Collaborative Digital Forensics: Architecture, Mechanisms, and Case Study

by

Michael Kent Mabey

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved July 2011 by the
Graduate Supervisory Committee:

Gail-Joon Ahn, Chair
Stephen S. Yau
Dijiang Huang

ARIZONA STATE UNIVERSITY

August 2011

ABSTRACT

In order to catch the smartest criminals in the world, digital forensics examiners need a means of collaborating and sharing information with each other and outside experts that is not prohibitively difficult. However, standard operating procedures and the rules of evidence generally disallow the use of the collaboration software and techniques that are currently available because they do not fully adhere to the dictated procedures for the handling, analysis, and disclosure of items relating to cases.

The aim of this work is to conceive and design a framework that provides a completely new architecture that 1) can perform fundamental functions that are common and necessary to forensic analyses, and 2) is structured such that it is possible to include collaboration-facilitating components without changing the way users interact with the system sans collaboration. This framework is called the Collaborative Forensic Framework (CUFF).

CUFF is constructed from four main components: Cuff Link, Storage, Web Interface, and Analysis Block. With the Cuff Link acting as a mediator between components, CUFF is flexible in both the method of deployment and the technologies used in implementation.

The details of a realization of CUFF are given, which uses a combination of Java, the Google Web Toolkit, Django with Apache for a RESTful web service, and an Ubuntu Enterprise Cloud using Eucalyptus. The functionality of CUFF's components is demonstrated by the integration of an acquisition script designed for Android OS-based mobile devices that use the YAFFS2 file system.

While this work has obvious application to examination labs which work under the mandate of judicial or investigative bodies, security officers at any organization would benefit from the improved ability to cooperate in electronic discovery efforts and internal investigations.

For my wife, Heidi, who still remains my greatest success story.

ACKNOWLEDGEMENTS

I have truly been blessed to have many wonderful people placed around me in my life that have helped me in one capacity or another in my efforts to finish this degree. Saying thank you just doesn't seem adequate, so I hope it is clear that in saying it I mean to indicate a much deeper emotion.

First, I want to thank my wife for never giving up on me, for bringing me food on so many days when I couldn't leave campus, and believing in me when I had serious doubts about my abilities to succeed.

To my parents, thank you for letting me grow at my own pace and in my own way, while instilling in me a high regard for the virtues of a good education and helping me with boosts of morale and finances when I needed it. To my brother, Dr. Glen W. Mabey, thank you for being a source of encouragement, motivation, and wisdom so many times when I needed it, and for setting a good example of hard work and dedication for me to follow.

To my committee chair, Dr. Gail-Joon Ahn, thank you for being able to see what I could not so many times, for helping me focus on the next critical step in my research, for being so approachable, and for being such an easy person to work with. Dr. Yau and Dr. Huang, thank you for your service on my committee and for expecting excellence in my work in your classes. To Dr. Chad Mano, thank you for playing such a critical role in igniting my desire to study computer security, and for being such an encouraging mentor.

To so many other family members, friends, study partners, classmates, and teachers, thank you for sharing yourselves with me and helping me grow to become the person I am today.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Computer crime has swiftly evolved into organized, and in some cases state-sponsored, cyber warfare. Between the Pentagon's recent declaration "that computer sabotage coming from another country can constitute an act of war," [24] and the recent escalation in attacks which have included those launched against Sony [45, 44], the US Senate [46], Lockheed Martin [28, 49], and other companies [18, 26, 12, 30, 31, 16], it is very clear that this important issue deserves the attention of the brightest and most talented security professionals, for certainly there are more attacks to come [25].

1.1 Challenges Related to Digital Forensics

Unfortunately, the digital forensics software solutions used by those combating cyber crime are too limited to take on the challenges that they will inevitably face as crimes are committed with emerging technology. Before long, fundamental changes in the industry will make many of the forensic techniques used today obsolete [22]. In this critical time, a new approach is needed which is robust enough to meet current and future demands.

1.1.1 Time Inefficiencies

Although many contributing elements to the problem with forensics software can be identified, the heart of the problem is that current examinations are too time-inefficient. The three key factors which contribute to forensic examinations taking too much time are (i) software limitations, (ii) size of evidence data, and (iii) increased examiner workload.

Software Limitations

Single workstation computers have served as the primary tool of our society's computing needs for a long time. Not until Foster and Kesselman's work in 2004 [19] was the idea of grid computing formalized. Only then were businesses able to take advantage of the resources of a large set of computers through computational and storage management mechanisms.[1]

Of course, today this technology has advanced to become cloud computing, which provides Infrastructure-, Platform-, and Software-as-a-Service at an incredible rate of scalability to a world that finally has the bandwidth necessary to make the offering viable. However, even with cloud computing being met with widespread enthusiasm from researchers, developers, and consumers alike, we would do well to remind ourselves that cloud-based services still have a ways to go before being fully mature, particularly in the area of digital forensics.

Because single workstations have been the main method of computing for so long, the majority of software development naturally centered around the use of single workstations, digital forensic software not excepted. With the evidence data sets being as large as they are, a single computer simply does not have the resources to deliver analysis results in a timely manner.

Since forensic analysis techniques depend greatly upon what operating system and file system the data originally resided, forensic software is typically written to be used on a small set of popular systems. Because digital forensic labs can be tasked with analyzing any type of system in existence, a wide range of tools must be purchased to be prepared for an examination. Even tools which are flexible with the types of systems they can analyze need to be run on the operating system for which it was compiled. This problem is of course reduced when a tool is open source, but often such tools lack in documentation or go long periods of time without being updated.

[1]Although CPU scavenging and volunteer computing projects like SETI@home were able to exploit the resources of networked machines across the Internet much earlier than 2004, no toolkit was in existence that could be used by developers to accomplish such computing distribution on a wide-scale basis until the Globus Toolkit was created, the effort of which was lead by Ian Foster, Carl Kesselman, and Steve Tuecke.

Size of Evidence Data

Today a 1TB hard drive can be purchased for about $60 and the average hard drive cost per GB is less than $0.10 [3]. Such low cost makes terabyte-sized systems commonplace among even non-tech-savvy consumers. With such a proliferation of huge storage systems filled with user data, like the game World of Warcraft: Cataclysm that requires 25GB, examiners are up against a mountain of stored data to work through [42]. Seven years ago, when building a terabyte-class storage system still cost over $800, it took Roussev and Richard four days to open an 80GB target [42] using FTK [8]. The problem is compounded when the situation involves a RAID [47], Network Attached Storage (NAS) unit, or other large storage method that is shared among individuals or employees.

Increased Examiner Workload

As if insufficient tools and large datasets were not enough, digital crime continues to increase in popularity [37, 27, 38], naturally resulting in more investigations. Furthermore, state-sponsored cyberwar promotes the development of increasingly sophisticated software. Simply trying to keep up with the latest methods of penetration, exfiltration, and attack is insufficient to accommodate the pace of digital crime. In addition, when cases become backlogged, only those designated as more urgent are worked on, potentially leaving suspects' co-conspirators at large, capable of making more victims out of innocent people.

To complicate things further, although a large number of companies and organizations have incident response teams working to analyze security breaches, malware infections, and other security incidents, their findings largely remain private and are not publicly shared for reasons of reputation maintenance, a lack of suitable conduits through which such information can be shared in a meaningful way, and because of legitimate concerns of exposing infrastructure and system information that malicious hackers are constantly seeking to obtain. The result of these conditions is an every-man-for-himself approach, where organizations essentially analyze the same or similar incidents many times, independent of each other and the expertise that could be contributed in a joint effort.

1.1.2 Heterogeneity of Evidence Data

In addition to the difficulties arising from the time inefficiencies of contemporary software, a second significant contributing element to the problem at hand is the diversity of devices capable of capturing data and the diversity of the data captured by those devices. Forensic labs currently need to be equipped with a wide range of software and equipment to be able to analyze whatever type of device is part of the examination, which can be quite costly and require separate training for proper operation. Additionally, there are few, if any, methods of automating the detection of any corroboration between two different types of evidence, e.g. network traffic data, hard drive contents, et cetera.

Another type of research area that should receive much more attention despite the tremendous difficulties associated with it is the creation of new abstractions of forensic data. By this we mean such data as Internet and social network information, keystroke fingerprinting [36], and other types of peripheral information that could significantly improve the credibility and authentication of evidence.

1.1.3 Application Domains

Many important industry sectors could benefit from a secure and robust collaborative framework with which to perform forensic examinations, but a solution does not currently exist that can also maintain the rules of evidence. Because of this, examiners must often resort to methods of sharing information that are can be quite cumbersome and even prohibitively difficult. For example, sharing a hard drive involved in a case currently means doing one of two things. Either the providing agency will store the image of the hard drive on a server and set up special access controls for that image so the other agency can access it, or an exact duplicate of the drive is shipped to the other location. The security challenges associated with these options are easily apparent.

The most obvious domain for a deployment of a system which solves these issues is law enforcement, especially considering the high likelihood of an occurrence similar to the one hypothesized below, where multiple agencies or departments need to quickly and efficiently share investigative results to catch suspects at large.

Most companies have a great need for forensic services at one point or another. While it is more economical for some companies to hire the services of outside security professionals that have the experience and tools to perform the needed operations, others have fully staffed incident response teams that need to maintain their own stock of software and tools. In the case of both the contract security consultants and the incident response teams, the individual professionals have undoubtedly developed their own scripts and tools when something doesn't satisfy a specific need. Most common forensics tools don't allow much in the way of adding custom scripts.[2]

1.2 Requirements for a Solution

The thing that would help this situation more than anything else is a secure and robust infrastructure to facilitate *collaborative forensics*, which we define as the willful cooperation between two or more forensic examiners during any step in the forensics process, for the benefit of sharing specialized knowledge, insight, experience, or tools. The advantages of collaboration are well known, but two are of particular interest to us. First, collaboration allows people to draw from others' expertise, which is invaluable when working on problems of a diverse nature or when the problem set of a job constantly changes. Second, collaboration is a method of spreading a workload, which

[2]There are at least two tools that do allow for user-created scripts to be used with the tool: Digital Forensics Framework (DFF) [4] and EnCase [29]. However, DFF is mainly limited to executing commands in an integrated shell or a Python interpreter, and EnCase only supports its own object-oriented "EnScript" scripting language.

results in less time needed for the job to be completed. These two advantages make collaboration a favored strategy in time-sensitive environments.

Consider the following hypothetical scenario. Past crimes committed by members of a white supremacists group have largely been contained to a single state. However, with a recent expansion of operations to include a higher level of digital organization and recruiting, the group has begun to spread its activities across multiple states. In such circumstances, further investigation efforts may call for a combination of state and federal efforts, where previously only state enforcement was involved. These circumstances result in investigation files being stored in multiple locations by multiple agencies, which adds a new layer of complexity to sharing and cross-referencing critical information relating to this group.

Even in more localized investigation cases, evidence seizure may yield a variety of digital evidence, such as a mix of Windows, Linux, and Mac computers, cell phones, GPS devices, gaming consoles, et cetera. Since examiners must be certified to work on a particular type of evidence (depending on the investigating agency), such a workload must be split up among personnel. Furthermore, because there is no tool which can accommodate all evidence types, the evidence presentation lacks uniformity in format and structure.

