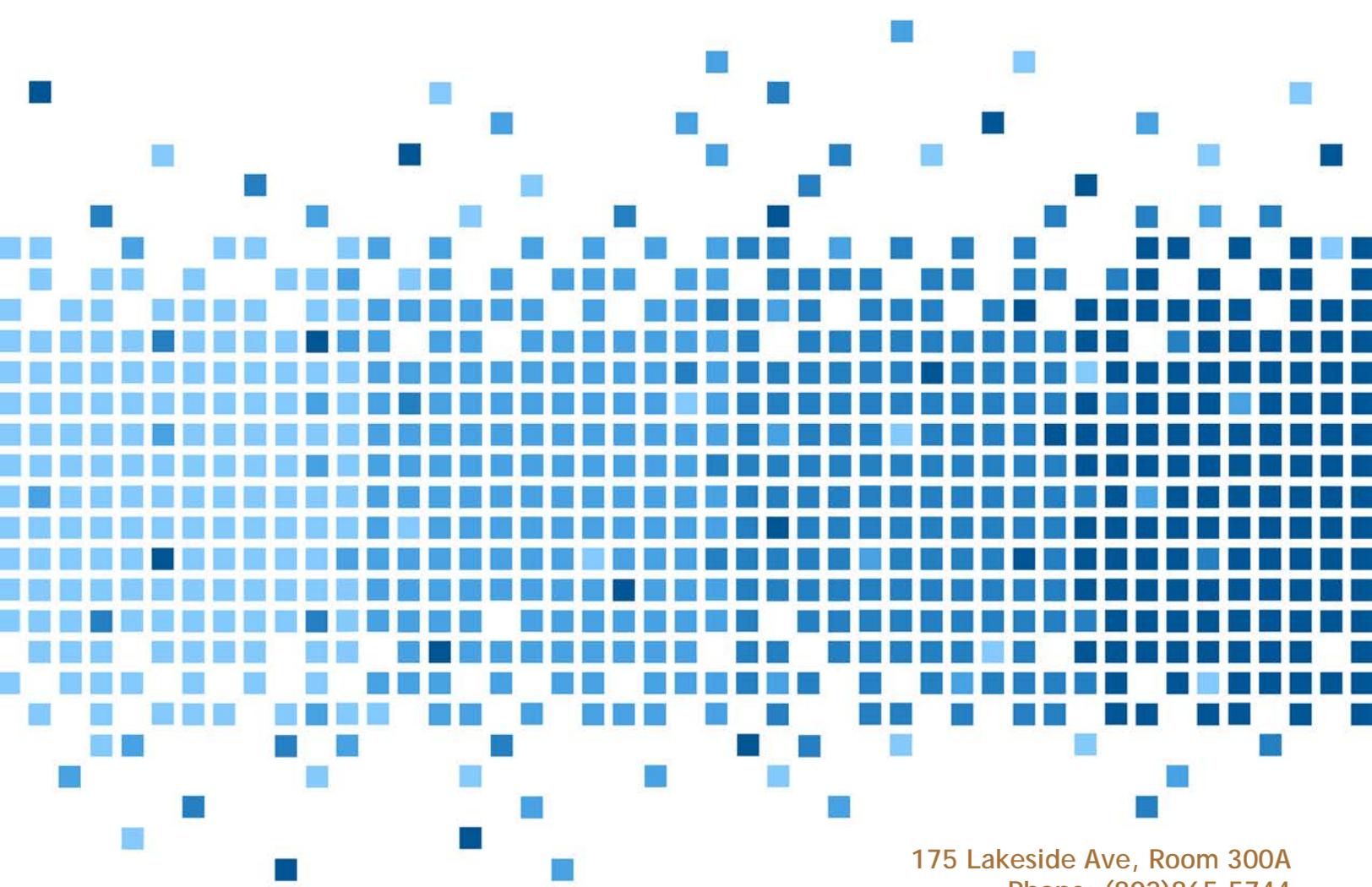


# [Raspberry Pi Forensics]



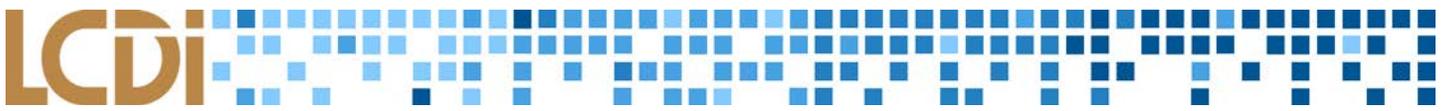


## Disclaimer:

*This document contains information based on research that has been gathered by employee(s) of The Senator Patrick Leahy Center for Digital Investigation (LCDI). The data contained in this project is submitted voluntarily and is unaudited. Every effort has been made by LCDI to assure the accuracy and reliability of the data contained in this report. However, LCDI nor any of our employees make no representation, warranty or guarantee in connection with this report and hereby expressly disclaims any liability or responsibility for loss or damage resulting from use of this data. Information in this report can be downloaded and redistributed by any person or persons. Any redistribution must maintain the LCDI logo and any references from this report must be properly annotated.*

## Contents

Introduction .....	2
Background: .....	2
Purpose and Scope:.....	3
Research Questions: .....	3
Terminology: .....	4
Methodology and Methods .....	4
Data Collection:.....	6
Analysis .....	8
Results.....	9
Conclusion.....	10
Further Work.....	11
References .....	12



## Introduction

The Raspberry Pi 2 Model B is a compact, programmable microcomputer designed to promote the education of basic computer science skills. Multiple input/output pins allow the Raspberry Pi to control other pieces of hardware.

The goal of this project is to evaluate the digital forensic capabilities of these new computers by using them as portable imaging devices. An image, as it pertains to digital forensic investigation, is a method of replicating a computer's hard drive by copying it into a series of files that reflects its contents. Forensic images come in two varieties: physical images that include the drive's unallocated space and are usually as large as the total capacity of the drive, and logical images that only copies the active data on the drive and is often much smaller in size.

Imaging the contents of a computer is an essential part of any digital forensic investigation because perusing the target drive directly will manipulate the data it contains and compromise the integrity of the evidence. Therefore, a mobile workstation would be an invaluable asset to analysts in the field, providing them with the ability to obtain data on-site that can be examined while remaining admissible in a court of law.

The Raspberry Pi 2 Model B is a programmable single card computer or "microcomputer." This allows a compact and mobile programming solution. It was designed with the intention of promoting basic computer science skills in education. The card features multiple I/O pins that allow the card to control other hardware.

