs and Shellbags findings.

Finally, we reviewed the user’s Shellbags.CSV by importing the data into Excel and creating a line graph shown in Figure 25. From the observed plots, he or she worked a consistent 8am till 5pm schedule. Also, around noon, the user’s activity drops, signifying possible lunch break, and then peaks again round 2pm till 5pm.

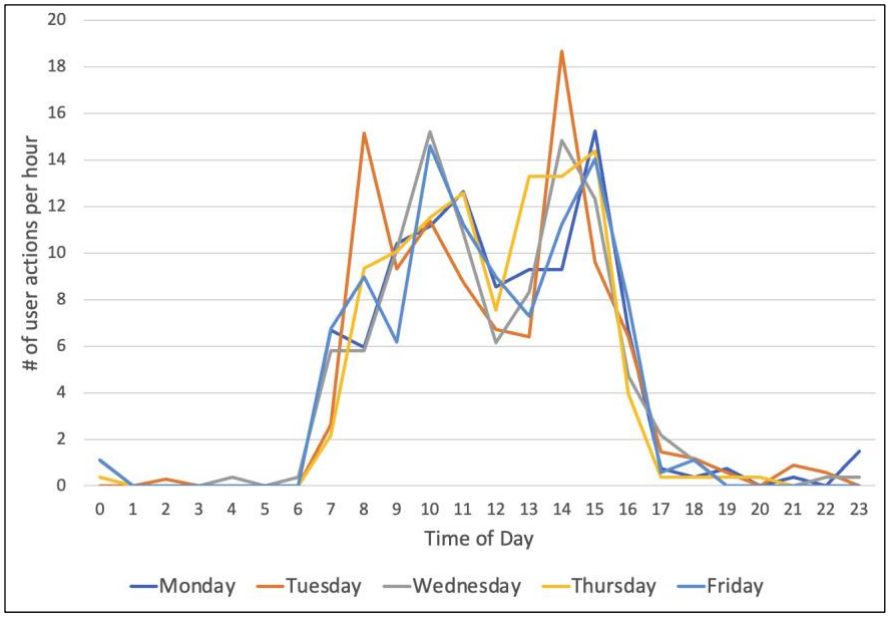


Figure 25 Daily Computer Usage Activity extracted and saved in the Shellbags.CSV file. [User is a Paralegal Employee].

3.5.3 Test Case Three

In this last test case, we collect all the information in the same manner as previous cases. Once our files have been acquired, we start evaluating each of the files. Both Word.CSV and Excel.CSV files have very little information, but a few keywords can be found, shown in Figure 26 and Figure 27.

	A	B
1	C:\Users\lauren\Desktop\MMS Manual\Madden-Updated.docx	
2	H:\Brandon\Useful Websites\CAD Commands.docx	
3	H:\Brandon\Useful Websites\Useful Websites.docx	
4	H:\MMS project schedule\Projects & Clients\All Souls.docx	
5	C:\Users\lauren\Desktop\Madden-Updated.docx	
6	C:\Users\lauren\Desktop\MMS Manual.docx	
7	C:\Users\lauren\Desktop\Madden.docx	

Figure 26 In the Word.CSV file we notice the filename “CAD Commands”, which means this user is possibly a Computer Aided Design (CAD) operator.

	A
1	H:\Data\21045\Stormwater\21045 BASIN DATA.xlsx
2	C:\Users\lauren\Desktop\PHONE EXTENSION LIST-2023.xlsx
3	H:\Data\21091\Stormwater\21091 Basin Data.xls
4	C:\Users\lauren\Desktop\Job List\JOB-LIST-ACTIVE-NUMERIC.xlsx
5	C:\Users\lauren\Desktop\Job List\JOB-LIST-ACTIVE - BY CLIENT.xlsx
6	H:\MMS project schedule\MMS Upcomming Job schedule chart.xlsx
7	H:\Payton\Design Aids\Calculators.xlsx
8	H:\Payton\Design Aids\Manning Sewer Pipe Calculator.xlsx
9	H:\MMS project schedule\MMS Project Schedule - 2.xlsx
10	H:\Data\22037\Stormwater\Stormwater Calcs.xlsx

Figure 27 Excel.CSV file lists a few keywords in filenames related to CAD, consistent with what we find in the Word.CSV file in Figure 26.

Next, we start to review the Powerpoint.CSV file and find nothing, meaning the user has never opened or used Microsoft PowerPoint. Now we examine the Basic.CSV, which will contain the user's login name, SID, and installed software. We discover an entry called "Autodesk," a developer of Computer Aided Design (CAD) software, is available to "Lauren" and shown in Figure 28

	A	B
46	Waves Audio	29-6-2020
47	lauren S-1-5-21-606991641-2029156010-1706123155-2183	
48	Adobe	20-5-2022
49	AppDataLow	13-5-2022
50	ATI	13-5-2022
51	Autodesk	27-1-2023
52	Bluebeam Software	14-2-2023
53	Canon	20-5-2022
54	Chromium	13-5-2022

Figure 28 In the Basic.CSV file we can identify the user's installed software. Note that "Autodesk" and "Bluebeam Software" are engineering applications useful in CAD design.

To continue the search to identify the user's job function, we examine the GeneralInfo.TXT file. This file also contains the user's "Run at Startup" information, to help identify programs that are executed when the user logs into the system. Again, we see CAD software at startup and hardware typically capable of running engineering applications, shown in Figure 29.

	A
1	Title:System Environment
2	NUMBER_OF_PROCESSORS -- 8
3	ProductName (OS) -- Windows 10 Enterprise
4	SystemProductName -- Precision Tower 7810
5	Title:Run at Startup (Local Machine)
6	Autodesk Desktop App -- "C:\Program Files (x86)\Autodesk\Autodesk Desktop App\
7	Autodesk Genuine Service -- C:\Program Files\Autodesk\Genuine Service\x64\Ger
8	

Figure 29 In the GeneralInfo.TXT file, we can identify the user's system hardware specifications and startup programs. Note that "Autodesk Desktop App" is scheduled to "Run at Startup" every time the user logs into the computer.

Finally, we review the user's Shellbag.CSV file, providing us user's activities. The user is shown to work different hours on different days and can be seen in Figure 30.

