Report Title: Final Report Cloud Forensics Tool Prototyping

Award Number: 2012-MU-CX-K002

Author: Victor Fay-Wolfe **Date**: December 2015

Abstract. Cloud computing, where applications and data storage are provided as services to users via the Internet, is becoming more and more prevalent - and because of it, state and local law enforcement investigators are facing new challenges in obtaining evidence. Instead of the evidence being on a device that they can seize, the evidence is likely located in data centers operated by a service provider. These data centers are often not geographically easily accessible and may even be in multiple locations (and jurisdictions) across the world. The problem is particularly acute for State and local law enforcement investigators where extensive traveling to obtain evidence is not feasible. Furthermore, the volume of data kept by these service providers is so vast that it is often impractical for an investigator armed with a warrant to extract the evidence from the data centers of most service providers, even if he/she were physically present.

The most practical approach for State and local law enforcement when cloud computing has been used is to execute a warrant through the service provider's Keeper of Records - and require the service provider to deliver the evidence.

This project augments the *Cloud Signature*, a tool produced on another DoJ project, that allows law enforcement investigators to quickly obtain the specifics of what to request in a warrant to a cloud application service provider. It extends it with two new tools: *Forensics SteadyState*, which allows the investigator to ensure his/her forensic workstation is free of previous cloud remnants and other potential forensic contaminants; and *Cloud Signature Creator*, which creates the cloud application signatures for *Cloud Signature*. All three tools have been prototyped, tested with law enforcement, and disseminated free to law enforcement.

Table of Contents

Executive Summary

Cloud computing, where applications and data storage are provided as services to users via the Internet, is becoming more and more prevalent - and because of it, state and local law enforcement investigators are facing new challenges in obtaining evidence. Instead of the evidence being on a device that they can seize, the evidence is likely located in data centers operated by a service provider. These data centers are often not geographically easily accessible and may even be in multiple locations (and jurisdictions) across the world. The problem is particularly acute for State and local law enforcement investigators where extensive traveling to obtain evidence is not feasible. Furthermore, the volume of data kept by these service providers is so vast that it is often impractical for an investigator armed with a warrant to extract the evidence from the data centers of most service providers, even if he/she were physically present.

The most practical approach for State and local law enforcement when cloud computing has been used is to execute a warrant through the service provider's Keeper of Records - and require the service provider to deliver the evidence.

This project augments the *Cloud Signature*, a tool produced on DoJ project 2011-FD-CX-K011, that allows law enforcement investigators to quickly obtain the specifics of what to request in a warrant to a cloud application service provider. It extends it with two new tools: *Forensics SteadyState*, which allows the investigator to ensure his/her forensic workstation is free of previous cloud remnants and other potential forensic contaminants; and *Cloud Signature Creator*, which creates the cloud application

signatures for Cloud Signature. All three tools have been prototyped, tested with law enforcement, and disseminated free to law enforcement.

This report is formatted by combining reports on the two new tool prototypes: *Forensics SteadyState (*Part 1 of this report) and *Cloud Signature Creator* (Part 2 of this report). Each part covers the problem, goals, background, methodology, results, conclusions, and dissemination for its particular tool. The final report for project 2011-FD-CX-K011 does a similar thing for the original *Cloud Signature* tool.

Part 1 Forensics SteadyState

1 Introduction – Forensics SteadyState

. How does law enforcement collect and analyze cloud-based remnants on devices in a way that preserves the probative value of digital evidence?

The digital forensic process is comprised of four main steps: seizure, acquisition, analysis, and documentation. Digital forensic investigations involve developing and testing hypotheses made about the present state of a computer. Law enforcement must seize any device they feel is necessary that may contain digital evidence. These evidence items include computer hard drives, cell phones, routers, switches, gaming systems, or any other electronic device that stores digital data.

Once evidence items are seized, they are acquired using tools installed onto a *forensic workstation*. Forensic workstations are often Windows or Linux machines that have software to prevent any original evidence from being altered in any way. During the acquisition phase, an exact bit-for-bit copy of the evidence is made in the form of an image file preserving the original state of the evidence. After the acquisition phase, evidence is analyzed using forensic software that observes the state of the evidence image. During analysis, investigators use this forensic software to search through the image to find information that either supports or refutes the hypotheses made for the investigation. It is important that all of these procedures are performed on a controlled, sterile environment that will not alter the original evidence in any way. Finally, once the

evidence image has been analyzed and the hypotheses made about the state of the computer has been made, all findings are documented and reported.

1.1 Statement of the Problem Forensics SteadyState

The purpose of this project is to improve an existing SteadyState™ replacement solution to create a *Forensics Steady State* tool for newer Microsoft Windows operating systems and to verify its forensic integrity so that it can be used by law enforcement investigators. The result is a controlled, forensically-sound, Windows 7 environment that can be reproduced, re-used, and is free from cross-contamination.

1.2 Justification for the Study - Forensics SteadyState

When a forensic investigator analyzes digital data on a forensic workstation, he/she must be careful to maintain the probative value of the resulting evidence by being able to demonstrate that their investigation did not corrupt the evidence in any way. A primary concern is *cross-contamination*, where digital evidence from one case that the investigator worked makes its way into another case, and in doing so undermines the probative value of the evidence. Evidence from a previous case can include, but is not limited to: pictures, documents, applications, email, and viruses. To prevent cross-contamination, most law enforcement agencies seek to have a controlled operating environment (e.g. the forensic workstation's operating system and installed files) that can be reset to a sterile state which ensures that all remnants of previous cases are not present.

Investigators often use ad hoc methods for establishing a controlled operating environment. One method is to wipe the forensic workstation's systems disk and re-

install the operating systems and tools completely before each investigation. Another method is to use a master image of the operating environment from which they clone a new investigation environment. While these methods are commonly used, they present serious limitations. The first method is extremely time consuming and can take upwards of an entire business day to prepare a new forensic environment. The second method can cause update lag, where new versions of tools are not updated in the master image for long periods of time due to the efforts required to produce a new image.

In addition to these ad hoc techniques, there have been some automated techniques, such as Microsoft Corporation's *Windows SteadyState™*, a free, simple, elegant solution to the controlled environment problem. It allows administrative users to protect their systems from viruses, malware, and unwanted application installations by tracking all changes to a machine's current state, and discarding them whenever the administrator chooses to do so. Upon discarding the changes, the machine is returned to its original state. Every change made to the system since its last save point is deleted. Windows SteadyState™ has proven to be an effective tool for forensic investigators. Unfortunately, Windows SteadyState™ has been phased out since December 31, 2010 with Windows Vista being the last operating system supported by the solution [Microsoft Support]. This has left forensic investigators without a viable automated solution for ensuring a controlled environment that protects the probative value of digital evidence that is compatible with Windows 7 and future operating systems.

The goals of this research project are as follows:

1. To make a controlled environment solution that ensures that a sterile digital forensics environment can be created each time a new case is started by law enforcement investigators.
2. To make a controlled environment solution that is easy for forensic practitioners to use.
3. To make a controlled environment solution that does not substantially delay investigations.
4. To have a solution that does not interfere with the forensic process.
5. To document the controlled environment solution behaviors proving forensic readiness.
6. The reboot process of the solution should automate the roll back procedure and boot directly into a Windows environment after completion, as required by the many law enforcement organizations.

1.3 Accomplishments - Forensics SteadyState

This project documented the controlled environment solution behaviors through extensive testing to prove forensic readiness of Forensics Steady State. The outcomes from this project ensured a stable, sterile digital forensics environment that does not substantially delay investigations or interfere with the forensic process. Forensics Steady State also has ease of use for forensic practitioners including the added function of the automatic roll back.

2 Literature Review - Forensics SteadyState

This section provides a survey of current solutions to the proposed problem presented in section 1.1 that support modern Microsoft operating systems. One solution, Faronics' Deep Freeze, is a commercial product that is marketed to preserve the state of a host operating system environment for general use. A community supported replacement for the original Microsoft SteadyState™ provides techniques for users to develop their own solution. Horizon DataSys' Drive Vaccine is also briefly explained.

2.1 Faronics' Deep Freeze

Faronics' *Deep Freeze* [Faronics] is software that can restore Windows, Mac OS, and Linux operating systems back to an original state that is pre-defined by the user. It can function as a SteadyState™ replacement, but previous research on Windows machines has shown that Deep Freeze is not compatible with the requirements of digital forensic investigations [Fonseca]. Problems exist with saving case information to external drives and removing external drives during the computer's operation. Older versions of Deep Freeze have been shown to crash Windows 7 computers when evidence hard drives are attached internally and removed after the imaging process when using drive trays.

Previous versions of Deep Freeze are also insufficient in that files can appear to be written to external drives, which are transparently locked, creating problems for retaining investigators' case information. We tested the newest version of Deep Freeze Enterprise and discovered that this problem still persists. There are options to add certain drive letters to a "whitelist" during the initial creation of a Deep Freeze solution that will allow those drives to be written to without write-protection. This option is not

easily accessible and the illusion of seeing the files copied to a destination drive, when the action does not actually occur, can be problematic for law enforcement investigations.

2.2 SteadyState™ for Windows 7

Currently, there is no formal SteadyState™ solution for Windows 7 provided by Microsoft. In July of 2001, Panos Macheras, a Microsoft developer, released his methodology for creating a Windows 7 SteadyState™ like tool for use in internet cafés, educational computer laboratories, and other establishments that use Windows 7 workstations [Macheras]. Macheras' procedure claims to work on any machine, but through initial testing we have concluded that it is not an adequate steady state replacement. Using Macheras' methodology to create the Windows 7 Steady State replacement, we were not able to create a working Steady State. The inability to reproduce Macheras' work fails to provide a solution for goal 1 of this project. Due to the fact that a working version of Macheras' solution could not be created, the solution also fails to meet goals 2-4 of this research project as well.

2.3 Horizon DataSys' Drive Vaccine

Horizon DataSys' *Drive Vaccine* is another application designed specifically for Microsoft Windows that shares similar functionality to Windows SteadyState™. Drive Vaccine operates under Microsoft Windows and has the capability to rollback to previous baseline images if a system becomes corrupt with any form of system error preventing normal operation. Drive Vaccine could be considered a viable option for digital forensic investigations, but it has not been forensically validated.

2.4 Steadier State

In 2012 Mark Minasi, of MR&D, released his own open source version of a SteadyState™ replacement for Windows 7 which is named *Steadier State* [Minasi]. *Steadier State* uses a technique called *differencing disks* to make the SteadyState™ solution function on a Windows 7 Enterprise or Ultimate machine that takes advantage of Windows 7's ability to boot from Virtual Hard Disks.

Virtual Hard Disk (VHD) files are virtual representations of hard disks that provide the same functionality as a regular hard disk drive. VHDs encapsulate hard disk images that can contain partitions and file systems specific to the operating system installed into the virtual disk file. VHDs were originally created for use as storage media for virtual machines that are running in Windows Virtual PC, Windows Virtual Server, or Hyper-V. Windows 7 Enterprise and Ultimate editions and Windows 8 now have native support for booting from VHD files eliminating the need for a hypervisor. When natively booting a VHD, performance is greatly enhanced and there is improved support for Operating System features that are not available in a hypervisor such as Windows Virtual PC [Calvert, 2009].

There are three different kinds of VHD formats that can be created: fixed, dynamic, and differencing. Fixed sized VHDs are a static size that is stored on a physical storage device when the virtual file is created. The size of fixed VHDs cannot be decreased, but

can be increased when the file is disconnected and able to be edited [Jain, 2010]. Dynamic VHDs only use as much space on the physical storage device as needed to store the data in the file and can expand as new blocks in the virtual disk are used. Typically, dynamic VHDs have slower read/write performance than fixed disks [Jain, 2010].

Differencing VHDs are comprised of two or more components, a parent VHD and one or multiple child VHD(s). Any child VHD files are linked to the parent and represent the current state of the VHD as a set of modified blocks in comparison to its parent [Jain, 2010]. The parent VHD can be either fixed or dynamic in relation to its differencing child VHD. Differencing VHDs are analogous to creating snapshots of a virtual machine when using a hypervisor.

Steadier State utilizes the differencing disk technology in order to recreate a Windows 7 SteadyState™. The baseline image.vhd acts as the parent VHD and snapshot.vhd acts as the child VHD caching any/all writes made when natively booted into the VHD file. Currently, no explicit documentation has been released by Microsoft that explains exactly how a machine boots into a VHD using Windows 7 Enterprise/Ultimate native boot to VHD ability.

To implement Steadier State, users must follow a detailed course of action (see Figure 1). First, users need to create a boot disk using tools from the Windows Automated Installation Kit (WAIK) and scripts provided by the Steadier State package. A media disk is connected to a prepared target Windows computer and a VHD image of the prepared computer is created onto the media drive using tools provided by the boot disk. The resulting image is a VHD file that will be used as the basis for the new machine

operating system. Once that image is created, the hard disk containing the original operating system is wiped and prepared, and the VHD file is copied to the disk. The disk now contains the VHD file that the PC boots into. For a fully detailed procedure of the process flow, including specific scripts for creating and deploying Steadier State, see Figure 1.

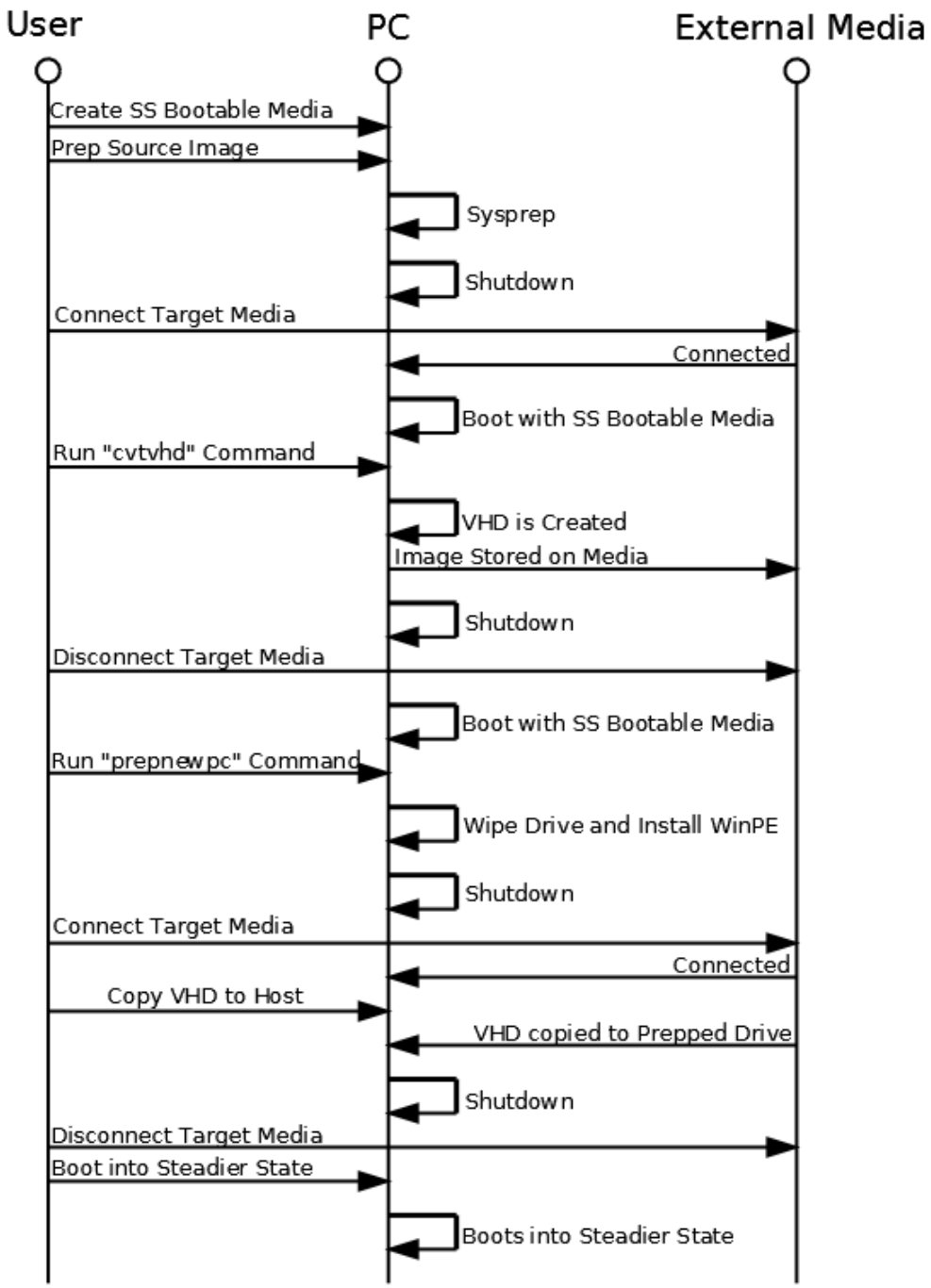

Figure 1 - Process Flow Diagram: Creating and Deploying Steadier State

When the user restarts the computer, two options are presented. The selected option will determine which of several states that Steadier State could be put into (see Figure 2). The default first option boots into the Windows 7 environment caching all writes into a snapshot.vhd file. The second option rolls back the system deleting the existing snapshot file containing all written changes to the disk made since the last rollback. A new empty snapshot file is then created and the system returns to its original state. If the user chooses to update software or make any changes to the system permanent, they must first place an empty text file named "noauto.txt" on the root of the D drive. When the user reboots the computer, selecting the roll back option opens the WinPE environment and pauses at a command line prompt. The user can then choose to run a merge script, provided by Steadier State, to accept permanent changes to the base image by merging the snapshot and parent image VHD files and creating a new empty snapshot file to cache future writes.

