

Forensic Methods and Tools for Web Environments

by

Michael Kent Mabey

A Dissertation Presented in Partial Fulfillment  
of the Requirements for the Degree  
Doctor of Philosophy

Approved November 2017 by the  
Graduate Supervisory Committee:

Gail-Joon Ahn, Co-Chair

Adam Doupé, Co-Chair

Stephen S. Yau

Joohyung Lee

Ziming Zhao

ARIZONA STATE UNIVERSITY

December 2017

## ABSTRACT

The Web is one of the most exciting and dynamic areas of development in today's technology. However, with such activity, innovation, and ubiquity have come a set of new challenges for digital forensic examiners, making their jobs even more difficult. For examiners to become as effective with evidence from the Web as they currently are with more traditional evidence, they need (1) methods that guide them to know how to approach this new type of evidence and (2) tools that accommodate web environments' unique characteristics.

In this dissertation, I present my research to alleviate the difficulties forensic examiners currently face with respect to evidence originating from web environments. First, I introduce a framework for web environment forensics, which elaborates on and addresses the key challenges examiners face and outlines a method for how to approach web-based evidence. Next, I describe my work to identify extensions installed on encrypted web thin clients using only a sound understanding of these systems' inner workings and the metadata of the encrypted files. Finally, I discuss my approach to reconstructing the timeline of events on encrypted web thin clients by using service provider APIs as a proxy for directly analyzing the device. In each of these research areas, I also introduce structured formats that I customized to accommodate the unique features of the evidence sources while also facilitating tool interoperability and information sharing.

To my children, for whom I hope to make it safer  
to explore the wonders of technology.

## ACKNOWLEDGMENTS

If I have succeeded at anything in life, it is because others have helped me pick myself back up when I no longer knew I still had strength to move forward.

Heidi, you are my best friend in the truest sense and still my greatest success story. With each passing year, I become even more grateful that I chose *you* to be my equal in everything and yet nothing, to be the person I go through hard times and happiness with, and to be the excellent mother to our beautiful children. I promise to never stop trying to deserve your love and loyalty.

To my family, my parents, siblings, and in-laws, you are the best family I could ever ask for. Thank you for creating an environment where I can let go and be myself, and for instilling in me the confidence to attempt hard things and constantly stretch my ambitions. Jenny, you are the best big sister ever! Thank you for always believing in me and helping me see through the mist of discouragement when it settled over my hopes.

Dr. Ahn, I look up to you and respect you tremendously. So much of who I have become over the last eight years is a direct result of your involvement in my life and your wisdom and guidance. Thank you for your unfailing optimism, your enormous aspirations, and most of all your friendship. I hope some day to be as daring, kind, and intelligent as you.

Dr. Doupé, I was a bit skeptical at first about having a professor that was two and a half years younger than me join and co-direct SEFCOM. Now I know it was one of the best things to happen to the lab. Personally, I owe so much of my research success to the help you so freely contributed. Thank you for all of your feedback, encouragement, and most of all for helping me to graduate on time!

Dr. Zhao, I hope that someday I can be as good of a researcher as you are. You have this amazing ability to pick out critical details and apply them in novel ways. I am proud

to call you my friend and to have spent so much time working with you. And thank you so much for the kindness you and Yiqi have always shown my family and me.

To the rest of my committee, Dr. Yau and Dr. Lee, thank you so much for your service and guidance. I feel honored to have learned from you and associated with you while here at ASU.

To my fellow SEFCOM lab members, thank you for being a little extension of my family for the last eight years. I have really enjoyed learning so much from all of you in our lab meetings and from talking about our research. And it is always a pleasure anytime we get to spend time outside the lab.

Finally, I would like to thank all the undergraduate and master's students that participated in my research: Justin Paglierani, Daniel Caruso, Justin Boots, Travis Seville, Samantha Juntiff, Chris Silvia, and Adric Rukkila. Your contributions definitely played a role in me being able to successfully complete my PhD.

My research was supported in part by grants from the U.S. Department of Defense Information Assurance Scholarship Program and the Center for Cybersecurity and Digital Forensics at Arizona State University. The information reported here does not reflect the position or the policy of the funding agency or project sponsor.

## TABLE OF CONTENTS

	Page
LIST OF TABLES .....	ix
LIST OF FIGURES .....	x
LIST OF CODE LISTINGS .....	xi
GLOSSARY OF DIGITAL FORENSICS TERMS .....	xii
CHAPTER	
1 INTRODUCTION .....	1
1.1 Digital Forensics .....	4
1.1.1 The Forensic Process .....	5
1.1.2 Rules of Evidence .....	6
1.2 Web Apps vs Traditional Programs .....	7
1.3 Web Thin Clients .....	8
1.4 Contributions of this Dissertation .....	9
2 BACKGROUND .....	12
2.1 Sharing Structured Data .....	12
2.2 Chrome OS: Key Features .....	13
2.2.1 Encryption .....	13
2.2.2 TPM and Secure Boot .....	15
2.2.3 Limited Local Storage .....	16
2.2.4 Extensions Enhance Functionality .....	16
2.2.5 Disk Layout .....	17
2.2.6 Anatomy of Extensions .....	18
2.2.7 Developer Mode .....	20
2.2.8 Users .....	21

CHAPTER	Page
2.3 Email Forensics: Preliminary Effort .....	22
2.3.1 Approach .....	23
2.3.2 Improving EFXML .....	31
2.3.3 Lessons Learned .....	32
3 CHALLENGES, OPPORTUNITIES, AND A FRAMEWORK FOR WEB ENVIRONMENT FORENSICS .....	34
3.1 Introduction .....	34
3.2 Motivating Scenario .....	36
3.3 Unique Forensic Challenges in Web Environments .....	37
3.3.0 C0: Rule of Completeness .....	37
3.3.1 C1: Associating a Suspect with Online Personas .....	38
3.3.2 C2: Evidence Access .....	39
3.3.3 C3: Relevant Context .....	40
3.3.4 C4: Tool Integration .....	41
3.4 A Framework for Web Environment Forensics .....	42
3.4.1 F1: Evidence Discovery and Acquisition .....	46
3.4.2 F2: Analysis Space Reduction .....	50
3.4.3 F3: Timeline Reconstruction .....	54
3.4.4 F4: Structured Formats .....	58
3.5 Related Work .....	62
3.6 Discussion .....	63
3.7 Summary .....	64
4 DBLING: IDENTIFYING EXTENSIONS INSTALLED ON EN- CRYPTED WEB THIN CLIENTS .....	66

CHAPTER	Page
4.1 Introduction .....	66
4.2 Background .....	69
4.3 dbling: Design and Implementation .....	71
4.3.1 Enrollment Phase .....	73
4.3.2 Identification Phase .....	77
4.3.3 Export Phase.....	83
4.4 Case Study .....	84
4.5 Limitations .....	87
4.6 Improving Accuracy .....	88
4.6.1 Image Conversion.....	89
4.6.2 eCryptfs Metadata .....	90
4.7 Related Work .....	92
4.8 Summary .....	94
<b>5 FORENSIC TIMELINE RECONSTRUCTION FOR ENTERPRISE</b>	
<b>WEB THIN CLIENTS .....</b>	<b>95</b>
5.1 Introduction .....	95
5.2 Background .....	99
5.2.1 Chrome OS System Logs .....	99
5.2.2 Self-Assignment of IPv6 Addresses .....	100
5.3 API Evaluation .....	102
5.3.1 G Suite API: Login Data .....	102
5.3.2 G Suite API: List of Enrolled Devices.....	103
5.3.3 G Suite API: Files and Metadata .....	104
5.3.4 Chrome API: User Info .....	106



CHAPTER	Page
5.3.5 Summary of Data .....	108
5.4 Beacon: Implementation .....	108
5.4.1 Collection and Retention .....	110
5.4.2 Caching .....	111
5.4.3 Evaluation .....	114
5.5 Timeline Reconstruction and Visualization.....	116
5.5.1 Reconstruction .....	116
5.5.2 Visualization .....	118
5.5.3 Providing Access to Law Enforcement .....	120
5.6 Discussion .....	121
5.7 Related Work .....	122
5.8 Summary .....	123
6 CONCLUSION .....	124
6.1 Future Work .....	127
REFERENCES .....	128
APPENDIX	
A EMAIL FORENSICS XML (EFXML) SCHEMA .....	141
B MATCHING EXTENSION RANKING LIST (MERL) SCHEMA .....	146
C OPEN-SOURCE PROJECTS .....	149

## LIST OF TABLES

Table	Page
1.1. Complaints Submitted to IC3. ....	2
2.1. List of Keys Used by Chrome OS to Encrypt User Data. ....	14
3.5. Alignment of the Framework Components with Challenges to Web Environ- ment Forensics. ....	43
4.1. U.S. business-To-Business (B2B) PC Category Growth from 2013 to 2014. ....	70
4.3. Statistics from Running the dbling Crawler on the Chrome Web Store. ....	74
4.5. Quick Reference for Symbols Used to Describe Dbling. ....	75
4.8. Statistics on Centroid Families. ....	82
4.11. Rankings for Matching Candidate Graphs to the Correct Extension. ....	86
5.8. Time Data Caching Rules. ....	112
5.9. Results of the Beacon Battery Use Test. ....	115

## LIST OF FIGURES

Figure	Page
1.2. The Four Steps of the Forensic Process.....	5
1.3. Comparison of Traditional Programs and Web Apps.....	7
2.2. Layout of Partitions on a Chrome OS Device. ....	18
2.5. The Traditional Forensic Workflow Combined with the Web Email Approach.	23
2.6. All the Cookies Created by Logging in to Gmail.....	26
3.1. Relation between Evidence Types Acquired during an Investigation. ....	35
3.2. Mallory's Two Personas. ....	36
3.3. Timeline of a Malicious User's Actions.....	41
3.4. My framework for Web Forensics. ....	42
4.2. The Three Phases of the dbling Framework. ....	72
4.4. More Statistics from the dbling Crawler and Case Study. ....	74
4.6. A 5D-FileTree Example and the 5D-Coordinates for Its Vertices. ....	77
4.7. Pruning the Full Directory Tree to a Few Candidate Graphs.....	79
5.2. Process for Self-Assigning an IPv6 Address. ....	101
5.3. The Four APIs Used and the Information They Provide. ....	102
5.7. Screenshot of the MAC and IPv6 Addresses of a Chrome OS Device. ....	107
5.10. Combining the Data from the Four APIs. ....	117
5.11. Visualization of Files Created by a User in Google Drive. ....	119

## LIST OF CODE LISTINGS

Listing	Page
2.3. Locations Where Extensions May Store User-Specific Data on Chrome OS...	20
2.4. Abbreviated Output of the Mount Command in Chrome OS. ....	22
2.7. A Sample EFXML File. ....	30
4.10. A Sample MERL File. ....	84
4.12. Example Usage of an EncryptedTempDirectory object. ....	92
5.1. Partial Output of the Mount Command in Chrome OS. ....	99
5.4. Sample Login Data Retrieved from the G Suite API. ....	103
5.5. List of Chrome OS Devices Enrolled with G Suite. ....	104
5.6. Metadata of a File from Google Drive. ....	105

## GLOSSARY OF DIGITAL FORENSICS TERMS

**Accuracy:** One of the Rules of Evidence (see Subsection 1.1.2). Accuracy requires the proper function of the processes used by examiners to acquire, process, and store the evidence [4]. It is vital to ensure that they make no alterations to the evidence being examined and produce consistent results, otherwise the evidence (or its forensic copy) may be considered by the court to be inauthentic [123] or at least not meeting the requirements to be admitted as a duplicate of the original evidence [121, 120].

**Admissibility:** One of the Rules of Evidence (see Subsection 1.1.2). When evidence is admissible, the actions taken to acquire, process, and store it were done in accordance with the laws and regulations of the applicable jurisdictions [122]. In the United States, this includes compliance with the Fourth Amendment, which protects individuals “against unreasonable searches and seizures” [89, p. 11]. Furthermore, if the evidence itself is “shown to be relevant, material, and competent, and is not barred by an exclusionary rule, it is admissible” [46].

**Artifact:** In digital forensics, practitioners use the term “artifact” to refer to “a [digital] object of ... interest” [47]. This could refer to any piece of data on a system or traversing a network, including those created by programs as part of their normal function as well as those authored by a human user.

**Authenticity:** One of the Rules of Evidence (see Subsection 1.1.2). The authenticity of evidence depends on whether it makes an explicit link to specific individuals and events. This is largely accomplished by providing sufficient supporting evidence “that the item is what the proponent claims it is” [123]. Means for establishing evidence’s authenticity include (but are not limited to) considering what access controls were in place, reviewing system logs, and evaluating any authentication

based on cryptography, e.g., public/private key authentication. In some cases, circumstantial evidence to help establish authorship [89, p. 159].

**Bitstream Copy:** See “Forensic Copy.”

**Chain of Custody:** Whenever any examiner does any work with a piece of evidence, they must log their actions in a ledger called a “chain of custody” form “to account for the integrity of each specimen ... by tracking its handling and storage from the point of specimen collection to final disposition of the specimen” [117, 116]. The form and fields of a chain of custody vary by jurisdiction, but the general accounting procedures are the same.

**Completeness:** One of the Rules of Evidence (see Subsection 1.1.2). Evidence must tell a complete narrative of a set of particular circumstances, setting the context for the events being examined so as to avoid “any confusion or wrongful impression” [118]. Under this rule, if an adverse party feels evidence lacks completeness, they may require introduction of additional evidence “to be considered contemporaneously with the [evidence] originally introduced” [118].

**Data at Rest:** Contemporary digital storage devices are generally divided into two main groups: those that require power to retain the information they store (volatile) and those that do not (non-volatile). The “rest” part of the term “data at rest” refers to the power state. Thus “data at rest” refers to data stored when the device is at a low (off) power state. Intuitively, there is typically no recoverable data at rest on volatile storage mediums.

**Disk Image:** See “Forensic Copy.”

**Drive-Specific File Type:** A file created in Google Docs, Sheets, Slides, Drawings, Maps, or Forms.

**Examiner:** An individual trained to conduct forensic analyses on digital evidence. In this work I also use the term “examiner” in an abstract sense, meaning that an examiner may be a person but it may also refer to a program that handles evidence.

**Exculpatory Evidence:** Evidence that suggests innocence [119].

**Fat Client:** A type of computer designed to operate independent of the processing and storage resources of other computers. Typically used in a context to contrast with the term “thin client.” See also “Thin Client.”

**Forensic Copy:** Also known as a *bitstream copy*. An exact duplicate of an original digital storage device, such as a hard drive [121, 48]. A forensic copy serves as a protection for the original evidence since the examiner works with it instead of the original, allowing them to perform analyses without the risk of compromising the integrity of the evidence. Methods for creating a forensic copy vary depending on the regulations of the jurisdiction. At times, practitioners will use what is called a “disk-to-disk” approach, in which they first obtain another storage device of the exact same make and model as the evidence, then copy every bit of data from the original device to the duplicate, thus making the copy exactly the same as the original in a physical and digital sense. In another common approach, known as “disk-to-image”, practitioners read the data stored on the device and save it to a file known as a *disk image*. Researchers have developed many formats of disk images that use compression to reduce the size of the *raw image* without compromising its integrity. See also *Integrity*.

**Inculpatory Evidence:** Evidence that suggests guilt [112].

**Integrity:** The nature and purpose of evidence is to tie an individual to an event of interest (see also *Authenticity*). The only way that examiners can definitively show this kind of connection is if the integrity of the evidence is preserved throughout the entire

process [116]. The reason for this is intuitive: if the evidence is tainted, changed, or in any way not in the same state or condition it was when it was collected, the narrative as told by the evidence changes as well. In particular, the person responsible for the state of the evidence may not be the suspect in custody. So, in a manner similar to the chain of custody, there must be some way of showing that the evidence has not changed, meaning its integrity remains intact. With digital evidence this typically means making a *forensic copy* of the original evidence, ensuring it is correct by taking cryptographic hashes of both, and then examining the copy instead of the original. But even when working with a copy of the original evidence, it is important that the tools used on the evidence not make any modifications when performing their analyses.

**Link-Local Address:** A network address that, according to RFC 4862 [85], has “link-only scope that can be used to reach neighboring nodes attached to the same link. All interfaces have a link-local unicast address.” See also “Unicast Address.”

**Live Acquisition:** One of two general *types* of acquisition. Live acquisitions are performed on systems in the process of running, often because “the computer has an encrypted drive” [89, p. 106] or to capture the contents of volatile memory. Live acquisitions are often contrasted with *static acquisitions*.

**Logical Acquisition:** One of two common *methods* of acquisition, often contrasted with sparse acquisitions. Logical acquisitions target “only specific files of interest” [89, p. 107].

**Lower Filesystem:** In a stacked encrypted filesystem, such as eCryptfs, the lower filesystem refers to the set of files that are encrypted and stored to non-volatile memory [58]. Typically, both the filenames *and* file contents of files in the lower filesystem



tem are encrypted, but eCryptfs may also be configured to not encrypt the filenames.

See also “Upper Filesystem,” “Data at Rest,” and “Non-volatile Memory.”

**Non-volatile Memory:** A digital storage device that does *not* require power to retain the information it stores. See also “Data at Rest.”

**Raw Image:** An uncompressed *disk image*. See also “Forensic Copy.”

**Sparse Acquisition:** One of two common *methods* of acquisition, often contrasted with logical acquisitions. Sparse acquisition is more complete and includes the same data as a logical acquisition with the addition of “fragments of unallocated (deleted) data” [89, p. 107].

**Static Acquisition:** One of two general *types* of acquisition. Static acquisitions are carried out against a data source which is unchanging, for example, “a computer seized during a police raid” [89, p. 106]. Static acquisitions are often contrasted with *live acquisitions*.

**Thin Client:** A type of computer designed to rely on other computers and servers to perform the majority of their computations and store most of their data. See also “Fat Client.”

**Unicast Address:** Defined by RFC 4862 [85] to be “an identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address.”

**Upper Filesystem:** In a stacked encrypted filesystem, such as eCryptfs, the upper filesystem refers to the set of decrypted files, sometimes referred to as “plain text” files [58]. The upper filesystem only ever exists in volatile memory and requires the successful loading of the encryption key(s) associated with the directory in which the files reside. See also “Lower Filesystem,” “Data at Rest,” and “Volatile Memory.”

**Volatile Memory:** A digital storage device that *requires* power to retain the information it stores. See also “Data at Rest.”

## Chapter 1

### INTRODUCTION

Since its advent and subsequent prevalence, the World Wide Web has transformed how people around the globe interact with each other, conduct business, access information, and enjoy entertainment, among many other activities. From its humble beginnings as a means for physicists to share experimental data with trusted colleagues, the Web has evolved not only in the content available but also in the methods and devices people employ to access the Web. In fact, in 2008 the number of devices connected to the Internet<sup>1</sup> surpassed the number of people using those devices [43]. With the prospect of 40+ billion devices connecting to the Web by the year 2020 as part of the Internet of Things (IoT) [2], this trend is not likely to change anytime soon. The estimated number of people with access to the Internet as of July 1, 2016 is 3.4 billion, or 46.1% of the world's 7.4 billion population [68]. These users have access to approximately 1.2 billion websites [69] and in a single day make approximately 3.5 billion Google searches [67] and send about 500 million tweets [70].

As the public's use of the Web has increased, so has technology-related crime. The FBI Internet Crime Complaint Center (IC3) issues annual reports summarizing the number of complaints they receive each year and the total financial impact of the incidents, as Table 1.1 summarizes [66]. When they first began releasing this annual report in 2008, the IC3 received more than 275,000 complaints and recorded \$264.6 million in losses. In the most recent report (2016), the number of complaints rose to more than 298,000, an

---

<sup>1</sup>Although the Internet and the Web are not synonymous, access to the Web requires access to the Internet; furthermore, traffic from HTTP and web services account for the vast majority of Internet traffic, according to a May 2015 report by Sandvine [102].

Table 1.1: Number of complaints received by the FBI IC3 by year with the financial losses reported (in USD).

Year	Complaints	Loss in Millions
2008	275,284	\$264.6
2009	336,665	\$559.7
2010	303,809	— <sup>2</sup>
2011	314,246	\$485.2
2012	289,874	\$525.4
2013	262,813	\$781.8
2014	269,422	\$800.5
2015	288,012	\$1,070.7
2016	298,728	\$1,450.7

increase of only 8.5% compared to 2008, but reported losses totaled more than \$1.4 billion, which is more than five times the losses from 2008. Although these figures from the IC3 are troubling, security experts believe the reported numbers are far lower than the actual financial losses, suggesting that “relatively few victims are reporting cyber fraud to federal investigators” [74].

Criminals’ motives vary widely for why they utilize technology in the crimes they commit. Sometimes technology is only incidental to a crime, such as when a mugger has a smartphone in their pocket, not realizing it is tracking their location. However, some criminals intentionally adopt technologies for the benefits they offer, such as scale and some degree of anonymity. For example, the perpetrators of Distributed Denial of Service (DDoS) attacks, cyber espionage, and phishing schemes all leverage the Web for the ease of reaching their victims and causing damage.

Here is a real-world example of how the Web is facilitating criminal activity in new and interesting ways. In a February 2017 article, WIRED magazine reported that operatives

---

<sup>2</sup>The 2010 IC3 report does not contain a figure for net losses, only subtotals for various categories.

originating from Moscow and St. Petersburg, Russia, had figured out how to cheat a class of aging slot machines by deducing the workings of their pseudo-random number generator (PRNG). Initially, this required them to record and then upload video of the slot machine as it operated, which would then be analyzed to “discern [the machine’s PRNG] pattern” by servers in Russia with the computing resources necessary to perform the calculations [73]. Security experts now suspect that these scammers are “streaming video back to Russia via Skype” to facilitate the process [73]. Whether or not this advancement has become common among these criminals, it is ultimately the Web and the ease of long-distance communication that makes this scam possible.

When any kind of technology is involved in a crime, it is the task of digital forensic examiners to discover, preserve, and analyze the data stored on the devices collected as evidence. In order to do this, examiners rely on specialized tools and programs designed to (1) recover and decode the data on the device, (2) help the examiner identify important information, and (3) export the examiner’s findings to a format usable in legal proceedings (in a criminal case) or administrative reviews (for internal investigations).

For digital forensic examiners, keeping pace with trends in technology and cyber crime is difficult. The Web’s rapid evolution and penetration into so many domains has resulted in a genealogy of web technology that is particularly dense. In other words, things change so quickly that new “generations” of web technology emerge at a rapid pace. In addition, the *content* on the Web is mercurial and changes constantly<sup>3</sup>. Consider for example a person’s Twitter feed; it may be static for only a few minutes or even seconds. Capturing and understanding information within such a short window before another change may occur is challenging.

---

<sup>3</sup>The existence of the Internet Archive’s Wayback Machine project (<https://archive.org/web/>), which captures snapshots of websites at different intervals, is another testament to the changeable nature of content on the Web.

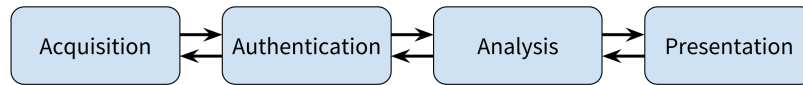
Forensic examiners need help with this challenge. They need methods that guide them to know how to approach this new type of evidence in a manner that fits well with their established procedures, but takes into account the unique characteristics of the Web. They also need tools that have been built specifically for the Web, but can also interface with other tools (existing ones as well as tools yet to be developed) to achieve more advanced analyses. While these needs go unmet, forensic examiners cannot fully perform their work of protecting the public from cyber criminals.

## 1.1 Digital Forensics

Digital forensics is the science of identifying, preserving, analyzing, and interpreting digital evidence, and has the goal of understanding what happened during an incident (crime, cyber intrusion, etc.) by, in a sense, looking into the past and reconstructing a timeline of events. In other words, digital forensics seeks to discover previously unknown data and frame it in a context relevant to the incident.

Forensic examiners are expected to maintain a certain set of standards in their work in two key areas. First, because digital forensics is a science, researchers and examiners have developed processes to help ensure that they approach their work in a systematic and thorough manner. Second, digital forensics is commonly used by law enforcement as part of their investigations, requiring examiners and their tools to abide by the applicable laws of the jurisdiction. I will now provide additional details on these two sets of standards.

Figure 1.2: The four steps of the forensic process.



### 1.1.1 The Forensic Process

The general process of digital forensics has four main steps<sup>4</sup> as illustrated in Figure 1.2: (1) acquisition, (2) authentication, (3) analysis, and (4) presentation (or report). In any investigation, once the examiner has secured the evidence, the first step is to *acquire* a “forensic copy,” which is an exact duplicate of the original. Forensic copies serve as a protection for the original evidence because the examiner works with the copies instead of the originals, allowing the examiner to perform analyses without the risk of compromising the integrity of the evidence (see the Glossary). The examiner must then *authenticate* the forensic copy to ensure the copy exactly matches the original.

With an authenticated copy of the evidence, the examiner may then begin to *analyze* the evidence for inculpatory (suggesting guilt) and exculpatory (suggesting innocence) information. After sorting through the data on the evidence, the examiner will then compile a *report* detailing her findings, where the information was located on the evidence, and any other relevant information.

Each of these stages requires sound approaches that (except for perhaps the *report* phase) have been tailored to every distinct type of digital evidence. Over the past few decades, digital forensics researchers have developed many tools to accommodate an array of evidence types [53], but digital forensic researchers are in a constant struggle to keep pace with technological advancements. Their struggle is due in part to the rapid pace of

---

<sup>4</sup>Since as early as 1984, researchers and practitioners have proposed several process models for digital forensic investigations. The model I use in this work is similar to the Generic Model proposed by Yusoff et al. in [124], but without the Pre-Process and Post-Process steps, which do not directly apply to examiners.

change that technology undergoes, but part of it is also because forensic examiners must maintain high standards in their work, such as complying with the Rules of Evidence (Subsection 1.1.2).

With regards to evidence originating from the Web, these four stages are even more challenging, particularly *acquisition* and *authentication*, because the examiner will not have direct access to the original data. For *acquiring* the evidence, if the examiner has the benefit of being able to subpoena the service provider for data, they are still at the mercy of the provider's willingness and ability to comply. Furthermore, the format of the data is likely to be slightly altered from how it was stored on the servers. If the examiner cannot rely on the impetus of a subpoena (as is the case for corporate and non-government examiners) the best they can hope for is to utilize a provider's APIs, but this is only a viable option in the event that they have the proper authentication credentials. For *authenticating* the evidence, without access to the original data it is not possible to perform the traditional evidence authentication techniques (e.g., cryptographic checksums). Therefore, examiners must devise other methods to ensure the integrity of the evidence they receive.

### 1.1.2 Rules of Evidence

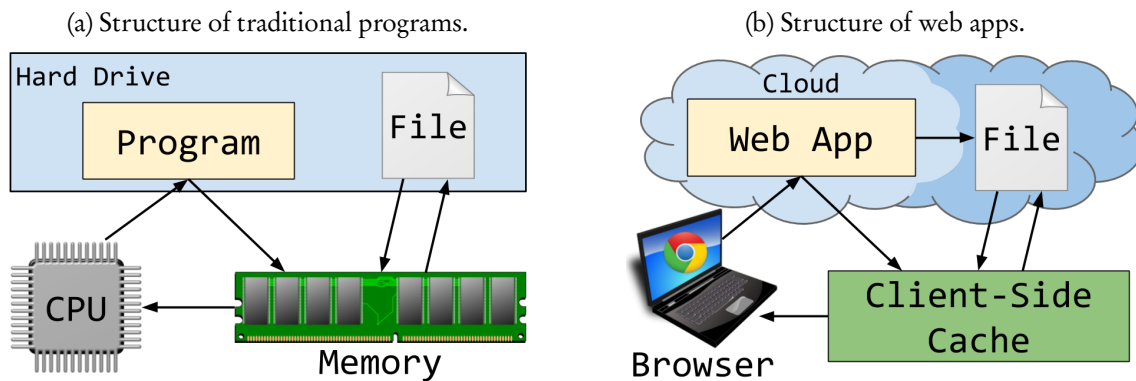
The Rules of Evidence give protection to both victims and suspects by helping ensure that the conclusions examiners draw from the evidence are accurate by establishing standards in four dimensions: Admissibility, Completeness, Authenticity, and Accuracy [4]. For the full definitions of these terms, please see the Glossary. Here are brief descriptions of each Rule:

**Admissibility:** The evidence was handled in accordance with applicable laws.

**Completeness:** The evidence tells the full narrative.



Figure 1.3: Comparison of traditional programs and web apps.



**Authenticity:** The evidence makes an explicit link to an individual.

**Accuracy:** The tools used on the evidence function correctly and consistently.

## 1.2 Web Apps vs Traditional Programs

Web apps differ from traditional computing models in significant ways, as Figure 1.3 illustrates. I will now explain these differences to highlight some of the unique forensic challenges introduced by web environments.

A traditional program (Figure 1.3a) first has its code loaded from the hard drive into memory, then individual instructions are executed by the CPU. The code may access files stored on the hard drive, which will require that the file also be loaded into memory. When the program saves changes to the file, its data will be copied from memory and written back to the hard drive.

The flow of a web app's execution (Figure 1.3b) is similar in structure to that of a traditional program, but incorporates different components<sup>5</sup>. The user tells the browser

---

<sup>5</sup>Of course, a web app's code still has to be loaded into memory and executed on the CPU, but the focus here is on the higher-level concepts that distinguish web apps from traditional programs.

which web app they want to use by entering its URL. The browser retrieves the web page, which includes some client-side code that the browser caches locally and then runs. As the user interacts with the app, they may open and edit files stored either in the same cloud as the web app or a different cloud altogether. The contents of the file as downloaded to the user's machine may be altered from the form in which it resides in the cloud. And finally, any changes the user makes to the file may be either uploaded by the browser directly to the location where the file is stored, or the server-side code of the web app may, on behalf of the user, make the changes to the file in the cloud.

### 1.3 Web Thin Clients

Web thin clients are a relatively new type of computer based on the idea of using web pages and web apps in lieu of more traditional desktop programs. One consequence of this approach is that the hardware requirements are considerably lower for running the operating system (OS), making them less expensive to manufacture. Also, web thin clients are simpler than traditional OSes, making them easier to maintain. As IDC analyst Linn Huang told TechCrunch in a recent interview, the administrative tools for web thin clients “[have] really been fantastically received and [have] made it possible for IT administrators to manage profiles on individual machines or manage multiple [users] on one machine” [61].

Web thin client products have experienced a high rate of adoption in the last few years, particularly in the consumer and K-12 markets. In 2013, web thin clients only accounted for approximately 9.6% of all sales of desktops, notebooks, and tablets [93]. Then, as of the third quarter of 2015, they accounted for more than 51% of all sales for the K-12 market in the U.S. [106]. In the first quarter of 2016, they “overtook Mac OS in the U.S.

in terms of shipments” [115]. Finally, as of July 2017, the sales of web thin clients “for K-12 continued its momentum unabated,” which helped PC sales in the U.S. to “[pull] ahead of forecast” because they “remained a source of significant volume growth” [65].

Despite its success in the consumer market, there has been very little forensic research done on this form of computing architecture [79]. The implication here is that, although the likelihood has been increasing that law enforcement will seize these devices at a crime scene, forensic examiners do not have any methods or tools for recovering any of the residual evidence stored on them. One reason there are no methods and tools for web thin clients is they are built to be secure, preserving users’ privacy and protecting against malicious software by using techniques such as encryption and operating system verification. I discuss these and other security features that relate to forensics in Section 2.2.

