



ANALYSIS OF FORENSIC SUPER TIMELINES

GRADUATE RESEARCH PROJECT

Stephen J. Esposito, Major, USAF

AFIT/ICW/ENG/12-04

**DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY**

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A: APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION UNLIMITED.

The views expressed in this report are those of the author and do not reflect the official policy or position of the United States Air Force, the Department of Defense, or the United States Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

AFIT/ICW/ENG/12-04

ANALYSIS OF FORENSIC SUPER TIMELINES

GRADUATE RESEARCH PROJECT

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Cyber Warfare

Stephen J. Esposito

Major, USAF

June 2012

DISTRIBUTION STATEMENT A: APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION UNLIMITED.

ANALYSIS OF FORENSIC SUPER TIMELINES

Stephen J. Esposito

Major, USAF

Approved:



Gilbert L. Peterson, PhD (Chairman)

21 MAY 2012
Date



Barry E. Mullins, PhD (Member)

21 May 12
Date

Abstract

As the use and adoption of networked electronic devices grows, their use in conjunction with crimes also increases. Extracting the probative evidence from these devices requires experienced digital forensics examiners. These examiners use several specialized tools that interpret the raw binary data present in digital media. Once the evidentiary artifacts are collected, one of the examiners goals is to assemble a narrative that describes when events occurred based on the time associated with the artifacts. Unfortunately, generating and evaluating these narrative super timelines is a manual and labor intensive process. This research focuses on aiding the examiner in evaluation through the generation of several queries that can extract and connect the temporal artifacts, and produce concise timelines. Extracting and analyzing these concise timelines allows the examiner to decrease the number artifacts to search through from hundreds of thousands of artifacts to only a hundred artifacts or less. Additionally, queries that correlate various artifacts allow the examiner to confirm or deny attribution of the user's actions. Application of the queries presented on a fictitious event demonstrates their ability to reduce the number of artifacts and facilitate the understanding of the activities surrounding the incident.

Table of Contents

	Page
Abstract.....	iv
Table of Contents.....	v
List of Figures.....	vii
List of Tables.....	ix
I. Introduction.....	1
1.1 Incident Response.....	2
1.2 Real Cases When a Super Timeline Was Needed.....	5
1.3 Timeline the Data.....	6
1.4 Correlating Various Different Artifacts.....	7
1.5 Research Goal.....	8
1.6 Summary.....	10
II. Related Work.....	11
2.1 Open Source Versus Closed Source.....	12
2.2 Log2timeline.....	14
2.3 Timestomp.....	18
2.4 Summary.....	20
III. Temporal Artifact Collection.....	21
3.1 Installing Log2timeline on a Windows-Based System.....	22
3.2 Executing Log2timeline.....	23
3.3 Verifying Log2timeline.....	24
3.4 Mounting VMware Image for Log2timeline Access.....	28
3.5 Known Error with Log2timeline.....	29
3.6 DD Image Tool.....	29
3.7 Collection Process.....	32
3.8 Summary.....	35
IV. Timeline Artifact Analysis.....	36
4.1 Developed Queries.....	36
4.1.1 All Artifacts Associated to Users.....	37
4.1.2 Web History.....	38
4.1.3 Recent Document History.....	39

4.1.4 User's Activity within a Timeframe	40
4.1.5 Mapping All Processes by Process ID.....	41
4.2 Test Configuration	42
4.3 Database Design	43
4.4 Query Searches	45
4.4.1 User Program Activity	45
4.4.2 User Web History	47
4.4.3 Recent Document Accesses.....	49
4.5 Windows 7 Incident.....	51
4.6 Windows XP Incident.....	55
4.7 Summary.....	62
 V. Conclusion and Future Work	 63
 Appendix A - Log2Timeline (ver. 0.62) Input Module Listing.....	 A.1
 Appendix B - Log2Timeline Quick Reference Guide	 B.1
 Appendix C – Script Used to Create Incident.....	 C.1
 Bibliography	 BIB.1

List of Figures

Figure	Page
1. - Seven Components of Incident Response (Mandia, Prosis & Pepe, 2003).	3
2. - Location of Internet Explorer and Firefox Web History.	16
3. - Simile Widget Example (Simile Widget Example, 2009).	17
4. - Original File MAC Times (Timestamp, 2010).	18
5. - MAC Times after Timestamp (Timestamp, 2010).	18
6. - Windows Installation Instructions for Log2timeline (\log2timeline\docs\install.txt).	23
7. - Capturing MAC Times with Windows DIR Command.	25
8. - Example Output from One of the Three DIR Commands.	26
9. - Batch File Converting Directory Command Output to CSV File	26
10. - Example Log2timeline CSV Output Viewed in MS Excel.	27
11. - Example Output from Batch File.	27
12. - Guymager Options Used to Acquire Col Mustard's VMware Image.	31
13. - Guymager Finished Acquiring and Verifying Image.	32
14. - Mounting DD Image in Mount Image Pro.	33
15. - Windows File Manager View of Mounted DD Image Using Mount Image Pro.	34
16. - MS Access' Table Properties of Log2timeline CSV File.	44
17. - Steganography Programs Found.	46
18. - More Programs Found.	47
19. - SQL Results for Mustard Gambling.	49
20. - SQL Query for Recent Documents.	50

21. – Query for Executable and Text Files during Miss Scarlet’s Logged on Time.	52
22. - Partial Listing from SQL Query above Displaying When System was Compromised.....	53
23. - Miss Scarlet's Activities (Logoff, Logon, & EXE programs).	56
24. - Query for Event ID 592 and 593, Starting and Ending of Processes.	58
25. - Processes by Time, its PID, and Parent PID during Miss Scarlet’s Logon Time. ...	59
26. - Query Results of Activities around ProcessList.exe and cmd.exe processes.	60

List of Tables

Table	Page
1. - Example Artifacts for User's Activity Query.	37

ANALYSIS OF FORENSIC SUPER TIMELINES

I. Introduction

Computers and digital equipment are everywhere today. When companies first started using computers they filled an entire room, now a computer is so small it can fit in a shirt pocket. These ever shrinking digital devices provide more portable capability to the user. Some of these devices are present during or used to commit a crime. This makes these devices important for locating evidence of a crime.

The Department of Defense (DoD) Chief Information Officer (CIO), Teresa Takai, states that her vision of “IT modernization targets the most pressing, near-term challenges and presents approaches to efficiently and effectively deliver agile, secure, integrated, and responsive IT capabilities” (DoD CIO, 2012). One way to ensure agile, secure, integrated, and responsive networks is through incident response. One of the key facets of incident response is how the temporal nature of data items correlates to one another. It is important to identify when the virus first appeared on a network, but it is even more important to understand how other temporal data items correlate to this virus like which user was logged in and when did this user log in and out of the system. Connecting these temporal items together creates a timeline of events that can explain and confirm the details of the incident to include who, what, where, and most importantly when.

This document discusses how a timeline of events can be captured, processed into a concise listing, and analyzed to correlate temporal items to produce knowledge about how the incident occurred on the system.

1.1 Incident Response

A computer security incident is any unlawful, unauthorized, or unacceptable action that involves a computer system or computer network (Mandia, Prosis & Pepe, 2003). The most common types of computer incidents are denial of service attack, an email with a rogue web link embedded, or malware that runs on the system. An incident response is the analysis of the incident, from interviewing the people involved, collecting the evidence from the affected systems, and synthesizing the evidence to produce an understanding of *who* did *what*, *when*, and *how* to produce the current incident response.

Several guidelines for handling an incident response exist (Pipkin, 2000, Mandia, Prosis & Pepe, 2003, and Phillip, Cowen, & Davis, 2009). All of which consists of similar process steps. Mandia, et al. (Mandia, Prosis & Pepe, 2003) define seven components of incident response: pre-incident preparation, detection of incident, initial response, formulate response strategy, investigate the incident, reporting, and resolution (Mandia, Prosis & Pepe, 2003). These components and their relationships are shown in Figure 1. The pre-incident preparation occurs during the company's regular day-to-day operations, and includes the company's awareness program. Pipkin sums up this component as "all of the people involved with information must understand the important of security and what their part is in maintaining the security of the information that is entrusted to them" (Pipkin, 2000).

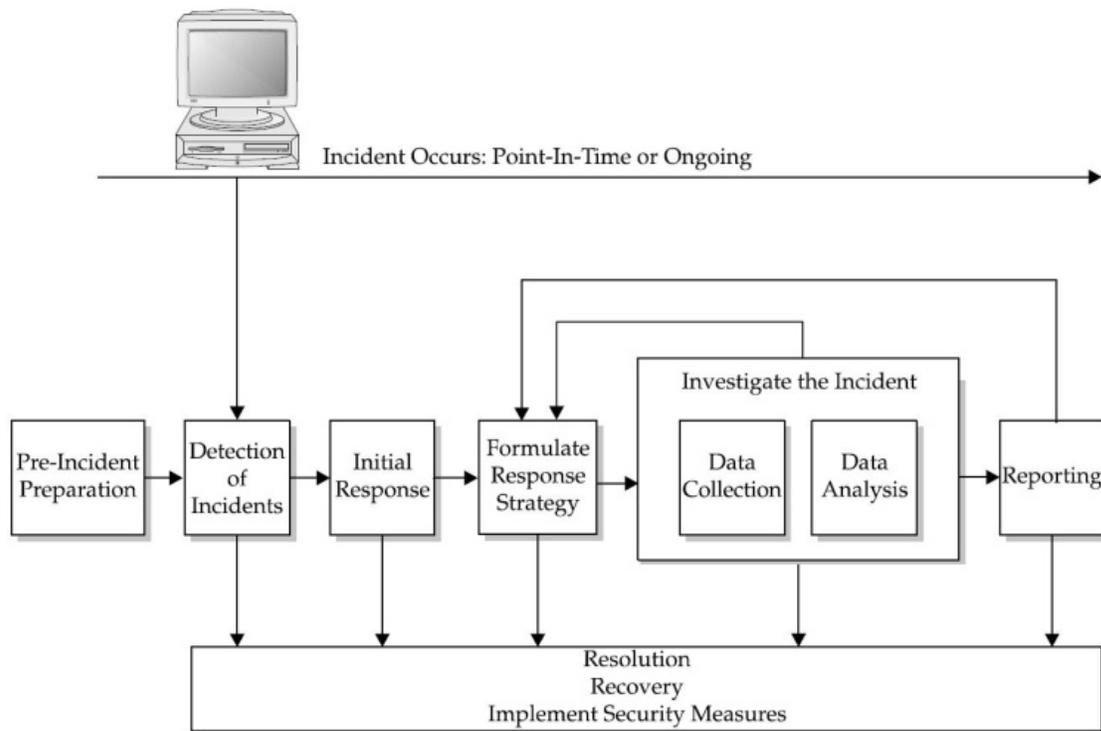


Figure 1 - Seven Components of Incident Response (Mandia, Prosis & Pepe, 2003).

Detection of an incident can be complex. It can occur through the use of an intrusion detection system (IDS) that notifies someone when anomalous activity on an information system is detected (Pipkin, 2000). Alternatively, the incident could also be detected by the user reporting anomalous behavior with the system.

Once an incident is confirmed, the initial response begins. The initial response involves gathering information about the incident so a response strategy can be formulated (Pipkin, 2000). The strategy is a structured approach to the incident based on the known facts that can include criminal or civil penalties. The component, investigating the incident, is broken down into two main parts, data collection and data analysis. During data collection, many pieces of data are collected from one or more systems, based on the type and extent of the incident. Data is always collected from the affected

system, but there are additional support systems that also may require collection like IDS logs, router logs, firewall logs, and authentication server logs.

On each affected system, both the volatile and non-volatile data is collected (National Institute of Justice, 2008). Volatile data includes the system's random access memory (RAM) or the central processing unit's cache memory, which becomes lost when the power is disconnected from the computer. Non-volatile data is the permanent data on the system like the data on the hard drive, memory stick, or basic input output system (BIOS). Forensic tools are used to collect data on the affected system looking for evidence of an event in malware, files, persistent network connections, deleted files, journal and master file tables. This component also includes collecting evidence of interest from the area around each effected system like notes, calendars, printed emails, wireless access points, CDs, DVDs, or memory sticks (National Institute of Justice, 2008).

Once the data is collected, examiners create working copies or duplicates of the data that the examiners then work with to construct a narrative of the event. A primary component of this timeline is the timeline of events leading up to, during, and after the incident. When multiple sources of data are collected into one timeline of events this timeline is called a super timeline (Guðjónsson, 2010). The super timeline is only one piece of information in the analysis and report, but a key one describing *when* events occurred.

1.2 Real Cases When a Super Timeline Was Needed

In the Casey Anthony trial (Jones, 2011), Mr. John Bradley, the author of a tool called Cacheback that retrieves Internet search history, was questioned about Casey Anthony's browsing history (Jones, 2011). The lawyer asked how much time transpired between the web browsing of a chloroform site and the next web site visited. His answer was three minutes. John Bradley was then asked if "three minutes was the longest time the user at the computer could view this particular chloroform material", which he answered, "that was correct" (Jones, 2011). Firefox version 2, the program in question, had tabbed browsing and the history file does not log when a website was closed so the user could have easily left the browser open and read it at a later time.

In another case, Dr. Conrad Murray, Michael Jackson's doctor, was questioned about his phone call timing just before, during, and just after Michael Jackson's death. The timeline for his cell phone records and testimony by Ms. Sade Anding indicated he was on the phone when Michael Jackson was found unresponsive. This timeline of events helped the prosecution to indicate Dr. Conrad Murray was wasting time with his personal life instead of caring for his superstar patient (Lloyd, Healy, & Klemack, 2011). Both of these cases point out that timeline information will timestamp a point in time, usually the execution time of a process, file, or action and not the termination time, so a process could start up three days before it infects the system creating a large gap of time to be accounted for to accurately recreate the incident. This can be particularly painful when analyzing and understanding what is on a timeline.

Both the Casey Anthony and Conrad Murray trials are examples of why timelines are important. Without a concise listing of temporal artifacts, the case may be lead astray due to misinformation or a lack of artifacts. This further confirms the need to create a concise super timeline of temporal artifacts.

1.3 Timeline the Data

When the examiner is collecting data from the computer system, eventually the data needs to be organized into a timeline of events for better comprehension (Edwards, 2011). This collection of data forms the super timeline where the examiner can make a better determination of how the incident occurred. The super timeline arranges the events by date and time so the examiner can determine temporal events that occurred just before, during, and after the incident. When these multiple data sources are collected together into one timeline it becomes easier to understand the events triggering the incident and can connect correlating information.

The two current processes for generating a super timeline are to manually generate the timeline, which is the most common and most time consuming, or to use the Log2timeline tool (Guðjónsson, 2010). The Log2timeline tool is an open source Perl script that is currently the only tool that can parse many of the various temporal artifacts within a computer operating system like Linux, Windows, or OS X. Log2timeline produces an output file of all artifacts that it supports found on the system with timestamps associated with the user's last interaction. The examiner can input the Log2timeline output into a database to produce queries and reports to capture key pieces of information in a quick concise manner. Searching the database allows the examiner to

cut through mountains of data to find the *needle in the haystack* quicker than looking at each piece of data separately. Since Log2timeline collects all the temporal artifacts into one timeline, the examiner can easily see temporal events that occurred just before, during, and after a file executes.