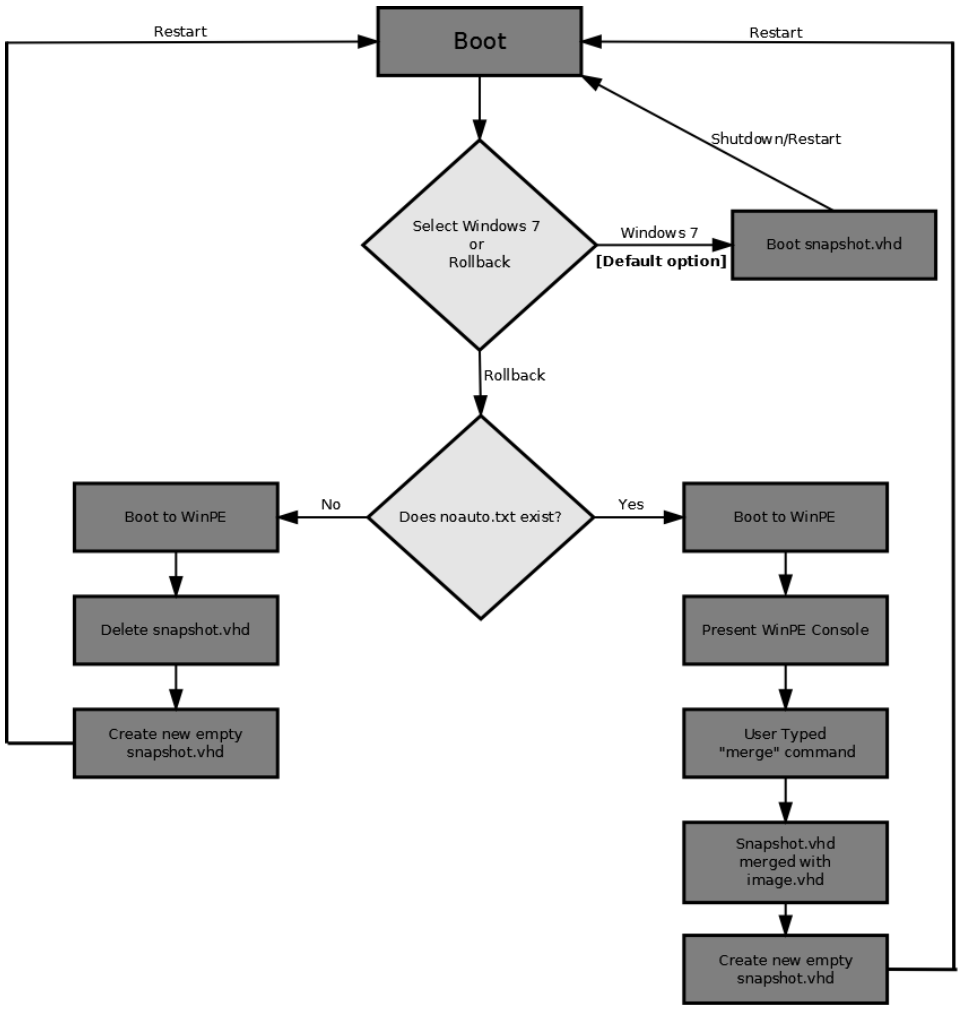

Figure 2 – Flow Chart for Steadier State

During the initial testing of Steadier State, the solution seemed promising, but was also too difficult for the general law enforcement user failing to meet the requirements of Goal 2. Furthermore, it has not been extensively tested for forensic validation, a necessity for law enforcement, failing to meet the requirements of Goal 5. Steadier State served as the base implementation on which Forensics Steady State is based to improve the solution and meet the goals of this project. Section 3.1 elaborates on the extensive

testing that has been conducted as part of the forensic validation of Forensics Steady State.

3 Methods - Forensics SteadyState

This section describes how this project created the Forensics Steady State tool to meet the goals of Section 1. It first outlines how this project thoroughly tested the existing Steadier State solution on which the implementation of Forensics Steady State is based. Section 3.2 describes possible future enhancements to Forensics Steady State.

Forensics Steady State was implemented using the existing files and command scripts provided by Steadier State with modifications. The original boot process of Steadier State provides the user with two boot options every time the machine is restarted with Windows 7 is always being selected as the default boot option when the pre-boot environment is displayed. Selecting the Windows 7 option boots the system into snapshot.vhd which will contain all changes and writes that currently reside on the image (Figure 2). After restarting or shutting down the machine, the user is always presented with the options to “Roll Back Windows” or “Windows 7”. This project modified Steadier State to change the default boot process and add in the ability to scan the D drive for extraneous files.

The modifications to Steadier State included adding commands into some of the provided command scripts to change the default boot order. Goal 6 specifies that the law enforcement organizations require the reboot process of their forensic workstations to automate the roll back procedure and boot directly into Windows after completion, without requiring any user input. In order to accomplish this task, the default boot entry

was changed from "Windows 7" to "Roll Back Windows" by modifying the system's Boot Configuration Data (BCD) store. The *Boot Configuration Data* store is a database file that contains the boot configuration for all bootable devices/partitions [Technet, 2007]. When the "prepnewpc" command is run during the Steadier State creation process (Figure 1), the hard drive that will contain Steadier State is wiped and partitioned to contain a copy of WinPE. Within the command script of "prepnewpc", commands were added that store the globally unique identifier (GUID) for the bootable WinPE partition after the storage device is prepared. Then, commands were added to the "rollback.cmd" script that stores the default GUID in a file named "defaultguid.dat" which is saved to the same partition that stores both image.vhd and snapshot.vhd. One last command was added to both the "rollback.cmd" and "merge.cmd" scripts that reads the GUID from "defaultguid.dat" and sets the Windows 7 boot option as the default for one time after a successful rollback or merge operation is performed.

Now, when the user restarts the computer, "Roll Back Windows" is the default boot option (Figure 3). The option to boot directly into the Windows 7 environment is still present in the pre-boot environment. Once the solution is rolled back, the BCD is modified to automatically boot into Windows 7 after the baseline image is restored. Essentially, a user could simply shutdown or restart their Forensics Steady State solution and have a pristine Windows 7 image restored without any further user interaction.

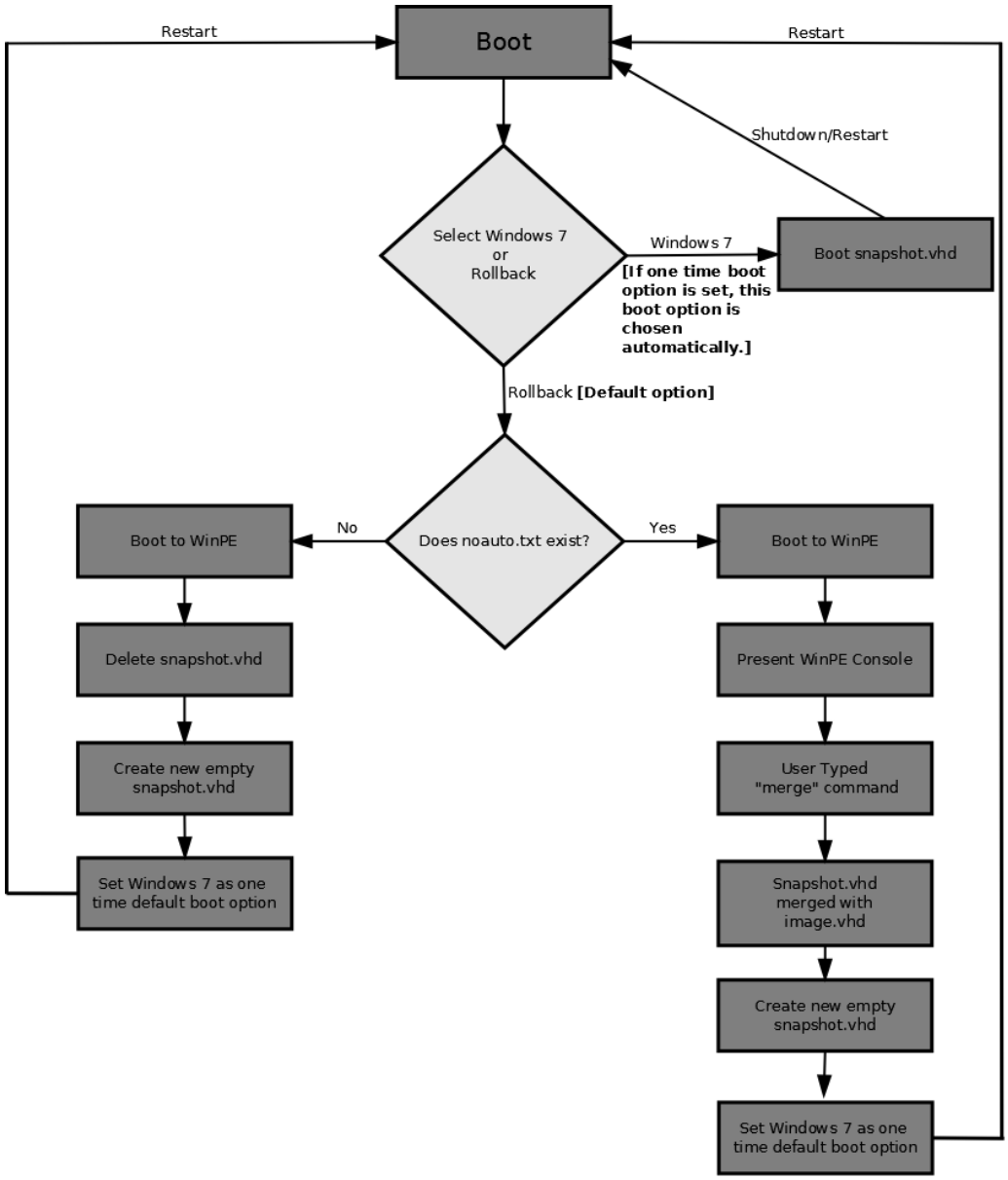

Figure 3 – Flow Chart for Forensics Steady State

Additional functionality was added into "rollback.cmd" that recursively scans the partition containing image.vhd, snapshot.vhd, and defaultguid.dat for extraneous files. If any files are found, the user is prompted with a notification that additional files reside on the drive that should be removed before beginning a new case.

In summary, the following additions were made to Steadier State to create Forensics Steady State:

- Automatic roll back upon shutdown or reboot of system without requiring user input.
- Recursive scanning of physical drive for extraneous files and recommendation for deletion before beginning a new digital forensic investigation.

3.1 Initial Steadier State Testing - - Forensics SteadyState

The first step of this project was to thoroughly test Forensics Steady State using the goals listed in section 1.

3.1.1 Test Goal 1 - Forensics SteadyState

This test determined if *Forensics Steady State ensures that a sterile digital forensics environment can be created each time a new case is started by law enforcement investigators*, as stated by Goal 1. The test started by determining if the stated functionality of Forensics Steady State is reliably achieved. It should meet its claims that:

- The .vhd file created by *Forensics Steady State* is never written to unless a merge script is executed.

- All writes to the disk are placed within the snapshot.vhd file created every time *Forensics Steady State* rolls back.
- The image.vhd file will never become changed. Physically changing the bytes of an image with the use of a hex editor, the image's bytes should always remain persistent and never change.

I created an image of a forensic workstation and performed tests that included:

- Adding files and folders to a Windows 7 forensics workstation, rolling back the machine, and confirming all changes were deleted.
- Installing applications to Windows 7, rolling back the machine, and confirming all changes were deleted.
- Changing individual bytes on both the image.vhd (baseline image) file and snapshot.vhd file to confirm that once the system is rolled back, all changes were not kept and the original baseline image was intact.
- Test the environment extensively with all of the most important digital forensic tools currently used by the Rhode Island State Police and other law enforcement agencies to ensure their proper functioning. These tools include X-Ways' Forensics, Guidance's EnCase, AccessData's Forensics Toolkit, and ForensicSoft's SAFEBlock.
- Test disk overflow behavior where more files are written to the environment than specified.

- Test the ability to recognize external disk drives that are connected to the forensic workstation and the stability of hot-swapping external media.

To confirm that a sterile environment is achieved in all of these tests, an MD5 hash of the image.vhd file will be taken to ensure it returned to its original state. In order to test assumptions about the system, bytes contained within the .vhd file were manually changed to test both "Roll back" functionality and write-blocking capabilities.

3.1.2 Test Goal 2 - Forensics SteadyState

This test determined if *Forensics Steady State is easy for law enforcement investigators to use*, as stated by Goal 2. In order to test the ease of use of the Steadier State solution, I compared it to other comparable products Deep Freeze and Pacheras' Steady State solution using the criteria:

- The time and process of reverting back to baseline images using each product.
- The time and process of updating the solution and retaining changes as a new baseline image for each product.
- The process of keeping changes temporarily for each product.

3.1.3 Test Goal 3 - Forensics SteadyState

This test determined if *Forensics Steady State did not substantially delay investigations*, as stated by Goal 3. In order to test this goal, several aspects of re-booting the solution were tested:

- For an on-going investigation, officers need the ability to turn off their workstations without "rolling back" the machine and have their current analysis saved. The re-boot time of the Forensics Steady State machine was timed and compared to a normal re-boot of a similar Windows 7 machine.
- When software needs to be updated on the Forensics Steady State machine, the "merge" script will be run so that the solution retains the updates even upon "roll back". The time it takes to merge the snapshot.vhd file with the baseline image was recorded.
- When an investigator has finished a case, the system must be rolled back, deleting any files, applications, or case remnants that may exist on the system and retuned back to its original baseline image. The time it takes to delete the snapshot file and re-boot to the clean environment was recorded and compared to the re-boot time of a normal Windows 7 re-boot on a similar machine.

3.1.4 Test Goal 4 - Forensics SteadyState

This test determined if *Forensics Steady State does not interfere with the forensic process* and is stable by measuring if/how often the system crashes and how often that it runs appropriately. Most of the tests in Test Goal 1 will help with testing stability, but more tests involving common forensics tools were also performed. The software that was tested on the solution was consistent with tools that the Rhode Island State Police use.

3.2 Testing Environment and Hardware - Forensics SteadyState

The Forensics Steady State solution was implemented on a 500 GB Seagate Barracuda Hard Drive. All hard drives used were tested on a Dell OptiPlex 760, x86-based PC with an Intel Core 2 Duo (2.66GHz) processor, and 4 GB of installed RAM. The Forensic Steady State image was created using the Steadier State procedure for a Microsoft Windows 7 Enterprise operating system.

Section 3.4 outlines specific testing procedures used to validate the forensic integrity of Steadier State. The testing procedures also include the testing of functionality of several programs commonly used by digital forensic investigators: X-Ways Forensics, EnCase, FTK, and ForensicSoft's SAFEBlock [ForensicsWiki].

The following hard drives were used during the testing phase of this project:

	Make/Model	**Capacity**	**Purpose**
1.	Seagate S/N: 5QM1H255 Model: ST3500320AS	500 GB	Steadier State
2.	Seagate S/N: 5Q61QCTL Model: ST3500630AS	500 GB	Source Windows 7 Enterprise
3	Seagate	500 GB	Contains and

.	S/N: 5QM1EXCS Model:		deploys image.vhd
4 .	Samsung Model: HD161GJ	160 GB	Contains Deep Freeze solution
5 .	Western Digital S/N: WMAV33252519 Model: WD1600AAJS-75M0A0	160 GB	Contains Drive Vaccine solution

Table 1- List of Hard Drives Used in Experimentation

3.3 Testing Procedures - Forensics SteadyState

The following sections detail specific test procedures performed to forensically validate the Forensics Steady State solution. The Forensics Steady State Test Plan was developed specifically to forensically validate Forensics Steady State to test all known areas of a hard disk that has the solution deployed to it. In this case, specific tests were developed to make logical and physical writes to the two known partitions created by the solution as well as unpartitioned space and the boot records associated with each area of the disk. Each procedure has detailed steps with test-specific functions built in to allow for a testing procedure to be re-used for validating each aspect of the goals for this project. The test-specific functions for each test procedure can be found in Section 4.

Section 4 discusses each test individually, including any test-specific functions performed and the results and implications of each test. These tests are re-usable for any solution similar to Forensics Steady State, such as Deep Freeze or Drive Vaccine, with the appropriate changes for each test made specific to tool being forensically validated. Table 2 lists all of the tests that were performed along with the expected results of each test.

Test	Requirement	Expected Results
File and Folder Write Test	Goal 1	All logical writes made to the system will be deleted upon rollback.
Application Write Test	Goal 1	All logical writes made to the system will be deleted upon rollback.
Raw Hex Write Test – Volume Boot Record of Partition 2	Goal 1	Raw hex writes made to the VBR of Partition 2 will be deleted upon rollback.
Raw Hex Write Test – Within Unpartitioned Space of the disk	Goal 1	Raw hex writes made to unallocated space of Partition 2 will be deleted

		upon rollback.
Raw Hex Write Test – Within image.vhd of Partition 2	Goal 1	Raw hex writes made within the image.vhd file will be deleted upon rollback.
Raw Hex Write Test – Within snapshot.vhd of Partition 2	Goal 1	Raw hex writes made within the snapshot.vhd file will be deleted upon rollback.
Raw Hex Write Test – Volume Boot Record of Partition 1	Goal 1	Raw hex writes made to the VBR of Partition 1 will be deleted upon rollback.
Raw Hex Write Test – Outside Volume Boot Record of Partition 1	Goal 1	Raw hex writes made to WinPE portion of Partition 1 will be deleted upon rollback.
Raw Hex Write to Virtualized C: Drive – Within Volume Boot Record	Goal 1	Raw hex writes made to VBR of virtualized C Drive will be deleted upon rollback.