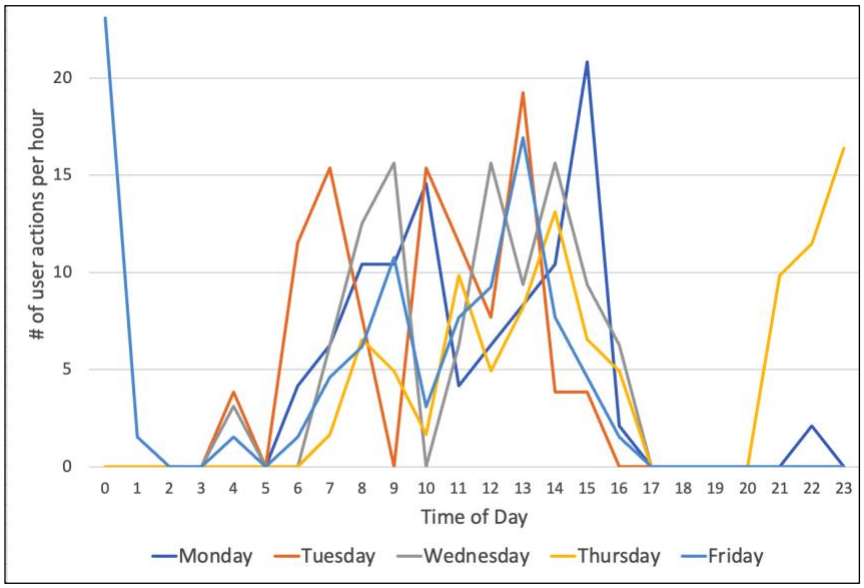


Figure 30 Daily Computer Usage Activity extracted and saved in the Shellbags.CSV file. [User is a civil engineer who frequently works late nights on Thursdays and Fridays].

3.5.4 Summary

In summary, the three real-world test cases provide a good inference on user's job functions. Having multiple indicators, such as MRUs of documents, spreadsheets, presentations, and installed applications helped establish the user's digital footprint, such as their software in use and essential documents.

More interestingly, building the user's working schedule provides insight into the culture of each profession. For example, the paralegal who worked consistently from 8am to 5pm, Monday through Friday; or the civil engineer that worked irregularly during the daytime and worked extended hours late at night in a couple of weekdays.

During our evaluation of the MRUs, we were alarmed to find a certain special files. Out of the three test cases, two users had documents named "passwords" on their network drive. Having worked in the IT industry for over 10 years, it was a surprise this practice is still used in today's security landscape. This reinforces the importance of this research to expose the dangers of effortlessly capturing a user's profile by running our custom application.

3.6 Limitations and Future Work

3.6.1 Administrative Privilege

In the context of the Windows operating system, a user account lacking administrative privileges is restricted from accessing registry data belonging to other users on the same local system. At present,

our methodology is constrained to demonstrating feasibility. We achieve this by launching our custom application from the account of a designated target user, enabling us to retrieve the digital traces associated with that user.

An alternative approach involves either employing process injection into a running program with administrative privileges or compromising the system to attain privileged access [32]. Through such means, a malicious program crafted by an attacker could potentially provide us with entry to the registry data of all user accounts on the system, enabling a more impactful attack to retrieve digital footprints of all existing users.

3.6.2 Future Work

Although the Windows registry holds an extensive volume of information, it does not retain specific details such as browsing history or exact timestamps of user interactions on websites. Nevertheless, a substantial proportion of users dedicate a significant amount of their leisure and working hours to browsing the Internet, accessing various websites. As a result, the way browsers are used, and the sites visited assume a crucial role in shaping a user's digital footprint. In our future work, we plan to incorporate a feature that enables custom or third-party applications to extract data linked to an Internet browser. Currently, the alternative approach involves utilizing a third-party utility to access these files. One such application is available from NirSoft’s web page and is accessible as freeware [33].

One more advantageous factor that could contribute to an individual's digital footprint is the incorporation of the system's user login and logout events. Although a portion of this data is stored in the registry, it qualifies as sensitive information and thus requires heightened privileges from the

operating system. Presently, the recommended approach is to make use of the "Event Viewer," a native Windows tool. In our forthcoming research, we will delve into the application of the Windows Event Log API and its integration within our customized software application.

3.7 Conclusion

In this chapter, demonstrated by our custom developed application, we revealed that it is possible to expose the digital footprint of a user's account running on a Windows machine by solely accessing the Windows registry. Alarmingly, this was achieved without triggering any security mechanisms. Using native Windows APIs in our programming, we identified a security flaw that enabled us to access a range of information from the system registry. This data covered aspects like recently opened files, timestamps related to system activities, network configurations, hardware specs, and software preferences. Such information could be leveraged to stage a sophisticated cyber-attack. Additionally, a malicious actor could glean both cyber and non-cyber personal information about the user, including work hours, lunch intervals, days off, and job functions.

CHAPTER 4: USER PRIVACY PROTECTION VIA WINDOWS REGISTRY HOOKING AND REAL-TIME ENCRYPTION

4.1 Introduction

The Windows registry is a fundamental component of the Windows operating system (OS) that stores a vast amount of configuration information and settings for various applications. It contains indispensable parameters that are essential for an enhanced user experience, such as profiles for each user, installed applications, document types, folder and icon settings, hardware configurations, and port references [36]. During OS operation, Windows will frequently interact with many of these settings.

Numerous techniques exist for extracting data from the Windows Registry. Two of the most common tools are RegEdit.exe, which offers a graphical user interface, and Reg.exe, which operates via a command-line interface. These tools are frequently employed to diagnose issues and optimize system configuration settings. In addition to these, there are several internet-based programs freely available such as RegScanner [37], Registry Explorer [38], and Registry Viewer [39].

Moreover, Windows PowerShell (powerful scripting language) can also access and parse registry data. Lastly, Microsoft offers developers a comprehensive and thoroughly documented Application Programming Interface (API) for interacting with registry records [40].

While the registry offers invaluable functionality, it simultaneously introduces substantial risks, particularly pertaining to privacy and system security. The data encased in the registry has the potential to unveil sensitive insights into a user's activities and professional responsibilities, rendering it

susceptible to privacy breaches. Furthermore, the registry unveils the user's system security configuration, potentially serving as an exploitable entry point for more advanced assailants.

It has been conclusively demonstrated that a tailored application developed using native Windows APIs can effectively collect enough registry information to construct a user's digital footprint [41]. This includes sensitive data from the user's most frequently used (MRUs) files, such as Word, Excel, and PowerPoint.

Considering the ease in which sensitive data can be accessed from the registry has prompted us to develop a defensive strategy to protect key areas of this repository. The essential strategy is to conduct real-time on the fly encryption and decryption of Widows registry data whenever user programs access the specific under-protection registry keys. The key values saved in registry are all encrypted so that an attacker cannot obtain the critical data even if he has access to the Windows registry.