However, Log2timeline does not have a correlating and reporting module capable of highlighting abnormal activity because this database of information would be too large for one machine to store. The examiner must sort through the data to determine the abnormal activity in an efficient manner. This research helps the examiner cut through mountains of data to narrow the focus to the artifacts of interest.

1.4 Correlating Various Different Artifacts

Many native operating system (OS) and open source tools are commonly used to capture data from a Windows system. The network connections, running processes, deleted data, prefetch information, registry, recent web searches, recent documents, USB storage, and network drives are all common data collection items. This list may grow or shrink depending on the type of incident. If the incident is solely a web search issue then network connections and running processes could be skipped to minimize the processing of extraneous data.

A test VMware image was used for developing and testing the timeline analysis queries. For this test image a forensic incident occurred on the system. The system was then shutdown so the forensic artifacts could be collected. The artifacts were collected through an imaging process where the entire hard drive is captured one byte at a time and stored into a series of large files. These large files were then mounted to the analysis

system in a read-only manner so the super timeline tool, Log2timeline, could capture all the artifacts. Once Log2timeline finished capturing all the artifacts, the results were imported into a Microsoft database so simple query language (SQL) statements could extract out concise timelines of various temporal artifacts.

Several query statements were generated that range from general information to successively more detail. The query statements extract the following.

- All user's activities (excludes items not associated with a user)
- Web history traffic with logon and logoff times of users
 - Refined query to add keyword search ability
 - Further refined query to add a user
- Documents the user's recently accessed
- All logon, logoff, and executable program run times within a specific timeframe
 - All artifacts during a specific timeframe (from user logon through logoff)
- Only audit log artifacts for process start and end times within a specific timeframe

Additional correlation queries verify when processes started or ended by merging multiple data sources. The system's audit log captures many artifacts like logon, logoff, process begin, and process end time, but these times need to be correlated to other temporal items like the last access time of the file that was executed. Correlating these data points to registry entries and the master file table further confirm when actions occurred during the incident. It's this correlation and confirmation of data points across many different types of artifacts that make the use of a super timeline so powerful.

1.5 Research Goal

The research goal of this paper is to identify reusable queries that will extract concise timeline artifacts that allow the examiner to decisively determine the activities of a

computer system incident. This research goal is broken down into two requirements, temporal artifact collection and timeline artifact analysis.

Chapter 3 focuses on the collection of temporal artifacts, which is the first requirement of the research goal. The collection of temporal artifacts is achieved through the use of a timeline tool called Log2timeline. An image of the file system is captured using a graphical interface version of the Unix DD tool called Guymager. This image is moved to the analysis machine and mounted to the existing file system so the Log2timeline tool can parse the files. The tool captures the artifacts and stores them in a comma separated values file. This file is then imported into a Microsoft Access database where queries are developed to extract concise listings.

Chapter 4 discusses the second requirement, timeline artifact analysis. This analysis develops queries against an Access database like a query to list all user's logon and logoff times with all executable program's begin and end times. This allowed the examiner to chunk the timeline into sections making the hundreds of thousands of artifacts easier to consume. Then narrowing the incident down by a timeframe allowed the examiner to develop a follow-on query that used this timeframe to find more specifics about the incident. Each subsequent and more detailed query would narrow down the hundreds of thousands of artifacts to one hundred artifacts or less allowing the examiner to minimize the time used to analyze the incident and focus the analysis on the artifacts of interest. Using the queries discussed in this research will shorten the time required for analysis as well as narrow the sheer number of artifacts to examine.

1.6 Summary

This paper discusses the data collection and analysis phases within the *Investigate the Incident* component of the incident response methodology. This includes the background surrounding timeline data, the use of the super timeline tool, Log2timeline, to collect data on Windows-based computer systems, and the analysis of the resulting data. The related work section discusses similar tools used within the *Investigate the Incident* methodology. The temporal artifact collection chapter describes how the Log2timeline tool is used to capture the system's data. The timeline artifact analysis chapter then analyzes the use of simple queries language (SQL) to correlating data items from the super timeline tool output to produce reports with a clear understanding of what happened on the system during the incident.

A system's timeline of events will not solely discover everything about an incident, but it's an important piece in the incident response methodology. A timeline can connect who did what, when and how the user did it. The analysis of the results and search strings will produce an understanding of temporal events surrounding an incident.

II. Related Work

Inside the *Investigate the Incident* section of the incident response methodology is the data collection and data analysis processes (Mandia, Prosis & Pepe, 2003). When performing data collection, there are several tools capable of capturing timeline artifacts. Some tools are a suite of tools with many capabilities, while other tools are simple in design with one function. Using multiple tools to create a timeline of artifacts can be cumbersome and time consuming. Creating a super timeline of various different artifacts using one tool is ideal over using many different tools to accomplish the same task. This chapter discusses the available forensic timeline tools, and the anti-forensic tool that can alter timeline artifacts.

Some tools are freely available while other tools are commercial requiring payment before use. There are benefits to both free and paid tools. The free tools can offer capability with options the examiner can customize to the unique incident at hand, but the tool requires a higher level of experience with the tool and possibly programming languages. The commercial programs have a company supporting the tool but may not have all the options or capabilities to collect the evidence needed for the incident. Collecting a timeline of events from a system can provide key evidence in identifying what happened when and by whom. Two popular commercial tools, FTK and EnCase, are great tools with many options and capabilities, but neither currently provides a collective timeline of evidence. The free Log2timeline tool is a single capability tool that only produces a super timeline. The examiner can narrow the scope of the tool by limiting the search location of the tool or even limit the types of files it parses.

Wrapping up this chapter is a short section on the time and data focused anti-forensic tool Timestomp (Timestomp, 2010). This tool is capable of altering time and date information about files (Timestomp, 2010). When the time and or date data is altered, the evidence can lead the examiner astray, but careful examination can reveal the anti-forensic tool used to alter the evidence.

2.1 Open Source Versus Closed Source

Open source tools are tools that have freely available code that can be reviewed, downloaded and compiled (Altheide, Cory, Carvey, 2011). For forensics this is important because if an incident goes to trial, the validity of the tool's results could be called into question similar to the Cacheback tool in the Casey Anthony trial. With an open community of people reviewing and using the tool, issues with the tool are identified and corrected quickly. Open source tools are more flexible and portable than closed source tools since the code can be modified to meet specific needs or recompiled on the system in question. Many of the open source tools have binaries already compiled for each operating system so the examiner can simply download the tool, get familiar with its use, begin operating the tool, and begin collecting data. For forensic collection, most examiners understand the need for a command line tool. This means the tool will not have a graphical user interface or GUI. Without a GUI, the tool may be more difficult to use but this difficulty is offset by the fact that the tool will use less memory on the system so it's disrupting less of the evidence on the system. If the collection can occur across the network, the examiner's system can use a GUI tool and the overhead of the tool is spent on the examiner's system, not on the system where the incident occurred.

This method disrupts even less evidence as it comes in through the network to perform the data acquisition. Another advantage that open source tools have over closed source tools is the price. Open source tools are free.

Closed source tools are tools that are commercially sold for profit. Some of the most common forensic closed source tools are FTK by Access Data (<http://accessdata.com>), EnCase by Guidance Software (<http://guidancesoftware.com>), and Helix by E-Fense Carpe Datum (<http://www.e-fense.com>). The first big downside to a closed source tool is the price. These closed source tools can cost thousands of dollars for a single license, but these tools easily work across the network to capture data and image hard drives. Another downside of closed source tools is the code. The code is not freely available and cannot be reviewed to ensure it is accurately collecting the data. The closed source tool will have a robust comprehensive suite of tools since these tools usually have a dedicated staff to the tools development and maintenance. If a flaw in the code is found, a closed source tool company may not act quickly enough for it to matter in a trial case. For example, Adobe waited over a month to patch Linux and Mac user of their Reader program because they felt there was “little danger” due to more users on the Windows operating system (Keizer, 2012). If a similar mentality was used for FTK or EnCase customers, people may go to jail or be set free because of a mistake in code and no one may even be aware of the flaw in the code for months or longer since the examiner must be aware there is an update available and perform the update on the examiner’s system before the faulty code is replaced. The closed source code is not the only ones affected by bugs requiring critical updating. The widely popular open source tool md5deep suite of tools

written by Jesse Kornblum that calculates hashes of files in a recursive manner, was recently updated because of “a bug found when using the standard input, hangs on DFXML output, and ‘too many files open’ on Mac” (Kornblum, 2012).

2.2 Log2timeline

Creating the timeline is one of the key pieces the forensic examiner needs to determine what happened during an incident. There are a limited number of tools for creating super timelines. There are many tools that extract artifacts that produces specific timelines like file system metadata using the *fls* tool from the Sleuth Kit. If the examiner only needs the Windows registry entries then using the tool *RegRipper* or *yaru* (yet another registry utility) will produce this output. If the examiner wants to list the user’s viewing preferences the *sbag* tool can be used, which parses NTUSER.DAT and USRCLASS.DAT files. These three tools will only produce a narrow view of the user’s activities. The examiner would need to manually standardize, connect, and correlate the data to produce a quality super timeline. Instead of manually creating the super timeline, the Log2timeline tool can parse all of these artifacts and more to produce a single timeline in one file that can be easily analyzed using queries.

The Log2timeline tool is the de facto standard for creating super timelines. The System Administration, Networking, and Security (SANS) Institute developed the SANS Investigate Forensic Toolkit (SIFT) workstation that is a collection of the most widely accepted and used open source investigation forensic tools compiled into a Ubuntu or VMware download that is freely available (SANS Computer Forensics and Incident Response, 2012). The Log2timeline tool was added to the SIFT workstation suite of tools

in version 2.12, which is the currently available version of the SIFT workstation. SANS describes Log2timeline as a “new capability ... that provides an enormous value to [the] investigator” (SANS Computer Forensics and Incident Response, 2012).

Log2Timeline is a very effective tool for creating timelines. It is open source and currently has the largest file type support with the most extensive capabilities. Log2Timeline is a Perl script that parses the file structure and other artifacts within files. The tool has many input modules and several output modules that span across most modern operating systems.

The input modules are grouped into operating system specific groups. The examiner identifies the operating system when executing Log2timeline and the tool will automatically use all of the current modules available for that operating system. This allows the examiner to keep the same script for running the tool against all Windows 7 machines, even after the tool updates over time. If the examiner would like to narrow the search, then individual modules can be executed. Appendix A includes the full module listing for Log2timeline version 0.62.

The *webhist* input module for Log2timeline takes inputted files and parses them according to the standards for each web browsers file specification. Currently, the tool can parse history and bookmarks from the Chrome, Firefox, Opera, Internet Explorer, Internet Information Service (IIS), Safari, and Sol web browsers. With the IIS module, this means the tool can parse the web server as well as client systems. This can be particularly helpful when searching through an internal network for evidence when an

internal web server and client are affected in the incident. For Internet Explorer and Firefox the files are found in the following locations.

```
C:\Documents and Settings\\Local Settings\Temporary Internet Files\Content.IE5\  
C:\Documents and Settings\\Application Data\Mozilla\Firefox\Profiles\\history.dat  
(Jones & Belnai, 2010)
```

Figure 2 - Location of Internet Explorer and Firefox Web History.

The files are then parsed by Log2timeline which outputs the Internet activity in the output module of choice like a text file. There are stand alone web history extraction tools that only work on Internet history extraction like Pasco (sourceforge.net/projects/odessa/) or Web Historian (www.mandiant.com), but there is no need to operate and understand multiple tools if Log2timeline can retrieve all the data for the examiner. Since Log2timeline has a recursive search capability, the examiner can execute the tool against the entire hard drive image to retrieve all items it's capable of parsing.

The output modules are rather extensive too. The tool can output a standard comma separated values (CSV) file. CSV files are a fairly universal file format that delimits each item it outputs with a comma and are easily read by text editors, spreadsheet, and database programs. To build and analyze the super timeline, testing used the CSV option for the Log2timeline output module to allow for easy importing into a database.

Another output option is the SIMILE output module (<http://www.simile-widgets.org/timeline/>). The Log2timeline outputs in the standardize format so that a SIMILE widget can read and display a moving timeline of information, as seen in the figure below. While this seems like an ideal way to view Log2timeline's output, there were many cautions against using this tool when there are more than 200 items as the

SIMILE widget loads and operates very slowly. With the smallest Log2timeline output against a small 2Gb VMware image producing over 35,000 line items, this timeline widget did not seem practical to use for the overall forensic super timeline. This tool may be practical for a small window of time that requires an easy to view and movable graphical representation; otherwise, this tool is not practical for viewing the whole super timeline at once.

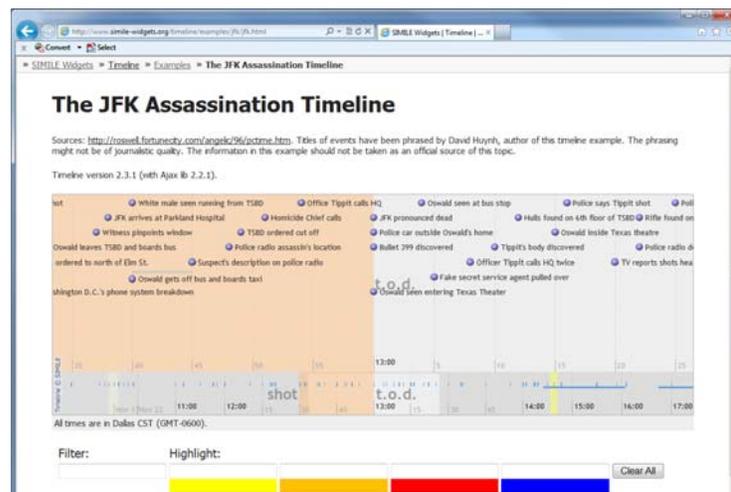


Figure 3 - Simile Widget Example (Simile Widget Example, 2009).

Both EnCase and FTK tools have various searching capabilities to find timestamp information but neither have a super timeline capability (Olsson & Boldt, 2009). Both of these tools have very extensive capabilities to recover the harder to find, more obscure data like alternate data streams, master file table data, and deleted files. Both tools give the examiner the ability to perform keyword searches and even narrow the search down by date and time, but neither tool presents the examiner with a full timeline of all the data collected. They are both critical tools, but if the examiner needs timeline information, they will need to use a separate tool like Log2timeline.