To accomplish our goal, the Raspberry Pi 2 Model B will be used to create a portable imaging device. In digital forensics an "image" most often refers to a copy of a hard drive, or disk image that is compressed into a series of files. A physical image includes unallocated space and results in an image roughly the same size as the drive itself. A logical image copies the portion of the drive with active data and creates an image smaller than the original drive. A device like this is particularly useful in a digital forensics environment, as imaging computers is essential when conducting digital forensic research and analysis. Obtaining an image of drive is crucial when looking for forensic artifacts on a hard drive because modifying the original drive would cause the evidence on that drive to become inadmissible in the court of law. This device would allow an investigator to obtain an image of a client's media without collecting the evidence.

### Background:

Forensic images are usually made on one of several programs generally accepted as by the digital forensic industry, such as FTK Imager by AccessData and Guidance Software's EnCase. Researchers have also been able to utilize commands built into Linux operating systems to create an image of a drive. These techniques have been proven as effective through extensive assessment, but there is little exploration of applying them to a device such as the Raspberry Pi that supports only limited computation power. Through our preliminary research, we found several technical setbacks that prevent us from approaching these programs contemporarily: the Raspberry Pi's chipset architecture and processor are incompatible with the command line version of FTK Imager. The Pi's small number of USB ports (four on the model used in the project) presents problems as well, as it limits its potential data transfer speed and the small amount of power allocated to each port has the potential to cause problems during testing.

Research was also done into prior studies of mobile imaging, which proved largely fruitless. Very little study has been conducted on imaging with low-power machines, so we shifted our focus towards identifying methods most compatible with the specifications of the Raspberry Pi.



We decided to abandon Windows operating systems during this phase because we needed an OS that allowed the Linux commands to be issued to the microcomputer, and we could not allocate the time to solve the numerous compatibility issues. The lack of prior research to reference meant that much of the project was based on experimentation and hypotheses which were tested largely through trial and error.