In addition, our research aligns with the growing trend towards privacy-focused technologies. As individuals become increasingly concerned about data privacy, solutions that prioritize security while maintaining user convenience are likely to gain popularity. By offering an inclusive and layered security approach, we can create an obstacle for unauthorized users or malware from gaining valuable information from the registry’s rich repository.

Our proposed solution provides a comprehensive defense against privacy and security threats. Using function hooking and software-based encryption on sensitive registry data, we can effectively shield against privacy and security concerns, creating a more secure environment for users.

The key contributions of this chapter are:

- Present a new paradigm of OS-based real-time function hooking and encryption of the Windows registry of user-specific data. The proposed registry encryption approach is transparent to users and applications, and has minimum impact on computer's performance.
- Prevent malware and unauthorized actors from reading user's confidential information in the registry
- Provide a customized ready-to-use application capable of hooking and encrypting user-specific registry data

4.2 Related Work

There are several methods available to help safeguard the Windows Registry from unauthorized access. These include using disk encryption technologies, implementing Access Control Lists (ACLs), and restricting or disabling access through system policies [42] [43]. Another practice is to use detection tools to identify any suspicious registry modifications [44]. Although the mentioned techniques may offer promise, our research focuses on a more fine-grained approach to protect private user data located in the Windows registry. Moreover, we explored methods using software-based encryption and decryption of specific data within the registry.

4.2.1 Using Encryption

Encrypting data offers strong protection for sensitive and privileged information. It requires a secret value, referred to as a key, to encode and decode the data. There are two types of encryptions: symmetric and asymmetric. Symmetric encryption uses one key for both encoding and decoding the data. This type of encryption is predominant with file and drive protection. Asymmetric encryption employs a key pair system, consisting of a private and public key, commonly used for web page encryption.

When considering full disk encryption (FDE), a private key is used to encrypt the entire volume of a system. Several solutions offer this capability, some include Microsoft's BitLocker, VeraCrypt, and Symantec Encryption. All the mentioned provide encryption for the entire disk, which helps protect registry information from data theft or exposure when a device is lost, stolen, or inappropriately decommissioned. However, the solutions only protect the data when the user has turned off the machine. In other words, when the user logs into the machine, the data is unencrypted and made accessible to any running process.

Using file-level encryption on the Windows registry is not feasible due to its architecture. The registry uses several system-level files on the disk to store its information. To implement this would require a significant overhaul of the Microsoft OS. Additionally, multiple users encrypting the registry could cause conflicts and disrupt normal system operation. For example, two users would have to use the same encryption key, otherwise the registry would be locked for one of them. This example clearly voids any privacy for both users.

4.2.2 Using Access Control

Another method of securing registry data is using access control lists (ACLs). Each registry key has a security descriptor that can be leveraged to configure access control for subkeys and their values. Common tools such as Windows regedit and Group Policy Editor can be used to add permissions or access to certain registry values.

While this solution may seem promising at first, it can be cumbersome to manage and could cause system issues if not implemented correctly. Also, some applications may not be compatible with the Windows registry access control model, which can limit their ability to interact with the registry. Moreover, some registry entries are created after the permissions are set by applications, which would require constant management.

4.3 Threat Model

In this section, we explore potential threats that our system needs to guard against. We consider various types of attacks and scenarios that could potentially compromise user privacy and security.

We focus on two main categories of attackers, each characterized by distinct motivations and methodologies. Our threat model is based on the following circumstances:

- Users with elevated privileges can access sensitive registry information of other users.
- An attacker, who has obtained root privilege either through an OS compromise or privilege escalation, can access all sensitive registry information.

To prevent unauthorized access to sensitive registry data, our system utilizes encryption and decryption techniques in conjunction with a system-managed encryption key. Each user on the system possesses a unique encryption key, enabling all legitimate users on the computer to unlock only their own data. This effectively blocks attempts by malevolent actors to access other users' sensitive registry information. Furthermore, the encryption key is securely stored and managed by the OS's security subsystem, rendering it inaccessible to users and applications. In other words, the key is seamlessly handled by the OS when required to encrypt or decrypt data, preventing each user to access each other's data.

4.4 Proposed Approach

In this section, we describe how our developed application safeguards user privacy by preventing unauthorized access to specific registry data. In this paper, our primary focus is on securing personal information stored by Microsoft Office applications (e.g., Word, Excel, and PowerPoint) in the Windows registry. The same technique and our developed application can be easily extended to protect other sensitive information in the registry created by other Windows software.

To achieve registry protection for MS Office applications, we deploy Microsoft native APIs to capture and encrypt /decrypt relevant registry information while maintaining application stability. Our objective is to focus on key registry values containing each user's private data, encrypting only the selected private registry information for each user. In our prototype, we incorporated the following design features:

- Used C++ with native Windows APIs for stability and support for low-level operations.

- Highly customizable, allowing for easy addition of new registry keys requiring encryption and decryption.
- Support both 32-bit and 64-bit applications, providing a comprehensive solution for various use cases.
- Transparent monitoring of the target application interaction with the Windows registry, performing encryption and decryption when it detects registry key name requiring protection.
- Used an industry-proven cryptographic algorithm called Advanced Encryption Standard (AES) for encryption. Considered widely adopted and tested for its reliability and security.
- Key management is performed by the Windows OS through its provided Data Protection API (DPAPI), delivering an additional layer of protection by hiding the keys from both applications and users.

4.4.1 Registry Modification

Applications typically interact with the Windows registry through APIs provided by the operating system. Microsoft advises against directly modifying the contents of the Registry file by applications, as it can lead to system instability. Instead, they should use the Windows-provided API functions to ensure safe and stable access and manipulation of registry values.

Our approach leverages Microsoft's Windows APIs and "Detours", a software package for monitoring and instrumenting API calls on Windows, to intercept and manipulate registry data [45]. As illustrated in Figure 31, this technique allows us to selectively modify specific registry values without

disrupting the target application's intended functionality. This strategy is comparable to a man-in-the-middle attack, where our program intercepts and alters the communication between the user's application and the registry. By utilizing this technique, we can achieve complete transparency for the original Windows applications while ensuring the desired changes are made to the registry data.