2.3 Timestomp

There are some anti-forensic tools to be aware of like Timestomp. An anti-forensic tool is a tool that distorts or changes the forensic value of data. The program Timestomp can alter the modification, access, and creation (MAC) dates that are stored for each file. According to the Forensic Wiki, “Timestomp cannot alter all 8 timestamp values, four of which exist in the \$STANDARD_INFORMATION attribute of an MFT entry, and the other four in \$FILE_NAME attribute” (Timestomp, 2010). If an attacker wants to cover his tracks he will likely use Timestomp to alter some or all of the timestamps of one or more files.

```
File Path: C:\Documents and Settings\P0WN3D\My Documents\test.txt
Created Date: 5/3/2009 2:30:08 AM
Last Accessed: 5/3/2009 2:31:39 AM
Last Modified: 5/3/2009 2:30:36 AM
```

Figure 4 - Original File MAC Times (Timestomp, 2010).

```
meterpreter > timestomp test.txt -f C:\\WINNT\\system32\\cmd.exe
[*] Setting MACE attributes on test.txt from C:\\WINNT\\system32\\cmd.exe
meterpreter > timestomp test.txt -v
Modified      : Tue Dec 07 08:00:00 -0500 1999
Accessed     : Sun May 03 05:14:51 -0400 2009
Created      : Tue Dec 07 08:00:00 -0500 1999
Entry Modified: Sun May 03 05:11:16 -0400 2009
```

Figure 5 - MAC Times after Timestomp (Timestomp, 2010).

This makes creation of an accurate timeline more difficult. Figure 4 shows the original MAC times for a file, test.txt that was created May 3, 2009. When the hacker uses Timestomp on the test.txt file to mimic the MAC times of the file cmd.exe (Figure 5), the hacker now has new dates and times that can throw off the investigation. Once Timestomp is used to tamper with file attributes there are remnants of the Timestomp program within the file system that also must be cleaned to ensure there are no traces that

Timestomp was even on the system. This includes deleting the Timestomp tool then clearing the recycle bin, the master file table, the journal table, the recent files listing, the prefetch directory, the super prefetch directory, the pagefile, the hibernation file, the shadow copy, the registry, and the Windows event logs. If the Timestomp tool was used and the system has not been rebooted then the tool may still be in live memory. After the tool is deleted and the system is rebooted, another tool from an external operating system must be used to clean the hard drive's empty space since the Timestomp tool executable is not removed from the hard drive after the deletion and clearing of the recycle bin occurs. If the Disk Defragmenter is used then the original location where the tool was written on the hard drive may be overwritten by other files, which will destroy the evidence the program was on the system. If Disk Defragmenter does not overwrite the location then a tool like Bcwipe by Jetico (<http://www.jetico.com/wiping-bcwipe/>) is needed to perform an erase of a particular file and hard drive location. It's a very effective tool, but the hacker now has the same issues as with Timestomp, there are remnants of Bcwipe left on the system that must be cleaned. It can be very difficult to completely clean a system that a program resided and ran on a system since there are so many ways a Windows system logs and tracks a file on the file system. There by cleaning traces that a cleaning occurred, even if done from a different operating system, are left behind.

The hacker can avoid detection of the Timestomp tool by renaming the tool before copying the tool to the system. Should this be suspected, the examiner can find the new name of the Timestomp program through the use of the Hashdeep tool and using its

capability to MD5 or SHA-1 hash all the files on the file system. The examiner can search for the Timestomp hash value to find the rogue tool. If a hash matches the Timestomp hash values then the program likely resided on the system and additional investigative measures should be taken.

2.4 Summary

There are many free and paid-only tools available capable of collecting evidence from an incident. The popular paid programs like FTK and EnCase has many features capable of discovering most evidence no matter how deep it's tucked away, but neither program has a timeline capability like Log2timeline. The freely available Log2timeline tool only collects data to output super timelines. This tool cannot be the examiner's only tool in the tool bag but it is certainly an important one. There are also anti-forensic tools available. The most popular tool used to alter date and or time stamping of files is Timestomp. While this is an effective tool it's difficult to erase the evidence that an anti-forensic tool was used.

III. Temporal Artifact Collection

Focusing on the data collection process within the *Investigate the Incident* component of the incident response methodology, the super timeline provides the examiner with vital information about the incident. The research goal is to identify reusable queries that will extract concise timeline artifacts that allow the examiner to decisively determine the activities of computer system incident. This research goal is broken down into two requirements, temporal artifact collection and timeline artifact analysis. This chapter focuses on the collection of temporal artifacts, which is the first requirement to meet the research goal. The timeline artifact analysis is the second requirement of the research goal that is discussed in Chapter 4 below.

To perform the data collection of temporal artifacts into a super timeline, the Log2timeline tool (www.log2timeline.org) is used. This Perl script was written and tested in the Linux and Mac operating systems with minimal testing in the Windows environment so additional installation of packages were required for proper operation in Windows, which is presented in Section 3.1. Because timeline analysis occurs on hard drive images, the methodology and tools associated with collecting, mounting, and executing the Log2timeline tool on these image are also discussed. Each test system affected by an incident was imaged so the entire hard drive was available for parsing by the tool. The hard drive image was then mounted and parsed. The tool parsed each file and determined what data could be extracted, and then it stored this data to an output file. The output file, or super timeline, was then verified for accuracy. This output file is the collection of data required to meet the research goal.

3.1 Installing Log2timeline on a Windows-Based System

The Log2timeline version 0.62 tool (<http://log2timeline.net/>) is used in all tests. The tool was run on a Windows 7 SP1 64-bit operating system. Since this tool is a Perl script, Perl for Windows was installed (<http://www.perl.org/>). There are two flavors of Perl, Active Perl and Strawberry Perl. The Active Perl version 5.12.4, build 1205, was used according to the installation instructions for using Log2timeline in Windows. Once the ActivePerl program is installed on the examiner's machine any Perl program can be executed by using the full name of the filename, for example, log2timeline.pl would execute the Log2timeline tool. There is no need to be in the Perl directory upon execution of a Perl script because the ActivePerl program appends the install directory to Path statement for Windows. This Path variable holds all the directories that executable programs reside. Executing an `echo %path%` at the Windows command line will display the Path statement and verify that Perl was added to this list of executable directories.

To get Log2timeline to operate properly with Perl on a Windows-based system there were several files that required changing. Since Log2timeline is open source, there were many inputs from various programmers to help develop Kristin Guðjónsson's timeline tool. Chris Pogue was the lead for porting this tool from the Linux and Mac operating systems over to Windows. The Windows installation instructions are located in the Log2timeleine/Docs folder in the Install.txt file. The installation of several packages is required by using the Perl Package Manager (PPM). The list of commands and the packages to install is shown in the figure below.

```

Download and install ActiveState Perl (latest version)
Open command prompt and run the following commands (install dependencies):

ppm install datetime
ppm install win32::api
ppm install date::manip
ppm install xml::libxml
ppm install carp::assert
ppm install digest::crc
ppm install data::hexify
ppm install image::exiftool
ppm install file::mork
ppm install datetime::format::strptime
ppm install parse::win32registry
ppm install html::scrubber

Download the latest source code for log2timeline
Download two additional libraries:
"http://search.cpan.org/CPAN/authors/id/B/BD/BDFOY/Mac-PropertyList-1.33.tar.gz"
"http://search.cpan.org/CPAN/authors/id/S/SI/SIXTEASE/XML-Entities-1.0000.tar.gz"

Uncompress

Inside the XML-Entities:
Copy the content of the lib/XML folder to c:/perl/lib/XML/

Inside the Mac-PropertyList:
Create the directory c:/perl/lib/Mac
Copy the content of the lib/* to c:/perl/lib/Mac

Inside the log2timeline directory
Delete the file lib/Log2t/input/pcap.pm
Copy the content of the lib/Parse/* to c:/perl/lib/Parse/
Copy the content of the folder lib/Log2t to c:/perl/lib/Log2t/*
Copy lib/Log2Timeline.pm to c:/perl/lib/
Copy log2timeline to c:/perl/bin/log2timeline.pl
Copy l2t_process to c:/perl/bin/l2t_process.pl
Copy timescanner to c:/perl/bin/timescanner.pl

```

Figure 6 - Windows Installation Instructions for Log2timeline (\\log2timeline\docs\install.txt).

3.2 Executing Log2timeline

Since Log2timeline has input modules in groups based on the operating system, telling the tool the operating system with the `-f` option executes the tool with all the modules available for the target operating system. Capturing everything has the least likelihood of an oversight but the tradeoff is that it also requires several hours to sift through all the collected data.

```

log2timeline.pl -z local -f %_OS% -v -r -c -l %_outDir%ERRORS.txt
-p -w %_outDir%Log2Timeline.csv %_searchDir%

```

The command above displays the command used to execute the Log2timeline tool from within a batch file. To help understand the command, anything that begins and ends with a percent sign (%) is a variable name so %_OS% is the variable that held the name of the operating system. The variable %_outDir% was the variable that stored the output directory, where all the output files were stored. The options used were -z to input the time zone the tool should offset each file's date and timestamp. This option can be omitted when executing the tool and the tool will use local time. Additionally, a time zone can be identified by using EST or US/Eastern, both of these are the same time zone identifier for the East Coast time zone. To see a comprehensive list of time zones execute the tool using the command: `log2timeline.pl -z list`. This will list all of the -z options the tool will accept. The -v added verbosity to the error output, and the -l option created a log file of the errors. The -r option was used to tell the tool to work recursively, so if the tool found any subdirectories from the starting location it searched in all subdirectories until exhausted. The -c option added a column to the output file and calculated the MD5 sum for each file accessed. The -p option tells the tool to use preprocessing so files are identified and parsed easier. The -w option tells the tool where to output the results, and the last variable %_searchDir% is the location to begin parsing files.

3.3 Verifying Log2timeline

Once Log2timeline was installed it was verified against several small directories. The tool was run against a small directory of files and the results were compared to the results of tools that also perform the same tasks. For comparing the MAC times, the Windows

DIR command was used. The DIR command was run three times, once for each date (creation, last accessed, last written). The -t option in the DIR command controlled this output. Using the /t:c option, the command outputted a listing of the files by its creation date. Using the /t:a option captured the listing by the file's last accessed date, and the /t:w option listed the files by its last written date. The other options used in the DIR command was the /a option to find all files even hidden and system files, the /s option to recursively search through subdirectories, the /-c option to remove the commas in the size field of the file (allowed for quicker post-processing), and the /o:d option that ordered the output by date, oldest first.

```
dir /t:c /a /s /-c /q /o:d %_searchDir% >> %_outDir%DirListC.txt - Creation  
dir /t:a /a /s /-c /q /o:d %_searchDir% >> %_outDir%DirListA.txt - Accessed  
dir /t:w /a /s /-c /q /o:d %_searchDir% >> %_outDir%DirListW.txt - Written
```

Figure 7 - Capturing MAC Times with Windows DIR Command.

A batch file was used to convert the outputted files into a comma separated value (CSV) list so it could be imported and compared to the Log2timeline output of the same directory. The output of the commands in Figure 8 produces the text shown in Figure 9. This will not compare against the CSV file that Log2timeline outputs so the batch file in Figure 10 was written to convert the text into a CSV type file.

```

Volume in drive C has no label.
Volume Serial Number is 8C76-4B6F

Directory of c:\
06/06/2007 09:58 AM          0 BUILTIN\Administrators MSDOS.SYS
06/06/2007 09:58 AM          0 BUILTIN\Administrators CONFIG.SYS
06/06/2007 09:58 AM          0 BUILTIN\Administrators IO.SYS
06/06/2007 09:58 AM          0 BUILTIN\Administrators AUTOEXEC.BAT
06/29/2011 10:48 AM      805306368 ...      pagefile.sys
07/06/2011 10:20 AM    <DIR>      BUILTIN\Administrators System Volume Information
07/06/2011 10:55 AM          211 BUILTIN\Administrators boot.ini
07/06/2011 10:55 AM      47564 BUILTIN\Administrators NTDETECT.COM
07/06/2011 10:55 AM      250032 BUILTIN\Administrators ntldr
07/06/2011 10:55 AM    <DIR>      VICTIM\user          RECYCLER
01/22/2012 02:48 PM    <DIR>      BUILTIN\Administrators WINDOWS
01/22/2012 02:48 PM    <DIR>      BUILTIN\Administrators Documents and Settings
01/22/2012 02:49 PM    <DIR>      BUILTIN\Administrators Program Files
01/22/2012 02:50 PM          0 VICTIM\user          DirList.txt
          9 File(s)          805604175 bytes

Directory of c:\Documents and Settings
07/06/2011 10:16 AM    <DIR>      BUILTIN\Administrators Default User
07/06/2011 10:16 AM    <DIR>      BUILTIN\Administrators LocalService
07/06/2011 10:16 AM    <DIR>      BUILTIN\Administrators NetworkService
01/22/2012 02:48 PM    <DIR>      BUILTIN\Administrators ..
01/22/2012 02:48 PM    <DIR>      BUILTIN\Administrators .
01/22/2012 02:48 PM    <DIR>      BUILTIN\Administrators All Users
01/22/2012 02:48 PM    <DIR>      BUILTIN\Administrators user
          0 File(s)          0 bytes

```

Figure 8 – Example Output from One of the Three DIR Commands.

```

ECHO Converting Directory Listing to CSV
ECHO =====
REM - Convert DirList.txt to CSV File, DirList.CSV
REM *****
FOR /F "tokens=1,2*" %%G IN (%_outDir%DirList.txt) DO IF NOT %%G == Volume IF NOT %%H == File(s) IF
NOT %%H == Dir(s) IF NOT %%H == Files (echo %%G %%H %%I>> %_outDir%DirList2.txt)
Echo Part 1 of 3 Complete for Directory Processing

FOR /F "tokens=1,2*" %%G IN (%_outDir%DirList2.txt) DO IF %%G == Directory (echo %%G %%G %%G %%G
%%G %%I>> %_outDir%DirList3.txt) ELSE (echo %%G %%H %%I>> %_outDir%DirList3.txt)
Echo Part 2 of 3 Complete for Directory Processing

REM - NOTE: Must use !_dir! instead of %_dir% since it's in the executing line of a loop
FOR /F "tokens=1,2,3,4,5*" %%G IN (%_outDir%DirList3.txt) DO @IF NOT "%L" == "." IF NOT "%L" ==
".." IF %%G == Directory (
    (SET _dir=%L)
) ELSE (echo %%G,%%H %%I,%%J,%%K,!_dir!\%L >> %_outDir%DirList4.txt)

REM - the FOR /F command will parse an input, usually a file, in the parentheses. If the DELIMS
REM - keyword is not used then the line is delimited by a space. Using DELIMS= sets the delimiter
REM - character(s). The double percent is how the variables are stored from the line read in. %%G
REM - IN (filename) will read the first piece of text into %%G, the next piece of text will
REM - automatically be stored in %%H then %%I and so on until the TOKENS run out. If the '*' is
REM - Tokens=1,2* then variables %%G, %%H, & %%I are used at the end of the tokens, then one more
REM - variable past the tokens is created. If used when parsing the line. If there are four
REM - delimiters and only three variables, then the rest of the line is stored in the last
REM - variable, but only when using the '*' at the end of the Tokens.

Echo Part 3 of 3 Complete for Directory Processing
Echo Copying Directory Processing to CSV file
REM - Copy Header Template file to the beginning of the Directory List so each column has a name
Copy /b %-dp0DirListHeader.txt+%_outDir%DirList4.txt %_outDir%DirList.csv
Echo Cleaning up...
Echo Part 4 Complete
Del %_outDir%DirList2.txt
Del %_outDir%DirList3.txt
Del %_outDir%DirList4.txt
Echo Directory Listing Complete, Directory List is Located at: %_outDir%DirList.csv
Echo.

