Mission Implausible: Defeating Plausible Deniability with Digital Forensics

Michael Smith

Mission Implausible: Defeating Plausible Deniability with Digital Forensics

GIAC (GCFE) Gold Certification

Author: Michael Smith, mesmith1@protonmail.com
Advisor: *David Cowen*

Accepted: 09 February 2020

Abstract

The goal of plausible deniability is to hide potentially sensitive information while maintaining the appearance of compliance. In simple terms, it is granting someone access to a safe but keeping items of real value successfully hidden in a false bottom.

Encryption platforms such as VeraCrypt and TrueCrypt achieve this goal in the digital realm using nested encryption. This nesting typically takes one of two forms; a deniable file system or a deniable operating system (OS). The deniable file system uses the interior of an encrypted container to mask its presence, akin to the false bottom to the safe analogy. The deniable operating system uses an encrypted bootable partition to mask the presence of a second OS, much like a safe that reveals a different compartment based on how a key turns in the lock.

The use of encryption to create a scenario for plausible deniability presents a significant threat to the success of law enforcement and digital forensic professionals. Performing registry analysis and digital forensics is the metaphorical equivalent of using a magnifying glass to look for clues inside the safe with a false bottom or a key-based compartment. When forensics is successful in revealing clues of a deniable file system, it effectively defeats the case for plausible deniability. The goal of this research is to explore the digital forensics metaphorical equivalent of such clues.

1. Introduction

The phrase 'Plausible Deniability' was first coined by Michael Roe. His initial intent was to provide a means of achieving non-repudiation in the use of cryptography (Roe, 1997). Roe's purpose for non-repudiation seemed narrowly focused on convincing others of the truth of the use of cryptography.

This concept of plausible deniability was expanded upon by Canetti et al. in their work, Deniable Encryption (Canetti, Dwork, Naor, & Ostrovsky, 1997). The evolution expressed by Canetti is more in line with the common usage of plausible deniability with regards to cryptography. They show it best in the statement, "Presenting convincing data is almost always more credible than saying 'I erased', or 'I forgot'" (Canetti, Dwork, Naor, & Ostrovsky, 1997).

In their research, Canetti et al. focused on text encryption. In 1999, McDonald and Kuhn presented a prototype for a deniable file system called StegFS. The goal of StegFS was to bring plausible deniability to data storage because previous text-based methods were only practical for short messages (McDonald & Kuhn, 1999). In 2004, The TrueCrypt Team released TrueCrypt version 1.0, and in 2014 The TrueCrypt Foundation abandoned the project under suspicious circumstances (Bischoff, 2018). TrueCrypt became very popular during its 10-year period of development and support. This popularity is evident by the spinoff of a nearly identical application called VeraCrypt ("VeraCryptFree Open source disk encryption with strong security for the Paranoid," n.d.). Some prominent features of TrueCrypt and, by extension, VeraCrypt, relate directly to plausible deniability. These features include a hidden volume and a hidden operating system.

The goal of VeraCrypt's plausible deniability is to convince others the use of encryption is benign, and give the appearance of full cooperation with an investigation. Many researchers have provided a myriad of scenarios where plausible deniability might be beneficial to the user: resisting secret police (Czeskis et al., 2008), usage by a terrorist (Balogun, 2013), oppressive regimes (Balogun, 2013), and more.

Michael Smith, mesmith1@protonmail.com

Fortunately for digital forensics professionals, the plausible deniability afforded to users of VeraCrypt can be defeated through intermediate and advanced forensic techniques. Techniques such as: Windows registry analysis, Windows operating system analysis, and memory forensics.

2. Threat Model

The threat model represents potential scenarios for the compromise of plausible deniability provided by VeraCrypt's encryption techniques. These scenarios consist of two key elements: the level of access and the scope of information leakage. The level of access describes the frequency a forensic examiner will have access to the media to identify information leakage. Meanwhile, the source of information leakage aids a forensic examiner in determining the difficulty of finding information leakage and the source of the leakage.

2.1. Access

The concept of access is simply to scope the level of contact a forensic professional has with the media. More contact, of course, will grant more opportunities for evidence to be captured, and for differences in the state of the media to be recognized.

2.1.1. One-Time Access

One-Time access is a scenario in which a forensic examiner or adversary has contact with the media for a single set period without simultaneous use by the user. Czeskis et al. describe it as a “single snap-shot of a disk image” like the police might take when they image a hard drive (2008). This scenario is the most common simply because for any of the other access scenarios to occur, there must be one-time access. Additionally, one-time access is the most challenging scenario. Without repetitive exposure, an examiner cannot base any analysis on trends, but must solely rely upon what they find with a single access. For example, Kedziora expresses a very pessimistic view of the ability of an examiner to identify the existence of a hidden volume: "Existence of the hidden volume, which is a DFS, cannot be proven via One-Time Access methods"

(Kedziora, 2017). Since single-access scenario is the most common and most challenging, it will be the focus of this analysis.

2.1.2. Intermittent Access

Intermittent access is the scenario in which the examiner has repeated but has not had routine contact with the media. Czeskis et al., describe it as "several snap-shots of the disk image, taken at different times," like "border guards who make a copy of Alice's hard drive every time she enters or leaves the country" (2008). The repeated access in this scenario enables the examiner to identify differences between the state of the media and analyze trends.

VeraCrypt's hidden file system hinges on the fact that encrypted files and randomized data both have very high entropy or randomization. This similarity in entropy between encrypted files and the unused section of the disk makes it almost impossible to differentiate encrypted files embedded in the unused space. Intermittent access overcomes this concealment by comparing the states of the unused areas of the data to previous states of the data to find where modifications have occurred between captures. This comparative analysis can identify portions of the disk, which may potentially contain hidden data (Chakraborti, 2018).