Raw Hex Write to Virtualized C: Drive – Outside Volume Boot Record	Goal 1	Raw hex writes made to virtualized C drive outside of VBR will be deleted on rollback.
System Update Test – Using snapshot.vhd without Sysprep	Goal 1	The snapshot file will not be accepted by Machine 2 and will fail.
System Update Test – Using snapshot.vhd with Sysprep	Goal 1	The snapshot file will not be accepted by Machine 2 and will fail.
Rollback Time Measurement	Goal 2	The rollback times of all tested solutions will be similar.
Update Time Measurement	Goal 2	The update times of all tested solutions will be similar.
Keeping Temporary Writes	Goal 2	Forensics Steady State will merge the snapshot.vhd and image.vhd files and work

		successfully.
Reboot Time Comparison	Goal 3	The reboot times of a normal Windows 7 machine and Forensics Steady State will be similar.
Forensic Steady State Rollback Comparison to Normal Windows Reboot	Goal 3	Forensics Steady State will take longer to rollback than a normal Windows 7 machine takes to perform a normal reboot.
Merge Time for Forensic Steady State	Goal 3	The average merge time for Forensics Steady State will take between 2 and 5 minutes.
Disk Overflow Test	Goal 4	Windows will not allow oversized files to overflow the disk and will prompt the user for additional storage media.
Image File Write Test	Goal 4	Forensics Steady State

		will be able to use forensics tools to create a disk image successfully.
Forensic Tool Test	Goal 4	Forensics Steady State will be able to run forensic software, create temporary case files, and be fully functional.
Fixed Disk Test – Copying Files to a Write-protected Internal Hard Disk	Goal 4	SAFE Block will be fully functional with Forensics Steady State and write-protect disk drives appropriately.
Fixed Disk Test – Copying Files to an Internal Hard Disk	Goal 4	Any files copied to the attached internal media will be copied successfully.

Table 2 - Expected Results of Tests Performed

3.3.1 Testing Procedure 1 - Forensics SteadyState

The purpose of Testing Procedure 1 is to investigate Forensics Steady State's behavior when raw disk writes are made while booted into the environment, and to ensure that all files, folders, applications, and raw disk writes are deleted upon rollback

of the solution, and that the original baseline image remains consistent and forensically sound. The procedure can be re-used for each different raw disk write test performed to test Goal 1 and satisfy the claims in Section 3.1.1.

This procedure uses Message-Digest algorithm 5 (MD5) which is a cryptographic hash function that generates a 128-bit hash value. In digital forensic investigations, the MD5 algorithm is used to generate a digital signature of files/disks. These signatures are then used to validate images made of digital evidence to ensure the image is a bit-for-bit copy of the original file/disk. Identical MD5 hash values indicate that the image is an exact copy of the original source evidence. If the signatures do not match, then the image cannot be forensically validated because it does not accurately reflect the original media [Hoog, 2008]. This procedure makes use of Backtrack 5 R3 64-bit gnome for taking MD5 hash values of image.vhd in a forensically sound environment. The procedure is:

1. Boot machine with hard drive containing Forensics Steady State with BackTrack Live CD.
2. Navigate to directory containing "image.vhd" and take MD5 hash of the file.
3. Record hash value and shutdown machine.
4. Boot machine into Forensics Steady State environment, click "Roll back" machine.
5. Perform test-specific functions.
6. Shutdown Forensics Steady State environment.
7. Restart machine and select "roll back".
8. When rollback is complete, shut down the machine.

9. Boot machine with hard drive containing Forensics Steady State with BackTrack Live CD.
10. Navigate to directory containing "image.vhd" and take MD5 Hash of the file.
11. Record hash value and shutdown machine.

It is important to note that if the MD5 hash value of image.vhd from Step 10 does not match the MD5 hash value taken in Step 2, then the baseline image of Forensics Steady State has been altered. In this case, a new image should be deployed onto a wiped hard drive to guarantee the workstation is forensically sound.

3.3.2 Testing Procedure 2 - Forensics SteadyState

The purpose of Testing Procedure 2 is to determine if Forensics Steady State is easy for law enforcement investigators to use. In order to test the ease of use of the solution, it was compared to other comparable products - *Faronics'* Deep Freeze, Pacheras' Steady State solution, and *Horizon DataSys'* Drive Vaccine. The procedure illustrates the main differences in behavior between Forensics Steady State, Deep Freeze, and Drive Vaccine. These differences include rollback times, updating times, and the method used to retain temporary changes until a rollback or merge is performed. The procedure can be re-used for each test-specific function to test Goal 2 and satisfy the claims in Section 3.1.2. The procedure is:

1. Boot machine with hard drive containing Deep Freeze.
2. Perform test-specific functions for Deep Freeze.
3. Shutdown Deep Freeze environment.

4. Boot machine with hard drive containing Drive Vaccine.
5. Perform test-specific functions for Drive Vaccine.
6. Shutdown Drive Vaccine.
7. Boot machine with hard drive containing Forensics Steady State
8. Perform test-specific functions for Forensics Steady State
9. Shutdown Forensics Steady State.

3.3.3 Testing Procedure 3 - Forensics SteadyState

The purpose of Testing Procedure 3 is to determine if Forensics Steady State does not substantially delay investigations. This procedure works directly with the Forensics Steady State solution and compares it to a normal Windows 7 Enterprise machine like the forensic workstations that may currently be in use by law enforcement agencies and other forensic practitioners. For the purposes of these tests, the source Windows 7 computer from which the baseline image for Forensics Steady State was built was used for comparison. The procedure can be re-used for each test specific function to test Goal 3 and satisfy the claims in Section 3.1.3. The procedure is:

1. Boot machine with hard drive containing Forensics Steady State.
2. Perform test-specific functions for Forensics Steady State.
3. Shutdown Forensics Steady State environment.
4. Boot machine with hard drive containing normal Windows 7 OS.
5. Perform test-specific functions for normal Win7 machine.
6. Shutdown Windows 7 machine.

3.3.4 Testing Procedure 4 - Forensics SteadyState

The purpose of Testing Procedure 4 is to determine if multiple Forensics Steady State solutions can be updated by copying and distributing the snapshot.vhd file from one updated machine to another Forensic Steady State machine. Typically, updating an entire laboratory of Windows machines can be time consuming. The motivation of this procedure is to determine if updating one Forensics Steady State solution can simplify the process of updating several machines simply by copying the snapshot.vhd file from the updated machine and overwriting the snapshot.vhd file on other un-updated machines. Observations will be recorded detailing if the changes are accepted and retained. The procedure can be re-used for each test specific function to test both Goal 2 in terms of ease of use, and Goal 3 in terms of not substantially delaying investigations. The procedure is:

1. Boot first machine with hard drive containing Forensics Steady State.
2. Choose the rollback option upon re-boot to ensure the original image is used.
3. Perform test-specific functions for Forensics Steady State on machine 1.
4. Shutdown Forensics Steady State environment.
5. Boot machine 1 with BackTrack Live CD and attach external hard drive.
6. Copy Snapshot.vhd from machine 1 hard drive to external hard drive.
7. Shutdown machine 1.
8. Boot machine 2 with hard drive containing Forensics Steady State.
9. Choose the rollback option upon re-boot to ensure snapshot file will be empty.
10. Shutdown machine 2.

11. Boot machine 2 with BackTrack Live CD and attach external hard drive.
12. Copy over the Snapshot.vhd file from the external hard drive to the hard drive belonging to machine 2, overwriting the old snapshot file.
13. Shutdown machine 2 and remove BackTrack Live CD
14. Boot machine 2 and record behavior.

3.3.5 Testing Procedure 5 - Forensics SteadyState

The purpose of Testing Procedure 5 is to determine if Forensics Steady State is functional with forensic software including X-Ways Forensics, FTK, EnCase, and ForensicSoft's SAFE Block software write-blocker. Typically, in digital forensic investigations, law enforcement must collect and image hard drives or other digital media and ensure all devices can be imaged while being write-protected to preserve the integrity of the evidence. Law enforcement must be able to use their normal forensic tool suite when performing investigations. The procedure can be re-used for each test specific function to test Goal 4 and satisfy the claims in Section 3.1.4.

1. Boot machine with hard drive containing Forensics Steady State.
2. Perform test-specific functions.
3. Shutdown Forensics Steady State.

4 Results - Forensics SteadyState

This section discusses the tests performed to forensically validate the Forensics Steady State solution. Table 3 lists all of the tests performed, along with which requirements they satisfy, and the result of each test. For more detailed procedure and results, please see corresponding sections in this section.

Test	Requirement	Result
4.1.1 File and Folder Write Test	Goal 1	Succeeded
4.1.2 Application Write Test	Goal 1	Succeeded
4.1.3 Raw Hex Write Test – Volume Boot Record of Partition 2	Goal 1	Failed – OS unable to reboot after boot sector is written to.
4.1.4 Raw Hex Write Test – Within Unpartitioned space of the disk	Goal 1	Failed – Writes remain after rollback
4.1.5 Raw Hex Write Test – Within image.vhd of Partition 2	Goal 1	Succeeded
4.1.6 Raw Hex Write Test –	Goal 1	Succeeded

Within snapshot.vhd of Partition 2		
4.1.7 Raw Hex Write Test – Volume Boot Record of Partition 1	Goal 1	Failed – OS unable to reboot after boot sector is written to.
4.1.8 Raw Hex Write Test – Outside Volume Boot Record of Partition 1	Goal 1	Failed – OS unable to reboot after disk is written to.
4.1.9 Raw Hex Write to Virtualized C: Drive – Within Volume Boot Record	Goal 1	Failed – OS unable to reboot after boot sector is written to.
4.1.10 Raw Hex Write to Virtualized C: Drive – Outside Volume Boot Record	Goal 1	Succeeded
4.1.11 System Update Test – Using snapshot.vhd without Sysprep	Goal 1	Succeeded
4.1.12 System Update Test – Using snapshot.vhd with Sysprep	Goal 1	Failed – OS unable to reboot due to snapshot.vhd belonging another sysprepped machine.

4.2.1 Rollback Time Measurement	Goal 2	Drive Vaccine has the fastest rollback time.
4.2.2 Update Time Measurement	Goal 2	Drive Vaccine's update time is instantaneous.
4.2.3 Keeping Temporary Writes	Goal 2	N/A
4.3.1 Reboot Time Comparison	Goal 3	Forensics Steady State reboots faster than a normal Windows 7 workstation.
4.3.2 Forensic Steady State Rollback Comparison to Normal Windows Reboot	Goal 3	Windows 7 workstation reboots faster than total rollback time of Forensics SteadyState.
4.3.3 Merge Time for Forensic Steady State	Goal 3	Forensics Steady State has an average update time of 01:33.60.
4.4.1 Disk Overflow Test	Goal 4	Successful
4.4.2 Image File Write Test	Goal 4	Successful

4.4.3 Forensic Tool Test	Goal 4	Successful
4.4.4 Fixed Disk Test – Copying Files to a Write-protected Internal Hard Disk	Goal 4	Successful
4.4.5 Fixed Disk Test – Copying Files to a Internal Hard Disk	Goal 4	Successful

Table 3 - Test Summaries

4.1 Goal 1 Forensic Validation Findings - Forensics SteadyState

Law enforcement and other digital forensic investigators must be able to ensure that they are using a clean Windows 7 environment upon each new investigation to prevent cross-contamination between cases and preserve the probative value of the evidence. Cross-contamination can include files/folders, malware, viruses, applications, and case data left over from a previous case that was analyzed on a specific forensic workstation.

The most common method of preventing cross-contamination is to perform each investigation on a hard drive that had been wiped, re-formatted, and reconfigured for forensic investigations [Forensics Investigations]. Forensics Steady State eliminates the requirement to start a forensic workstation from scratch because all writes made to the system are placed into a snapshot file that is easily discarded once rolled back, providing

the user with a clean baseline Windows 7 image to perform more investigations. While Forensics Steady State may be a fast and elegant solution for law enforcement to use, proper forensic validation of the solution is necessary to prove cross-contamination has not occurred.

4.1.1 File and Folder Write Test - Forensics SteadyState

The purpose of this test was to ensure that any files or folders created, used, or accessed during a forensic investigation are removed upon rollback of the host. Digital forensic investigations usually yield output files in the form of recovered/exported evidence files, case files, reports, and temporary files. The artifacts produced by an investigation must be completely removed to prevent cross-contamination between cases.

An MD5 hash value of the baseline image.vhd file was taken before the test to later prove the rolled back solution accurately reflects a clean Windows 7 image. Test Procedure 1, detailed in Section 3.3.1, was followed directly for this test. The test specific functions included adding a text file and folder to the desktop. Notepad was used to produce a plain text file "Test.txt" that was saved to the desktop. An empty folder "Test Folder" with a copy of "Test.txt" in the directory was also added to the desktop. Once both the text file and the folder containing a copy of the text file were added, the test continued with Test Procedure 1, which included rolling back the system and taking another MD5 hash of image.vhd.

As expected, once the solution was rolled back, both the text file and folder containing a copy of the text file were deleted and solution was back to its baseline state.

The MD5 hash of image.vhd remained the same retaining the forensic integrity of Forensics Steady State.

Pre-test MD5 Hash	2aacf3dbe0501ad125290e24cf4c3c88
Post-test MD5 Hash	2aacf3dbe0501ad125290e24cf4c3c88

Table 4 - Matching Hash for image.vhd After Test 4.1.1

4.1.2 Application Write Test - Forensics SteadyState

The purpose of this test is to ensure any applications installed during an investigation on a forensic workstation running Forensics Steady State will be removed upon rollback of the solution.

An MD5 hash value of the baseline image.vhd file was taken before the test to later prove the rolled back solution accurately reflects a clean Windows 7 image. Test Procedure 1, detailed in Section 3.3.1, was followed directly for this test. The test specific functions included installing an application and making sure any remnants/artifacts of that application are completely removed upon rollback of the solution. Apple's iTunes was installed on Forensics Steady State, and once the install completed, the test continued with Test Procedure 1, which included rolling back the system and taking another MD5 hash of image.vhd.

As expected, no artifacts/remnants remained on the system after rollback and the solution was back to its baseline state. The MD5 hash of image.vhd remained the same retaining the forensic integrity of Forensics Steady State.

Pre-test MD5 Hash	2aacf3dbe0501ad125290e24cf4c3c88
Post-test MD5 Hash	2aacf3dbe0501ad125290e24cf4c3c88

Table 5 - Matching Hash for image.vhd After Test 4.1.2

4.1.3 Raw Hex Write Test – Volume Boot Record of Partition 2

The second partition of the hard disk containing the Forensics Steady State solution houses both image.vhd and snapshot.vhd. The volume boot record for Partition 2 could

be a target of malicious software, such as Cidox Trojan Horse [Symantec], that may be the result of cross-contamination from suspect evidence. The purpose of this test to determine if malicious software can make writes to the volume boot record of the partition containing the vhd file that may not be reversed upon rollback of the solution. In order to emulate writes to the boot sector, a hex editor, X-Ways' WinHex, was used to make raw hexadecimal writes while the solution was fully booted.

An MD5 hash value of the baseline image.vhd file was taken before the test to later prove the rolled back solution accurately reflects a clean Windows 7 image. Test Procedure 1, detailed in Section 3.3.1, was followed directly for this test. The test specific functions included using WinHex to make raw hex writes to the VBR of Partition 2. After the solution was fully booted, WinHex was opened. Using a physical hexadecimal view of the hard disk, 4 bytes located at offset 0x130 within the volume boot record of partition 2 were changed from "00 66" to "AA AA". WinHex immediately displayed a Windows error message explaining that the disk writes would not be allowed:

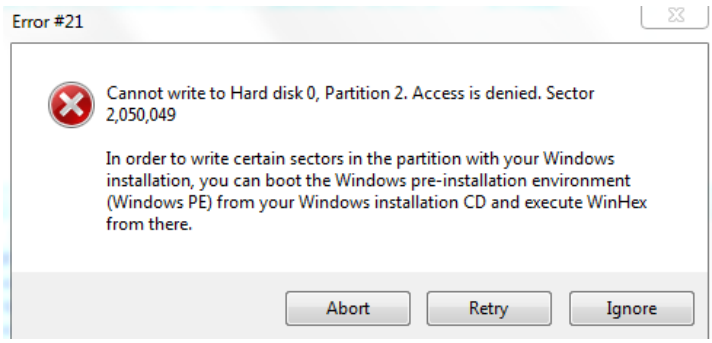

Figure 4 - Windows Error Message Not Allowing Disk Writes

After the writes were made, the test continued with Test Procedure 1, which included rolling back the system and taking another MD5 hash of image.vhd. The MD5 hash of image.vhd remained the same retaining the forensic integrity of the baseline image.vhd image.

Pre-test MD5 Hash	2aacf3dbe0501ad125290e24cf4c3c88
Post-test MD5 Hash	2aacf3dbe0501ad125290e24cf4c3c88

Table 6 - Matching Hash for image.vhd After Test 4.1.3

However, after rollback, inspection of the bytes located at offset 0x130 within the VBR of Partition 2 had retained the raw disk writes despite the Windows error message. The installed anti-virus software, Microsoft Security Essentials, also did not notice that the volume boot record had been altered. This unexpected result presents a vulnerability of Forensics Steady State in that writes made directly to the disk outside of image.vhd or snapshot.vhd may infect a forensic workstation, causing the need for a clean Forensics Steady State environment to be created.

4.1.4 Raw Hex Write Test – Within Unpartitioned Space of the Disk

The hard disk containing Forensics Steady State was also examined outside of the two main partitions present on the disk within unpartitioned space. The purpose of this test is to examine if malicious software has the ability to write to unpartitioned space on

the hard disk. Once again, WinHex was used to emulate raw hexadecimal writes to the unallocated space while the solution was fully booted.

An MD5 hash value of the baseline image.vhd file was taken before the test to later prove the rolled back solution accurately reflects a clean Windows 7 image. Test Procedure 1, detailed in Section 3.3.1, was followed directly for this test. The test specific functions included using WinHex to make raw hex writes to the unpartitioned space. After the solution was fully booted, WinHex was opened. Using a physical hexadecimal view of the hard disk, 4 bytes located at offset 0xE8E0C00000 within the unallocated space of Partition 2 were changed from "00 00" to "AA AA". Windows unexpectedly allowed the writes, and after continuing with Testing Procedure 1 and rolling back the solution, the writes were retained. This unexpected result presents a vulnerability of Forensics Steady State in that writes made directly to the disk within unpartitioned space may infect a forensic workstation, causing the need for a clean Forensics Steady State environment to be created. The MD5 hash of image.vhd, however, was verified to be unchanged:

Pre-test MD5 Hash	2aacf3dbe0501ad125290e24cf4c3c88
Post-test MD5 Hash	2aacf3dbe0501ad125290e24cf4c3c88

Table 7 - Matching Hash for image.vhd After Test 4.1.4

4.1.5 Raw Hex Write Test – Within image.vhd of Partition 2

With both the boot sector and unallocated space within Partition 2 of the hard disk being tested, it is necessary to examine writes to image.vhd and snapshot.vhd. The

purpose of this test is to examine the behavior of making hexadecimal writes within the image.vhd file while Forensics Steady State is fully booted.