Extensive research has been done on creating forensic images through software such as AccessData's FTK Imager and Guidance Software's EnCase, two popular programs in this field. There are many sources that describe how to use Linux commands to properly make forensic images from a command line interface but little exploration has been done in how these processes would run on the limited computing power of the Raspberry Pi. One thing that was discovered through the research was the inability to run command line FTK Imager on the Raspberry Pi due to incompatibilities with the processor and chipset architecture type. Another major setback is the lack of ports on the Raspberry Pi; it only has 4 USB 2.0 ports, limiting the speed for data transfer. Combined with the lack of power available to the ports can cause problems.

Our research in to mobile imaging was fairly short, as we could find very little reference online to alternate methods of imaging on a smaller, homebrew style device. Our research mainly consisted of identifying operating systems with Raspberry Pi compatibility and Linux commands used on a desktop environment for the creation of images. This research helped us toss the idea of working with Windows, simply due to the time constraint and compatibility issues with the Raspberry Pi. Due to the lack of previous research in this field, there were large portions of the project that were based entirely on our own hypothesis and were tested through trial and error.

## **Purpose and Scope:**

This project was designed to examine the newest technologies in the current market and attempt to use them as innovations to the digital forensics industry. A mobile imager made from a Raspberry Pi could potentially redefine how digital forensic investigations are conducted by adding a new level of affordability and simplicity to the process. Therefore, over the course of this project we prioritized our efforts towards designing an imaging method that the hardware and software of a Raspberry Pi machine could utilize effectively while maintaining its forensic authenticity.

The purpose of this project is to look at the capabilities of new technology as forensic tools. The first part of this project is exploring evidence imaging. The most important part of this project is to ensure that the images being created by the Raspberry Pi's are forensically sound.

There are several benefits of creating a mobile imaging unit based around the Raspberry Pi. This device is on the extreme low end of the pricing range for computing units, thus reducing costs to users. It is also very small, fitting easily in the palm of a hand. The size and pricing have the potential to make it a reasonable alternative for incident response teams. This tool would also allow forensic investigators to image digital evidence without taking the device from the user. With the addition of the touchscreen and executable script, this tool could also be easier for new users to quickly adapt to.

## **Research Questions:**

1. Will a mobile imaging unit expedite the imaging process?
2. Which Raspberry Pi operating system will produce the best performance?
3. How practical is a mobile imaging unit, opposed to conventional ways?
4. Which data connection to the drive will produce the best results?



### Terminology:

**Image** - In digital forensics, this term refers to a copy of the hard drive that is compressed into a series of files. Physical images include all information (ones and zeroes) on the hard drive whether the space is being used or not and the resulting image is near the same size as the actual hard drive itself. A logical image only acquires the parts of the hard drive that have active data and dismisses the rest of the drive. A logical image can be the same size or smaller than the original depending on the amount of data stored.

**FTK Imager** - A free extension of FTK 4.1, this tool is a powerful imaging program used to create forensic images of a drive that can be processed by most forensic examination software.

**Write Blocker** - A tool used to disable write permissions to a hard drive to prevent data destruction, alteration or contamination of data during the acquisition of a hard drive.

**Kali Linux** - A Debian-derived Linux distribution designed for digital forensics and penetration testing.

## Methodology and Methods

Three drives of varying capacity were procured for testing of the Raspberry Pi imager - 80GB, 500GB, and 1TB respectively. Each drive was formatted and given a clean install of a Windows operating system prior to their use in the project. The team's first step was to generate data onto the drives by surfing the web, creating documents and downloading files. Next, the drives were imaged with FTK Imager 3.1 (creating MD5 hash values to reference later) and assigned to separate LCDI workstations via write blockers connected by either eSATA or USB 3.0, allowing team members to begin testing.

Several operating systems were tested, including several variations of the Raspberry Pi's standard OS, Raspbian Wheezy, and a Kali Linux distribution designed for the Pi. We also explored the possibility of a Windows OS, which would enable us to use the GUI version of FTK Imager; however, further testing showed that only a mobile Windows OS was compatible with the Raspberry Pi's ARM processor, and we concluded that this operating system did not have the capabilities to handle programs that utilize full-size architectures, like desktop file systems.

The imaging was done using command line tools based on GNU dd which can be utilized to create rudimentary images of drives in .raw format. These variants, such as dcfldd and dc3dd, add enhanced functionalities to the command that address some of its shortcomings when compared to modern imaging software. For example, both allow synchronous hashing while images are being created, post-image verification and external logging, features dd did not have originally. Improvements like these will allow the Raspberry Pi imager to use more lightweight software to carry out its objective while still retaining an acceptable level of data assurance and admissibility in a court of law.