2.1.3. Regular Access

Regular access is the scenario where there is frequent or routine contact with the media. Czeskis presents an example scenario where police break into Alice's apartment every day when she is away and make a copy of the disk each time (2008). Kedizora et al., propose an alternate scenario to regular access termed Live Response Access (2017). This scenario posits availability to the system while it is online or possibly in use. This scenario can make the defeat of plausible deniability relatively trivial since it can include screen scrapers, keyloggers, and other remote access capabilities.

2.2. Information Leakage

Czeskis et al., in their 2008 research, helpfully defined terminology for the scope of potential data leakage, proposing two types of leakage: (1) from above and (2) from below. I would like to posit a third scenario, which is: (3) leakage at the application.

2.2.1. Leakage from Below

Information leakage from below is evidence of a hidden container identified below the operating system. Some examples of this kind of information include raw device block wear-leveling, and file system journaling (Czeskis, 2008). Wear-leveling and file system journaling can reveal the frequency of modifications to specific blocks or files in an Intermittent or Regular access threat model.

2.2.2. Leakage from Above

Information Leakage from above is evidence of a hidden container above the operating system. According to Czeskis et al., this includes evidence of hidden volumes leaked through the operating system, 'primary applications', and 'non-primary' applications (2008).

Operating system sources include shortcut (LNK) files and specifically their ability to link the Volume Serial Number for the Hidden Volume to a specific file path (Czeskis, 2008).

'Primary applications' include applications closely associated with the creator of the operating system, such as Microsoft Office products. In the case of Czeskis et al., the exemplar application is Microsoft Word with its evidentiary artifact of Word Auto Saves.

'Non-Primary Applications' involve third-party applications. For example, Google Desktop is an application that provides detailed information regarding user activity (Czeskis, 2008). However, there is one conclusion drawn by Czeskis et al., which is problematic. They conclude that, "The Windows registry does not appear to directly compromise the deniability of a TrueCrypt hidden volume" (Czeskis, 2008). One of the goals of this research is to demonstrate the problematic nature of this conclusion.

2.2.3. Leakage from the Application

Information leakage from the application is a category of leakage that is necessary to better define current and future weaknesses in VeraCrypt's plausible deniability. This category is needed because flaws to VeraCrypt's plausible deniability are present within the design and execution of the application itself. For example, Kedziora et al. identified a weakness to the hidden operating system proposed by VeraCrypt by performing entropy

analysis on the sectors of the revealed external volume (2017). The presence of this weakness within the external volume means that it is a weakness in the application itself.

Similarly, any gap in how the application manages its memory is considered leakage from the application. For example, in 2008, Jesse Kornblum identified a means to extract the TrueCrypt encryption master key from memory (Kornblum, 2008). This research seeks to find a weakness in how VeraCrypt protects the hidden volume using memory analysis.

3. TrueCrypt / VeraCrypt

3.1. History

The TrueCrypt Team released TrueCrypt in 2004 as an on-the-fly encryption (OTFE) program based on Encryption for the Masses (E4M).

The creator of TrueCrypt remains a secret. The founders of TrueCrypt went by the name of "The TrueCrypt Team" initially (TrueCrypt Team, 2004). Later in 2004, the name became "the TrueCrypt Foundation" and remained the same until the final release of TrueCrypt 7.2 and the abandonment of the project in 2014 (Bar-El, 2014) (TrueCrypt Foundation, n.d.).

Since the initial release, plausible deniability was a critical component of True Crypt (TrueCrypt Team, 2004). The plausible deniability component of the encryption provided by TrueCrypt has always been a hurdle to digital forensics professionals and law enforcement. Balogun states in 2013:

> The TrueCrypt software went even further by providing users with plausible deniability and non-repudiation abilities. This makes digital forensics investigations of encrypted disk drives harder and less feasible.

When the TrueCrypt Foundation abandoned the TrueCrypt project in 2014, a fork of the project named VeraCrypt quickly filled the void left by TrueCrypt. VeraCrypt relies heavily on the TrueCrypt 7.1a codebase ("VeraCrypt Free Open source disk encryption with strong security for the Paranoid," n.d.) and has since made significant improvements to performance and security.

The popularity and effectiveness of TrueCrypt and VeraCrypt drew and continues to draw the attention of governments and digital forensics professionals, leading some to suggest that the disk drive manufacturers should provide a backdoor for digital forensics investigators to gain access to some encrypted disk drives (Balogun, 2013). This highlights the seriousness of the challenge presented by encryption. The willingness by some in the community to sacrifice privacy and security to gain access to encrypted content increases the drive of their counterparts to find alternatives. As a result, any approach which gleans information of user activity within encrypted volumes is highly sought after by those forensics investigators that also wish to preserve privacy and security.

3.2. Hidden Volumes

VeraCrypt provides a deniable filesystem through a feature known as a hidden volume. The methodology behind a VeraCrypt hidden volume is to fill the entire disk space of a VeraCrypt container with random high-entropy data. Then, a second encrypted container is embedded in the first container. Since the encrypted container appears the same as high-entropy data, the randomness of the unused area of the parent container masks the presence of the hidden container ("VeraCrypt Free Open source disk encryption with strong security for the Paranoid," n.d.). Thus, making the unused disk space appear indistinguishable from the encrypted container hidden within.

3.3. Hidden Operating System

In 2008, security researchers proposed the general methodology of a hidden operating system which the TrueCrypt Foundation implemented in TrueCrypt v6.0:

> Another possible direction would be to create a “True- Crypt Boot Loader” that, upon entering one password, decrypted the disk one way and booted the OS. And, upon entering a different password, decrypted the deniable portion of the disk and booted the OS in the deniable partition. (Addendum: such a boot loader is now implemented in TrueCrypt v6.0.) (Czeskis, 2008)

The implementation of the Hidden Operating System has matured over time. The TrueCrypt Foundation recommends some distinct countermeasures to strengthen the

plausible deniability of its hidden operating system against forensic analysis. These countermeasures are readily available in the "Hidden Operating System" documentation on the VeraCrypt website.