An MD5 hash value of the baseline image.vhd file was taken before the test to later prove the rolled back solution accurately reflects a clean Windows 7 image. Test Procedure 1, detailed in Section 3.3.1, was followed directly for this test, with the test specific functions of using WinHex to make raw hex writes within the image.vhd file of Partition 2. After the solution was fully booted, WinHex was opened and using a physical hexadecimal view of the hard disk, 4 bytes located at offset 0x74C6FC090 within image.vhd were changed from "00 00" to "AA AA". As expected, Windows would not allow the disk writes to be made.

Testing Procedure 1 was then continued, which included rolling back the system and computing another MD5 hash of image.vhd. After the test concluded, the solution was booted once again and WinHex was used to verify the writes had not been made despite the Windows error message.

Pre-test MD5 Hash	2aacf3dbe0501ad125290e24cf4c3c88
Post-test MD5 Hash	2aacf3dbe0501ad125290e24cf4c3c88

Table 8 - Matching Hash for image.vhd After Test 4.1.5

4.1.6 Raw Hex Write Test – Within snapshot.vhd of Partition 2

After making raw hex writes to the volume boot record, unallocated space, and image.vhd files within Partition 2 of the hard disk, it was necessary to observe the behavior of the solution when writes are made directly to the snapshot.vhd file.

An MD5 hash value of the baseline image.vhd file was taken before the test to later prove the rolled back solution accurately reflects a clean Windows 7 image. Test Procedure 1, detailed in Section 3.3.1, was followed directly for this test, with the test specific functions of using WinHex to make raw hex writes within the snapshot.vhd file of Partition 2. After the solution was fully booted, WinHex was opened and using a physical hexadecimal view of the hard disk, 4 bytes located at offset 0x2D090 within snapshot.vhd were changed from "00 00" to "AA AA". As expected, Windows would not allow the disk writes to be made. It is also important to note that any raw hexadecimal writes made to any other files or the free space of this partition are also protected by the Forensics SteadyState solution.

After finishing up with Testing Procedure 1, the test concluded with expected results in that the disk writes were not allowed and never made. This was verified using a physical view of the hard disk within WinHex. The image.vhd file remained unchanged:

Pre-test MD5 Hash	2aacf3dbe0501ad125290e24cf4c3c88
Post-test MD5 Hash	2aacf3dbe0501ad125290e24cf4c3c88

Table 9 - Matching Hash for image.vhd After Test 4.1.6

4.1.7 Raw Hex Write Test – Volume Boot Record of Partition 1

With the second partition testing being completed, the forensic integrity of Partition 1 also needed to be verified. Partition 1 contains the WinPE environment that Forensics Steady State uses to rollback and merge the solution. The volume boot record of Partition 1 could be contaminated by remnants left over from a previous or current investigation being performed on a forensic workstation. The purpose of this test to determine if malicious software can make writes to the volume boot record of the partition that may not be reversed upon rollback of the solution. Once again, WinHex was used to make raw hexadecimal writes to the disk while booted into the solution.

An MD5 hash value of the baseline image.vhd file was taken before the test to later prove the rolled back solution accurately reflects a clean Windows 7 image. Test Procedure 1 was followed directly for this test, with the test specific functions of using WinHex to make raw hex writes within the volume boot record of Partition 1. After the solution was fully booted, WinHex was opened and using a physical hexadecimal view of the hard disk, 4 bytes located at offset 0x within the volume boot record were changed from “00 00” to “AA AA”.

Windows unexpectedly allowed the writes, and after continuing with Testing Procedure 1 and rolling back the solution, the writes were retained. The installed anti-virus software, Microsoft Security Essentials, also did not notice that the volume boot record had been altered. This unexpected result presents a vulnerability of Forensics Steady State in that writes made directly to the disk within partition 1’s boot record may infect a forensic workstation, causing the need for a clean Forensics Steady State

environment to be created. The MD5 hash of image.vhd, however, was verified to be unchanged:

Pre-test MD5 Hash	2aacf3dbe0501ad125290e24cf4c3c88
Post-test MD5 Hash	2aacf3dbe0501ad125290e24cf4c3c88

Table 10 - Matching Hash for image.vhd After Test 4.1.7

4.1.8 Raw Hex Write Test – Outside Volume Boot Record of Partition 1

The first partition of the Forensics Steady State solution must also be examined outside of the volume boot record. The rest of the partition contains the WinPE operating system that is used to rollback and merge the solution when selected from the pre-boot environment. The purpose of this test is to determine if any writes can be made to the first partition of the hard disk when booted into solution. WinHex was used to make the raw hex writes to the disk.

An MD5 hash value of the baseline image.vhd file was taken before the test to later prove the rolled back solution accurately reflects a clean Windows 7 image. Test Procedure 1 was followed directly for this test, with the test specific functions of using WinHex to make raw hex writes outside of the volume boot record within Partition 1. After the solution was fully booted, WinHex was opened and using a physical hexadecimal view of the hard disk, 4 bytes located at offset 0x3D0 were changed from "00 00" to "AA AA".

The results of this test were unexpected, as seen similarly in Test 4.1.7, in that Windows ultimately allowed the writes to be made without error. After continuing with

Testing Procedure 1, the writes were retained after rollback. These writes could be made to any file contained on partition 1. It is also important to note that any raw writes made to the free space of this partition are also unprotected by the Forensics Steady State solution. This result presents a vulnerability of Forensics Steady State in that writes made directly to the disk outside of the boot record of Partition 1 may infect a forensic workstation, possibly affecting the rollback and merge functionality of the solution. This will cause the need for a clean Forensics Steady State environment to be created. The MD5 hash of image.vhd, however, was verified to be unchanged:

Pre-test MD5 Hash	2aacf3dbe0501ad125290e24cf4c3c88
Post-test MD5 Hash	2aacf3dbe0501ad125290e24cf4c3c88

Table 11 - Matching Hash for image.vhd After Test 4.1.8

4.1.9 Raw Hex Write to Virtualized C: Drive – Within Volume Boot Record

This test is designed to determine the behavior of Forensics Steady State when raw hexadecimal writes are made to the volume boot record of the virtualized C drive when the solution is fully booted.

An MD5 hash value of the baseline image.vhd file was taken before the test to later prove the rolled back solution accurately reflects a clean Windows 7 image. Test Procedure 1 was followed directly for this test, with the test specific functions of using WinHex to make writes to the disk. After the solution was fully booted, WinHex was opened and the virtual C drive was accessed with a logical view of disk. 4 bytes located at offset 0x20 within the volume boot record were changed from "00 00" to "AA AA".

Once the write was attempted, a Windows error was displayed: "Unable to lock the drive, other programs may be using it. Access Denied". Even though the error message was present, the writes were made to the disk. The installed anti-virus software, Microsoft Security Essentials, also did not notice that the volume boot record had been altered. This is evident because after Testing Procedure 1 was complete and the solution was rolled back, the solution was unable to boot. This result presents a vulnerability of Forensics Steady State in that writes made directly to the logical C drive outside of the boot record may infect a forensic workstation, possibly affecting the rollback and merge functionality of the solution. This will cause the need for a clean Forensics Steady State environment to be created. The image.vhd file remained unchanged:

Pre-test MD5 Hash	2aacf3dbe0501ad125290e24cf4c3c88

Post-test MD5 Hash	2aacf3dbe0501ad125290e24cf4c3c88

Table 12 - Matching Hash for image.vhd After Test 4.1.9

4.1.10 Raw Hex Write to Virtualized C: Drive – Outside Volume Boot Record

This test is designed to determine the behavior of Forensics Steady State when raw hexadecimal writes are made to the virtualized C drive outside of the volume boot record within any file when the solution is fully booted.

An MD5 hash value of the baseline image.vhd file was taken before the test to later prove the rolled back solution accurately reflects a clean Windows 7 image. Test Procedure 1 was followed directly for this test, with the test specific functions of using WinHex to make writes to the disk. After the solution was fully booted, WinHex was opened and the virtual C drive was accessed with a logical view of disk. 4 bytes located at offset 0x1040 outside of the volume boot record were changed from "00 00" to "AA AA".

Once the write was attempted, the same Windows error from test 4.1.10 was displayed explaining the write cannot be made. Even though the error message was present, the writes appear to have been made to the disk and after Testing Procedure 1 was complete and the solution was rolled back, the writes were successfully removed. It is important to note that the free space of the virtualized C drive is also protected by the Forensics Steady State solution. The image.vhd file remained unchanged:

Pre-test MD5 Hash	2aacf3dbe0501ad125290e24cf4c3c88
Post-test MD5 Hash	2aacf3dbe0501ad125290e24cf4c3c88

Table 13 - Matching Hash for image.vhd After Test 4.1.10

4.1.11 System Update Test – Using snapshot.vhd without Sysprep

System updates are an important aspect of using forensic workstations within a lab environment. As Windows or forensic tool updates release, it is important for computer forensic investigators to make use of all current technology as well as keeping their computers secure through Windows patches. Updating multiple machines at once can be extremely time consuming, thus delaying future investigations. Usually, when a master image is deployed onto multiple machines, the System Preparation (Sysprep) tool is used. The sysprep tool prepares an image of windows for duplication and removes any system specific data from that Windows installation so that the image can be reused [TechNet].

This project explores the idea of using the snapshot.vhd file located on one machine to update another machine. Specifically, it explores the notion of updating one Forensics Steady State solution and using that machine's exiting snapshot.vhd file to update multiple machines by replacing the older snapshot.vhd file.

For this specific test, two Forensics Steady State solutions were deployed *without* the use of sysprepping the original baseline image. Testing Procedure 4, detailed in

Section 3.3.4, was followed directly for this test. Once the first solution was rolled back, a text file and a test folder were added to the desktop. The latest version of Apple iTunes was also installed. With this solution being updated, it was shut down and the snapshot.vhd file was copied to external media through the use of a BackTrack Live boot disk. The hard drive belonging to the second Forensics Steady State solution was then attached to the machine, and the snapshot.vhd file from the first machine was copied to the hard disk of the second machine, overwriting the second machine's snapshot.vhd file.

The second machine's hard disk contained an exact copy of the snapshot.vhd file from the first machine. The second machine was then booted successfully. When the Windows environment was fully loaded, all of the updates made to the first machine were accurately reflected.

Although it is highly unlikely that a digital forensics laboratory would deploy multiple machines without sysprepping, the possibility of using the snapshot.vhd file to update another machine, validated by this project, may prove helpful.

4.1.12 System Update Test – Using snapshot.vhd with Sysprep

Test 4.1.14 attempted to update a Forensics Steady State machine by copying over an updated snapshot.vhd file from another machine that was running Forensics Steady State. Both machines were deployed from a baseline image that was not sysprepped. This test aimed to use the same procedure, but using two machines that *were* deployed from a sysprepped image.

Additional test preparation was required for this test. The original source machine was sysprepped and new baseline image was produced for deployment on Machine 1 and Machine 2. The original Windows 7 image that was used to create image.vhd was sysprepped using the files and instructions in the Steadier State package. A new image.vhd was then created using the Steadier State boot media and the image was deployed to both Machine 1 and Machine 2. Testing Procedure 4, detailed in Section 3.3.4, was followed directly for this test. Once the first solution was rolled back, a text file and a test folder were added to the desktop. The latest version of Apple iTunes was also installed. With this solution being updated, it was shut down and the snapshot.vhd file was copied to external media through the use of a BackTrack Live boot disk. The hard drive belonging to Machine 2 was then attached to the machine, and the snapshot.vhd file from Machine 1 was copied to the hard disk of the Machine 2 overwriting the snapshot.vhd file.

The reboot process of the Machine 2, now containing the updated snapshot.vhd file, failed and was unable to boot into the Windows 7 environment. This result proves that a snapshot file cannot be shared between sysprepped machines and using the snapshot file is not an option for updating multiple machines running Forensics Steady State.

4.2 Goal 2 Forensic Validation Findings - Forensics SteadyState

Law enforcement investigators are typically trained to follow digital forensic procedures in the acquisition, preservation, and analysis of digital evidence. This includes having a working knowledge of existing forensic software and hardware tools. These investigators often do not possess the skills necessary to troubleshoot, fix, or

deploy forensic workstations. An important aspect of using a forensic workstation during multiple on-going investigations is the system's ease of use. For a less technical savvy investigator, re-imaging a workstation after an investigation can seem like a daunting task. The following tests focus on the ease of use of Forensics Steady State as compared to other solutions such as Faronics' Deep Freeze and Horizon DataSys Inc's Drive Vaccine.

4.2.1 Rollback Time Measurement

This first test investigates the process and time of rolling back a machine with Forensics Steady State, Deep Freeze, or Drive Vaccine. Testing Procedure 2, detailed in Section 3.3.2, was used to measure the rollback times of all three solutions. The test specific functions included simply choosing the appropriate rollback option for each individual solution. In order to rollback Forensics Steady State, the user simply shuts down or reboots the computer leaving all default options selected. Likewise, both Deep Freeze and Drive Vaccine roll back to their initial states with a simple shutdown or reboot of the system.

Time was recorded with a stopwatch as soon as the machine was shutdown from a running Windows environment and the rollback option was chosen. The time ceased to be recorded once every solution's Windows environment completed the booting process. The test was performed five times for each solution due to varying boot times and the results are presented in the figure below:

Forensic	Deep Freeze	Drive Vaccine

s Steady State		
02:23.51	02:21.35	01:27.26
02:35.88	02:21.98	01:50:05
03:01.55	02:02.49	01:14.77
02:34.48	02:22.65	01:15.78
03:08.65	02:30.06	01:10.00

Table 14 - Rollback Time Comparisons

	Forensics Steady State	**Deep Freeze**	**Drive Vaccine**
Average Rollback Time	02:44.80	02:19.70	01:23.60

Table 15 - Average Rollback Times

On average, Forensics Steady State has the slowest rollback time of the three solutions, with Deep Freeze falling slightly behind and Drive Vaccine being the fastest. In terms of ease of use, all three of the solutions have intuitive functionality.

4.2.2 Update Time Measurement

Being able to perform Windows updates and updates to forensic software is imperative for law enforcement to keep their machines secure and guarantee they will be using the latest cutting edge tools. The purpose of this test is to investigate both the update time and ease of updating Forensics Steady State, Deep Freeze, and Drive Vaccine.

In order to perform updates to Forensics Steady State the user must make all desired updates and then place an empty text file named "noauto.txt" at the root of the C drive while booted into the solution's environment. Then, the user must reboot the system and select the rollback option from the pre-boot environment. WinPE will then start and instead of automating the process of deleting the snapshot.vhd file, it will instead halt at the command prompt due to the text file at the root of the drive. The user must then type "merge" and hit enter to update the baseline image.

Deep Freeze is updated using a control console on a separate workstation. After making all selected updates to the system, the user selects to "thaw" the solution from the console application which will reboot the system without any write-protection to any fixed hard disks. Once updates are performed, the user selects the "Reboot Frozen" option from the console and system retains all changes when restarted.

Once desired updates are performed, Drive Vaccine's baseline image is easily updated by simply clicking on the Drive Vaccine application within the environment and selecting the update option [Drive Vaccine User Manual]. No reboot process is required.

For Forensics Steady State, the update time was measured from the point in which the merge command was entered in the WinPE environment until the solution was booted into Windows. The update time for Deep Freeze started when "Reboot Frozen" was selected from the console and ended once the system was booted into Windows. Drive Vaccine's update time was not recorded because it happens instantaneously and is negligible. The test was performed five times with a stopwatch for each solution due to varying boot times and the results are presented in the figure below:

Forensics Steady State	**Deep Freeze**
01:34.78	01:33.93
01:33.56	01:31.72
01:20.84	01:13.92
01:31.51	01:23.80
01:47.33	00:56.35

Table 16 - Update Time Comparisons

	Forensics Steady State	**Deep Freeze**
Average Update Time	01:33.60	01:19.90

Table 17 - Average Update Times

On average, updating Forensics Steady State takes approximately 14 seconds longer than Deep Freeze and it does require minor technical ability.

4.2.3 Keeping Temporary Writes

Digital investigations performed by law enforcement may require an extensive amount of time on a forensic workstation. These machines must be able to store case files/folders, keep imaging programs open, or let any processes continue working during non-work hours. Each of the three solutions examined in Section 4.2 have methods of retaining temporary writes to the system without the worry of losing current work or data.

Drive Vaccine allows users to keep changes temporarily by either leaving the active solution booted into a Windows environment or updating the baseline image to keep changes, which is not desired. Deep Freeze can retain temporary changes by either keeping the solution in a "frozen" state or "thawed" state. Keeping the solution in a write-protected mode will keep changes until the next reboot and all writes made to the system in a thawed state will be retained permanently, which is not desired. Potential problems could occur if power was cut to an active machine with Drive Vaccine or Deep Freeze installed. For example, if a forensic workstation was taking more than a few hours to image a drive, this process may be left in the laboratory to complete overnight. If power is lost to the building or machine in some way, the solution will rollback to its baseline image upon reboot, causing a possible loss of data.

Forensics Steady State, however, operates much like a normal Windows 7 workstation. Any writes made to the disk will be kept temporarily until the rollback option is chosen from the pre-boot environment. The user also has the option to shutdown or restart the system and continue working with the current state of the solution. All changes are only discarded when the baseline image is restored. This can provide law enforcement with a form of safety net for digital investigations because the baseline image can only be restored if the option is chosen; losing power or other unforeseen circumstances will not cause the loss of data in Forensics Steady State.