As web thin clients continue to increase in popularity, it will be increasingly more important for examiners to have effective forensic methods and tools that have been designed for web thin clients and can overcome the obstacles examiners currently face. In addition to new acquisition tools and storage formats, examiners will also need new analysis techniques since none currently exist.

#### 1.4 Contributions of this Dissertation

Because of its widespread use today, web services and other web environments contain important information for forensic practitioners. However, there exists a non-trivial capability gap in the methods and tools available that prevents examiners from taking full advantage of this data in their investigations.

In this dissertation, I present the details of my work to design and create forensic tools and methods for web environments that benefit digital forensic examiners by pro-

viding them with (1) previously unknown data and (2) relevant context of the incident. I accomplish these two objectives by ensuring that, collectively, these forensic tools and methods can: (1) discover and acquire previously inaccessible evidence, (2) narrow down the analysis space of the evidence, (3) reconstruct a timeline of the events that transpired, and (4) leverage structured formats for storing evidence and analysis results. These two objectives and four requirements guide my work on web environment forensics. Following is a summary of the contributions from this dissertation.

Before developing forensic methods and tools for a new type of technology, it is necessary to (1) study the technology and gain an accurate understanding of its characteristics, and (2) create a plan outlining how to approach the problem while accommodating the distinct features of the technology. To this end, I developed a new framework for forensics in web environments (Chapter 3) based on the objectives and requirements listed above in such a way that it addresses issues common to all web environments. The framework gives examiners a new way to think about how to approach forensics in this new domain and identifies key areas they must take into consideration, yet it is also flexible enough to fit into any existing examination workflow.

Using my framework for web environment forensics, I developed a novel approach to extract residual evidence stored on web thin clients (specifically Chrome OS devices) that allows an examiner to determine which extensions and apps are installed on an encrypted web thin client, without breaking or otherwise extracting the encryption keys (Chapter 4). In my approach, I generate signatures or fingerprints of extension and app code that persist after encryption, and I am able to use these fingerprints to identify the installed extensions and apps.

To complement the residual evidence I extract from web thin clients, I devised a method to leverage service providers' APIs to act as a proxy for the device to acquire the data that

is inaccessible directly from the device due to encryption (Chapter 5). This data is used to reconstruct the timeline of events that took place on the device and create a visualization of the timeline data to assist a forensic examiner.

In addition to the methods and tools I created for web thin clients and web service providers, I also designed several structured formats for storing both the evidence from the Web and the analysis results from tools that analyze that evidence. In the appropriate chapters, I also discuss how these structured outputs may be integrated with other tools for even more advanced analysis or processing.

## Chapter 2

### BACKGROUND

#### 2.1 Sharing Structured Data

Digital forensic researchers have long recognized the benefits of leveraging structured, machine-readable formats to store data, such as improved collaboration, automated analysis, tool integration, and verifiable output [52, 35, 51]. But some of the more recent developments in this area have come from the field of incident response, a discipline closely related to digital forensics<sup>6</sup>.

Incident responders' primary duty is to investigate cyber incidents against their organization. To do this, these professionals use the same (or similar) tools to those used by forensic examiners. The ultimate goal of an incident responder is to understand what happened for the purpose of stopping any ongoing attacks and preventing future attacks.

As the Web has become an increasingly hostile environment, security experts and incident responders from both industry and government organizations have placed greater importance on sharing data about the security incidents they experience, also referred to as "threat intelligence." Typically, these organizations rely on two factors for the sharing process to work. First, each participating organization must have some level of trust in those with whom they share their data [57]. Trust is of paramount importance when such data may contain sensitive information about the reporting organization's customers, intellectual property, financial information, internal structure, procedures, and so forth.

---

<sup>6</sup>In fact, many times incident response (IR) and digital forensics (DF) are lumped together and referred to by the acronym DFIR for things such as Twitter hashtags.

The second requirement is that there be a way for different organizations' security tools to ingest this data, the key to which is for the data to be stored in a structured format. The Structured Threat Information eXpression (STIX™) language [9] has become one of the most common formats for organizations to use to exchange their incident data.

The only differences between how digital forensic practitioners and incident responders use structured data are that (1) threat intelligence has more of a bias towards network-related data than forensics typically does and (2) forensics targets law enforcement as its end users, which means everything must abide by the laws governing evidence handling and analysis tool standards. Incident responders and digital forensic examiners alike can leverage structured formats to achieve the many benefits I mentioned above.

## 2.2 Chrome OS: Key Features

Chrome OS has a number of features that distinguish it from other operating systems. While I will not attempt to enumerate all of these features here, I will introduce several that have direct implications for researching methods to preserve forensic evidence stored on a Chrome OS device.

### 2.2.1 Encryption

Chrome OS encrypts all user data using eCryptfs [58, 29, 32], and it uses several keys to encrypt both the names and content of the files (see Table 2.1). This practice protects users' privacy from opportunistic thieves [33]. In accordance with common practices in digital forensics, I make the assumption that the data at rest (on the hard drive) will be the main focus of an analysis. I also assume I do not have the users' credentials and that

Table 2.1: List of keys used by Chrome OS to encrypt user data. The System-Wide Key is also known as the TPM Cryptohome Key, hence the acronym TPM\_CHK. The length (in bits) of the TPM\_CHK is undocumented. The VK consists two 128-bit keys: the file encryption key (FEK) and file name encryption key (FNEK).

Key name	Type/Bits	Scope	Origin	Encrypted by	Stored on
Storage Root Key (SRK)	RSA 8192	Hardware	Manufacturer	TPM	TPM
System-Wide Key (TPM_CHK)	RSA	System	TPM on 1 <sup>st</sup> boot	SRK	TPM
Vault Keyset Key (VKK)	AES 256	User	Random	TPM_CHK and TK	Disk
Vault Keyset (VK)	AES 128	User	TPM on 1 <sup>st</sup> log-in	VKK	Disk
Temporary Key (TK)	SHA 256	User	Hash of password	N/A	N/A



breaking the encryption is infeasible, which means the filenames and their contents are unavailable. However, from my analysis of eCryptfs as Chrome OS uses it, there is one feature of eCryptfs that I can use to my advantage.

As described in [58], “eCryptfs is a stacked filesystem that encrypts and decrypts the files as they are written to or read from the lower filesystem.” The “lower filesystem” is the set of encrypted files that are at rest on the disk, and the “upper filesystem” refers to the decrypted files. Because eCryptfs encrypts files *individually* before writing them to disk, the file’s metadata remains mostly intact. Although there are slight variations in the timestamps and file size (specifically, AES is a block cipher, so eCryptfs will round the size of a file up to the nearest 4096 bytes), the metadata of the encrypted file deviates within a negligible range.

### 2.2.2 TPM and Secure Boot

Chrome OS stores some of the encryption keys in the Trusted Platform Module (TPM) as shown in Table 2.1. As described in full detail in the Chrome OS documentation [32, 30], a number of malicious actions will jettison the encryption keys, making the data unrecoverable.

Chrome OS also uses Secure Boot<sup>7</sup> to check that the firmware and kernel both pass integrity checks before booting, thereby preventing tampering with the operating system [34]. Secure Boot also prevents an examiner from booting a custom OS for doing forensics, as has been done previously with challenging systems [104], without effectively destroying the evidence on the disk. Without being able to boot such a custom OS, it is also infeasible to perform a memory dump or memory analysis.

---

<sup>7</sup>Sometimes referred to as verified boot.

### 2.2.3 Limited Local Storage

A benefit of web thin clients is that, due to the local hard drive being a cache of the users' cloud data, they typically do not require much storage space for local files and thus can use hard drives that are cheap and small, yet fast. From a business perspective, using a 16 GB solid state hard drive provides a good user experience with quick boot times while keeping costs down. Because Chrome OS devices are thin clients and rely on servers for computation and storage, this also makes sense to only store a cache of recently used files from the user's Google Drive.

From a forensics perspective, this is a new challenge to have a suspect's device that has some *unknown subset* of their files stored locally instead of having all their data on the acquired evidence. Given the Completeness rule of evidence [3], this means that by analyzing just the files on the device, an examiner cannot tell the complete narrative of the crime or incident.

### 2.2.4 Extensions Enhance Functionality

Chrome supports extensions that “modify and enhance the functionality of the Chrome browser” [26]. In fact, in Chrome OS, extensions provide *all functionality* outside the browser itself, including the bookmarks manager, PDF viewer, and Zip file unpacker.

While extensions help Chrome OS devices mimic the familiar functionality of fat client PCs by using a variety of programs with the system, the differences here are that (1) because the data is encrypted, it is presently unclear what extra programs the suspect used with the system, and (2) it is unlikely that all the files and data created with these

extensions are stored locally, both because of the caching issue discussed previously and because many of these extensions rely heavily on web and cloud services.

Strictly speaking, there are two distinct classes of Chrome OS apps: those that run strictly as a traditional web application and store all data on the servers and those that store some data locally (for the client) to allow for “offline mode.” Some rich Internet applications and HTML5 web applications, however, may act as both.

In this dissertation I will refer almost exclusively to extensions for Chrome OS. This is not because my approaches only work with extensions and not with apps. On the contrary, apps *are* extensions that include functionality not meant to be a part of the regular browser window. At the filesystem level, Chrome OS stores all the program files for an app in the same directory as those for extensions, and the folder is named “Extensions”<sup>8</sup>. Because Chrome OS treats apps as a subset of extensions, and because Google’s support for apps is declining [39], throughout this dissertation I will use the term “extensions” as a generic way of saying “extensions and apps.”

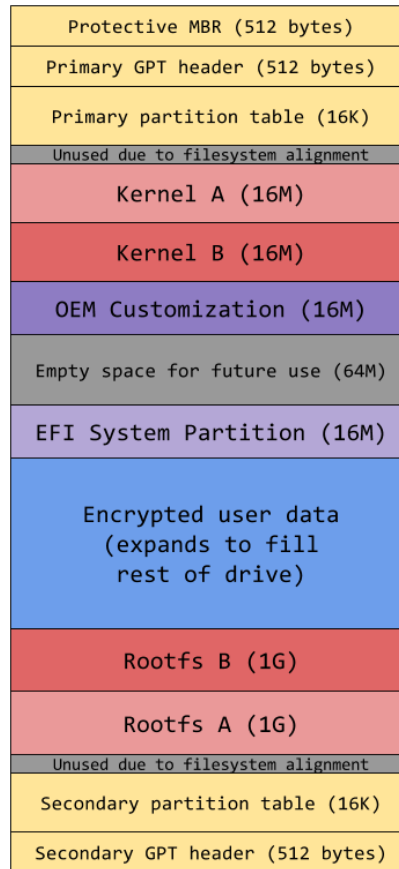
### 2.2.5 Disk Layout

Chrome OS has a well-defined disk layout with a specific number of partitions, each with a prescribed purpose, as Figure 2.2 illustrates [29]. One partition is reserved for OEM customization, in which manufacturers may store “web pages, links, themes, etc.” [29]. Another is the user state partition, or “stateful partition”, in which the operating system stores all the users’ encrypted files. This partition has the name STATE and has a mount point of /home once booted. The remaining partitions are reserved for the kernel, root filesystem, and partition table header and entries. This last set of partitions are not

---

<sup>8</sup>The Chrome browser also functions in this manner.

Figure 2.2: Layout of partitions on a Chrome OS device.



normally writable by the user, and their modification would interrupt the normal boot sequence as discussed in Subsection 2.2.2.

All models of devices running Chrome OS will have the same layout, with the possible exception of the expansion of the STATE partition to fill a larger hard drive. Variations on Chrome OS, including Chromium OS and CloudReady, also use this disk layout.

### 2.2.6 Anatomy of Extensions

All extensions are installed on a per-user basis, meaning the installation is specific to each user. While there may be a set of extensions installed by default or as part of a

group policy<sup>9</sup>, these will only be installed on a particular device once the user sets up their account and logs into that device.

When a developer creates a new Chrome OS extension, they package it in a CRX file, which is an archive file format with a specific layout and set of custom headers [25]. When the browser installs the extension, it validates the headers, extracts the contents of the archive, rewrites the extension's manifest to include the developer's public key, performs localization, and converts all images to PNG format (without changing the file extension).

Chrome OS installs extensions to the following upper filesystem path<sup>10</sup>, where <uid> is the system-assigned ID for the user:

---

```
/home/user/<uid>/Extensions/<extension_ID>/<version>/
```

---

which maps to the encrypted (lower filesystem) path:

---

```
/home/.shadow/<uid>/vault/user/{Extensions}/{<extension_ID>}/  
↪ {<version>}/
```

---

Chrome OS stores the extension's files according to the extension's version, so it is possible for the extension directory to have multiple versions. However, when Chrome updates an extension it removes the old version after the next restart; therefore, in practice there will only be one version directory per extension except when one was recently updated.

Extensions do not have permission to write in the folder where its files are located. This is a key insight from my analysis into the extension installation and configuration process

---

<sup>9</sup>See <https://support.google.com/chrome/a/answer/188453>.

<sup>10</sup>Angle brackets <> indicate *variable* directory names, curly braces {} indicate *encrypted* directories.

Listing 2.3: Locations where extensions may store user-specific data on Chrome OS.

---

```
/home/user/<uid>/databases/  
/home/user/<uid>/IndexedDB/  
/home/user/<uid>/Local Storage/  
/home/user/<uid>/Storage/
```

---

on Chrome OS and yields two important corollaries. First, none of the user-generated or user-specific data or configurations are stored in the Extensions directory. Instead, extensions may store such data in one or more of the upper filesystem paths in Listing 2.3. The second important corollary is that, for a given version of an extension, the files in the installed extension directory will be constant.

### 2.2.7 Developer Mode

Since it was first released, every Chrome OS device has had some way of switching to what Google calls “developer mode,” which relaxes the secure boot requirements and allows users to “run custom (non-Google-signed) images” [31]. Developer mode also makes it possible for users to access a command prompt and a Bash-like shell via either virtual terminal 2 (VT-2) or `crosh`, one of the built-in extensions that provides very minimal shell-like commands when not in developer mode. In both these cases, users can log in as the `chronos` user, which “includes the ability to do password-less `sudo`” [31].

Because users have shell access in developer mode, they have the ability to access the full functionality of the underlying Linux kernel. The Chrome OS enthusiast community

has developed clever ways to take advantage of this access, including installing regular Linux distributions using the `chroot` operation<sup>11</sup>.

When a Chrome OS device is seized as evidence, it may or may not be in developer mode. Although a device in developer mode would allow a forensic examiner greater access to the device and potentially permit them to view data that would typically be encrypted, it is more probable that the device has not been put into developer mode and such courses of action would not be possible.

### 2.2.8 Users

Chrome OS manages users in a slightly different way than other Unix-based operating systems. To remove some ambiguity, I will use the term “system user” for a user in the Unix sense and “Chrome user” for a user with a Google account.

In Chrome OS, the `chronos` system user is the owner of all files related to Chrome users. So, when Chrome OS adds a new Chrome user to the system, rather than adding an entry to the `passwd` and `shadow` files, the actions it takes are primarily related to setting up a new `eCryptfs` store for the user and syncing the Chrome user’s settings, extensions, etc.

Another interesting fact comes from the output of the `mount` command, which Listing 2.4 captures (once again, `<uid>` represents the system-assigned ID for the user). This output shows that Chrome OS mounts the encrypted stateful information directly at the home directory of the `chronos` system user, and the Chrome user’s files are mounted in five different locations in the filesystem.

From a forensics perspective, there are two key takeaways from this information. First, at the filesystem level, all files under `/home/user/` are owned by the `chronos` system user.

---

<sup>11</sup>The most popular `chroot` solution is `Crouton` — <https://github.com/dnschneid/crouton>.

Listing 2.4: Abbreviated output of the mount command in Chrome OS.

---

```
/dev/sda1 on /home type ext4
/dev/mapper/encstateful on /home/chronos type ext4
/home/.shadow/<uid>/vault on /home/.shadow/<uid>/mount type ecryptfs
/home/.shadow/<uid>/vault on /home/chronos/user type ecryptfs
/home/.shadow/<uid>/vault on /home/user/<uid> type ecryptfs
/home/.shadow/<uid>/vault on /home/chronos/u-<uid> type ecryptfs
/home/.shadow/<uid>/vault on /home/root/<uid> type ecrypt
```

---

So, when looking at the filesystem metadata, it is not immediately apparent which Chrome user a particular file actually belongs to.

Second, there is an additional user management layer provided by Chrome OS that is completely separate from the system user layer. In this layer, a Chrome user is authenticated when their password successfully decrypts the home directory of the user they claim to be. Implied in this is the limitation that, since only one Chrome user can be logged into the system at a time, only one encrypted vault can be decrypted and mounted at a time.

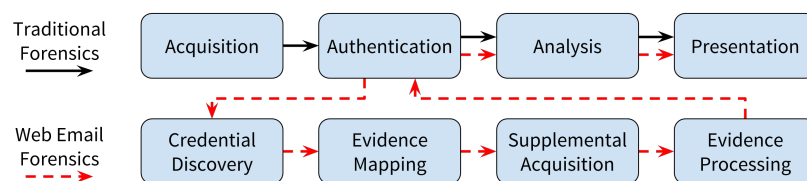
### 2.3 Email Forensics: Preliminary Effort

Some of the work in this dissertation, particularly the framework in Chapter 3, was motivated by my previous work on web email forensics [96]. In that project, I recognized that web email differs from traditional email solutions in ways that are significant in a digital forensics context and which (when the work was published) existing forensic tools did not accommodate. This lack of support resulted in greater difficulty for examiners to discover, acquire, authenticate, store, and analyze email evidence from these sources.

Because my work to create a framework for web email forensics taught me several lessons that affected the research I conducted for this dissertation, I will present the high-



Figure 2.5: The traditional forensic workflow combined with the web email approach.



level details of that work, including some improvements I have made since it was first published.

### 2.3.1 Approach

My approach provides the means whereby analysis tools can handle and analyze evidence originating from online sources. In brief, my process consists of discovering online credentials from acquired evidence, mapping those credentials to their corresponding services, acquiring evidence from each service, authenticating and processing that evidence into a standardized representation format, and then performing the actual analysis, as Figure 2.5 depicts. This process is an evolution of the standard forensic process I discuss in Subsection 1.1.1, with changes that (1) still fit within the standard process, yet (2) accommodate the unique characteristics of web email.

In addition to presenting a methodology based on this new forensic process, I also apply the process to a case study in which I acquire emails from a Gmail account by reusing the cookies created when the user logs into their account in the Google Chrome browser, which are stored to disk and then recovered by an examiner.

### 2.3.1.1 Initial Acquisition

As in any investigation, once the examiner has secured the evidence, the next step is to acquire a forensic copy<sup>12</sup> of it and authenticate that the copy is exact; in this work I will refer to these actions as the *initial acquisition*. It is necessary to complete the initial acquisition before the examiner can search the disk for any information related to the suspect's online activity and accounts.

### 2.3.1.2 Credential Discovery

In my process I use the term “credential” to denote any data which can identify the owner of the credential (e.g., the suspect) in such a way as to enable an examiner to reestablish a connection with online services to extract evidence. The breadth of this definition allows for its application without respect to the format in which the data is stored or the type of service to which it is mapped. Additionally, I define a process phase, “credential discovery”, in which I search for and detect credentials stored in a piece of evidence which an examiner can use to further recover additional evidence for the investigation.

In my proof of concept, I used a method for discovering credentials that I call **Search known locations**, in which I search for files known to regularly store credentials, such as key ring databases, cookie files or databases, registry entries, etc. My implementation specifically targeted the cookies for the Google Chrome browser, which, on a Windows 7/8/10 machine, it stores at %USERPROFILE%\AppData\Local\Google\Chrome\User Data\Default\Cookies. As long as the filesystem is intact enough to navigate to

---

<sup>12</sup>See the Glossary for a discussion on the term “forensic copy”.

this location, and assuming the user uses Chrome, this method of discovery will always yield some credentials.

#### 2.3.1.3 Evidence Mapping

Following the discovery of credentials, it is necessary to map them to a source of evidence before performing any further acquisition. In other words, this mapping determines what online service the suspect accessed using the credential. The level of difficulty of the mapping process may vary depending on the approach taken to discover a set of credentials, the form in which they were stored, and any accompanying data stored with the credentials. Examples include email addresses or cookies which specify the domain to which they belong, spreadsheets organized in a manner which makes this information evident to an examiner, or a text file may store a user name and password with no indication of the service for which they are valid. Manual examination may be necessary to complete the mapping process if this is the case.

In my example, identifying the cookies for a Gmail account is straightforward because the fields shown in Figure 2.6 will be present. The mapping module of my implementation searches for these fields and upon detecting them identifies this cookie database as containing credentials for Gmail.

#### 2.3.1.4 Supplemental Acquisition

After mapping a credential to a service, the next step is to acquire a forensically sound copy of the service's data. In my approach, I *reuse* the credentials previously discovered, *acquire* the most complete representation of the email (including headers and body), and

Figure 2.6: All the cookies created by logging in to Gmail with the “Stay signed in” option selected. The expires, secure, httponly, last\_access\_utc, has\_expires, and persistent columns were omitted for readability.

creation_...	host_key	name	value	path
13003520...	accounts.google.com	GALX	[REDACTED]	/
13003520...	accounts.google.com	__utma	[REDACTED]	/
13003520...	accounts.google.com	__utmb	[REDACTED]	/
13003520...	accounts.google.com	__utmc	[REDACTED]	/
13003520...	accounts.google.com	__utmz	[REDACTED]	/
13003520...	mail.google.com	S	[REDACTED]	/mail
13003520...	accounts.google.com	GAPS	[REDACTED]	/
13003520...	accounts.google.com	RMME	[REDACTED]	/
13003520...	.google.com	NID	[REDACTED]	/
13003520...	.google.com	SID	[REDACTED]	/
13003520...	accounts.google.com	LSID	[REDACTED]	/
13003520...	.google.com	HSID	[REDACTED]	/
13003520...	.google.com	SSID	[REDACTED]	/
13003520...	.google.com	APISID	[REDACTED]	/
13003520...	.google.com	SAPISID	[REDACTED]	/
13003520...	mail.google.com	GX	[REDACTED]	/mail
13003520...	mail.google.com	GMAIL_AT	[REDACTED]	/mail
13003520...	.google.com	PREF	[REDACTED]	/
13003520...	mail.google.com	gmailchat	[REDACTED]	/mail

then *store* them as a separate copy in an intermediate format for the purpose of evidence processing into a format which examiners will later use.

The method employed to acquire data from the service provider may vary widely depending on the access methods the service provider makes available (e.g., APIs). For my proof of concept with Gmail, I researched what options and settings are available when I only have the browser cookies to log in. While the optimal acquisition method for retrieving a copy of all emails is to do so via IMAP, cookies are specific to the HTTP protocol and will not work to authenticate through IMAP. If plain text credentials (user name and password) were discovered, acquisition via IMAP would be possible.

Since I only have the browser cookies to work with in my proof of concept, I have a limited ability to change any account settings that will help the process of acquisition. Fortunately, Gmail allows users to grant other Gmail addresses access to their account without reentering their password. The account that is granted access is called a “delegate” and can read all the emails in the granter’s account as well as send emails on their behalf.

While Gmail does not provide IMAP access to grantee accounts, creating the delegate prolongs access to the target account past the two week expiration date of the cookies, allowing for any needed follow-up acquisition.

With this understanding, I wrote a pair of tools in Python to complete the supplemental acquisition. The first tool, which I call Crumbler (see Appendix C.6), imports the cookie database from its native SQLite format to a custom subclass of the common `Cookie` Python object. The second tool automates the process of adding a delegate to an account using the Selenium<sup>13</sup> web driver. It opens a browser and connects to Gmail, and as long as the cookies are still valid it performs each of the steps for adding a delegate as outlined in the Google help pages<sup>14</sup>. Once Crumbler has completed this process, the grantee can access the target account by logging in to Gmail, clicking on their email address in the top right hand corner of the screen, and selecting the target account from the list of accounts to which they have access.

The final challenge to acquiring data from Gmail is that the only method for retrieving the raw email data is to essentially “screen scrape” the pages returned during a web session, parsing through the HTML and using regular expression patterns or searching through the Document Object Model (DOM) for the desired elements. The contents of the Gmail account should then download and process the messages into a standard format.

#### 2.3.1.5 Evidence Processing and Authentication

During the development of my proof of concept, I surveyed the strengths and weaknesses of existing formats and concluded that the mbox format (RFC 4155 [60]) was best

---

<sup>13</sup>See <http://seleniumhq.org/>.

<sup>14</sup>See <http://support.google.com/mail/answer/138350>.

suitable to my purposes. The mbox format is a flat-file, plain text representation of email which is human readable and easy to parse. I assert that mbox is valid for use as a forensic copy format as it is “output readable by sight, shown to reflect the data accurately” and thus “is an original” [89, pg. 162], so long as the acquisition process used was sound.

A common practice in digital forensics is the use of XML as a data representation format, allowing for a firm layer of abstraction “between feature extraction and analysis” and “a single, XML-based output format for forensic analysis tools” [5]. For evidence representation in existing methodologies, DFXML is a standard to represent disk, partition, filesystem, and file data in a unified manner [50]. A major benefit of DFXML is the generality of its representation; regardless of disk geometry or forensic copy format, the evidence is represented in the same manner.

When analyzing email evidence, the most significant metadata is contained within the header of the email itself. Email headers include information such as the sender and recipient (From and To), unique message identifiers (Message-ID), reply addresses (Reply-To), and more (see RFC 4021 [72]). Even without considering the content or body of emails, these headers have been shown to be useful in forensic investigations as a means to achieve author attribution, detect attempts to obfuscate sequences of events during a time period of interest [8], as well as identify communication flows and perform social networking analysis.

Although DFXML is quite efficient for representing massive amount of data from a filesystem, it is ill-suited for storing the header information of emails, as filesystem metadata is not closely related to the analysis of evidence contained within email messages. While it could be extended to fit this purpose, the resulting format would become overly encumbered and its size efficiency greatly reduced.

I have defined a new evidence representation format, suitable for email forensics, that

borrowing concepts from previous work [5, 50] that I call Email Forensics XML (EFXML). Since email is text-based and can be easily represented in XML without complicated encoding, EFXML achieves many of the same benefits discussed by Alink et al. [5] and Garfinkel [50], including easy searching and classification of information. As an added benefit of using an XML-based format, EFXML has a clearly defined schema which can verify the output from tools that generate this format.

To reflect the data extraction capabilities provided by the “byte\_runs” element in DFXML, I included a simplified element which details the span of bytes within the mbox file where the email resides and can be extracted from using tools such as those designed for DFXML, the Unix command `dd`, or other comparable programs. While line numbers would be equally useful with regard to the mbox format, I decided to use the “byte\_runs” field in each representation to follow existing standards. Listing 2.7 gives an abbreviated sample of an EFXML representation of a mailbox.

The structure of EFXML as specified in its schema helps overcome one shortcoming of mbox as it relates to forensics, which is that there is no way to add metadata to the file. Of particular interest are the fields which help maintain the chain of custody by storing information on the name of the program that created the mbox and EFXML, the version of the programs, the date and time of their creation, the target email address, the size of the mbox file, and MD5 and SHA1 checksums for the mbox file. With this information, it is possible to keep track of how the evidence was acquired, authentication information for the entire set of emails<sup>15</sup>, and what programs handled the evidence at what time, all of which are required by the rules of evidence.

---

<sup>15</sup>Unless the target account has been frozen by the provider, acquiring emails from the same account at two different times will likely yield two distinct data sets, preventing the checksums from matching. As such, the checksums are provided for integrity checks against the same mbox archive to ensure it does not change while being analyzed and not against past or future acquisitions.

Listing 2.7: An Abbreviated Sample of an EFXML file that conforms to the EFXML Schema.

---

```
<?xml version='1.0' encoding='UTF-8'?'>
<efxml
  xmlns="https://mikemabey.com/schema/efxml" version="0.1"
  xmlns:dfxml="http://www.forensicswiki.org/wiki/Category:Digital_Forensics_XML"
  xmlns:h="https://mikemabey.com/schema/headers"
  ...
>
  <mailbox type="mbox">
    <Folder>
      <message>
        <h:Subject>&lt;! [CDATA[[dovecot] Re: some problem with
        ↵ dovecot]]&gt;</h:Subject>
        <h>Date>
          <h:year>2003</h:year>
          ...
        </h>Date>
        <h:From>
          <h:address>
            <h:alias>&lt;! [CDATA[Jesse Peterson]]&gt;</h:alias>
            <h:email>jpeterson275@attbi.com</h:email>
          </h:address>
        </h:From>
        <h:Not-In-RFC>
          <X-Original-To>dovecot@procontrol.fi</X-Original-To>
        </h:Not-In-RFC>
        <byte_run file_offset="1101673" len="2159">
          <dfxml:hashdigest type="md5">8e4bb7462b991183cf5b2adc87970227
          ↵ </dfxml:hashdigest>
        </byte_run>
      </message>
    </Folder>
  </mailbox>
</efxml>
```

---

### 2.3.1.6 Analysis

The next step after acquiring, processing, and authenticating the evidence is to perform forensic analyses that will be informative for the purposes of the investigation. Since my methodology only provides a process for the acquisition and storage of supplemental



evidence, the implementation of new analysis tools is beyond the scope of this work. However, my approach creates a well-defined, structured, and verifiable representation of email data. Since it is an XML format, developers can easily craft tools that adhere to the specification, validate their tools' output, and use common XML parsing libraries to facilitate the analysis process.

#### 2.3.1.7 Project Summary

In the email forensics project, I defined a general methodology for carrying out email forensics that accommodates the unique characteristics of web email while still fitting within normal forensic processes. My approach broadens the definition of credentials and demonstrates the need for a generic evidence representation. My implementation showed an example of credential discovery for Gmail accounts, a method for reestablishing existing Gmail sessions, the steps needed to carry out a supplemental acquisition, and a completed evidence processing phase generating representations in both an intermediate mbox format and the proposed EFXML format, which stores email headers to facilitate header analysis as well as collaboration among developers and organizations.

#### 2.3.2 Improving EFXML

The original version of the EFXML schema that was created as part of the web email forensics project had some issues with how it was designed. Although the schema did provide a way to include non-standard email headers in an EFXML file, the structure was a little strict in that the definition of email header types were all explicit sub-tags of the EFXML-specific tags.

To fix these issues, I completely rewrote the EFXML schema (see Appendix A) and divided it into two parts. The first part is a schema that defines the structure of an EFXML document, including tags for a mailbox, a Folder (which is a division inside a mailbox), and a message. The message tag then allows an unbounded number of header tags.

For the second part, I moved the definition of all the email headers out of the EFXML schema and into their own schemas. In total, I created four separate schemas that represent the four types of header data: email, MIME, Netnews, and HTTP<sup>16</sup>. I used as a guide IANA's list of all message headers<sup>17</sup> and the many RFCs cited there. The four header schemas all inherit from a generic header schema, which makes it possible to only reference one type of header or multiple types. For example, according to the email RFCs, MIME headers are allowed to be included in an email message, so an EFXML document can use both header types and still validate against the schemas.

Within the schema files, each header tag definition includes an annotation that (1) lists which protocol the header type belongs to, (2) references the RFCs that define the header type, (3) lists whether the header is permanent or provisional, and (4) gives the status of the header (e.g., standard, obsolete, etc.). All of this information is meant to make the header schemas useful for a variety of XML documents, not just EFXML.

### 2.3.3 Lessons Learned

As I mentioned previously, the web email forensics project taught me valuable lessons that proved critical to the development of the techniques presented in this dissertation.

---

<sup>16</sup>Currently, only the email and MIME header schemas have been created. However, since the project is open source, anyone can contribute and help complete the set.

<sup>17</sup>Available at <https://www.iana.org/assignments/message-headers/message-headers.xhtml>.