```

Figure 9 - Batch File Converting Directory Command Output to CSV File

```

Date, Time, Size, User, Filename
06/06/2007,09:58 AM,0,BUILTIN\Administrators,c:\MSDOS.SYS
06/06/2007,09:58 AM,0,BUILTIN\Administrators,c:\CONFIG.SYS
06/06/2007,09:58 AM,0,BUILTIN\Administrators,c:\IO.SYS
06/06/2007,09:58 AM,0,BUILTIN\Administrators,c:\AUTOEXEC.BAT
06/29/2011,10:48 AM,805306368,...,c:\pagefile.sys
07/06/2011,10:20 AM,<DIR>,BUILTIN\Administrators,c:\System
07/06/2011,10:55 AM,211,BUILTIN\Administrators,c:\boot.ini
07/06/2011,10:55 AM,47564,BUILTIN\Administrators,c:\NTDETECT.COM
07/06/2011,10:55 AM,250032,BUILTIN\Administrators,c:\ntldr
07/06/2011,10:55 AM,<DIR>,VICTIM\user,c:\RECYCLER
01/22/2012,02:48 PM,<DIR>,BUILTIN\Administrators,c:\WINDOWS
01/22/2012,02:48 PM,<DIR>,BUILTIN\Administrators,c:\Documents
01/22/2012,02:49 PM,<DIR>,BUILTIN\Administrators,c:\Program
01/22/2012,02:50 PM,0,VICTIM\user,c:\DirList.txt
07/06/2011,10:16 AM,<DIR>,BUILTIN\Administrators,c:\DocumentsandSettings\Default
07/06/2011,10:16 AM,<DIR>,BUILTIN\Administrators,c:\DocumentsandSettings\LocalService
07/06/2011,10:16 AM,<DIR>,BUILTIN\Administrators,c:\DocumentsandSettings\NetworkService
01/22/2012,02:48 PM,<DIR>,BUILTIN\Administrators,c:\DocumentsandSettings\All
01/22/2012,02:48 PM,<DIR>,BUILTIN\Administrators,c:\DocumentsandSettings\user

```

Figure 10 – Example Output from Batch File

7	6/6/2007	10:56:06 EST	M...	LNK	Shortcut L Modified	-	VICTIM	C:/WINDOWS/system32/accwiz.exe
8	8/4/2004	0:56:47 EST	.A..	LNK	Shortcut L Access	-	VICTIM	C:/WINDOWS/system32/accwiz.exe
9	6/6/2007	10:56:06 EST	..CB	LNK	Shortcut L Created	-	VICTIM	C:/WINDOWS/system32/accwiz.exe
10	6/6/2007	10:56:13 EST	M...	LNK	Shortcut L Modified	-	VICTIM	C:/WINDOWS/system32/calc.exe
11	8/23/2001	8:00:00 EST	.A..	LNK	Shortcut L Access	-	VICTIM	C:/WINDOWS/system32/calc.exe
12	6/6/2007	10:56:13 EST	..CB	LNK	Shortcut L Created	-	VICTIM	C:/WINDOWS/system32/calc.exe

Figure 11 - Example Log2timeline CSV Output Viewed in MS Excel.

The above figure displays the converted original text that was outputted from one of the three DIR commands shown above. This file can now be imported into Microsoft Excel or Access for a comparison to the CSV file Log2timeline outputted. The Log2timeline output uses several additional columns but adds one in particular, the MACB column, to annotate each file's attribute: modified, accessed, or created. When comparing the batch file's output with the Log2timeline's output, the outputs were comparable. This provided validation and credibility the Log2timeline tool is operating correctly and accurately portraying the file system.

3.4 Mounting VMware Image for Log2timeline Access

Several VMware images were used as test incidents. Each image was mounted using the VMware Virtual Disk Development Kit (VDDK). When the VDDK is installed on the system, the tool `vmware-mount.exe` is installed also. This `vmware-mount` tool allows a VMware image to be mounted as a drive in the Windows file explorer environment. The tool allows the VMware image to be mounted to the local system as a drive letter, or the tool can mount images remotely across a network. A username and password from the VMware image must be used to mount the image to the Windows file system. If the account used is not an account with administrator privileges then many files are locked from access. The `vmware-mount` tool retains the permissions of the account used to mount the image to the file structure. For the best results, each image was mounted with an administrator account to ensure the maximum access to the VMware image's file structure. Even with administrator access there were issues accessing many parts of the file system. The master file table was not accessible nor was the event logs. This limited the evidence collection that Log2timeline was capable of parsing. This method might be useful for a quick scan of the drive but ultimately was not useful since not all the evidence was accessible by the tool. Each VMware image was mounted using the following command where the VMware image was mounted to the local system's file structure as the `Z:\` drive.

```
VMware-mount z: E:\My Virtual Machines\WinXP\Disk1.vmdk /u:userid  
/p:password
```

3.5 Known Error with Log2timeline

The Log2timelin command above was originally used to collect artifacts from the test images. It was noticed when the `-c` option was used with the command there was a significantly smaller output from the tool. When the `-c` option is omitted, the tool produced the expected output. The issue was discussed with Kristin Guðjónsson through email to discover there was an error in the code. Because of this issue, the File Checksum Integrity Verifier (FCIV) tool was used to capture MD5 hashes of the files. The FCIV tool is a Microsoft tool that is capable of “computing the MD5 or SHA-1 cryptographic hash values” (<http://support.microsoft.com/kb/841290>).

3.6 DD Image Tool

Since the vmware-mount tool was limiting the Log2timeline acquisition of artifacts, the DD Linux tool was used to acquire an image of the system. Then, Log2timeline was run against the DD image file. Using the DD image tool ensured access to files not normally accessible like the \$MFT, or master file table, and the deleted records in the empty space of the hard drive.

The DEFT, version 7 (www.deftlinux.net), incident response boot DVD was used to capture an image of the VMware file system. Once the VMware image was booted from the DEFT DVD, the DEFT Live CD option was selected and the Guymager tool was used to capture a raw DD image of the VMware’s hard drive.

An external hard drive was the destination for the image from the Guymager tool. There was an issue when a USB 3.0 external hard drives was used with the VMware image. When the USB 3.0 hard drive was connected, the VMware application would not

recognize the hard drive. It showed a USB hard drive was connected to the system but was not available for mounting in the file manager application. Once the hard drive was connected to a USB 2.0 port, the VMware application recognized the hard drive and the hard drive was visible in the file manager application. While in the file manager viewer and upon right-clicking the drive, two options appeared on the pop-up menu, *mount read-only* and *mount drive*. Choosing the *mount drive* option allowed the Guymager tool to access and write to the hard drive. Once this option was chosen the hard drive appeared as an available device in the drive listing from within the Guymager tool.

The Guymager tool is a graphical interface front-end to the DD tool. Once the tool loads up, it displays all the hard drive and flash memory devices mounted to the system. Right-click the VMware image and choose Acquire Image from the pop-up menu then the acquire image menu pops up. Choosing the Linux Raw DD image capture option, identifying a destination drive and filename, and by clicking the ok button the Guymager tool begins capturing an image of the VMware drive.

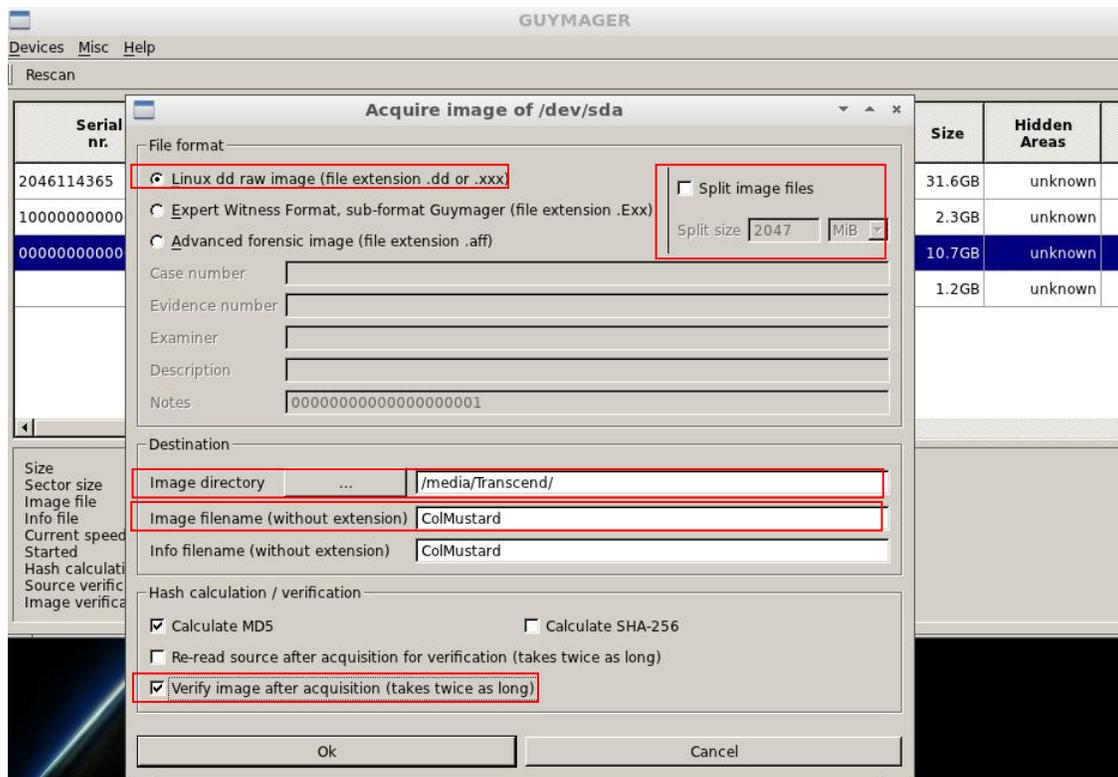


Figure 12 - Guymager Options Used to Acquire Col Mustard's VMware Image.

The Split image files option was used even though in the figure above it is not checked. This option was originally not used but the Guymager tool would crash without this option. According to NTFS.com, the limit for Fat32 file systems is 4GB for one file. This was likely the cause of Guymager crashing since the resulting image was 10GB for a 10GB VMware image and the destination drive was an external hard drive formatted as FAT. Once the Guymager tool completed the acquisition of the drive, it performed a verification of what it copied. When it is completed, the drive image is finished and the screen will look similar to the following.

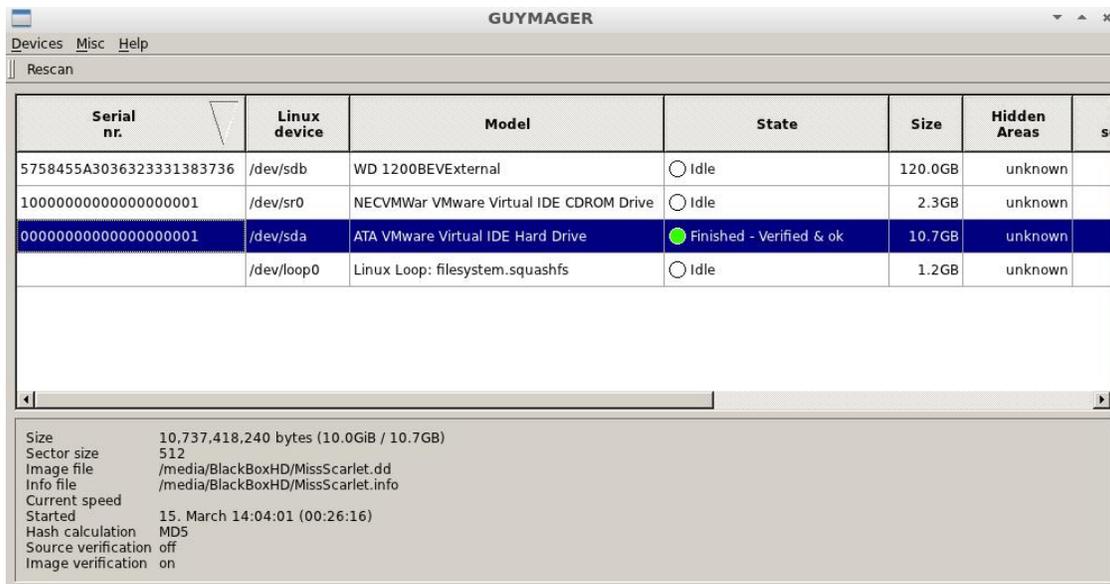


Figure 13 - Guymager Finished Acquiring and Verifying Image.

The resulting Guymager image was copied over to the analysis system so the Log2timeline tool could parse artifacts from the image.

3.7 Collection Process

In order for the Log2timeline tool to parse the DD image, the image needed to be mounted since Log2timeline cannot parse a DD image directly. One possible tool is OSFMount (www.osforensics.com). The OSFMount tool mounts the DD image to the Windows operating system as a read only hard drive. Just like the VMware tool vmware-mount, the OSF Mount tool retained the Windows permissions so not all of the drive was accessible. The Mount Image Pro, version 4, tool (www.mountimage.com) was ultimately used to mount the DD image so all areas of the hard drive were accessible. This tool allowed Log2timeline to parse items like the master file table, allowed full access to all folders, and displayed a separate folder tree of deleted items. This allows the Log2timeline tool to parse the entire file system without requiring permissions for

When the Mount Disk button was used the files look just like OSF Mount and the VMware tool, the disk-level files like \$MFT were not visible nor accessible. When the Mount Filesystem button is used then the full file structure is visible as seen in the Figure 15.