The 80GB drive was imaged using the command dcfldd, an alternate version of the older dd framework that is not initially available offered in the Kali Linux distribution designed for use with a Raspberry Pi. This was installed to the machine through the command line. The drive was attached to an external dock, then connected to an active write blocker and using a USB connection to interface with the Raspberry Pi.

Next, the command was executed and the resulting hash value was compared to one created earlier by the FTK Imager files. In order to develop the best syntax that produced consistent results, multiple subsequent tests were run. After each image was created, the drive containing the image was connected to the lab computer to store the test's log. The evolution of the syntax is shown in the logs, which are attached below. Tests 11, 15, 16, 19 and 20 provide no final time



in the command prompt; however, tests 7 and 15 did because they were executed in FTK Imager, which has progress statistics built in. We also used `dcfldd` to test the 500GB drive, but also decided to explore `dc3dd`, a patch of the `dd` command that updates in conjunction with it (as opposed to a spin-off of the antiquated original version, like the framework of `dcfldd`).

After learning about `dc3dd`, we attempted to switch over to it fully in order to compare the results to the outcome of the `dcfldd` tests. Although the forensic soundness of `dc3dd` is well-known, our experimentation with it only produced a single matching hash value, while running Kali Linux and using the internal hard drive in a dock. The lack of successful results leads us to consider this testing inconclusive.

Raspbian Wheezy was the OS chosen for the 1TB drive. We tried `dd`, `dc3dd` and `dcfldd` to image it, in order to directly compare the results of all three base commands. When writing to anything besides an internal drive on a workstation, an external dock is required to power a destination of this size because of the Raspberry Pi's inability to divert enough power to its connections. Tests 4, 5 and 6 were conducted on both the designated 1TB project drive as well as the internal 1TB drive of an LCDI lab computer.

The three drives tested were initially forensically wiped and given a clean install of a Windows. They were then utilized in a controlled environment in order to generate data similar to that of the average user; web traffic, downloaded files and a variety of document types, the purpose of this was to provide a reasonably similar drive to one that might be found in the field.

All three of the testing drives were first imaged through FTK Imager 3.1, the standard imaging software installed on the Leahy Center for Digital Investigation workstations. They were then connected to a write blocker via a powered SATA connection and the write blocker was connected to the lab computer by either eSATA or USB 3.0. After generating a hash value for comparison, each of the drives was be given to a team member to test within their setup. The operating systems we selected to test a Kali Linux distro specifically designed for the Raspberry Pi and several variants of the Raspbian Wheezy, the standard Raspberry Pi operating system. We investigated the possibility of attempting to run Windows in order to utilize the GUI version of FTK Imager, but it was concluded that the version of Windows designed for ARM does not have the processing capabilities to run programs designed for traditional desktop Windows architecture.

The 80GB drive was tested on a Raspberry Pi using Kali Linux using the `dcfldd` command. This utility is not installed on the version of Kali Linux distributed for Raspberry Pi and must be downloaded through the command line. The 80GB drive was connected to a powered write blocker using USB 2.0 to interface with the Raspberry Pi. The destination media was mounted on an external dock connected to the Raspberry Pi through USB 2.0. The external dock is a hardware device that an interface to mount and power a standard hard drive. The command was then tested and the hash values compared. After the image was created, the destination drive was then connected to a lab machine in order to process the log and compare it to previous results.

Subsequent tests were done to insure consistent results and develop the best syntax. As seen in the test logs below, the syntax was continuously evolving. For tests 11, 15, 16, 19 and 20 there was no time flag included in the command prompt and no time is provided. Tests 7 and 15 include times as they were done through FTK Imager which automatically includes progress statistics.

The 500GB drive was tested with both `dcfldd` and `dc3dd`. Once we learned about `dc3dd` we switched over and started recording all data that we could about `dc3dd`. This allowed us to test `dcfldd` versus `dc3dd`. All of the recorded tests for this drive were done with `dc3dd` and FTK imager. Although `dc3dd` is known to be forensically sound, we were only able to get 1 matching hash. The matching hash was when we used the internal hard drive in a dock with Kali. Because we were only able to get a matching hash once with this drive, we consider this series of tests inconclusive for the 500GB drive.