While many generic collaboration solutions exist today, none of them have been crafted specifically for the needs of the digital forensics industry. To be truly effective, a collaborative forensics infrastructure should maintain the strict privacy and integrity principles the discipline demands, while also giving examiners the flexibility to communicate however is best for the situation. For example, when communicating quick ideas or assignments, text-based messaging may be the natural medium of choice. However, when more detailed explanations are required which may prompt a question and answer type of dialog, voice or video conferencing may be better suited. Whatever the circumstances may be, any communication regarding the case evidence should be logged and remain coupled to the evidence being discussed, both for future reference and for compliance and integrity checking purposes. This demands a level of robustness that is simply not offered by collaboration tools at present.

Beyond just communication, collaboration also implies a sharing of resources. For a proper exchange of data (whether it be files needing to be analyzed or the results of an analysis), there must first be a uniform representation of that data, and then a common storage space solution where all collaborators can keep their resources secure. This will require the establishment of standards to ensure that all parties can access and interpret the data. Means to efficiently manage resources will also be needed.

If examiners are to collaborate on a large scale, it will also be crucial for this infrastructure to provide vast amounts of computing power, which is best accomplished through some distributed processing method. Ideally, a distributed processing solution would also include scalable resources. Because there is not a single technological solution that will properly meet this need for all organizations, there must be a generic way to interface for such processing resources.

To best facilitate collaboration among examiners, a collaborative forensics solution should not be limited to supporting its use on a small number of operating systems. This would hinder the collaboration process and may exclude experts who could offer potentially crucial insight.

As forensic analysis and presentation methods evolve, examiners need to incorporate these methods into their digital forensics software tool. Any solution which aims to have a reasonable lifetime expectancy must be flexible enough to accommodate add-ons and modifications without undue strain. This also broadens its potential usefulness in examination tasks that may have been inconceivable at its deployment. Furthermore, it would allow for the analysis of an arbitrary number of supported file systems, essentially consolidating into one what used to require multiple tools or versions of software to accomplish.

1.3 Related Work

Much research has been conducted which relates directly to the work presented in this thesis. Here we categorize the works we have reviewed by those related to digital forensics and storage techniques.

1.3.1 Forensics

Digital forensics is a complex discipline, with many different areas of interest to researchers. Three of these areas include general approaches to difficult forensics tasks, using distributed processing to improve forensics analysis performance, and creating standards for the digital forensics science.

General Digital Forensics

Two challenging types of evidence that forensics examiners need to be able to analyze at times are Redundant Array of Independent Disks (RAID) storage systems and drives protected with encryption. In [47], Urias and Liebrock attempted to use a parallel analysis system on RAID storage systems, and documented the issues and challenges they faced with that approach. Similarly, multiple methods of properly handling the challenges presented by encrypted drives have been presented by Casey and Stellatos in [15] and by Altheide et al. in [13].

Distributed Processing for Forensics

With distributed processing in use so much today and in so many distinct settings, it is natural to think of using it to divide the workload of digital forensics processing. Several years ago, when the use of distributed processing was not yet as common as it is today, Roussev and Richard proposed a method for moving away from single workstation processing for forensic examination to a distributed environment [42]. A few years later, Liebrock et al. proposed improvements upon Roussev and Richard's system in [32], which introduced a decoupled front-end to a parallel analysis machine.

In [43], Scanlon and Kechadi introduced a method for remotely acquiring forensic copies of suspect evidence which transfers the contents of a drive over a secure Internet connection to a central evidence server. While this effort is a step for the better in terms of making evidence centrally accessible, it is difficult to see the direct utility of such an approach without accompanying software or analysis techniques to take advantage of storing the evidence on a server. Furthermore, the presented approach relies on either using the suspect's Internet connection to upload the image or images, or the use of a mobile broadband connection. Given the relatively abysmal upload speeds for current mobile broadband when dealing with data sets that are hundreds of gigabytes or even a few terabytes large, this approach will continue to be prohibitively inadequate.

Forensics Standardizations

Garfinkel has made great efforts to create standards to improve the overall digital forensic examination process. Garfinkel et al. presented the details of a forensic corpora in [21] with the purpose of giving researchers a systematic way to measure and test their tools. Garfinkel took this a step further in [23] with his work to represent file system metadata with XML. Finally, in [22] Garfinkel put forth a challenge to researchers and developers everywhere to take note of the current industry trends and take them head on with innovative forensic solutions that match the properties of emerging technologies.

1.3.2 Storage

Since our realization of our framework is built upon a cloud, we also consider work done by researchers to address some of the issues related to shared storage in a cloud. Du et al. proposed an availability prediction scheme for sharable objects, such as data files or software components, for multi-tenanted systems in [17]. In [48], Wang et al. introduced a middleware solution to improve shared IO performance with Amazon Web Services [1]. Increasing the security of the data stored in a cloud has been improved upon by Liu et al. in [33] and by Zhao et al. in [50].

Chapter 2

CUFF: A GENERIC FRAMEWORK

Based on the features and requirements necessary to achieve collaborative forensics as enumerated in Chapter 1, this chapter describes our framework, called the Collaborative Forensic Framework (CUFF), and elaborates what mechanisms are needed to facilitate these features [34]. Our purpose in doing this is to set a standard for the necessary components of any implementation of the framework, regardless of the tools or technologies available at the time of implementation.

2.1 Objectives of the Framework

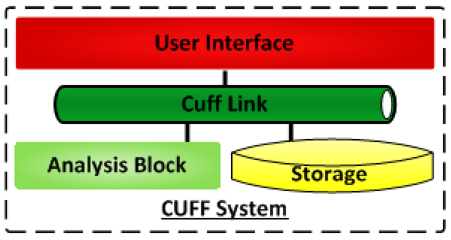
Our framework consists of four core components that (i) mediate communication between components in the system, (ii) coordinate the distributed analysis processing, (iii) maintain the shared storage space, and (iv) provide a basic interface to the system for the user interface. Figure 2.1 gives an overview of how these components relate to each other. While a precise set of application programming interfaces (APIs) for these four components may vary for the deployment setting, they should always fulfill specific foundational operations and always have the same basic interactions with the other components. We now discuss these two objectives in context of each component.

2.2 Analysis Block

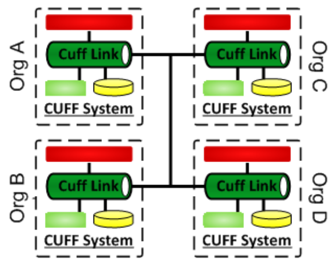
The Analysis Block is the workhorse of the system, and all other components are simply in place to either provide an interface to it, or to facilitate its proper function. The Analysis Block includes an Analysis Block Controller (ABC) as well as all processing resources, divided into independently functional units to which we refer by the term "nodes."

2.2.1 Analysis Block Controller

The ABC performs two critical roles. First, it manages a queue of analysis requests, or "jobs." Jobs can be initiated by either the user or the system. An example scenario of



(a) The Cuff Link provides the means for inter-system communication.



(b) Multiple deployments of CUFF can communicate with each other through the Cuff Link.

Figure 2.1: Overview of CUFF.

when the system might create an analysis job would be when a new evidence image is uploaded to the Storage component, it may be marked as needing some general analyses performed on it that is common to all evidence in the system. An example scenario of when a user would create an analysis job would be when they are browsing the contents of the evidence and want to have a specific analysis performed on a single file, a group of files, or all files of a given type, e.g. picture files in JPEG format.

When assigning processing resources a job, the ABC would ideally take into account both the required computational resources of the job (which can be computed internally) as well as the criticality level of the case with which the job is associated (which would need to be entered by the user). For example, if a high-profile case is opened and input to the system, or if a data set requires analysis by an algorithm that takes a particularly long time to complete, the jobs associated with these tasks would be placed closer to the front of the queue. However, the ABC must be flexible enough to be able to revise its initial assessment of a job's placement if, for example, a user demotes the level of criticality of a job based on evolving circumstances regarding the case.

Maintaining the queue of jobs and prioritizing their assignment will be a balancing act between the above two strategies. From the perspective of the user, it may be more favorable to prioritize based on the criticality of the case of which the evidence is a part. However, from the perspective of the overall system performance, it may be more favorable to prioritize based on the load a given request or set of requests will place on the system's computational resources. Of course, we do not claim that these are the only two approaches to prioritizing jobs in the system. Hence, other approaches may be more suited to accommodating both perspectives or accommodating an entirely different set of criterion that turn out to be more important.

The second role of the ABC is to balance the processing resources with the distinct characteristics of the queued jobs. By this we imply two things. First, that analysis requests will consist of a specified algorithm and a file or group of files on which the algorithm should be executed. Second, that there may be multiple types of analysis nodes, each of which provides a distinct set of analysis algorithms that are compatible with the platform of the node and the other algorithms or tools installed on it.

Because there will often be a disproportion between the ratio of the requested algorithms and the ratio of algorithms made available by the instances currently running, the ABC compensates for such a situation by shutting down a number of running instances to reclaim resources, and then instantiating new instances of a different node type which better fit the current "needs" of the system. If the analysis nodes are all heterogeneous, there will be no need for this feature of the ABC.

2.2.2 Analysis Nodes

Depending on the implementation of CUFF, the structure of the analysis nodes may employ any distributed processing approach. As such, configuring the analysis nodes introduces many of the same challenges as do other distributed processing models, such as finding an appropriate way to divide the problem into atomic sub-problems and then recombine the results into a comprehensible whole. The good news is that in digital forensics, the majority of analysis algorithms and techniques target either individual files or fragments of individual files, which provides a natural unit of distribution [42]. Furthermore, the results of such analyses can be stored such that they are associated with that file, reducing the effort necessary for recombination of results.

2.3 Storage

The Storage component keeps track of all acquired disk images, the analyses of their contents, comments and notes from users, and all other related information. To do this, it must accept incoming data streams of acquired disk images, and strictly maintain the integrity of the data through validation of the original checksums.

Additionally, there will inevitably be a need for each organization using a given implementation of CUFF to have databases for storing such information as case and evidence identifiers, user credentials, user aptitude data (e.g. what types of devices and operating systems a user has been certified to work on), the cases to which each user has been assigned, and progress information for the analysis of evidence, to name a few. Managing such information should also be the charge of the Storage component.

Requests will come at a high rate from the processing resources in the Analysis Block, which will be the dominant source of requests, so the Storage component's response time needs to be controlled. Data, such as analysis results, user comments, and communications between users, may need their own distinct class method for transferring to the Storage component, which can be inherited from an interface that gives the standard structure for all system transmissions.

In coordination with whatever access control mechanism is implemented, the Storage component also maintains strict confidentiality of the data it stores. The Storage component must also be flexible enough to allow temporary and/or limited access to case data for consulting professionals, allowing them to collaborate with those directly responsible for the case.

The Storage component should additionally use a standard method for uniformly representing the structure of acquired images of all types. Establishing or using a data representation standard will simplify the transmission of data segments between components in the system, as well as the transmission between distinct CUFF-enabled systems. The use of a standard also enables the storage component to verify the accuracy of the evidence representation.

2.4 User Interface

The user interface is the access portal to the entire system, which means that all features implemented in the system need to be coupled with the interface.

The first and most essential of forensic operations is the acquisition of disk images for their storage in the system. Considering the precision of hardware control necessary to perform acquisition on evidence, in some implementation environments it may be unreasonable to expect the framework to directly support this function. However, the user interface can easily expose a mechanism for uploading the images of the evidence once it has been acquired.