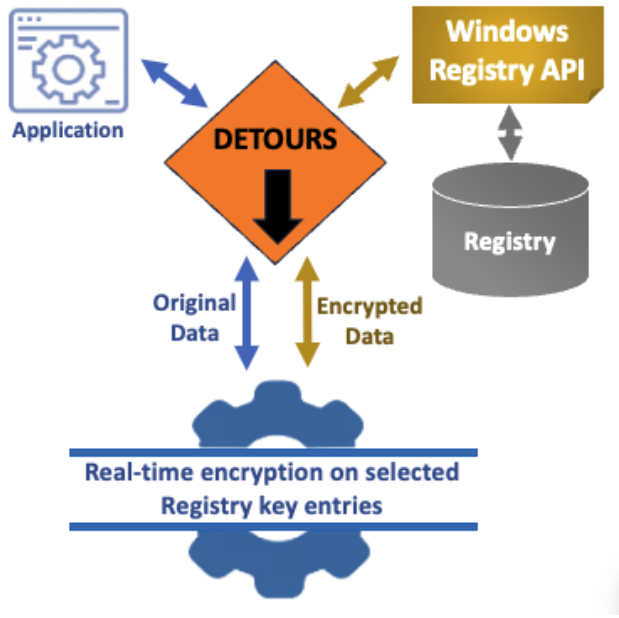


Figure 31 Detours function to hook and capture application's Windows registry API calls and modify selected registry key data.

4.4.2 Securing with Encryption

Encryption is a vital component in securing data. The robustness of encryption algorithms and key management techniques are essential factors to protect sensitive information. Considering the significance of these factors, we decided to use Microsoft's Data Protect API (DPAPI), which is a built-in component of Windows 2000 and later versions [46]. It utilizes Advanced Encryption Standard (AES) and

supports transparent (no user-interface) encryption and decryption of data securely. Additionally, DPAPI is designed to simplify the process of employing cryptographic techniques, by removing the need for users to manage and store encryption keys [47].

DPAPI offers two types of encryption keys: user-specific and system credentials. Both key types are managed securely by the OS, either tied to a user account or to the system itself. In our approach, we employ the user-specific method, which ties encryption to a specific user's login credentials. This requires the same user to be logged in to successfully decrypt his or her own data. If the system credentials were used instead, any process or other users running on the system could potentially decrypt any user's data, which could lead to security issues. Therefore, using the user-specific approach ensures that each user's data is protected and can only be accessed by the same user.

4.4.3 Security Analysis

Data Protection API (DPAPI) provides exceptional security against attacks that aim to reverse-engineer or intercept encrypted data. Its high level of security is substantiated by its use in Microsoft Edge, which encrypts sensitive information, such as passwords and credit card numbers, securely when they are saved [48]. If an attacker gains local administrator rights and accesses the encrypted data stored on the local machine while the user is not logged in, DPAPI is designed to prevent decrypting the data.

DPAPI offers some advantages in managing the encryption key, but it also has a drawback. Specifically, any other application running on the same device and under the same user account as the protected data could potentially access that data if they know about its existence. To further secure the encryption process, DPAPI provides an entropy feature that adds an extra layer of complexity by

incorporating a randomly generated key phrase into the application's code. This makes it more difficult for an attacker to access the encrypted data, as they would need to perform a thorough analysis of the application's code to extract the key phrase.

In summary, an attacker can access a user's encrypted information in two main scenarios. First, if the user is currently logged into the system and running the attacker's malicious code. Second, if an attacker has local or remote session access using the target user's credentials, such as their username and password. Even if the attacker has root privileges to the system, a session with the user's credentials must be used to decrypt any encrypted data. It's important to note that each user has their own encryption key to prevent users from accessing each other's encrypted data.

4.5 Implementation

As proof of concept that supports the intent of this paper, we created a program named RunRegProtect. This program uses a special technique known as hooking where the registry Application Programming Interface (API) calls are intercepted. Then, each read from and write to the registry is monitored and the data potentially altered.

This section contains an explanation of the program and its inner workings. It was written in C++ since this language has been integral in Windows development for many years. C++ is also on a lower level than other languages such as C# and a bit easier for programs of this type.

The entire project can be opened, edited, and recompiled using Visual Studio. While Version 2022 was utilized, later versions of Visual Studio should also be compatible. The project is open source and can be found on GitHub at the link: https://github.com/eamoruso/UserPrivacyProtect.

4.5.1 Making Registry Calls

The Windows API functions for registry access are rich with functionality. The functions in Table 1 are a small subset, but represent the functions, which when hooked, meet our needs. To create a registry entry and write data to the registry in the newly created entry can be performed with the sequence of calls to RegCreateKeyExW, RegSetValueExW, and RegCloseKey as shown in Figure 32.

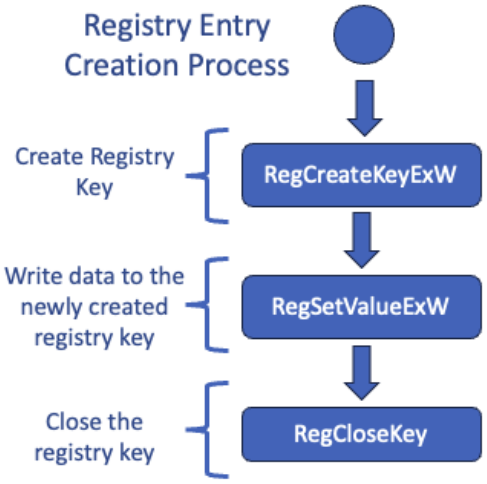


Figure 32 Creating and Saving Data to a New Registry Key.

If a registry key already exists, then a call to the API function RegOpenKeyExW followed by calls to RegSetValueExW and RegCloseKey can be made. This process is shown in Figure 33 (b).

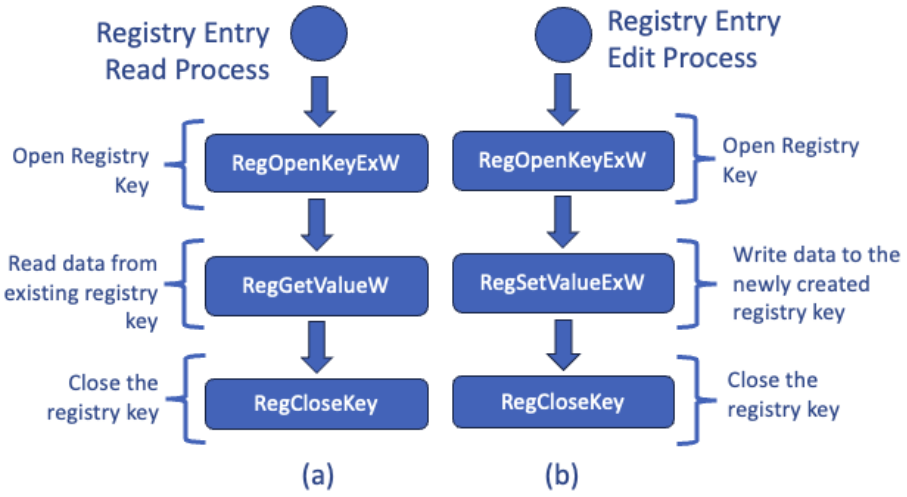


Figure 33 (a) Reading data from a registry key entry. (b) Creating and saving data to an existing registry key.