The 1TB drive was tested on Raspbian Wheezy, the standard Raspberry Pi operating system, using dd, dcfldd, and dc3dd. This drive was one of the base drives to test the functionality of the different commands. The drive was tested using a write blocker and the lab computers through an eSATA connection, or using the Raspberry Pi and the write blocker to write to an external drive. Due to the inability to use an external drive without power, (Raspberry Pi cannot provide the power needed for the drive) a dock is needed to mount the destination drive we are using to move the image to. For tests 4, 5, and 6, we used both the 1TB test drive and the 1TB internal drive from lab computer #7 to image the user folders on each drive.

## Data Collection:

Table 1: Equipment Used

Quantity	Type	Details
2	Internal Drive	1 TB Western Digital® Caviar® Black™ SATA drive 64MB Cache@ 7200 RPM(DataGen)
1	Internal Drive	1TB Seagate® Barracuda™ SATA drive 32MB Cache@ 7200 RPM
1	Internal Drive	500Gb Drive Western Digital® SATA drive 16MB Cache@ 7200 RPM (DataGen)
1	Internal Drive	80Gb Western Digital® SATA drive Specs UNKNOWN (DataGen) (Failed)
2	Internal Drive	80Gb Western Digital SATA drive 8MB Cache@ 7200 RPM (DataGen)
2	External Drive	SanDisk® Ultra 32Gb USB 3.0 Flash Drive
1	External Drive	Western Digital® My Passport Black 2TB USB 3.0 (WDBY8L0020BBK-01)
3	Write Blocker	WiebeTech® Forensic UltraDock™ v5 - storage controller - ATA / SATA 3Gb/s - e
4	Raspberry Pi	<a href="#">CanaKit™ Raspberry Pi 2 Ultimate Starter Kit</a>
1	External Dock	Masscool® SuperSpeed USB 3.0 SATA Hard Drive Dock For 2.5 inch and 3.5 inch drives.
8	Lab Workstation	Intel® Core™ i7-3770K CPU at 3.5GHz, 16 Gb of Memory, NVIDIA GeForce GTX 660 Ti, 64 bit Microsoft® Windows 7
4	Operating System	Kali Linux by Offensive Security, <a href="#">Raspberry Pi 2 version Model B</a>
	Operating System	Raspbian Jessie by the Raspberry Pi Foundation, based on Debian Jessie
	Operating System	Raspbian Wheezy by Raspberry Pi Foundation, based on Debian Wheezy

	Operating System	Microsoft Windows 7
	Software	FTK Imager 3.1 by AccessData, Windows and Command Line editions
	Software	Vmware Workstation 12.0.0 Pro
	Commands	dcfldd command line utility - based off the dd command and developed by the Department of Defense Computer Forensic Lab (DCFL)
	Commands	dc3dd command line utility - an updated version of dd with many other options
	Commands	dd command line utility - the base-line tool to copy large folders and files from one location to another
	Commands	md5sum dcfldd parameter - adds a md5 log to the image creation file after the image is done

Table 2: Data Collected

Test #	Test Type	Drive Size	Destination Drive Size	Image Time	MD5 Hash	Notes
1	Image Creation	1TB	2TB	N/A	f252cb8534ee876b6d0665abde79b467	Failure
2	Image Creation	1TB	2TB	3h 13m 52s	f252cb8534ee876b6d0665abde79b467	Match
3	FTK Image Baseline	500GB	1TB	1h 36m 29s	a99b58f9bc035faeb9fc4eb1af597860	Baseline image
4	User Folder	1TB - User Folder	2TB	2m 33s	7f0e7556606c17b78a642847b476a97b	No appdata or metadata
5	User Folder	1TB - User Folder	2TB	N/A	7f0e7556606c17b78a642847b476a97b	No appdata or metadata
6	User Folder	1TB - User Folder	2TB	N/A	04969ed925a80d1927f5f5d59381286e	No appdata or metadata
7	Image Creation	80GB	1TB	42m	d6f783bb56491006d9eee7a0af3fb2df	First Hash
8	FTK Compression Test (5)	500GB	1TB	6m	N/A	Failed - User did not have access
9	FTK Compression Test (7)	500GB	1TB	N/A	N/A	Failed - Computer Blue Screened
10	Pi Image Creation	1TB	1TB	N/A	f252cb8534ee876b6d0665abde79b467	Failed - write permission error
11	Pi Image Creation	80GB	1TB	N/A	15e19dd7a1aa749c0a9d243026a2c8cd	Time not included in parameters