The user interface component is also responsible for providing the means for users to communicate and share data and information with each other. While a lot is implied in this requirement, the degree to which it is supported will determine how effective the users will be able to collaborate with each other. Since this is the core objective of CUFF, this element of the user interface demands priority attention.

For every distinct analysis algorithm used in the system, there will be an equally distinct result produced. While establishing a standard way for results to be presented would be useful and good, not all algorithms will be suited to giving output in a standardized form, particularly those algorithms that have creative ways of compiling information for human understanding. The user interface must be flexible for both cases.

2.5 Cuff Link

The Cuff Link is the most important component of CUFF. It plays several key roles which include mediating communication, validating input, managing the forensics examination flow, and exposing APIs for the other components in the system.

If CUFF were designed such that each of the components communicated directly with each other, it would have a relatively high level of coupling, resulting in greater difficulties in scaling, making modifications to code, and adding new elements to the system. By introducing the Cuff Link, we add a layer to the system which provides some degree of abstraction for components to communicate with each other. By only requiring each component to initially be able to communicate with the Cuff Link, the necessary implementation is simplified for all other components. The Cuff Link keeps track of all other components in the system to make this possible.

Because all messages and requests pass through the Cuff Link, it can ensure that requests are of a valid format before being passed on to the destination component, which will be either the Storage or Analysis Block component. Naturally, this validation must be done independently for the Storage and Analysis Block components to be tailored to their implementation and APIs.

A certain flow of events is associated with a digital forensics examination which is closely associated with the specific forensic tasks discussed in Section 3.6. First, when a case is opened, it must be evaluated for the types of evidence involved, which dictates the requisite skill set of the examiner that is to be assigned to the case. Second, an available examiner is assigned to the case. Third, the examiner proceeds to execute each of the standard forensic tasks, resulting in the extraction of items of interest from the evidence items. Fourth, the report is finalized and sent to the examiner's supervisor.

The events in this process can be easily detected because the onset of each step is so distinct from the preceding step. With the Cuff Link configured to detect these distinctions, it can create notifications to users in the system according to the current process phase for each case in which they have a particular role. For example, when a user logs on, the user interface can present a notification of the new cases to which they have been assigned. Also, when an examiner completes the report for a case they were working on, their supervisor can receive a notice of the case's completion.

Some of the basic APIs that must be exposed by the Cuff Link include the following:

- User login: The user will be required to login from the user interface. Their credentials should be forwarded to whatever authentication mechanism is employed in the implementation.
- Evidence transmission: Since all evidence images are to be stored in the Storage component, they will need to be transmitted across the system, typically being initialized from the user interface.
- Analysis request: Multiple requests for some piece of evidence to be analyzed by the Analysis Block will be made throughout the course of each case's examination. These will also typically be initialized from the user interface.
- Analysis results: While using the user interface, users will make many requests to see the results of the analyses of various pieces of evidence. The retrieval of this information and its display in a coherent manner to the user are critical to the usability of the system.

These operations are critical to the basic functionality of CUFF and are independent of the approach taken to implement the framework.

Chapter 3

REALIZATION AND IMPLEMENTATION

In this chapter we describe our efforts at taking CUFF from an abstraction to a usable implementation upon which mechanisms can be built to make the system ready for practical use. The User Interface, Cuff Link, and Analysis Block components are all deployed as cloud computing virtual machine (VM) instances. While several cloud architectures exist with many comparable features, we have decided to use Eucalyptus [39] for our implementation of CUFF. Eucalyptus has the benefits of open-source projects, while also maintaining a high level of compatibility with Amazon's commercial cloud computing offerings (EC2/S3).

3.1 User Interface

For our initial implementation of a CUFF system, we have used the Google Web Toolkit (GWT) and the Smart GWT set of widgets to create a simple web interface. Our initial prototype is shown in Figure 3.2. The web interface has the following key components which each provide a distinct functionality for the user: the navigation pane, the sub-navigation pane, the views pane, and the comments pane. In addition, there is a menu bar at the top with options that do not directly contribute to the user's ability to analyze evidence, but are there for convenience in using the system. Once the user logs into the system and selects a case to work on, each component of the interface is populated with the appropriate information.

GWT makes it possible to separate the code for sections of a web interface into what are called composites which are programmed as separate Java classes. Each of the key components of the interface are their own composites that are then combined in a wrapper Java class that handles the layout of the components it contains. Using this approach, it is very simple to program multiple component layouts while reusing the underlying functional code.

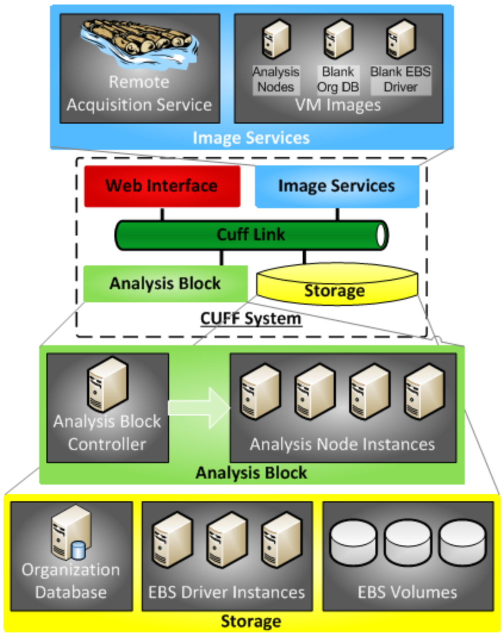
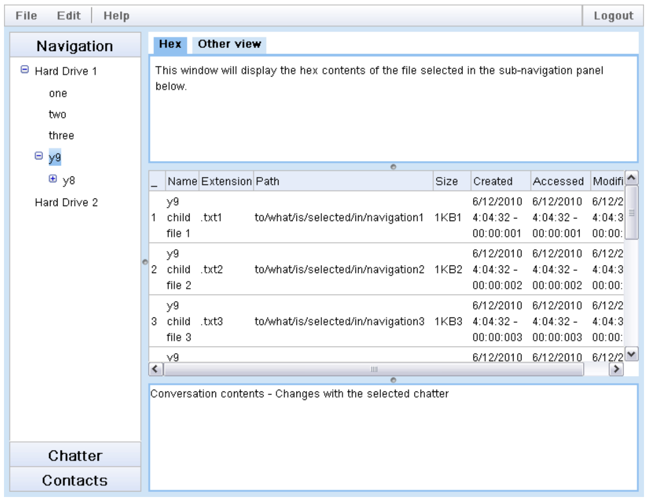


Figure 3.1: The combined major components of CUFF with their various subcomponents shown.



This window will display the hex contents of the file selected in the sub-navigation panel below.

_	Name	Extension	Path	Size	Created	Accessed	Modifi
1	y9 child file 1	.txt1	to/what/is/selected/in/navigation1	1KB1	6/12/2010 4:04:32 - 00:00:001	6/12/2010 4:04:32 - 00:00:001	6/12/2 4:04:3 00:00:
2	y9 child file 2	.txt2	to/what/is/selected/in/navigation2	1KB2	6/12/2010 4:04:32 - 00:00:002	6/12/2010 4:04:32 - 00:00:002	6/12/2 4:04:3 00:00:
3	y9 child file 3	.txt3	to/what/is/selected/in/navigation3	1KB3	6/12/2010 4:04:32 - 00:00:003	6/12/2010 4:04:32 - 00:00:003	6/12/2 4:04:3 00:00:
	y9				6/12/2010	6/12/2010	6/12/2

Conversation contents - Changes with the selected chatter

Figure 3.2: Simple web interface for CUFF.

3.1.1 Navigation

The navigation pane consists of the entire left side of the interface and includes an evidence navigation tool and list of contacts within the system. The evidence navigation tool has a drop down selection list of evidence items associated with the opened case. Upon selecting one of these evidence items, the web interface signals the web server

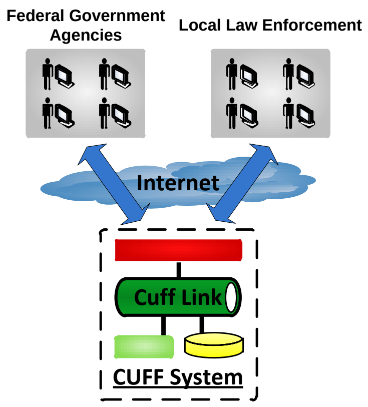


Figure 3.3: Multiple organizations can collaborate through the web interface of the implementation of CUFF.

to query the Storage server for the item's file listing. This is returned as a DFXML file (see Section 3.5.1) which is then parsed by the evidence tree widget, which has been programmed to understand the tags in the DFXML file and extract hierarchical information when present.

Also available from the evidence navigation tool's drop down menu is an option to "Add evidence item." Selecting this option displays the evidence upload dialog box, which is the main method of adding evidence to the system. In the dialog box are options for the evidence item, including:

1. Evidence item number
2. Location where evidence was obtained
3. Description of evidence
4. Vendor name
5. Model number or serial number
6. Recovery information, including the investigator's name that recovered it and the date and time it was taken into custody

Each of the above items is necessary for maintaining a proper chain of custody for evidence, and is a vital part of storing case information.

The list of contacts in the navigation pane is an essential part of the web interface, because it is the component that allows the user to give access to other users in the system to the items in a case. A user's contacts are separated by those that already have access to the currently open case and those that do not. By clicking an "add" button next to a contact without access, a request can be sent to the supervisor on file for granting the individual access to the case.

3.1.2 Sub-Navigation

The sub-navigation pane comprises of a grid widget that displays common metadata for the files in its list, including name, extension, path, size, and time stamps for its creation, last access, and last modification. Each of these fields is sortable using the advanced filter tool that is built-in to the Smart GWT grid widget.

If a directory is selected in the navigation pane, all of its non-directory children (i.e. files) are listed in the sub-navigation pane. When a file is selected in the sub-navigation pane, its contents are displayed in the currently selected view of the view pane. If a file is selected in the navigation pane, all of its siblings are listed in the sub-navigation pane and its contents are sent to the view pane.

3.1.3 Views

The views pane is where the user can actually see a particular file's contents. Multiple tabs at the top of the view pane separate the distinct viewing tools available for the selected file. These tabs will change depending on the type of the file, since different tools are appropriate for any given file type. For example, an image viewing tool will not be necessary for a text file containing only ASCII characters. The various viewing tools can be incorporated into the web interface by use of the FastCGI protocol. With this approach, customized visualization tools can be added on to the interface without much developer burden or performance cost. For implementation simplicity, the only view that is currently listed in the views pane is one that displays the contents of the file in hexadecimal.

3.1.4 Comments

The comments pane is the mechanism by which users can add their comments on a particular file, directory, or evidence image in a case. It consists of one portion that displays comments that have previously been logged by users and another for the user to add a new comment. The list of previous comments is a grid widget that lists the user that originated the comment, the date and time of the comment, and the first several words of the comment. The new comment section consists of a rich-text editor and buttons for clearing the text contents and for submitting the new comment.

3.2 Cuff Link

As stated in Chapter 2, the Cuff Link provides a couple of key functions for the overall system. It mediates communication, manages the forensics examination flow, validates input, and exposes an API for the other components. In order to accomplish these goals, the Cuff Link is not deployed in a single location, but acts as a layer of abstraction in multiple locations. This is illustrated in Figure 3.4 where elements of the Cuff Link are running on the actual servers where the Analysis Block component and Storage component are running.