When examining our code for all calls to RegCreateKeyEx and RegOpenKeyExW, it becomes apparent that read and write permissions are specified as parameters. These parameters are represented as KEY_READ | KEY_WRITE. Once the registry key is created or opened, it allows both writing data to the key and reading data from it. The RegGetValueW function is used to retrieve the data stored in the key. This process is demonstrated in Figure 33 (a).

4.5.2 Hooking the Registry

Windows offers many features to application developers. Including the underlying systems that support file input/output, graphics, and networking. These are all provided to application developers through API calls.

The Windows architects realized early on that in special cases, user applications needed to intercept API calls, perform tasks, or manipulate data, and then let the Windows API calls perform their tasks normally. When a user application does this for an API call, it's called hooking. Most developers never use this technique, and most developers are not aware of its existence.

Windows hooking is an advanced technique. For this reason, the demonstration program uses a library named Detours, which is provided by Microsoft and can be used free of charge for non-commercial use [49]. It allows a programmer to monitor and instrument API Calls providing a safe way to implement hooks. A reader can find this software package at https://github.com/microsoft/Detours or you can use NuGet within Visual Studio to import it.

The process of hooking the registry essentially inserts code between the caller (which in this paper is Microsoft Word, Excel, and PowerPoint) and the base API code. The steps to do this for each hooked API function are as follows:

1. Create a function with an identical structure and parameters as the base API function. For example, “my_func(par_1,par_2)” is identical to original function called “real_func(par_1, par_2)” we are going to hook.

2. Get the address of the base API in a variable so that our code can call the base API.

3. In the new function call, use the base API after the additional processing is completed with the saved base API address.

4. Call DetourAttach function to insert the new function in the chain. The operating system will now call our function instead of the base API function. When our function has completed its processing, it will in turn call the base API function.

By using the hooked code, our developed software decides whether to encrypt or decrypt data based on a registry key name (e.g., Item 1, Item 2, Item 3, …) that is being passed by the application, in this case Word. A total of seven registry functions were hooked, and these essential functions used by Microsoft Office applications are listed in Table 6.

Table 6 Registry API Functions Hooked by the RunRegProtect.

Registry Function	Description
RegOpenKeyExW	Opens the specified registry key
RegCreateKeyExW	Creates the specified registry key
RegSetValueExW	Sets the data and type of a specified value under a registry key
RegQueryValueExW	Retrieves the type and data for the specified value name associated with an open registry key
RegEnumValueExW	Enumerates the values for the specified open registry key. The function copies one indexed value name and data block for the key each time it is called
RegEnumKeyExW	Enumerates the subkeys of the specified open registry key. The function retrieves information about one subkey each time it is called
RegCloseKey	Closes a handle to the specified registry key

4.5.3 Registry Encryption

To effectively secure sensitive information, it is important to have a robust encryption method that can be reliably reversed and decrypted. For demonstrating a protocol, a widely used encryption standard known as the Data Protection API (DPAPI) was employed in this program. The DPAPI offers convenient services for encrypting and decrypting data without the need to manage cryptographic keys manually. For example, either plaintext data is passed to DPAPI, and an obscure protected data BLOB is received back, or the protected data BLOB is passed to DPAPI, and the plaintext data is received back. The encryption and decryption process can be seen in Figure 34.

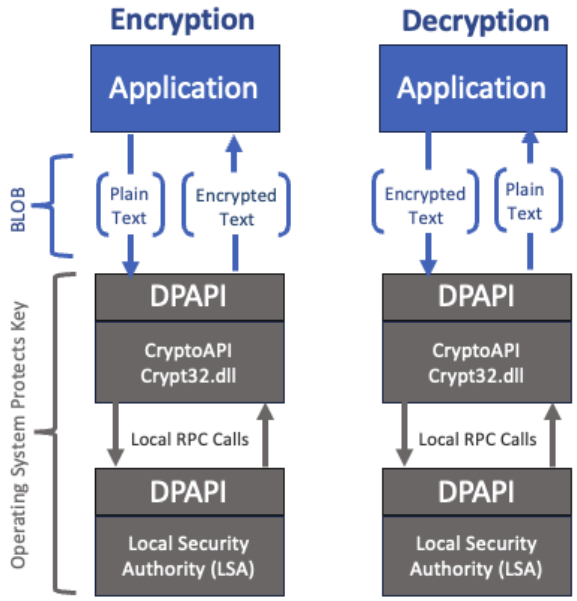


Figure 34 DPAPI Encryption and Decryption Process where User's Encryption Key is Managed by the Operating System's Local Security Authority (LSA) thru Local RPC Calls.

In our developed program, we created a pair of functions named "CryptProtectData" and "CryptUnprotectData". Together, these functions perform the encryption and decryption tasks. A reader

can easily modify this section of source code to use other cryptographic algorithms such as DES. Windows operating system has included most standard encryption methods.

Our program includes a built-in comparison function named “wcsncpm” that holds the registry key names containing data we will encrypt and decrypt. In our custom application, we compare the key name with “Item”, which is used by Word to store document names created by the user. Any key names other than these will be ignored, leaving the data unencrypted. Readers can easily add new key names to this function to extend encryption protection to other targeted Windows applications.

4.6 Evaluation

In our evaluation, we illustrate the success and efficiency of our custom-developed application, RunRegProtect, in encrypting sensitive user data stored within the Windows registry. For the proof of concept, we demonstrate the encryption of data generated by Microsoft Office applications, particularly Word, into the user's designated registry keys. These keys are comprised of filenames assigned by the user when saving their documents in Word, spreadsheets in Excel, and presentations in PowerPoint. Using Word as an example, Figure 35 depicts all files created by the current user in the registry, displayed as plain text.