- The first countermeasure is the use of a decoy VeraCrypt encrypted operating system to make the presence of the VeraCrypt bootloader plausibly deniable.
- The second countermeasure is to embed the hidden operating system in a hidden volume in a second partition or disk. This action leverages the plausible deniability of the hidden volume to mask the presence of the second operating system.
- The third countermeasure is to destroy the evidence of setting up and accessing the hidden operating system by completely wiping and reinstalling the decoy operating system. The final countermeasure is that by default, when the user logs into the hidden operating system, all other drives are mounted as read-only, significantly reducing and possibly eliminating data leakage.

This set of countermeasures presents a significant challenge for any forensics investigator. The wiping of the installed operating system and read-only mounting of external volumes effectively defeats leaking information from above, as previously described in section 2.2.2. The embedding of the operating system in a hidden volume effectively defeats leaking from below, as described in section 2.2.1. The maturity of the countermeasures adopted by The TrueCrypt foundation effectively leaves only one alternative, to identify leaking from within the application itself, as described in section 2.2.3.

4. Methodology

Before performing the analysis, the objectives and methods that will be used to evaluate the digital traces left by the use of VeraCrypt's plausible deniability features.

4.1. Role of doubt

Achieving plausible deniability is not a purely technical pursuit. Revelations or suspicious actions by the user while their data is analyzed can destroy their plausible

deniability. Similarly, any unexplained evidence not well-supported by the user's initial account of device usage can destroy plausible deniability. Therefore, any activity possibly attributed to VeraCrypt, which is not attributable to the access granted by the user, will raise suspicion and potentially compromise their plausible deniability.

In other words, any evidence of user activity not supported by the presence of the encrypted container revealed in good faith will weaken the user's plausible deniability. For example, the user provides access to a container with a folder labeled 'sensitive information', which contains legitimately sensitive tax documents. The examiner finds that activity and also recovers evidence of user access to a folder named "attack plans" and similarly named records within. The presence of this unaccounted-for activity results in increased doubt of the compliance of the user. This doubt weakens the plausible deniability they wish to establish with their cooperation.

4.2. Windows 10 Enterprise

A fully patched Windows 10 Enterprise System is the test system that will be used for forensic analysis. This operating system should enable identification of all possible means of OS leakage since Windows 10 Enterprise will have the full suite of operating system utilities available to potentially catch evidence of activity. Pausing automatic updates after patching the system helps to reduce noise while performing the analysis.

4.3. Latest Release of VeraCrypt

The forensic analysis will be targeting VeraCrypt 1.24-Hotfix1.

4.4. Forensics Utilities

Dynamic registry analysis of VeraCrypt is captured and analyzed using Systracer Pro. Windows registry artifacts which require parsing will be par which

This analysis uses Zimmerman’s forensic utilities to parse Windows Registry and system files that are not human-readable and that require parsing.

5. Intermediate Forensic Analysis

In the field of forensics, the evidence used to indicate activity is not typically conceived to benefit a forensic examiner. For example, a criminal does not intentionally leave hair follicles, fingerprints, or surveillance videos for detectives and forensic professionals. These evidentiary clues are byproducts of everyday living. Likewise, digital forensics bases its conclusions on particular user activity by identifying the byproducts of routine digital device usage. With this in mind, the simplest way to approach the weaknesses of VeraCrypt's plausible deniability to forensic analysis will be to focus on user activity—specifically, how user activity influences VeraCrypt's plausible deniability and what forensic analysis can reveal about the actions taken by the user.

5.1. Installation of VeraCrypt

To conduct digital forensics, an examiner establishes a baseline of user activity and identifies deviations from the baseline, which act as evidence of activity. To identify activity linked to the usage of VeraCrypt, we must determine how VeraCrypt's interactions with the system deviate from the norm.

5.1.1. Registered GUIDs

Taking a snapshot of the registry and filesystem before and after installation of VeraCrypt will help identify drivers and GUIDs unique to VeraCrypt. As established in RFC 4122, a Globally Unique Identifier (GUID), or Universally Unique Identifier (UUID), was initially developed as a means of guaranteeing uniqueness across space and time (Leach, Mealling, & Salz, 2005). Given the specificity of these indicators, they can reliably be used to validate other instances of VeraCrypt use.

The installation of VeraCrypt registers two applications associated with VeraCrypt. When Windows registers the two applications, it generates three GUIDs for the VeraCrypt Format and the VeraCrypt application. The three GUIDs represent the Application ID (AppID) \ Class ID (CLSID), Interface, and Type Library (TypeLib). (Figure 1).

Registered Item	Type	Registered GUID

VeraCryptFormat.exe	AppID / CLSID	{A96D3797-9F31-49f4-A0CE-9657392CF789}
VeraCrypt.exe	AppID / CLSID	{FE8B3B95-C80C-41f7-830F-FBA271C26F7E}
ITrueCryptFormatCom	Interface	{7AB357D9-A17F-466E-BCD6-F49E97C218D8}
ITrueCryptMainCom	Interface	{C786E27C-2801-482C-B45D-D4357B270A29}
VeraCryptFormat	TypeLib	{56327DDA-F1A7-4E13-B128-520D129BDEF6}
VeraCrypt	TypeLib	{9ACF6176-5FC4-4690-A025-B3306A50EB6A}

Figure 1: Registered GUIDs after installation of VeraCrypt

Windows registers the above GUIDs for their corresponding objects. Additionally, Windows made some distinctive updates to the registry that are useful as indications of the installation of VeraCrypt.

5.1.2. Classes

All files with extension ".hc" are registered as VeraCrypt Volumes by default. The registry records this information under the following key:

HKEY_LOCAL_MACHINE\SOFTWARE\Classes\.hc\.

5.1.3. VeraCrypt Driver

The driver for VeraCrypt registers as VeraCrypt.sys. The path to System32\drivers\Veracrypt.sys can be found under registry key:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\veracrypt\.