4.3 Goal 3 Forensic Validation Findings - Forensics SteadyState

Performing a digital investigation in a timely manner is important for law enforcement to be able to process and complete as many cases as possible. The following tests will determine if Forensics Steady State does not substantially delay investigations by comparing the solution to a normal machine running a Windows 7 Enterprise 64-bit operating system.

4.3.1 Reboot Time Comparison

Two machines were set up to measure the reboot time of Forensics Steady State and Windows 7 Enterprise. Test Procedure 3, detailed in Section 3.3.3, was used to measure the reboot time of each machine. Each machine was fully booted into their operating system environments. After fully loaded, each machine was restarted and duration of time it took each machine to fully restart into Windows was recorded. The test was

performed five times with a stopwatch for each solution due to varying boot times and the results are presented in the figure below:

Forensic Steady State	Windows 7 Workstation
01:00	01:10
01:03	01:15
01:02	01:18
01:04	01:28
01:03	01:27

Table 18 - Reboot Time Comparison

	Forensic Steady State	Windows 7 Workstation
Average Re-boot Time	01:02	01:19

Table 19 - Average Reboot Times

On average, the re-boot time of Forensic Steady State was 17 seconds faster than the machine running just Windows 7 Enterprise.

4.3.2 Forensic Steady State Rollback Comparison to Normal Windows Reboot

Two machines were set up to measure the rollback time of Forensics Steady State and a simple reboot time of Windows 7 Enterprise. Test Procedure 3, detailed in Section 3.3.3, was used to measure the appropriate times of each machine. Each machine was fully booted into their operating system environments. After Forensics Steady State was loaded, the time it took to restart and roll back to the baseline image was recorded. The Windows 7 Enterprise machine was simply restarted and the duration of time between reboot and being fully restarted was recorded. The test was performed five times with a stopwatch for each solution due to varying boot times and the results are presented in the figure below:

Forensics Steady State (Rollback)	**Windows 7 Workstation (Reboot)**
02:23.51	01:10
02:35.88	01:15
03:01.55	01:18
02:34.48	01:28
03:08.65	01:27

Table 20 - Rollback of Solution Compared to Normal Windows 7 Boot

	Forensics Steady State (Rollback)	**Windows 7 Workstation (Reboot)**
Average	02:44.80	01:19.00

Table 21 - Average Rollback Time (FSS) compared to Average Reboot Time (Win 7)

Simply re-booting a machine running Windows 7 Enterprise is significantly faster than rolling back the Forensics Steady State solution by approximately 01:25. The two machines are performing completely different functions but it proves that rolling back to a pristine baseline image in Forensics Steady State merely takes 01:25 longer than a simple reboot of a normal forensic workstation.

4.3.3 Merge Time for Forensic Steady State

The time it takes to update a baseline image of Forensics Steady State was measured in test 4.2.2. Test Procedure 3, detailed in Section 3.3.3, was used to measure the update time of the solution. The test specific functions included adding a test file, "Test.txt" and test folder, "Test Folder", to the desktop. The latest version of Apple's iTunes was also installed. The solution was then restarted, and the "merge" command was run to update the baseline image. The update time was measured from the point in which the merge command was entered in the WinPE environment and was no longer recorded once the solution was booted into Windows. The test was performed five times with a stopwatch and the results are presented in the figure below:

Forensics Steady State
01:34.78
01:33.56
01:20.84
01:31.51
01:47.33

Table 22 - Recorded Times to Update Baseline Image

Average Update Time:	01:33.60

Table 23 - Average Baseline Image Update Time

On average, it took 01:33.60 for the baseline image to be completely updated in the Forensics Steady State solution. This time is optimal and can allow for quick updates to be made to the baseline image for further investigative use.

4.4 Goal 4 Forensic Validation Findings - Forensics SteadyState

Goal 4 of this project focuses on system stability and functionality with forensic software to ensure the solution does not interfere with the forensic process. Tests

include observing how the system reacts to fixed disks being hot-swapped with the machine, disk overflow behavior, and general functionality with software that is consistent with tools used by the Rhode Island State Police.

4.4.1 Disk Overflow Test

The purpose of this test is to demonstrate what happens when the Forensics Steady State solution runs out of hard disk capacity. Additional preparation for this test was required in the form of adding a second storage device to the forensic workstation. In this case, a 1 TB internal hard disk was added to the system.

Test Procedure 1 was followed directly for this test, with the test specific functions including the use of FTK imager to image the newly inserted 1 TB hard disk. Once the solution was fully booted, FTK imager was opened and used to create an E01 image of the 1 TB drive. The virtualized C: drive of Forensics Steady State was selected as the destination for the image file. FTK Imager started to image the drive, segmenting it into parts until it required more space. The remaining image segments were selected to be put on the D: drive until that too ran out of space. Once all storage locations were full, FTK Imager displayed a low disk space warning, reporting that only 978 MB of free space remained on the hard disk:

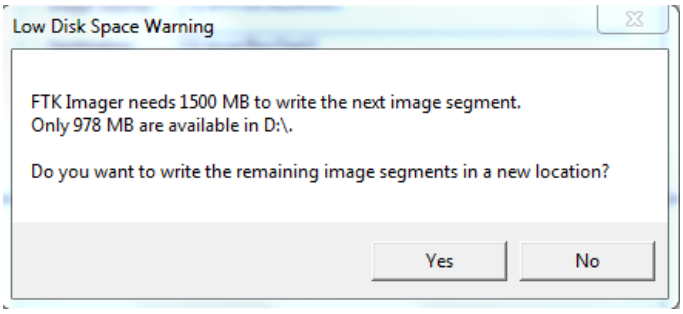

Figure 5 - Disk Overflow Test Indicating Low Disk Space

Imager also asked to write the remaining image segments to a new location, but no additional storage devices were added to finish the imaging.

After finishing Testing Procedure 1, the solution was properly rolled back erasing all traces of the large E01 file and an MD5 hash of image.vhd was computed to verify the baseline image remained unchanged.

4.4.2 Image File Write Test

The purpose of this test is to verify that the Forensics Steady State solution functions appropriately when imaging external media.

Test Procedure 1 was followed directly for this test, with the test specific functions including the use of FTK imager to image a 1 GB USB thumb drive. Once the solution was fully booted, FTK imager was opened and used to create a raw dd image of the thumb drive to the desktop. The image was created and saved successfully and after finishing

Testing Procedure 1, all remnants of the image file were removed and a clean baseline image of the Forensics Steady State solution remained.

4.4.3 Forensic Tool Test

The purpose of this test is to verify that Forensics Steady State functions properly with the use of a forensic tool and a software write-blocker. For the purposes of this test, X-Ways' Forensics was the selected forensic tool and ForensicSoft's SAFE Block was used as the software write-blocker. Both of these tools were included in the baseline image of Forensics Steady State.

Test Procedure 1 was followed directly for this test. The first test-specific function was to open X-Ways Forensics once the solution was booted. After ensuring that SAFE Block was enabled, a 1 GB USB thumb drive was then inserted into the machine and was successfully write-blocked. Using X-Ways Forensics, an E01 image of the drive was made and saved to the internal hard disk. The image was then opened in X-Ways and two deleted files were recovered and saved to the hard disk to simulate case files that can/would be created from a digital investigation. A total of 359 MB of case data, including the image, was created and saved to the machine. X-Ways was then closed, and Testing Procedure 1 was finished.

Both the forensic tool and write-blocker behaved as expected and all created files were deleted upon rollback of the solution.

4.4.4 Fixed Disk Test – Copying Files to a Write-protected Internal Hard Disk

Similar Steady State solutions, such as Deep Freeze, have caused problems for law enforcement when trying to copy files from a forensic workstation's local storage to an additional fixed hard disk in the system. Internal hard drives are often attached to forensic workstations via an eSata port or card for either imaging or exporting of case files. It is imperative for law enforcement to be able to hot-swap hard drives during an investigation on a forensic workstation while using software write-blockers and forensic tools simultaneously.

Testing Procedure 5, detailed in Section 3.3.5, was used for this test. After booting into the Forensics Steady State environment, a 1 TB hard drive was connected to the system via an internal eSata card during operation. SAFE Block was then opened to ensure the drive was write-protected. To simulate writes to the newly attached write-protected hard drive, a text file, "Test.txt", and test folder, "Test Folder", were created and then copied over to the destination drive. Since the drive was write-protected, a Windows error message was displayed indicating that writes could not be made to the drive. The external drive was then disconnected from the system to ensure no abnormal behavior occurred upon removing the disk.

The results of this test were as expected. The solution had no problems recognizing the newly inserted fixed disk, write-protecting the drive, or detaching the drive during the machines operation.

4.4.5 Fixed Disk Test – Copying Files to a Internal Hard Disk

This test follows directly from Test 4.4.4 with the only difference being that the fixed disk added to the system will not be write-protected. Law enforcement needs the

ability to export case files and folders to external or internal media. Solutions such as Deep Freeze write-protect all drive letters, unless this option is manually changed. Copying files to external media connected to a machine running Deep Freeze can be misleading in that the files may appear to be copied to their destination, but are not actually written to the drive. Investigators may mistakenly think they are copying case files to another storage device, reboot their machine to a clean baseline image, and realize the case files that appeared to be copied no longer exist.

Testing Procedure 5, detailed in Section 3.3.5, was used for this test. After booting into the Forensics Steady State environment, a 1 TB hard drive was connected to the system via an internal eSata card during operation. SAFE Block was then opened and the newly attached drive was un-blocked allowing for the possibility of writes to be made. To simulate writes to the newly attached hard drive, a text file, "Test.txt", and test folder, "Test Folder", were created and then copied over to the destination drive successfully. The additional storage device was then powered off during the machines operation to ensure no abnormal behavior occurred. Once the machine was shutdown, it was then booted into a Linux environment using a BackTrack Live boot disk to make certain the test files were copied to the additional hard disk.

Writing the test files to the additional storage device was successful. In additional testing with Deep Freeze, the test files appear to be written to the storage device successfully without error. After examining the storage device in BackTrack Live, the files did not exist and were never written to the drive. Forensics Steady State is more

intuitive in that it acts exactly as the user would expect. Once attached storage devices are un-blocked, writes *are* made to the drive, as expected.

5 Conclusions - Forensics SteadyState

5.1 Discussion of Results - Forensics SteadyState

The results of all tests performed in Section 4 of this report helped meet all of the goals of this project. This section will discuss all of the findings according to which goal each test satisfied.

Goal 1 - To make a controlled environment solution that ensures that a sterile digital forensics environment can be created each time a new case is started by law enforcement investigators.

Goal 1 was met because of the results of the tests in Section 4.1. The logical write Tests 4.1.1 and 4.1.2 both performed as expected. Once the solution was rolled back, any files, folders, or applications added to the solution were deleted and the solution was back to its baseline state.

Tests 4.1.3 through 4.1.10 tested Forensics Steady State's ability to recognize raw hexadecimal writes made to different areas of the hard disk, from both a physical and logical view, and cache those writes within the snapshot VHD file. Tests 4.1.5, 4.1.6, and 4.1.10 all performed as expected. Any raw hex writes made in Partition 2 within unallocated space, within image.vhd, or within snapshot.vhd were all deleted upon rollback of the solution, as expected. Any writes made to the virtual C drive outside of the volume boot record were also eradicated upon rollback. Test 4.1.4, however, failed in that it allowed raw writes to be made to the unpartitioned space of the disk showing that unused area of the hard disk are not protected by Forensics Steady State.

Test 4.1.11 attempted to use the snapshot.vhd file located on one machine to update another. Updating one Forensics Steady State solution and using that machine's existing snapshot.vhd file to update multiple machines by replacing the older snapshot.vhd file was successful for solutions that were not sysprepped. This may not serve any practical purposes because the forensic workstations in a laboratory are most likely sysprepped beforehand. Test 4.1.12 explored the idea of performing the same test on two sysprepped machines, but failed proving multiple machines cannot be updated by simply using a snapshot file from another machine. In this case, either each machine would need to be updated separately or a new updated master image could be used to re-image each individual machine in a laboratory setting.

Tests 4.1.3, 4.1.7, 4.1.8, 4.1.9, and 4.1.12 all had unexpected results and are discussed later in Section 5.2.

Goal 2 - To make a controlled environment solution that is easy for forensic practitioners to use.

Goal 2 was met because of the results of the tests in Section 4.2. Test 4.2.1 focused on the process of rolling back the solution and the roll back time measurement of Forensics Steady State, Deep Freeze, and Drive Vaccine. In terms of ease of use, the user simply needs to restart or shutdown any of the three solutions to roll back and return to them to their initial state. Forensics Steady State has the slowest rollback time of the three solutions, with Deep Freeze falling slightly behind, and Drive Vaccine being the fastest. On average, Forensics Steady State took 02:44.80 to rollback to its original state.

Test 4.2.2 focused on the process of updating each solution and measured the update time of each of the three solutions previously discussed. In terms of ease of use, the process for each solution is described in detail in section 4.2.2. Each solution has its own intricacies involved with updating the baseline image, and can be fully understood with the provided literature for each. In this case, Drive Vaccine updated the fastest. On average, Forensics Steady State took 01:33.60 to merge the snapshot.vhd and image.vhd files and update the original baseline image.

Test 4.2.3 tested Forensics Steady State's ability to keep writes temporarily. This allows investigators the ability to shutdown a forensic workstation and continue working with the current state of the solution at any time without losing any data until the solution is rolled back. This can provide law enforcement with a form of safety net for digital investigations because the baseline image can only be restored if the option is chosen. Any unforeseen circumstances will not cause the loss of data.

Goal 3 - To make a controlled environment solution that does not substantially delay investigations.

Goal 3 was met because of the results of the tests in Section 4.3. Test 4.3.1 measured and compared the re-boot time of Forensics Steady State and a normal workstation with Windows 7 Enterprise installed. Unexpectedly, Forensics Steady State rebooted with an average time 01:02. On average, the re-boot time of Forensics Steady State was 17 seconds faster than the machine running Windows 7.

Test 4.3.2 compared the rollback time of Forensics Steady State to the re-boot time of the same Windows 7 workstation in Test 4.3.1. As expected, the machine running Windows 7 re-booted 01:25 faster than Forensics Steady State could rollback to its baseline image. Although each machine was performing different functions, it proves that rolling back to a pristine baseline image in Forensics Steady State merely takes 01:25 longer than a simple re-boot of a normal forensic workstation.

Test 4.3.3 measured the time needed for Forensics Steady State to merge its snapshot.vhd and image.vhd files permanently changing the baseline image. On average, it took 01:33.6 for the baseline image to be completely updated in the solution. This time allows for quick updates to be made to the baseline image for further investigative use.

Goal 4 - To have a solution that does not interfere with the forensic process.

Goal 4 was met because of the results of the tests in Section 4.4. Test 4.4.1 observed the behavior of Forensics Steady State when hard disk capacity was low or about to overflow. As expected, the solution gave an error message stating that disk space was low when an oversized image was being saved to the hard disk.

Tests 4.4.2 and 4.4.3 utilized functions of FTK Imager, X-Ways' Forensics, and SAFE Block to ensure proper functionality with Forensics Steady State. All tools worked as expected, and any temporary files such as images, case files, and recovered files were removed upon rollback of the solution.

Tests 4.4.4 and 4.4.5 tested the actions of attaching and write-protecting internal hard disks while Forensics Steady State was operating. Specifically, Test 4.4.4 verified

that inserting a fixed disk, write-protecting that disk, and attempting to copy files to the disk all behaved as expected. Test 4.4.5 tested the same procedure, but without write-protecting the fixed disk to make sure normal copying functions could be performed. Previous tests with older versions of Deep Freeze caused system crashes when internal disks were added to the system during operation. In additional testing with newer versions of Deep Freeze, the test files appear to be written to the storage device successfully without error, but after examining the storage device with BackTrack Live, the files did not exist and were never actually written to the drive. This result would not be desired for investigators trying to export reports or temporary case files to an external device, thus proving Forensics Steady State is a more viable solution for digital forensic investigations.

Goal 5 - To document the controlled environment solution behaviors proving forensic readiness.

Goal 5 was met through the production of this report. All of the tests developed in the Forensics Steady State Test Plan provides a forensic validation of Forensics Steady State. The tests were performed in a scientific manner and all results were carefully documented. Each test was reproduced several times to prove that particular behaviors of a specific test occurred each time the same test was performed.

Goal 6 – The reboot process of the solution should automate the roll back procedure and boot directly into a Windows environment after completion, as required by law enforcement organizations.

Goal 6 was met through the functionality that was added to the base implementation of Steadier State to create Forensics Steady State. With the addition of automating the rollback procedure, a user could essentially perform a simple shutdown or restart of their Forensics Steady State solution and have a pristine Windows 7 image restored without any future user interaction. Also, the additional functionality that scans the physical drive for extraneous files gives investigators notification that more files should be deleted manually before beginning a new case to prevent cross-contamination.

5.2 Interesting Results - Forensics SteadyState

Several tests run during the process of this project yielded interesting results. Tests 4.1.3, 4.1.7, and 4.1.9 all used Test Procedure 1 to make raw hex writes to the volume boot records of specific locations on disk. Test 4.1.3 included making hex writes to the volume boot record of Partition 2, the area of the disk storing image.vhd and snapshot.vhd. Test 4.1.7 made hex writes to the volume boot record of Partition 1, the area of the disk storing the WinPE operating system files. Finally, Test 4.1.9 made hex writes within the volume boot record of the virtualized C drive that can be viewed logically when booted into Forensics Steady State. Despite Windows error messages and the fact that the anti-virus software Microsoft Security Essentials did not catch that writes were made, all of the writes were allowed to be made to the boot sectors and resulted in failure to boot the solution after restarting or shutting down the system.