The first key component of the Cuff Link layer is the implementation of a Representational State Transfer (REST) web service on both the Storage and Analysis Block components. Within the logic of these web services, we are able to perform the input validation and mediation necessary for other components to access the various resources available. Additionally, by adding the appropriate filters to the RESTful web service on the Storage component, we can manage the forensics examination flow. More details on this are given in the sections which present the details of the Storage

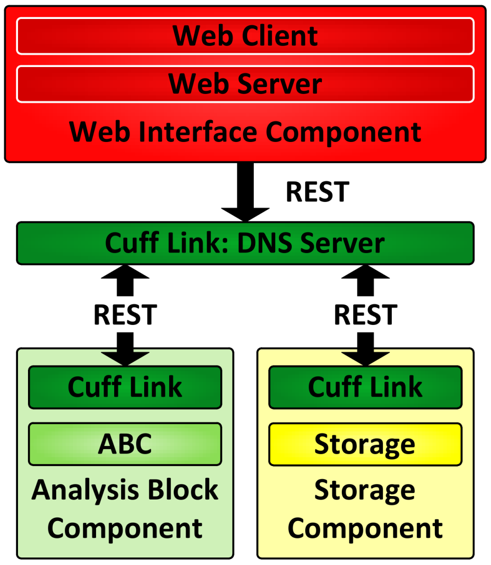


Figure 3.4: Inter-component communication through the Cuff Link, which is deployed in multiple locations to provide the layer of abstraction necessary for the various components.

(Section 3.3) and Analysis Block (Section 3.4) components.

The second key component of the Cuff Link layer is the Domain Name System (DNS) server. This central component makes it easy for other CUFF components to send traffic to a specific destination without knowing specific details about the destination. The component making a request only needs to specify the generic name for the destination server, such as "http://cuff.storage.example/<RESTful request>." The Cuff Link DNS server can then resolve the name to the appropriate server.

3.3 Storage

While the other components we have discussed in this chapter are implemented using the cloud infrastructure, the storage component in our realization is not. Cloud-based storage options are certainly viable for a production-ready implementation, but in order to simplify the proof-of-concept implementation while also demonstrating the flexibility of our architecture, we instead use a server that provides access to its local storage resources through the Cuff Link element's REST web API.

3.3.1 Coordinator

It is important to remember that while the details are given here for the mechanisms which provide the RESTful API to the Storage Coordinator, these details are actually a part of the Cuff Link layer.

The RESTful Storage Coordinator runs an Apache web server alongside the Django [5] web framework and the Django REST Framework [6]. Django offers fast and easy deployment through use of the Python language, while also being flexible in how HTTP requests are handled. This functionality is extended with the Django REST Framework, that allows for building "well-connected, self-describing RESTful Web APIs."

The Django REST Framework is extremely verbose in terms of allowing a developer to customize how the server responds to GET requests. This is accomplished by specifying what are called "URL configurations" that tell a dispatcher which Python module should handle a given request. For example, we specify in the URL configuration of the coordinator that all requests matching the following pattern should be sent to the `CaseListing` module:

```
urlpat = patterns(r'^listing/(?P<case>[-.\w]{1,64})), 'cuff_ds.
                              CaseListing')
```

The command above uses a regular expression to tell Django that for any GET requests that begin with `listing/` and then have a sequence of characters that follows, it should save the matching text and send it as the `case` parameter to the `get()` method of the `CaseListing` object in the `cuff_ds` package. This is an important detail, because it allows the Storage Controller to have incredible flexibility with the way it handles requests at run time, which is something that could not be accomplished by simply exposing resources in the Storage Space with the mechanisms native to Apache.

For example, a URL to the Storage Coordinator such as

```
http://cuff.storage.example/listing/2398-56-1-9125
```

appears to request the file “2398-56-1-9125” (which is a case number) in the folder “listing” which is located at the root of the web server. With the proper Django code we can interpret this as a request for the evidence listing for the case number 2398-56-1-9125. Knowing this we can retrieve the proper file or set of files from whatever location we have decided to use for such resources. Once again, this shows how CUFF provides a set of APIs that abstract the details of how a certain component functions so that other elements only have to be familiar with the established API, and nothing more.

3.3.2 Storage Space

In our implementation, the total volume of the resources in the Storage Space is small enough that we can host them on the same server as the Storage Coordinator. Files for all cases are in a common root directory, with folders for each case named with the case’s unique identifier. All evidence images and supplemental files are stored in the case’s folder, including evidence listings, analysis results, and reports. Although not directly implemented by us, distinct handlers for each type of the above data types can be exposed using the method described earlier.

Of course, this is not the only way the Storage Space can be implemented. Any number of methods for storing vast amounts of data can be used, including direct-attached storage (DAS), network-attached storage (NAS), storage area network (SAN), or any of the available cloud storage options popular today. With the way we have abstracted the access of resources with the Storage Controller, the choice of storage technology will not effect how the rest of the system functions. Furthermore, the internal hierarchy of files is likewise abstracted, making it possible to use any variation on this implementation detail as well.

3.4 Analysis Block

As stated in the framework specification in Chapter 2, there are two types of components that make up the Analysis Block, the Controller and the Nodes. Also, much like the Storage component described in Section 3.3, the Analysis block has a Cuff Link element which abstracts access. Here we present the details on each of these features of the Analysis Block.

3.4.1 Analysis Block Controller

Once again we emphasize that the details given here which pertain to the implementation of the REST web API for the Analysis Block Controller are actually part of the Cuff Link layer.

Just as was described in Section 3.3.1, the Analysis Block Controller runs an Apache web server with Django and the Django web REST framework. The URL configurations allow for the following types of GET requests:

- Analyze — Indicates an analysis request. The request should follow one of the following forms:
  - `http://cuff.analysis.example/analyze/<algorithm/tool ID>/<case ID>/<item ID>/file/<file ID>`
  - `http://cuff.analysis.example/analyze/<algorithm/tool ID>/<case ID>/<item ID>/bytes/<start sector>/<end sector>`
- Analyze Group — For use with analysis algorithms or tools that are designed to be used on multiple items *simultaneously*, which is very different from independently analyzing a set of items with the same algorithm or tool. The request should follow the following form:

  - `http://cuff.analysis.example/groupanalyze/<algorithm/tool ID>/<group ID>`

- Add to Group — Adds an item to a group which can afterward be analyzed together *simultaneously*. Group IDs are case-specific. If the group ID given does not already exist, it will be created. Groups are stored in the organization's database. The request should follow one of the following forms:

  - `http://cuff.analysis.example/groupadd/<case ID>/<group ID>/<item ID>/file/<file ID>`
  - `http://cuff.analysis.example/groupadd/<case ID>/<group ID>/<item ID>/bytes/<start sector>/<end sector>`

- List Group — Returns a list of the items (by their item ID and file ID or byte run information) that are currently stored in the specified group. If the group ID does not already exist, returns an empty list. The request should follow the following form:

  - `http://cuff.analysis.example/grouplist/<case ID>/<group ID>`

- Remove from Group — Removes an item from the specified group. No change is made if the group ID, file ID, or byte run information is invalid. The request should follow one of the following forms:

  - `http://cuff.analysis.example/groupremove/<case ID>/<group ID>/<item ID>/file/<file ID>`
  - `http://cuff.analysis.example/groupremove/<case ID>/<group ID>/<item ID>/bytes/<start sector>/<end sector>`

- Progress — Returns the status of an analysis request, which will be either "queued," indicating analysis on the item has not yet begun; "processing," indicating the item has been sent to an Analysis Node; "complete," indicating the analysis has terminated successfully; or “unknown,” indicating an erroneous request. The request should follow one of the following forms:

- `http://cuff.analysis.example/progress/<case ID>/<group ID>/<item ID>/file/<file ID>`
- `http://cuff.analysis.example/progress/<case ID>/<group ID>/<item ID>/bytes/<start sector>/<end sector>`

Each of the above URL configurations is set to call a Python module that handles the request in the manner specified. For those methods that need evidence item data, RESTful calls are made to the Storage component before execution. When the analysis of an item is completed, the results are sent to the Storage component. Hence, it was not necessary to list a request for the analysis results above, since such requests are not handled by the Analysis Block.

To demonstrate the feasibility of the Analysis Block’s motivating concepts, our implementation does not provide the prioritization or ratio balancing features described in Section 2.2.1. Since we simply needed the Analysis Block to perform some rudimentary analysis, the ABC itself is configured to calculate the checksum of a file and return the value. This demonstrates the theory behind the concept of having an Analysis Block fully developed with a distinct controller and set of analysis nodes, without the overwhelming task of implementing them both completely. Because of this, the information given on analysis nodes and node agents concern our design if we had been able to complete the implementation of all components in the system.

3.4.2 Analysis Nodes

The concept of the Analysis Node component is to take advantage of a cloud architecture’s infrastructure to distribute the load of analyzing portions of evidence across a large number of virtual machine instances that have analysis tools installed on them. Nodes are sent a small number of files to process with a specified algorithm or tool.

Once the analysis has been performed and the results have been obtained, they are sent to the Storage component. The method used to store the results depends on their format, but typically they will be saved as a file in the same location as all the other case data. A reference to the result is then inserted in the item's DFXML entry (see Section 3.5.1) so it can be retrieved easily afterward.

Since it is unreasonable to expect that all tools that examiners will wish to have available in CUFF will be compatible with each other, either because they are available for different operating systems or for some other reason, it will be necessary for there to be multiple VM images created to be Analysis Node instances. These will, of course, be stored by the cloud's Eucalyptus Walrus, and can be added, updated, or removed by the system's administrator using the commonly available euca-tools package that was built for performing such operations.

Node Agents

To make a generic way to interface to all analysis algorithms, inter-node communication and data transfer is handled by agents running on the analysis nodes. Employing this approach serves as a means to standardize communication protocols without forcing the protocols upon the programs that use them, making the system much more flexible with the types of analysis programs that can be run on it while maintaining the philosophy of detail abstraction used all throughout CUFF.

In addition to being uniformly programmed with a standard set of communication protocols, each distinct analysis node's agent will be customized to the analysis programs being hosted on that node type. This allows the agent to store whatever parameters or arguments necessary to interface with the programs as well as retrieve the analysis results. Because much of the implementation for these nodes will be the same for all node types, this improves the ability to support new file systems, operating systems, analysis algorithms, and so forth.

3.5 Improving Storage Efficiency

To aid in improving the efficiency and standardization of how CUFF stores and transmits data, we employ two data representation formats that are both extensions of the Extensible Markup Language (XML).

3.5.1 DFXML

As with all families of software, the tools available to digital forensic examiners over the years have been incompatible with each other, using proprietary formatting for image files and analysis data storage, with the exception of those open-source projects that have embraced open standards.

Garfinkel's Digital Forensics XML (DFXML) representation for file system metadata [23, 20] is an excellent beginning to giving researchers and developers a standard way of representing and accessing the contents of an imaged drive. The basic concept is to use an XML file to store the metadata of a disk image, including addresses and lengths of all "byte runs" (file fragments) on the disk. This XML file can then be used with the disk image for accessing specific files.

Garfinkel has developed a tool which produces this XML file and named it "fiwalk", short for "file and inode walk." The fiwalk tool depends on the Sleuth Kit (TSK) [11] for its interaction with the disk or image to be mapped, and is based on TSK's data structures. However, fiwalk does eliminate any confusion users may have about how to set certain options by acting as an abstraction layer in front of TSK.