5.2. Creating a Volume

The process for creating a volume did not leave any meaningful traces outside of evidence of execution. However, when a hidden volume is creating the outer volume is mounted to driver letter Z.

5.3. Mounting a Volume

For security mechanisms to be useful, they must also provide availability. This juxtaposition of purpose is a classic dichotomy that plays out in all levels of safeguarding

data. In the enterprise, it may manifest as IT working to maintain the availability of resources while security tries to restrict and protect access to valuable resources. For example, in daily life, password complexity and length increase security while making the password hard to remember. Similarly, password uniqueness challenges a user's memory by asking them to have a distinct password for each authentication. It should be no surprise then that the need for availability leads many people to reuse passwords and simplify their complexity.

Where encryption is involved, an encrypted container is only useful if it is accessible. A container is made accessible to the Windows operating system through a process called mounting. This section will explore different forensic artifacts generated as a result of mounting an encrypted container with VeraCrypt. Mounting will include three different scenarios: the standard volume (S), the outer volume of a hidden container (O), and the hidden container (H).

5.3.1. Storport Registry Key

There has been a trend among storage devices to use the Storport driver as a result of the increased efficiency it provides (Cowen, 2018). Storport.sys has been around since Microsoft Windows Server 2003 for use with high-performance buses such as fiber channel and RAID adapters (Hudek, 2017). Exploration of Storport Registry artifacts failed to identify any usage of this driver in the mounting of VeraCrypt volumes.

VeraCrypt has a custom driver aptly named veracrypt.sys, which is used to mount and access the contents of VeraCrypt encrypted volumes.

5.3.2. MountedDevices Registry Key

Plug and Play (PnP) is a concept that was introduced to the Windows ecosystem with Windows 95 (Roden & Jystad, 1995). The goal of PnP is to simplify the use of different hardware and software significantly without the user needing to make manual configuration changes (Hudek & Sherer, 2017). The MountedDevices Registry Key is an element of the plug and play framework which helps Windows track connected devices by universally unique identifiers (UUIDs). This tracking improves Windows usability by monitoring the required drivers and software of each device.

Unfortunately, every time a VeraCrypt volume is mounted, Windows generates a new UUID. Fortunately, VeraCrypt provides a convenient link from VeraCrypt's volume to the mounted drive letter (Figure 2). When a user selects to create the hidden container and its outer container simultaneously, the outer container is automatically mounted to drive letter (Z:\) and is registered in MountedDevices.

Container	Name	Value
Standard	\??\Volume{45467bcd-3107-11ea-9c8a-8c85909496fb}	VeraCryptVolumeS
Standard	\DosDevices\S:	VeraCryptVolumeS
Outer	\??\Volume{ea97ea6c-2ddf-11ea-9c88-8c85909496fb}	VeraCryptVolumeO
Outer	\DosDevices\O:	VeraCryptVolumeO
Hidden	\??\Volume{ea97eafa-2ddf-11ea-9c88-8c85909496fb}	VeraCryptVolumeH
Hidden	\DosDevices\H:	VeraCryptVolumeH

Figure 2: MountedDevices Evidence of Volume Mounting

The randomness of the UUID makes linking other registry artifacts to the same volume, which experiences reuse, significantly more difficult. This also means the variation of these GUIDs associated with mounted volumes has no impact on plausible deniability.

5.4. Accessing the Volume

Accessing the volume is straightforward. This action includes navigating to the newly mounted drive with Windows Explorer and the Command Prompt.

5.4.1. Shellbags

Shellbags is a well-known forensic artifact used to identify which folders a user has right-clicked on, opened, copied, moved with 'cut', or deleted (Duranec, Topolcic, Hausknecht, & Delija, 2019, p. 1416). Using Eric Zimmerman's Shellbags Explorer

v1.3.0, the drive letters discovered from the previous MountedDevices analysis can be confirmed (Figure 3).

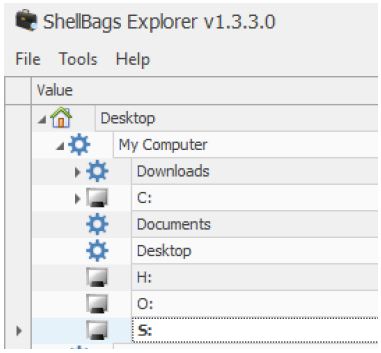

Figure 3: Shellbags Evidence of Volume Access

5.4.2. MountPoints2 Registry Key

The MountPoints2 registry key is a forensic artifact introduced with the concept of drive mapping. This registry key keeps track of drives that were mapped to the operating system and presented to the user with a drive label ("Drive Map," 2016). The most common appearance of a drive label is an alphabetical character with a colon and a slash, e.g., C:\. The mountpoints2 registry key improves usability by remembering the last name the user used to reference the mounted volume. If a user renames an external volume "personal files" when the device is presented by Windows as "personal files" in the future, that is a result of the mountpoints2 registry key. With regards to VeraCrypt, the MountPoints2 Registry key will provide the diver letter that is available in the MountedDevices registry key. Therefore, if the MountedDevices key says "VeraCryptVolumeZ", MountPoints2 will have a value of Z:\.

MountPoints2 only showed activity after the VeraCrypt containers were mounted multiple times. MountPoints shares the same weakness as MountedDevices; the system assigns random GUIDs to mounted volumes when the volumes are mounted.

5.5. Creating and accessing folders within the Volume

Creating and accessing folders within a volume leaves significant traces of user activity.

5.5.1. Shellbags

As previously mentioned in 5.4.1, Shellbags provides evidence of folder access and manipulation.

Shellbags activity of folder access in aggregate can defeat plausible deniability on its own if there is sufficient disparity in content, particularly if it is a significant disparity in the type of contact. For example, if a user suspected of terrorism provides access to a volume with folders labeled which indicate potentially compromising communications, and the Shellbags reveals folders named after attack plans or known terrorist groups and personalities, their plausible deniability is effectively defeated. Another example would be a suspected child pornography distributor, with a disclosed container containing embarrassing but not illegal pornographic material and Shellbags revealing folder names associated with child pornography.