First, when it comes to developing forensic approaches for new types of evidence or domains, it is beneficial to build on and adapt existing forensic processes. Not only does this help make the new approach more accessible for forensic examiners and more likely to be implemented by them, it also provides a systematic way of thinking about the issues.

Second, creating new acquisition techniques for web-based evidence requires novel methods and innovative thinking. The nature of web-based evidence disallows the use of existing techniques without modification, yet the principles and standards (such as the Rules of Evidence) must still persist.

Third, structured formats assist in storing evidence and analysis results such that tools can be chained together and interoperate. Publishing the standards for these formats encourages developers to adopt them and researchers to improve on them.

Finally, it is possible to reduce the size of an evidence set while still allowing for meaningful analyses.

## Chapter 3

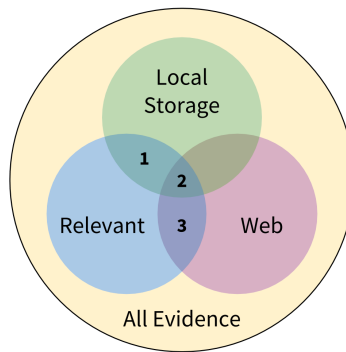
# CHALLENGES, OPPORTUNITIES, AND A FRAMEWORK FOR WEB ENVIRONMENT FORENSICS

### 3.1 Introduction

The Web transformed how people around the globe interact with each other, conduct business, access information, and enjoy entertainment, among many other activities. Web environments, including all types of web services and other cloud services with web interfaces, now offer mature feature sets that, just a few years ago, could only have been provided by software running on a desktop computer. As such, the Web provides users with new levels of convenience and accessibility, which has led to one phenomenon that critically impacts digital forensic examiners—people are storing less and less data on their local devices in favor of web-based solutions.

Today's digital forensic techniques are very good at answering questions about the evidence stored *on* devices involved in an incident, however—up until now—the techniques have struggled to breach this boundary into the set of data stored *remotely* on the Web. As the diagram in Figure 3.1 illustrates, if forensic examiners depend only on the storage of the devices they seize as evidence, they will miss some relevant and potentially vital information. Areas 1 and 2 represent what digital forensic examiners typically spend their time searching for: relevant artifacts that reside on the local devices' storage originating (1) from programs and services running on the local machine and (2) from the Web, such as files cached by a web browser or email client. As area 3 depicts, the suspect may have stored data on the Web that is relevant to the investigation but cannot be retrieved directly

Figure 3.1: Relation between evidence types acquired during an investigation.



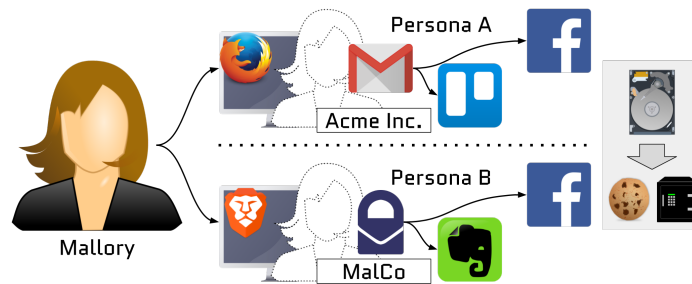
from the devices seized. Everything outside the top and right circles represents non-digital evidence.

The rise of a new generation of cybercrime represents a completely different era that presents issues that traditional forensic techniques cannot solve. As part of this chapter, I identify five unique challenges web environments pose to digital forensics namely (which I describe and elaborate on throughout this chapter):

- C0 Complying with the Rule of Completeness
- C1 Associating a suspect with online personas
- C2 Gaining access to the evidence stored online
- C3 Giving the evidence relevant context in terms of content and time
- C4 Integrating tools to perform advanced analyses

Currently, *forensic examiners have no strategy or framework* to guide them in their analysis of cases involving devices and users where data is dispersed on the device and on the Web. Therefore, in this chapter I propose a framework designed for conducting analyses in web environments that addresses challenges C0–C4 and can be integrated into existing processes. Using this framework, forensic examiners will be able to obtain and

Figure 3.2: Mallory uses two online personas to obscure her illicit activities. Her hard drive will contain residual evidence of her online accounts in the form of browser cookies, password vaults, etc.



give relevant context to previously unknown data while also adhering to the rules of evidence.

### 3.2 Motivating Scenario

Consider the following fictional scenario, illustrated by Figure 3.2, which demonstrates the motivation for this chapter.

Mallory, an employee at Acme Inc., is using company resources to start a new business, MalCo. This action is a violation of her employer's waste, fraud, and abuse policies and the non-compete agreement she signed. She knows that eventually her computer may be analyzed by the IT department for evidence of her actions to provide grounds for Acme claiming ownership of MalCo once it launches. So, she decides to use a number of web services anytime she works on her new company to help minimize the evidence left on her computer.

Mallory conscientiously segregates her web browsing between things she does for Acme and for MalCo and even uses different web browsers; this segregation effectively creates different personas, which I will refer to as Persona A (Acme) and Persona B (MalCo), respectively.

When using Persona A, Mallory uses Firefox as her web browser. Because Acme Inc. uses Google's G Suite, her work email is essentially a Gmail address. She used this email to create accounts on Trello and Facebook. Mallory's team at Acme uses Trello to coordinate with each other and Facebook to engage with their clients.

When assuming Persona B to work on MalCo, Mallory is careful to only ever use the Brave web browser. For email, she created an account with Proton Mail for its extra encryption features. She used her Proton Mail address to create accounts on Evernote and Facebook. In Evernote, Mallory stores all of her business plans, client lists, and product information. With her Persona B Facebook, Mallory secretly contacts Acme Inc. customers to gauge their interest in switching to MalCo once it launches.

### 3.3 Unique Forensic Challenges in Web Environments

I identified five areas in which web environments pose unique challenges to digital forensic examiners. For convenience in referring back to these challenges, I have numbered them C0–C4.

#### 3.3.0 C0: Rule of Completeness

The Rules of Evidence give protection to both victims and suspects by helping ensure that the conclusions examiners draw from the evidence are accurate. The Completeness rule states that evidence must tell a complete narrative of a set of particular circumstances, setting the context for the events being examined so as to avoid “any confusion or wrongful impression” [118]. Under this rule, if an adverse party feels evidence lacks completeness,

they may require introduction of additional evidence “to be considered contemporaneously with the [evidence] originally introduced” [118].

The Rule of Completeness relates closely to the other challenges I discuss in this section, which is why I numbered it C0<sup>18</sup>. By attempting to associate a suspect with an online persona (C1), an examiner is increasing the completeness of the evidence. The same is true when an examiner gains access to evidence stored on the Web (C2).

In a way, the Rule of Completeness is the counterpart to Relevant Context (C3). By properly giving evidence context, examiners can ensure that the evidence provides the requisite “complete narrative.” However, in the process of giving the evidence context, examiners must take care not to omit evidence that would prevent confusion or wrongful impression.

### 3.3.1 C1: Associating a Suspect with Online Personas

When an individual creates an account with an online service provider, they create a new persona that, to some degree, represents who they are in the real world. The degree to which the persona accurately represents the account owner depends on a number of factors. For instance, some attributes captured by the service provider may not correlate with any real-world attributes, such as a customer identification number. Also, users may provide fraudulent information about themselves, or they may create parody, prank, evil-twin, shill, bot, or Sybil accounts.

The challenge examiners face is to associate these personas with an individual so as to assign responsibility to an individual for the actions known to have been performed

---

<sup>18</sup>For details on how I address the other rules of evidence, please see Section 3.6.



by the persona. If they are unable to establish this link, the perpetrator will effectively remain anonymous.

In addition to it being difficult to make an explicit link to an individual, it is also difficult to discover these personas in the first place if the forensic examiner only (at least initially) has access to data from devices found in the suspect's possession. This difficulty is because web environments tend to store very little (if any) data on the user's local device that may reveal these personas.

In Mallory's case, the data left behind revealing her personas includes browser cookies and her password vault. After determining which online services these credentials belong to, the examiner still must find a way to show that it was actually Mallory that created and used those accounts. Intuitively, this is more difficult if many users share the same computer.

### 3.3.2 C2: Evidence Access

Many times in a forensic investigation, the examiner will determine that some service provider is likely to have additional data created or stored by the suspect. In this case, the typical course of action is to subpoena the service provider for the data. However, this is an option only available to law enforcement and government agencies. If an investigation does not merit being pursued as a civil or criminal case, corporate and non-government entities are essentially left to gather what evidence they can on their own.

While many web services provide APIs for programs to access data, there does not exist a single, unified API to access the data from all web services, nor should such an API exist. Accordingly, with all web services being so disparate, a unique acquisition approach must be developed for each web service. And because there is no guarantee that APIs will

remain constant, it may be necessary to revise an approach when the service or its APIs change.

### 3.3.3 C3: Relevant Context

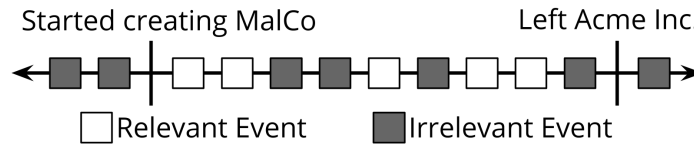
Digital forensic examiners' objective is to distill evidence down to those artifacts that tell the story of what happened during an incident by increasing the relevance of the artifacts' context. I observe that context comes in two forms, both of which are critical to any investigation.

The first form is *thematic* context, which effectively places labels on artifacts that indicate its subject or theme. Using these labels, examiners can filter out those that do not concern the investigation, thereby giving focus to those that help prove or disprove the suspect's involvement in the incident. A commonly-used tool for thematic context is a keyword search, in which the examiner enters some keywords and the tool searches the content of all the files and returns those containing the provided text or related text, if the tool uses a fuzzy-matching algorithm.

The second form of context is *temporal* context, or placing an artifact in a timeline to indicate its chronological ordering relative to events in the non-digital world as well as other digital artifacts. Creating such a timeline provides an examiner with the perspective of what happened when, which may be critical in the outcome of the investigation.

Although these forms of context have always been important objectives for digital forensic examiners, web environments make it much more difficult to attain them because the Web allows users to generate artifacts and events at a higher pace than more traditional evidence. Furthermore, the Web consists of diverse types of data, such as multimedia,

Figure 3.3: Timeline of Mallory’s actions. *Thematic* context is separating the relevant from the irrelevant. *Temporal* context is ordering the events chronologically and setting a window of interest.



many of which require either human effort or very sophisticated software to assign subjects to the data before its thematic context can be determined.

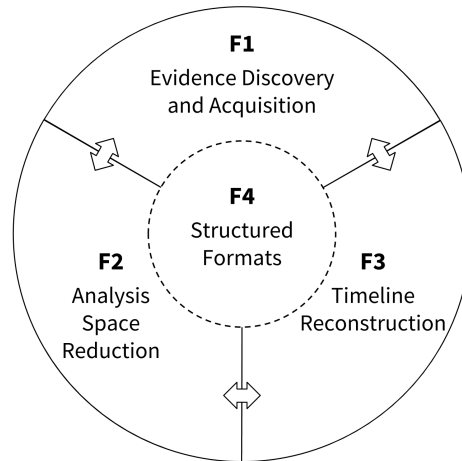
Figure 3.3 depicts Mallory’s actions to create MalCo. Identifying the relevant events from the irrelevant ones provides thematic context. Examiners provide temporal context to the events by placing them in chronological order and creating a window of interest by determining the points when Mallory engaged in her inappropriate behavior.

### 3.3.4 C4: Tool Integration

Digital forensics researchers have long decried the shortcomings of the two models of tools available to examiners. On one hand, one-off tools are usually designed to perform very specific actions or analyses, and may not have very good technical support or may be outdated, poorly documented, or have other issues. On the other hand, monolithic tools seek to cover as many use cases as possible in a single package. While these products often enjoy the benefits of being commercial software<sup>19</sup>, they have an obvious interest in keeping their tools and the tools’ output proprietary to maintain a competitive edge.

<sup>19</sup>As an open-source project that continues to be actively developed after nearly fifteen years, the Sleuth Kit (and its graphical user interface, Autopsy) is a notable exception to this generalization (see <https://www.sleuthkit.org/sleuthkit/history.php>).

Figure 3.4: My framework for web forensics.



Also, the design of monolithic tools often does not facilitate scripting, automation, or importing from and exporting to other tools [114, 17, 50].

Considering the complexity of the situation as I have discussed it thus far, it is unreasonable to assume a single tool or approach will be able to provide solutions to all the challenges to web environment forensics. With this constraint in mind, it is clear that forensic tools designed to properly accommodate evidence from web environments will need to overcome the status quo and integrate with other tools to accomplish their work.

### 3.4 A Framework for Web Environment Forensics

To help address the challenges unique to web environment forensics, I developed a framework that can guide a digital forensic examiner that is handling a web environment case. My framework is depicted in Figure 3.4, and it incorporates four components designed to directly address the challenges discussed in Section 3.3 with the expected outcome of providing examiners with (1) previously unknown data and (2) relevant context of the

Table 3.5: Alignment of the framework components with the challenges discussed in Section 3.3. Key: ●: Addresses. ○: Does not address.

	F1	F2	F3	F4
C0: Rule of Completeness	●	●	●	○
C1: Associating Personas	●	○	○	○
C2: Evidence Access	●	○	○	○
C3: Relevant Context	○	●	●	○
C4: Tool Integration	○	○	○	●

incident. Table 3.5 illustrates which challenges each component addresses. The four components of the framework are:

F1 Evidence Discovery and Acquisition

F2 Analysis Space Reduction

F3 Timeline Reconstruction

F4 Structured Formats

The F1–F3 components interrelate with each other non-sequentially, meaning that the sequence in which an examiner uses each of the components is not dictated by the components themselves, but rather by the flow of the investigation and the examiner’s needs. As such, when an examiner completes one component, she may subsequently have need of one, both, or neither of the other two components. This non-sequential relationship also allows examiners to incorporate the components into existing workflows as necessary and as befits each investigation.

For example, after acquiring new evidence from the Web, it may be necessary to narrow the focus of the examination, which in turn may tell the examiner where to find new, previously inaccessible evidence to acquire, thus creating a F1→F2→F1 sequence. Similarly, an examiner may use acquired data to reconstruct a timeline of events, which may then be most useful after having been reduced to those periods that indicate heightened

activity. With a focus on these events, it may then become necessary to create a timeline of even finer granularity or to acquire new evidence specific to the period of interest. The sequence of these steps would be  $F1 \rightarrow F3 \rightarrow F2 \rightarrow F3$ .

The **F4** component relates to the other three components in a special way that I will discuss in Subsection 3.4.4.

I will now introduce each of the four components of the framework by describing the objectives of the component, giving a formal description of the component, describing what the process of fulfilling that component would look like for an examiner, discussing the research challenges currently impeding progress on the component, introducing approaches from related work, and then identifying the key research opportunities for the component.

To facilitate a formal discussion on each of the framework components, I give the following definitions:

**Definition 3.6** (Evidence).  $E = \{a_0, a_1, \dots, a_n\}$  is the set of forensic artifacts to which an examiner has access.

**Definition 3.7** (Labels).  $L = \{l_0, l_1, \dots, l_n\}$  is the set of all possible labels, including keywords, subjects of documents, and so forth.

**Definition 3.8** (Labels of Interest). The labels of interest  $L_I \subset L$  is the set of labels relevant to the investigation as chosen by a forensic examiner.

**Definition 3.9** (Evidence Labels).  $l : E \rightarrow \{L' \mid L' \subseteq L\}$  is a mapping between evidence artifacts and a subset of all labels.

**Definition 3.10** (Thematically Relevant Set).  $R_L = \{a \in E \mid l(a) \subseteq L_I\}$  is the set of evidence artifacts that have at least one evidence label contained in the set of the labels of interest.

**Definition 3.11** (Time).  $T_{all}$  is the ordered set of all time values.

**Definition 3.12** (Times of Interest).  $T_I = \{\{t_i, t_j\} \mid t_i \in T_{all}, t_j \in T_{all}, t_i < t_j\}$  is a set of ordered pairs of time values such that the first time value precedes (is less than) the second time value. The two time values in each ordered pair represent the start and end of a window of time, respectively.

**Definition 3.13** (Time Attributes).  $t : E \rightarrow \{T' \mid T' \subset T_{all}\}$ , where  $|T'| \geq 0$ , is a mapping between evidence artifacts and time values.

**Definition 3.14** (Temporally Relevant Set).  $R_T = \{a \in E \mid \exists t_i \in t(a) \exists \{x, y\} \in T_I \mid x \leq t_i \leq y\}$  is the set of evidence artifacts with at least one time attribute that falls between the start and end times of a time of interest.

**Definition 3.15** (Reduced Evidence).  $E_R = R_T \cup R_L \subseteq E$ , is the set of evidence artifacts that are either thematically or temporally relevant. It may be that  $E_R = E$  if all artifacts in  $E$  are relevant. A more strict version of the reduced set of evidence artifacts  $E'_R = R_T \cap R_L$  only includes the artifacts that are both temporally *and* thematically relevant.

**Definition 3.16** (Guilt).  $g : E \rightarrow \{x \in \mathbb{R} \mid -1 \leq x \leq 1\}$  maps evidence artifacts to a real number in the range  $[-1, 1]$  such that -1 indicates the artifact is absolutely exculpatory<sup>20</sup>, 1 indicates the artifact is absolutely inculpatory, and 0 indicates the artifact either has not yet had its guilt value determined or is ambiguous as to whether it is exculpatory or inculpatory. The examiner must be the one to determine the guilt function's value for a particular artifact.

---

<sup>20</sup>Exculpatory evidence suggests innocence, inculpatory evidence suggests guilt.

**Definition 3.17** (Inculpatory Artifacts).  $A_{in} = \{a_{in} \in E_R \mid g(a_{in}) > 0\}$  consists of those artifacts from  $E_R$  that have a *positive* guilt value. Artifacts in  $A_{in}$  must come from  $E_R$  (and not  $E_R \cup E$ ) because they must be *relevant* in order to be inculpatory.

**Definition 3.18** (Exculpatory Artifacts).  $A_{ex} = \{a_{ex} \in E_R \mid g(a_{ex}) < 0\}$  consists of those artifacts from  $E_R$  that have a *negative* guilt value. Artifacts in  $A_{ex}$  must come from  $E_R$  (and not  $E_R \cup E$ ) because they must be *relevant* in order to be exculpatory.

### 3.4.1 F1: Evidence Discovery and Acquisition

The objective of framework component **F1** is to overcome the challenges of (1) establishing associations between a suspect and online personas (**C1**), and (2) gaining access to the evidence stored on web services by those personas (**C2**). It is important to note that in this component I do not attempt to discern whether or not the data is *relevant* to the investigation. The focus here is to discover and acquire web-based evidence created by the suspect but not stored on the devices seized—evidence that would not otherwise be accessible to the examiner. Of course, component **F1** also helps examiners comply with the Rule of Completeness (**C0**).

Because **F1** gives the examiner access to new evidence from the web,  $E_{web}$ , the evidence set is augmented to become  $E' = E \cup E_{web}$ , such that  $|E'| > |E|$ .

#### 3.4.1.1 F1: Examiner's Process

**Discovery** To begin the process of discovering previously inaccessible evidence, examiners will need to analyze the storage of the devices in their custody for clues that connect the user to evidence stored on the Web. Examples of such clues include web session cookies,



authentication credentials, and program-specific artifacts such as those collected by the community and posted to ForensicArtifacts.com. Finding and identifying these artifacts will require a sound understanding of their critical characteristics and, in some cases, a database of artifact samples to facilitate efficient comparison.

For certain formats of authentication artifacts, the process of discovering the artifacts can be automated, relieving human examiners from attempting manual discovery, which does not scale well. However, even with automated discovery, it may still be necessary for an examiner to manually determine the service to which the credential gives access. For example, if a user (for some reason) stores their username and password in a text file, even if such an artifact happened to have a structure such that a program could accurately extract the credentials, it may require a human operator to consider the context of the file (name of the directory or file) to derive the corresponding service.

**Acquisition** After the examiner discovers an authentication artifact and identifies the corresponding service, it is necessary to devise a means to acquire data from the service. Considering the variety of web services, an approach for acquiring data from one source may not directly apply to other sources. Examiners and tool developers will need to understand what principles are transferable and design their workflow or tools to be as general-purpose as possible [100]. They should also leverage structured storage formats (F4) for the acquired evidence.

#### 3.4.1.2 F1: Challenges

**Discovery** The task of discovering and acquiring evidence from the Web is challenging for a few reasons. First, the potential volume of data a suspect has stored on the Web is

nearly unlimited. Not only does this present a challenge in terms of storage requirements for holding the evidence, but also makes the task of analyzing it more complex.

Second, the boundaries of this data set are nebulous in a geographical sense as well as in terms of which service contains the data. By contrast, the boundaries of a hard drive's storage (e.g., total number of sectors) are well-defined and an examiner can discover them easily by means of some simple analyses of the disk media itself. However, it is difficult for examiners to find a starting point for discovering evidence in web environments. As a counter-example, examiners know that the best place to start analyzing a desktop computer is its hard drives. The best analog for evidence in the Web is for examiners to start with what they have access to, which, in most instances, means a device with storage capabilities, such as a smart phone, computer, laptop, GPS guidance system, or DVR. However, it is also possible that the physical possessions of a suspect contain no trace of where their data is stored on the Web.

A third challenge is derived from the likelihood that the suspect has created many accounts on the Web, both by having an account with several web services and by having multiple accounts with the same service. While it is possible that all of a user's accounts are active and accessible, it is more likely that some subset of their accounts have been suspended or deactivated due to a period of inactivity, intentional lockout, reaching the maximum number of unsuccessful authentication attempts, or other circumstances. Furthermore, with so many web services and accounts, it is not uncommon for any of us to forget we created an account with a particular web service months or years after the fact. While it is less likely that the data from an inactive or forgotten account will play a critical role in an investigation, this illustrates the challenge of *discovering* all the data created by a user on the Web. Having a large number of accounts also makes it more

difficult to evaluate the relevance of each, although this challenge relates more directly to component **F2**.

**Acquisition** The second part of this component, acquiring the data, presents its own set of challenges. First, the data stored by these web services may be changing on a near-continual basis. This is particularly true when the data is being automatically generated on behalf of a user. With the continued proliferation of Internet-of-Things (IoT) devices, such as home automation systems<sup>21</sup>, examiners are likely to see an ever-increasing amount of automatically generated data for the foreseeable future. Such data is not dissimilar to evidence requiring live acquisition, having a more fragile composition and requiring special care and handling.

The other significant challenge to acquiring data from the service provider is finding a way to access the data to perform the actual acquisition, as discussed in Subsection 3.3.2. With no unified API to use when acquiring data from all web services, considerable manual effort may be required for an examiner to understand and know how to interface with each service.

#### 3.4.1.3 F1: Related Approaches

There are very few approaches currently available to examiners to complete this component of the framework, and even fewer that provide an automated approach [96]. Dykstra and Sherman evaluated how well some existing forensic acquisition tools performed on an Amazon EC2 instance [41]. In general, the tools did well considering they were not designed specifically for that kind of evidence acquisition. However, this approach only

---

<sup>21</sup>For example, see <https://home-assistant.io/>.

works for instances under the control of the examiner at the guest OS, virtualization, and host OS layers, not at the web application layer.

#### 3.4.1.4 F1: Research Opportunities

Artifact repositories such as ForensicArtifacts.com and ForensicsWiki.org are valuable resources for forensic examiners. However, one critical shortcoming they share is that the information they contain is only suitable for human consumption, meaning it is not currently possible for automated tools to leverage the data hosted on these sites. Future research should look to convert this information into a structured format (**F4**) that gives the data the necessary semantics to facilitate automation.

Although each web service has its own set of APIs, it may be possible, through a rigorous study of a wide range of services, to create an abstraction of the various calls, and thus create a more generic and reusable method of facilitating the acquisition process.

#### 3.4.2 F2: Analysis Space Reduction

Not every evidence artifact is equally important to the purposes of the investigation, and it would greatly benefit examiners to have help identifying and focusing on those that are the most relevant (**C3**). With irrelevant data filtered from examiners' view by a preliminary analysis, they can make the best use of their time and effort to complete their analyses more quickly—this is the motivation and objective of component **F2**. Formally, **F2** creates  $R_L$ , the set of thematically relevant artifacts.

Although component **F2** removes evidence from view, this process helps examiners

comply with the Rule of Completeness (C0) because the narrative of the evidence is unfettered by irrelevant artifacts.

While analysis space reduction through improved thematic context can benefit the forensic analysis of digital evidence of all types, due to the virtually limitless storage capacity, analyses of evidence from web environments stand to gain particular performance improvements from the incorporation of component **F2**.

#### 3.4.2.1 **F2**: Examiner's Process

There are two general approaches to reducing the analysis space of evidence: classification and identification. The first approach requires classifying the data in the evidence, then indicating the types of data that either are or are not of interest. Classification is the more common form of thematic context and aligns well with the example I provided in Subsection 3.3.3. Examiners may also wish to classify or separate artifacts according to when they were created, modified, or last accessed on the system, in which case techniques from **F3** would be helpful.

The second approach to reducing the analysis space is to identify *what* the evidence is, which is particularly helpful when the evidence is encrypted or otherwise unreadable directly from the device's storage. When using this approach, the primary task is more about identifying the data rather than classifying it or determining its relevance. However, identification is still a method for providing *thematic* context because it allows the examiner to determine if the data is relevant to the investigation or not. The main difference is that instead of identifying the subject of the data directly, the identity of the data will allow the examiner to determine the subject from the data's identity.

One potential method to reduce the analysis space with the identification approach

would be to use what information is available about the data (such as metadata) to eliminate what the data *cannot* be, thereby incrementally approaching an identification via true negatives. Intuitively, this approach only applies when the set of possibilities is finite (i.e., it does not apply to arbitrary files created by a user).

Because the ultimate goal of **F2** is to end up with less (but more relevant) evidence than the original data set, **F2** tools may export their results in the same format as that of the data ingested by the tools. This provides the benefit that **F2** tools can be incorporated into existing workflows without requiring changes to how other tools process data. However, even in cases where the reduction of the analysis space yields data of a different type than the input (e.g., the identification approach), tools should still use structured formats for the reasons I discuss in Subsection 3.4.4.

### 3.4.2.2 **F2**: Challenges

Reducing the analysis space in an automated manner requires careful consideration of a number of factors. First, the implication here is that an algorithm will be given responsibility to understand the nature of the evidence and make a judgment (albeit a preliminary one) concerning its bearing on a person's guilt or innocence. While any false positives<sup>22</sup> result in an analysis space that is sub-optimally reduced, a false negative obscures a relevant artifact from the examiner's view, which could alter the outcome of the investigation. The latter is an unacceptable outcome.

Exculpatory evidence is particularly sensitive to false negatives given that it is inherently more difficult to identify than inculpatory evidence because, by definition, it tends to clear from a charge of guilt. In other words, evidence that exonerates a suspect is more difficult

---

<sup>22</sup>In this context, a false positive is determining an artifact to be relevant when it is not.

to interpret in an automated fashion because it may not directly relate to the incident under investigation, may require correlation with evidence from other sources, or it may be the absence of evidence that is of significance.

Aside from the challenges related to the accuracy of the tools that reduce the examiner's analysis space, it is also important to consider that the volume of data stored by the suspect on the Web may be very large, and even with an accurate reduction to relevant data, the size of the resulting data set may still be quite large and time-consuming for a human examiner to carefully analyze.

#### 3.4.2.3 F2: Related Approaches

Researchers have developed many different data classification techniques that may apply, such as using object recognition of photos, identifying the topics of a document, etc. [80]. Another classification example is the National Software Reference Library (NSRL) [86], which is a list of known files from benign programs and operating systems. By leveraging the NSRL as a classification for evidence that is *not* of interest, examiners can reduce the analysis space of the examination by eliminating from consideration the files they know were not created by the user and do not pertain to the investigation.

#### 3.4.2.4 F2: Research Opportunities

Perhaps the most important potential research area in reducing examiners' analysis space is that of developing methods to minimize false negatives without risking false positives. Such an undertaking would undoubtedly benefit from any related advances in the fields of natural language processing (NLP), computer vision (CV), and other artificial

intelligence domains. The better a tool can understand the meaning of the digital evidence, the more likely it will be able to accurately minimize false negatives.

Just as people regularly use more than one device on a typical day, the evidence they leave behind is not contained on a single device. Examiners would greatly benefit from improved cross-analytic techniques that combine the evidence from multiple sources to allow examiners to correlate artifacts and identify themes that otherwise would have been obscured had each source been analyzed individually.

Researchers have already shown that it is possible to identify encrypted data without decrypting it [99, 79]. Although such approaches may not be well-suited to every investigation with encrypted data, the fact that it is possible under the proper circumstances demonstrates there are additional research opportunities in this area.

### 3.4.3 F3: Timeline Reconstruction

The objective of framework component **F3** is to improve the temporal context of the evidence by reconstructing the timeline of what happened, giving the artifacts a chronological ordering relative to other events. This timeline, in turn, helps tell a more complete story of the user's activities and the incident under investigation. This additional information also contributes to a more complete narrative, helping to satisfy the Rule of Completeness (**C0**). Formally, **F3** creates  $R_T$ , the set of temporally relevant artifacts. Chronological ordering is a partial ordering of the set of time values  $\{x \mid x \in t(a) \forall a \in R_T\}$ .



#### 3.4.3.1 F3: Examiner's Process

For an examiner to reconstruct the timeline from web environment data, the first step is to collect all available information from the evidence that records values of time in connection with other data. Collecting this information will require the examiner to use F1 tools and methods to discover and acquire the data. Accordingly, all the challenges and approaches I discussed in Subsection 3.4.1 apply here as well.

All of the collected timeline information should be combined into a single archive or database, which will require the use of a unified storage format (F4) that can accommodate the various fields and types of data that the original information included. However, because the information originates from several sources, the compiled timeline may include entries that are not relevant to the investigation. In this case, it would be beneficial to leverage F2 approaches to remove entries that do not provide any meaningful or relevant information and thereby improve the thematic context of the evidence. Similarly, if a particular time frame is of significance in the investigation, removing events that fall outside that window would improve the evidence's temporal context.

With the timeline information compiled and filtered, it is necessary to establish relationships between entries. If everything is ordered chronologically, establishing the sequence of events is a simple matter. Other types of relations that may prove insightful include correlation of events (e.g., event *a* always *precedes* events *b* and *c*), and clustering (e.g., event *x* always happens *close to* the time that events *y* and *z* occur). Finally, examiners may leverage existing analysis and visualization tools on the timeline data, assuming of course that they are compatible with the chosen storage format.

### 3.4.3.2 F3: Challenges

Although analyzing traditional digital evidence for timeline information is well-researched [103, 14, 22, 84], it presently remains unclear if the currently-available approaches can be directly applied to web environments due to the inherent differences. For example, timeline reconstruction typically incorporates file metadata as a source of time data, and from previous work it is clear that web service providers regularly store custom metadata fields [100]. These metadata fields are typically a superset of the well-known modification, access, and creation (MAC) times, and include values such as cryptographic hashes, email addresses of users with access to the file, revision history, and so forth. Clearly, these fields are valuable to examiners, but cannot be accommodated by current timeline tools.

For forensic tool developers to incorporate these fields into their tools, they would need to first overcome some additional challenges. Considering that each web service provider may use a distinct set of metadata fields, it will be critical to determine a method of supporting different sets of fields. One approach would be to create a structured storage format (F4) with the flexibility to store *arbitrary* metadata fields. Another approach would be to unify the sets of metadata fields through an abstract hierarchy, such as an ontology, so the semantics of the metadata are preserved when combining them with fields from other sources.