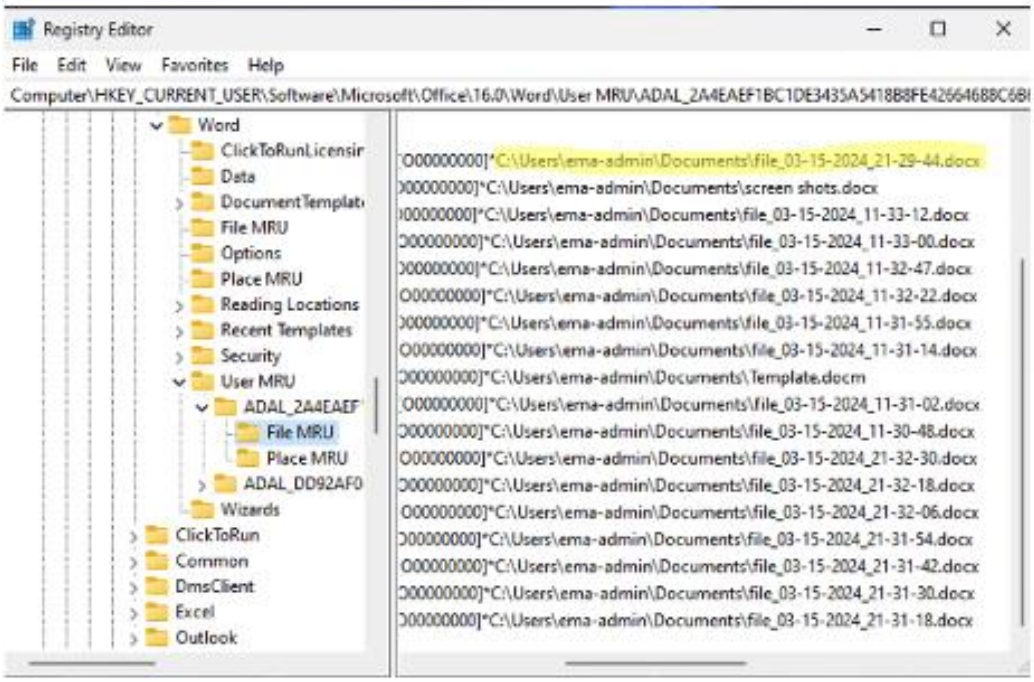


Figure 35 Registry entry listing all Word files the user created on the system. The highlighted entry specifies the location and name of the user's file.

To produce our results, we utilized a virtual machine (VM) hosted on a Microsoft Hyper-V server running Windows 11 and Microsoft Office 365. Our remote access to the environment was achieved using remote desktop. Additionally, it is important to note that the development and testing of our application were carried out on a Parallels VM operating Windows 11 and Office 2019 Professional Plus. Although we used a later version of Office for development, both versions shared the same registry storage structure, resulting in our custom-developed application performing identically across environments. The specifications of the evaluation system's hardware and software are outlined in Table 7.

Table 7 Hardware Specifications for Evaluation System

Type	Description
System Hardware	Dell PowerEdge R440 Intel Xeon Silver 2.1 GHz System 32 GB Memory
Operating System	Window 11 Enterprise Version 23H2 (OS build 22631.3296)
Microsoft Office	Office 365 MSO 64-bit for enterprise Version: 2401 Build 16.0.1731.20290

4.6.1 Experiment

To assess the effectiveness and reliability of our application in the context of Word, we conducted a series of experiments utilizing MacroRecord [50], a tool capable of recording user mouse and keyboard interactions and providing playback functionality to replicate these actions up to a specified extent. Furthermore, we crafted a Word macro file (e.g., template.docm) to complement our simulation. This macro, developed in Visual Basic for Applications (VBA), facilitates the automation of repetitive tasks and data processing functions. Specifically, we leveraged these capabilities to generate a file stamped with the current date and time, subsequently saving it in the user's documents directory. The code utilized to execute this process is illustrated in Table 8.

Table 8 Document Template with Macro.

Template.docm
Sub AutoOpen() Dim strFileName As String Dim rngHeader As Range ' Create a header that says "RegRunProtect" Set rngHeader = ActiveDocument.Sections(1). Headers(wdHeaderFooterPrimary).RangerngHeader.Text = "RegRunProtect" ' Specify the file name with today's date and current time strFileName = "C:\path\to\your\file_" & Format(Now, "mm-dd-yyyy_HH-MM-SS") & ".docx" ' Save and close the document with the specified file name ActiveDocument.SaveAs2 FileName:=strFileName ActiveDocument.Close End Sub

Lastly, we recorded the encryption and decryption processing times for each simulated iteration and saved the results to a comma-separated values (CSV) file. The results of this file were then used to demonstrate the efficiency of our application in the next section.

4.6.2 Results

Our results demonstrate the successful encryption of registry entries containing users' private information, specifically the location and name of the document. In Figure 5, we present an example of

normal registry entries created by Word to store filenames that the user has generated. After running Word with RunRegProtect, our application encrypted those filenames, as depicted in Figure 36.

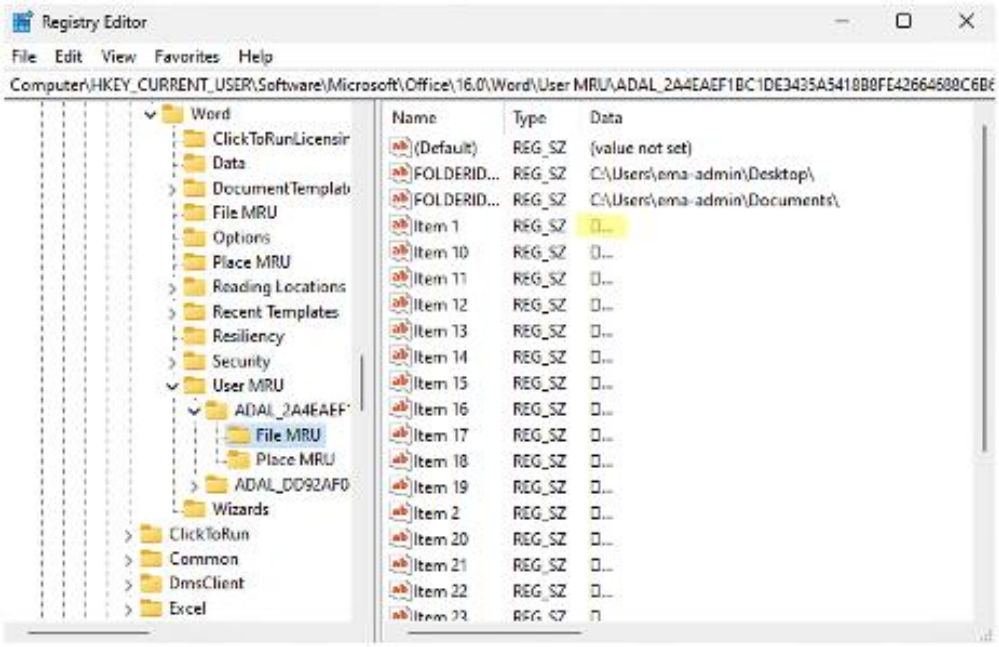


Figure 36 Registry entries of Word documents successfully encrypted by RunRegProtect. The highlighted item is encrypted.