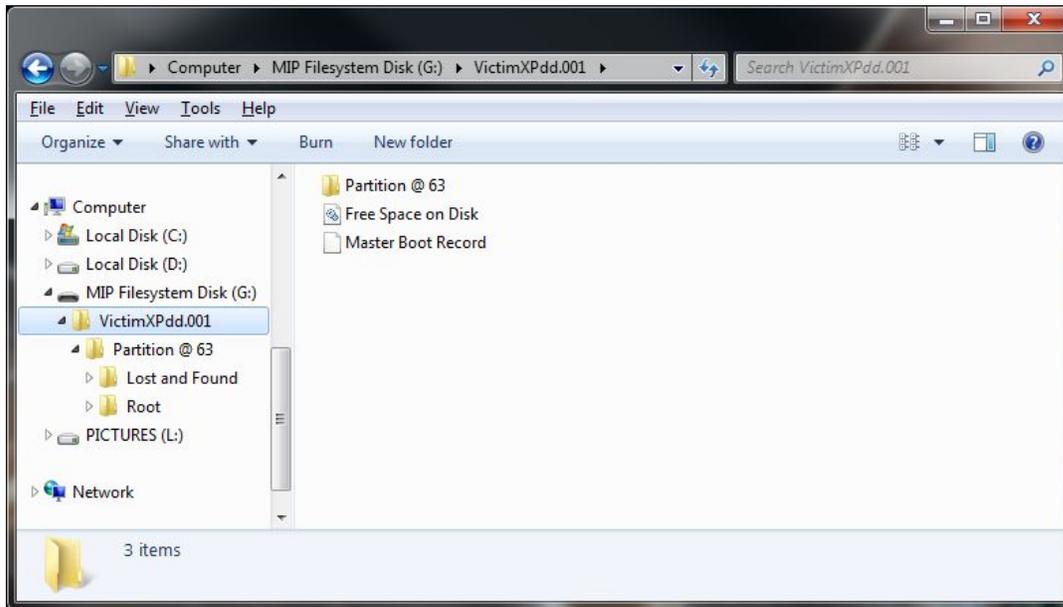


Figure 15 - Windows File Manager View of Mounted DD Image Using Mount Image Pro.

Looking at the above figure, when exploring the Lost and Found folder the examiner will discover deleted files are available for parsing. Exploring the Root folder will display the typical Windows file structure.

The Log2timeline tool was executed against several of these mounted images. The images had known incidents inside each one. The Log2timeline tool parsed all the artifacts and wrote the items to a comma separated file. Each resulting Log2timeline file was converted into an Access database where simple queries language (SQL) statements extracted the important information about the incident. The following command was executed to begin parsing the mounted DD image (M:\) by Log2timeline.

```
log2timeline.pl -z local -f WinXP -r -l  
C:\supertimeline\ColMustardDD\Log2TimelineERRORS.txt -p -w  
C:\supertimeline\ColMustardDD\Log2Timeline.csv m:\
```

The Log2timeline output directories were changed when each DD image was captured so each folder contained the DD image and the two outputted Log2timeline files. The first outputted file, Log2timelineERRORS.txt, collected any errors Log2timeline encountered. The other file outputted by the tool was the Log2timeline.csv file, which contained all the parsed artifacts from the DD image. The error file was reviewed for lost evidence due to the tool's inability to access files or directories, and the second file was imported into Microsoft Access for analysis of the artifacts.

3.8 Summary

Within the data collection process the examiner must extract all the artifacts from the affected systems. Installing the Perl script tool Log2timeline required extra attention in order for the tool to operate correctly on a Windows-based system. Verifying the tool's output is essential before the actual evidence collection occurs since the examiner must have accurate data. Once the tool is verified, Log2timeline can be executed against a system to create a super timeline. To ensure Log2timeline extracted all possible data from the system, the system's hard drive was duplicated using the DD tool. The image was mounted to the Windows-based analysis machine where the Log2timeline tool collected all the artifacts and stored them to a comma separated values file. Next, the discussion will transition into the data analysis process of the *Investigate the Incident* component of the incident response methodology where the collected data will be analyzed for evidence.

IV. Timeline Artifact Analysis

In the data analysis process, the examiner must interpret the collected data to make sense of it all. The examiner must answer who did what, when, how and to whom. This requires collecting and extracting the data, correlating the artifacts, and determining how the incident occurred. The super timeline aids the examiner in the *Reporting* component of the incident response methodology since it details what occurred when and by whom during the incident.

The research goal is to identify queries that extract concise timeline artifacts that allow the examiner to decisively determine the activities of a computer system incident. The timeline artifact analysis is the second requirement of the research goal that is discussed in this chapter. This chapter is broken down into three sections, the developed queries, the data used, and the results from the queries on the test data.

The query statements sift through hundreds of thousands of artifacts to extract the artifacts of interest. Using specific queries, the examiner can extract lists of less than one hundred artifacts to examine a smaller portion of the overall super timeline. With a smaller super timeline of artifacts, the examiner can correlate temporal artifacts to determine what activities were conducted by whom. Additionally, the examiner can reduce the time and effort deciphering the artifacts because the queries provide a concise view of the activities.

4.1 Developed Queries

This chapter discusses the correlation of all the parsed data by using simple query language statements to produce concise listings of artifacts. The various artifacts like

program use, file access, metadata, and event logs are just some of the artifacts included in the output. The data is imported into a Microsoft Access database where simple queries created a concise super timeline of the incident as well as more precise mini timeline of events. The queries search for a correlation between various artifacts and users to reconstruct the timeline of events that occurred just before, during, and after the incident. The following queries were designed to correlate the artifacts.

4.1.1 All Artifacts Associated to Users

The examiner can start off with this query, which displays artifacts with an entry in the user's name field. This query gives the examiner an overall view of a user's general activity. The query displays artifacts like the user assist registry key, which is created when the user executes a program like notepad. But it does not display operating system actions like the Master File Table or Prefetch artifacts for the same execution of notepad. Below is some example data, and this query will only display the highlighted row.

Table 1 - Example Artifacts for User's Activity Query.

Source	Type	User	Filename
NTFS \$MFT	\$\$I [MAC.] time	-	/WINDOWS/Prefetch/NOTEPAD.EXE-336351A9.pf
UserAssist key	Time of Launch	Mustard	UEME_RUNPATH:C:/WINDOWS/system32/notepad.exe [Count: 1]
XP Prefetch	Last run	-	NOTEPAD.EXE-336351A9.pf - [NOTEPAD.EXE] was executed - run count [9]- full path: [C:/WINDOWS/SYSTEM32/NOTEPAD.EXE]

When looking at the results, the examiner can look for anomalous user's activities, and instead of sorting through 303,000 artifacts this query reduced the listing to just over 3,000. This will certainly save the examiner from searching through so many artifacts. An example statement for this query is defined below.

```

SELECT date, time, user, filename
FROM Log2timeline
WHERE Not user = " "
ORDER BY date DESC, time DESC;

```

4.1.2 Web History

A common type of incident is when a user intentionally or unintentionally goes to a website that creates an incident. The examiner can extract out the web history artifacts to list the user's visited websites. Add the logon and logoff artifacts and the query allows the examiner to determine what user was logged in when a website was visited. The logon and logoff events were found by querying for the audit log Event ID 528 and 538 for a Windows XP system, and Event ID 4624 and 4647 for a Windows 7 system (UltimateWindowsSecurity.com). The web history artifacts are found by searching for artifacts where the source is logged as "Webhist". In the query below the statement uses the code `filename Like "*Security/528;*"` , which finds `Security/528;` anywhere in the artifact. This is different from `source = "WEBHIST"`, since this code will only find artifacts where the source is logged exactly as `WEBHIST`.

```

SELECT date, time, source, user, filename
FROM Log2timeline
WHERE (source = "WEBHIST") OR (filename Like "*Security/528;*" Or filename
    Like "*Security/538;*")
ORDER BY date DESC, time DESC;

```

The examiner can narrow down the query based on artifacts initially found. If the user Mustard has several gambling websites listed on the above query, the examiner would

narrow the query to the user Mustard and search for the keyword “gambling”. The following example query would narrow the listing.

```
SELECT date, time, source, user, filename
FROM Log2timeline
WHERE (source = "WEBHIST") AND (user = "mustard") AND ((filename Like
    "*gambling*") OR (filename Like "*Security/528;*") Or (filename Like
    "*Security/538;*"))
ORDER BY date DESC, time DESC;
```

4.1.3 Recent Document History

When a user accesses a file or resource a link file is created in the user’s recent document folder. These artifacts are valuable as it can tell the examiner what files and resources the user accessed recently. Below is an example query for recent documents.

```
SELECT date, time, source, user, filename
FROM Log2timeline
WHERE filename Like "*recent*"
ORDER BY date DESC, time DESC;
```

If the examiner finds a program that ran from a location other than the hard drive, then the examiner should look for USBstor artifacts. The examiner can refine the query like the following example query to find any USB devices.

```
SELECT Date, Time, source, description
FROM Log2Timeline
WHERE description Like "*usbstor*"
ORDER BY Date DESC, Time DESC;
```

4.1.4 User's Activity within a Timeframe

Once the artifacts point the examiner to a specific user or specific timeframe, the super timeline can be queried for a timeframe. A query for a specific timeframe and filtering the artifacts so only the executable programs are listed, allows the examiner to focus on the main activities. Once the examiner understands the user's activities the query can be expanded to include all artifacts during the timeframe to add correlative temporal artifacts to the timeline. Correlating the additional artifacts will allow the examiner to fully understand the activities during the incident. The basic query for executable artifacts within a specific timeframe would look like the following.

```
SELECT Date, Time, source, User, File_Description
FROM Log2Timeline
WHERE (((Date)=#4/27/2012#) AND ((Time)>#17:50:52# And (Time)<#18:21:8#) AND
      ((source = "Event Log") AND ((File_Description) Like "*528*" Or
      (File_Description) Like "*538*")) OR (((Date)=#4/27/2012#) AND
      ((Time)>#17:50:52# And (Time)<#18:21:8#) AND ((File_Description) Like
      "*.exe*"))
ORDER BY Date DESC, Time DESC;
```

This query looks rather long and complicated but this query displays executable programs ((File_Description) Like "*.exe*") and user's logon and logoff times ((File_Description) Like "*528*" or "*538*") from 27 April 2012 ((Date)=#4/27/2012#)) between the time 17:50:52 and 18:21:08 ((Time)>#17:50:52# And (Time)<#18:21:8#)). Instead of using an equal sign, the greater than and less than symbol was used to create an inclusive timeframe for the query.

Once the examiner wants to expand the artifact listing to display all artifacts during a given timeframe, the examiner can use the same query above and remove the section of the statement that displays the executable programs ((File_Description) Like "*.exe*").

4.1.5 Mapping All Processes by Process ID

If the examiner identifies a rogue process during the investigation a process mapping can determine what processes came from the user and what processes came from other processes. When a user logs on, the Explorer.exe process begins and a process identification (PID) number is assigned. If the user then launches Notepad this process is assigned a PID and associated with its parent PID, Explorer.exe's PID. When a process begins or ends a Windows event is written to the audit logs with the PID of the process, the parent PID number, and the fully qualified path of the process (UltimateWindowsSecurity.com). The Event ID 592 is logged when a process begins and Event ID 593 is logged when a process exits (UltimateWindowsSecurity.com). The examiner can query for these audit logs so a mapping of all processes can determine what processes the user did and did not launch. If a questionable process is not linked directly back to a user's Explorer.exe process then this process requires additional investigation. This allows the examiner to determine the true origins of the processes that ran during the incident. Below is an example of this kind of query.

```
SELECT Date, Time, source, File_Description
FROM Log2Timeline
WHERE (source = "Event Log") AND ((File_Description Like "*592*") Or
    (File_Description Like "*593*"))
```

ORDER BY Date DESC, Time DESC;

4.2 Test Configuration

The test data used for this research was incidents that were created with specific activities. The script used to create these incidents is located at Appendix C. This allowed the research to determine what data was collected by the Log2timeline and how effective the tool was in collecting various artifacts. The test operating systems were Windows XP and Windows 7. Both systems provided various temporal artifacts with Windows XP providing the most artifacts. The Windows auditing function was turned on to provide additional artifacts, because the current standard for a system to operate on an Air Force network is to properly configure the system according to the Security Technical Information Guide (STIG). The STIG category II rules require auditing for logon, logoff, process execution, and user account management (<http://iase.disa.mil/stigs/>). According to the Windows 7 STIG rule SV-35985r1, turning on these audit logs “can help identify configuration errors, troubleshoot service disruptions, and analyze compromises that have occurred as well as detecting attacks”.

Once the incident was created on the system, it was booted using the DEFT forensic live CD where an image of the hard drive was captured to an external hard drive using the Guymager tool. These image files were copied to the analysis system where it was mounted using the Mount Image Pro tool. Using an Administrator command shell, the Log2timeline script was executed against the mounted image to capture the artifacts into a CSV file. The error log and output file was saved to a local folder on the analysis system.

The Log2timeline output was then imported into a Microsoft Access database where queries were executed against the data. Four fields (short, desc, filename, and notes) were converted from a Text data structure to a Memo data structure to ensure no loss of artifact data. The queries started off general like listing all user's logon and logoff times and when executable programs began and ended. This listing gives the examiner a repeatable starting point with any incident. Looking at the results, more complex queries are inputted narrowing down to a specific timeframe based on the investigation of the first query. Once the super timeline is narrowed down to the incident then artifacts can be correlated to determine what happened. When multiple different artifacts like the user assist registry key, last accessed time of the file, and audit log process begin time all correlate to the same time then this process is confirmed for a specific timeframe.

No one artifact will be the sole item to determine what activity occurred during the incident because it takes many artifacts, both normal and anomalous artifacts, to reconstruct the activities in any incident. The examiner can create query statements to quickly generate concise super timelines of artifacts to qualitatively evaluate the improvement in interpretability.

4.3 Database Design

The data within the database is designed as one table with many columns. All the data was imported into Microsoft Access 2007 using the Import function under the External Data tab on the ribbon. Clicking the Text File button in the Import group on the ribbon allowed the importing of the Log2timeline.csv file. Four field's data types were changed from Text to Memo when imported: short, desc, filename, and notes. Without this

change, file truncation errors were produced by Access, which lead to the loss of artifacts. The following figure details the field names and data type properties of the Log2timeline comma separated values file after it was imported into Access.

Field Name	Data Type
ID	AutoNumber
date	Date/Time
time	Date/Time
timezone	Text
MACB	Text
source	Text
sourcetype	Text
type	Text
user	Text
host	Text
short	Memo
desc	Memo
version	Number
filename	Memo
inode	Number
notes	Memo
format	Text
extra	Text

Figure 16 - MS Access' Table Properties of Log2timeline CSV File.

When the 116 megabyte Log2timeline file was imported into Access, the table resulted in over 303,000 rows of entries. Another Log2timeline file that was analyzed in Access was 354 megabytes, which produced over 784,000 rows. Both of these were controlled environments where the user did not perform many actions before the incident occurred and the artifacts were collected. Additionally, with the ever growing size of hard drives, the number of artifacts to collect will grow too. Keep in mind, on an actual incident response the hard drive will be much larger than the 10 gigabyte and 15 gigabyte (respectively) hard drives from these two test cases. The biggest factor the examiner will fight during a super timeline collection and reconstruction of an incident is the amount of time the user operated on the system since the computer was turned on the first time and

not necessarily the size of the hard drive. The more operations a user performs on a system, the more artifacts produced and the larger the log files become. Log file sizes may be set to a larger cap before erasing content based on the total size of the hard drive so larger hard drives will likely mean larger log files, temporary directories, and page files. Ultimately, this means user activity determines the number of resulting artifacts collected.

4.4 Query Searches

The information about each incident was known so specially formulated queries were created to search for artifacts of interest. Each incident is detailed out and the reusable queries are identified.