Another challenge to incorporating metadata from web environments into a timeline reconstruction is that the variety of log types and formats continues to grow as new types of devices emerge, such as IoT devices. Many such devices have the capability to perform actions on behalf of their users and may interface with arbitrary services on the

Web through the addition of user-created “skills” or apps. Forensic researchers have only recently begun to evaluate such devices for the forensic data they store [64, 98].

Finally, as with any attempt to reconcile time information from different sources, it is critical to properly handle any differences in timezones. While it is a common practice for web services to store everything in UTC, tools and examiners cannot take for granted that this will always be the case. And as has been shown in other projects<sup>23</sup>, correlating data from different timezones can be a complicated task.

### 3.4.3.3 F3: Related Approaches

As I mentioned previously, it is uncertain if any existing approaches to timeline reconstruction will be helpful to examiners with regard to evidence from web environments, and the literature does not yet contain any approaches designed specifically for this purpose. However, because the first step in timeline reconstruction is to collect data with time information, some approaches for cloud log forensics may provide good starting points.

Marty [81] presents a logging framework for cloud applications. Marty’s approach requires the developer of an application to be responsible for its implementation and provides more of a set of guidelines for what to log and when. As such, it may complement other efforts to collect logs, but it may not be directly applicable to web environments in general.

---

<sup>23</sup>For example, see the Maya project: <https://github.com/kennethreitz/maya>.

#### 3.4.3.4 F3: Research Opportunities

Although visualization of timeline data seems to be an area of research with a considerable amount of activity [105, 95, 22, 108], there will always be new and better ways to visualize this data to help examiners understand it. For example, virtual and augmented reality technologies may help examiners better understand data by presenting truly three-dimensional views of the timeline.

One aspect of timeline reconstruction that seems to remain unexplored is that of standardizing the storage format (F4) for the data representing the timeline. Separating the data from the tool would facilitate more objective comparisons of visualization tools and allow examiners to change the tool without having to start the timeline reconstruction process over from the beginning.

When reconstructing a timeline from multiple sources, there is a chance that some subset of the time data will correspond to an unspecified timezone. A worthwhile research area would be to develop an approach to elegantly resolve such ambiguities.

#### 3.4.4 F4: Structured Formats

The purpose of using structured formats is to provide a way of storing information that is not specific to one tool or process to facilitate tool interoperability and integration (C4). Structured formats also allow researchers to compare the output from similar tools to measure their consistency and accuracy, which are key measurements of forensic tools to determine their suitability to be used on evidence.

I positioned F4 at the center of the framework because components F1–F3 all leverage some kind of storage format, even if the format itself is not part of the component. For

example, after discovering and acquiring new evidence, a tool must store the evidence somehow; and, of course, the format in which the evidence is stored should have a generic yet well-defined structure. In this case, the structure used by the acquisition tool does not change how it performs its principal task, but is a peripheral aspect of its operation.

Structured formats are critical to the proper function of the framework. For a tool providing one component to communicate with a tool providing a different component, they must have a way to transmit data that they can both understand. Defining the data's *structure* is what makes this possible.

#### 3.4.4.1 F4: Examiner's Process

Because the purpose of structured formats is to facilitate tool interoperability and integration, examiners should not have to work directly with them, save on rare occasions.

#### 3.4.4.2 F4: Challenges

To realize the previously-mentioned benefits, a structured format must satisfy three conditions: (1) it must be a precise representation of the original evidence with no loss of information during conversion<sup>24</sup>, (2) there must be a way to verify that data conforms to the format's specifications, which requires that (3) its specifications must be published and accessible to tool developers.

While many different storage formats exist, none of them is perfect or can cover every use case. Similar to many other software engineering projects, format designers are

---

<sup>24</sup>The one exception to this is when the format is designed to store a deliberately reduced version of the evidence as the result of component F2.

constantly faced with the need to compromise or make trade offs in their work, and this in turn makes them less suitable for certain circumstances. For example, some storage formats fully accommodate file metadata fields used by Windows filesystems, but not Unix filesystems. This illustrates how difficult it can be to incorporate the correct level of detail in the format specification [18]. In this regard, open-source formats have an advantage in that the community can help make improvements or suggest ways to minimize the negative effects of trade offs.

It is critical to the framework and to the principle of composability that analysis tools use structured formats to store their analysis results in addition to the evidence itself. This is the only way to allow the kind of advanced analyses necessary to address the challenges of large evidence data sets, such as those originating from the Web.

#### 3.4.4.3 F4: Related Approaches

Many different structured formats have been proposed specifically for digital forensics over the years, most of which are designed for storing hard disk images [37, 113]. Here is a brief summary of some of these formats.

The Advanced Forensic Format (AFF) [36] framework provides a flexible way to store forensic evidence of many types. As Casey et al. explained in the paper introducing AFF4<sup>25</sup>, “Unlike [the Expert Witness Forensic (EWF) file format], AFF [employs] a system to store arbitrary name/value pairs for metadata, using the same system for both user-specified metadata and for system metadata, such as sector size and device serial number” [36].

The Cyber Observable eXpression (CybOX™) [83] language was designed “for speci-

---

<sup>25</sup>The versioning of AFF is a little unusual because the initial release included the format standard and a set of tools. Newer versions of the tools were released before the format was updated; so to avoid confusion with the tools, the second release of the format was named AFF4 [36].

fyng, capturing, characterizing, or communicating ... cyber observables”<sup>26</sup>. As Casey et al. showed with their work on the Digital Forensic Analysis eXpression (DFAX) [18], CybOX can be extended to represent additional data related to forensic investigations. The project has since been folded into version 2.0 of the Structured Threat Information Expression (STIX™) specification [9].

The Cyber-investigation Analysis Standard Expression (CASE) [20], which is a profile of the Unified Cybersecurity Ontology (UCO) [107], is a structured format that is an evolution of CybOX and DFAX. CASE describes the relationships between digital evidence artifacts, and because it is an ontology, it facilitates reasoning. Because CASE is extensible, it is a strong candidate for representing evidence from web environments.

Digital Forensics XML (DFXML) [50] is designed to store the metadata of all files on a disk with the idea being that a concise representation of simply the metadata will allow examiners to still perform some analyses related to the evidence while facilitating remote collaboration by virtue of DFXML’s smaller size. Because it is written in XML, other schemas can extend DFXML to suit various scenarios.

Email Forensics XML (EFXML) [96] was designed to store email evidence in a manner similar to DFXML. Rather than storing the entire email, EFXML only stores the metadata (i.e., the headers) of all the emails in a data set. EFXML was designed to accommodate email evidence originating from traditional devices as well as from the Web.

The Matching Extension Ranking List (MERL) [79], is more specialized than the other formats I have discussed. Rather than storing evidence, MERL files store the analysis results from identifying extensions installed on an encrypted web thin client, such as a Chromebook. As such, it does not have the flexibility to store any other kinds of data.

---

<sup>26</sup>See <http://cyboxproject.github.io/about/>.

However, unlike many of the other formats, it has been created for use with web-based evidence.

Of course, none of these formats were created to capture all types of evidence from the Web. However, some, such as AFF4, may provide enough flexibility to accommodate storing arbitrary forms of web-based evidence.

#### 3.4.4.4 F4: Research Opportunities

In 2004, Buchholz and Spafford evaluated the role of filesystem metadata in digital forensics and proposed new metadata fields that would assist in forensic examinations [15]. It would be of great use to forensic researchers for a similar study to be conducted for web environment evidence. As I have discussed, web services will each have their own custom metadata that serve the organization, but it is important to understand how these differ from service to service, which ones have value to examiners, how to unify (or at least reconcile the semantics of) the various fields, and what fields would be useful if a forensic examiner was able to make suggestions.

### 3.5 Related Work

In previous sections, I discussed many approaches related to the individual components of my framework. Now I discuss some approaches that relate to the framework as a whole.

Paglierani et al. developed a framework for automatically discovering, identifying, and reusing credentials for web email to facilitate acquisition of the emails as evidence [96]. Although their approach directly addresses the objectives of discovering and acquiring web



evidence (F1) and provides a concise, structured format for storing email evidence (F4), their framework is too tailored to web email to apply it to web environments in general.

Ruan et al. [101] enumerated many forensic challenges and opportunities inherent to cloud computing in general in a manner similar to what I have done in this work for web environments. However, Ruan et al. do not provide any kind of guide for examiners on how to begin using the cloud for forensic examinations, they only highlight the potential benefits of doing so. Additionally, although much of today's Web is built on cloud computing, the two are not synonymous. As such, many of the *challenges* they list, such as data collection difficulties, services depending on other services, and blurred jurisdictions, apply to web environments, but the *opportunities*, such as providing forensics as a service, apply mainly to *implementing* forensic services in the cloud.

Birk and Wegener [11] provide some recommendations for forensics in clouds, separated by the type of cloud service provider (CSP), Infrastructure as a Service (IaaS), Platform as a Service (PaaS), or Software as a Service (SaaS). Of these the most applicable to web environments is, of course, SaaS. However, Birk and Wegener place the responsibility for providing a means of forensic acquisition on the shoulders of the CSP. My framework helps guide examiners to know what they can accomplish even with an uncooperative CSP.

### 3.6 Discussion

Throughout this chapter, I have discussed how my framework addresses the Rule of Completeness. My inclusion of the Rule of Completeness as one of the unique challenges to web environment forensics (Section 3.3) was motivated by two things: (1) how closely it relates to the other unique challenges and (2) the fact that the overall emphasis of the

framework is aligned with the objectives of the Rule of Completeness. However, it is also important to discuss how my framework addresses the three other key Rules of Evidence. For full definitions of each of these rules, please refer to the Glossary.

*Admissibility* requires that the evidence be handled in accordance with applicable laws. Relevance is one of the criteria for evidence being admissible, and the framework helps give both thematic relevance (F2) and temporal relevance (F3) to the evidence. Furthermore, certain structured formats (F4) include provenance and chain of custody information, both of which also contribute to maintaining the admissibility of the evidence.

*Authenticity* requires the examiner to make an explicit link between evidence artifacts and an individual, which the discovery part of F1 directly addresses. Furthermore, reconstructing the timeline (F3) may provide direct or circumstantial evidence linking an individual to certain actions that took place during the incident under investigation.

*Accuracy* requires that the tools used on the evidence function properly and consistently. One established method of determining the accuracy and consistency of forensic tools is to compare the output of tools on the same set of evidence [55], a process which is greatly facilitated by the use of structured formats (F4).

### 3.7 Summary

Conducting forensic analyses for web environments is difficult for examiners because of the need to comply with the Rule of Completeness, associate a suspect with an online persona, gain access to the evidence, give the evidence relevant context, and integrate tools together. To mitigate these challenges, I presented in this work a framework that guides examiners in how to address evidence from the Web using their existing workflows. I

believe that web environments provide exciting challenges to digital forensics, and that this area is ripe for research and innovation.

## Chapter 4

# DBLING: IDENTIFYING EXTENSIONS INSTALLED ON ENCRYPTED WEB THIN CLIENTS

### 4.1 Introduction

In his book *Weaving the Web*, Sir Tim Berners-Lee gives a history of the World Wide Web and describes the formation of many web-related technologies [10]. One observation he shares is that with the introduction of the Java programming language by Sun Microsystems in 1995 came the possibility of writing small application programs called *applets* that “could be sent directly between computers over the Internet, and run directly inside a Web page on a browser.” The potential of this development was that instead of requiring computers to have the storage and memory capacity for all the software the user wanted to run, lower-priced computers with fewer hardware resources could “call up a Web site and download a Java applet” as necessary.

This idea of a computing architecture where lightweight, local machines are configured to interact with remote machines that are responsible for storage and/or computation power, was not new in 1995 nor was it lost when Java applets never fully realized the potential Sir Tim Berners-Lee saw in them. Starting in the 1960s, multi-user mainframes allowed users to connect through terminals, or what came to be known as thin clients. However, as desktops became more powerful, this type of architecture drifted to the side, and these fat clients dominated. The industry is starting to come full circle, with the rise of cheap thin clients that leverage the power of cloud computing, the *web thin client*.

Web thin clients are a type of platform that has recently gained momentum in consumer

markets, as described in Chapter 1 [93, 106, 115]. Web thin clients are also poised to become “a platform for smart TVs and IoT devices” [40], broadening their impact as part of the 40+ billion devices projected to go online by 2020 [2]. In fact, several IoT products from Google already run a web thin client operating system, including Chromecast, Chromecast Audio, Google Wifi, OnHub (a smart router), as well as kiosks and digital signage devices [28, 97, 13].

As web thin clients continue to proliferate in the consumer marketplace, a key question of concern for the forensic community is how to extract residual evidence from web thin clients. This is a difficult question to answer for web thin clients in general, as they can store user data on the device or on *any number of web and cloud service providers* due to their nature as a thin client.

I focus on a web thin client powered by Google’s Chrome OS, a specialized version of the open-source Chromium OS [27]. Chrome OS dominates the web thin client market with the Google Chromebook (the most common form factor of Chrome OS device) and other web thin client devices. The key idea behind Chrome OS is that it only runs a browser natively, and web applications provide all other functionality. Consequently, Chrome OS in particular offers unique forensic challenges, as (1) all data associated with users is encrypted, (2) secure boot safeguards prevent common acquisition vectors, (3) local storage is mainly a cache for online data, and (4) apps and extensions<sup>27</sup> allow users to extend the functionality of the system.

In this chapter, I describe a novel approach to extract residual evidence stored on Chrome OS devices. Specifically, I am able to determine which extensions are installed on an encrypted Chrome OS device *without breaking or otherwise extracting the encryption keys*, accessing the users’ data stored online, or circumventing the secure boot features. I

---

<sup>27</sup>See Subsection 2.2.4 for a discussion on the differences between apps and extensions in Chrome OS.

style my approach after biometrics: I generate fingerprints of extensions' code that persists after encryption, and I am able to use these fingerprints to identify the installed extensions. To accomplish this, I generate a mathematical template for the extensions, which I style as a fingerprint, that I can then compare with portions of the encrypted filesystem.

The information of installed extensions is vitally important to forensic examiners when analyzing a web thin client. As the users' data may be stored on web and cloud providers, an examiner can use this information to understand *which* web and cloud providers might have the users' data. While my approach cannot reveal the data stored on the device (because I do not break the device encryption), I believe that the information of installed extensions is critical information to assist a forensic examiner to discover more sources of user data and ultimately establish a more complete narrative of the incident.

The main contributions of this chapter are:

- I identify important, at rest file metadata that persists after encryption on Chrome OS devices.
- I describe a mathematical template to turn the metadata into fingerprints.
- I develop a novel approach to identify extensions that are installed on Chrome OS devices using only an image of the encrypted drive, and I implement this approach in a tool called dbling.
- I evaluate my approach by creating fingerprints of 160,025 Chrome OS extensions, and I perform a case study by installing 14 extensions on a Chrome OS device and attempt to find their fingerprints.

## 4.2 Background

Thin clients are computers that rely on a server to perform the majority of their computations [111, 82, 7]. A *web* thin client natively supports basic web browsing capabilities, and for any other functionality relies on a combination of web applications and web storage. Web thin clients may also support “packaged apps” that integrate with the browser to run locally to some degree, and in some cases can run offline as well.

This abstract definition of web thin clients helps identify systems of this type, but to understand them thoroughly, it is critical to focus on a particular web thin client instance. As of this writing, there are two principal implementations on the market: Mozilla Firefox OS and Google Chrome OS. I use Chrome OS as the subject of my analysis for the following reasons:

1. Chrome OS is built on the open-source Chromium OS, allowing me to make customizations as necessary for my experimental setup.
2. The number of devices running Firefox OS is limited, the documentation is not as extensive as that for Chrome OS, and Firefox OS is in the process of being retargeted to connected devices [40].
3. The ecosystem of Chrome OS is more mature, has more users, and more extensions, allowing my forensic techniques to have greater impact.

To support the last point above, according to NPD Group [93] the share of Chromebook unit sales for the U.S. commercial channel (which includes desktops, laptops, and tablets) went from 0.2% in 2012 to 9.6% in 2013. In 2014 Chromebooks then experienced a 125% growth in the U.S. business-to-business market, as shown in Table 4.1, and the sales of Chromebooks have continued to increase year after year [92, 91, 1].

Table 4.1: U.S. business-to-business (B2B) PC category growth from 2013 to 2014.

Unit Growth	2013	2014
Chromebooks	N/A	125%
Windows Notebooks	-0.4%	1.9%
Total Desktops	9.0%	5.8%
Build-To-Order (BTO) PCs (Notebooks and Desktops)	8.4%	28.3%
Thin Clients	-4.4%	27.4%

Chrome OS has many unique features which make forensic analysis on them both challenging and rich with opportunities. Section 2.2 discusses these features in detail, but I will now reiterate a few points in the context of the approach I present in this chapter.

Chrome OS's use of a TPM and Secure Boot make it more delicate than other types of systems, given that these make it sensitive to tampering. An unsuccessful attempt to bypass these features trigger the TPM to flush the encryption keys, which in turn forces Secure Boot to wipe the hard drive. Any forensic techniques that aim to support Chrome OS must take these features into account so as to maintain evidence integrity. Therefore, in this work I assume that I cannot extract the encryption keys stored in the TPM and that I cannot boot a custom firmware or OS.

The disk of a Chrome OS device always has a consistent layout, which makes my approach compatible with almost any deployment of Chrome OS. While it is certainly possible to make alterations to the source code of Chromium OS and adjust its behavior so as to foil my approach, accounting for such a scenario is outside the scope of this work.

Although Chrome OS uses eCryptfs to encrypt files, this encryption occurs on each file *individually*. The process of encrypting each file leaves the metadata mostly intact, deviating within a negligible range from the metadata of the unencrypted version of the file. This is the basis from which I create fingerprints of installed extensions.

The files for a Chrome OS extension are static. Extensions can store local data specific



to a user in one of the locations listed in Subsection 2.2.6, but because the data stored in these locations are encrypted and have no precise structure, I cannot use them as part of an extension's fingerprint. However, the fact that an extension's files do not change coupled with the fact that file-level metadata persists after encryption are the key factors that allow me to generate a fingerprint for an extension.

### 4.3 dbling: Design and Implementation

As I discussed to some degree in Section 2.2, the content of a Chrome OS device has a very specific format, which is helpful in determining what data the device stores, where it stores it, and what the limitations are when acquiring it. Furthermore, it is a common practice to eliminate from an examiner's consideration those files known to be part of benign programs, operating systems, and other traceable software applications<sup>28</sup>. By taking advantage of this kind of approach while keeping in mind that Chrome OS encrypts all user data, relatively little data remains that can be extracted from the device for analysis. In fact, almost all of the remaining user-generated data consists of the metadata of the users' files.

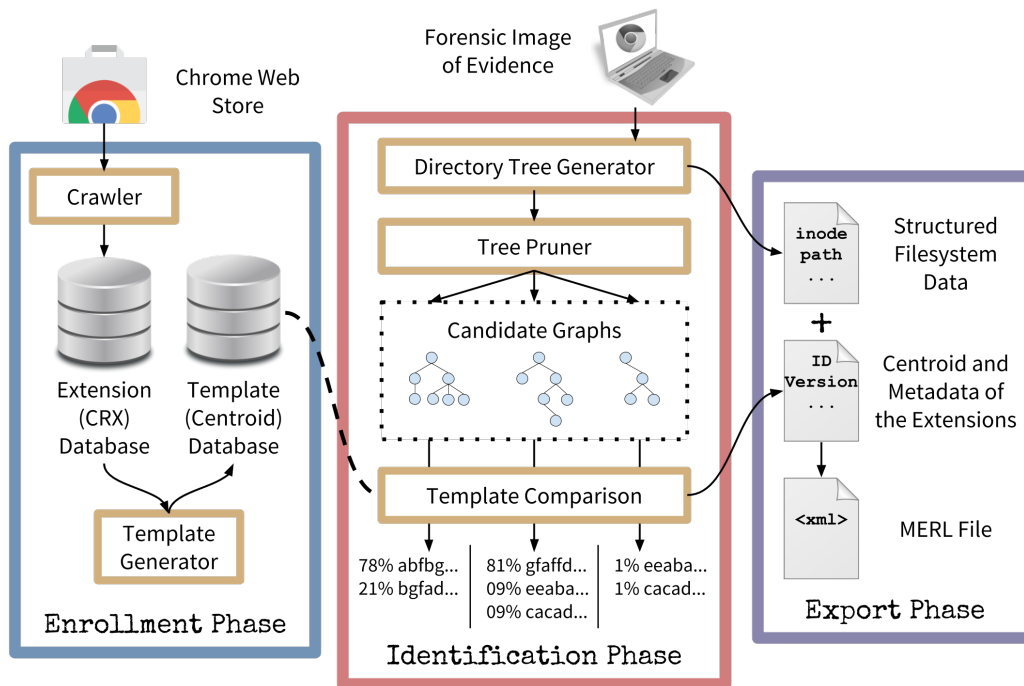
To utilize file metadata under the condition of not having access to filenames and their contents, one sensible approach would be to analyze the system for sets of files that at least have a predictable structure and location within the filesystem.

I leveraged my analysis of the Chrome OS encryption scheme (Subsection 2.2.1) and the Chrome OS extension installation process (Subsection 2.2.4) to create a framework that is able to identify extensions installed on an encrypted Chrome OS device, without access

---

<sup>28</sup>Possibly the best-known example of this is the National Software Reference Library (NSRL) [86] maintained by the National Institute of Standards and Technology (NIST), available at <http://www.nsrll.nist.gov/>. The NSRL uses the term "traceable software applications" to describe the programs they catalog.

Figure 4.2: The dbling framework consists of three phases: Enrollment, Identification, and Export.



to the encryption keys or the ability to load a custom OS [79]. My framework, which I call **dbling**, finds all known Chrome OS extensions, creates fingerprints of extensions, and uses the fingerprints to detect an installed extension. My framework consists of three phases: Enrollment, Identification, and Export. Figure 4.2 depicts the three phases and their components.

Before proceeding, I want to make it clear that the assumptions discussed in Section 2.2 and Section 4.2 remain in effect so as to ensure that my approach is applicable to the widest range of scenarios that examiners may face, namely:

- The data at rest on the hard drive is the main focus of the examiner’s analysis.
- The examiner does not have the user’s credentials.
- Booting to a custom OS is not an option.

- Breaking the device’s encryption is infeasible.
- Because of the encryption, the filenames and file contents are unavailable.
- It is infeasible to perform a memory dump or memory analysis of the device.
- The device is not in developer mode.

In this section and Section 4.4, I use a number of symbols to describe my approach. Table 4.5 summarizes these symbols for easier reference.

#### 4.3.1 Enrollment Phase

Similar to the enrollment phase in biometrics, which generates a mathematical template based on the biometric trait, my approach enrolls extensions by taking a CRX file as input and producing a template of the extension.

The Enrollment phase has a crawler component and a template generator component. To identify an arbitrary, installed extension, dbling must have prior knowledge of all possible extensions. Therefore, the purpose of the **Crawler** is to obtain extensions so that dbling can generate a template for each one.

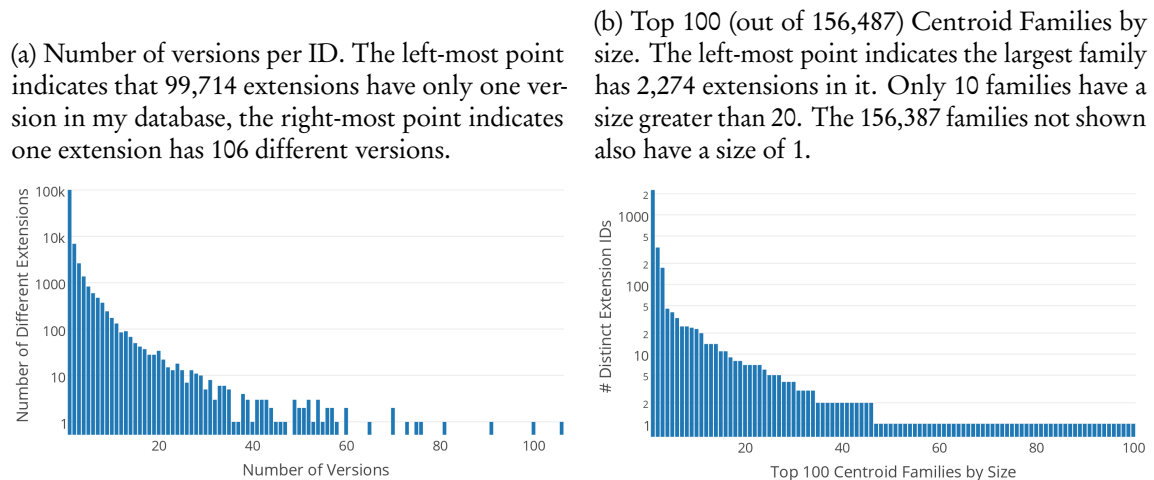
The dbling Crawler is a web crawler that finds all extensions currently listed on the Chrome Web Store, downloads them, and adds them to the CRX database. Table 4.3 and Figure 4.4 show the statistics from running the Crawler over a six month period. In total the Crawler downloaded 121,225 unique extensions. The number of versions it collected of each extension ranged from 1 to 106, as shown in Figure 4.4a. The current implementation of the Crawler only downloads free extensions.

The **Template Generator** creates a template for each of the new CRX files obtained by the Crawler. I adapt the concept of a *centroid* to generate reliable templates for extension representations. The concept of a centroid comes from physics and represents the center

Table 4.3: Statistics from running the Crawler on the Chrome Web Store. The average new CRXs per day includes new versions of previously listed extensions as well as new extensions. I discuss the reasons why I could not download some CRXs in Section 4.5.

Crawling period	6m/158d
Total unique CRX IDs downloaded	121,225
Total CRXs downloaded	160,025
Total Web Store listings at start	74,253
Total Web Store listings at end	82,353
Average listing $\Delta$ per day	51.0
Average new CRXs per day	459.4
Average CRXs not downloaded per day	3.4%

Figure 4.4: More statistics from the dbling Crawler and case study.



of gravity of an object<sup>29</sup>. Centroids have several unique characteristics that make them a good candidate for fast and reliable template generation: (1) if two examined objects have the same centroid, they are almost certain to be the same object; (2) when an object changes a little bit, its centroid will not change a lot; and (3) centroids are sortable, which decreases pairwise comparison time. Given these centroid characteristics, Chen et al. applied centroids to detect malicious application clones on the Android markets [23, 24].

<sup>29</sup>Technically, an object's centroid is only the same as its center of gravity when (1) the object is in a uniform gravitational field and (2) the object's density is also uniform.

Table 4.5: Quick reference for symbols used to describe dbling.

$u_p$	Number of child directories of a file $p$
$v_p$	Number of child files of a file $p$
$x_p$	Mode (permissions) of a file $p$
$y_p$	Depth of a file $p$ from root ( $/$ )
$z_p$	File type of a file $p$
$\varpi_p$	Size of a file $p$
$\varpi$	“Weight” of the extension
$\phi$	$ V $ , the number of files and directories in an extension
$\tau(\phi)$	Number of extensions with the same value of $\phi$
$conf()$	Confidence function

To adapt centroids for my purposes, I first transform the native CRX package into a set of unpacked directories and files. To accomplish this, I use a systematic approach that accounts for the special headers and format of CRX packages [25]. The process is similar to unpacking a zip archive. Then, I generate a 5D-FileTree from the unpacked files.

**Definition 4.1** (5D-FT). *A 5D-FT = (V, A) is a file tree where V is the set of vertices and A is the set of arcs. In a 5D-FT each vertex has a 5D-Coordinate. A vertex in a file tree is either a directory or a file.  $a(p, q) \in A$  is an arc denoting that q is a file or sub-directory in directory p.*

**Definition 4.2** (5D-Coordinate). *A 5D-Coordinate is a 5-dimensional vector  $\langle u, v, x, y, z \rangle$  where u is the number of out-neighbor vertices that are directories, v is the number of out-neighbor vertices that are any other type of file, x is the directory or file’s mode (permissions), y is the depth from the root of the file tree, and z is the file type number (block file, character device file, etc.).*

To derive the 5D-FT for a version of an extension, I set the root of this tree as the version directory and walk through the set of unpacked files in it. All leaf vertices in the 5D-FT are either files or empty directories. By “files” I mean all non-directory file types, including files of an unknown type, regular files, character and block devices, named

pipes, sockets, and symbolic links. In practice, however, extension directories will only contain regular files and more directories. To generate the coordinate of a file or directory as described in Definition 4.2, I execute the `stat` command on the filename of each  $p \in V$  and extract the relevant portions.

I define the centroid of a 5D-FT as follows:

**Definition 4.3** (Centroid). *A centroid  $\vec{c}$  of a 5D-FT is a 7-dimensional vector  $\langle c_u, c_v, c_x, c_y, c_z, \varpi, \phi \rangle$ <sup>30</sup> where  $c_u = \frac{\sum_{a(p,q) \in A} (\varpi_p u_p + \varpi_q u_q)}{\varpi}$ ,  $c_v = \frac{\sum_{a(p,q) \in A} (\varpi_p v_p + \varpi_q v_q)}{\varpi}$ ,  $c_x = \frac{\sum_{a(p,q) \in A} (\varpi_p x_p + \varpi_q x_q)}{\varpi}$ ,  $c_y = \frac{\sum_{a(p,q) \in A} (\varpi_p y_p + \varpi_q y_q)}{\varpi}$ ,  $c_z = \frac{\sum_{a(p,q) \in A} (\varpi_p z_p + \varpi_q z_q)}{\varpi}$ ,  $\varpi = \sum_{a(p,q) \in A} (\varpi_p + \varpi_q)$ , and  $\phi = |V|$ . Additionally,  $C = \{\vec{c}_0, \vec{c}_1, \dots, \vec{c}_n\}$  is the set of all centroids.*

The  $\varpi_p$  parameter is the size of the file  $p$ , which is analogous to the weight of an object from the physics model. Because both the filesystem and the AES encryption used in eCryptfs use 4,096 bytes as the block size, the encrypted file size is the original file size rounded up to the nearest 4,096 bytes<sup>31</sup>. To calculate  $\varpi_p$ , I compute the unencrypted file's size when encrypted and divide it by 4,096 to get the number of blocks occupied by the corresponding encrypted file. Not only does this simplify dbling's view of the file size for the context of working with eCryptfs, but it also reduces variance in the centroids of each file.

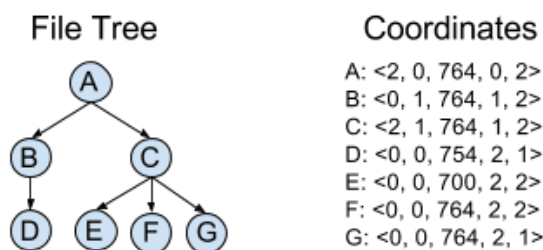
The inclusion of  $\phi$  in the centroid was a direct result of my observations while experimenting with dbling. In the database,  $\phi$  ranges from 3 to 36,743.

---

<sup>30</sup>The glyph  $\varpi$  is called “variant pi” or “pomega.”

<sup>31</sup>In reality, calculating the size of the encrypted version of a file is not quite this simplistic. See Subsection 4.6.2 for more details.

Figure 4.6: A 5D-FileTree example and the 5D-Coordinates for its vertices.



*Example:* Figure 4.6 shows a 5D-FT example. Vertex A is the version directory<sup>32</sup>. It has two child directories and no other file. Its mode is 764, which means that owner can read, write and execute, group can read and write, and others can only read. The depth from the version directory to A (itself) is 0, and it is a directory, which is type 2 in ext filesystems. So its coordinate is  $\langle 2, 0, 764, 0, 2 \rangle$ . Accordingly, I can calculate the centroid of this tree with the coordinates of all its vertices using Definition 4.3.