4.6.3 Performance

During our performance assessment, we discovered that our application did not negatively impact user experience. Prior to collecting our results, a recording of the steps for our simulation was generated, taking a total of 15 seconds for the entire process which involved creating, modifying, saving (with a designated filename), and closing a document. Then this simulation was repeated 45 times. The data collected was then transferred to a spreadsheet to construct the graph depicted in Figure 7. On average, the process without protection took 15.218 seconds, while with the RunRegProtect feature

enabled, it took 15.269 seconds. The difference between these two timings indicates that the additional time required was a mere 0.051 seconds beyond the standard process time.

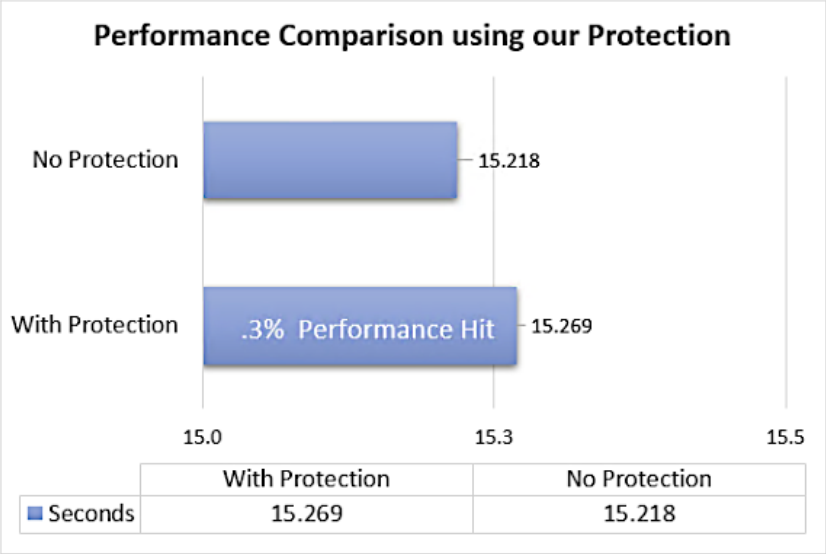


Figure 37 Performance comparison between using our protection and no protection.

4.7 Conclusion

Our proposed solution offers a novel approach to safeguarding user privacy in the Windows registry. By leveraging Microsoft APIs, we intercept, monitor, and encrypt sensitive user data stored in the registry, ensuring that only authorized individuals and applications have access to this information. This helps prevent unauthorized actors from exploiting personal data for malicious purposes. Our custom-developed program provides users with a sense of security, knowing that their personal data is protected within the registry, while also offering transparency to both the user and application.

CHAPTER 5: CONCLUSION

The escalating prevalence of cybersecurity breaches has resulted in extensive financial losses for individuals and organizations worldwide. Despite the continuous development of strategies and methods to counteract this growing threat, the fight against cybercrime seems futile. It is imperative to question whether we are addressing this security pandemic appropriately. Some experts attribute the cause of most compromises to the underlying software architecture in operating systems and those who create the applications we rely on today. Others point the blame at nefarious individuals who exploit vulnerabilities and other techniques to disrupt both users and organizations.

To help combat this, our research pivots around the Windows registry, shedding light on its pertinence to analysis, potential vulnerabilities, and defense mechanisms. By examining key components of the Window registry, particularly shellbags, we provide digital forensic investigators with an efficient and effective tool, SeeShells, to analyze suspect user activities. Additionally, we introduced methodology in creating a user's digital profile or footprint by solely accessing the registry. This is easily demonstrated by our custom developed application. In our final proposition, we present a method to shield users from bad actors who might exploit the registry for malicious intents. We leverage the Windows Data Protection Application Programming Interface (DPAPI) to encrypt sensitive data stored in the registry, enhancing user privacy and security.

LIST OF 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] 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]

[11] E. Bott, *Windows 10 IT Pro Essentials Top 10 Tools*. First Edition (1st. ed.). USA,: Microsoft Press, 2016. ch. 9, pp. 128-129.

[12] R. Kazanciyan, M. Hastings. "Investigating powershell attacks." Mandiant. Accessed: Feb. 7, 2023. [Online]. Avaliable: https://www.blackhat.com/docs/us-14/materials/us-14-Kazanciyan-Investigating-Powershell-Attacks-WP.pdf

[13] W. Munroe, "Why Malicious Actors Love PowerShell Attacks and How to Defend Them." Rangeforce.com. https://www.rangeforce.com/ blog/powershell-attacks-and-how-to-defend-them (accessed Feb. 13, 2023).

[14] T. Wojewoda. "Hunting and Gathering with PowerShell," SANS Institute., White Paper, 2019. [Online]. Available: https://www.giac.org/research-papers/38842/ (accessed Feb. 14, 2023).

[15] Y. Zhu, P. Gladyshev, and J. I. James, "Using shellbag information to reconstruct user activities," *Digital Investigation*, vol. 6, pp. S69–S77, Sep. 2009, doi: 10.1016/j.diin.2009.06.009.

[16] R. Mize, "Behavior of Shellbags in windows 10." Ph.D. dissertation, Utica College, 2018. [Online]. Available: https://www.proquest.com/dissertations-theses/behavior-shellbags-windows-10/docview/2108990957/se-2

[17] A. Đuranec, D. Topolčić, K. Hausknecht and D. Delija, "Investigating file use and knowledge with Windows 10 artifacts," 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 2019, pp. 1213-1218, doi: 10.23919/MIPRO.2019.8756877.

[18] "Publicly Available Tools Seen in Cyber Incidents Worldwide," *Cybersecurity and Infrastructure Security Agency CISA*, Accessed: Feb 14, 2023. [Online] Available: https://www.cisa.gov/news-events/cybersecurity-advisories/aa18-284a

[19] R. Cobb. “Entering a Covenant: .NET Command and Control.” Specterops.io., Accessed: Apr. 22, 2023 [Online]. Available: https://posts.specterops.io/entering-a-covenant-net-command-and-control-e11038bcf462?gi=ce1cb24bd25a

[20] A. Barakat and A. Hadi, "Windows Forensic Investigations Using PowerForensics Tool," *2016 Cybersecurity and Cyberforensics Conference (CCC)*, Amman, Jordan, 2016, pp. 41-47, doi: 10.1109/CCC.2016.18.