4.4.1 User Program Activity

To begin the SQL searching, a listing of the users and what each user did on the system is a great way to begin the search for information about an incident. This allows the examiner to take a quick look over the user's activities for anything obvious. The following query was run against the database.

```
SELECT Log2timeline.[date], Log2timeline.[time], Log2timeline.[source],
       Log2timeline.[sourcetype], Log2timeline.[type], Log2timeline.[user],
       Log2timeline.[short], Log2timeline.[desc], Log2timeline.[filename],
       Log2timeline.[notes], Log2timeline.[format], Log2timeline.[extra]
FROM Log2timeline
WHERE ((Not (Log2timeline.[user])="-"))
ORDER BY Log2timeline.[date] DESC, Log2timeline.[time] DESC;
```

This and all query produced the listing in order by date and time in descending order. Each item listed produced the user's name (`Log2timeline.[user]`) associated with the activity. Using the pivot filter option at the top of the *Type* column within Access further filters the results. When *Time of Launch* is the only option chosen from the pivot filter, the results displayed what programs each user executed. Log2timeline extracts *Time of Launch* artifacts from the NTUSER.DAT file under each user's profile.

7/27/2011	4:30:39 PM	Time of Launch	peacock	[[X86] System32] %windir%/system32/cmd.exe	[[X86] System32] %windir%/system32/cmd.exe [Count: 4] nr. of times app had focus: 10 and duration of focus: 440392ms
7/27/2011	4:29:09 PM	Time of Launch	peacock	E:/wbstego/wbStego4.3open.exe	E:/wbstego/wbStego4.3open.exe [Count: 2] nr. of times app had focus: 6 and duration of focus: 20186ms
7/27/2011	4:19:14 PM	Time of Launch	peacock	[User Pinned] %APPDATA%/Microsoft/Internet Explorer/Quick Launch/User Pinned/TaskBar/Internet Explorer.Ink	[User Pinned] %APPDATA%/Microsoft/Internet Explorer/Quick Launch/User Pinned/TaskBar/Internet Explorer.Ink [Count: 4] nr. of times app had focus: 0 and duration of focus: 4ms
7/27/2011	4:19:14 PM	Time of Launch	peacock	Microsoft.InternetExplorer.Default	Microsoft.InternetExplorer.Default [Count: 4] nr. of times app had focus: 21 and duration of focus: 1483303ms
7/27/2011	2:53:57 PM	Time of Launch	mustard	[Programs] %ALLUSERSPROFILE%/Microsoft/Windows/Start Menu/Programs/Camouflage/Camouflage Settings.Ink	[Programs] %ALLUSERSPROFILE%/Microsoft/Windows/Start Menu/Programs/Camouflage/Camouflage Settings.Ink [Count: 1] nr. of times app had focus: 0 and duration of focus: 1ms
7/27/2011	2:53:57 PM	Time of Launch	mustard	[Program Files] [X86] %ProgramFiles% (%SystemDrive%/Program Files)/Camouflage/Camouflage.exe	[Program Files] [X86] %ProgramFiles% (%SystemDrive%/Program Files)/Camouflage/Camouflage.exe [Count: 1] nr. of times app had focus: 2 and duration of focus: 88188ms
7/27/2011	2:53:19 PM	Time of Launch	mustard	E:/camo/Setup.exe	E:/camo/Setup.exe [Count: 1] nr. of times app had focus: 0 and duration of focus: 0ms
7/27/2011	2:41:45 PM	Time of Launch	white	[User Pinned] %APPDATA%/Microsoft/Internet Explorer/Quick Launch/User Pinned/TaskBar/Windows Explorer.Ink	[User Pinned] %APPDATA%/Microsoft/Internet Explorer/Quick Launch/User Pinned/TaskBar/Windows Explorer.Ink [Count: 1]

Figure 17 - Steganography Programs Found.

In the above figure, the steganography programs are highlighted. Additionally, the count for each program is identified, which is the number of times the program executed, and the amount of time in milliseconds the program ran. For example, the Wbstego program ran for 20,186 milliseconds. This means the program started at 4:19:14 pm and ran for just over 20 seconds before terminating. The Log2timeline also determined that Peacock was logged into the system at the time when the Wbstego program ran.

7/27/201 1	5:03:49 PM Time of Launch	scarlet	[[X86] System32] %windir%/system32/cmd.exe	[[X86] System32] %windir%/system32/cmd.exe [Count: 1] nr. of times app had focus: 0 and duration of focus: 6802ms
7/27/201 1	5:01:30 PM Time of Launch	scarlet	E:/actualspy.exe	E:/actualspy.exe [Count: 1] nr. of times app had focus: 0 and duration of focus: 0ms
7/27/201 1	5:00:26 PM Time of Launch	mustard	[[X86] System32] %windir%/system32/control.exe	[[X86] System32] %windir%/system32/control.exe [Count: 1] nr. of times app had focus: 0 and duration of focus: 0ms
7/27/201 1	4:54:54 PM Time of Launch	mustard	E:/wbstego/wbStego4.3open.exe	E:/wbstego/wbStego4.3open.exe [Count: 2] nr. of times app had focus: 5 and duration of focus: 16474ms

Figure 18 - More Programs Found.

In the above figure, also captured from the same SQL query, two additional pieces of evidence were identified with the user Scarlet using a key logging program and Mustard using the Wbstego program. It can be seen, from the above two figure, how useful this SQL query has become, which identified what program was run by what user.

4.4.2 User Web History

The next query ran extracted the web history by user. This SQL statement used the following query to output all of the web history and all of the logon and logoff information. The users are identified by their logon and logoff events in the audit logs. The following shows the SQL statement used to list all the web history events with the logon and logoff times for each user.

```
SELECT Log2timeline.[date], Log2timeline.[time], Log2timeline.[source],
       Log2timeline.[sourcetype], Log2timeline.[type],
       Log2timeline.[user], Log2timeline.[host], Log2timeline.[short],
       Log2timeline.[desc], Log2timeline.[filename],
       Log2timeline.[notes], Log2timeline.[format],
       Log2timeline.[extra]
FROM Log2timeline
WHERE (((Log2timeline.[source])="webhist")) OR (((Log2timeline.[short])
       Like "*528*")) OR (((Log2timeline.[short]) Like "*538*"))
ORDER BY Log2timeline.[date] DESC, Log2timeline.[time] DESC;
```

If more specific results are needed because the incident involves gambling then the SQL statement is updated with the following SQL statement, which pares down the results to find the keyword gambling in the artifact.

```
SELECT Log2timeline.date, Log2timeline.time, Log2timeline.source,
       Log2timeline.sourcetype, Log2timeline.type, Log2timeline.user,
       Log2timeline.host, Log2timeline.short, Log2timeline.desc,
       Log2timeline.filename, Log2timeline.notes, Log2timeline.format,
       Log2timeline.extra
FROM Log2timeline
WHERE (((Log2timeline.source)="webhist") AND ((Log2timeline.short) Like
       "*gambling*")) OR (((Log2timeline.short) Like "*528*")) OR
       (((Log2timeline.short) Like "*538*"))
ORDER BY Log2timeline.date DESC, Log2timeline.time DESC;
```

The results from the above query showed gambling web history results with all the user's logon and logoff times. To narrow the results down to a specific user the SQL statement was further modified as follows.

```
SELECT Log2timeline.date, Log2timeline.time, Log2timeline.source,
       Log2timeline.sourcetype, Log2timeline.type, Log2timeline.user,
       Log2timeline.host, Log2timeline.short, Log2timeline.desc,
       Log2timeline.filename, Log2timeline.notes, Log2timeline.format,
       Log2timeline.extra
FROM Log2timeline
WHERE (((Log2timeline.source)="webhist") AND ((Log2timeline.short) Like
       "*gambling*")) OR (((Log2timeline.user) ="mustard") AND
       ((Log2timeline.short) Like "*528*")) OR
       (((Log2timeline.user)="mustard") AND ((Log2timeline.short) Like
       "*538*"))
ORDER BY Log2timeline.date DESC, Log2timeline.time DESC;
```

This query pinpointed the gambling web queries that user *mustard* typed into his web browser and displayed the logon and logoff time of the user. This can be a very effective way for an examiner to narrow down hundreds of megabytes of data from Log2timeline tool and achieve a specific result.

date	time	source	sourcetype	type	user	short
7/27/2011	1:43:08 PM	WEBHIST	Internet Explorer	Last Access	mustard	visited http://en.wikipedia.org/wiki/Gambling
7/27/2011	1:43:08 PM	WEBHIST	Internet Explorer	Last Access	mustard	visited http://en.wikipedia.org/wiki/Gambling
7/27/2011	1:43:03 PM	WEBHIST	Internet Explorer	Last Access	mustard	visited http://www.bing.com/search?q=gambling&src=IE-SearchBox&FORM=IE8SRC
7/27/2011	1:43:03 PM	WEBHIST	Internet Explorer	Last Access	mustard	visited http://www.bing.com/search?q=gambling&src=IE-SearchBox&FORM=IE8SRC
7/27/2011	9:43:08 AM	WEBHIST	Internet Explorer	Last Access/Las	mustard	visited http://en.wikipedia.org/wiki/Gambling
7/27/2011	9:43:08 AM	WEBHIST	Internet Explorer	Last Access/Las	mustard	visited http://en.wikipedia.org/wiki/Gambling
7/27/2011	9:43:06 AM	WEBHIST	Internet Explorer	Content saved	-	visited http://en.wikipedia.org/wiki/Gambling
7/27/2011	9:43:03 AM	WEBHIST	Internet Explorer	Last Access/Las	mustard	visited http://www.bing.com/search?q=gambling&src=IE-SearchBox&FORM=IE8SRC
7/27/2011	9:43:03 AM	WEBHIST	Internet Explorer	Last Access/Las	mustard	visited http://www.bing.com/search?q=gambling&src=IE-SearchBox&FORM=IE8SRC&format=rss
7/27/2011	9:43:03 AM	WEBHIST	Internet Explorer	Content saved	-	visited http://www.bing.com/search?q=gambling&src=IE-SearchBox&FORM=IE8SRC
7/27/2011	9:43:03 AM	WEBHIST	Internet Explorer	Last Access/Las	mustard	visited http://www.bing.com/search?q=gambling&src=IE-SearchBox&FORM=IE8SRC
7/27/2011	9:43:03 AM	WEBHIST	Internet Explorer	Last Access/Las	mustard	visited http://www.bing.com/search?q=gambling&src=IE-SearchBox&FORM=IE8SRC&format=rss
7/27/2011	8:56:45 AM	REG	NTUSER key	Last Written	mustard	AppEvents/Schemes/Apps/.Default/WindowsLogon/.Current

Figure 19 - SQL Results for Mustard Gambling.

As highlighted in the figure above, the user *mustard* was searching for gambling from the Bing.com search engine. The user also visited Wikipedia.com about gambling. This SQL query displays how the examiner can start off with a general query and by adding specific keywords associated with the incident, produce a concise super timeline of artifacts about a user's activities on the system. Additionally, it prevents the examiner from reading through hundreds of thousands of artifacts if it is known the incident involves gambling; thereby, saving the examiner many hours of time and effort.

4.4.3 Recent Document Accesses

Another area considered important for analysis is the recent documents for all the users. On a Windows system when a document is touched or media device is added by the user, a link file is created in the user's recent folder. A query for all recent documents can provide some useful information about the user's activities.

```

SELECT Log2timeline.date, Log2timeline.time, Log2timeline.source,
       Log2timeline.sourcetype, Log2timeline.type, Log2timeline.user,
       Log2timeline.filename, Log2timeline.short, Log2timeline.desc,
       Log2timeline.notes, Log2timeline.extra
FROM Log2timeline
WHERE (((Log2timeline.desc) Like "*recent*"))
ORDER BY Log2timeline.date DESC, Log2timeline.time DESC;

```

Using the above query, the following results were part of the output of the recent documents SQL statement.

date	time	source	sourcetype	type	user	short
7/27/2011	4:56:50 PM	LNK	Shortcut LNK	Access	-	C:/Program Files/Common Files/Adobe/Help/en_us/reader/X/using
7/27/2011	4:56:50 PM	LNK	Shortcut LNK	Modified	-	C:/Program Files/Common Files/Adobe/Help/en_us/reader/X/using
7/27/2011	4:56:50 PM	REG	RecentDocs ke	File opened	mustard	Recently opened file of extension: .htm - value: readme2.htm
7/27/2011	4:56:50 PM	FILE	NTFS \$MFT	\$FN [MAC.] tim	-	/Users/mustard/AppData/Roaming/Microsoft/Windows/Recent/using.Ink
7/27/2011	4:56:50 PM	FILE	NTFS \$MFT	\$SI [MAC.] tim	-	/Users/mustard/AppData/Roaming/Microsoft/Windows/Recent/using.Ink
7/27/2011	4:56:50 PM	FILE	NTFS \$MFT	\$FN [MACB] tir	-	/Users/mustard/AppData/Roaming/Microsoft/Windows/Recent/readme2.htm.Ink
7/27/2011	4:56:50 PM	FILE	NTFS \$MFT	\$SI [MACB] tim	-	/Users/mustard/AppData/Roaming/Microsoft/Windows/Recent/readme2.htm.Ink
7/27/2011	4:56:50 PM	FILE	NTFS \$MFT	\$SI [M.C.] time	-	/Users/mustard/AppData/Roaming/Microsoft/Windows/Recent/AutomaticDestinat
7/27/2011	4:55:34 PM	FILE	NTFS \$MFT	\$FN [MACB] tir	-	/Users/mustard/AppData/Roaming/Microsoft/Windows/Recent/readme.htm.Ink
7/27/2011	4:55:34 PM	FILE	NTFS \$MFT	\$SI [MACB] tim	-	/Users/mustard/AppData/Roaming/Microsoft/Windows/Recent/readme.htm.Ink
7/27/2011	4:55:11 PM	REG	RecentDocs ke	File opened	mustard	Recently opened file of extension: .txt - value: too late.txt
7/27/2011	4:55:11 PM	FILE	NTFS \$MFT	\$FN [MAC.] tir	-	/Users/mustard/AppData/Roaming/Microsoft/Windows/Recent/too late.txt.Ink
7/27/2011	4:55:11 PM	FILE	NTFS \$MFT	\$SI [MAC.] tim	-	/Users/mustard/AppData/Roaming/Microsoft/Windows/Recent/too late.txt.Ink
7/27/2011	4:55:11 PM	FILE	NTFS \$MFT	\$SI [MAC.] tim	-	/Users/mustard/AppData/Roaming/Microsoft/Windows/Recent/USB DISK (E).Ink
7/27/2011	4:55:11 PM	FILE	NTFS \$MFT	\$FN [MAC.] tir	-	/Users/mustard/AppData/Roaming/Microsoft/Windows/Recent/USB DISK (E).Ink
7/27/2011	4:54:07 PM	LNK	Shortcut LNK	Modified	-	C:/Program Files/Common Files/Adobe/Help/en_us/reader/X/using/readme.htm

Figure 20 - SQL Query for Recent Documents.