#### 4.3.2 Identification Phase

In the dbling framework, the Identification phase is the part that forensic examiners will use. Because of this, the phase must address the three main steps of the forensic process<sup>33</sup>: acquisition, authentication, and analysis.

The methods of acquiring and authenticating evidence from a Chrome OS device depend on what hardware the manufacturer used for the device. With increasing frequency, Chrome OS devices come with an embedded solid state hard drive soldered directly to the motherboard, typically in the eMMC form factor. These types of drives make the initial

---

<sup>32</sup>For more information on the directory structure of extensions, please refer to Subsection 2.2.6.

<sup>33</sup>I stated in Subsection 1.1.1 that there is a fourth step to the forensic process, presentation. I did not include presentation in this list since the Export Phase (Subsection 4.3.3) provides this step.

acquisition difficult in that the examiner must have the necessary equipment to conduct what is called “chip-off” forensics. Chip-off extraction necessitates the removal of the flash drive from the circuit board to which it is attached, then reading its contents using a specialized chip programmer or adapter. However, in the event that the device has a removable hard drive, typical acquisition and authentication methods that are compatible with Linux operating systems will work on a Chrome OS device.

For the analysis step, the high-level goal is to identify installed extensions. However, dbling must first identify the extension directories in the encrypted filesystem, for which it does not have the unencrypted names of files or directories or their contents. To overcome this initial challenge, dbling searches the encrypted filesystem to identify the **candidate graphs** that represent the most likely directories to contain user-installed extensions, generates templates for the candidate graphs, and performs a one-to-many search to determine what extensions each may be. Intuitively, dbling cannot perform a one-to-one verification because the files are encrypted and cannot self-identify as any particular extension.

To begin, the examiner completes the process of acquiring and authenticating a forensic image from the device. The image does not need to be of the entire disk, but can be of the partition named STATE [29], whose mount point is /home and stores all the users’ data<sup>34</sup>.

The next step in the process is to create a graph of the filesystem on the disk image. To do this, the examiner executes dbling and points it to the forensic image taken previously. Then, dbling mounts the image using a read-only loop device and walks through the filesystem, creating a vertex for each inode and an arc between parent directories and the files they contain, as described earlier.

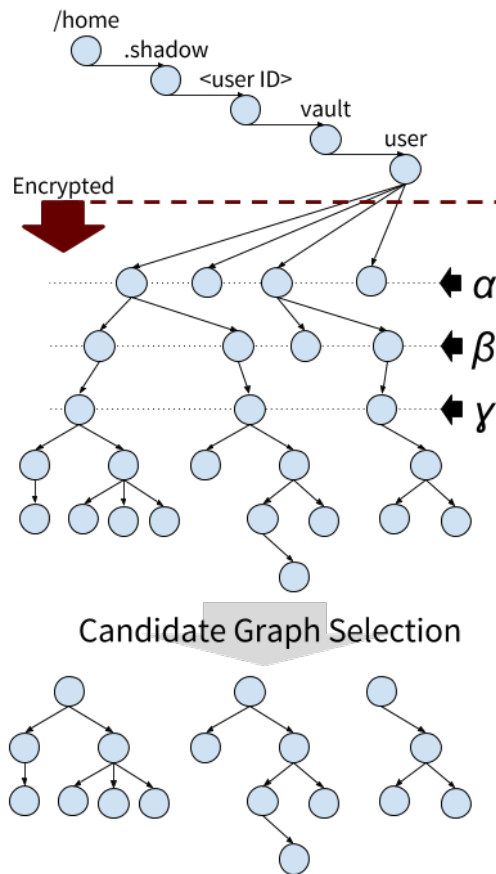
Next, dbling creates a graph representing the whole filesystem to make sure to not miss any relevant files. However, given that only a subset of these files are relevant to

---

<sup>34</sup>This is why Figure 4.7 shows /home as the root vertex.



Figure 4.7: Pruning the full directory tree to a few candidate graphs. The vertices at level  $\alpha$  are the potential “Extension” directories, vertices at level  $\beta$  are potential extension ID directories, and vertices at level  $\gamma$  are potential extension version directories.



identifying extensions, dbling leverages a number of identifying characteristics to label vertices it knows cannot be part of an extension. With these labels, dbling prunes irrelevant files from the graph. In addition to the details presented in Section 2.2, these helpful characteristics are:

1. All directories outside the scope of `/home/.shadow` will *never* contain user-installed extension files and so are of no interest to dbling.
2. The Extensions directory will *always* be six directories deep from `/`, level  $\alpha$  in Figure 4.7.

3. The `Extensions` directory will *always* be encrypted.
4. The `Extensions` directory will *always* contain *only* directories (i.e., no regular files), one for each installed extension.
5. The directory of a specific extension will *always* contain *only* directory files at level  $\gamma$ , one for each version of the extension currently installed (typically 1).
6. The extension version directory *must* be non-empty since extensions always require a manifest file.
7. The extension version directory will *only* contain directories and regular files, no symbolic links or other types of files.

While generating the directory tree, `dbling` also uses the `stat` program on each file to get its metadata, then it generates the 5D-Coordinate for the file and stores it in connection with the vertex corresponding to the file. Once the graph is complete, `dbling` prunes the tree by filtering out all unnecessary files using the labels it created earlier.

Having created the pruned directory tree, `dbling` next selects and separates the candidate graphs from the directory tree as shown in Figure 4.7, making them proper 5D-FTs. Then `dbling` calculates the centroid for each candidate graph using the formula in Definition 4.3.

With the centroids of each of the candidate graphs, the next step is to perform the one-to-many matching for each candidate. The naïve approach would be to attempt to match a centroid against all of the known templates. However, this would be inefficient as the size of the known templates grows. Therefore, `dbling` takes advantage of two aspects of the templates. First, because centroids are vectors, `dbling` can sort them and then set a distance threshold to limit how many neighboring centroids it will include in the pairwise comparison. Second, the value of  $\phi$  must match exactly, reducing the number of entries to consider.

However, due to the coarse granularity of the centroids<sup>35</sup>, it can still be the case that centroids from multiple extensions are the same value, either because they are different versions of the same extension or because the structure and metadata values are the same. I call these *centroid families*.

**Definition 4.4** (Centroid Family). *A centroid family  $CF_i \subseteq C$  is a set of centroids such that  $\vec{c}_j = \vec{c}_k \forall \vec{c}_j, \vec{c}_k \in CF_i$ .*

In essence, centroid families provide a measure for the uniqueness of the templates generated by dbling. Also, it is natural that multiple extensions will correspond to the same centroid. Consider the case of a simple extension that has only one file of less than 4096 bytes. In this case, any other extensions that also fit these simple criteria will have the same centroid and be in the same centroid family. Thus, when a candidate graph template matches a template of this centroid family, dbling cannot determine for certain which extension is installed, only that it is one of the extensions in the centroid family.

Table 4.8 describes statistics from the centroid families. Of particular note is that 156,441 centroids correspond to unique extensions, which means that only 46 centroids correspond to multiple extensions. Figure 4.4b investigates the distribution of centroid families to multiple extensions. In the worst case (the left-most bar in Figure 4.4b), a single centroid corresponds to 2,275 extensions. Still, even in the worst case, this narrows down the possible extensions installed on the Chrome OS device from the 160,025 possible extensions to just 2,275.

The key takeaway from all these details on centroid families is that, as long as the Template Generator accurately predicts what the centroid of an extension will be after Chrome OS installs and encrypts it, dbling will be able to identify that extension with

---

<sup>35</sup>This coarseness is due to the effect of encryption on the file metadata: exact file sizes are unavailable.

Table 4.8: Statistics on centroid families. Membership numbers are for distinct extension IDs.

Total extension entries	160,025
Total centroid families	156,487
Biggest centroid family	2,274
# of families $\geq 20$	11
# of families $= 1$	156,441

a high degree of accuracy, given how unlikely a centroid is to be a duplicate of another centroid already recorded.

With the potential matches for each candidate graph identified, the next step is to normalize the centroid vectors. In my experiments, I noticed that the range of values for each field in the centroid vectors is not the same. For example, the valid values for a file’s 5D-Coordinate for  $x$  (the file’s mode) have a range of 0–777, but  $z$  (the file type number) has a range of 1–7. To accommodate this characteristic and avoid allowing certain fields to dominate the distance calculation (see Equation 4.9), dbling normalizes centroids using the following definitions:

**Definition 4.5** (Normalized Centroid). *A Normalized Centroid  $\vec{c}'$  is a 7-dimensional vector, where each component  $c'_i$  (except for  $\phi$ ) is calculated as  $c'_i = \frac{c_i}{cn_i}$ , where  $c_i$  is the  $i^{\text{th}}$  component of the non-normalized centroid  $\vec{c}$  and  $cn_i$  is the  $i^{\text{th}}$  component of the Normalizing Vector. The  $\phi$  component is never normalized.*

**Definition 4.6** (Normalizing Vector). *The Normalizing Vector is a 6-dimensional vector  $\vec{cn} = \langle cn_u, cn_v, cn_x, cn_y, cn_z, cn_\omega \rangle$  where each component  $cn_i$  of  $\vec{cn}$  is the maximum value for that component in the database of collected centroids.*

With the normalized centroids calculated, dbling finalizes the Identification phase by ranking the matches according to its confidence level in the matching. For a candidate

graph  $a$  and extension  $x$ , dbling calculates the confidence level  $conf(a, x)$  as follows:

$$conf(a, x) = \frac{e^{-\delta|\vec{c}'_a - \vec{c}'_x|}}{n} \quad (4.9)$$

Confidence levels will be in the interval  $(0, 1]$ , indicating a percentage likelihood that candidate graph  $a$  is extension  $x$ . In the above formula,  $\delta$  is a weight for adjusting the rate at which the confidence drops as the distance between  $\vec{c}'_a$  and  $\vec{c}'_x$  increases, and  $n = |CF_i|$  given  $\vec{c}'_x \in CF_i$ .

### 4.3.3 Export Phase

In the Export phase, dbling stores the matching and ranking results from the Identification phase in a structured format to allow examiners to share them. This phase of my approach is important considering the increasing emphasis from the security community to standardize intelligence formats for machine-to-machine communication and other sharing purposes [57].

To export the ranked results so as to not lose their meaning in the context of the original forensic image, dbling makes a connection between the evidence artifacts and the results in a cohesive format. To accomplish this, I developed an XML schema that directly references the inode information from the forensic disk image and adds the ranking information from the Identification Phase. I call these **Matching Extension Ranking List (MERL)** files. Listing 4.10 shows the MERL file that lists matching extensions with confidence levels.

Listing 4.10: A MERL file with the extensions that might match the extension version directory from the source image.

---

```
<?xml version="1.0" encoding="UTF-8"?>
<merl xmlns="https://mikemabey.com/schema/merl" version="0.1" ... >
  <source>
    <image_filename>chromebook_test01.img</image_filename>
    <mount_point>/mnt/dbling/</mount_point>
  </source>
  <creator>
    <program>dbling_merl</program>
    <version>0.0.1</version>
    <execution_environment>
      <command_line>sudo dbling_merl -v -m /mnt/dbling/</command_line>
    </execution_environment>
  </creator>
  <match>
    <inode>259696</inode>
    <candidate>
      <ext_id>abfbgiablndngkapfilcncjplidclalae</ext_id>
      <ext_ver>1.0.3</ext_ver>
      <ext_name>Mangekyou1999</ext_name>
      <ext_vendor>Mangekyou1999</ext_vendor>
      <confidence>0.78</confidence>
    </candidate>
    ...
  </match>
</merl>
```

---

## 4.4 Case Study

I conducted a case study of dbling in a testing environment to demonstrate its effectiveness. The target device was an Acer C720 Chromebook, which I selected based on its removable NGFF hard drive. The version of Chrome OS running on the device at the time was 48.0.2564.92.

I chose 14 extensions from the Chrome Web Store, some from the “Popular Apps” category and others from the “Productivity” category that were listed as working offline. I used these criteria to help simulate the scenario where the user has installed extensions

that use online services that store files somewhere in the cloud, as well as the case where the extension does more work locally by having more code installed on the Chromebook.

I also constrained the case study to extensions and versions that the Crawler had downloaded. This was necessary because of the types of extensions I was unable to download (see Section 4.5).

With the selected extensions that met my criteria, I installed all 14 of them on the target device, took a forensic image of the disk using `dd`, then ran `dbling` on the image. From the MERL results, I searched for the IDs of the installed extensions and recorded their ranks among the matching centroids to get an idea for how accurately `dbling` was able to identify the extension. Table 4.11 summarizes the results from these experiments.

The  $\tau(\phi)$  column of Table 4.11 shows how many extensions have the same value of  $\phi$  (number of files and directories) in the database. In this case I consider different versions of an extension separately. As the table shows, the higher the value of  $\tau(\phi)$ , the more difficult it was to find the correct extension. This makes sense because `dbling` uses  $\phi$  as a pre-filter when looking for matching centroids, so the lower the value of  $\tau(\phi)$  the fewer centroid distances I have to calculate and rank.

The numbers for  $\phi$  and  $\tau(\phi)$  in Table 4.11 appear to have an inverse relation. Considering that many extensions in the Chrome Web Store are simply links to a web application, it makes sense that there are so many extensions with  $\phi = 5$ , for example.

The results from this case study show that, for complex applications, `dbling` is able to narrow down the scope of digital evidence for a forensic examiner. In the worst case, which is in fact  $\phi = 5$ , `dbling` narrowed down the scope to 11,768 potential extensions from the total of 160,025. In other cases `dbling` was able to identify the extension exactly.

Table 4.11: Rankings for matching the candidate graphs to the correct extension using centroids. Numbers given in ranges indicate multiple versions of the extension matched the candidate.

Extension Name	Version	Offline	# Users	$\phi$	$\tau(\phi)$	Rank
Lucidchart Diagrams — Desktop	1.116	Y	389,087	3507	3	1-3
Marxico	1.6.1	Y	29,379	711	7	1-7
Piconion Photo Editor	1.9.0.0	Y	24,684	370	12	2
Reditr — The Best Reddit Client	0.3.3.1	Y	53,687	134	71	1
Google Hangouts	2016.120.1336.1	N	5,468,622	150	78	1-19
Advanced REST client	4.6.0	N	1,040,107	131	102	8-12
Evernote Web	1.0.8	N	3,638,599	41	580	308
TweetDeck by Twitter	3.10	N	1,584,975	15	2,808	894; 1,904
Any.do	3.1.1	Y	260,355	13	3,894	3,217
ShiftEdit	1.43	Y	159,170	6	5,766	2,175
Outlook.com	1.0.2	N	1,485,767	7	9,259	4,318; 4,520
Google Play	3.1	N	3,624,424	5	11,768	6,919; 6,989
Netflix	1.0.0.4	N	1,769,793	5	11,768	9,630
Little Alchemy	1.4.0	Y	1,518,447	5	11,768	10,328



## 4.5 Limitations

Although my approach may introduce high-level principles that could apply to other encrypted filesystems, dbling cannot presently operate on anything other than the discussed web thin client operating systems because of its dependency on the disk layout discussed in Subsection 2.2.5 and the precise location of static extension files as discussed in Subsection 2.2.6.

The success of my approach relies on a complete enrollment of all extensions, as dbling will never be able to identify an extension that is not enrolled. However, in practice I encountered several challenges to enrolling all possible extensions.

Chrome seems to have some interesting mechanisms in place for interacting with the Chrome Web Store that I have not yet been able to reverse-engineer. For instance, developers can opt to list their extension as being compatible only with Chrome OS, thereby disallowing installations of that extension on the Chrome browser in an arbitrary operating system.

To investigate this feature, I first attempted to get around this check by changing the “User-Agent” header of my requests to mimic Chrome OS, however I was still not able to download the extension. Next, I browsed the Chrome Web Store with the Firefox browser. Many page elements were missing and it was clear Firefox did not have all of the JavaScript files connected with the site.

These two results indicate that Chrome and Chrome OS have “insider knowledge” of how to interact with the Chrome Web Store. The evaluation of compatibility is clearly not done server-side, and the client-side browser needs to know how to fill in the gaps from the Store’s HTTP responses. Since I am not privy to these details, I was unable to spoof the identity of my downloader to obtain these extensions.

Other reasons why I was unable to download certain extensions are: (1) either the developer or Google removed the extension from the Chrome Web Store<sup>36</sup>, (2) the developer released the extension as a closed beta version or by invite only [26], or (3) the extension is non-free.

Although my approach helps examiners identify which extensions the user has likely *installed*, dbling cannot in its current form give any indication as to which extensions the user actually *uses* or with what frequency. Such a feature would be a logical evolution of dbling's current abilities.

The problem of extensions generating the same centroid and belonging to the same centroid family impacts dbling's detection results. I believe that this large number may be due to published extensions that are clones of either other extensions on the Chrome OS store or very simple extensions. As shown in my case study, the more complex an extension, in both file size and number of files, the more likely it is to be unique. Furthermore, I argue that it is a natural result of the loss of data and metadata from the encryption of the filesystem.

#### 4.6 Improving Accuracy

In the evaluations I conducted as part of the case study (Section 4.4), dbling's accuracy when identifying extensions left room for improvement. The key issue was that the profile dbling generated for an extension after the Crawler downloaded it did not match the profile of the same extension when installed on a Chromebook. As I mentioned when discussing the purpose of computing centroid families, the most important component of dbling

---

<sup>36</sup>This applies to extensions that the Crawler discovered and saved its ID in the database. The Crawler periodically checks if previously seen IDs are still valid and accessible.

is the Template Generator, as its accuracy in mimicking the extraction and installation process of Chrome OS has the greatest effects on the rest of the process.

In this section, I will describe the various approaches I implemented to help improve the accuracy of dbling's Template Generator.

#### 4.6.1 Image Conversion

As mentioned in Subsection 2.2.6, one of the steps Chrome OS takes when installing an extension is it converts all image files in the extension to PNG format. The reasons for this are not documented in the source code of Chromium OS or the online documentation for the project. Nor is there an indication as to the encoding settings Chrome uses in the conversion process.

I implemented an image conversion step in the unpacking code used by dbling<sup>37</sup> that uses the Pillow<sup>38</sup> library for Python to do the actual conversion. While the image conversion worked just fine, there were two major problems with this improvement.

First, the process of loading the image files into Pillow, converting them to PNG, and saving them to disk again proved to be particularly time-consuming. Previous to integrating the conversion into the process, crawling the Chrome Web Store, downloading new extensions, and generating templates for them typically took about 10 hours, short enough to complete the whole process twice a day. After including the image conversion, the process took so long that I eventually stopped the Crawler after two weeks because it had not yet finished a single batch of new extensions.

I concluded that the most important thing to determine was if the conversion was

---

<sup>37</sup>See Appendix C.2 for additional details on dbling's unpacker.

<sup>38</sup>See <http://pillow.readthedocs.org/>.

improving the profiler's accuracy when identifying installed extensions. So I regenerated the templates for just the extensions used in the case study and compared the pre-calculated templates with the templates from the experimental device. Unfortunately, the results showed that the new centroid values were only nominally improved from the version of the Crawler that did not include image conversion in the template generation process. And this was the second major problem I encountered: it did not seem to make much of a difference whether or not dbling converted the images.

Since there was no evident advantage to incurring the extra processing load from the image conversion, I made no further attempts to integrate this improvement into dbling.

#### 4.6.2 eCryptfs Metadata

The next attempt I made to improve the accuracy of dbling's templates was to incorporate eCryptfs directly into the template generation process. To provide some context, eCryptfs affects the size of the files and filenames it encrypts due to padding and other operations executed during encryption [78]. Some aspects of these alterations are predictable and I made some attempts to capture them in a formula, but the sizes never matched up. The idea was that, by using eCryptfs, it would no longer be necessary to calculate the file sizes because they would be easily observable. At a high level, the approach is to extract the CRX to a directory mounted as an eCryptfs upper filesystem and then generate a template using the encrypted version of the files in the lower filesystem.

There were two principal motivating factors for incorporating eCryptfs into the process. The first is, as just discussed, to remove the guesswork from calculating the sizes. Second, by profiling the *encrypted* files when generating a template for a CRX, dbling

would be operating under conditions that more closely resemble those present for an extension installed in Chrome OS.

The approach I took to make this change was to implement a subclass of Python's `TempDirectory`<sup>39</sup> class, which I named `EncryptedTempDirectory`<sup>40</sup> and which has two principal features.

The first important feature of `EncryptedTempDirectory` is that, because it inherits from the `TempDirectory` class, the directories it creates only exist in memory. In Linux, this typically means the directory will be in `/tmp`, `/var/tmp`, or `/usr/tmp`. By using a temporary directory as the extraction path for CRXs, `dbling` avoids incurring as much of a performance penalty from disk write operations.

The second key feature of this approach is that it automatically configures `eCryptfs` and executes the necessary mount commands to prepare the upper and lower directories. As long as the required packages are installed, using `EncryptedTempDirectory` is straightforward. Listing 4.12 shows an example of how to use `EncryptedTempDirectory` taken from the documentation<sup>41</sup>. In the example, a `TempDirectory` is created and used as the upper directory, and then of course the `EncryptedTempDirectory` is set as the lower directory. Within the `with` clause, the two directories can be referenced using the `upper` and `lower` variables. At the end of the `with` clause, both of the temp directories are automatically destroyed.

After implementing the code for `EncryptedTempDirectory`, I ran the Template Generator on all the extensions in the `dbling` database once again and then compared the

---

<sup>39</sup>See <https://docs.python.org/3/library/tempfile.html#tempfile.TemporaryDirectory>.

<sup>40</sup>See Appendix C.2 for additional details on the `EncryptedTempDirectory` library.

<sup>41</sup>Available at [http://crx-unpack.readthedocs.io/en/latest/encrypted\\_temp.html](http://crx-unpack.readthedocs.io/en/latest/encrypted_temp.html).

Listing 4.12: Example usage of an EncryptedTempDirectory object.

---

```
with TemporaryDirectory() as upper, \  
    EncryptedTempDirectory(upper_dir=upper) as lower:  
    # Code using upper and lower directories goes here
```

---

centroids for the extensions from the case study with the templates from the experimental device. Unfortunately, incorporating eCryptfs into the enrollment phase only slightly improved the results, similar to the image conversion improvement.

#### 4.7 Related Work

Garfinkel first introduced DFXML to the forensics community [52]. As described by Nelson et al. in [88], “DFXML is an XML syntax that summarizes storage systems. ... For files, DFXML encodes metadata one expects of an inode, and some data summary attributes, like content checksums.” The fact that DFXML stores a summary of a system in a well-defined, extensible format has a lot of benefits, and many researchers have used DFXML to great effect. However, DFXML does have some shortcomings. As noted by Garfinkel et al. in [54] (and later reiterated by Nelson in [87]), a tool cannot assume “a unique path for each object in [the DFXML of] a file system.” In other words, because of how DFXML stores information, it is possible to have duplicate entries for the same piece of data on the system. This property has caused some problems for my work on web thin clients. My solution has been to walk the filesystem myself and gather information about the system rather than attempting to sanitize the DFXML. Hopefully in the future I can overcome this issue so I can incorporate this storage format into my work.

Chrome OS uses ext4 as the underlying filesystem, which is also adopted by the Android and Tizen operating systems. Fairbanks [44] and Fairbanks et al. [45] presented

thorough analyses of ext4 in general for digital forensics. Kim et al. studied how to use the ext4 filesystem journal logs to extract what files users accessed and how to recover deleted files [71]. Eo et al. examined how to recover deleted files in Tizen [42]. Corbin studied performing forensic analyses on Chromebooks, in which they assumed the examiner is able to log in to the device before trying to run some simple commands on the system, like `ls`, `ls -l`, `grep`, etc. [38]. However, all the aforementioned approaches assume the filesystem is unencrypted, either because it is stored in plaintext [71, 42] or decrypted before analysis [38]. This assumption dramatically limits forensic analysts' ability when they can only acquire images of encrypted filesystems.

Other researchers have studied the impact of full disk encryption on digital forensics [19]. Efforts that tried to do forensics on encrypted systems all explored ways to recover the encryption keys from memory or other places [6, 59]. In these approaches, the success of forensic analysis totally relies on the success of extracting the keys. Little literature exists on directly doing forensics on the encrypted filesystems of Chrome OS or any other type of web thin client.

To overcome the obstacles inherent to encrypted filesystems, my work follows the line of metadata analysis for digital forensics. Buchholz and Spafford in [15] discussed not only what types of information are available by analyzing metadata in FAT and NTFS, they also outlined what new types of metadata would aid forensic investigations. Oliver leveraged metadata analysis to perform forensics on databases [94]. Castiglione et al. used document metadata to retrieve forensic information from Microsoft Office documents [21]. Thorpe et al. examined cloud log metadata for forensic analysis of virtual machines and operating systems running in clouds [109].

## 4.8 Summary

Web thin clients are an exciting and expanding computing platform. They offer significant benefits for users, allowing them to leverage the power of cloud computing in an inexpensive device. However, this new class of devices brings interesting and unique forensic challenges, particularly in the case of Chrome OS, where the filesystem is encrypted by default.

By using file metadata that is preserved after encryption, I am able to extract forensic evidence without breaking the encryption or loading a custom operating system. My approach significantly helps narrow down analysis space on the forensic image. Furthermore, I believe that this is a significant first step toward comprehensive web thin client forensics.



## Chapter 5

# FORENSIC TIMELINE RECONSTRUCTION FOR ENTERPRISE WEB THIN CLIENTS

### 5.1 Introduction

As enterprises and organizations seek to simplify the administration and management of the devices they use, one solution they are increasingly turning to (especially in K-12 schools) is web thin clients [61], which I define in Section 4.2 as computers that natively support basic web browsing capabilities, but rely on a combination of web applications and web storage for any other functionality. The most prominent example of both a web thin client and a solution that organizations are leveraging for improved administration is the Google Chromebook and its companion service, G Suite.

Chromebooks<sup>42</sup> are laptops that run Google's Chrome OS, a web thin client operating system with several security features designed to protect users' privacy [79]. While these security features are a good thing under normal circumstances, they make it infeasible to reconstruct a timeline using only the device, for reasons I explain in Subsection 5.2.1. Because the device itself cannot be used to collect timeline data, when a forensic examiner needs to reconstruct the incident, it will be necessary to turn to a different data source. For a more detailed discussion of Chrome OS's features, please refer to Section 4.2.

G Suite is an enterprise-oriented service offered by Google that gives organizations access to several of Google's productivity and communication services in a manner designed

---

<sup>42</sup>While manufacturers do produce and sell Chrome OS devices of other form factors, the Chromebook is the most common and the most popular form factor.

to meet workplace needs. If, in addition to these services, an organization wants to provide its personnel with Chrome OS devices, G Suite allows the administrator to “enroll” devices with the organization, which allows for more control over things such as who can log into the devices and what extensions are installed. The main interface for administrators to use G Suite is a web-based management dashboard that shows user activity, license status, storage quota usage, and more. In January 2017, Google reported having over 3 million paying businesses as G Suite customers, up from 2 million as of November 2015 [76].

Google also offers a version of G Suite customized for educational institutions, called G Suite for Education, at no cost to schools<sup>43</sup>. The combination of no subscription cost and simplified administration tools have undoubtedly been factors in the high adoption rate of G Suite and Chrome OS devices in the K-12 market [65, 106]. In fact, of all computers purchased by K-12 schools in the U.S. in 2014, 2015, and 2016, Chrome OS devices accounted for 38%, 50%, and 58%, respectively [49]. In January 2017, Google reported G Suite for Education has 70 million users worldwide, up from 60 million the previous year and 50 million in October 2015 [75].

In addition to the management dashboard, G Suite provides administrators with an API that gives them a programmatic method to retrieve information on enrolled Chrome OS devices, on users’ accounts, and on the files and metadata stored by Google’s services. Admins may also access information created and stored by users in Google Drive, Google Photos, Google Plus, Gmail, and Google Contacts. Accessing users’ data in this way is called “impersonation” or “masquerading”<sup>44</sup>.

---

<sup>43</sup>Implied in the fact that schools do not pay for G Suite for Education is that they were not included in the 3 million paying customers mentioned previously.

<sup>44</sup>In order to impersonate users and access their data, the admin must first manually enable a G Suite feature called “domain-wide delegation.” For more information, see <https://developers.google.com/admin-sdk/reports/v1/guides/delegation>.

Although the G Suite APIs provide a wealth of information, there still remain two key issues that prevent examiners from being able to use this information to reconstruct a timeline. First, the data available from G Suite alone does not provide sufficient granularity to conclusively link the time-related data from the disparate API calls (in Section 5.3, I give an in-depth explanation of the information the APIs provide). Second, although the dashboard shows most of the information available through the API, only some of the information can be downloaded and G Suite does not offer a way to correlate data from different sections of the dashboard. Accordingly, administrators and examiners would currently have to implement their own tool to access the raw data from the API and perform finer-grained analyses.

In this chapter, I present my method of leveraging the data available from APIs in such a way that it effectively acts as a proxy for obtaining the information from the device itself. Additionally, I seek to answer forensic questions about the timeline of events, namely:

- *Who used the device?* By answering this question, examiners will be able to address the Authenticity Rule of Evidence, which requires an explicit link to be made between a specific individual and an event.
- *When did the individual use the device?* Answering this question also addresses Authenticity, providing the element of time in the link to an event.
- *What did the individual do while on the device?* Knowing what activities the suspect performed while on the device gives substance to the timeline and helps the examiner address the Completeness Rule of Evidence.
- *What files may contain inculpatory/exculpatory<sup>45</sup> evidence regarding the incident based on when they were created and modified?* Answering this question allows examiners

---

<sup>45</sup>Inculpatory evidence suggests guilt, exculpatory evidence suggests innocence.

to reduce their analysis space, thereby improving the efficiency of their investigation efforts.

In my approach, I use two sets of APIs to accomplish these objectives. The first is the G Suite API, which provides information on devices and login attempts, as well as files and other data. I also leverage the API for Chrome apps<sup>46</sup> to acquire data that complements the information available from G Suite and allows me to make an explicit connection between a specific individual and the device or event (Authenticity).

The main contributions of this chapter are:

- I evaluate the time-related data available from the G Suite and Chrome APIs for their relevance in reconstructing the timeline of an incident's events.
- I describe a method of connecting the time-related data to complete the narrative of the incident, and give the details of my proof-of-concept implementation called Beacon.
- I implement a simple command line tool, that I call Gripper, that allows the examiner to retrieve the raw data from Google and optionally adjust the start and end dates of the data to retrieve using command line parameters. The tool is also capable of exporting the timeline data in a structured format (JSON).
- I provide a visualization of the timeline data with customizable granularity and input fields.

---

<sup>46</sup>In Subsection 2.2.4, I explain the differences between apps and extensions. In this instance, I use some APIs that are specific to apps and are not available for extensions.

Listing 5.1: Partial output of the `mount` command in Chrome OS showing the three directories mounted using the Linux kernel’s device mapper driver.

---

```
/dev/mapper/encstateful on /mnt/stateful_partition/encrypted type ...  
/dev/mapper/encstateful on /var type ext4 ...  
/dev/mapper/encstateful on /home/chronos type ext4 ...
```

---

## 5.2 Background

I will now explain two items that are integral to understanding my approach. First, I will discuss the technical details of why it is infeasible to acquire timeline data directly from a Chrome OS device, which motivates my use of the G Suite and Chrome APIs as a proxy for the device. Second, I will describe the method used by Chrome OS to self-assign its network adapters an IPv6 address, which is an essential component of my implementation that creates a direct link between a device and when a user logs in.