There are four significant advantages to this approach. The first is tool and algorithm independence. With the wide availability of XML libraries for popular programming languages, virtually any tool can be converted to read this data format. The second major advantage is the abstraction of low-level storage details. Researchers are free to develop their analysis algorithms without needing to focus on such things as ensuring they have calculated their byte offsets properly according to the sector size of the current disk.

Third, distribution of a disk image's metadata facilitates sharing the image's contents since the DFXML file acts as an index, which is small in size compared to the entire disk image. To use an example from Garfinkel's work, researchers collaborating over the Internet can store both the disk images and their DFXML files in a password-protected directory accessible by all appropriate parties. To access a desired file across a network, a researcher needs only browse the DFXML files, request the byte runs associated with the file, and concatenate the results.

The fourth advantage is that the use of an extensible markup allows for the metadata to be added upon without limitation. This means that any relevant data can be marked up and inserted with its associated schema URI.

While still an emerging standard, DFXML plays a vital role in the transmission of file segments between CUFF system components. For our use in CUFF, DFXML plays a vital role in the transmission of file segments. Since one of the goals of the system is to distribute the processing workload quickly, files and file segments are referenced by their byte runs. This enables the analysis nodes to request only those runs that are to be analyzed in a very concise form, much in the same way that Garfinkel and his colleagues used it.

In our use of DFXML, we noticed one limitation to Garfinkel's approach. The fiwalk script currently only produces a simple Document Type Definition (DTD) specification for each DFXML document that is derived from a list of tags used during the document's creation and is inserted at the beginning of the document. As with any DTD specification, this does not allow for type validation. Of course this wouldn't be a problem if fiwalk was the only program to ever produce a DFXML document from an image, but this defeats the purpose of publishing an open standard for representing file system metadata.

To help encourage the adoption of DFXML as a standard, we have created an XML schema detailing tag hierarchy and complex data types. While this schema was based mostly on the current version of fiwalk at the time of writing (0.6.2), and hence some of the Sleuth Kit's data structures, it still allows for the markup to be extended, and care was taken to specify data types which are sensible for what they are meant to represent, irrespective of the type of device on which the data originally resided. Using this schema to validate an image's file system representation, any digital forensic tool can reliably use this standard in its interactions with the disk image.

3.5.2 EDRM XML

As a supplement to DFXML, we make use of the Electronic Discovery Reference Model (EDRM) XML [7], which is a robust and flexible load file format that aids in data exchange in forensic and e-discovery processes. As files are marked by examiners for bundling for the final report in CUFF, they are added to an EDRM XML file. Using this standard format not only helps compatibility between tools, but it also allows organizations to quickly change the format of their report template without effecting previously rendered reports, much like changing the CSS documents for a web site.

3.6 Forensic Tasks

The most important feature of our realization is that it accommodates the main tasks of any digital forensic investigation:

1. Acquisition: Examiners can use an upload tool from the common web interface while in the lab, and can afterwards access it through the system regardless of their physical location.

2. Validation: Any forensic tool must be able to guarantee the integrity of files from the earliest stages of the investigation to the final reporting. Our system fulfills this requirement through the use of hash values for every device image

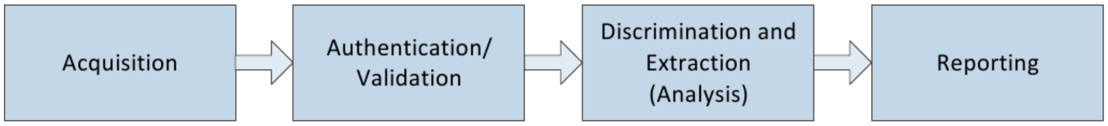

Acquisition
Authentication/ Validation
Discrimination and Extraction (Analysis)
Reporting

Figure 3.5: The four main tasks in the flow of a digital forensics examination.

and file before and after every data transmission. This is enforced by the Storage component and by the node agents.

3. Discrimination: Our system supports the use of discrimination and filter tools, which can use sets of hashes of known good files, such as the Reference Data Set provided by the National Institute of Standards and Technology (NIST) [10], to highlight those files which are unknown, effectively eliminating an extensive number of files the examiner needs to look at. This feature can be implemented as an analysis algorithm on a node image.

4. Extraction: As the task which is typically most demanding of examiners' time, we have focused on making the features which support extraction as robust as possible. This is principally supported by the web interface implementation.

5. Reporting: Comparable to contemporary tools, our system allows for users to generate reports of their findings with comments, as well as maintaining a strict log of activities any users performed on any of the evidence. This is likely best implemented as a program on a node image that can parse the case data, analysis results, and examiner logs and comments, and put them all into a standard template.

Two of the above tasks, acquisition and analysis, are of particular interest since they utilize many different components in the system. The sequence in which the different components are accessed and the messages that are exchanged between them are illustrated in Figures 3.6 and 3.7. We will briefly discuss them both.

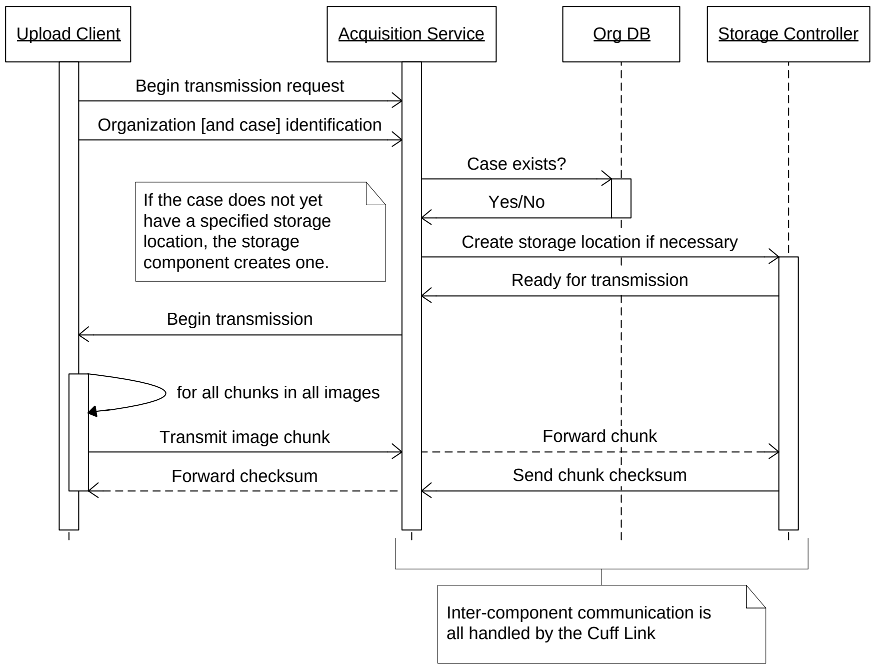

Upload Client
Acquisition Service
Org DB
Storage Controller
Begin transmission request
Organization [and case] identification
Case exists?
Yes/No
If the case does not yet have a specified storage location, the storage component creates one.
Create storage location if necessary
Ready for transmission
Begin transmission
for all chunks in all images
Transmit image chunk
Forward chunk
Forward checksum
Send chunk checksum
Inter-component communication is all handled by the Cuff Link

Figure 3.6: The sequence of events for acquiring an image with CUFF.

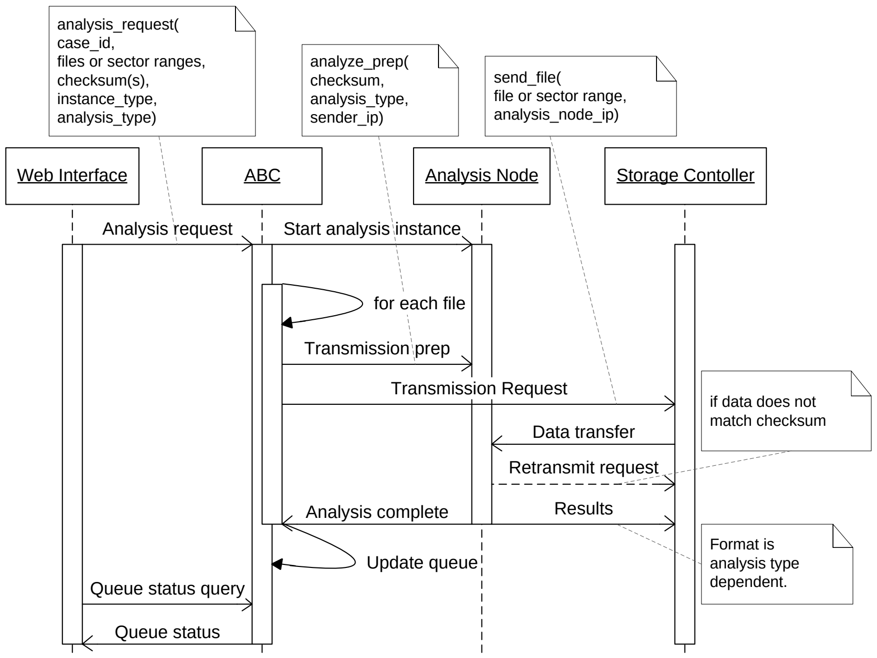

analysis_request(case_id, files or sector ranges, checksum(s), instance_type, analysis_type)
analyze_prep(checksum, analysis_type, sender_ip)
send_file(file or sector range, analysis_node_ip)
Web Interface
ABC
Analysis Node
Storage Contoller
Analysis request
Start analysis instance
for each file
Transmission prep
Transmission Request
if data does not match checksum
Data transfer
Retransmit request
Analysis complete
Results
Format is analysis type dependent.
Update queue
Queue status query
Queue status

Figure 3.7: The sequence of events for analyzing evidence in CUFF.

When acquiring evidence in CUFF, the user will first invoke the evidence upload tool (see Section 3.1.1), select the image file to be uploaded, and input all other information required to identify the case and evidence item. A class method on the web server then queries the database belonging to the user's organization to determine if the case already exists in the system and has a storage location created, and creates them if necessary. At this point, the system is ready to begin uploading the evidence image.

The upload tool breaks the image into manageable chunks for transmission, generates a checksum for each chunk, and uploads it to the storage location. The Storage Controller replies to the upload tool with the checksum of the received chunk, which tells the upload tool if the upload was successful or if the chunk was corrupted in transmission and needs to be uploaded again. This process is repeated until all portions of the image have been successfully transmitted and reassembled, and they generate the same checksum as the original image.

The process of analyzing evidence portions is typically initialized by the user, although the system may also mark newly uploaded evidence for general analysis as well. When the user submits an analysis request, the ABC determines if a running analysis node instance is suitable for handling the request, and starts a new instance if necessary. The ABC informs the node that it is to expect the incoming transmission of evidence for each of the files or byte runs in the batch to be analyzed. The ABC then requests that the data be sent from the Storage component to the node directly. The node and the Storage Controller communicate until the data is completely transmitted with its integrity validated against the stored checksums.

The node then performs the requested analysis on the data, notifies the ABC that the job is complete (indicating it is available to fulfill other analysis requests), and transmits the results to the Storage Controller to be added to the data associated with the case. When the user requests the status of the analysis request at this time, they will be informed of its completion. The results can then be viewed in the appropriate view tab in the web interface, which fetches them from the Storage component.