These unexpected results present a vulnerability of Forensics Steady State in that the boot sectors of any physical or logical partitions are not protected. This opens the possibility of viruses/malware infecting the boot sectors of the logical or physical boot

sectors on the hard disk. Although it is highly unlikely that an infection would occur in the boot sector of the WinPE partition, an investigator would still be able to maintain the probative value of the evidence located in the second partition because the virtual files are fully protected by Forensics Steady State. Any viruses/malware that may affect the boot sector of the second partition would rely on another piece of malicious code residing within the current state of the system, which can be rolled back and eradicated making the infection harmless to the evidence. Finally, if the solution no longer functioned as a result of an infection, an investigator would still be able to extract any current evidence from the solution because the virtual disks are completely protected by Forensics Steady State.

5.3 Future Work - Forensics SteadyState

In order to ensure Forensics Steady State can be used for an extensive period of time for forensic practitioners, some future work is required. Most importantly, Forensics Steady State should be extended for use with Windows 8 and future Microsoft Operating Systems to keep up to date with any forensic tools that require newer operating systems. Other future work includes making changes to the existing Forensics Steady State solution to make it more functional for law enforcement.

One such addition would be to prevent accidental rollback of the solution by adding in a warning prompt whenever the option is chosen in the pre-boot environment. Another recommended addition would be the provision of MD5 and SHA-1 hashes of image.vhd in the pre-boot environment to verify that a sterile environment has been

achieved and the integrity of the baseline image is preserved. This would include re-hashing image.vhd every time the system is rolled back.

Another functional addition to Forensics Steady State would be to add the ability to scan new system for drivers to be included in WinPE while before solution is deployed. This would eradicate any problems with the system's communication to hardware.

5.4 Summary - Forensics SteadyState

The results of this research show that Forensics Steady State is a viable solution for law enforcement to use. Without having the capability of using current Windows operating systems, law enforcement investigations have been delayed severely. Eliminating the need to wipe hard drives belonging to forensic workstations upon completion of an investigation will facilitate the possibility of completing more case investigations. Instead of spending part or all of a business day preparing a new forensic workstation, investigators can simply use Forensics Steady State's ability to rollback to a forensically sound baseline image within just a few minutes. Updating the solution is also an easy task and can keep the Windows 7 environment secure and forensic tools up to date.

This research does suggest that Forensics Steady State is vulnerable to malware or other infectious viruses that particularly affect the boot sectors of the solution. It is important to note that the competitor products, Deep Freeze and Drive Vaccine, both exhibit the same behavior when raw writes are made to the disk. Forensics Steady State also lacks in speed when it comes to updating and rolling back the solution, but the

minimal extra time required is negligible and still saves law enforcement the extensive process of starting from scratch after each investigation.

In conclusion, this project forensically validated and added features to the Steadier State solution created by Mark Minasi for use at the Rhode Island State Police Computer Crimes Unit. Forensics Steady State is an elegant, free Steady State solution that is forensically sound for use in digital forensic investigations.

6 References - Forensics SteadyState

"About VHD." *Windows MSDN*. N.p., 26 Oct. 2010. Web.

"Boot Configuration Data Editor Frequently Asked Questions." *Boot Configuration Data Editor Frequently Asked Questions*. TechNet, 25 Apr. 2007. Web.

Calvert, Charlie. "Charlie Calvert's Community Blog." *Booting from a VHD*. N.p., 2 Sept. 2009. Web.

"Computer Forensics, Malware Analysis & Digital Investigations: Forensic Analysis of "Frozen" Hard Drive Using Deep Freeze." N.p., 3 Oct. 2010. Web. Jan. 2013.

"Deep Freeze Enterprise." *Faronics*. N.p., n.d. Web. Apr. 2013.

Drive Vaccine V10.3 User Manual. N.p.: Horizon Data Sys Inc., 12 Mar. 2014. PDF.

"Encase Image File Format." - *ForensicsWiki*. N.p., 15 July 2015. Web.

"Forensic Investigations." *Forensic Investigations*. XYZ Media, n.d. Web. 15 Apr. 2014.

Hoog, Andrew. "ViaForensics." *ViaForensics MD5 Comments*. N.p., 30 Nov. 2008. Web.

Jain, Ranjana. "Simplifying Windows." *Virtual Hard Disk (VHD) Architecture Explained*. Technet, 23 Mar. 2010. Web.

Macheras, Panos. "Infrastructure Architecture Blog by Panos Macheras." *Implementing a Windows 7 SteadyState by Utilizing Differencing VHDs Files and the "Boot from VHD" Feature*. N.p., 23 Jan. 2011. Web.

Macheras, Panos. "Windows 7 SteadyState™ Solution Simplified!" *Technet.com*. N.p., 7 July 2011. Web. <http://blogs.technet.com/b/panosm/archive/2011/07/07/windows-7-steadystate-solution-simplified.aspx>.

Minasi, Mark. "Steadier State." *Steadier State*. N.p., 2012. Web. <http://www.steadierstate.com/>.

"Tools." *ForensicsWiki*. N.p., 4 Nov. 2014. Web.

"Trojan.Cidox." *Endpoint, Cloud, Mobile & Virtual Security Solutions*. N.p., 7 July 2011. Web.

"What Is Sysprep?" *What Is Sysprep?* Microsoft TechNet, n.d. Web.

"Windows SteadyState™ Will Be Phased out." *Microsoft Support*. N.p., 17 Sept. 2010. Web. <http://support.microsoft.com/kb/2390706/en-us>.

Part 2 Cloud Signature Creator

Abstract – Cloud Signature Creator

As the percentage of computer users that utilize cloud-computing services grows, more potential evidence for state and local law enforcement investigators is being stored with these cloud services and not on a local computer's hard drive. To address this problem, this project created a tool called *Cloud Signature Creator* as a solution that allows an investigator to locate potential areas of a computer's file system that contains evidence useful to their investigation. The Cloud Signature Creator solution leverages existing technologies and implements a new software application that provides the end user with a listing of files and locations that might indicate a cloud service was utilized on a suspect computer.

1 Introduction – Cloud Signature Creator

Today's Internet allows its users to store, create, and share documents and other files in the cloud. Suspects of a crime can store files of potential evidentiary value in these cloud-computing applications. State and local law enforcement investigators need to have a way to uncover this evidence, which may no longer be stored on a user's physical workstation.

1.1 Statement Of The Problem – Cloud Signature Creator

The increase in Internet users utilizing cloud-computing applications for the purpose of data storage, running a desktop environment, or processing some type of data provides some unique new challenges to state and local law enforcement when performing digital forensics investigations. Due to the rise of the use of such applications, criminals may no longer need to store the evidence of their crimes on a local device that a law enforcement officer is capable of seizing. As a result, law enforcement requires a means of determining whether certain cloud-computing applications were utilized on a computer, and to determine information about the user that might allow them to request information from the service provider.

1.2 Justification For And Significance Of The Study -– Cloud Signature Creator

According to the National Institute of Standards and Technology (Mell & Grance, 2011), cloud computing is a model for on-demand access to configurable

computing resources such as storage, applications, and services. A Fortune article (Griffith, 2014) cites Dropbox, Microsoft's OneDrive, and Google's Drive as the top consumer cloud storage providers. Dropbox claims to service 300 million users as of May 2014, Microsoft claims over 250 million, and Google provides for 240 million users as of September 2014. Each of these companies offer free data storage plans, making the option attractive to computer users. When a user stores a photo, video, or document with a cloud storage provider they can access it from any location on any device with an Internet connection. This ultimate portability provides a challenge for state and local law enforcement in performing digital forensics investigations.

Due to 4th Amendment limitations on the scope of search warrants, when a law enforcement officer requests data from a cloud service provider company, the officer must provide enough information to identify the user, the time range, and the type of information requested for which probable cause has been established. The use for an application that establishes cloud-computing application artifacts is to provide assistance to the investigating law enforcement officer as to where important information might be stored on the local computer.

1.3 Goals – Cloud Signature Creator

The goal of this project is to create a solution that may be used to determine specific files and locations that are modified during the use of a cloud-computing service. In order to accomplish this goal, the solution must:

1. Monitor file system changes that are a result of usage of a cloud service.
2. Accept information collected from the monitoring system and hash the files, removing duplicates from the list.
3. Compare the hash lists from multiple monitoring instances to show files that are present in both instances.

1.4 Summary Of Accomplishments - – Cloud Signature Creator

The result of this project was the creation of the *Cloud Signature Creator* application that may be used to provide the investigator with a listing of files with potential evidentiary value. The *Cloud Signature Creator* application met the goals specified in Section 1.3 by utilizing pre-existing software and implementing new software that parses data from the monitoring software.

2 Literature Review - – Cloud Signature Creator

This section discusses conceptual and technical materials that aided in the development of the *Cloud Signature Creator* application that corresponds to the goals of Section 1.3 by providing context and foundation to the research. The section will begin by discussing technologies that the application leverages to accomplish stated goals. Next, it will discuss related works that serve as the basis and inspiration behind this project. Lastly, the target audience of the *Cloud Signature Creator* application is defined.

2.1 Technologies

This section elaborates on the technical components required to build an application to reveal artifacts of a cloud-computing application's use. Specifically, this section will describe any software tools required and programming components that are utilized by the resulting application.

2.1.1 Sysinternals' Process Monitor

Sysinternals' Process Monitor is a monitoring tool for Windows that reports activity to file system, registry, and process/thread objects. (Russinovich & Cogswell, 2014) This Windows utility will be leveraged in this project to track changes made to the files stored on the computer's hard drive and report them to the investigator. The software developed on this project will pare down the information that Process Monitor provides to provide only data that is relevant to obtaining cloud-based evidence.

2.1.2 Microsoft Visual Studio

Microsoft Visual Studio (Microsoft Visual Studio) is an integrated development environment that is used to develop and test computer programs for the Microsoft Windows environment. Visual studio supports a multitude of computer programming languages, to include C, C++, and C#. C# has been selected for the primary programming language in this application due to it being a simple, modern, object-oriented programming language.

2.2 Related Works

This section provides insight into several related works that serve as both a basis and an inspiration to the *Cloud Signature Creator* application. These works include previous research projects that are a launching point for this project and commercial products used in the digital forensics industry.

2.2.1 Internet Evidence Finder

Magnet Forensics' Internet Evidence Finder, or IEF, (Magnet Forensics) is a software solution used to find, analyze, and present evidence found on computers, smart phones, and tablets related to Internet activity. One of the subsets of applications that IEF supports is cloud-computing applications. IEF supports a large number of applications, but with more cloud-computing applications being released there is a delay on when user's of the software will have access to evidence from a new cloud-computing application. The user selects the artifacts that the program should search for

from their interface. The number of artifacts the user has selected to search for will alter the amount of time the user must wait for the results to be completely reported.

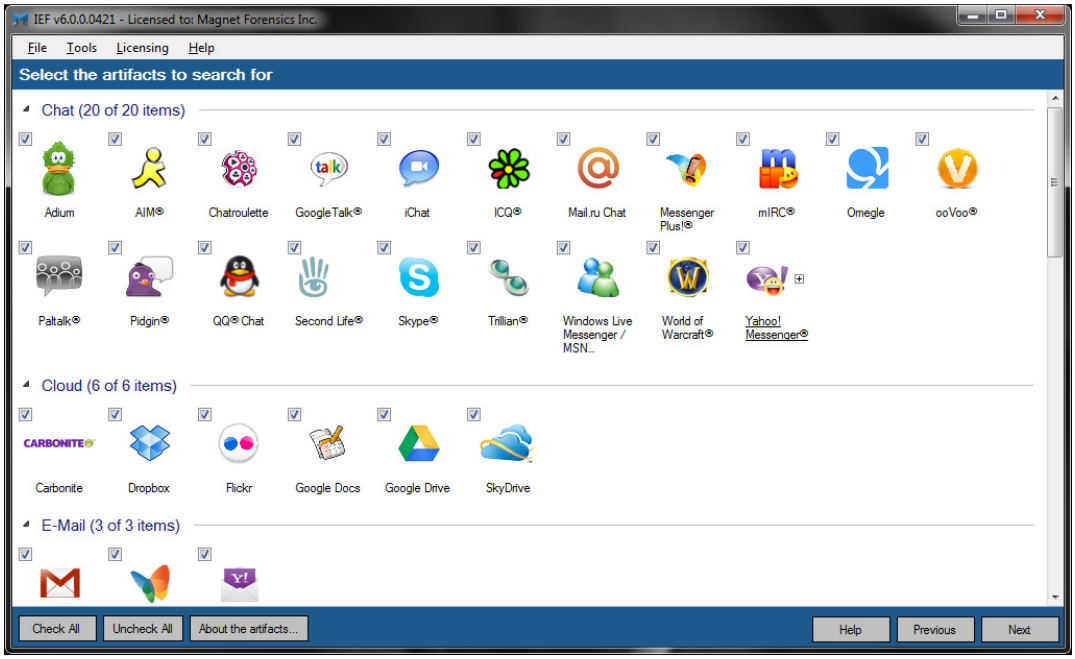

Figure 6 - Internet Evidence Finder Interface

2.2.2 EnCase Forensic

Guidance Software's EnCase Forensic software (Guidance Software) is a forensic solution that examiners frequently utilize to analyze hard drives and removable media. EnCase is an effective tool for data collection and investigations for active and deleted files, but is a full forensic tool for investigations of the entire contents of a hard drive. EnCase does not specifically search for and parse information of cloud services that had been utilized on the device it is analyzing, however scripts

can be created to attempt to parse information when one knows where that data is stored.

2.2.3 FTK

AccessData's Forensic Toolkit (FTK) (AccessData) is another digital investigation platform that examiners utilize to analyze hard drives and removable media. FTK is a powerful tool for searching and filtering data on the devices, but does not specifically target any cloud applications. The examiner would need to know where the data they are interested in is being stored by the cloud application in order to extract any information that might be able to further their investigation of the suspect's use of a particular cloud service.

2.2.4 Cloud Signature

Cloud Signature (Koppen, 2012) is a software tool created by the University of Rhode Island's Digital Forensics and Cyber Security Center (DFCSC) to provide the user with information about a specific list of supported cloud-computing applications that it detects on a computer. This tool parses the information it has detected in specific files known to be associated with supported cloud-computing applications, but the release of new applications and updates to existing supported applications causes the tool to be out-of-date too quickly.

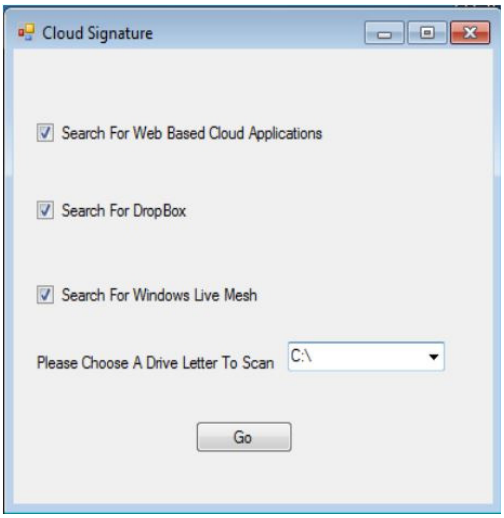

Figure 7 - Cloud Signature Interface

2.3 Target Audiences

Based on the goals of this project, two target audiences have been identified as potential users of the *Cloud Signature Creator* application: *Forensic Application Developers* and *Computer Crime Investigators*.

2.3.1 Forensic Application Developers

Forensic application developers that are interested in adding new cloud applications to their current products may have a need for this application to point out changes in the file system while using cloud services. These changes may indicate locations from which useful information might be able to be regularly extracted

2.3.2 Computer Crime Investigators

Computer crime investigators are users that have a particular case that existing tools do not support a cloud service that a suspect is known to be utilizing. This investigator must have a technical knowledge in order to use the service that their suspect is utilizing, but the use of this application might lead the investigator to some new evidence that could result in further legal process or an arrest.

3 Methods - – Cloud Signature Creator

This section consists of three sections. The first section discusses the procedures used to develop the *Cloud Signature Creator* application and discusses design decisions made throughout the process. The second section describes different use cases for the end user and explains how one's workflow might be. The third section discusses testing procedures used to measure the effectiveness of the application and to determine that the project met the goals stated in Section 1.3.

3.1 Application Development – Cloud Signature Creator

The development of the application can be divided into three main areas: the overall conceptual design of the application, the design and development of the user interface, and the design and development of the reporting function of the application. The conceptual design of the application takes Koppen's work on *Cloud Signature* (Koppen, 2012) adapts it to be able to detect general cloud-based installations on a computer to add to the tool's ability to stay relevant with quickly changing cloud applications. The user interface is both a control center for the application, and

instructions the user as the application leverages other tools to work properly. The reporting function of the new tool produces two different reports that the user may be able to utilize to further their investigation.

3.1.1 Conceptual Design – Cloud Signature Creator

The initial concept of *Cloud Signature* was to create a tool that parses through a hard drive and provides a user with the data that might be able to help them to sufficiently fulfill the needs of a service provider to respond to legal process. Keeping this tool up-to-date, not only on the cloud services initially supported, but for support of new cloud services, is requires significant effort. In fact, this became more evident with Magnet Forensics' purchase of Internet Evidence Finder and commercializing a similar product (Magnet Forensics). The very high rate at which the cloud services were updating and releasing new applications dictates the rate at which updates to *Cloud Signature* would need to be released in order to keep up with this growing industry. Keeping up with this manually, as *Cloud Signature* was originally designed, was prohibitive. The *Cloud Signature Creator* too created in this project was designed so that it could stay relevant by processing a hard drive of a suspect that is known to be using a cloud service that is not yet supported by the industry tools. To do this the application monitors changes made to a system while a user is utilizing a cloud service. These changes are monitored over several uses of the service and then generated into a report that provides file locations for an examiner to manually parse through to locate some data of potential evidentiary value.

The *Cloud Signature Creator* application does those by monitoring the file system's changes to determine which files were being created and/or modified during the use of a cloud service. To do this, Process Monitor was utilized to follow a

specified process running on a Windows computer. (Luttgens, Pepe, & Mandia, 2014) Process Monitor is already known to be capable of tracking file system changes in the sense of file creations and modifications. It can also be directed at one specific process, such as a web browser, and filter the results to include only changes made as a result of that process. This tool is a good basis to be leveraged for the purpose of file system monitoring in the *Cloud Signature Creator* application.