This is exemplified in Figure 4, using the obvious folder names to identify the intent of the different volumes. The standard volume (S) for routine or 'boring' use, the outer volume (O) serves as the decoy to provide plausible deniability, and the hidden volume (H) contains the undisclosed folders which can compromise the user's plausible deniability.

Key-Value Chain	Recorded Folder Path
BagMRU\1\5\0-0	Desktop\My Computer\O:\Outer Folder\Decoy Folder
BagMRU\1\4-0	Desktop\My Computer\S:\Standard Folder\boring folder
BagMRU\1\6\0-0	Desktop\My Computer\H:\Hidden Folder\Undisclosed Folder

Figure 4: Shellbags 1.3.3.0 Recovered Paths after folder manipulation

5.5.2. Shell Link Binary (LNK) F iles

LNK files, or shortcuts, were introduced to Windows in July 2010 ("[MS-SHLLINK]: Shell Link (.LNK) Binary File Format," 2019). LNK files are known as shortcuts because they can be used to create a reference to a target file. LNK files are

ubiquitous within Windows. When installing an application, Windows will often ask if the user would like to add an icon to the desktop and the start menu. Both of these icons are LNK files that reference the installed executable. The key benefit of LNK files for a forensic examiner is Windows "Recent docs" or "Recent places". These user suggestions are drawn from a folder storing a significant number of LNK files that reveal actions a user has previously taken. Using Eric Zimmerman's Link File Parser LECmd, the file paths of the folder accesses within all three of our volumes can be confirmed.

A simple but powerful difference between LNK files and Shellbags is the recording of the Volume Serial Number (Figure 5). The Volume Serial Number is significant because it is a fixed value generated at the time of the volume's creation. The forensic examiner doesn't need to rely on the character of the content disclosed as compared to the content revealed by the folder name. The Volume Serial Number immediately identifies if the file that generated the LNK file originated within the decoy volume or not (Czeskis, 2008). Volume Serial Numbers can be edited with effort and technical skill, so it is not foolproof. Running a simple "dir" command in the root of a volume will return its Volume Serial Number.

LNK File Name	Path	Volume Serial Number
boring folder.lnk	S:\Standard Folder\boring folder	01DB-AAE8
Undisclosed Folder.lnk	H:\Hidden Folder\Undisclosed Folder	5904-7082
Decoy Folder.lnk	O:\Outer Folder\Decoy Folder	D0E4-0743

Figure 5: Results from LNK files in AppData\Roaming\Microsoft\Windows\Recent

5.5.3. Jump Lists

Windows 7 introduced new functionality via Taskbar Extensions (Satran, 2018). These taskbar extensions included Jump Lists. The intent of Jump Lists was to give a user quick access to the most recent files or applications that were accessed or executed through the application previously. When a user hovered over the application in the taskbar, the recent destinations would 'jump' up to give the user quick access to historical activity.

Structurally, and by extrapolation, forensically, this artifact bears striking similarities to the LNK file. Jump List artifacts share many of the LNK file's shell components. Using Eric Zimmerman's Jump List Parser LECmd, the file paths of the folder accesses within all three of our volumes can be confirmed.

In figure 6, the data parsed from the Jump List bears a striking resemblance to the data parsed from the LNK files and provides corroboration for LNK file data or an additional source of similar user activity in case the LNK files are deleted.

Jump List Item	Path	Volume Serial Number
16	S:\Standard Folder\boring folder\Standard Text.txt	01DB-AAE8
24	H:\Hidden Folder\Undisclosed Folder	5904-7082
21	O:\Outer Folder\Decoy Folder	D0E4-0743

Figure 6: Jump List data parsed from AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations\f01b4d95cf55d32a.automaticDestinations-ms

5.6. Creating and Accessing Documents in a Volume

As previously stated, the goal of these sections is to detail how specific actions interacting with an encrypted volume can expose activities that a user may intend to keep hidden. This section will cover the simple act of creating and editing a text file. Many of the forensic artifacts cited previously will yield a significant amount of information about user interactions with text files.

5.6.1. LNK Files

As previously mentioned in 5.5.2, LNK files provide a significant amount of information about file access. In the case of file access, LNK files identified the use of six different documents within the VeraCrypt volumes. As before, these LNK files contain the Volume Serial Number so that a forensic examiner can identify files accessed from the disclosed, outer container and other non-disclosed volumes. Additionally, if the

LNK files have different drive letters but the same Volume Serial Number, a forensic examiner can reasonably assume the user accessed the files from the same volume.

In figure 6, LNK files tie the individual file access within a particular volume directly to the volume serial number. This volume serial number can used as a means of identifying if the file is from the disclosed volume or as a means of differentiating the origins of the files from each other. For example, an examiner knows immediately that the "attack plans.rtf" is not from the same volume as "decoy-communication.txt" because the volumes serial numbers do not match. Likewise, running the dir command in the root of the disclosed container will provide an examiner the volume serial number and they will know which files were not accessed from the container.

LNK File Name	Path	Volume Serial Number
Standard Text.txt.lnk	S:\Standard Folder\boring folder\Standard Text.txt	01DB-AAE8
Taxes and stuff.rtf.lnk	S:\Standard Folder\boring folder\Taxes and stuff.rtf	01DB-AAE8
Secret badguy pass phrase.txt.lnk	H:\Hidden Folder\Undisclosed Folder\Secret badguy pass phrase.txt	5904-7082
attack plans.rtf.lnk	H:\Hidden Folder\Undisclosed Folder\attack plans.rtf	5904-7082
decoy-communication.txt.lnk	O:\Outer Folder\Decoy Folder\decoy-communication.txt	D0E4-0743
decoy-documents.rtf.lnk	O:\Outer Folder\Decoy Folder\decoy-documents.rtf	D0E4-0743

Figure 7: LECmd results from LNK files in AppData\Roaming\Microsoft\Windows\Recent

5.6.2. Jump Lists

As previously mentioned in section 5.5.3, Jump List contains similar data to LNK files, as seen in Figures 6 and 7. One significant difference not mentioned in 5.5.3 is that

Jump Lists also have a counter for the number of times accessed so that they can present jump files to the user according to their frequency of use.