[21] S. Zavala, N. Shashidhar and C. Varol, "Cybersecurity Evaluation with PowerShell," in *2020 8th International Symposium on Digital Forensics and Security (ISDFS)*, Beirut, Lebanon, 2020, pp. 1-6, doi: 10.1109/ISDFS49300.2020.9116258.

[22] A. Singh, H. S. Venter, and R. A. Ikuesan, “Windows registry harnesser for incident response and digital forensic analysis,” *Australian Journal of Forensic Sciences*, vol. 52, no. 3, pp. 337–353, Dec. 2018, doi: 10.1080/00450618.2018.1551421.

[23] Y. Zhu, P. Gladyshev, and J. I. James, “Using shellbag information to reconstruct user activities,” *Digital Investigation*, vol. 6, pp. S69–S77, Sep. 2009, doi: 10.1016/j.diin.2009.06.009.

[24] A. D. Kent and L. M. Liebrock, "Differentiating User Authentication Graphs," 2013 IEEE Security and Privacy Workshops, San Francisco, CA, USA, 2013, pp. 72-75, doi: 10.1109/SPW.2013.38.

[25] “Windows Registry Cheat Sheet,” www.13cubed.com. https://www.13cubed.com/downloads/windows_registry_cheat_sheet.pdf (accessed Feb 11, 2023).

[26] F. Korkmaz, “Windows Artifacts - Fahri Korkmaz - Medium,” *Medium*, Jan. 07, 2022. [Online]. Available: https://r4bb1t.medium.com/windows-artifacts-8fae778aa8c7

[27] V. D. Munister, A. L. Zolkin, A. V. Ishkov, O. V. Kosnikova, and I. A. Poskryakov, “Computational analysis of the digital footprint using machine learning and artificial intelligence,” *Journal of Physics*, vol. 2094, no. 3, p. 032003, Nov. 2021, doi: 10.1088/1742-6596/2094/3/032003.

[28] R. Yasin, “Stealing Data By ‘Living Off The Land,’” *Dark Reading*, Sep. 03, 2015. [Online]. Available: https://www.darkreading.com/analytics/stealing-data-by-living-off-the-land-

[29] “Companies using Microsoft Office 365.” Enlyft.com. https://enlyft.com/tech/products/microsoft-office-365 (accessed May 30, 2023).

[30] “How to reset user options and registry settings in Word - Microsoft 365 Apps,” *Microsoft Learn*, May 05, 2022. https://learn.microsoft.com/en-us/office/troubleshoot/word/reset-options-and-settings-in-word (accessed May 30, 2023).

[31] R. Alabdan, “Phishing Attacks Survey: Types, vectors, and technical Approaches,” *Future Internet*, vol. 12, no. 10, p. 168, Sep. 2020, doi: 10.3390/fi12100168.

[32] A. Mohanta and A. Saldanha, “Code injection, process hollowing, and API hooking,” in *Apress eBooks*, 2020, pp. 267–329. doi: 10.1007/978-1-4842-6193-4_10.

[33] BrowsingHistoryView. (2023), NirSoft, [Online]. Available: https://www.nirsoft.net/utils/browsing_history_view.html

[34] “Phishing Guidance: Stopping the Attack Cycle at Phase One.” Internet Crime Complaint Center (IC3), Oct. 2023, www.ic3.gov/Media/News/2023/231018.pdf

[35] Ani Petrosyan, “U.S. Most Frequently Reported Cyber Crime by Number of Victims 2022.” Statista, Aug. 29 2023, [Online]. Available: www.statista.com/statistics/184083/commonly-reported-types-of-cyber-crime-us/.

[36] “Windows registry information for advanced users”, Microsoft, 2024, [Online] Available: https://learn.microsoft.com/en-us/troubleshoot/windows-server/performance/windows-registry-advanced-users. [Accessed Feb. 11th, 2024]

[37] Sofer, Nir, “RegScanner”, NirSoft, 2024, [Online] Available: https://www.nirsoft.net/utils/regscanner.html. [Accessed: Feb. 18, 2024]

[38] Zimmerman, E., “Registry Explorer”, Eric Zimmerman’s Tools, 2024, [Online]. Available: https://ericzimmerman.github.io/#!index.md. [Accessed: Feb. 18, 2024]

[39] “Registry Viewer”, Exterro, 2017, [Online] Available: https://www.exterro.com/ftk-product-downloads/registry-viewer-2-0-0. [Accessed: Feb. 18, 2024]

[40] “Registry Functions”, Microsoft, 2024, [Online] Available: https://learn.microsoft.com/en-us/windows/win32/sysinfo/registry-functions. [Accessed: Feb. 28, 2024]

[41] Amoruso, Edward L., Cliff C. Zou, and Richard Leinecker, "User Profiling Attack Using Windows Registry Data," 2023 IEEE 14th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 2023, pp. 171-181, Doi: 10.1109/UEMCON59035.2023.10315968.

[42] Halsey, M., Bettany, A. (2015). Securing the Registry. In: Windows Registry Troubleshooting. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-0992-9_5.

[43] Paine, Luke. “The Defender’s Guide to the Windows Registry.” SpecterOps, 12 Dec. 2023, specterops.io/blog/2022/10/31/the-defenders-guide-to-the-windows-registry/.

[44] Sakthisrini. “The Role of Windows Registry in Cybersecurity.” Medium, Medium, 1 Nov. 2023, medium.com/@sakthisrini23/the-role-of-windows-registry-in-cybersecurity-21d18eca848c.

[45] Brubacher, Doug. "Detours: Binary interception of Win32 functions." Windows NT 3rd symposium (windows NT 3rd symposium). 1999.

[46] C. Lin, “October 2022,” cuteprogramming, https://cuteprogramming.blog/2022/10/ (accessed Mar. 14, 2024).

[47] “Windows Data Protection,” Microsoft Learn, https://learn.microsoft.com/en-us/previous-versions/ms995355(v=msdn.10)?redirectedfrom=MSDN (accessed Mar. 15, 2024).

[48] Dan-Wesley, “Microsoft edge password manager security,” Microsoft Learn, https://learn.microsoft.com/en-us/deployedge/microsoft-edge-security-password-manager-security (accessed Mar. 12, 2024).

[49] “Detours,” Microsoft Research, https://www.microsoft.com/en-us/research/project/detours/ (accessed Mar. 20, 2024).

[50] B. Media, “Record Mouse and keyboard actions for infinite replay...,” Macro Recorder for Windows/Mac, https://www.macrorecorder.com/ (accessed Mar. 14, 2024).