### 5.2.1 Chrome OS System Logs

Like most other Linux-based systems, Chrome OS stores logs of system events, including when users log in<sup>47</sup> and when various programs perform significant actions. Chrome OS stores these log files under `/var/log/`; however, accessing the log directory’s actual contents is not as simple as traversing the filesystem. Further inspection shows that `/var/` is one of three directories that Chrome OS manages using the kernel’s device mapper driver, as shown in the output of the `mount` command in Listing 5.1.

The device mapper, as the Red Hat Enterprise Linux guide to Logical Volume Man-

---

<sup>47</sup>In this context, I am specifically referring to system users (see Subsection 2.2.8), but Chrome OS uses a similar mechanism to log Chrome users.

ager Administration defines it, “is a kernel driver that provides a framework for volume management. It provides a generic way of creating mapped devices, which may be used as logical volumes” [77].

The device mapper supports several types of mapping formats, called “targets,” that change what operations the driver performs on the files when passing through the specified device. The `crypt` mapping target encrypts the data using the kernel Crypto API [77]. Chrome OS uses the `crypt` target on its `/dev/mapper/encstateful` mapping device, which means that when the data in `/var/` is at rest, it is encrypted as well as combined into the underlying mapper file with the other two logical volumes, making it impossible to directly read the file from the hard drive without a way to break the encryption.

Assuming that it remains infeasible to break the encryption used by the device mapper, the only practical method of accessing the contents of the file is to boot the system and allow the kernel to mount the filesystem, including the device mapper. However, once booted, Chrome OS does not allow users to access the underlying filesystem<sup>48</sup>, so the log files remain inaccessible. Chrome OS also applies this restriction to extensions by sandboxing them and denying them access the root filesystem.

To summarize, Chrome OS maintains system logs that would be very beneficial to a forensic investigation. However, because of the security features of the operating system, all of these logs are inaccessible unless the device is in developer mode.

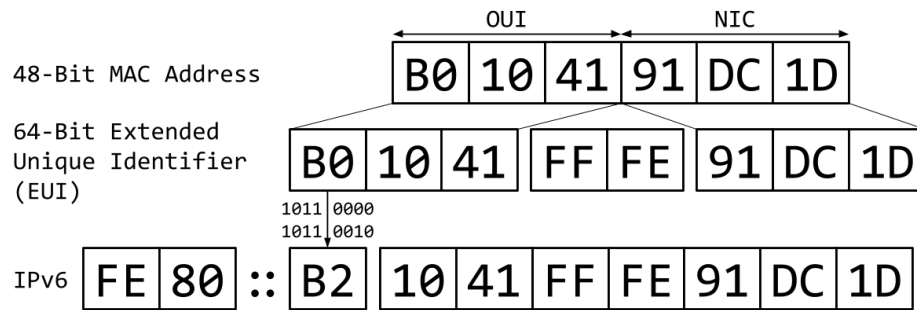
## 5.2.2 Self-Assignment of IPv6 Addresses

RFC 2373 [62], RFC 4291 [63], and RFC 4862 [85] describe methods for a machine to

---

<sup>48</sup>Of course, if the device is in developer mode, users can access any part of the filesystem using the password-less `sudo` Chrome OS provides them.

Figure 5.2: Process for self-assigning an IPv6 address derived from a 48-bit MAC address.



assign its network adapter an IPv6 address based on the adapter's MAC address. Figure 5.2 illustrates the process, beginning with the 48-bit MAC address, which is comprised of two parts. The first three octets (24 bits) are the Organizationally Unique Identifier (OUI) and the last three octets are specific to the NIC. The MAC address is converted to an Extended Unique Identifier (EUI) by splitting the MAC address into its two halves and inserting the 16-bit value 0xFFFE in the middle, which designates the EUI as having originated from a 48-bit MAC address.

Next, the seventh bit of the first octet in the EUI is flipped. The IEEE has designated this as the universal/local bit, which, when part of the OUI section of a MAC address, is always 0, meaning it was assigned in a local scope<sup>49</sup>. Thus, flipping it to 1 indicates a change to universal scope. To complete the conversion to an IPv6 address, the EUI is prefixed with 0xFE80, which identifies it as a link-local unicast address<sup>50</sup>.

Figure 5.3: The four APIs used and the information they provide. The first three belong to G Suite and the fourth comes from Chrome.



### 5.3 API Evaluation

Before attempting a reconstruction of the timeline, it is first necessary to have a clear understanding of *what* information is available to an examiner via the G Suite and Chrome APIs. So, in this section, I will enumerate the key data that G Suite and Chrome provide. Figure 5.3 illustrates the high-level details.

#### 5.3.1 G Suite API: Login Data

The first set of data available from the G Suite APIs is a log of all login attempts by users, returned as JSON data, as Listing 5.4 shows. Included in this list are all successful and unsuccessful authentication attempts, the account (email address) the individual attempted to access, the authentication method used, the time of the login attempt, and the originating IP address. The data returned is limited to the last 180 days of activity<sup>51</sup>.

<sup>49</sup>For more information on the universal/local bit, please refer to RFC 2373 [62].

<sup>50</sup>For definitions of the terms “link-local address” and “unicast address”, please refer to the Glossary.

<sup>51</sup>See <https://developers.google.com/admin-sdk/reports/v1/reference/activities>.



Listing 5.4: Sample login data retrieved from the G Suite API.

```
1 {"login": [{
    "actor": {"email": "mmabey@sefcom.org"},
    "events": [{
        "name": "login_success",
    5     "parameters": [{"name": "login_type", "value": "google_password"}],
        "type": "login"}],
    "id": {"time": "2017-09-14T21:16:02.000Z"},
    "ipAddress": "209.xxx.xxx.55"
  }, {
10  "actor": {"email": "mmabey@sefcom.org"},
    "events": [{
        "name": "login_failure",
        "parameters": [
15     {"name": "login_type", "value": "google_password"},
        {"name": "login_failure_type", "value": "login_failure_invalid_password"}
    ],
        "type": "login"}],
    "id": {"time": "2017-09-11T22:34:32.000Z"},
    "ipAddress": "209.xxx.xxx.55"
20  }
  ]}
```

### 5.3.2 G Suite API: List of Enrolled Devices

The second set of information available from the G Suite APIs is the list of enrolled Chrome OS devices, meaning the organization has purchased a license for each device and manually configured them to be associated with the organization’s G Suite account<sup>52</sup>. Enrolling a device also allows the organization to track and manage the device and its users. The G Suite API returns the list of enrolled devices as a JSON object, as Listing 5.5 shows.

The list of enrolled devices provided from the G Suite API includes a list of “active time ranges.” However, this term is a bit misleading, since the granularity is rather coarse

---

<sup>52</sup>For device enrollment instructions, see the related Google support article at <https://support.google.com/chrome/a/answer/1360534>.

Listing 5.5: Sample list of enrolled Chrome OS devices retrieved from the G Suite API.

---

```
1  [{
    "osVersion": "60.0.3112.80",
    "deviceId": "323xxxxx-xxxx-xxxx-xxxx-xxxxxxxxxf3f",
    "lastSync": "2017-09-11T23:41:59.738Z",
5   "macAddress": "b0104191dc1d",
    "firmwareVersion": "Google_Peppy.4389.117.0",
    "annotatedUser": "mmabey@sefcom.org",
    "serialNumber": "NXSxxxxxxxxxxxxxxxx600",
    "bootMode": "Verified",
10  "activeTimeRanges": [
    { "date": "2017-09-11",
      "activeTime": 30000
    },
    "lastEnrollmentTime": "2017-09-11T23:41:54.044Z",
15  "platformVersion": "9592.71.0 (Official Build) beta-channel peppy",
    "model": "Acer C720 Chromebook",
    "status": "ACTIVE",
    "orgUnitPath": "/"
  ]
}
```

---

and only indicates specific *days* the device was active and the total number of milliseconds it was active on that day.

### 5.3.3 G Suite API: Files and Metadata

The third set of data provided by G Suite is all the files and metadata stored on Drive, Photos, Plus, Gmail, and Contacts. Listing 5.6 is a sample of the JSON data returned from Drive. More metadata on each file in Drive is available, but for brevity I have included only a few key fields.

As Listing 5.6 shows, Drive stores information about the user that was the last to modify the file and when the modification took place. Drive also makes a distinction between the last time *any user* modified the file and last time the *impersonated user* modified the file. For Drive-specific file types (e.g., Google Slides), Drive also maintains a finer-

Listing 5.6: Abbreviated list of metadata fields for a file in Google Drive as returned from the G Suite API.

---

```
1  [{
    "id": "1AWlN5TJIGjte26cfy-E6W-cFIFGunl-A6VvDYQQkjlw",
    "name": "Untitled presentation",
    "createdTime": "2017-09-13T21:14:47.415Z",
    5  "viewedByMeTime": "2017-09-14T19:49:05.886Z",
    "modifiedByMeTime": "2017-09-14T19:49:05.886Z",
    "modifiedTime": "2017-09-14T19:49:05.886Z",
    "modifiedByMe": true,
    "viewedByMe": true,
    10  "ownedByMe": true,
    "owners": [{
        "permissionId": "03658834898655264426",
        "emailAddress": "mmabey@sefcom.org",
        "displayName": "Mike Mabey",
    15  "me": true
    }],
    "lastModifyingUser": {
        "permissionId": "03658834898655264426",
        "emailAddress": "mmabey@sefcom.org",
    20  "displayName": "Mike Mabey",
        "me": true
    },
    "version": "12",
    "trashed": false
    25  }]
```

---

grained revision history, available by making an additional API call for each file, using the `id` metadata field the file identifier.

### 5.3.4 Chrome API: User Info

The Chrome browser<sup>53</sup> provides three pieces of information important to the reconstruction of the timeline: (1) the email address of the user, (2) the start and end times the user is logged into the device, and (3) the MAC address of the device's network interface.

The user's email address is available from the `getProfileUserInfo()`<sup>54</sup> Chrome API function. This specifically retrieves the email address of the user signed into the Chrome browser. In the context of a Chrome OS device, this will be the email used to log into the device.

A user's activities on a device are logically separated by what I will term a *user session*. A user session is constrained by two points in time: when the user logs into a device and when the device becomes inactive. Two events will cause a device to be considered inactive: (1) the user logs out of the device or (2) the user suspends the device without logging out. Note that a user session does not correspond to *how much* the user actually uses the device. To capture the start and end times of a user session only requires a call to the JavaScript `Date()` function, which returns the current date and time, including the time zone.

Capturing the MAC address of the device is the most difficult of the three pieces of information to collect because of the safeguards employed by the Chrome browser. Web browsers have generally avoided providing API access to any unique hardware information about the system on which it is running (such as a MAC address), because, as one Stack

---

<sup>53</sup>Naturally, the web browser used by Chrome OS is Chrome, so all the APIs for Chrome are also available in Chrome OS.

<sup>54</sup>See <https://developer.chrome.com/apps/identity#method-getProfileUserInfo>.

Figure 5.7: Screenshot showing the MAC and IPv6 addresses of a Chrome OS device. Accounting for the derivation process described in Subsection 5.2.2, the two values are equivalent. Note that the MAC address listed matches the device shown in Listing 5.5 as being enrolled with G Suite.



Overflow user noted<sup>55</sup>, this “would basically be an undeletable cookie. For privacy reasons, browsers (and browser-extension APIs) don’t provide such identifiers.”

However, as a different Stack Overflow user shared<sup>56</sup>, Chrome OS seems to use the MAC address of its adapters to generate and self-assign an IPv6 address, a method consistent with the approach I describe in Subsection 5.2.2. My own observations support this, as can be seen in Figure 5.7, a screenshot taken from my test device. Accordingly, it is straightforward to retrieve the IPv6 address using the `chrome.system.network.getNetworkInterfaces()`<sup>57</sup> API call and then extract the MAC address.

The concern about “an undeletable cookie” is particularly relevant for devices that are *not* in developer mode (such as those enrolled in G Suite) because Chrome OS does not natively provide a way for users to change the MAC address used for an interface [12]. In other words, only when a Chrome OS device is in developer mode can a user change the information that could uniquely identify their device across the Web. Since G Suite

<sup>55</sup>See <https://stackoverflow.com/a/15556829>.

<sup>56</sup>See <https://stackoverflow.com/a/38994590>.

<sup>57</sup>See [https://developer.chrome.com/apps/system\\_network#method-getNetworkInterfaces](https://developer.chrome.com/apps/system_network#method-getNetworkInterfaces).

prevents users from putting enrolled devices in developer mode<sup>58</sup>, addressing this scenario is outside the scope of this research.

### 5.3.5 Summary of Data

Before discussing how I use the data from these APIs in my approach, the following is a summary of the data available from each API:

- The report on login attempts gives information on the email address of the accessed account, the time of authentication, and the originating IP address.
- The report on enrolled Chrome OS devices gives information on the days the device was active and the hardware, including serial number, MAC address, and device ID.
- Drive (and other Google services) provide access to file contents; the times each file was created, revised, or commented on; and the revision history (for select types of files).
- The Chrome browser provides the ability to record the device's MAC address, the user's email address, and the duration (start and end times) the user was active.

## 5.4 Beacon: Implementation

As I mentioned in Section 5.1, forensic examiners must seek to answer certain questions to ensure they properly reconstruct the incident timeline. However, to provide the answers to these questions, the data from the four APIs described in Section 5.3 must be explicitly

---

<sup>58</sup>To prevent users from switching a device to developer mode, administrators must turn on "Forced Re-enrollment" (on by default), which will halt the boot process if a user attempts to activate developer mode and restart in verified mode again.

linked together, which is only possible with the data captured from the Chrome API. Because of its essential role in the timeline reconstruction, I will now elaborate on this data's function, an implementation of a tool that gathers the data, and some related considerations.

There is one key difference between the G Suite API and the Chrome API. While administrators can query the G Suite API for data anytime, the data from the Chrome API must be collected on an ongoing basis because it can only be captured on the user's device and while the user is logged into it. Additionally, because an administrator or examiner cannot recover data stored locally on the encrypted device, once the information has been captured, it must be transmitted to a server for long-term storage, at which point the administrator or examiner may query the data at their pleasure. Hence, I call the tool that collects this data a *beacon* because it must periodically contact the server and transmit the data.

The beacon is implemented as a Chrome app intended to be installed on all of an organization's enrolled devices. G Suite makes this possible by allowing administrators to specify a set of apps and extensions that will be automatically force-installed on all enrolled devices<sup>59</sup>. Chrome OS implements the force-install policy in such a way that users cannot circumvent the process or disable or uninstall the extensions and apps. Because of the force-install policy, administrators can consistently rely on this data collection process to occur on all enrolled devices, which is critical to a forensic investigation. Furthermore, the beacon app runs in the background, so there is no direct indication to the user that the beacon is active. Thus, the approach of leveraging a Chrome app to acquire this data is suitable for any enterprise Chrome environment.

---

<sup>59</sup>See <https://support.google.com/chrome/a/answer/6306504>.

### 5.4.1 Collection and Retention

As I previously described in Subsection 5.3.4, the beacon collects three pieces of information about the user: (1) the email address of the user, (2) the start and end times the user is logged into the device, and (3) the MAC address of the device's network interface. The email address and the MAC address only have to be collected once from the Chrome API, after which they can be stored locally using Chrome's local storage API and retrieved quickly each time the beacon attempts to transmit the data to the collection server. Although an app's local storage is not encrypted, it is protected from other extensions or apps accessing it. Storing these two pieces of information using Chrome's local storage API has two additional benefits: (1) the number of asynchronous API calls for retrieving this information is reduced from two to one<sup>60</sup> and (2) it eliminates the need to calculate the MAC address each time.

As described in Subsection 5.2.2, verifying that an IPv6 address is derived from a MAC address is as simple as confirming that the fifth and fourth bytes from the end are 0xFF and 0xFE, respectively. In the worst case, the address will have the form of an address derived from a MAC address, but will not match any MAC addresses of the enrolled devices. If this should happen, although the user's login activity and files are still accessible, it will no longer be possible to link their activity to a specific device.

The frequency at which the beacon collects the time information depends on the level of granularity that the administrator requires of the collected data. An intuitive collection period is once per minute, which (1) corresponds with the granularity of the login data (see Listing 5.4) and (2) keeps the resource usage relatively low. In other words, if the time data

---

<sup>60</sup>Reducing the asynchronous API calls from two to one is only significant in terms of control flow, not necessarily in terms of performance.



is collected less than once per minute, it would be more difficult to correlate the beacon data with the login data; and if collected more frequently, it may begin to cause slight user experience degradation or create an unnecessary resource burden on the collection server. Ultimately, however, the frequency can and should be tuned by administrators, using real-world data to guide their decision.

Another customization parameter that administrators must consider is the length of time to retain the data from the beacon. The login data provided by G Suite only covers the last 180 days, so this may be a good starting point for a data retention policy. If an administrator believes there is cause to retain the data for longer than 180 days, it will be necessary to independently store the login data from G Suite to preserve the ability to link the login data with the beacon data. Of course, before implementing any data retention policy, administrators should consult with competent legal council to ensure the policy complies with applicable laws.

#### 5.4.2 Caching

The beacon has a data caching feature that is critical to its implementation. Caching the time data provides the beacon an important level of flexibility with the conditions under which it is expected to operate, while simultaneously achieving greater compliance with the Completeness Rule of Evidence. Because this is such an important feature, I will now present the details of the beacon's caching mechanism.

When the beacon collects the information from the device on which it is running, it immediately attempts to transmit the data to the collection server. If the Chrome OS device is online and can reach the server, the beacon only needs to include a single time value in its transmission, the current time. However, if the device is in offline mode or

Table 5.8: Time data caching rules, where  $t_n$  is the current time,  $t_{n-p}$  is the current time minus the collection period  $p$ ,  $t_0$  is the first time cached in the current user session, and  $t_{n-i}$  (where  $i > p$ ) is a cached time that differs from  $t_n$  by more than the period of collection. If there is a gap in the times, the beacon stores  $t_n$  in a new user session array.

1. No data cached	$[[ -, - ]]$	$\rightarrow$	$[[ -, t_n ]]$
2. One time cached	$[[ -, t_{n-p} ]]$	$\rightarrow$	$[[ t_{n-p}, t_n ]]$
3. >1 time cached	$[[ t_0, t_{n-p} ]]$	$\rightarrow$	$[[ t_0, t_n ]]$
4. Gap in time	$[[ t_0, t_{n-i} ]]$	$\rightarrow$	$[[ t_0, t_{n-i} ], [ -, t_n ]]$

some other circumstance occurs that prevents the beacon from contacting the server, it must cache the time data until the device goes back online because, although the user may not be connected, they may still be active on the device, using the offline mode of their extensions and creating or changing files. Additionally, the purpose of capturing this data is to know precisely what the beginning and ending times are of the user session, which does not correlate at all to whether or not the user is online.

The beacon caches time data in an array where each array element corresponds to a user session. Each user session is represented as an array of length 2, which stores the first and most recent times the user was active, respectively. The beacon stores new time data according to the following conditional statements, which Table 5.8 illustrates.

1. The first time the beacon is unable to transmit data to the server (i.e., it has not yet cached any other time data), it stores the current time,  $t_n$ , as the last element in the user session array.
2. If the beacon has cached a single time,  $t_{n-p}$ , that differs from  $t_n$  by the period,  $p$ , it moves  $t_{n-p}$  to be the first element in the array, indicating it is the first time of the user session, and stores  $t_n$  as the second element.
3. If the beacon has cached more than one time (meaning both user session array

elements have times), as long as the most recent time differs from  $t_n$  by the period,  $p$ , the beacon stores  $t_n$  as the most recent time.

4. If the most recent time differs from  $t_n$  by more than  $p$ , the beacon considers this a new user session, creates a new session array of length 2, and stores  $t_n$  as the second element. This scenario occurs if the user suspends or logs off of the machine for longer than the collection period.

The beacon does not require much storage space when caching the time data, because only two time stamps are required to represent each user session and the likelihood of many user sessions starting and stopping all while being offline is unlikely<sup>61</sup>. However, there is still a shortcoming to this approach that is a direct result of how Chrome OS devices function. In the event that the device is offline while one user, user A, completes their session and logs off (meaning the beacon still has cached data), it is possible that the next user to log in will be a different user, user B. Even if the device is connected to the Internet while user B is logged in, the data cached by the beacon for user A is inaccessible and cannot be transmitted. In general, it is a good thing that Chrome OS maintains this division of user data, but in the case of the beacon, it means that user A will have to log into the device when it is connected to the Internet for the beacon to be able to upload its data.

In the event that user A never does log into the device when it can connect to the Internet, the cached time data will remain inaccessible to the administrator. However, the impact of not having this data during an investigation will be minimal for the following reasons. If the device is offline, the user cannot edit files with any online services, which means they are limited to editing locally stored files. However, none of the edits to locally

---

<sup>61</sup>Since the primary usage model of Chrome OS devices relies on an Internet connection, users are most likely to be connected when using them.

stored files will be synced to any online service, which means none of the file modifications (e.g., as stored in metadata in Drive) will correspond to the user session during the timeline reconstruction (see Section 5.5). Of course, the user may still physically transfer files using a USB flash drive or a similar device, but this is outside the scope of this research.

### 5.4.3 Evaluation



From an administrator's point of view, one important factor to take into consideration when deploying Beacon on all enrolled devices is what impact it will have on users and the battery life of their device. To address this and evaluate Beacon's performance, I conducted an experiment to determine the impact Beacon has on a device's battery life using `battery_test`, a utility that comes with Chrome OS and is available in `crash` even when the device is not in developer mode<sup>62</sup>. Below are the details of the experimental setup:

- Device model: Acer C720 Chromebook
- Chrome OS version: 61.0.3163.123
- Test length: 10 minutes (600 seconds)
- Beacon transmission period: 60 seconds
- Battery status: charged, but not plugged in
- Extensions: all disabled, except Beacon
- Screen brightness: off
- Peripherals plugged in: none
- Open tabs: `crash` for running the test, no others

---

<sup>62</sup>For more details on `crash` and developer mode, see Subsection 2.2.7.

Table 5.9: Results of the Beacon battery use test. The total anticipated impact that Beacon will have on battery life when WiFi is enabled is 6.25%, and less than that when WiFi is turned off.

	WiFi Off 	WiFi On 
No Beacon	0.46%	0.48%
Beacon	0.46%	0.51%
Additional power used	<0.01%	0.03%
Total battery life impact	—	6.25%

I conducted the experiment under four conditions, with WiFi off and Beacon *not* installed, with WiFi on and Beacon *not* installed, with WiFi off and Beacon installed, and with WiFi on and Beacon installed. The two tests without Beacon served as controls for the device’s regular battery use, which is further stabilized by disabling all extensions, turning the screen brightness all the way down, and not having any other tabs open other than the one performing the test. By setting the transmission period to 60 seconds (which conforms with the recommendations in Subsection 5.4.1) and the test length to 10 minutes, Beacon attempted to transmit its data ten times during the test.

Table 5.9 summarizes the evaluation results. When WiFi is off, the Chromebook used nearly the same amount of battery when Beacon was installed as when it was not installed. Since `battery_test` only reports numbers to the nearest hundredth of a percent, it is difficult to say exactly how much power Beacon uses in this state without constructing an entirely new experiment, such as measuring the power use with an external device. However, after repeating the test a few times, I took the lowest power usage for when Beacon was *not* installed and the highest power usage for when Beacon *was* installed. With WiFi turned on, Beacon used an additional 0.03% of the battery, which is 6.25% more than when it was not installed.

The fact that Beacon’s battery usage was negligible when WiFi was turned off indicates that the increased power used when WiFi was on was a result of Beacon using the network

to transmit the data. This coincides with the typical power consumption of mobile devices, which is impacted much more by network devices than by the CPU or RAM during periods of higher (i.e., not standby) network activity [16]. However, the proof of concept implementation transmits the data immediately after collecting it, an approach not based on this experimental data. To minimize the battery life impact, Beacon could be tuned to only transmit the data in batches and thereby reduce the number of network requests it makes.

## 5.5 Timeline Reconstruction and Visualization

In the previous sections, I discussed what timeline data is available from APIs (Section 5.3) and how I collect this data (Section 5.4). In this section, I describe **Gripper**, a tool that builds upon these initial steps and (1) reconstructs the timeline from the raw data provided by the APIs and (2) uses the timeline data to create a visualization of the user's activity.

### 5.5.1 Reconstruction

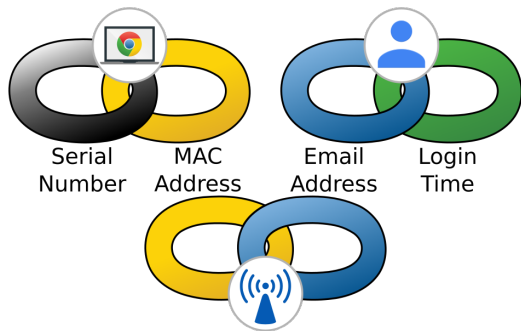
In Section 5.1, I listed four questions that examiners need to answer when conducting a proper timeline reconstruction. The first two questions require an explicit link between a user session and a specific device:

- *Who used the device?*
- *When did the individual use the device?*

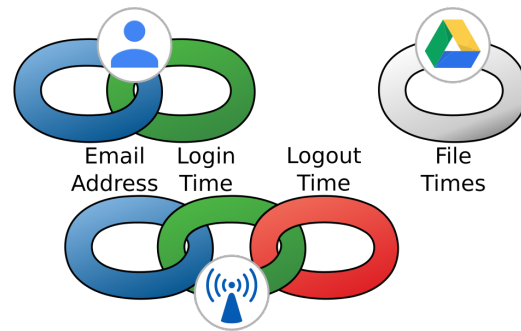
To identify who used a specific device and when, Gripper combines the data from (1) the report on devices, (2) the login report, and (3) the beacon (see Figure 5.10a). The

Figure 5.10: Reconstructing the timeline requires combining data from the four sources.

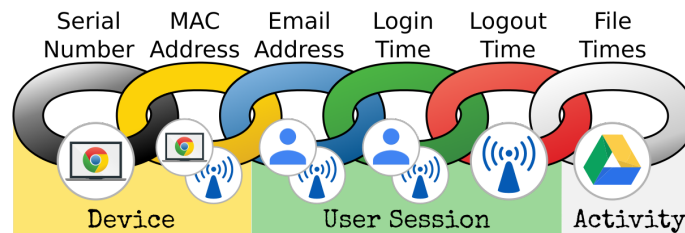
(a) The data required to know what device the user used and when.



(b) The data required to know what the user did while logged in.



(c) With the data from the beacon, the data from the other three sources can be linked together.



devices report lists the serial number and MAC address, which uniquely identify a specific device. The login report lists when a user logged into a specific email address. The beacon connects these two sets of data by providing the MAC address of the device, which should match an entry in the device report, and the user's email address, which should match the login report.

The last two questions from Section 5.1 require an explicit link between a user session and actions executed by the user:

- *What did the individual do while on the device?*
- *What files may contain inculpatory/exculpatory evidence regarding the incident based on when they were created and modified?*

To enumerate the user's activities while on the device and what files may be relevant

to the investigation, Gripper combines the data from (1) the login report, (2) the metadata of the files in Google Drive<sup>63</sup>, and (3) the beacon (see Figure 5.10b). The login report lists when a user logged into a specific email address. The metadata from Drive shows what actions the user did with respect to creating, modifying, and deleting files. The beacon connects these two sets of data by giving a more specific range of when the user was actually logged into the device.

With the bounds provided by the user session information, Gripper can filter through the metadata from Drive and return information on just those files that were created, modified, or deleted during the session. Gripper's command line options also allow the user to specify the start and end dates to acquire. When specifying these dates, Gripper supports various date/time formats, including ISO 8601, RFC 3339 [90], as well as natural language phrases<sup>64</sup>, such as "three days ago."

### 5.5.2 Visualization

To create a visualization of the timeline data, I recognized the importance of giving the examiner both a high-level view of a user's activity while still allowing the examiner to view more specific, granular information for a particular window of time. I also knew that examiners would find it helpful to have integrated into the visualization tool some method of highlighting periods of increased or unusually high activity as well as a simple method of showing what the user's typical usage behavior is.

With these objectives in mind, I designed an interactive heat map of a user's hourly

---

<sup>63</sup>Google Drive is just one example of a source for discovering the user's activities. The examiner may also query Google Photos, Google Plus, Gmail, and Google Contacts as described in Subsection 5.3.3.

<sup>64</sup>The support for natural language phrases is provided by Maya, the Python library Gripper uses to handle all time parsing and time zone comparisons.



Figure 5.11: Visualization of files created by a user in Google Drive, separated by hour. The complete data set has been reduced to a specific time window of early January 2017 to August 2017 using the window selection tool at the bottom. The data for a specific data point is available when the user hovers over it. In the sample data set, the user created 4,122 files between 0800 and 0859 on May 2, 2017.



activity, as seen in the sample in Figure 5.11. By taking this approach, Gripper is able to show the user's entire user history or just their activity for a few weeks, giving control to the examiner via (1) slider widgets that define the upper and lower bound of the data displayed and (2) toggles (top left corner) that adjust the view window to predefined sizes. The cell colors of the heat map clearly distinguish low levels of activity from high levels, with colors that use an exponential scale to help unusually high levels of activity to stand out. Additionally, by giving cells with no activity a white color, the user's typical behavior with respect to the time of day is clearly evident.

To create the activity heat map visualization, this proof-of-concept implementation uses Plotly<sup>65</sup>, a Python library for creating graphs and charts. At this point, Gripper does not correlate the timeline data with information about devices or login sessions.

### 5.5.3 Providing Access to Law Enforcement

One important part of Gripper's configuration and setup is the creation of an OAuth token, created by the G Suite administrator, that allows Gripper to authenticate to Google without prompting Gripper's user for a password when acquiring data via the APIs. The token is specific to the G Suite account for which it was created and administrators can create multiple tokens that are all associated with their G Suite account.

Because an OAuth token is just a piece of data stored in a file, it is possible for an administrator to create an API access key specifically for law enforcement when necessary. Because the key is controlled by the organization's G Suite administrator, they can invalidate the key after the investigation without disrupting their own access via Gripper.

---

<sup>65</sup>See the Plotly web site: <https://plot.ly/>.

## 5.6 Discussion

My approach is specifically designed to be used on enterprise Chrome OS devices, meaning they are enrolled with either G Suite or G Suite for Education. While not all Chrome OS devices fall into this category, I believe it is still worthwhile to provide this kind of approach for two reasons. First, as I mentioned in Section 5.1, it is not possible to retrieve timeline data directly from a Chrome OS device, so using the APIs to provide similar information is a reasonable alternative.

Second, in spite of its limitations, my approach fills a need for enterprises to acquire data about an incident timeline with explicit links between a device, a user session, and user activity. The fact that I do not rely on the device itself can actually be an advantage for administrators, because they do not need to locate the device and remove it from use to examine its data, eliminating the time required to do so and the inconvenience it may cause to users.

Because my approach focuses on using Google's APIs to acquire data, it is inherently limited to detecting user activity within the Google ecosystem of services. For example, my approach is not capable of detecting edits to files not stored in Google Drive, such as locally created files stored outside Drive's cache (e.g., the Downloads folder). Neither am I attempting to retrieve data on user activity with non-Google services, such as Microsoft Office Online. While approaches that address these scenarios would certainly complement my approach, it is outside the scope of this work.

## 5.7 Related Work

My approach for using a proxy for acquiring forensic evidence is not without precedent. Grispos et al. demonstrated how both iOS and Android mobile devices retain information from cloud storage services that act as a proxy for the data stored in the cloud [56]. In a way, my work takes the opposite approach, using a cloud service (G Suite APIs) as a proxy for what evidence is on the device.