Additionally, Jump Lists are grouped by application. In Figure 8, the results from file access of .txt files can be used to determine the difference in origin of the three files accessed as described in section 5.6.1. Figure 9 presents the results from file access of .rtf files from within the three different containers.

Jump List Item	Path	Volume Serial Number
12	S:\Standard Folder\boring folder\Standard Text.txt	01DB-AAE8
10	H:\Hidden Folder\Undisclosed Folder\Secret badguy pass phrase.txt	5904-7082
11	O:\Outer Folder\Decoy Folder\decoy-communication.txt	D0E4-0743

Figure 8: Jump List data parsed from AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations\9b9cdc69c1c24e2b.automaticDestinations-ms

Jump List Item	Path	Volume Serial Number
5	S:\Standard Folder\boring folder\Taxes and stuff.rtf	01DB-AAE8
3	H:\Hidden Folder\Undisclosed Folder\attack plans.rtf	5904-7082
4	O:\Outer Folder\Decoy Folder\decoy-documents.rtf	D0E4-0743

Figure 9: Jump List data parsed from AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations\469e4a7982cea4d4.automaticDestinations-ms

5.6.3. Recent File List

The Recent File List is a Windows Registry Key which provides evidence of file access organized by individual application. For example, this key recorded the files accessed within the VeraCrypt Volume at this registry path:

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Applets\Wordpad\Recent File List\

Similar to Shellbags, defeating plausible deniability with the Recent File List will require evaluating the content and character of the disclosed volume with the character of the file paths found in the Windows registry (Figure 10).

SubKey Name	SubKey Data
File1	S:\Standard Folder\boring folder\Taxes and stuff.rtf
File2	H:\Hidden Folder\Undisclosed Folder\attack plans.rtf
File3	O:\Outer Folder\Decoy Folder\decoy-documents.rtf

Figure 10: Results from Recent File List

5.7. Executing an application within a VeraCrypt Volume

5.7.1. UserAssist

The UserAssist registry key is a well-known forensic artifact that tracks a user's execution history to recommend popular applications the user uses. The registry key captures the full path of the applications executed, but encodes the path in a cipher known as ROT-13. The location for the UserAssist name decoded in Figure 11 is:

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}\Count\

Encoded Key	F:\Fgnaqneq Sbyqre\obevat sbyqre\WhzcYvfgRkcybere.rkr
Decoded Key	S:\Standard Folder\boring folder\JumpListExplorer.exe

Figure 11: Decoded Application Path found in UserAssist

5.7.2. Background Activity Moderator (BAM)

Available in Windows 10 after the Fall Creators Update – version 1709, the background activity moderator records application usage on a Windows operating system (Katsavounidis, 2018). One distinct difference with this artifact is that it links the application usage directly to a VeraCrypt Volume's mount point as provided in the MountedDevices registry key (Figure 12). In other words, it saves an examiner a lot of time and eliminates the potential for error.

Subkey Name
\Device\VeraCryptVolumeS\Standard Folder\boring folder\JumpListExplorer.exe

Figure 12: Results from HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-1056752351-4210658029-253887316-1000\

5.8. New Windows Artifacts

Windows is always pushing new features to improve the usability of its operating systems and digital ecosystem as a whole. This section will address the impact that some of these new features will have on forensic analysis of VeraCrypt usage. None of these features were available when Czeskis et al., initially published their analysis of the forensic weaknesses of TrueCrypt's plausible deniability

5.8.1. CloudStore

In Windows 10 1703, Microsoft began to transition data associated with roaming profiles away from the Local App Data folder in the filesystem and into the Windows registry (Rankin, 2017). Running VeraCrypt created the registry entry:

HKCU\Software\Microsoft\Windows\CurrentVersion\CloudStore\Store\Cache\DefaultAccount\$de${8d9be71d-53c7-48d5-82bc-5367bcefc513}$$windows.data.unifiedtile.localstartvolatiletilepropertiesmap\Current\IDRIX.VERACRYPT

To defeat plausibility in a single pass scenario, CloudStore simply provides evidence of VeraCrypt execution. However, in an enterprise environment where these changes to a roaming profile might be logged and analyzed, identifying usage of VeraCrypt could indicate a possible insider threat.

5.9. Reinstallation of Windows

The previous sections have all covered leakage from above, and the artifacts included are by no means comprehensive. There are many other registry keys and operating system features capable of corroborating or revealing user activity within VeraCrypt volumes. Additionally, there will be discoveries of current and future features that leave traces of information that reveal user activity. In short, the volume of data shown by the operating system alone presents a serious threat to a user's plausible deniability.

This risk of leakage from above is why the TrueCrypt Foundation architected the implementation of the hidden operating system the way they did. The instruction in the setup of a hidden operating system to wipe the original installation of Windows removes all of the evidence of leakage from above on the local machine. The TrueCrypt Foundation's implementation of only mounting non-hidden drives as read-only while in the hidden operating system ensures that the hidden OS does not leak to the outer operating system going forward.

6. Advanced Forensic Analysis

The TrueCrypt Foundation's implementation of wiping the outer operating system and implementing read-only mounts in the hidden operating system presents a significant challenge. The challenge means forensic examiners cannot rely on leakage from above alone, but must also explore leakage from the application. As previously discussed Kedziora, et al., attacked the application through entropy analysis of the sectors of the outer volume and identified dips in the entropy corresponding to the beginning and end of the hidden volume (2017). The approach in this research will be to identify potential anomalies within the memory of the application.