Chapter 4

CASE STUDY

The nature of our framework does make it difficult to interface with current forensic tools. But it is the nature of advancing to a new paradigm to have to leave behind entrenched practices that are no longer fitting for the circumstances. Thus, for a case study as to the usability of our framework, we abandoned the idea of attempting to construct a mechanism whereby we could embed a commercial product like FTK into CUFF. Such an effort would undoubtedly lead to such complex tasks as reverse-engineering the software to develop an artificial API for its native functionality. Work of that nature is simply outside the scope of this work.

Even though the approaches for handling distinct file systems in digital forensics are all specifically formulated to match the properties of that file system, the procedures for acquiring, validating, and analyzing all file systems is the same, as discussed in Section 3.6. Hence, in terms of CUFF's ability to facilitate this flow, all file systems are alike. Instead of choosing a more common file system as part of our case study, we picked a new and interesting file system that has had relatively little work done on it, which also demonstrates the flexibility of our system.

Because our case study works with a new file system, it was necessary for us to first implement an acquisition tool which can retrieve the data stored on the evidence. This acquisition tool provides the means to perform the first step in the examination process. The web interface can then be used to perform the rest of the examination steps. The details of this tool are given below.

While it is clear that acquisition is only the first step in a forensic examination and therefore distinct in both procedure and complexity from the rest of the examination process, the acquisition process uses all components of CUFF to get the evidence into the system, and is thereby a satisfactory beginning evaluation of CUFF's abilities.

4.1 YAFFS Acquisition Results

We now discuss our motivation for this particular case study, followed by the details of our developed approach and its limitations. Then in Section 4.2 we give an outline of how the process of evidence acquisition uses all the components of CUFF.

4.1.1 Motivation

The move towards the post-desktop model of pervasive computing is, well, pervasive. According to a recently released report from comScore on the U.S. mobile subscriber market share, "74.6 million people in the U.S. owned smartphones during the three months ending in April 2011, up 13 percent from the three-month period ending in January 2011" [41]. As a result of such a boom in the market, mobile computing has been involved and will continue to be involved in various crimes, fulfilling a variety of functions that include:

1. Storing information that is incidental to the committed crime, such as a suspect's contact list
2. Assisting in the commission of a crime, like fraud
3. Being the target of the crime, like identity theft

With relation to mobile smartphone platforms, Google has been the leader for some time, and currently 36.4% of the total market share is running its Linux-based Android operating system. We can only hypothesize that statistics for Android devices collected in evidence seizures would be very similar.

In the field of digital forensics, Android presents an intriguing set of challenges. First, a common storage option of choice for device manufacturers is NAND flash memory because of its significant cost advantages. While this isn't a direct feature of

Android, it is a necessary item to consider given the substantial differences relating to how NAND flash stores data internally [9].

Second, all versions of Android up to 2.3 use Yet Another Flash File System (YAFFS) [35] as the default file system [40]. YAFFS was created specifically to take advantage of the unique features of NAND flash memory, and hence is unique in how it stores and keeps track of files in the system. The majority of forensic tools were not designed to work with these differences, so new approaches are needed to keep pace with and anticipate usage trends.

These circumstances have afforded us the opportunity to take part in some of the first efforts to develop a method for retrieving data off a YAFFS-based mobile device that is systematic, repeatable, and complies with the rules of evidence to the extent possible.

4.1.2 Current Method

We have developed a Python script to acquire the data from the internal memory on Android-based devices that use YAFFS, which we call AndGrab. We recognize that even though YAFFS is currently the default file system for Android, it is not used on all current devices, e.g. the Motorola Droid X. Furthermore, Google has announced that, starting with devices that ship with Android 2.3 "Gingerbread," YAFFS will be replaced as the default file system by ext4 [14]. Despite this coming change, our method should continue to be applicable with very few changes.

Preparations: The first requirement of our acquisition method is to gain root privileges on the device and to have Busybox installed. Unfortunately, this does alter the device from the state in which it was when originally collected. However, when new data is stored in YAFFS2[1], it is always written to chunks that were previously erased (all

[1]YAFFS2 is the second version of the file system that was introduced in 2005. It differs from the original mostly in its adherence to the "write once" requirement of modern NAND flash. YAFFS2 is backward compatible with YAFFS1.

bits set to 1), in which case no *previously recoverable* data was overwritten. The only time writing data on the device should ever cause recoverable data to be overwritten is if the write causes garbage collection to be triggered.

Furthermore, if the suspect had intentionally altered the device to allow for hiding sensitive data, the device would likely already be rooted, resulting in even less evidence contamination.

The only remaining requirements of our method are for the acquiring workstation to have the Android Debug Bridge (ADB) and Python 2.6+ installed. AndGrab has been written to use ADB's connection with a connected device. Currently, all commands to the device are sent through ADB and the results are piped back into the script.

Acquisition Method: The first thing AndGrab does is check for the settings necessary for it to function properly, including the path to ADB and the checksum algorithms to be used during the verification operations. If it cannot determine these settings, the user is prompted to input them manually and they are stored for future executions.

As AndGrab proceeds, it searches for any connected Android devices. If more than one device is connected, the user is asked to select which should be used. It then executes commands to determine the mount locations of the storage devices using the YAFFS file system and begins executing `dd` commands. As mentioned earlier, the output of all the commands that AndGrab executes are piped from ADB's `stdout` back into the script. In practice, AndGrab takes about 12.5 minutes to run on the original Motorola Droid, resulting in images, information, and checksums for 4 partitions totaling 496.2 MiB.

Another direction we are working on with this project is making it possible to generate a DFXML representation of a YAFFS file system. Because of the inherent differences in how data is stored on NAND flash memory (which is used by all Android

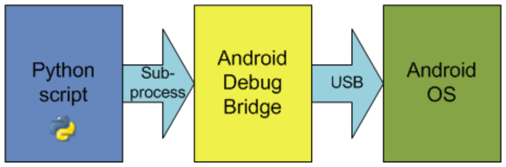


Figure 4.1: In AndGrab, the script invokes ADB as a subprocess, which then connects to and communicates with the device via USB.

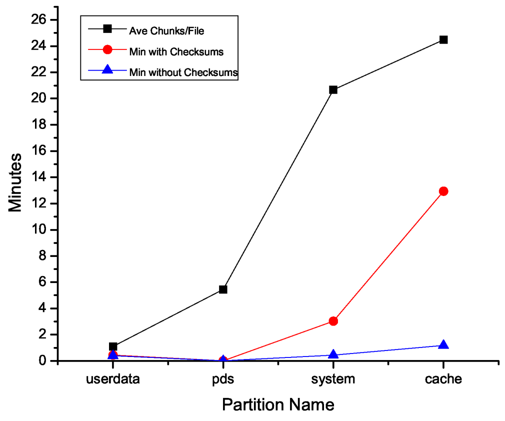


Figure 4.2: Run time of yafCrawl in relation to the average number of chunks per file object.

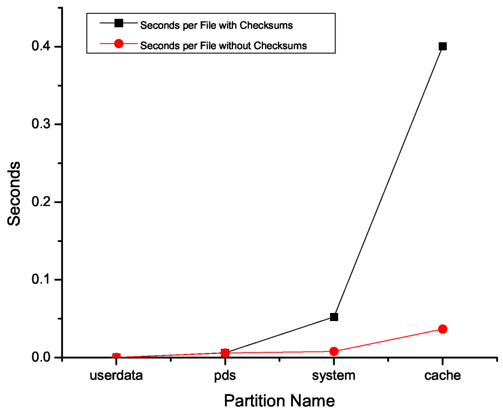


Figure 4.3: Comparison of yafCrawl's run time per file when calculating checksums and when not.

devices), no disk imaging tool can properly handle full disk dumps from the Android OS. As such, fiwalk cannot generate a representation of the disk image acquired by our script. However, we have written another script, which we call "yafCrawl," which gives partial information about the system in a DFXML format by identifying those chunks which do not have any information, i.e. are blank.

Performance results are shown in Figures 4.2 and 4.3, and illustrate that as the average size of files in the image increase, the run time also increases. The 'userdata' partition, which is usually 261 MB on a Motorola Droid, took an average of 28.6 seconds to process.

The reason the information "yafCrawl" generates is incomplete is due to the difficulty in extracting the file system metadata. In other file systems, this is relatively straightforward, and is easily integrated into the sparse acquisition process. By contrast, YAFFS stores this metadata in the out-of-bounds area in the NAND memory, which is not typically accessible through the operating system.

4.1.3 Limitations of Approach

Carriage Returns: As we experimented with the first iteration of our script, we found that the checksums of the original data on the phone and that of the acquired bit stream didn't match. Upon further investigation, we discovered that a single carriage return character (0x0D or '\r') was inserted before any occurrence of the hex sequence 0x0A, which translates to the newline character or '\n' in ASCII.

Naturally we supposed this to simply be a consequence of piping output from a Linux-based device to a Windows-based system. So to try to eliminate the corruption, we used the script in Linux, but the problem persisted. This suggests the corruption may be due to the libraries ADB uses. We were unable to reach a definite conclusion on this matter, but we modified our script to remove these inserted values, which then results in the images generating the same checksums as the original volumes.

Lack of OOB Data: As we discussed earlier, much of the forensically essential information about the disk structure of a YAFFS system is stored in the out-of-bounds (OOB) area of the NAND memory. Current methods of extracting this information rely on using the SD card to dump the information. We are in the early stages of investigating a promising possibility which will not require the use of the SD card but will still allow the extraction of this data.

Corruption of "userdata" Partition: While performing various tests on our script, we noticed that one side effect of executing the necessary commands on the phone while it is live has been the corruption of several chunks of data in the "userdata" partition. None of the other partitions failed the validation test after the above mentioned sanitization process. We were able locate the differing chunks between two sequential acquisitions, but were still not able to determine the root cause.

To put our approach in a different light, one would not think to try and acquire a forensically sound bit-stream copy of a hard disk using software that had to first be installed on the hard drive that was to be acquired. Such a practice would not work in a courtroom because of the obvious corruption of the original data that resided on the drive. However, with a traditional hard drive, it is possible to physically remove the drive and isolate it in such a way that data can only be read from the drive and not written.

Mobile devices do not provide such a convenient way to access the data. The NAND flash chips are mounted to the circuit board and accessed via a Flash Translation Layer (FTL) which may or may not communicate accurate information about which blocks of space are actually occupied by retrievable data since its job is to abstract these sort of details from whatever is accessing the memory. Furthermore, FTL algorithms are vendor-specific and may or may not provide an adequate API for exercising the necessary level of control needed during the acquisition phase.

To further complicate things, many vendors such as Motorola traditionally lock out users from the boot loader, which would allow access to the device memory without starting up the operating system. Even if the boot loader of a device can be used for this purpose, the FTL challenge described above may still prevent an exact copy of the memory to be made, even if a copy can be verified by a subsequent acquisition that yields the same checksum values.

Ultimately, it may only be possible for these issues to be solved by device manufacturers coming together to establish a verifiable method of creating forensically sound bit-stream copies of internal flash memory, and then make this information public.

4.2 Acquisition Flow in CUFF

It is important to make clear how this case study as a whole demonstrates the flexibility and utility of CUFF to facilitate the examination of a new piece of technology such as a YAFFS file system partition. To do this, we now describe the flow of events that occur during the acquisition process, clearly identifying the roles played by the distinct components of CUFF. It is also important to note that the flow of events described here are not peculiar to YAFFS, except for the steps necessary to extract the data off the device.