12	Command Test	32GB	2TB	15m 32s	N/A	N/A
13	Pi Image Creation Test	80GB	1TB	3h 22m	00fb1fb64dedbdbb12eb3c6c2bd03d12	N/A
14	Dc3dd Command Test	500GB	1TB	5h 18m	a99b58f9bc035faeb9fc4eb1af597860	dc3dd imaged drives with matching md5
15	Pi Image Creation	80GB	1TB	N/A	f560009d6a820d42d0a48691ab4cd7b6	Time not included in parameters
16	Pi Image Creation (Issues Test)	80GB	1TB	N/A	8610cf421d9fa94b1e1d81e04c3bfb8c	Did not create a log file
17	Image Creation	80GB	1TB	43m	d6f783bb56491006d9eee7a0af3fb2df	N/A
18	Dc3dd Command Test	500GB	2TB	5h 11m 23s	ce8422b1efe67d4e647d0d5151763e4d	dc3dd imaged - non matching md5 - caused by change in command
19	Pi Image Creation	80GB	1TB	N/A	d6f783bb56491006d9eee7a0af3fb2df	Time not included in parameters
20	Pi Image Creation	80GB	1TB	N/A	d6f783bb56491006d9eee7a0af3fb2df	Time not included in parameters

## Analysis

In this experiment, we used three different commands (dd, dcfldd, and dc3dd), to test the ability of a Raspberry Pi 2 to become a mobile imaging unit. We started by imaging the drives using the LCDI Workstations through a write blocker with FTK Imager. This gave us the initial hash value that we compared the rest of the values to. The write blockers ensured that the original hash value were accurate based on the data gen. Then each of the team members obtained one of three hard drives, (1TB, 500GB, 80GB), and completed their own work on the Pi and on image creation. They utilized the commands listed above to create the images using a write blocker, the subject drive, the destination drive, the Raspberry Pi, and a hard drive dock. Two of the Raspberry Pi's have Kali Linux installed on them, and one Pi has Raspbian (the base operating system of the Raspberry Pi). By testing both Kali and Raspbian operating systems, we were able to test different environments to determine which was more successful.

We know that the Raspberry Pi will be slower than a normal computer based on the low memory and smaller hard drive size, so we are not expecting it to be as quick as one of the lab computers, but the hope is that it will work fast enough to be practical in a law enforcement environment. We are also constricted to using USB 2.0 (60 MBps max speed) connection with the Raspberry Pi, compared to the eSATA connection (3Gbps max speed) on the lab computers. That is what two members used for a data connection. One tested the possibilities of moving their data to an external drive directly in order to test the alternate setup and investigate the possibility of reducing the amount of hardware involved.



This command `dcfldd` was used in conjunction with the Raspberry Pi and the Kali Linux operating system. The evidence drives used was a 80GB, a 500GB and a 1TB Western Digital SATA drive along with a 32GB SanDisk flash Drive. In testing this command, experimentation had to be done to establish the exact syntax that would yield the desired result. It was important to make sure the entire drive was imaged, not just a single partition. If not paid careful attention to, this single parameter has the potential to create a different image and thus different hash. This initially caused confusion in our testing until the proper input file was noted. The command we found to be successful was:

```
time dcfldd if=/evidencedrive of=/destinationpath/filename.dd hash=md5
md5log=/logfilepath/logname.dd
```

This command was originally deemed of interest as it allows for on the fly hashing of the data, as indicated by the “`hash=md5`” option. The command generates a separate text file containing the hash to be checked against the original hash generated by FTK Imager. There is also the option to generate sha1 hashes; however we chose to focus on md5.

The “`time`” parameter of `dcfldd` was included in the final syntax for the purpose of tracking progress and having an understanding of how long the Raspberry Pi was taking to generate the imaging. This command allowed us to leave the workstation and have a detailed report on the time taken for each step in the imaging process.

The `dc3dd` command was tested with Raspbian Jessie, Raspbian Wheezy and Kali Linux. Both versions of Raspbian were not a good match because they weren’t compatible NTFS file system.