When the TrueCrypt Foundation implemented the hidden container, they encountered a difficult problem. The premise of plausible deniability is that the user's encrypted activity in the outer volume is ready for disclosure. The activity in the outer volume then must be current. However, frequent usage of the outer volume by the user will increase the likelihood of their activity overwriting parts of the hidden volume and possibly corrupting it.

This need to provide access to the outer volume while removing the potential to corrupt the hidden volume lead to the TrueCrypt Foundation introducing a key feature. This feature is the ability for the user to provide the passwords to the outer volume and the hidden volume simultaneously in order to open the outer container and protect the hidden container. The user providing both passwords and the application opening the outer volume while protecting the inner volume must leave some traces in the memory of the application.

6.1.1. Memory Forensics

The most important feature of memory for the sake of forensics is the fact that memory is unavoidable for applications. Anything an application passes to the Central Processing Unit (CPU) must be passed to memory first. The CPU does not read directly from disk.

Therefore, if VeraCrypt is providing continuous protection to the hidden volume while the outer volume is mounted and accessible, something must reflect this different state of the application in memory.

The comparison of a memory dump of VeraCrypt protecting the hidden volume to a memory dump while VeraCrypt was not protecting the hidden volume yielded some observations. These observations have not been analyzed to determine their root cause, but they are potentially indicative of the different state of the application we are trying to identify. In Figure 13, the two key anomalies observed are described along with their offsets.

Offset	Observation

0x41AC	The size byte for the proceeding data is increase by 24, and an additional 24 bytes of data is found before memory segments synchronize again
0xAE2A	An additional entry for MSFTEdit.ddl is present in what appears to be a library table

Figure 13: Observations during memory analysis

Unfortunately, these results were unable to be consistently replicated when testing against different volume sizes and file system formats. As a result, these observations are unable to serve as a signature identifying hidden volume protection. Even if signatures are identified in the future, application to the single-access scenario will remain a challenge. The ability to find the signatures in non-volatile memory caches, like the hibernation file or memory dumps, will need to be supported.

7. Recommendations and Implications

This research provides examples of how to defeat the plausible deniability of VeraCrypt's hidden volume by performing standard Windows forensics. Additionally, this paper explores an approach to potentially defeat the plausible deniability of VeraCrypt's hidden operating system by leveraging the memory forensics of the outer operating system. A few key take-aways from this analysis are:

- VeraCrypt has its own distinct driver
- VeraCrypt’s driver clearly states the volume letter in MountedDevices
- The volume serial number of VeraCrypt volumes changes for each volume
- Background Activity Moderator directly links application usage from within the volume to the VeraCrypt Volume

7.1. Recommendations for Practice

The most significant and surprising take away from this research is the unsung utility of the Windows Background Activity Moderator.

7.1.1. VeraCrypt.sys

VeraCrypt's custom driver means any indications of the driver's presence or use defeat a user's deniability of knowledge or utilization of VeraCrypt. Therefore, forensic analysis can identify VeraCrypt activity performed by a user even if the said user attempts to conceal the VeraCrypt application by running it from a portable device or another protected location.

7.1.2. VeraCryptVolume<DriveLetter>

The inclusion of the drive letter used to mount the decrypted volume in the MountedDevices registry key is a crucial underpinning for any analysis of user activity associated with VeraCrypt volumes. This drive letter serves as the root of all paths associated with file access, folder access, and application execution. If the activity is recent, timeline analysis can definitively link specific user activity to a VeraCrypt volume. If the activity is older, then it must be noted that this association of activity to a VeraCrypt volume becomes tenuous. For example, in a scenario where a user mounts their VeraCrypt volume to the D or E drive and uses CDs, DVDs, and removable media user activity associated with the drive letter can easily be associated with any of the above devices instead of the VeraCrypt volume.

7.1.3. Volume Serial Number

It appears that VeraCrypt uses standard filesystem formatting when it creates the filesystem within an encrypted container. Consequently, when it creates a volume, a semi-unique volume serial number is assigned to the volume based on the filesystem type. If the filesystem selected is NTFS or FAT, then a forensic examiner has a frequently used and reliable means of corroborating file access using the volume serial number. More importantly, for defeating plausible deniability, if a user attempts to conceal the use of a hidden volume by mounting it to the same drive letter as the outer volume, the volume serial numbers will be different. A forensic examiner will know the user is using either two different volumes or a hidden volume.

7.1.4. Background Activity Moderator (BAM)

The background activity moderator addresses the path to an executed application using the hardware device instead of the drive letter. This path with the hardware device means that BAM records attribution directly to a VeraCrypt volume and avoids any ambiguity caused by mixed-use drive letters.

7.2. Implications for Future Research

7.2.1. Background Activity Moderator (BAM)

The utility of the path stored by the Background Activity Moderator is ripe for analysis by malware analysts and other high-technology crime forensic examiners. Malware authors and bad actors are constantly trying to find concealed locations to run applications from. It would be interesting to see how the hardware device path would reflect some of these concealed locations.

7.2.2. Memory Analysis

This research was unable to identify any potential signatures that can distinguish the standard use of VeraCrypt versus using its functionality to protect the hidden volume. Identifying such a signature would have been a very significant find for the digital forensics community and extremely damaging to the plausible deniability of the use of VeraCrypt's hidden volume and its hidden operating system.

8. Conclusion

In closing, defeating the plausible deniability of VeraCrypt's hidden volume is feasible with forensic analysis of the Windows operating system. Especially for recent activity, which generates LNK files and JumpList entries. Overcoming the plausible deniability of VeraCrypt's hidden operating system is also possible using entropy sector hashing, as presented by Kedziora. Yet, a technique using memory analysis remains elusive.