Figure 4.4 gives an illustration of the seven steps taken when using AndGrab with CUFF. The black lines connecting the different elements of the illustration are meant to indicate the communication channels through which data is sent from one element to another. The connection between the workstation and the Android device uses a USB cable as the medium and ADB as the protocol. The connection between the workstation and the web interface is internal, since the web interface will be displayed to the examiner on the workstation's monitor. Connections between the remaining elements in the figure were previously described in Chapter 3.

Intuitively, the various components of CUFF fulfill their prescribed functions during the acquisition process. The web interface gives the examiner access to the components of CUFF which pertain to the current task. During acquisition all components of CUFF are utilized and their capabilities are abstracted through the exposed upload tool. The Cuff Link serves to intermediate the transmission of the acquired images and to initiate the generation of a DFXML file for the evidence. The Storage component verifies the integrity of the evidence using the checksum generated from the acquisition tool and of course stores the images. The Analysis Block helps complete the acquisition process by generating a DFXML file per the analysis request entered by the Cuff Link. Once the DFXML file is available, the evidence can be browsed in the web interface and further analyses of files and byte runs can commence upon requests entered by the user.

The process of acquisition follows the following steps as enumerated in Figure 4.4:

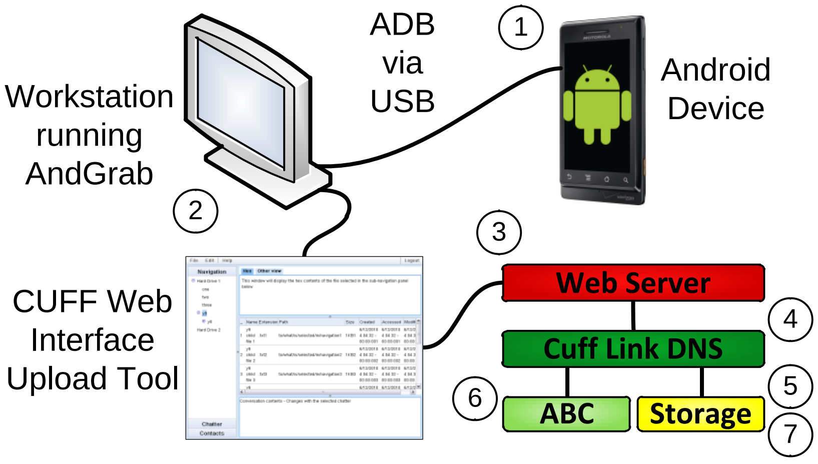
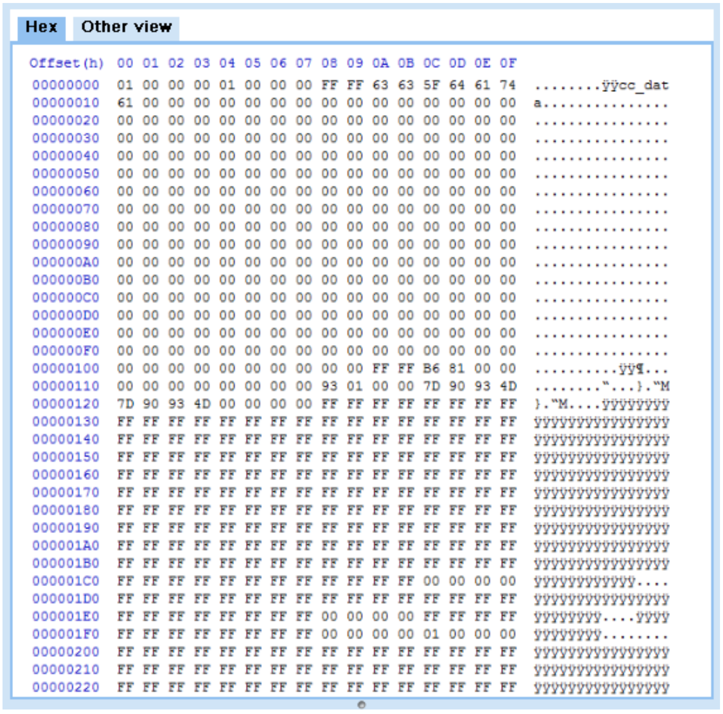


Figure 4.4: The seven steps of acquiring the data from an Android device.

1. The workstation runs AndGrab on the device using ADB over a USB connection.
2. The examiner uses the Upload Tool in the web interface to upload the acquired partition images.
3. The images are cached on the web server before transmission through the system.
4. The web server queries the Cuff Link DNS Server for `cuff.storage.example`, to which it replies with the IP address of the Storage component.
5. The web server transmits the acquired partition images to the Storage component. If the seized device is part of an existing case, then information about the device is added to the existing case data. Otherwise, a new case is created to store all the information.
6. The Cuff Link recognizes the process of uploading a new evidence item and queues the acquired device partition images for processing by the Analysis Block to generate a DFXML file for each of the images.
7. Once the Analysis Block has completed generating the DFXML file for each image, it is transfered to the Storage component to be stored along with the disk image it represents.



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  01 00 00 00 01 00 00 00 FF FF 63 63 5F 64 61 74  ........ÿÿcc_dat
00000010  61 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  a...............
00000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000100  00 00 00 00 00 00 00 00 00 00 FF FF B6 81 00 00  ..........ÿÿ¶...
00000110  00 00 00 00 00 00 00 00 93 01 00 00 7D 90 93 4D  ........"...}."M
00000120  7D 90 93 4D 00 00 00 00 FF FF FF FF FF FF FF FF  }."M....ÿÿÿÿÿÿÿÿ
00000130  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000140  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000150  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000160  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000170  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000180  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000190  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000001A0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000001B0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000001C0  FF FF FF FF FF FF FF FF FF FF FF FF 00 00 00 00  ÿÿÿÿÿÿÿÿÿÿÿÿ....
000001D0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000001E0  FF FF FF FF FF FF FF FF 00 00 00 00 FF FF FF FF  ÿÿÿÿÿÿÿÿ....ÿÿÿÿ
000001F0  FF FF FF FF FF FF FF FF 00 00 00 00 01 00 00 00  ÿÿÿÿÿÿÿÿ........
00000200  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000210  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000220  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

Figure 4.5: The views pane showing the hex values of the userdata partition from a Motorola Droid phone.

Once the acquisition process has completed, the examiner can then use the web interface to view the contents of the partitions of the device. For example, Figure 4.5 shows the very beginning of the userdata partition being displayed in the Hex tab of the views pane. The userdata partition is of particular interest to forensic examiners since it is the location of nearly all user-generated data.

Earlier we discussed many objectives of CUFF and the challenges it helps to resolve, including a shared storage solution for evidence, the ability for examiners to collaborate, management and automation of the forensics examination process, providing a generic interface to processing resources, and maintaining the integrity of evidence for adherence to the rules of evidence. Having now discussed the flow of events during acquisition and the roles played by each of the various components of CUFF, it is clear how CUFF continues to resolve these challenges with relation to the case study.

To overcome the challenge of a shared storage solution, the acquired YAFFS partitions are stored in the Storage component which is shared among examiners with permission to the case. Examiners may collaborate by adding comments relating to the files and data recovered from the phone, which CUFF keeps track of by using the DFXML file of the evidence item. By automatically sending the YAFFS partitions to the Analysis Block for the creation of the DFXML file, the Cuff Link manages the examination process and uses the generic analysis interface of the Analysis Block. Finally, the Storage component maintains the integrity of the evidence by verifying checksums of the acquired partitions.

4.3 Lessons Learned

Having used CUFF to perform this case study we were able to make some observations about the framework. First, because of the relatively shallow control that the web interface can exercise over a workstation's hardware, it may be impossible to rely solely on CUFF to perform the task of acquisition. So-called "traditional" forensic acquisition techniques have proven themselves very capable in this regard, and various equipment has been created for the purpose of protecting evidence integrity during acquisition. In light of this, CUFF should be looked upon as a compliment to currently established techniques with respect to acquisition. Of course, this does not take away from the benefits of collaboration offered by the framework during the other phases of an examination.

Second, because there are currently no tools for analyzing the evidence acquired from a YAFFS-based Android device that can be integrated into CUFF, it is difficult to advance an examination through the next phases of the forensics process. Of course this is not a shortcoming that is specific to the case study because the case study itself was the first effort put forth to implement a tool with the specific purpose of integrating it into CUFF.

Two interesting side-effects of the Android operating system being open-source and free for manufacturers to use on devices have been the staggering number of distinct devices that use the operating system [2] as well as substantial fragmentation in the versions of Android installed on devices. Potentially, each distinct device and OS version combination could use slightly different internal database structures for storing user data, such as contacts, text messages, email, et cetera. This leads to a need for analysis tools to be written to support the database structure of a specific set of devices, potentially necessitating the implementation of several versions of the same analysis tool.

This brings us to the final limitation of CUFF identified by our case study. CUFF currently does not have an intuitive method of organizing the tools made available for a specific type of evidence. If, for example, an examiner were to attempt examining an Android device that uses a more obscure database structure, it may be difficult for the examiner to find the correct analysis tools which match that particular device if it was embedded within a long list of similar analysis tools. It would be quite advantageous if an intuitive classification mechanism were devised to help separate analysis tools by their purpose and application to different evidence types. Ideally, part of the work of separation would be performed by CUFF as it determines or is told by the examiner exactly what type of device is being worked on, and excluding those tools which are incompatible.

Despite these slight limitations to our framework, we are satisfied that CUFF was able to perform as expected in this case study with handling an arbitrary file system and demonstrating that each of the components of CUFF do provide the services they were designed to provide.

Chapter 5

CONCLUSION

5.1 Summary

As we discussed in Chapter 1, the desktop-model architecture common to current forensics software is incapable of providing a means for examiners to (i) collaborate efficiently with each other, or (ii) perform advanced analyses on large-sized and diverse evidence. The reason for this deficiency is a combination of the limitations of available forensics software, a tremendous increase in evidence data size, seemingly monotonically increasing workloads for examiners, and a diverse set of device types that can hold potentially critical information for the investigation.

Furthermore, the software environment in which examiners now perform their work makes it prohibitively difficult to collaborate with other experts on a wide-scale basis. We believe that collaboration is the key to improving the digital forensics process such that examiners can bring criminals to justice, and that a fundamentally different forensics software approach is needed to facilitate such collaboration.

5.1.1 Contributions

The work that has been presented here addresses the essential requirements set forth in Section 1.2 for a framework suitable to facilitate collaborative forensics. CUFF maintains the integrity of the data stored in it by verifying checksums of all transmitted content before and after being sent, and by storing all checksums to ensure integrity continues to be perpetually maintained. While sophisticated communication mechanisms were not developed, examiners may record their thoughts and insights on evidence items such that they are easily accessible to their colleagues working on the same case. Furthermore, the flexibility of the web interface allows for additional communication techniques to be integrated into the system.

CUFF facilitates the sharing of all resources by allowing users to give other users access to a case and all items connected thereto. It is also flexible enough that developers and system administrators can add analysis tools to new or existing analysis node images, which images can be of any platform as long as the Analysis Node Agent has been properly configured. By adding tools to the system in this way, developers share access to their efforts to broaden the toolset available within CUFF. Because our implementation is built on a cloud, the processing resources used by these tools are highly scalable, while also taking advantage of distributed processing techniques. Accessing these analysis resources has been designed to be very generic, supporting a REST web API through the Cuff Link elements.