After determining how to address the file system changes, the project added the ability to compare several collections of the changes during the use of the cloud service in question. This collection of data results in a listing of files and hash values that were created and modified during the use of the cloud service, and the comparison takes two of these collections and results in a listing of hash values that are common across multiple runs. Unfortunately, some of the files that would be most important to the investigator will not have the exact same contents across multiple data collections. Therefore, the comparison phase also results in a listing of all files that were created and/or modified in both collection runs. The investigator can then utilize this information to manually look at the contents of any or all of these files for pertinent data.

3.1.2 Implementation – Cloud Signature Creator

For this project, the selected programming language was C# with Microsoft Visual Studio as a development environment. This environment was chosen for its ease of generating Windows graphical user interfaces and the robustness of the

standard libraries associated with the C# language (C# Programming Guide). The C# language's ability to create classes of objects and its library's prebuilt forms make it very simple to create Windows forms and dialogs.

3.1.3 User Interface – Cloud Signature Creator

The *Cloud Signature Creator* user interface has two phases. The first phase is the *data collection phase*. In order for this application to work to its fullest potential, the cloud service must be run several times while tracking the file system changes in order to remove some file anomalies that are a result of normal web browsing and computer usage. Multiple runs of the service will result in the most concise results of files that might contain useful user data.

The second phase is the *comparison phase*. The comparison phase takes input of two listings of files that were collected from previous runs of the cloud service as a result of the collection phase. The collected file listings are compared and produce a report of files that are the same across both collections. The figure below is the opening screen of the *Cloud Signature Creator* application.

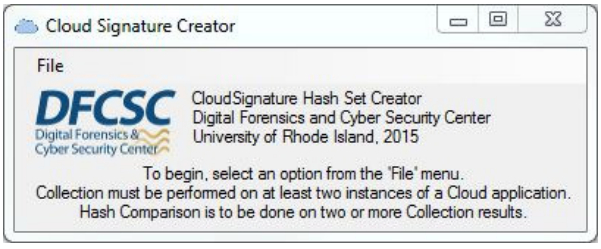

Figure 8 - Cloud Signature Creator Interface

3.1.3.1 Collection Phase – Cloud Signature Creator

The collection phase leverages Process Monitor to track the changes in the file system while a cloud service is being used. In order to do this, the user must set up the filter in Process Monitor to show results only for the web browser or other client with which the user is accessing the cloud service. The user is directed to start Process Monitor and adjust the filter before browsing to, and using, a cloud service. The interface asks for a location to a comma-separated value (CSV) file produced by Process Monitor, a Unique Identifier (UID) for the test, and a save location for the resulting CSV.

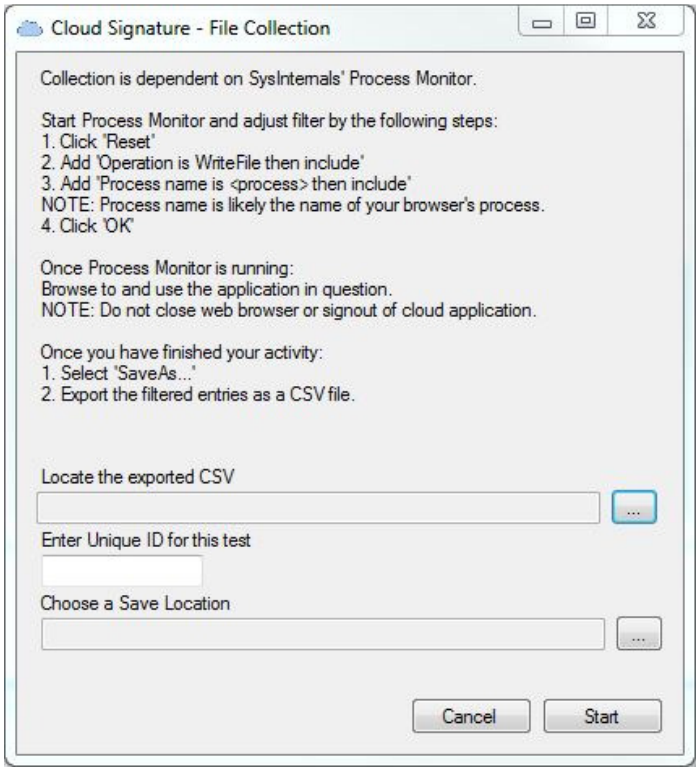

Figure 9 - Cloud Signature Creator File Collection Dialog

The *Cloud Signature Creator* application will process the CSV exported from Process Monitor to generate MD5 hash values of the listed files and output a CSV of the hash values for use in the comparison phase.

3.1.3.2 Comparison Phase – Cloud Signature Creator

The comparison phase of the *Cloud Signature Creator* application is utilized following the collection of at least two runs of the cloud service in question. The application requests the two CSV files generated from the collection phase to compare hash values. The result of the comparison phase is a listing of hash values that are the same across both runs of the collection phase. Additionally, the application provides an output of all file names across both runs. Theoretically, the hash values that match are files that were created during both runs and have the exact same content. While these files may be useful to determine that a particular website or cloud service was visited, it is not particularly useful to gather information from specific files accessed in the cloud service if they are not identical. The files that do not have matching hash values, but are stored in the same locations, would be the files that might contain data useful to the examiners investigation. Therefore, the *Cloud Signature Creator* tool will provide the user with a list of common hash values and a list of all files modified during the use of the cloud service. Once the user has these things, they can conduct a more focused investigation.

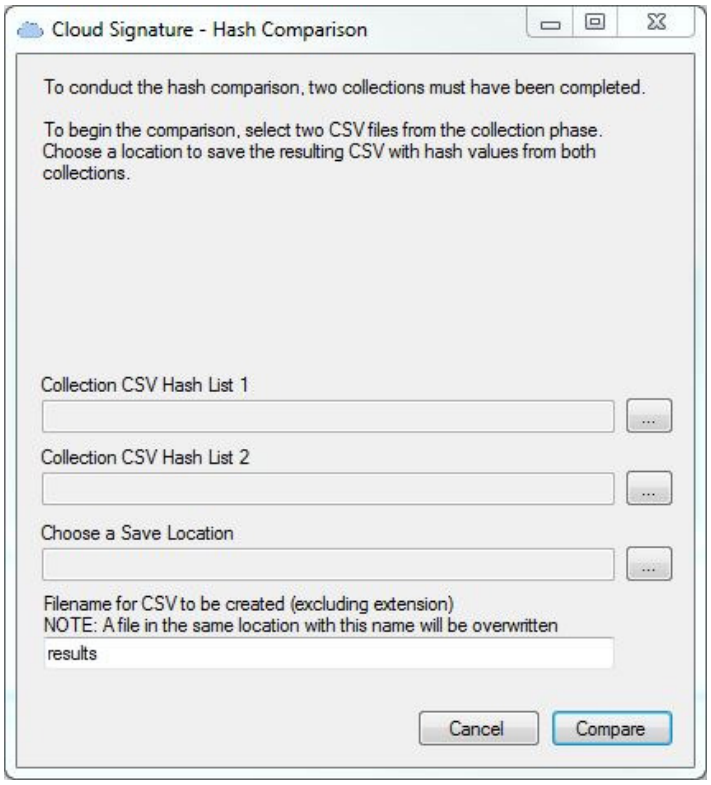

Figure 10 - Cloud Signature Creator Hash Comparison Dialog

3.2 Use Cases And Workflow – Cloud Signature Creator

In this section, two different use cases will be examined and a workflow will be introduced. The *Cloud Signature Creator* tool is essentially the combination of two different use cases for the application that are combined to make one workflow.

3.2.1 Use Case 1 – Collection

The first use case that will be discussed will be the file collection use case. The user in this case would start the File Collection Dialog and follow directions listed on dialog. This includes the use of Process Monitor and *Cloud Signature Creator*. This

activity can be utilized to show changes that occur from very specific actions that can be scripted by the user. At the end of this scenario, the user has a list of files that had been created or modified during the use of the application, and the hash values associated with those files. The following sequence diagram shows the users interaction with the applications involved for this use case.

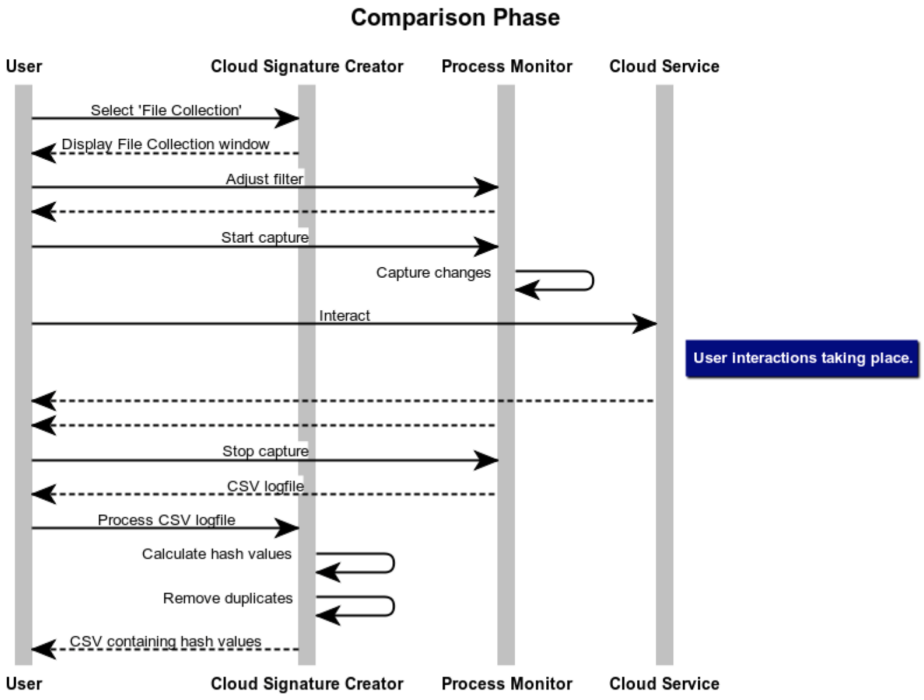

Figure 11 - Comparison Phase Sequence Diagram

3.2.2 Use Case 2 – Find Common Hash Values

The second use case is to compare hash values from two lists and receive a hash list with only values that are present in both lists. The user in this case would start with the Hash Comparison dialog in the *Cloud Signature Creator* and follow

directions listed on the dialog. This method will generate a list of common MD5 hash values with any two comma-separated value (CSV) documents, so long as the first value in each row is an MD5 hash value. The following sequence diagram shows the users interaction with the applications involved for this use case.

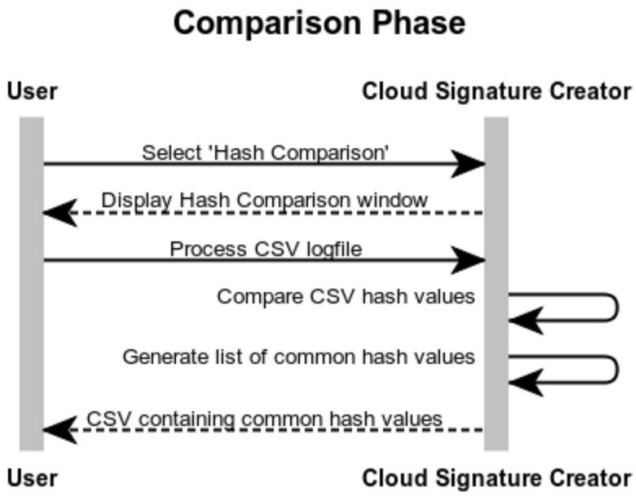

Figure 12 - Comparison Phase Sequence Diagram

3.2.3 Workflow

The minimum workflow for this solution is presented as a list below. This is the minimum workflow because it will include only two collections with Process Monitor. The more collections that are performed, the more narrowed down the list of common MD5 hash values should be. *Cloud Signature Creator* at first is a user intensive solution, however this is a necessity in some respects due to the nature of the problem being solved [Jerry – in the previous sentence elaborate on what respects require it to be user-intensive]. In the beginning, the user starts the two different

applications and adjusts the filter in Process Monitor to suit their needs for a particular cloud service. The minimum workflow is as follows:

1. Launch *Cloud Signature Creator*
2. Open File Collection dialog.
3. Launch Process Monitor.
4. Adjust the Process Monitor filter.
5. Start the Process Monitor capture.
6. Interact with the suspect cloud service.
7. End the Process Monitor capture.
8. Save the filtered data into a CSV file.
9. Process the CSV file with *Cloud Signature Creator* to hash and remove duplicate records.
10. Repeat steps 5-9 as a second collection run for comparison.
11. Open Hash Comparison dialog.
12. Locate two CSV files from the collection phase.
13. Process the hash lists with *Cloud Signature Creator* to create a list of common MD5 hash values.

3.3 Testing Procedures – Cloud Signature Creator

This section describes the methodologies used to test the implementation of the *Cloud Signature Creator* solution. This section will identify the experiments that were

conducted to evaluate how effectively this solution meets the goals defined in Section 1.3.

3.3.1 Testing Procedure 1

The purpose of Testing Procedure 1 is to determine that Process Monitor is capable of monitoring file system operations that will be appropriate to use for this solution. The Process Monitor application has dozens of different operations that can be the basis for filtering. With the number of files system changes that occur during normal usage, it is important to filter the results to get the most useful information possible. The default filter in Process Monitor removes any entries that are a result of Process Monitor running. This testing process will conclude with an analysis of the Process Monitor log and filtering the data to determine that the data can be filtered to show only changes made by the user while using the cloud service. The testing procedure is as follows:

1. Start Process Monitor.
2. Select Default Filter.
3. Begin capture.
4. Perform testing activity.
5. Stop capture.
6. Conduct analysis.

Testing activity includes opening files, folders, and applications, and browsing the Internet as normal usage of the system. Once the capture has completed and the analysis begins, the Process Monitor filters will be used to determine the best filters for the final solution's implementation.

3.3.2 Testing Procedure 2

The purpose of Testing Procedure 2 is to determine if *Cloud Signature Creator* accepts input of a Process Monitor log and is able to properly calculate a Message-Digest 5 (MD5) hash value. In order to conduct this test, a sample set of data will be created of a reasonable size to allow for hashing to be performed on each of the files in question with an external application. The testing procedure is as follows:

1. Start *Cloud Signature Creator*.
2. Open File Collection dialog.
3. Browse to sample Process Monitor log.
4. Enter Unique Identifier for this test.
5. Choose a save location.
6. Click "Start".
7. When completed, view results.
8. Verify the hash values generated match those from the external application.

Once the results are available in the save location, the resulting files will be analyzed for accuracy. The MD5 values will be externally verified using AccessData's FTK Imager to ensure that the hash values are being correctly calculated.

3.3.3 Testing Procedure 3

The purpose of Testing Procedure 3 is to verify that *Cloud Signature Creator* successfully and accurately removes duplicates from the collection phase be hash value. A sample set of data will be created of a reasonable size to allow for duplicate removal to be performed manually. The testing procedure is as follows:

1. Start *Cloud Signature Creator*.
2. Open File Collection dialog.
3. Browse to sample Process Monitor log.
4. Enter Unique Identifier for this test.
5. Choose a save location.
6. Click "Start".
7. When completed, view results.
8. Verify that known duplicate entries are removed.

To be considered a success, this test will result in the removal of duplicate hash values from the resulting CSV file. Verification will be conducted manually on a sample set of data.

3.3.4 Testing Procedure 4

The purpose of Testing Procedure 4 is to determine that the Comparison interface correctly compares output from different runs of the collection phase. Two sample data sets will be created to allow manual comparison. The testing procedure is as follows:

1. Start *Cloud Signature Creator*.
2. Open Hash Comparison dialog.
3. Locate two sample data sets.
4. Choose a save location.
5. Input a filename for resulting CSV.
6. Click “Compare”.
7. When complete, view results.
8. Verify that all common hash values are reported.

A successful test will result in the resulting CSV file being a single list of hash values that are contained in both collection hash lists along with the file names associated with the hash values from both collection CSV files.

3.3.5 Testing Procedure 5

The purpose of Testing Procedure 5 is to determine if the *Cloud Signature Creator* solution is a viable solution for directing the user towards a set of specific files for further analysis in the suspected use of a particular cloud service used via the Internet. The testing procedure is as follows:

1. Start *Cloud Signature Creator*.
2. Start Process Monitor.
3. Perform test-specific operations.
4. Analyze results.

3.3.6 Testing Procedure 6

The purpose of Testing Procedure 6 is to determine if the *Cloud Signature Creator* solution is a viable solution for directing the end user towards a set of specific files for further analysis in the suspected use of a particular cloud service used via a locally installed client application. The testing procedure is as follows:

5. Start *Cloud Signature Creator*.
6. Start Process Monitor.
7. Perform test-specific operations.
8. Analyze results.

4 Findings – Cloud Signature Creator

This section discusses the tests performed to validate the *Cloud Signature Creator* solution. Discussion will consist of results of testing procedures from Section 3.3 and dialogue on the strengths and limitations of the *Cloud Signature Creator* solution.

4.1 Testing Procedure 1 Results – Cloud Signature Creator

This set of testing is based on Testing Procedure 1, from Section 3.3.1. After the testing procedure was completed, results of the Process Monitor capture were reviewed and analyzed. For every second of computer usage, there are hundreds of operations that Process Monitor reports. The most commonly occurring operations, of thirty-six operations, were CreateFile, WriteFile, CloseFile, and ReadFile. Based on the analysis of these results, it has been determined to filter in only entries with these operation types.