Michael Smith, mesmith1@protonmail.com

References

[MS-SHLLINK]: Shell Link (.LNK) Binary File Format. (2019, February 14). Retrieved from https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-shllink/16cb4ca1-9339-4d0c-a68d-bf1d6cc0f943

Balogun, A., & Zhu, S. (2013). Privacy Impacts of Data Encryption on the Efficiency of Digital Forensics Technology. International Journal of Advanced Computer Science and Applications, 4(5), 36-40. Retrieved from https://arxiv.org/pdf/1312.3183v1.pdf

Bar-El, H. (2014, May 30). *The status of TrueCrypt*. Retrieved December 19, 2019, from http://www.hbarel.com/analysis/itsec/the-status-of-truecrypt

Canetti, R., Dwork, C., Naor, M., & Ostrovsky, R. (1997, May 17). *Deniable Encryption*. Retrieved from https://link.springer.com/content/pdf/10.1007/BFb0052229.pdf

Chakraborti, A., Sion, R., & Chen, C. (2017). DataLair: Efficient Block Storage with Plausible Deniability against Multi-Snapshot Adversaries. Privacy Enhancing Technologies. Retrieved from https://arxiv.org/abs/1706.10276v2

Cowen, D. (2018, November 12). *Daily Blog #536: USB 3.0 External Storage Drive Forensics: Changes in registry locations*. Retrieved from https://www.hecfblog.com/2018/11/daily-blog-536-usb-30-external-storage.html

Czeskis, A., St. Hilaire, D. J., Koscher, K., Gribble, S., Kohno, T., & Schneier, B. (2008). Defeating encrypted and deniable file systems: TrueCrypt v5.1a and the case of the tattling OS and applications. Proceedings of the 3rd conference on Hot topics in security, 1-7. Retrieved from https://www.usenix.org/legacy/event/hotsec08/tech/full_papers/czeskis/czeskis.pdf

Davies, A. (2014). *A Security Analysis of TrueCrypt: Detecting hidden volumes and operating systems* (Master's thesis, Royal Holloway, University of London, Surrey, United Kingdom). Retrieved from http://www.ma.rhul.ac.uk/static/techrep/2014/RHUL-MA-2014-10.pdf

Michael Smith, mesmith1@protonmail.com

Drive Map. (2016, August 1). Retrieved from https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/dn581924(v=ws.11)#drive-letter-5

Duranec, A., Topolcic, D., Hausknecht, K., & Delija, D. (2019). Investigating file use and knowledge with Windows 10 artifacts. *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. doi:10.23919/mipro.2019.8756877

Hargreaves C., Chivers H. (2010) Detecting Hidden Encrypted Volumes. In: De Decker B., Schaumüller-Bichl I. (eds) Communications and Multimedia Security. CMS 2010. Lecture Notes in Computer Science, vol 6109. Springer, Berlin, Heidelberg

Hudek, T., & Sherer, T. (2017, June 15). Introduction to Plug and Play. Retrieved from https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/introduction-to-plug-and-play

Hudek, T. (2017, April 19). History of Storport. Retrieved January 3, 2020, from https://docs.microsoft.com/en-us/windows-hardware/drivers/storage/history-of-storport

Katsavounidis, C. (2018, February 26). An Alternative to Prefetch -> BAM [Web log post]. Retrieved from https://www.linkedin.com/pulse/alternative-prefetch-bam-costas-katsavounidis/

Kedziora, M., Chow, Y., & Susilo, W. (2017). Defeating Plausible Deniability of VeraCrypt Hidden Operating Systems. *Applications and Techniques in Information Security*, 3-13. doi:10.1007/978-981-10-5421-1_1

Kedziora, M., Chow, Y., & Susilo, W. (2017). Improved Threat Models for the Security of Encrypted and Deniable File Systems. *Mobile and Wireless Technologies 2017*, 223-230. doi:10.1007/978-981-10-5281-1_24

Kornblum, J. (2008). *cryptoscan.py*. Retrieved December 17, 2019, from http://jessekornblum.com/tools/volatility/cryptoscan.py

Leach, P., Mealling, M., & Salz, R. (2005, July 1). *RFC 4122 - A Universally Unique IDentifier (UUID) URN Namespace*. Retrieved from https://tools.ietf.org/html/rfc4122

Michael Smith, mesmith1@protonmail.com

Rankin, J. (2017, June 14). Roaming profiles and Start Tiles (TileDataLayer) in the Windows 10 1703 Creators Update [Web log post]. Retrieved from https://4sysops.com/archives/roaming-profiles-and-start-tiles-tiledatalayer-in-the-windows-10-1703-creators-update/

Roden, T. A., & Jystad, G. E. (1995, September 1). Plug and Play Run-Time Services. Retrieved from https://www.drdobbs.com/windows/plug-and-play-run-time-services/184409623

Skillen, A., & Mannan, M. (2014). Mobiflage: Deniable Storage Encryption for Mobile Devices. IEEE Transactions on Dependable and Secure Computing, 11(3), 224-237. Retrieved fromhttps://www.ccsl.carleton.ca/~askillen/publications/Mobiflage-NDSS13.pdf

TrueCrypt Foundation. (n.d.). Final Release. Retrieved from http://truecrypt.sourceforge.net

TrueCrypt Team. (2004, February 2). *TrueCrypt User's Guide, version 1.0*. Retrieved December 19, 2019, from https://alt.security.scramdisk.narkive.com/0PPFbtws/copy-of-truecrypt-user-manual

VeraCrypt Free Open source disk encryption with strong security for the Paranoid. (n.d.). Retrieved December 22, 2019, from https://www.veracrypt.fr/en/Home.html

Zimmerman, E. (n.d.). *ShellBags Explorer v1.3.3.0*. Retrieved January 8, 2020, from https://ericzimmerman.github.io/#!index.md

Zimmerman, E. (n.d.). *LECmd v1.3.3.0.* Retrieved January 8, 2020, from https://ericzimmerman.github.io/#!index.md

Michael Smith, mesmith1@protonmail.com