Highlighted above is a file called *too late.txt*, this file's original location was on a USB disk mounted to the drive letter E:\. Additionally, the user *mustard* can be attributed to opening this file for two reasons. First, the file was opened and logged by the operating system as opened by the user *mustard* and the master file table also had entries in the standard information (\$SI) and file name (\$FN) attributes where the file was located in the folder */user/mustard*. This information tells the examiner to search for a USB memory stick that contains the file *too late.txt*. The examiner can ascertain evidentiary information about the incident when searching through the recent document links in each user

account. The examiner should not use any one artifact to attribute activity to a user, but correlate the artifact to other temporal artifacts to confirm or deny attribution.

4.5 Windows 7 Incident

A Windows 7 system was the first test complete incident analyzed. The incident was a system where the user, Miss Scarlet, logged on at 11:36, inserted a USB device, ran a program that created a backdoor to the system, ran Internet Explorer to look at Google news, and then logged off. The hacker used the Netcat tool to access the system after the program was run from the USB device that created a backdoor so the hacker could create a file called *KillBoddy.txt*.

Since the timeframe is known, a query into what executables and text files were accessed during this timeframe is the starting point with the following query and results.

```
SELECT Log2Timeline.Date, Log2Timeline.Time, Log2Timeline.MACB,
       Log2Timeline.source, Log2Timeline.sourcetype, Log2Timeline.type,
       Log2Timeline.User, Log2Timeline.desc
FROM Log2Timeline
WHERE (((Log2Timeline.Date)=#4/26/2012#) AND
      ((Log2Timeline.Time)>#12/30/1899 11:35:0# And
      (Log2Timeline.Time)<#12/30/1899 11:49:0#) AND
      ((Log2Timeline.type)="4624" Or (Log2Timeline.type)="4647")) OR
      (((Log2Timeline.Date)=#4/26/2012#) AND
      ((Log2Timeline.Time)>#12/30/1899 11:35:0# And
      (Log2Timeline.Time)<#12/30/1899 11:49:0#) AND ((Log2Timeline.short)
      Like "*.exe*" Or (Log2Timeline.short) Like "*.txt*")) OR
      (((Log2Timeline.Date)=#4/26/2012#) AND
      ((Log2Timeline.Time)>#12/30/1899 11:35:0# And
      (Log2Timeline.Time)<#12/30/1899 11:49:0#) AND ((Log2Timeline.desc)
      Like "*.exe*" Or (Log2Timeline.desc) Like "*.txt*")) OR
      (((Log2Timeline.Date)=#4/26/2012#) AND
      ((Log2Timeline.Time)>#12/30/1899 11:35:0# And
      (Log2Timeline.Time)<#12/30/1899 11:49:0#) AND
      ((Log2Timeline.filename) Like "*.exe*" Or (Log2Timeline.filename)
      Like "*.txt*"))
ORDER BY Log2Timeline.Date DESC , Log2Timeline.Time DESC;
```

Date	Time	MACB	source	sourcetype	type	User	desc
4/26/2012	11:48:40	-	AuditLog	Application	1530	NT AUTHORITY\	Windows detected your registry file is still in use by other applications or services. The file will be unkl
4/26/2012	11:48:40	-	AuditLog	Security	4647	N/A	User initiated logoff: Subject: Security ID:S-1-5-21-2380117116-3663119347-1970249807-1001
4/26/2012	11:45:27	M.C.	FILE	NTFS \$MFT	SSI [M.C.] time	-	/Users/Miss Scarlet/Documents/KillBoddy.txt
4/26/2012	11:42:54	A.B	FILE	NTFS \$MFT	SSI [A.B] time	-	/Users/Miss Scarlet/Documents/KillBoddy.txt
4/26/2012	11:42:54	MACB	FILE	NTFS \$MFT	\$FN [MACB] time	-	/Users/Miss Scarlet/Documents/KillBoddy.txt
4/26/2012	11:39:15	MACB	FILE	NTFS \$MFT	\$FN [MACB] time	-	/Users/Miss Scarlet/AppData/Local/Temp/eW_98A9.tmp/services2000.exe
4/26/2012	11:39:15	MACB	FILE	NTFS \$MFT	SSI [MACB] time	-	/Users/Miss Scarlet/AppData/Local/Temp/eW_98A9.tmp/services2000.exe
4/26/2012	11:39:15	MACB	FILE	NTFS \$MFT	\$FN [MACB] time	-	/Users/Miss Scarlet/AppData/Local/Temp/eW_98A9.tmp/ProcessHacker.exe
4/26/2012	11:39:15	MACB	FILE	NTFS \$MFT	SSI [MACB] time	-	/Users/Miss Scarlet/AppData/Local/Temp/eW_98A9.tmp/ProcessHacker.exe
4/26/2012	11:39:14	MACB	REG	UserAssist key	Time of Launch	Miss Scarlet	E:/ProcessList.exe [Count: 1] nr. of times app had focus: 0 and duration of focus: 0ms
4/26/2012	11:39:14	MACB	REG	UserAssist key	Time of Launch	Miss Scarlet	E:/ProcessList.exe [Count: 1] nr. of times app had focus: 0 and duration of focus: 0ms
4/26/2012	11:36:31	-	AuditLog	Security	4624	N/A	Logon Type:2 New Logon: Security ID:S-1-5-21-2380117116-3663119347-1970249807-1001 A

Figure 21 – Query for Executable and Text Files during Miss Scarlet’s Logged on Time.

The results show, during Miss Scarlet’s logon from 11:36:31 – 11:48:40, three executable programs and one text file were accessed, ProcessList.exe, ProcessHacker.exe, Services2000.exe, and KillBoddy.txt. The search was expanded to display all evidence between the timeframe of 11:36:31 – 11:48:40.

```

SELECT Log2Timeline.date, Log2Timeline.time, Log2Timeline.MACB,
       Log2Timeline.source, Log2Timeline.sourcetype, Log2Timeline.type,
       Log2Timeline.user, Log2Timeline.desc, Log2Timeline.filename
FROM Log2Timeline
WHERE (((Log2Timeline.date)=#4/26/2012#) AND
       ((Log2Timeline.time)>#12/30/1899 11:36:0# And
       (Log2Timeline.time)<#12/30/1899 11:49:0#))
ORDER BY Log2Timeline.date DESC, Log2Timeline.time DESC;

```


when a connection was made, a command shell was execute (cmd.exe). The tool did not produce any artifacts about cmd.exe and the fact that it was executed at 11:40. When the timeline was queried for cmd.exe it showed an execution time of 09:33 on 26 April, two hours prior to Miss Scarlet logging in. The file in question, *KillBoddy.txt*, did not show an owner so there would be some suspicion based on this fact. When a user creates a file, the operating system adds the username to the owner file attribute. The Log2timeline tool showed no owner of the *KillBoddy.txt* file and using the `DIR /Q` command also confirmed no owner for the file. The other packed process, ProcessHacker.exe, is a freely available program that displays running processes very much like Task Manager for Windows. The program ProcessList.exe is the process that wrapped the two executable programs.

The Log2timeline does not parse Windows 7 artifacts well enough to produce good evidence about an incident. Additional tools are required to produce more artifacts to prove Miss Scarlet's innocence. A live capture of the volatile RAM, while the system is being exploited, would capture the Netcat and command shell programs running in memory. Additional event logging is needed to capture more of the incident's activities; these logged events would allow the examiner to determine more about the incident. This incident clearly shows why these audit logs are vital to ensuring all artifacts are available for incident reconstruction. In the next section, a Windows XP incident has the additional event logging turn on so it will be evident how important this event logging is for incident reconstruction.

4.6 Windows XP Incident

Using a similar incident on a Windows XP system, the results were quite contrasting to the above Windows 7 system. This Windows XP system was logged on by Miss Scarlet at 17:50. The user opened the notepad program, then the calculator program. After closing both, the user inserted a USB memory stick and ran the program ProcessList.exe, which spawned two programs, ProcessHacker.exe that the user saw running on the desktop, and Services2000.exe that the user didn't see running in the background. Like above, the Services2000.exe program is the Netcat program in listener mode waiting for the hacker to connect to the system. When the Netcat connection is established a command shell is passed to the hacker allowing the hacker access to Miss Scarlet's system. The user left the Process Hacker program running on the desktop and ran the solitaire program. The user played one game of solitaire then logged off the system at 18:21. This incident centers around a file called *HateLetter.txt*, which contains text that appears to be from Miss Scarlet to Mr. Boddy stating she's had enough of his antics and it going to kill him. Was the file created by Miss Scarlet?

Since this incident focuses on the user Miss Scarlet, the first query is to list what programs the user executed while logged in by using the following SQL statement. The results of the query shows any executable program executed (`like "*.exe"`) between 17:50:52 and 18:21:08 on 27 April 2012, and also displays all logon and logoff events (`like "*528*" and like "*538"`).

```
SELECT Log2Timeline.Date, Log2Timeline.Time, Log2Timeline.MACB,  
       Log2Timeline.source, Log2Timeline.sourcetype, Log2Timeline.type,
```

```

Log2Timeline.User, Log2Timeline.host, Log2Timeline.desc,

Log2Timeline.short, Log2Timeline.filename

FROM Log2Timeline

WHERE (((Log2Timeline.Date)=#4/27/2012#) AND

((Log2Timeline.Time)>#12/30/1899 17:50:52# And

(Log2Timeline.Time)<#12/30/1899 18:21:8#) AND

((Log2Timeline.sourcetype)="Event Log") AND ((Log2Timeline.short)

Like "*528*" Or (Log2Timeline.short) Like "*538*")) OR

(((Log2Timeline.Date)=#4/27/2012#) AND

((Log2Timeline.Time)>#12/30/1899 17:50:52# And

(Log2Timeline.Time)<#12/30/1899 18:21:8#) AND

((Log2Timeline.type)<>"Time generated/written") AND

((Log2Timeline.short) Like "*.exe*"))

ORDER BY Log2Timeline.Date DESC, Log2Timeline.Time DESC;

```

Date	Time	MACB	source	sourcetype	type	User	host	desc
4/27/2012	18:20:56	A..	FILE	NTFS SMFT	\$\$I [A..] time	-	-	/Program Files/VMware/VMware Tools/VMwareTray.exe
4/27/2012	18:16:22	MACB	EVT	Event Log	Time generated/written	S-1-5-21-139C VICTIM	-	Security/528:Success;MissScarlet - VICTIM - (0x0-0xABEE7) - 2 - Advapi - Negotiate - VICTIM --
4/27/2012	18:16:22	A..	FILE	NTFS SMFT	\$\$I [A..] time	-	-	/WINDOWS/system32/tourstart.exe
4/27/2012	18:16:22	A..	FILE	NTFS SMFT	\$\$I [A..] time	-	-	/WINDOWS/system32/spider.exe
4/27/2012	18:16:22	A..	FILE	NTFS SMFT	\$\$I [A..] time	-	-	/WINDOWS/system32/usmt/migwiz.exe
4/27/2012	18:16:22	A..	FILE	NTFS SMFT	\$\$I [A..] time	-	-	/WINDOWS/system32/Restore/rstrui.exe
4/27/2012	18:16:22	MACB	EVT	Event Log	Time generated/written	S-1-5-21-139C VICTIM	-	Security/538:Success;MissScarlet - VICTIM - (0x0-0xABCCF) - 2
4/27/2012	18:16:22	MACB	EVT	Event Log	Time generated/written	S-1-5-21-139C VICTIM	-	Security/528:Success;MissScarlet - VICTIM - (0x0-0xABCCF) - 2 - Advapi - Negotiate - VICTIM --
4/27/2012	18:16:22	MACB	EVT	Event Log	Time generated/written	S-1-5-21-139C VICTIM	-	Security/538:Success;MissScarlet - VICTIM - (0x0-0xABEE7) - 2
4/27/2012	18:16:22	A..	FILE	NTFS SMFT	\$\$I [A..] time	-	-	/WINDOWS/system32/mshearts.exe
4/27/2012	18:14:58	A..	FILE	NTFS SMFT	\$\$I [A..] time	-	-	/WINDOWS/system32/attrib.exe
4/27/2012	18:06:04	MACB	FILE	NTFS SMFT	\$\$I [MACB] time	-	-	/WINDOWS/Prefetch/CMD.EXE-087B4001.pf
4/27/2012	18:05:54	A..	FILE	NTFS SMFT	\$\$I [A..] time	-	-	/WINDOWS/system32/cmd.exe
4/27/2012	18:05:54	MACB	PRE	XP Prefetch	Last run	-	-	CMD.EXE-087B4001.pf - [CMD.EXE] was executed - run count [35]- full path: [C:/WINDOWS/SYST
4/27/2012	18:00:47	MACB	REG	UserAssist key	Time of Launch	MissScarlet	-	UEME_RUNPATHC:/WINDOWS/system32/sol.exe [Count: 1]
4/27/2012	18:00:47	A..	FILE	NTFS SMFT	\$\$I [A..] time	-	-	/WINDOWS/system32/sol.exe
4/27/2012	18:00:08	AC	FILE	NTFS SMFT	\$\$I [AC] time	-	-	/Documents and Settings/MissScarlet/Local Settings/Temp/eW_3.tmp/ProcessHacker.exe
4/27/2012	18:00:08	MACB	FILE	NTFS SMFT	\$\$FN [MACB] time	-	-	/Documents and Settings/MissScarlet/Local Settings/Temp/eW_3.tmp/ProcessHacker.exe
4/27/2012	18:00:08	MACB	FILE	NTFS SMFT	\$\$I [MACB] time	-	-	/Documents and Settings/MissScarlet/Local Settings/Temp/eW_3.tmp/services2000.exe
4/27/2012	18:00:07	M..B	FILE	NTFS SMFT	\$\$I [M..B] time	-	-	/Documents and Settings/MissScarlet/Local Settings/Temp/eW_3.tmp/ProcessHacker.exe
4/27/2012	18:00:07	MACB	FILE	NTFS SMFT	\$\$FN [MACB] time	-	-	/Documents and Settings/MissScarlet/Local Settings/Temp/eW_3.tmp/ProcessHacker.exe
4/27/2012	18:00:06	MACB	REG	UserAssist key	Time of Launch	MissScarlet	-	UEME_RUNPATHC:/ProcessList.exe [Count: 1]
4/27/2012	18:00:02	MACB	FILE	NTFS SMFT	\$\$I [MACB] time	-	-	/WINDOWS/Prefetch/VERCLSID.EXE-3667BD89.pf
4/27/2012	18:00:02	MACB	PRE	XP Prefetch	Last run	-	-	VERCLSID.EXE-3667BD89.pf - [VERCLSID.EXE] was executed - run count [63]- full path: [C:/WIND
4/27/2012	17:59:01	A..	FILE	NTFS SMFT	\$\$I [A..] time	-	-	/Program Files/Internet Explorer/ieplora.exe
4/27/2012	17:54:01	MACB	REG	UserAssist key	Time of Launch	MissScarlet	-	UEME_RUNPATHC:/WINDOWS/system32/calc.exe [Count: 1]
4/27/2012	17:53:08	MACB	REG	FileExts key	Extension Changed	MissScarlet	-	File extension [.txt] opened with [notepad.exe]
4/27/2012	17:53:08	MACB	REG	NTUSER key	File Opened	MissScarlet	-	Most recently opened file in Windows (in an "Open" dialog): C:/Documents and Settings/MissScarlet/
4/27/2012	17:52:36	MACB	FILE	NTFS SMFT	\$\$I [MACB] time	-	-	/WINDOWS/Prefetch/NOTEPAD.EXE-336351A9.pf
4/27/2012	17:52:25	MACB	REG	UserAssist key	Time of Launch	MissScarlet	-	UEME_RUNPATHC:/WINDOWS/system32/notepad.exe [Count: 1]
4/27/2012	17:52:25	MACB	PRE	XP Prefetch	Last run	-	-	NOTEPAD.EXE-336351A9.pf - [NOTEPAD.EXE] was executed - run count [9]- full path: [C:/WINDO
4/27/2012	17:51:20	MACB	FILE	NTFS SMFT	\$\$I [MACB] time	-	-	/WINDOWS/Prefetch/WMPRVSE.EXE-28F301A9.pf
4/27/2012	17:51:12	MACB	EVT	Event Log	Time generated/written	S-1-5-20	VICTIM	Security/528:Success;NETWORK SERVICE - NT AUTHORITY - (0x0-0x3E4) - 5 - Advapi - Negotia
4/27/2012	17:51:10	MACB	EVT	Event Log	Time generated/written	S-1-5-20	VICTIM	Security/528:Success;NETWORK SERVICE - NT AUTHORITY - (0x0-0x3E4) - 5 - Advapi - Negotia
4/27/2012	17:51:10	MACB	PRE	XP Prefetch	Last run	-	-	WMPRVSE.EXE-28F301A9.pf - [WMPRVSE.EXE] was executed - run count [23]- full path: [C:/WIN
4/27/2012	17:51:07	MACB	FILE	NTFS SMFT	\$\$I [MACB] time	-	-	/WINDOWS/Prefetch/VMWARETRAY.EXE-029F476F.pf
4/27/2012	17:51:05	MACB	FILE	NTFS SMFT	\$\$I [MACB] time	-	-	/WINDOWS/Prefetch/EXPLORER.EXE-082F38A9.pf
4/27/2012	17:51:05	MACB	FILE	NTFS SMFT	\$\$I [MACB] time	-	-	/WINDOWS/Prefetch/WGATRAY.EXE-0ED38BED.pf
4/27/2012	17:51:04	MACB	FILE	NTFS SMFT	\$\$I [MACB] time	-	-	/WINDOWS/Prefetch/USERINIT.EXE-30B18140.pf
4/27/2012	17:51:00	A..	FILE	NTFS SMFT	\$\$I [A..] time	-	-	/WINDOWS/system32/imap.exe
4/27/2012	17:50:55	MACB	PRE	XP Prefetch	Last run	-	-	EXPLORER.EXE-082F38A9.pf - [EXPLORER.EXE] was executed - run count [14]- full path: [C:/WIN
4/27/2012	17:50:55	MACB	PRE	XP Prefetch	Last run	-	-	VMWARETRAY.EXE-029F476F.pf - [VMWARETRAY.EXE] was executed - run count [15]- full path:
4/27/2012	17:50:55	MACB	PRE	XP Prefetch	Last run	-	-	WGATRAY.EXE-0ED38BED.pf - [WGATRAY.EXE] was executed - run count [8]- full path: [C:/WIND
4/27/2012	17:50:54	MACB	EVT	Event Log	Time generated/written	S-1-5-21-139C VICTIM	-	Security/528:Success;MissScarlet - VICTIM - (0x0-0x24DEF) - 2 - User32 - Negotiate - VICTIM --
4/27/2012	17:50:54	MACB	PRE	XP Prefetch	Last run	-	-	USERINIT.EXE-30B18140.pf - [USERINIT.EXE] was executed - run count [13]- full path: [C:/WINDO
4/27/2012	17:50:53	MACB	EVT	Event Log	Time generated/written	S-1-5-21-139C VICTIM	-	Security/538:Success;MissScarlet - VICTIM - (0x0-0x24BDB) - 2
4/27/2012	17:50:53	MACB	EVT	Event Log	Time generated/written	S-1-5-21-139C VICTIM	-	Security/528:Success;MissScarlet - VICTIM - (0x0-0x24BDB) - 2 - Advapi - Negotiate - VICTIM --