As a result of this we had to install a NTFS driver on the system. This method was unreliable and slow. Kali has built in support for this file system, allowing it to work with all drives. At the end of the project we figured out how to use an external hard drive with the Raspberry Pi. Initially we had many problems with the Pi not being able to put enough power out to the USB ports. After combining research from [Rageweb.Info](http://Rageweb.Info) and [Raspberrypi.org](http://Raspberrypi.org), we found a way to get the power to 1.2Amps instead of 600mA in Raspbian but not Kali.. Kali has the file that is similar to a BIOS under a hidden unmounted partition. We found this and with the code `max_usb_current=1` and this changed the available current from 0.6A to 1.2A. We also included `safe_mode_gpio=4` because the forum on [raspberrypi.org](http://raspberrypi.org) also included it.

## Results

The results of our tests show that it takes an average of 3 hours and 20 minutes to image and hash an 80GB drive, about three times as long as it would take using FTK Imager. One important discovery we made is, while constructing the syntax for the command-line imager, we were to specify imaging the entire physical drive and not reference any partitions. Potential complications with hardware, along with the large amount of time it takes to successfully image a drive, mitigates the effectiveness of the mobile imager as it stands at the conclusion of this project. We were able to partially attribute the speed issue to the lack of strong data connections available on the Raspberry Pi itself. Since all of the data was being pushed through the Pi for processing, the imaging process was being slowed by its USB 2.0 connections, which are limited to a 60 Mbps maximum data transfer speed, while the other devices utilized in the imager’s hardware setup were able to take advantage of an eSATA connector that hosts transfer speeds of up to 3Gbps. Until these issues are rectified, the Raspberry Pi cannot be called a practical imaging device on its own.

Larger drives proved troublesome to image. The 500GB drive tests showed discrepancies in the resulting hashes while using `dc3dd` as well as miscellaneous file system errors. We were able to verify that the hardware was not the cause of the different hash values but were unable to ascertain where the issue stemmed from. The 1TB drive tests were rarely completed without issue. During the first baseline tests on the lab workstations, FTK Imager was unable to complete its processes, resulting in us not having an original hash value. Further, the images that were produced during these tests



did not reflect any user folder data, leaving us with little information to work with on a forensic level. After moving on to testing with the Raspberry Pi imager, we received read/write file system errors that we could not address, barring us from creating a hash value for the image.

The syntax for a final command to create a forensically sound image using `dc3dd` was never decided on, because it was not the main focus of the project team. The imager's setup was originally configured with a Raspbian operating system, but file system incompatibilities led us to replace this with Kali Linux. This change showed the least amount of complications over the course of further testing, thus driving us to conclude that the Kali Linux distribution would be the most efficient OS in a mobile imager at this time.

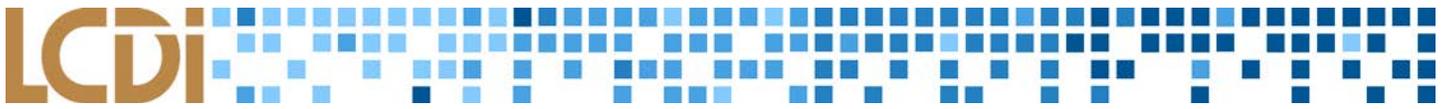
Through several tests it was established that 3 hours and 20 minutes was an average length of time for creating an image and hash of an 80GB drive using this command. This is roughly three times longer than FTK Imager. We found that the most important aspect of creating a forensically sound image was ensuring that the command specified the entire physical drive to image and not any of the partitions. Potential complications with this setup include the lengthy image time and the hardware. Through this testing we concluded that a portable imaging unit takes longer for the actual act of imaging than traditional techniques. The question of speed is partially connected to the types of data connection being used within the setup. The data transfer speed is bottlenecked at the USB 2.0 ports on the Pi, which are currently the only options offered on the Pi. This makes any other, more efficient, data connections in the setup irrelevant as all data was being forced through the Raspberry Pi itself for processing. Because of this time increase and the number of hardware devices that required independent power supplies, the Raspberry Pi imaging setup is not in its current incarnation more practical than traditional means of imaging.