The Process Monitor user can filter based on process name, which will only show operations selected that were triggered by a specific process. For example, a Process Monitor filter for use with this solution and a cloud service being run in Internet Explorer would look like this:

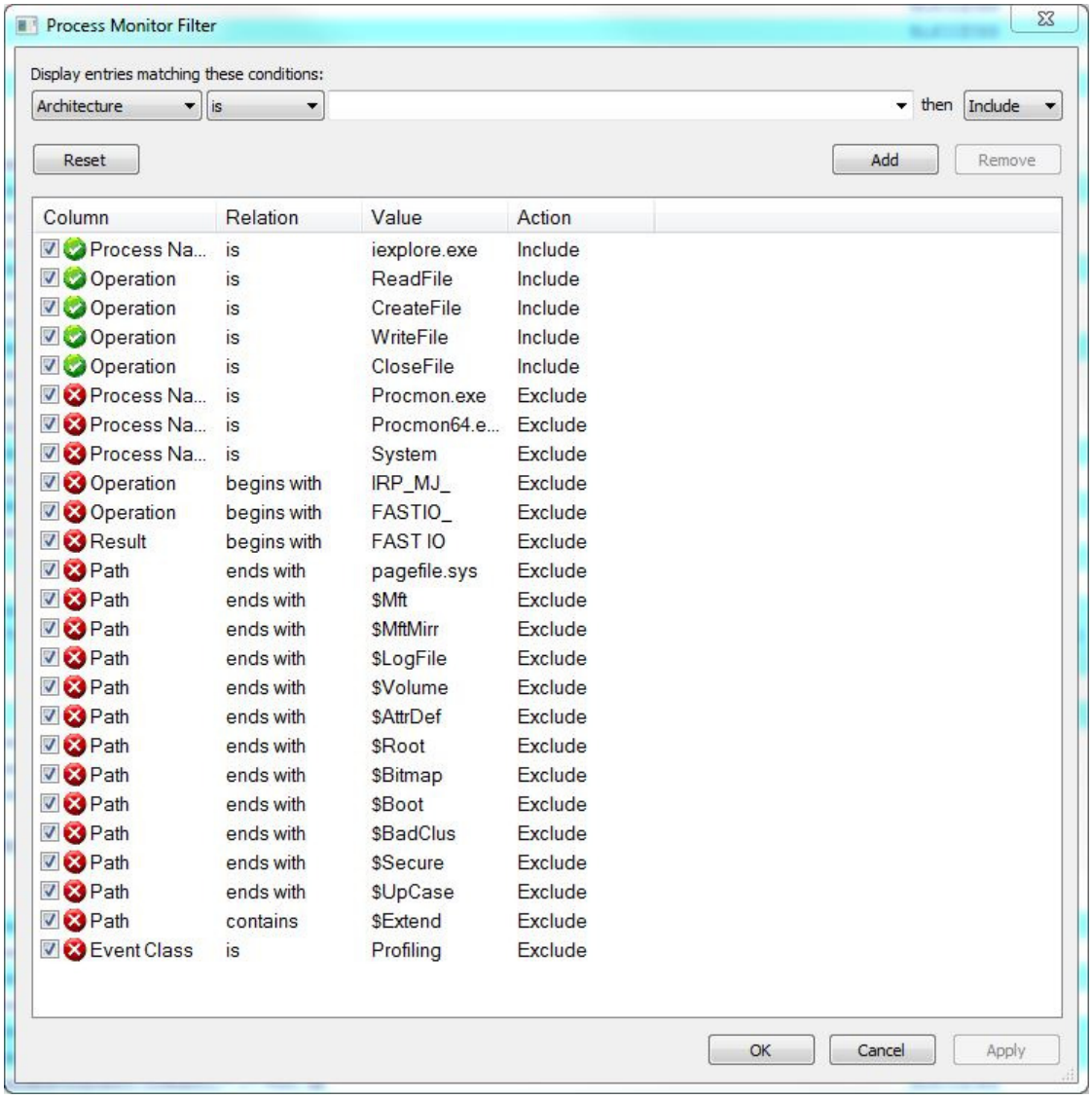

Figure 13 - Process Monitor Filter

In the Process Monitor Filter window, the green check marked rows are the filtering options that are to be included and the red check marked rows are the filtering options that are to be excluded from the final results. In this example, the above filtering resulted in a reduction of the number of events displayed to 13,196 from

272,883. This is only 4.8 percent of the total events, resulting in much less manual filtering and analysis from the end user.

Although this example filtered in a web browser, the user may have instead filtered in a client application for a particular cloud service. This would result in the events shown only being a result of actions taken by the cloud services application, rather than the web browser.

4.2 Testing Procedure 2 Results – Cloud Signature Creator

This set of testing was intended to determine that the *Cloud Signature Creator* tool was properly calculating MD5 hash values. For this test, a sample set of twenty entries was created in the format of a Process Monitor log. These known files were located and hashed with the use of FTK Imager (FTK Imager version 3.4.0, 2015). The files were located using the path provided in the Process Monitor log and were added to an AD1 custom content image file. An AD1 custom content image is a container to store multiple files without altering their contents. Once the files were added to the image, the image was processed with FTK Imager and a hash list of the enclosed files was generated. After running the test, the hash list generated by the *Cloud Signature Creator* tool was compared with the generated hash values from FTK Imager. In order to compare these values, both sets were loaded into an Excel spreadsheet and compared using Excel's comparison functionality (Microsoft Knowledge Base). As a result of the comparison, it was determined that the *Cloud*

Signature Creator tool correctly calculated the MD5 hash values for the files in question. The results were also manually verified.

4.3 Testing Procedure 3 Results – Cloud Signature Creator

This set of testing was intended to determine that *Cloud Signature Creator* properly removes duplicate hash value entries in the file collection phase. For this test, a sample set of twenty entries, to include five duplicates, was created in the format of a Process Monitor log. The files were located using the path provided in the Process Monitor log and were added to an AD1 custom content image file. Once the files were added to the image, the image was processed with FTK Imager and a hash list of the enclosed files was generated. The hash values were loaded into an Excel spreadsheet and duplicates were removed using the Remove Duplicates function (Microsoft Office Support). After the test was conducted using the *Cloud Signature Creator* tool, the resulting list and the sample set with its duplication removed were loaded into an Excel spreadsheet and compared using Excel comparison functionality. As a result of the comparison, it was determined that the *Cloud Signature Creator* tool correctly removed duplicate hash values for the files in question. The results were also manually verified.

4.4 Testing Procedure 4 Results – Cloud Signature Creator

This set of testing was intended to determine that *Cloud Signature Creator* properly compares the data from two CSV files and reports only the entries that report having that same hash value. For this test, a two sample data sets with twenty entries each were created. The entries contained a hash value and a file name. The files were located using the path provided in the Process Monitor log and were added to an AD1

custom content image file. Once the files were added to the image, the image was processed with FTK Imager and a hash list of the enclosed files was generated. Both data sets were loaded into an Excel spreadsheet and the columns were compared using Excel's built-in comparison features. After conducting the test, the resulting list of hash values and file names were reported correctly. This conclusion was reached after loading the resulting hash list and the generated data set into an Excel spreadsheet and utilizing Excel's built-in comparison features. The results were also manually verified.

4.5 Testing Procedure 5 Results – Cloud Signature Creator

This set of testing was designed to determine the viability of the *Cloud Signature Creator* solution in the investigation of cloud services by a forensic investigator. In this test a cloud service is accessed via Internet Explorer 11. Internet Explorer was used in testing because it stores its browsing data and caches files in a format that lends itself to parsing. For this test, a Dropbox account was setup with a small amount of known files. One Word document, one JPEG image file, one Excel Spreadsheet, one PDF file, and one text document were uploaded to Dropbox prior to the start of the test. These known files were accessed and opened via the Dropbox web portal. The same steps for the acquisition of these files from Dropbox were used on two separate occasions for the file collection phase and then passed on to the hash comparison phase. After running the test procedure, the individual runs of the collection phase reported 484 and 453 hash values. These sets were run through the comparison phase and the resulting list of files in common between the two runs

contained 434 files. Following the test runs, the two hash sets from the collection run were loaded into an Excel spreadsheet and compared using Excel's comparison features. The results confirmed those of the *Cloud Signature Creator* solution.

After an analysis of the results, we observed that some of the documents were able to be located in the file system after the testing was concluded and the web browser was shutdown. This includes the text document, the PDF file, and the JPEG file. The Word and Excel documents were not plainly observed, but evidence that a spreadsheet or document were viewed is present in the form of a JavaScript file that is setting the frame for the files. Contents were not located in the file set. The majority of the files found to be common across both collection phase runs were graphics and windows DLL files. These files could be used to confirm that the cloud service was utilized, but do not seem to contain any additional user data.

4.6 Testing Procedure 6 Results – Cloud Signature Creator

This set of testing was designed to determine the viability of the *Cloud Signature Creator* solution in the investigation of cloud services by a forensic investigator. In this test, a cloud service is accessed via a locally installed client application. The cloud service being tested is Dropbox. . For this test, a Dropbox account was setup with a small amount of known files. One Word document, one JPEG image file, one Excel Spreadsheet, one PDF file, and one text document were uploaded to Dropbox prior to the start of the test. These known files were accessed and opened via the Dropbox folder. In a locally installed client environment, Dropbox

has a background process that monitors the usage of the Dropbox folder to determine if new files or changes need to be synced with its servers to provide the user with the most up-to-date data in all locations. The same steps for the acquisition of these files from Dropbox were used on two separate occasions for the file collection phase and then passed on to the hash comparison phase. After running the test procedure, the individual runs of the collection phase reported forty-two and sixty hash values. These sets were run through the comparison phase and the resulting list of files in common between the two runs contained twenty-six files. Following the test runs, the two hash sets from the collection run were loaded into an Excel spreadsheet and compared using Excel's comparison features. The results confirmed those of the *Cloud Signature Creator* solution.

After an analysis of the results, we observed that many of the common files were system files. Additionally, several of the other files that were reported were some database files that would change had new files been modified, uploaded, or removed from the Dropbox folder. Note that the files being accessed are actually resident on the local computer, so their contents, if not encrypted, will be available in a Dropbox folder on the hard drive.

5 Conclusions – Cloud Signature Creator

The previous section presented the results collected from several tests that were outlined in Section 3. These tests were designed to address the goals laid out in

Section 1.3 and the overall viability of the solution proposed to help state and local law enforcement investigators identify files and file locations that might contain information to further an investigation involving cloud-computing services.

5.1 Goal 1 Conclusions – Cloud Signature Creator

The first goal of this project was to monitor file system changes that are a result of the cloud service usage. The Testing Procedure 1 Results, described in Section 4.1, discuss the findings of a test for file system monitoring and Process Monitor log filtering. The monitoring system utilized for this solution does a good job of tracking changes to files and folders in the file system. Additionally, the Windows Registry can be monitored with different operations than those that were selected as filtering options. The Windows Registry is a hierarchical database that stores data about the users and current configuration of a Windows system. With this solution, ignoring the registry was a decision that was made because of the fact that hash values were being utilized. Because the registry is always available while the system is running, and so many reads and writes are made to it, hash values of the registry are constantly changing. More useful to the *Cloud Signature Creator Solution* would be to take note of exactly which registry keys and values are being written and read and to provide these to an investigator as a specific location of potential evidence.

5.2 Goal 2 Conclusions – Cloud Signature Creator

The second goal of this project was to develop an application that receives input from the monitoring service and locates the files listed in a log file. Once the

application locates a file, an MD5 hash value of the file is calculated, and duplicate hash values are removed from the application's report. While the MD5 value is calculated correctly, the hash value is only calculated if the file can be located after the collection run. Perhaps more important than the files that can be located in the current file system after the cloud service was used would be the files that are no longer able to be located. Most likely, these files have been removed or deleted by the cloud service cleaning up after itself when usage is completed. As a result, the files that are deleted are never reported by the *Cloud Signature Creation* application.

The duplicate removal is successful and does seem to be a helpful feature. If the same file is created, read, written, or closed more than one time, being reported that files hash value or location several times only contributes to the growing number of results for the end user to sift through.

5.3 Goal 3 Conclusions – Cloud Signature Creator

The third goal of this project was to compare the hash lists from multiple monitoring instances to show files that are present in both instances. After the user performs two collection runs of with the *Cloud Signature Creator* solution, the resulting hash lists are compared to determine which hash values are common between the two lists. The *Cloud Signature Creator* solution was successful at comparing these hash values, but the results of the comparison are interesting. The resulting hash list of common files are, generally, system files or graphics that might be loaded with a particular web page in the cloud service's usage.

More important to the end user are the file's with hash values that do not match across different collection runs of *Cloud Signature Creator*. These files might actually be the ones that contain the contents of files that were accessed through the cloud service or information about the user of a cloud service. All files that could be located and hashed are reported by the collection phase, so the investigator could look into the files and locations that are listed in that location. However, files that could not be located because they had been deleted are never actually reported to the investigator besides being in the Process Monitor log.

5.4 Future Work – Cloud Signature Creator

The *Cloud Signature Creator* solution developed for this project was able to meet the goals described above. However, certain limitations exist in the solution that lend it to the possibility of additional work to improve the tool. The following sections discuss some of these areas of future work.

5.4.1 Monitoring System

Currently, the monitoring system is Process Monitor, which limits the use of this application to working only with Windows workstations. Additionally, the fact that this solution leverages an external tool for this aspect means that it relies on the support of the external tool for continued availability. If Process Monitor support is discontinued, the *Cloud Signature Creator* solution is in jeopardy of discontinuation. Aside from the support being discontinued, if the Process Monitor log file format is adjusted, the *Cloud Signature Creator* log reading functions may not work correctly.

As future work to help sustain the usability of this solution, the monitoring system should be integrated into the *Cloud Signature Creator* application. As a secondary advantage to this work, integrating the monitoring system would allow for a more automated process that would require less user input when it comes to working with the cloud service.

5.4.2 File Hashing Algorithms

In an effort to advance the usage of this solution, hashing algorithms could be updated to include other algorithms, such as SHA-1 and SHA-256 algorithms. A more interesting adjustment to the file hashing aspect of the application would be the use of a fuzzy hashing algorithm. Fuzzy hashing is a hashing method that hashes files in smaller sections, so as to be able to identify parts of files that are the same. In this case, it would be useful to know that two files are partially the same, indicating that maybe the rest of the file is session dependent and includes user data or date and time information.

5.4.3 Reporting

The reporting features of the *Cloud Signature Creator* application can be expanded to include some other data that would likely be useful to the investigator. One way to adjust the reporting of the current set of data given to the user would be to separate the graphic files and system files from the rest of the results to possibly show the end user files that might contain actual data that could identify some type of evidence.

Another possible change to the reporting would be to utilize the National Software Reference Library (NSRL) data set to reduce the hash values provided to the investigator by checking for known system files that are common on all computer systems and would have no information about the usage of the suspect cloud service.

Finally, the reporting of the *Cloud Signature Creator* application can include the reporting of files that were unable to be located and hashed due to them being deleted. These files might be useful to an investigator that could then attempt to carve the file's contents from unallocated areas of the hard drive. In order to do this, the investigator would also need some indication that these files might actually contain some information that is useful to their investigation, which the file's name and location may or may not provide them.

5.5 Summary – Cloud Signature Creator

In conclusion, the *Cloud Signature Creator* solution was successful in meeting the goals described in Section 1.3. It has potential to be helpful in an investigation for state and local law enforcement when the suspect is known to be using a cloud service that is not supported by current industry tools. Although this project was successful at meeting its goals, there are some clear areas for improvement that will allow the solution to be more helpful to state and local law enforcement agencies.

6 References – Cloud Signature Creator

AccessData. (n.d.). *Forensic Toolkit (FTK)*. Retrieved November 11, 2015, from AccessData: accessdata.com/solutions/digital-forensics/forensic-toolkit-ftk

C# Programming Guide. (n.d.). Retrieved November 11, 2015, from Microsoft Developer Network: https://msdn.microsoft.com/en-us/library/67ef8sbd.aspx

FTK Imager version 3.4.0. (2015, March 16). Retrieved April 23, 2015, from AccessData: accessdata.com/product-download/digital-forensics/ftk-imager-version-3.4.0

Griffith, E. (2014, November 6). *Who's winning the consumer storage wars?* Retrieved April 23, 2015, from Fortune: http://fortune.com/2014/11/06/dropbox-google-drive-microsoft-onedrive/

Guidance Software. (n.d.). *Computer Forensic Software - Encase Forensic*. Retrieved November 11, 2015, from Guidance Software: https://www2.guidancesoftware.com/products/Pages/encase-forensic/overview.aspx

Koppen, J. (2012). *Cloud Signature: An Application to Detect Cloud-Computing Application Artifacts.* University of Rhode Island, Department of Computer Science and Statistics, Kingston.

Luttgens, J., Pepe, M., & Mandia, K. (2014). *Incident Response & Computer Forensics* (Third Edition ed.). McGraw-Hill Education.

Magnet Forensics. (n.d.). *Internet Evidence Finder*. Retrieved April 23, 2015, from Magnet Forensics: http://www.magnetforensics.com/mfsoftware/internet-evidence-finder/

Mell, P., & Grance, T. (2011). *The NIST Definition of Cloud Computing* (Vols. 800-145). NIST Special Publication .

Microsoft Knowledge Base. (n.d.). *How to compare data in two columns to find duplicates in Excel*. Retrieved November 11, 2015, from Microsoft Support: https://support.microsoft.com/en-us/kb/213367

Microsoft Office Support. (n.d.). *Filter for unique values or remove duplicate values*. Retrieved November 11, 2015, from Microsoft Office Support: https://support.office.com/en-us/article/Filter-for-unique-values-or-remove-duplicate-calues-ccf664b0-81d6-449b-bbe1-8daaec1e83c2

Microsoft Visual Studio. (n.d.). *Home Page*. Retrieved April 23, 2015, from Visual Studio - Microsoft Developer Tools: https://www.visualstudio.com

Russinovich, M., & Cogswell, B. (2014, March 7). *Process Monitor v3.1*. Retrieved April 23, 2015, from Microsoft TechNet: https://technet.microsoft.com/en-us/library/bb896645.aspx