As it goes with many software frameworks, the utility of CUFF in a practical setting will depend a great deal upon the availability and sophistication of analysis tools that can work within the framework. One advantage to CUFF that helps mitigate this shortcoming is the fact that forensic examiners can upload the custom analysis tools they have developed and used over the years and begin to use them in an automated examination environment, which contributes an improvement to the efficiency of methods currently in use.

Standard data formats, specifically DFXML and EDRM XML, have been employed for the storage and reference of evidence items and their associated analysis results. The use of these standards helps to manage the shared storage space in the system by making data transmissions concise.

The web interface has been designed to accommodate multiple types of analysis results through the use of modular web components on the web client and through the support of FastCGI on the web server. And since our implementation is cloud-based and hence web-accessible, a new frontier of possibilities open up for examiner use-cases, including heavy-duty analysis of evidence initiated from a tablet or other form of web-enabled thin-client. Furthermore, the virtualization provided by cloud computing has additional exciting possibilities, such as allowing an examiner to:

1. Boot a disk image in a highly-controlled environment to perform certain analyses that can only be done on live instances

2. Use a large number of nodes to distribute the workload of cracking a disk protected by full-disk encryption

Through the results of our case study, we demonstrate that even a device and file system that have relatively immature forensics methods developed for them can be supported by the basic elements of CUFF. Although lacking in detail, a DFXML file can be generated from the acquired contents of an Android device that have been uploaded to CUFF, which can then be used to transmit its contents between the system's components, populate the navigation pane in the user interface, and reference analysis results.

The cumulative results of our work will not only aid law enforcement to combine their efforts with other departments and agencies, but can also be of great assistance to IT administrators when complying with e-discovery requests, performing internal investigations, and providing data recovery support.

5.2 Future Research

As we move forward with our research in this area, there are a number of things we wish to improve. To be truly useful, our framework will need to have more tools integrated into it that can be used in the analysis block. For existing tools, this will require the effort of interfacing the tool with the node agent that will handle sending commands to the tool and then sending back the results. Other useful tools can also be developed specifically for CUFF to take advantage of its unique features.

In a live deployment, the solution we used for the storage component will not be sufficiently responsive, nor will it have the necessary storage capacity. A much more advanced storage method will need to be employed that can stand up to the demands of tracking and storing countless disk images of hard drives, flash drives, GPS devices, cell phones, game consoles, and other related media capable of capturing data, plus the analysis results of all the stored evidence.

Our current implementation does not provide any means of access control because this was outside the scope of our work. However, like all systems today, CUFF will need a secure and verbose access control mechanism to keep its resources accessible to only authorized individuals. Since a great deal of quality research has been conducted in this area, we would refer any who wish to deploy a fully functional version of CUFF to other works that have made access control their focus.

Our approach for acquiring data from an Android device does not live up to the standards met by existing acquisition tools. Even so, the problems with our method arise from the inherent properties of the type of storage medium used in the device, which has been a major research challenge for multiple other researchers. We wish to continue our investigation of other possibilities for a non-destructive acquisition method for Android. In addition, we would like to begin researching how to build effective analysis tools for Android that can be used on the acquired data.

[1] Amazon web services. http://aws.amazon.com/.

[2] Comparison of android devices. http://en.wikipedia.org/wiki/List_of_Android_devices.

[3] Cost of hard drive storage space. http://ns1758.ca/winch/winchest.html.

[4] Digital forensics framework project home page. http://www.digital-forensic.org/.

[5] Django project home page. https://www.djangoproject.com/.

[6] Django rest framework project home page. http://django-rest-framework.org/.

[7] The electronic discovery reference model xml project. http://edrm.net/projects/xml.

[8] Forensic toolkit (ftk). http://accessdata.com.

[9] Memory technology device (mtd) subsystem for linux.

[10] National software reference library. http://www.nsrl.nist.gov/Downloads.htm.

[11] The sleuth kit (tsk) & autopsy: Open source digital investigation tools.

[12] Google says china disrupting e-mail service. BBC Online http://www.bbc.co.uk/news/business-12802914, March 2011.

[13] Cory Altheide, Claudio Merloni, and Stefano Zanero. A methodology for the repeatable forensic analysis of encrypted drives. In *EUROSEC '08: Proceedings of the 1st European Workshop on System Security*, pages 22–26, New York, NY, USA, 2008. ACM.

[14] Tim Bray. Saving data safely. Android Developers Blog: http://android-developers.blogspot.com/2010/12/saving-data-safely.html, December 2010.

[15] Eoghan Casey and Gerasimos J. Stellatos. The impact of full disk encryption on digital forensics. *SIGOPS Oper. Syst. Rev.*, 42(3):93–98, 2008.

[16] Robert N. Charette. More cyberattacks or just more media attention? IEEE Spectrum Online `http://spectrum.ieee.org/computing/networks/more-cyberattacks-or-just-more-media-attention`, July 2011.

[17] Juan Du, Xiaohui Gu, and Douglas S. Reeves. Highly available component sharing in large-scale multi-tenant cloud systems. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, pages 85–94, New York, NY, USA, 2010. ACM.

[18] Daniel Emery. Personal data stolen from uk developer codemasters. BBC Online `http://www.bbc.co.uk/news/technology-13731822`, June 2011.

[19] I. Foster and C. Kesselman. *The grid: blueprint for a new computing infrastructure*. The Morgan Kaufmann Series in Computer Architecture and Design Series. Elsevier, 2004.

[20] Simson Garfinkel. Aff and aff4: Where we are, where we are going, and why it matters to you. In *Sleuth Kit and Open Source Digital Forensics Conference*, 2010.

[21] Simson Garfinkel, Paul Farrell, Vassil Roussev, and George Dinolt. Bringing science to digital forensics with standardized forensic corpora. *Digital Investigation*, 6(Supplement 1):S2–S11, 2009. The Proceedings of the Ninth Annual DFRWS Conference.

[22] Simson L. Garfinkel. Digital forensics research: The next 10 years. *Digital Investigation*, 7(Supplement 1):S64 – S73, 2010. The Proceedings of the Tenth Annual DFRWS Conference.

[23] S.L. Garfinkel. Automating disk forensic processing with sleuthkit, xml and python. In *IEEE Systematic Approaches to Digital Forensics Engineering*, pages 73 –84, May 2009.

[24] Siobhan Gorman and Julian E. Barnes. Cyber combat: Act of war. Wall Street Journal Online `http://online.wsj.com/article/SB10001424052702304563104576355623135782718.html`, May 2011.

[25] Doug Gross. Cia, senate hackers gleefully promise more. CNN Online `http://www.cnn.com/2011/TECH/web/06/16/cia.hackers.lulzsec/index.html`, June 2011.

[26] Mark Hachman. Update: Leak exposes apple ipad emails, ids. PC Magazine Online http://blogs.pcmag.com/securitywatch/2010/06/alleged_leak_exposes_apple_ipa.php, June 2011.

[27] Kelly Jackson Higgins. Zeus attackers deploy honeypot against researchers, competitors. *DarkReading*, November 2010.

[28] Nathan Hodge and Ian Sherr. Lockheed martin hit by security breach. Wall Street Journal Online http://online.wsj.com/article/SB10001424052702303654804576350083016866022.html?mod=googlenews_wsj, May 2011.

[29] Guidance Software Inc. Encase forensic. http://www.guidancesoftware.com/.

[30] Cecilia Kang. Google: Hundreds of gmail accounts hacked, including some senior u.s. government officials. Washington Post Online http://www.washingtonpost.com/blogs/post-tech/post/google-hundreds-of-gmail-accounts-hacked-including-some-senior-us-government-officials/2011/06/01/AGgASgGH_blog.html, June 2011.

[31] Yoko Kubota. Sega says 1.3 million users affected by cyber attack. Reuters Onlinehttp://www.reuters.com/article/2011/06/19/us-sega-hackers-idUSL3E7HJ01520110619, June 2011.

[32] L. M. Liebrock, N. Marrero, D. P. Burton, R. Prine, E. Cornelius, M. Shakamuri, and V. Urias. A preliminary design for digital forensics analysis of terabyte size data sets. In *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, pages 190–191, New York, NY, USA, 2007. ACM.

[33] Qin Liu, Guojun Wang, and Jie Wu. Efficient sharing of secure cloud storage services. In *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, pages 922 –929, 292010-july1 2010.

[34] Mike Mabey and Gail-Joon Ahn. Towards collaborative forensics: Preliminary framework. In *Information Reuse and Integration (IRI), 2011 IEEE International Conference on*, 2011.

[35] Charles Manning. How yaffs works. http://www.yaffs.net/how-yaffs-works-internals, January 2010.

[36] R.A. Maxion and K.S. Killourhy. Keystroke biometrics with number-pad input. In *Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International Conference on*, pages 201 –210, 28 2010-july 1 2010.

[37] Joseph Menn. *Fatal System Error: The Hunt for the New Crime Lords Who are Bringing Down the Internet*. PublicAffairs, first edition, 2010.

[38] Joseph Menn. U.s. experts close in on google hackers. `http://www.cnn.com/2010/BUSINESS/02/21/google.hackers/index.html`, February 2010.

[39] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The eucalyptus open-source cloud-computing system. In *Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on*, pages 124 –131, May 2009.

[40] Ryan Paul. Ext4 filesystem hits android, no need to fear data loss. Ars Technica Online`http://arstechnica.com/open-source/news/2010/12/ext4-filesystem-hits-android-no-need-to-fear-data-loss.ars`, January 2011.

[41] PRNewswire. comscore reports april 2011 u.s. mobile subscriber market share. `http://www.prnewswire.com/news-releases/comscore-reports-april-2011-us-mobile-subscriber-market-share-123098853.html`, June 2011.

[42] Vassil Roussev and Golden G. Richard III. Breaking the performance wall: The case for distributed digital forensics. In *The Proceedings of the Fourth Annual DFRWS Conference*, 2004.

[43] Mark Scanlon and Mohand-Tahar Kechadi. *Online Acquisition of Digital Forensic Evidence*, volume Volume 31 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 122–131. Springer Berlin Heidelberg, 2009.

[44] Patrick Seybold. Update on playstation network and qriocity. PlayStation Blog `http://blog.us.playstation.com/2011/04/26/update-on-playstation-network-and-qriocity/`, April 2011.

[45] Ian Sherr. Hackers breach second sony service. Wall Street Journal Online `http://online.wsj.com/article/SB10001424052748704436004576299491191920416.html?mod=e2tw`, May 2011.

[46] CNN Wire Staff. Senate website under review after hacker gains access to server. CNN Online `http://www.cnn.com/2011/POLITICS/06/13/senate.website.hacked/index.html`.

[47] Vincent Urias, Curtis Hash, and Lorie M. Liebrock. Consideration of issues for parallel digital forensics of raid systems. *Journal of Digital Forensic Practice*, 2008.

[48] Jianzong Wang, Peter Varman, and Changsheng Xie. Middleware enabled data sharing on cloud storage services. In *Proceedings of the 5th International Workshop on Middleware for Service Oriented Computing*, MW4SOC '10, pages 33–38, New York, NY, USA, 2010. ACM.

[49] Jim Wolf. Lockheed says thwarted ”tenacious” cyber attack. Reuters Online `http://www.reuters.com/article/2011/05/29/us-usa-defense-lockheed-idUSTRE74S09220110529`, May 2011.

[50] Gansen Zhao, Chunming Rong, Jin Li, Feng Zhang, and Yong Tang. Trusted data sharing over untrusted cloud storage providers. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 97 –103, 302010-dec.3 2010.