There were many issues with getting the `dc3dd` command to work with the 500GB drive, from not getting the correct hash to file system errors. We got a baseline hash from FTK Imager that was compared to each of hashes generated on the Raspberry Pi. We were unable to find the cause of the changing hash with `dc3dd` but we were able to eliminate the hardware as the source. We were unable to finalize the command to consistently get a proper hash on this setup. This command was not the focus of our investigation and thus would require further tweaking of syntax

## Conclusion

The project team experienced a number of difficulties throughout testing which greatly affected their ability to produce concrete results. Eventually, a method to create a forensically sound image was established using a Raspberry Pi device running a Kali Linux operating system, an external write blocker, a hard drive dock and a variation of the command-line imaging tool `dcfldd`. This produced an image of the 80GB test drive with a matching hash value to the image created by FTK Imager on a lab computer. Most of the discrepancies were caused by the command only specifying certain partitions to image, as opposed to the entire drive. After working through that issue by fixing the command's syntax, we decided that this mobile imager concept has the potential to be utilized by investigators as a digital forensic tool, despite the Raspberry Pi's shortcomings regarding its hardware compatibilities and processing power. However, the larger drives needed to be removed from testing because of the marked increase in complications and the impractical amount of time it took to create an image. The `dc3dd` command was also unable to create forensically sound images without further modification of the syntax, so more testing would be required to utilize it instead of `dcfldd`.

We ran in to a number of difficulties through this project that resulted in being able to produce very few concrete results. We were eventually successful in creating forensically sound images when the Raspberry PI was setup to use Kali



Linux, a write blocker, an external dock and a variation on the dcfldd command. This setup was tested using a 80GB drive.

The best connection that was used was from the Raspberry Pi to an external dock with the destination drive. In order to create an image matching that of FTK Imager, the command must be written such that it copies the entire physical drive. We realized this after multiple failed tests, ending in different hash values. We were in fact copying the partition of the drive, rather than the full physical drive. Once that was solved, the image worked out, with the correct hash. It is limited in the aspects of its processing power and hardware setup but it does have the capability to be the base of a mobile forensic imaging unit when using this setup. As noted above, this setup was tested using only the 80GB evidence drive.

The 1TB drive was removed from the testing process due to complications in the file system compatibility and the excessive length of time required for imaging that made it impractical for our limited scale testing process. The 500GB drive was tested on a setup using a different command for comparison to the dcfldd setup. We were unable to properly tweak this command (dc3dd) in order to make it produce forensically sound images.

## Further Work

The area of creating forensic images using Linux commands needs to be explored more. As mentioned above, generating matching hashes has proven to be difficult when comparing them to hashes generated by FTK Imager. The reason for the hashes not matching has not been discovered yet. Further research will have to be done to find a forensically sound Linux command. Unfortunately the Linux command line version of FTK Imager is incompatible with the ARM architecture of the Raspberry Pi 2. At the time of publishing, there is no supported version of full Windows that would be capable of executing the GUI version of FTK Imager on the Raspberry Pi 2 Model B.

The team looked into using Windows IoT Core as a means of running FTK Imager with no success. Windows IoT core is an operating system created by Microsoft that allows users to execute their .NET and C# projects on IoT devices such as the Raspberry Pi 2. Unfortunately, it is not a desktop version of Windows, and FTK Imager cannot be installed. This presents an issue to examiners. If we trust FTK Imager to give us accurate hashes, what Linux command(s) are going to give us the same hash for the exact same drive. There should be no differences in the source drive, as the group has been using a hardware write blocker for each image.

## References

AccessData. *Using Command Line Imager*. Lindon, UT: AccessData, n.d. PDF.

Bearnes, Brennen. "Kali Linux on the Raspberry Pi with the PiTFT." *Overview*. Adafruit®, 21 Apr. 2015. Web. 11 Dec. 2015.

Microsoft. "Learn about Windows 10 IoT Core." *Windows IoT*. Microsoft, n.d. Web. 11 Dec. 2015.

Rageweb. *"/boot/config.txt in Kali."* *Rageweb.Info*. Rageweb, 07 Nov. 2013. Web. 03 Dec. 2015.

"Raspberry Pi • View Topic - Max\_usb\_current=1 Ineffective on RPi 2 B." *Raspberry Pi • View Topic - Max\_usb\_current=1 Ineffective on RPi 2 B*. Raspberrypi.org, 26 Mar. 2015. Web. 03 Dec. 2015