Roussev et al. introduced a way to acquire forensic data directly from cloud storage services using their APIs [100]. Many aspects of their work is similar to mine in that we both rely on a service provider's API for the raw forensic data. However, the purpose of their approach is to acquire the contents of all the user's files for analysis. Because of this, Roussev et al. take a generic approach so as to be able to support as many cloud storage providers using the same tool. By contrast, the focus of my approach is to reconstruct the timeline of events as they pertain to enterprise Chrome OS devices. As such, it only makes sense for me to support the G Suite APIs and to give greater attention to file metadata than file contents.

Tassone et al. analyzed the suitability of a variety of visualization techniques to communicate different types of information in a forensic examination [108]. As part of their analysis, they evaluated each technique against a number of criteria, such as repeatability, lucidity (ease for a user to understand), and the ability of the user to quickly identify anomalies. My work to create effective visualizations of timeline data may be enhanced by adopting some of the results from this research.

Mabey et al. were the first to attempt a forensic analysis of the residual evidence on web thin clients without depending on root access to the device or being able to decrypt the contents of the hard drive [79]. While their efforts provided a good first step toward

comprehensive forensics for web thin client, they did not attempt a reconstruction of the timeline. My approach, then, is complementary to theirs.

## 5.8 Summary

In this chapter, I have presented my approach for reconstructing and visualizing timeline data for enterprise web thin clients, specifically Chrome OS devices. To accomplish this, I first evaluated the data available from APIs provided by Google, then devised a way to acquire time-related data from these APIs and link them together to make explicit links between specific devices, user sessions, and user activity. I also created a visualization tool for the timeline data that provides an examiner insight into a user's typical usage patterns while also highlighting activity spikes.

Gripper and its related components still have many unexplored research opportunities, including finding ways to correlate data on multiple users in the same G Suite organization or separate organizations. Also, because Gripper uses such a generic structured format to store the timeline data (JSON), there is significant potential for developing new visualization methods without changing anything else about how Gripper works or processes data. Finally, the proof-of-concept implementation currently only acquires data from Google Drive, but adding support for the other services for which G Suite has APIs is already partially implemented and could be completed easily.

## Chapter 6

### CONCLUSION

Throughout this dissertation, I have discussed many ways the Web has transformed technology, as well as the resulting challenges for forensic examiners to adequately discover, acquire, store, and analyze evidence from this domain. I have proposed a framework that (1) accommodates the distinct traits of web environments, (2) provides examiners with practical guidance for how to approach investigations involving web-based evidence, and (3) emphasizes four critical areas for which examiners and forensic tools should make provision: **F1 Evidence Discovery and Acquisition**, **F2 Analysis Space Reduction**, **F3 Timeline Reconstruction**, and **F4 Structured Formats**.

To show the value of my framework, I applied it to web thin clients, specifically Chrome OS devices, in the ways listed below.

**F1: Evidence Discovery and Acquisition** In my work on identifying extensions installed on encrypted Chrome OS devices (Chapter 4), I addressed this framework component in a couple different ways. First, it was essential to know what to look for when examining a Chrome OS device, which the Enrollment phase of dbling accomplishes (Subsection 4.3.1). In the Enrollment phase, I gather the data necessary to draw conclusions from the information on the evidence. Specifically, this requires crawling the Chrome Web Store, downloading the extensions, and generating a profile for each one. I then store all this information and these artifacts in a database to facilitate efficient comparison.

Second, I devised a method of isolating and acquiring data from web thin clients in the Identification phase of dbling (Subsection 4.3.2). In the Identification phase, I leverage my understanding of Chrome OS to acquire a forensic copy of a device's hard drive, isolate

the useful information stored on the disk, and identify the extensions installed. This process effectively transforms encrypted data into usable information that was previously inaccessible.

In my work on timeline reconstruction for enterprise web thin clients (Chapter 5), it was first necessary to discover and enumerate the data available via the G Suite and Chrome APIs and determine what data are relevant to the reconstruction of an incident timeline (Section 5.3). With this knowledge, I then devised an approach to acquire the timeline data via the APIs.

**F2: Analysis Space Reduction** The Identification phase of dbling (Subsection 4.3.2) helps narrow down the analysis space of the evidence on a web thin client by (1) isolating the useful information stored on the disk (as mentioned previously), and (2) identifying the extensions installed on the device, thereby reducing the search space down from all possible extensions.

While reconstructing the timeline of a Chrome OS device, I reduced the analysis space in a few different ways. First, with the knowledge of what data available from the G Suite and Chrome APIs are relevant to the timeline (as mentioned above), the next step was to focus only on the relevant data, effectively reducing the total data to be analyzed.

Second, by correlating the time data from the four APIs, I reduce the analysis space even further by removing irrelevant entries, meaning those entries that do not correlate with data from the other sources.

Finally, my timeline reconstruction tool accepts command line options that allow the examiner to manually reduce activity window to be analyzed before any API calls are made.

**F3: Timeline Reconstruction** One important observation I made in my research on Chrome OS is that, although it stores data relevant to timeline reconstruction, the timeline data is inaccessible due to how the operating system encrypts the data. Because it is infeasible to retrieve the timeline data directly from the device, it is only possible to reconstruct the timeline by acquiring data from other sources. In my approach, the sources I use are the G Suite and Chrome APIs.

To use the G Suite and Chrome APIs to reconstruct the timeline, it is necessary to correlate data about the devices, user sessions, and activity. Although the G Suite APIs provide information on all three of these subjects, correlating them is only possible by including data acquired from the Chrome APIs. With data from all four sources, I can identify when a user logged into a specific device and what actions they performed while using it, and thus complete the reconstruction of the timeline for the device that was not possible by analyzing the device alone (Subsection 5.5.1).

With the correlated timeline data, I also implemented Gripper (Subsection 5.5.2), a visualization tool that highlights a user's activity in a heat map with cooler and hotter colors indicating lower and higher levels of activity, respectively, which gives visual contrast to any anomalous activity. Examiners can also adjust the time frame shown to view a limited period of interest or the entire activity history.

**F4: Structured Formats** In the Export phase of dbling (Subsection 4.3.3), I created a new structured format to store the analysis results that I call MERL. By structuring dbling's output in this manner, I help facilitate inter-tool compatibility and make it easier for forensic tool developers to incorporate dbling into their workflow.

Similarly, the data returned by the G Suite and Chrome APIs is structured as JSON, which Gripper can also export. Because Gripper uses and stores the timeline data in such



a generic format, other tools can use the same data to create different visualizations or perform different analyses without having to first transform the data to a different format.

## 6.1 Future Work

Going forward, there are still many opportunities to advance the research I presented in this dissertation. It would be interesting to conduct a study with digital forensic examiners to evaluate the real-world benefits of my framework for web environment forensics (Chapter 3). Additionally, I listed many research opportunities related to the four framework components that, if further developed, would also greatly benefit examiners.

The dbling project could benefit from further refinements to its extension profile creation process (Subsection 4.3.1) so that it even more precisely replicates how Chrome OS unpacks and installs extensions. With such improvements, the accuracy of the identification phase (Subsection 4.3.2) would also improve.

My preliminary implementation of Gripper only supports acquiring data on user activity from Google Drive for reconstructing a timeline. However, G Suite also provides APIs for other services, including Google Photos, Google Plus, Gmail, and Google Contacts. Extending Gripper to support these services and, additionally, correlate the data acquired from all of them would (1) increase the value of the data that Gripper provides examiners and (2) provide a more complete reconstruction of the timeline.

## REFERENCES

- [1] ABI Research. *ABI Research Forecasts Chromebooks to Top 8 Million Unit Shipments and Lead 2015 Global Growth in Notebook PCs*. Dec. 2015. URL: <https://www.abiresearch.com/press/abi-research-forecasts-chromebooks-top-8-million-u/>.
- [2] ABI Research. *The Internet of Things Will Drive Wireless Connected Devices to 40.9 Billion in 2020*. Aug. 2014. URL: <https://www.abiresearch.com/press/the-internet-of-things-will-drive-wireless-connect/>.
- [3] Atif Ahmad and Anthonie B Ruighaver. “Towards Identifying Criteria for the Evidential Weight of System Event Logs”. In: *2nd Australian Computer Network & Information Forensics Conference, November 25th 2004, Perth, Western Australia, Forensic Computing - Evidence on the move from Desktops to Networks, Conference Proceedings*. School of Computer and Information Science, Edith Cowan University, Western Australia, 2004, pp. 40–47. DOI: 10.1.1.69.8461.
- [4] Gail-Joon Ahn. *Computer Forensics — Basic Concepts*. Lecture Slides. Arizona State University CSE 469. Jan. 2015.
- [5] W Alink et al. “XIRAF – XML-based indexing and querying for digital forensics”. In: *Digital Investigation* 3 (Sept. 2006), pp. 50–58. DOI: 10.1016/j.diin.2006.06.016. URL: <http://linkinghub.elsevier.com/retrieve/pii/S1742287606000776>.
- [6] Cory Altheide, Claudio Merloni, and Stefano Zanero. “A Methodology for the Repeatable Forensic Analysis of Encrypted Drives”. In: *Proceedings of the 1st European Workshop on System Security*. EUROSEC '08. New York, NY, USA: ACM, 2008, pp. 22–26. DOI: 10.1145/1355284.1355289.
- [7] Lee Badger et al. “Cloud Computing Synopsis and Recommendations Recommendations of the National Institute of Standards and Technology”. In: *NIST Special Publication (2012) 800-146*. URL: <http://csrc.nist.gov/publications/nistpubs/800-146/sp800-146.pdf> (2012).
- [8] M Tariq Bandy. “Analysing E-Mail Headers for Forensic Investigation”. In: *Journal of Digital Forensics, Security, and Law* 6.2 (2011), pp. 49–64. URL: <http://ojs.jdfsl.org/index.php/jdfsl/article/view/34>.
- [9] Sean Barnum. *Standardizing Cyber Threat Intelligence Information with the Structured Threat Information eXpression (STIX™)*. Tech. rep. The MITRE Corporation, Feb. 2014. URL: <http://stixproject.github.io/getting-started/whitepaper/>.

- [10] Tim Berners-Lee and Mark Fischetti. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web*. First. Harper Business, Nov. 2000. URL: <http://dblp2.uni-trier.de/rec/bib/books/daglib/0020403>.
- [11] Dominik Birk and Christoph Wegener. “Technical Issues of Forensic Investigations in Cloud Computing Environments”. In: *2011 Sixth IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering*. IEEE, May 2011, pp. 1–10. DOI: 10.1109/SADFE.2011.17. URL: <http://ieeexplore.ieee.org/document/6159124/>.
- [12] Heather Bloomer. *How To Change the Mac Address on a Chromebook*. Tech Junkie. Dec. 2016. URL: <https://www.techjunkie.com/change-mac-address-chromebook/> (visited on 10/17/2017).
- [13] Gabriel Brangers. *‘Fievel’, Chrome OS and the Internet of Things*. Oct. 2016. URL: <https://chromeunboxed.com/fievel-chrome-os-and-the-internet-of-things/> (visited on 10/12/2016).
- [14] Florian Buchholz and Courtney Falk. “Design and Implementation of Zeitline: A Forensic Timeline Editor”. In: *DFRWS (2005)*, pp. 1–7. URL: [https://users.cs.jmu.edu/buchhofp/publications/zeitline\\_dfrws.pdf%20http://www.dfrws.org/sites/default/files/session-files/paper-design\\_and\\_implementation\\_of\\_zeitline\\_-\\_a\\_forensic\\_timeline\\_editor.pdf](https://users.cs.jmu.edu/buchhofp/publications/zeitline_dfrws.pdf%20http://www.dfrws.org/sites/default/files/session-files/paper-design_and_implementation_of_zeitline_-_a_forensic_timeline_editor.pdf).
- [15] Florian Buchholz and Eugene Spafford. “On the role of file system metadata in digital forensics”. In: *Digital Investigation* 1.4 (Dec. 2004), pp. 298–309. DOI: 10.1016/j.diin.2004.10.002. URL: <http://linkinghub.elsevier.com/retrieve/pii/S1742287604000829>.
- [16] Aaron Carroll and Gernot Heiser. “An Analysis of Power Consumption in a Smartphone”. In: *2010 USENIX Annual Technical Conference, Boston, MA, USA, June 23-25, 2010*. Ed. by Paul Barham and Timothy Roscoe. USENIX Association, 2010. URL: <https://www.usenix.org/conference/usenix-atc-10/analysis-power-consumption-smartphone>.
- [17] Andrew Case et al. “FACE: Automated digital evidence discovery and correlation”. In: *Digital Investigation* 5 (Sept. 2008), S65–S75. DOI: 10.1016/j.diin.2008.05.008. URL: <http://linkinghub.elsevier.com/retrieve/pii/S1742287608000340>.
- [18] Eoghan Casey, Greg Back, and Sean Barnum. “Leveraging CybOX™ to standardize representation and exchange of digital forensic information”. In: *Digital Investigation* 12.S1 (Mar. 2015), S102–S110. DOI: 10.1016/j.diin.2015.01.014. URL: <http://dx.doi.org/10.1016/j.diin.2015.01.014>.

- [19] Eoghan Casey and Gerasimos J Stellatos. “The impact of full disk encryption on digital forensics”. In: *ACM SIGOPS Operating Systems Review* 42.3 (Apr. 2008), p. 93. DOI: 10.1145/1368506.1368519. URL: <http://portal.acm.org/citation.cfm?doid=1368506.1368519>.
- [20] Eoghan Casey et al. “Advancing coordinated cyber-investigations and tool interoperability using a community developed specification language”. In: *Digital Investigation* 22 (Sept. 2017), pp. 14–45. DOI: 10.1016/j.diin.2017.08.002. URL: <http://linkinghub.elsevier.com/retrieve/pii/S1742287617301007>.
- [21] Aniello Castiglione, Alfredo De Santis, and Claudio Soriente. “Taking advantages of a disadvantage: Digital forensics and steganography using document metadata”. In: *Journal of Systems and Software* 80.5 (2007), pp. 750–764.
- [22] Yoan Chabot et al. “A complete formalized knowledge representation model for advanced digital forensics timeline analysis”. In: *Digital Investigation* 11 (2014), S95–S105. DOI: 10.1016/j.diin.2014.05.009. URL: <http://www.sciencedirect.com/science/article/pii/S1742287614000528>.
- [23] Kai Chen, Peng Liu, and Yingjun Zhang. “Achieving Accuracy and Scalability Simultaneously in Detecting Application Clones on Android Markets”. In: *Proceedings of the 36th International Conference on Software Engineering. ICSE 2014*. New York, NY, USA: ACM, 2014, pp. 175–186. DOI: 10.1145/2568225.2568286.
- [24] Kai Chen et al. “Finding Unknown Malice in 10 Seconds: Mass Vetting for New Threats at the Google-Play Scale”. In: *24th USENIX Security Symposium (USENIX Security 15)*. USENIX Association, Aug. 2015, pp. 659–674. URL: <https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/chen-kai>.
- [25] Chrome Developer Documentation. *CRX Package Format*. 2016. URL: <https://developer.chrome.com/extensions/crx>.
- [26] Chrome Developer Documentation. *What are extensions?* 2016. URL: <https://developer.chrome.com/extensions>.
- [27] Chromium Projects. *Chromium OS FAQ: What’s the difference between Chromium OS and Google Chrome OS?* 2016. URL: <https://www.chromium.org/chromium-os/chromium-os-faq#TOC-What-s-the-difference-between-Chromium-OS-and-Google-Chrome-OS->.

- [28] Chromium Projects. *Developer Information for Chrome OS Devices*. 2017. URL: <https://www.chromium.org/chromium-os/developer-information-for-chrome-os-devices> (visited on 08/25/2017).
- [29] Chromium Projects. *Disk Format*. 2016. URL: <https://www.chromium.org/chromium-os/chromiumos-design-docs/disk-format>.
- [30] Chromium Projects. *Firmware Verified Boot Crypto Specification*. 2016. URL: <https://www.chromium.org/chromium-os/chromiumos-design-docs/verified-boot-crypto>.
- [31] Chromium Projects. *Poking around your Chrome OS Device*. 2017. URL: <https://www.chromium.org/chromium-os/poking-around-your-chrome-os-device> (visited on 08/25/2017).
- [32] Chromium Projects. *Protecting Cached User Data*. 2016. URL: <https://www.chromium.org/chromium-os/chromiumos-design-docs/protecting-cached-user-data>.
- [33] Chromium Projects. *Security Overview*. 2016. URL: <http://www.chromium.org/chromium-os/chromiumos-design-docs/security-overview>.
- [34] Chromium Projects. *Verified Boot*. 2016. URL: <https://www.chromium.org/chromium-os/chromiumos-design-docs/verified-boot>.
- [35] M.I. Cohen. “PyFlag – An advanced network forensic framework”. In: *Digital Investigation* 5 (Sept. 2008), S112–S120. DOI: 10.1016/j.diin.2008.05.016. URL: <http://www.sciencedirect.com/science/article/pii/S1742287608000418>.
- [36] Michael Cohen, Simson Garfinkel, and Bradley Schatz. “Extending the advanced forensic format to accommodate multiple data sources, logical evidence, arbitrary information and forensic workflow”. In: *Digital Investigation* 6 (Sept. 2009), S57–S68. DOI: 10.1016/j.diin.2009.06.010. URL: <http://linkinghub.elsevier.com/retrieve/pii/S1742287609000401>.
- [37] Common Digital Evidence Storage Format Working Group. “Survey of Disk Image Storage Formats”. Sept. 2006. URL: <http://old.dfrws.org/CDESF/survey-dfrws-cdesf-diskimg-01.pdf>.
- [38] George Corbin. “The Google Chrome operating system forensic artifacts”. Capstone Report. Utica College, 2014.

- [39] Andrew Cunningham. *Google is killing Chrome apps on Windows, Mac, and Linux*. Ars Technica. Aug. 2016. URL: <http://arstechnica.com/gadgets/2016/08/google-is-killing-chrome-apps-on-windows-mac-and-linux/> (visited on 09/01/2016).
- [40] James Darvell. *Firefox OS*. Jan. 2016. URL: <http://www.linuxjournal.com/content/firefox-os>.
- [41] Josiah Dykstra and Alan T Sherman. “Acquiring forensic evidence from infrastructure-as-a-service cloud computing: Exploring and evaluating tools, trust, and techniques”. In: *Digital Investigation* 9 (Aug. 2012), S90–S98. DOI: 10.1016/j.diin.2012.05.001. URL: [http://ac.els-cdn.com/S1742287612000266/1-s2.0-S1742287612000266-main.pdf?\\_tid=19658544-8dd7-11e7-aeac-00000aacb35f&acdnat=1504134245\\_1832dc04af09a21baa95c7046ec3730a%20http://linkinghub.elsevier.com/retrieve/pii/S1742287612000266](http://ac.els-cdn.com/S1742287612000266/1-s2.0-S1742287612000266-main.pdf?_tid=19658544-8dd7-11e7-aeac-00000aacb35f&acdnat=1504134245_1832dc04af09a21baa95c7046ec3730a%20http://linkinghub.elsevier.com/retrieve/pii/S1742287612000266).
- [42] Soowoong Eo et al. “A Phase of Deleted File Recovery for Digital Forensics Research in Tizen”. In: *IT Convergence and Security (ICITCS), 2015 5th International Conference on*. IEEE. 2015, pp. 1–3.
- [43] Dave Evans. *The Internet of Things [INFOGRAPHIC]*. Blog. July 2011. URL: <http://blogs.cisco.com/diversity/the-internet-of-things-infographic>.
- [44] Kevin D Fairbanks. “An analysis of Ext4 for digital forensics”. In: *Digital Investigation* 9 (Aug. 2012), S118–S130. DOI: 10.1016/j.diin.2012.05.010. URL: <http://linkinghub.elsevier.com/retrieve/pii/S1742287612000357>.
- [45] Kevin D Fairbanks, Christopher P Lee, and Henry L Owen III. “Forensic implications of ext4”. In: *Proceedings of the sixth annual workshop on cyber security and information intelligence research*. ACM. 2010, p. 22.
- [46] FindLaw. *Summary of the Rules of Evidence*. URL: <http://corporate.findlaw.com/litigation-disputes/summary-of-the-rules-of-evidence.html> (visited on 11/07/2017).
- [47] ForensicsWiki. *Computer Forensics: Artifact*. 2015. URL: [http://forensicswiki.org/wiki/Computer\\_forensics#Artifact](http://forensicswiki.org/wiki/Computer_forensics#Artifact).
- [48] ForensicsWiki. *Disk image*. URL: [http://forensicswiki.org/wiki/Disk\\_image](http://forensicswiki.org/wiki/Disk_image) (visited on 11/07/2017).
- [49] Futuresource Consulting Ltd. *Sales of Mobile PCs into the US K-12 Education Market Continue to Grow*. Mar. 2017. URL: <https://www.futuresource-consulting.com/Press-K-12-Education-Market-Qtr4-0317.html> (visited on 10/17/2017).

- [50] Simson Garfinkel. “Digital forensics XML and the DFXML toolset”. In: *Digital Investigation* 8.3-4 (Feb. 2012), pp. 161–174. DOI: 10.1016/j.diin.2011.11.002. URL: <http://www.sciencedirect.com/science/article/pii/S1742287611000910>.
- [51] Simson L. Garfinkel. “AFF: A New Format for Storing Hard Drive Images”. In: *Communications of the ACM* 49.2 (Feb. 2006), p. 85. DOI: 10.1145/1113034.1113076. URL: <http://portal.acm.org/citation.cfm?doid=1113034.1113076>.
- [52] Simson L Garfinkel. “Automating Disk Forensic Processing with SleuthKit, XML and Python”. In: *Systematic Approaches to Digital Forensic Engineering, 2009. SADFE '09. Fourth International IEEE Workshop on*. May 2009, pp. 73–84. DOI: 10.1109/SADFE.2009.12.
- [53] Simson L Garfinkel. “Digital forensics research: The next 10 years”. In: *Digital Investigation* 7 (Aug. 2010), S64–S73. DOI: 10.1016/j.diin.2010.05.009. URL: <http://www.sciencedirect.com/science/article/pii/S1742287610000368>.
- [54] Simson Garfinkel, Alex J. Nelson, and Joel Young. “A general strategy for differential forensic analysis”. In: *Digital Investigation* 9.SUPPL. (Aug. 2012), S50–S59. DOI: 10.1016/j.diin.2012.05.003. URL: <http://linkinghub.elsevier.com/retrieve/pii/S174228761200028X>.
- [55] Simson Garfinkel et al. “Bringing science to digital forensics with standardized forensic corpora”. In: *Digital Investigation* 6.Supplement 1 (2009), S2–S11. DOI: 10.1016/j.diin.2009.06.016. URL: <http://www.sciencedirect.com/science/article/pii/S1742287609000346>.
- [56] George Grispos, William Bradley Glisson, and Tim Storer. “Using Smartphones as a Proxy for Forensic Evidence Contained in Cloud Storage Services”. In: *2013 46th Hawaii International Conference on System Sciences*. IEEE, Jan. 2013, pp. 4910–4919. DOI: 10.1109/HICSS.2013.592. URL: <http://ieeexplore.ieee.org/document/6480436/>.
- [57] Jon C Haass, Gail-Joon Ahn, and Frank Grimmelmann. “ACTRA: A Case Study for Threat Information Sharing”. In: *Proceedings of the 2nd ACM Workshop on Information Sharing and Collaborative Security*. WISCS '15. New York, NY, USA: ACM, 2015, pp. 23–26. DOI: 10.1145/2808128.2808135.
- [58] Mike Halcrow. “eCryptfs: a Stacked Cryptographic Filesystem”. In: *Linux Journal* 0.156 (Apr. 2007), pp. 54–58. URL: <http://www.linuxjournal.com/article/9400>.
- [59] J Alex Halderman et al. “Lest We Remember: Cold Boot Attacks on Encryption Keys”. In: *Proc. 2008 USENEX Security Symposium*. Vol. 52. 5. New York, NY,

- USA: ACM, May 2008, pp. 91–98. DOI: 10.1145/1506409.1506429. URL: <http://portal.acm.org/citation.cfm?doid=1506409.1506429>.
- [60] E Hall. *The application/mbox Media Type*. RFC 4155 (Informational). Sept. 2005. URL: <https://tools.ietf.org/html/rfc4155>.
- [61] Brian Heater. *As Chromebook sales soar in schools, Apple and Microsoft fight back*. Apr. 2017. URL: <https://techcrunch.com/2017/04/27/as-chromebook-sales-soar-in-schools-apple-and-microsoft-fight-back/> (visited on 07/26/2017).
- [62] Robert M. Hinden. *IP Version 6 Addressing Architecture*. RFC 2373 (Standards Track). July 1998. URL: <https://tools.ietf.org/html/rfc2373>.
- [63] Robert M. Hinden and Stephen E. Deering. *IP Version 6 Addressing Architecture*. RFC 4291 (Standards Track). Feb. 2006. URL: <https://tools.ietf.org/html/rfc4291>.
- [64] Jessica Hyde and Brian Moran. ‘*Alexa, are you Skynet?*’ SANS DFIR Summit & Training 2017. June 2017. URL: <https://www.sans.org/summit-archives/file/summit-archive-1498230402.pdf>.
- [65] IDC Research Inc. *Traditional PC Market Fares Slightly Better Than Expectations as Component Shortage Pressures Ease, According to IDC*. July 2017. URL: <http://www.idc.com/getdoc.jsp?containerId=prUS42882717> (visited on 07/19/2017).
- [66] Internet Crime Complaint Center (IC3). *IC3 Annual Reports*. 2015. URL: <https://www.ic3.gov/media/annualreports.aspx>.
- [67] Internet Live Stats. *Google Search Statistics*. 2017. URL: <http://www.internetlivestats.com/google-search-statistics/#sources> (visited on 10/09/2017).
- [68] Internet Live Stats. *Number of Internet Users (2016)*. 2016. URL: <http://www.internetlivestats.com/internet-users/> (visited on 07/14/2017).
- [69] Internet Live Stats. *Total number of Websites*. 2017. URL: <http://www.internetlivestats.com/total-number-of-websites/> (visited on 07/14/2017).
- [70] Internet Live Stats. *Twitter Usage Statistics*. 2017. URL: <http://www.internetlivestats.com/twitter-statistics/> (visited on 10/09/2017).
- [71] Dohyun Kim et al. “Forensic Analysis of Android Phone Using Ext4 File System Journal Log”. In: *Future Information Technology, Application, and Service*. Springer, 2012, pp. 435–446. DOI: 10.1007/978-94-007-4516-2\_44. URL: [http://www.springerlink.com/index/10.1007/978-94-007-4516-2\\_44](http://www.springerlink.com/index/10.1007/978-94-007-4516-2_44).



- [72] G Klyne and J Palme. *Registration of Mail and MIME Header Fields*. RFC 4021 (Proposed Standard). Mar. 2005. URL: <https://tools.ietf.org/html/rfc4021>.
- [73] Brendan Koerner. “A Russian Slot Machine Hack Is Costing Casinos Big Time”. In: *WIRED* (Feb. 2017). URL: <https://www.wired.com/2017/02/russians-engineer-brilliant-slot-machine-cheat-casinos-no-fix/>.
- [74] Brian Krebs. *FBI: Extortion, CEO Fraud Among Top Online Fraud Complaints in 2016*. Krebs on Security. June 2017. URL: <https://krebsonsecurity.com/2017/06/fbi-extortion-ceo-fraud-among-top-online-fraud-complaints-in-2016/> (visited on 06/23/2017).
- [75] Frederic Lardinois. *Google says its G Suite for Education now has 70M users*. TechCrunch. Jan. 2017. URL: <https://techcrunch.com/2017/01/24/google-says-its-g-suite-for-education-now-has-70m-users/> (visited on 10/17/2017).
- [76] Frederic Lardinois. *More than 3M businesses now pay for Google’s G Suite*. TechCrunch. Jan. 2017. URL: <https://techcrunch.com/2017/01/26/more-than-3m-businesses-now-pay-for-googles-g-suite/> (visited on 10/17/2017).
- [77] Steven Levine. *Appendix A. The Device Mapper*. Red Hat Enterprise Linux 6: Logical Volume Manager Administration Guide. 2017. URL: [https://access.redhat.com/documentation/en-US/Red\\_Hat\\_Enterprise\\_Linux/6/html/Logical\\_Volume\\_Manager\\_Administration/device\\_mapper.html](https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Logical_Volume_Manager_Administration/device_mapper.html) (visited on 10/02/2017).
- [78] Mike Mabey. *How eCryptfs Affects Filename Lengths*. [https://mikemabey.com/blog/2017/08/ecryptfs\\_filenames.html](https://mikemabey.com/blog/2017/08/ecryptfs_filenames.html). Aug. 2017.
- [79] Mike Mabey et al. “dbling: Identifying extensions installed on encrypted web thin clients”. In: *Digital Investigation* 18 (Aug. 2016), S55–S65. DOI: 10.1016/j.diin.2016.04.007. URL: <http://linkinghub.elsevier.com/retrieve/pii/S174228761630038X>.
- [80] Fabio Marturana and Simone Tacconi. “A Machine Learning-based Triage methodology for automated categorization of digital media”. In: *Digital Investigation* 10.2 (Sept. 2013), pp. 193–204. DOI: 10.1016/j.diin.2013.01.001. URL: <http://linkinghub.elsevier.com/retrieve/pii/S1742287613000029>.
- [81] Raffael Marty. “Cloud application logging for forensics”. In: *Proceedings of the 2011 ACM Symposium on Applied Computing - SAC '11*. New York, New York, USA: ACM Press, 2011, p. 178. DOI: 10.1145/1982185.1982226. URL: <http://portal.acm.org/citation.cfm?doid=1982185.1982226>.

- [82] Peter Mell and Timothy Grance. “The NIST Definition of Cloud Computing: Recommendations of the National Institute of Standards and Technology”. In: *NIST Special Publication (2011) 800-145*. URL: <http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf> (2011).
- [83] MITRE Corporation. *CybOX – Cyber Observable eXpression*. <https://cybox.mitre.org>. 2017.
- [84] Shariq Murtuza et al. “A TOOL FOR EXTRACTING STATIC AND VOLATILE FORENSIC ARTIFACTS OF WINDOWS 8.x APPS”. In: *Advances in Digital Forensics XI: 11th IFIP WG 11.9 International Conference, Orlando, FL, USA, January 26-28, 2015, Revised Selected Papers*. Ed. by Gilbert Peterson and Sujeet Sheno. Vol. 462. IFIP Advances in Information and Communication Technology. Cham: Springer International Publishing, 2015, pp. 305–320. DOI: 10.1007/978-3-319-24123-4\_18. URL: [http://link.springer.com/10.1007/978-3-319-24123-4\\_18](http://link.springer.com/10.1007/978-3-319-24123-4%20http://link.springer.com/10.1007/978-3-319-24123-4_18).
- [85] Thomas Narten, Susan Thomson, and Tatuya Jinmei. *IPv6 Stateless Address Auto-configuration*. RFC 4862 (Standards Track). Sept. 2007. URL: <https://tools.ietf.org/html/rfc4862>.
- [86] National Institute of Standards and Technology (NIST). *National Software Reference Library*. National Institute of Standards and Technology. URL: <http://www.nsrll.nist.gov/>.
- [87] Alex Nelson. “XML Conversion of the Windows Registry for Forensic Processing and Distribution”. In: *Advances in Digital Forensics VIII: 8th IFIP WG 11.9 International Conference on Digital Forensics, Pretoria, South Africa, January 3-5, 2012, Revised Selected Papers*. Ed. by Sujeet Peterson, Gilbert and Sheno. Vol. 383 AICT. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. Chap. 4, pp. 51–65. DOI: 10.1007/978-3-642-33962-2\_4. URL: [http://link.springer.com/10.1007/978-3-642-33962-2\\_4](http://link.springer.com/10.1007/978-3-642-33962-2_4).
- [88] Alex J Nelson, Erik Q Steggall, and Darrell D E Long. “Cooperative mode: Comparative storage metadata verification applied to the Xbox 360”. In: *Digital Investigation* 11.Supplement 2 (2014), S46–S56. DOI: <http://dx.doi.org/10.1016/j.diin.2014.05.004>. URL: <http://www.sciencedirect.com/science/article/pii/S1742287614000474>.
- [89] Bill Nelson et al. *Guide to Computer Forensics and Investigations*. Third. Boston, Mass: Thomson Course Technology, 2008.

- [90] Chris Newman and Graham Klyne. *Date and Time on the Internet: Timestamps*. RFC 3339 (Standards Track). July 2002. URL: <https://tools.ietf.org/html/rfc3339>.
- [91] NPD Group. *Chromebooks are a Bright Spot in a Stagnant B2B PC and Tablet Market, According To NPD*. Aug. 2015. URL: <https://www.npd.com/wps/portal/npd/us/news/press-releases/2015/chromebooks-are-a-bright-spot-in-a-stagnant-b2b-pc-and-tablet-market-according-to-npd/>.
- [92] NPD Group. *NPD: U.S. B2B Channel Volumes Rise to More Than \$62 Billion, Indicate Steady and Consistent Growth for Market*. Mar. 2015. URL: <https://www.npd.com/wps/portal/npd/us/news/press-releases/2015/npd-us-b2b-channel-volumes-rise-to-more-than-62-billion-indicate-steady-and-consistent-growth-for-market/>.
- [93] NPD Group. *U.S. Commercial Channel Computing Device Sales Set to End 2013 with Double-Digit Growth, According to NPD*. Dec. 2013. URL: <https://www.npd.com/wps/portal/npd/us/news/press-releases/u-s-commercial-channel-computing-device-sales-set-to-end-2013-with-double-digit-growth-according-to-npd/>.
- [94] Martin S Olivier. “On metadata context in database forensics”. In: *Digital Investigation* 5.3 (2009), pp. 115–123.
- [95] Jens Olsson and Martin Boldt. “Computer forensic timeline visualization tool”. In: *Digital Investigation* 6 (Sept. 2009), S78–S87. DOI: 10.1016/j.diin.2009.06.008. URL: <http://linkinghub.elsevier.com/retrieve/pii/S1742287609000425>.
- [96] Justin Paglierani, Mike Mabey, and Gail-Joon Ahn. “Towards Comprehensive and Collaborative Forensics on Email Evidence”. In: *Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), 2013 9th International Conference Conference on*. Oct. 2013, pp. 11–20. DOI: 10.4108/icst.collaboratecom.2013.254125.
- [97] Robby Payne. *Google WiFi Powered By Chrome OS*. Oct. 2016. URL: <https://chromeunboxed.com/google-wifi-powered-by-chrome-os/> (visited on 10/12/2016).
- [98] Jonathan Rajewski. *Internet of Things Forensics*. Enfuse 2017, Guidance Software. May 2017. URL: [https://www.jonrajewski.com/wp-content/uploads//2017/07/Enfuse\\_2017\\_Rajewski\\_IOT\\_Forensics.pdf](https://www.jonrajewski.com/wp-content/uploads//2017/07/Enfuse_2017_Rajewski_IOT_Forensics.pdf)[https://www.guidancesoftware.com/enfuse-conference/sessions?cmpid=side\\_menu\\_r#event8](https://www.guidancesoftware.com/enfuse-conference/sessions?cmpid=side_menu_r#event8).
- [99] Andrew Reed and Michael Kranch. “Identifying HTTPS-Protected Netflix Videos in Real-Time”. In: *Proceedings of the Seventh ACM on Conference on Data and*

*Application Security and Privacy - CODASPY '17*. New York, New York, USA: ACM Press, 2017, pp. 361–368. DOI: 10.1145/3029806.3029821. URL: <http://dl.acm.org/citation.cfm?doid=3029806.3029821>.

- [100] Vassil Roussev, Andres Barreto, and Irfan Ahmed. “API-Based Forensic Acquisition of Cloud Drives”. In: *Advances in Digital Forensics XII: 12th IFIP WG 11.9 International Conference, New Delhi, January 4-6, 2016, Revised Selected Papers*. Ed. by Gilbert Peterson and Sujeeet Shenoj. Vol. 484. Springer International Publishing, Sept. 2016. Chap. 11, pp. 213–235. DOI: 10.1007/978-3-319-46279-0\_11. URL: [http://link.springer.com/10.1007/978-3-319-46279-0\\_11](http://link.springer.com/10.1007/978-3-319-46279-0_11).
- [101] Keyun Ruan et al. “Cloud Forensics”. In: *Advances in Digital Forensics VII: 7th IFIP WG 11.9 International Conference on Digital Forensics, Orlando, FL, USA, January 31 – February 2, 2011, Revised Selected Papers*. Springer Berlin Heidelberg, 2011, pp. 35–46. DOI: 10.1007/978-3-642-24212-0\_3. URL: [http://link.springer.com/10.1007/978-3-642-24212-0\\_3](http://link.springer.com/10.1007/978-3-642-24212-0_3).
- [102] Sandvine Incorporated ULC. *Global Internet Phenomena Report: North America and Latin America*. May 2015. URL: <https://www.sandvine.com/downloads/general/global-internet-phenomena/2015/global-internet-phenomena-report-latin-america-and-north-america.pdf> (visited on 01/01/2016).
- [103] Bruce Schneier and John Kelsey. “Secure audit logs to support computer forensics”. In: *ACM Transactions on Information and System Security* 2.2 (May 1999), pp. 159–176. DOI: 10.1145/317087.317089. URL: <http://dl.acm.org/citation.cfm?id=317087.317089>.
- [104] A P Singh et al. “Acer Aspire One Netbooks: A Forensic Challenge”. In: *Computer Software and Applications Conference, 2009. COMPSAC '09. 33rd Annual IEEE International*. Vol. 2. July 2009, pp. 404–409. DOI: 10.1109/COMPSAC.2009.167.
- [105] Johannes Stadlinger and Andreas Dewald. *Email Communication Visualization in (Forensic) Incident Analysis*. Whitepaper. ERNW Enno Rey Netzwerke GmbH, June 2017, p. 19. URL: [https://www.ernw.de/download/newsletter/ERNW\\_Whitepaper59\\_EmailForensicsVisualisation\\_signed.pdf](https://www.ernw.de/download/newsletter/ERNW_Whitepaper59_EmailForensicsVisualisation_signed.pdf).
- [106] Jon Swartz. *Apple loses more ground to Google's Chromebook in education market*. USA Today online. Jan. 2016. URL: <http://www.usatoday.com/story/tech/news/2016/01/11/apple-loses-more-ground-googles-chromebook-education-market/78323158/>.
- [107] Zareen Syed et al. “UCO : A Unified Cybersecurity Ontology”. In: *Workshops of the Thirtieth AAAI Conference on Artificial Intelligence Artificial Intelligence*

- for Cyber Security: Technical Report WS-16-03 Figure 1* (2016), pp. 195–202. URL: <http://www.aaai.org/ocs/index.php/WS/AAAIW16/paper/view/12574>.
- [108] C. Tassone, B. Martini, and K.-K.R. Choo. “Forensic Visualization”. In: *Contemporary Digital Forensic Investigations of Cloud and Mobile Applications*. Elsevier, 2017, pp. 163–184. DOI: 10.1016/B978-0-12-805303-4.00011-3. URL: <http://linkinghub.elsevier.com/retrieve/pii/B9780128053034000113>.
- [109] Sean Thorpe et al. “Cloud log forensics metadata analysis”. In: *Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual*. IEEE, 2012, pp. 194–199.
- [110] Erik Trickel et al. “Shell We Play A Game? CTF-as-a-service for Security Education”. In: *2017 USENIX Workshop on Advances in Security Education (ASE 17)*. USENIX Association, Aug. 2017. URL: <https://www.usenix.org/conference/ase17/workshop-program/presentation/trickel>.
- [111] Chief Information Officer/G-6 United States Army. *Thin/Zero Client Computing Reference Architecture Version 1.0*. Mar. 2013. URL: [http://ciog6.army.mil/Portals/1/Architecture/ApprovedThinClient-ZeroComputingReferenceArchitecturev1-0\\_14Mar13.pdf](http://ciog6.army.mil/Portals/1/Architecture/ApprovedThinClient-ZeroComputingReferenceArchitecturev1-0_14Mar13.pdf).
- [112] USLegal Inc. *Inculpatory Evidence Law and Legal Definition*. URL: <https://definitions.uslegal.com/i/inculpatory-evidence/> (visited on 11/07/2017).
- [113] Sally Vandeven. *Forensic Images: For Your Viewing Pleasure*. Whitepaper. The SANS Institute, Sept. 2014. URL: <https://www.sans.org/reading-room/whitepapers/forensics/forensic-images-viewing-pleasure-35447>.
- [114] Oscar Vermaas, Joep Simons, and Rob Meijer. “Open Computer Forensic Architecture a Way to Process Terabytes of Forensic Disk Images”. In: *Open Source Software for Digital Forensics*. Boston, MA: Springer US, 2010. Chap. 4, pp. 45–67. DOI: 10.1007/978-1-4419-5803-7\_4. URL: [http://link.springer.com/10.1007/978-1-4419-5803-7\\_4](http://link.springer.com/10.1007/978-1-4419-5803-7_4).
- [115] Tom Warren. *Chromebooks outsold Macs for the first time in the US*. May 2016. URL: <http://www.theverge.com/2016/5/19/11711714/chromebooks-outsold-macs-us-idc-figures> (visited on 01/01/2016).
- [116] Wex Legal Dictionary / Encyclopedia. *10 CFR 26.129 - Assuring specimen security, chain of custody, and preservation*. Legal Information Institute at Cornell University Law School. URL: <https://www.law.cornell.edu/cfr/text/10/26.129>.

- [117] Wex Legal Dictionary / Encyclopedia. *10 CFR 26.5 - Definitions*. Legal Information Institute at Cornell University Law School. URL: <https://www.law.cornell.edu/cfr/text/10/26.5>.
- [118] Wex Legal Dictionary / Encyclopedia. *Doctrine of Completeness*. Legal Information Institute at Cornell University Law School. URL: [https://www.law.cornell.edu/wex/doctrine\\_of\\_completeness](https://www.law.cornell.edu/wex/doctrine_of_completeness).
- [119] Wex Legal Dictionary / Encyclopedia. *Exculpatory*. Legal Information Institute at Cornell University Law School. URL: <https://www.law.cornell.edu/wex/exculpatory>.
- [120] Wex Legal Dictionary / Encyclopedia. *Rule 1001. Definitions That Apply to This Article*. Legal Information Institute at Cornell University Law School. URL: [https://www.law.cornell.edu/rules/fre/rule\\_1001](https://www.law.cornell.edu/rules/fre/rule_1001).
- [121] Wex Legal Dictionary / Encyclopedia. *Rule 1003. Admissibility of Duplicates*. Legal Information Institute at Cornell University Law School. URL: [https://www.law.cornell.edu/rules/fre/rule\\_1003](https://www.law.cornell.edu/rules/fre/rule_1003).
- [122] Wex Legal Dictionary / Encyclopedia. *Rule 402. General Admissibility of Relevant Evidence*. Legal Information Institute at Cornell University Law School. URL: [https://www.law.cornell.edu/rules/fre/rule\\_402](https://www.law.cornell.edu/rules/fre/rule_402).
- [123] Wex Legal Dictionary / Encyclopedia. *Rule 901. Authenticating or Identifying Evidence*. Legal Information Institute at Cornell University Law School. URL: [https://www.law.cornell.edu/rules/fre/rule\\_901](https://www.law.cornell.edu/rules/fre/rule_901).
- [124] Yunus Yusoff, Roslan Ismail, and Zainuddin Hassan. "Common Phases of Computer Forensics Investigation Models". In: *International Journal of Computer Science and Information Technology* 3.3 (June 2011), pp. 17–31. DOI: 10.5121/ijcsit.2011.3302. URL: <http://www.airccse.org/journal/jcsit/0611csit02.pdf>.

APPENDIX A

EMAIL FORENSICS XML (EFXML) SCHEMA

Below is the full schema for Email Forensics XML (EFXML) files. The current version is publicly available at <https://github.com/jpaglier/efxml>.

```
1 <?xml version="1.0" encoding="utf-8"?>
  <xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:dfxml="http://www.forensicswiki.org/wiki/Category:Digital_Forensics_XML"
5    xmlns:efxml="https://mikemabey.com/schema/efxml"
    targetNamespace="https://mikemabey.com/schema/efxml"
    version="0.1"
    elementFormDefault="qualified">

10    <xs:annotation>
      <xs:documentation>
        This is the schema file for Email Forensics XML (EFXML) version 0.1.

        Authors: Justin Paglierani and Mike Mabey
15      </xs:documentation>
    </xs:annotation>

    <xs:import
      ↪ namespace="http://www.forensicswiki.org/wiki/Category:Digital_Forensics_XML"
      ↪ schemaLocation="dfxml/dfxml.xsd"/>

20    <xs:import namespace="https://mikemabey.com/schema/headers"
      ↪ schemaLocation="https://mikemabey.com/schema/email_headers.xsd"/>

    <!-- Elements -->

    <xs:element name="efxml">
25      <xs:complexType>
        <xs:sequence>
          <xs:element ref="efxml:mailbox" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="version" type="xs:string" use="required">
30          <xs:annotation>
            <xs:documentation>The version of the EFXML schema to which the EFXML file
              ↪ adheres.</xs:documentation>
          </xs:annotation>
        </xs:attribute>
      </xs:complexType>
35    </xs:element>

    <xs:element name="mailbox">
      <xs:complexType>
        <xs:sequence>
40          <xs:element ref="efxml:Folder" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string"/>
      </xs:complexType>
    </xs:element>
  </xs:schema>
```



```

    </xs:complexType>
  </xs:element>
45
  <xs:element name="Folder">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="efxml:message" minOccurs="0" maxOccurs="unbounded"/>
50      </xs:sequence>
      <xs:attribute name="Name" type="xs:string"/>
    </xs:complexType>
  </xs:element>

55  <xs:element name="message">
    <xs:complexType>
      <xs:sequence>
        <xs:any namespace="https://mikemabey.com/schema/headers" minOccurs="0"
          ↪ maxOccurs="unbounded"/>
        <xs:element name="byte_run" type="efxml:byteRunType" minOccurs="0"
          ↪ maxOccurs="1"/>
60      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!-- Custom Types -->
65
  <xs:complexType name="byteRunType">
    <xs:annotation>
      <xs:documentation>
        Corresponds with the DFXML element "byte_run" but with unused attributes
          ↪ omitted.
70      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element ref="dfxml:hashdigest" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
75    <xs:attribute name="file_offset" type="xs:nonNegativeInteger"/>
    <xs:attribute name="len" type="xs:nonNegativeInteger"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
80 </xs:schema>

```

As seen above on line 20, the EFXML schema imports another schema called `email_headers.xsd`, and, as seen on line 58, the `<message>` tag may contain any number of tags from this file as long as it is within the “headers” namespace. This allows for a very flexible inclusion of all email headers defined in this additional schema.

The full schema for email headers is available at [https://github.com/mmabey/header\\_schemas](https://github.com/mmabey/header_schemas), but is quite verbose. An abbreviated version is below.

---

```

1  <?xml version="1.0" encoding="utf-8"?>
   <xs:schema
     xmlns:xs="http://www.w3.org/2001/XMLSchema"
     xmlns:head="https://mikemabey.com/schema/header_info"
5   xmlns:email="https://mikemabey.com/schema/headers"
     targetNamespace="https://mikemabey.com/schema/headers"
     version="0.1"
     elementFormDefault="qualified">

10  <!-- Allow Non-RFC Headers -->

     <xs:element name="Not-In-RFC" type="email:any_field"/>

     <!-- Permanent Message Header Field Elements -->

15  <xs:element name="Bcc" type="email:address_field">
     <xs:annotation>
       <xs:appinfo><head:header protocol="mail" rfc="5322 4021 822 2822"
         ↪ type="permanent" status="standard"/></xs:appinfo>
     </xs:annotation>
20 </xs:element>

     <xs:element name="Cc" type="email:address_field">
     <xs:annotation>
       <xs:appinfo><head:header protocol="mail" rfc="5322 4021 2822 822"
         ↪ type="permanent" status="standard"/></xs:appinfo>
25 </xs:annotation>
     </xs:element>

     <xs:element name="From" type="email:address_field">
     <xs:annotation>
30 <xs:appinfo><head:header protocol="mail" rfc="5322 6854 4021 2822 822"
       ↪ type="permanent" status="standard"/></xs:appinfo>
     </xs:annotation>
     </xs:element>

     <xs:element name="Keywords" type="email:keywords_field">
35 <xs:annotation>
       <xs:appinfo><head:header protocol="mail" rfc="5322 4021 2822 822"
         ↪ type="permanent" status="standard"/></xs:appinfo>
     </xs:annotation>
     </xs:element>

40 <xs:element name="Message-ID" type="head:string">
     <xs:annotation>
       <xs:appinfo><head:header protocol="mail" rfc="5322 4021 822" type="permanent"
         ↪ status="standard"/></xs:appinfo>
     </xs:annotation>
     </xs:element>

```

45

```
<xs:element name="Reply-To" type="email:address_field">  
  <xs:annotation>  
    <xs:appinfo><head:header protocol="mail" rfc="5322 4021 2822 822"  
      ↪ type="permanent" status="standard"/></xs:appinfo>  
    </xs:annotation>  
50 </xs:element>
```

50

```
<xs:element name="Subject" type="head:string">  
  <xs:annotation>  
    <xs:appinfo><head:header protocol="mail" rfc="5322 4021 2822 822"  
      ↪ type="permanent" status="standard"/></xs:appinfo>  
55 </xs:annotation>  
</xs:element>
```

55

```
<xs:element name="To" type="email:address_field">  
  <xs:annotation>  
60 <xs:appinfo><head:header protocol="mail" rfc="5322 4021 2822 822"  
      ↪ type="permanent" status="standard"/></xs:appinfo>  
    </xs:annotation>  
</xs:element>
```

60

```
</xs:schema>
```

---

APPENDIX B

MATCHING EXTENSION RANKING LIST (MERL) SCHEMA

Below is the full schema for Matching Extension Ranking List (MERL) files. The current version is publicly available at [https://github.com/mmabey/merl\\_schema](https://github.com/mmabey/merl_schema).

```
1 <?xml version="1.0" encoding="UTF-8"?>
  <xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:dfxml="http://www.forensicswiki.org/wiki/Category:Digital_Forensics_XML"
5    xmlns:merl="https://mikemabey.com/schema/merl"
    targetNamespace="https://mikemabey.com/schema/merl"
    version="0.1"
    elementFormDefault="qualified">

10    <xs:annotation>
      <xs:documentation>
        This is the schema file for Matching Extension Ranking List (MERL) version
        ↪ 0.1.
      </xs:documentation>
    </xs:annotation>

15    <xs:import
      namespace="http://www.forensicswiki.org/wiki/Category:Digital_Forensics_XML"
      schemaLocation="dfxml/dfxml.xsd">
      <xs:annotation>
20        <xs:documentation>
          The DFXML schema needs to be imported to validate with the `xmllint`
          ↪ utility.
        </xs:documentation>
      </xs:annotation>
    </xs:import>

25    <xs:element name="merl">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="merl:source" minOccurs="0" maxOccurs="1"/>
30          <xs:element ref="dfxml:creator" minOccurs="0" maxOccurs="1"/>
          <xs:element ref="merl:match" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
      </xs:complexType>
35    </xs:element>

    <!--All further elements listed in alphabetical order.-->

40    <xs:element name="candidate">
      <xs:complexType>
        <xs:all>
          <xs:element ref="merl:ext_id" minOccurs="1" maxOccurs="1"/>
          <xs:element ref="merl:ext_ver" minOccurs="1" maxOccurs="1"/>
          <xs:element ref="merl:ext_name" minOccurs="0" maxOccurs="1"/>

```

```

45     <xs:element ref="merl:confidence" minOccurs="0" maxOccurs="1"/>
      </xs:all>
    </xs:complexType>
  </xs:element>

50  <xs:element name="confidence" type="xs:float"/>

    <xs:element name="ext_id" type="xs:string"/>

    <xs:element name="ext_name" type="xs:string"/>
55  <xs:element name="ext_ver" type="xs:string"/>

    <xs:element name="inode" type="xs:nonNegativeInteger"/>

60  <xs:element name="match">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="merl:inode" minOccurs="1" maxOccurs="1"/>
        <xs:element ref="merl:candidate" minOccurs="1" maxOccurs="unbounded"/>
65  </xs:sequence>
      </xs:complexType>
    </xs:element>

    <xs:element name="mount_point" type="xs:string"/>

70  <xs:element name="source">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="dfxml:image_filename" minOccurs="0" maxOccurs="1"/>
75  <xs:element ref="merl:mount_point" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

80 </xs:schema>

```

---

APPENDIX C  
OPEN-SOURCE PROJECTS

Most of the code I wrote for the tools and experimental setups discussed in this dissertation are open-source and publicly available. In addition, I also created and contributed to a number other of open-source projects while in school. This appendix introduces all of these projects (those associated with this dissertation as well as others) and briefly describes my contributions to each.

### C.1 dbling

*Available at:* <https://github.com/sefcom/dbling>

*Docs available at:* <http://dbling.readthedocs.io/>

dbling is the main software contribution of this dissertation. It includes the following main components:

**Crawler** The Crawler finds and downloads the list of the currently-available extensions on the Chrome Web Store, determines which extensions are at a version that has already been downloaded, downloads those that have not yet been downloaded, and adds information on the newly downloaded extensions to the database<sup>66</sup>.

**Template Generator** The Template Generator runs concurrently with the Crawler. For each new extension downloaded by the Crawler, the Template Generator calculates the centroid of the extension and stores it in the database. The Template Generator does not run inside Chrome or Chrome OS, and so it does not use the same mechanisms for unpacking and installing that Chrome does natively. Instead, the primary function of the Template Generator is to mimic as closely as possible the Chrome's functions as they pertain to unpacking and installing extensions.

**Profiler** The Profiler is a command line tool designed for use by forensic examiners to create profiles of the extensions installed on a disk image. This is the piece of code that uses the information about Chrome OS's disk and file system layout to identify the most likely encrypted directories to contain installed extensions. It then leverages the MERL Exporter to interpret the results against the database of extension templates and store them to a MERL file.

**MERL Exporter** The MERL Exporter creates a MERL file based on a set of information on extension candidates. The MERL Exporter is used directly by the Profiler to query the database of extension fingerprints (originally created by the Template Generator) for matching extension profiles and filters them using a set of given criteria. It then saves the results of the profile search by translating the results into XML entries that conform to the MERL schema.

**Gripper** Gripper leverages the APIs provided by G Suite to acquire data about a user's activities on a Chromebook. Specifically, Gripper answers the following questions:

---

<sup>66</sup>In the middle of my research on this project, Google changed how they listed the current extensions, which required me to re-write a large portion of the Crawler.



(1) Who was logged into a specific device? (2) When were they logged in? (3) What did they do while logged in? (4) What files may contain potentially critical information based on when they were created or modified?

In addition to the components listed above, `dbling` also leverages the following independent projects in its source code: `crx_unpack`, `ImgVault`, `pyuefi`, and `dbling-webstore`.

## C.2 `crx_unpack`

*Code available at:* [https://bitbucket.org/mmabey/crx\\_unpack](https://bitbucket.org/mmabey/crx_unpack)

*Docs available at:* <http://crx-unpack.readthedocs.io/>

*PyPI Listing:* <https://pypi.org/project/crx-unpack/>

`crx_unpack` is a module for Python 3. It contains several utilities for working with Google Chrome extension files (CRXs), which have the `*.crx` file extension. The goal of this project is to mimic as closely as possible the functionality of Google Chrome when these extensions are unpacked and installed.

The first module is `crx_unpack`, which handles the headers and structure of the CRX itself. The second module is Encrypted Temp Directory, which gives a way to use `eCryptfs` to encrypt a directory that only ever exists in memory by inheriting from Python's `TemporaryDirectory`<sup>67</sup> class and hooking into some `eCryptfs` tools to handle the encryption of the file contents and file names (handled by different keys).

I am presently the sole maintainer and developer for this project.

## C.3 `ImgVault`

*Available at:* <https://bitbucket.org/mmabey/imgvault>

`ImgVault` is a forensics utility designed to help leverage cloud computing services as part of an examination. `ImgVault` interfaces with the cloud hypervisor to help perform actions such as mounting or unmounting a volume to or from an instance, taking a snapshot of an instance, reverting an instance snapshot, and so forth. By abstracting away these actions, the goal is to allow automated tools to work with evidence stored in the cloud in much the same way that a human examiner would work with physical (digital) evidence. The implementation of `ImgVault` has been primarily focused on supporting hypervisors via `libvirt`, but is flexible enough to support other hypervisors as well.

I am presently the sole maintainer and developer for this project.

---

<sup>67</sup>See <https://docs.python.org/3.5/library/tempfile.html#tempfile.TemporaryDirectory>.

#### C.4 pyuefi

*Available at:* <https://bitbucket.org/mmabey/pyuefi>

*PyPI Listing:* <https://pypi.org/project/pyuefi/>

PyUEFI is a pure Python tool for extracting UEFI partition information and layout from hard drives and hard drive images.

I am presently the sole maintainer and developer for this project.

#### C.5 dbling-webstore

*Available at:* <https://bitbucket.org/mmabey/dbling-webstore>

The `dbling-webstore` module acts as a proxy for the Chrome Web Store. This allows `dbling` to tell Chrome on startup to use the IP address of this server as the Web Store. With this flexibility, `dbling` is not constrained to installing only the latest available version of each extension, but can also specify an older version that `dbling` downloaded previously.

I am presently the sole maintainer and developer for this project.

#### C.6 Crumbler

*Available at:* <https://bitbucket.org/mmabey/crumbler>

`Crumbler` was a critical element of the work I did with Justin Paglierani on web email forensics [96]. `Crumbler` bridges the gap between how the Chrome browser stores its cookies and how the Selenium web driver requires cookies, thereby making it possible to re-use the cookies from Chrome in an automated fashion.

I am the sole developer for this project.

#### C.7 ictf-framework

*Available at:* <https://github.com/ucsb-seclab/ictf-framework>

This is the framework that the UC Santa Barbara Seclab and Shellphish use to host the International Capture the Flag (iCTF) competition. The initial framework has been expanded by the Arizona State University SEFCOM lab to allow for cloud-based deployments. This framework can be used to create Capture The Flag (CTF) competitions.

As part of the SEFCOM team that implemented the necessary features to allow the framework to be deployed in a cloud environment, I made the following contributions to this project:

- Updated the framework's code to make it compatible with current versions of the Python programming language

- Implemented code that creates multiple framework components (cloud VMs) simultaneously
- Improved and added code for automatically detecting and recovering from failures when creating the game framework in the cloud

Details of our work to make the competition available for others to use in the cloud were published in [110].

## C.8 python-vagrant

*Available at:* <https://github.com/toddeluca/python-vagrant>

Python-vagrant is a Python module that provides a thin wrapper around the vagrant command line executable, allowing programmatic control of Vagrant virtual machines (boxes).

As part of my work on the `ictf-framework` project (see Appendix C.7), I contributed code to `python-vagrant` that allows output from the call to Vagrant to be filtered for important values about the created VM. Ultimately, the project maintainer and I agreed to some alterations to my code for the sake of maintaining simplicity of code and common utility for most users of the module (see the full discussion at <https://github.com/toddeluca/python-vagrant/pull/56>). I was credited as a co-author of the final changes for version 0.5.15 of the library<sup>68</sup>.

## C.9 natmgr

*Available at:* <https://github.com/mmabey/natmgr>

NATMGR helps ease the management of the port forwarding rules on a NAT server by keeping track of who requested which forwarding rules and when each rule should no longer be in effect. NATMGR gives users a command-line interface for viewing current forwarding rules and adding and removing rules. NATMGR then creates and activates iptables rules that correspond with the user's input.

I am presently the sole maintainer and developer for this project.

## C.10 resume

*Available at:* <https://bitbucket.org/mmabey/resume/overview>

While creating an academic curriculum vitae (CV), the template I was using (created by Trey Hunner) did not have all the features I thought were necessary for a truly generic, yet flexible, CV. So, I created a new L<sup>A</sup>T<sub>E</sub>X class that introduced several new features, including:

---

<sup>68</sup>See <https://github.com/toddeluca/python-vagrant/blob/0.5.15/CHANGELOG.md>.

- Automatically generated PDF index of the sections and subsections.
- Reordering of the fields for experience entries so that the position title and date are on the first line and the organization and location are on the second line.
- Ability to specify a “logo” which appears in the upper right-hand corner of the document.
- Separation, renaming of sections to allow more flexibility in the document layout.
- Introduction of footers that show the date the document was revised, page number (for documents >2 pages long), and a name (such as a last name).
- Improvement of margins and spacing to improve readability and minimize content breaking between pages when it should not break.
- A personalized bibliography for researchers with published work they want to showcase, and a way to divide the publications into groups.
- Flags for specifying that the document is a CV (which usually has way more content than a resume), allowing for dynamic content inclusion. With this, a resume and a CV can be produced from the same source files, reducing the amount of work necessary to keep the two current with each other.

I am presently the sole maintainer and developer for this project.

### C.11 Andgrab

*Available at:* <https://bitbucket.org/mmabey/andgrab>

Andgrab is a simple forensic tool for extracting data off an Android phone. It is quite old and was written around Spring 2011. It also has a number of shortcomings that I was never able to work out (some of which I have documented by creating issues). Having said that, the code and the approach may be of use to someone, so I decided to make it available.

I am the sole developer for this project.

### C.12 Adafruit Soundboard Drivers for MicroPython and CircuitPython

*Code available at:* [https://github.com/mmabey/Adafruit\\_Soundboard](https://github.com/mmabey/Adafruit_Soundboard)

*Docs available at:* <http://adafruit-soundboard.readthedocs.io/>

The family of Adafruit Soundboards make it easy for makers to add sound to their projects. However, the driver provided by Adafruit only supports Arduino-compatible microcontrollers. I translated the Arduino driver for these boards to MicroPython, an implementation of the Python programming language designed to run on resource-constrained devices, such as microcontrollers.

The first version of the driver caught the attention of the maintainers of CircuitPython, a fork of MicroPython created by Adafruit. They expressed interest in me making a version that was fully compatible with CircuitPython (there are a number of incompatibilities

between the two languages) and even sent me three different microcontroller boards for me to use in my development. This version of the driver has been included in the bundle of community-developed libraries, available at [https://github.com/adafruit/CircuitPython\\_Community\\_Bundle](https://github.com/adafruit/CircuitPython_Community_Bundle).

I am presently the sole maintainer and developer for this project.

### C.13 CircuitPython Driver for the HC-SR04 Ultrasonic Range Sensor

*Code available at:* [https://github.com/mmabey/CircuitPython\\_HCSR04](https://github.com/mmabey/CircuitPython_HCSR04)

*Docs available at:* <http://circuitpython-hcsr04.readthedocs.io/>

The HC-SR04 is an inexpensive solution for measuring distances using microcontrollers. This library provides a simple driver for controlling these sensors from CircuitPython, Adafruit's port of MicroPython. This has been included in the bundle of community-developed libraries, available at [https://github.com/adafruit/CircuitPython\\_Community\\_Bundle](https://github.com/adafruit/CircuitPython_Community_Bundle).

I am presently the sole maintainer and developer for this project.