Figure 23 - Miss Scarlet's Activities (Logoff, Logon, & EXE programs).

In the above results, the highlighted section shows how a program, ProcessList.exe, was executed and two additional programs were executed all within 2 seconds (6:00:06 – 6:00:08). This is certainly suspicious behavior since three programs all ran so close together and only ProcessList.exe ran from the E:\ directory while the other two ran from a temporary directory. The program Verclsid.exe was ignored because this is an operating system process that validates other processes. The next step was looking at what processes ran when. The following SQL statement displayed all the processes that ran during Miss Scarlet's logon time.

```
SELECT Log2Timeline.Date, Log2Timeline.Time, Log2Timeline.sourcetype,
       Log2Timeline.desc, Log2Timeline.short, Log2Timeline.filename
FROM Log2Timeline
WHERE (((Log2Timeline.Date)=#4/27/2012#) AND
       ((Log2Timeline.Time)>#12/30/1899 17:50:52# And
       (Log2Timeline.Time)<#12/30/1899 18:21:8#) AND
       ((Log2Timeline.sourcetype)="Event Log") AND (((Log2Timeline.desc)
       Like "*592*" Or (Log2Timeline.desc) Like "*593*") And
       ((Log2Timeline.desc) Like "*1928*" Or (Log2Timeline.desc) Like
       "*908*" Or (Log2Timeline.desc) Like "*2004*" Or (Log2Timeline.desc)
       Like "*1436*" Or (Log2Timeline.desc) Like "*808*" Or
       (Log2Timeline.desc) Like "*1612*" Or (Log2Timeline.desc) Like
       "*1824*" Or (Log2Timeline.desc) Like "*220*" Or (Log2Timeline.desc)
       Like "*1992*")))
ORDER BY Log2Timeline.Date DESC, Log2Timeline.Time DESC;
```

Date	Time	sourcetype	desc
4/27/2012	18:21:06	Event Log	Security/593;Success;1928 - C:/WINDOWS/explorer.exe - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	18:21:06	Event Log	Security/593;Success;2004 - C:/Program Files/VMware/VMware Tools/VMwareTray.exe - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	18:20:32	Event Log	Security/593;Success;908 - /Device/Harddisk1/DP(1)0-0+5/ProcessList.exe - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	18:20:32	Event Log	Security/593;Success;1612 - C:/DOCUME-1/MISSSC-1/LOCALS-1/Temp/eW_3.tmp/ProcessHacker.exe - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	18:20:18	Event Log	Security/593;Success;1992 - C:/WINDOWS/system32/sol.exe - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	18:16:01	Event Log	Security/593;Success;220 - C:/WINDOWS/system32/cmd.exe - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	18:16:01	Event Log	Security/593;Success;1824 - C:/DOCUME-1/MISSSC-1/LOCALS-1/Temp/eW_3.tmp/services2000.exe - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	18:14:58	Event Log	Security/592;Success;1656 - C:/WINDOWS/system32/attrib.exe - 220 - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	18:05:54	Event Log	Security/592;Success;220 - C:/WINDOWS/system32/cmd.exe - 1824 - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	18:00:47	Event Log	Security/592;Success;1992 - C:/WINDOWS/system32/sol.exe - 1928 - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	18:00:08	Event Log	Security/592;Success;1612 - C:/DOCUME-1/MISSSC-1/LOCALS-1/Temp/eW_3.tmp/ProcessHacker.exe - 908 - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	18:00:08	Event Log	Security/592;Success;1824 - C:/DOCUME-1/MISSSC-1/LOCALS-1/Temp/eW_3.tmp/services2000.exe - 908 - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	18:00:06	Event Log	Security/592;Success;908 - /Device/Harddisk1/DP(1)0-0+5/ProcessList.exe - 1928 - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	18:00:02	Event Log	Security/592;Success;772 - C:/WINDOWS/system32/verclsid.exe - 1928 - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	17:54:22	Event Log	Security/593;Success;808 - C:/WINDOWS/system32/calc.exe - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	17:54:01	Event Log	Security/592;Success;808 - C:/WINDOWS/system32/calc.exe - 1928 - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	17:53:11	Event Log	Security/593;Success;1436 - C:/WINDOWS/system32/notepad.exe - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	17:53:08	Event Log	Security/592;Success;192 - C:/WINDOWS/system32/verclsid.exe - 1928 - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	17:53:08	Event Log	Security/592;Success;1568 - C:/WINDOWS/system32/verclsid.exe - 1928 - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	17:52:25	Event Log	Security/592;Success;1436 - C:/WINDOWS/system32/notepad.exe - 1928 - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	17:50:55	Event Log	Security/592;Success;2012 - C:/Program Files/VMware/VMware Tools/vmtoolsd.exe - 1928 - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	17:50:55	Event Log	Security/592;Success;2004 - C:/Program Files/VMware/VMware Tools/VMwareTray.exe - 1928 - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	17:50:55	Event Log	Security/592;Success;580 - C:/WINDOWS/system32/verclsid.exe - 1928 - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	17:50:55	Event Log	Security/592;Success;1928 - C:/WINDOWS/explorer.exe - 332 - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	17:50:55	Event Log	Security/592;Success;896 - C:/WINDOWS/system32/verclsid.exe - 1928 - MissScarlet - VICTIM - (0x0-0x24DEF)

Figure 24 - Query for Event ID 592 and 593, Starting and Ending of Processes.

The above results are read from the bottom up since the bottom is the oldest result. The highlighted row is where Miss Scarlet's Windows environment was created with the process Explorer.exe. The number 1928 in this row is the process' PID, and the number 332 is the parent PID that Miss Scarlet's Explorer.exe runs under. Looking at the three processes in question, the blue highlighted area, ProcessList.exe (PID 908) ran from Miss Scarlet since the parent process is 1928. Both of the questionable programs, ProcessHacker.exe (PID 1612) and Services2000.exe (PID 1824) are a child process of ProcessList.exe because their parent PID is 908, which is the PID assigned to ProcessList.exe. Even more suspicion is raised when the cmd.exe file is examined. Highlighted by the red arrows in Figure 24, the cmd.exe file began at 18:05:54 creating the Event ID 592, and the process ended when the Event ID 593 was logged at 18:16:01. The cmd.exe file has a parent PID of 1824, which means this command shell was spawned from the Services2000.exe process. All files and events during these 10 minutes of time the command shell ran are now suspect. Figure 25 shows a graphical

representation of the processes from the above figure, the start and end time for each process, its PID, and parent PID. A blue line represents the time a processes ran. The gold line represents a child process that points to its parent process. The red lines are the times the suspicious processes ran.

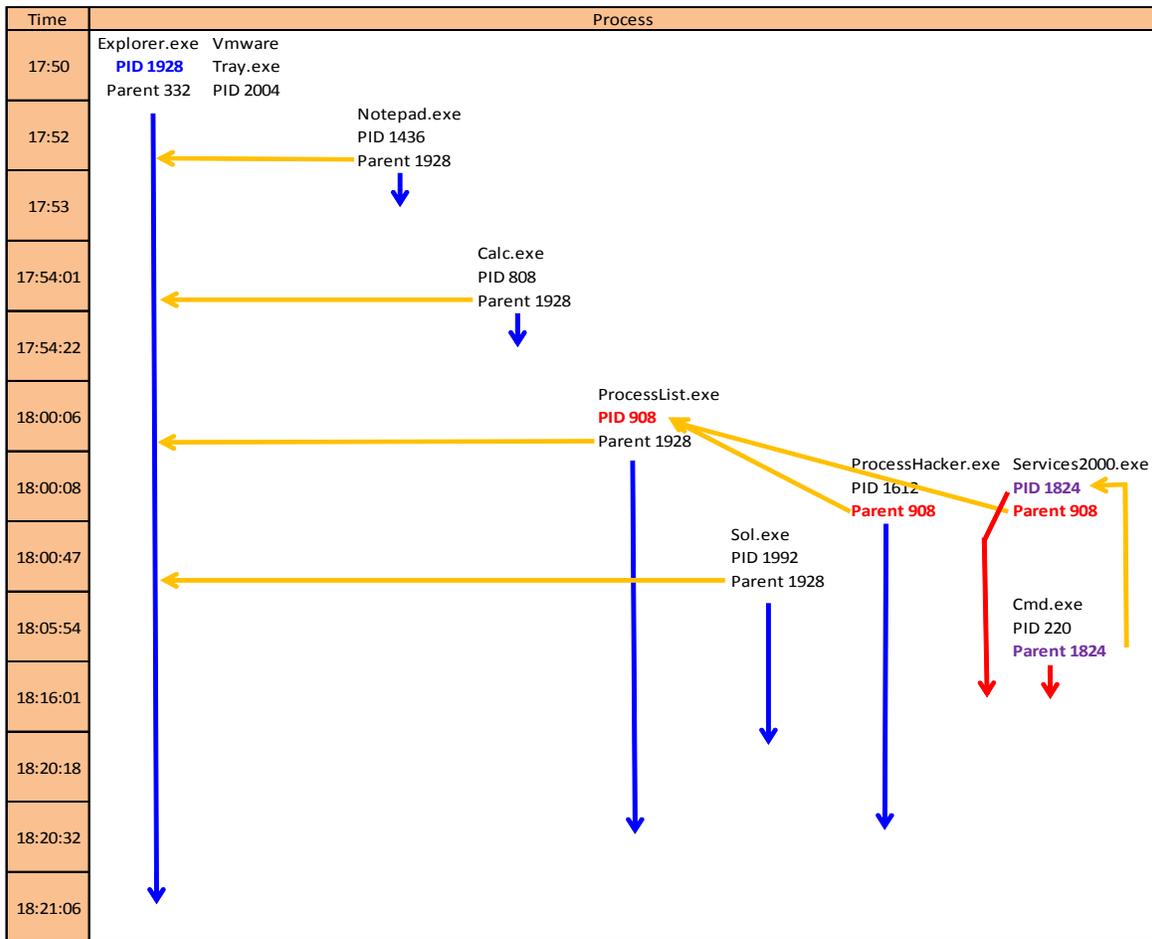


Figure 25 - Processes by Time, its PID, and Parent PID during Miss Scarlet's Logon Time.

Now it's very easy to see that cmd.exe came from Services2000.exe, which came from ProcessList.exe that came from Explorer.exe.

If the scope of time is narrowed to when ProcessList.exe began, 18:00:06, through the time the cmd.exe process ended, 18:16:01, and all the artifacts are listed then the

examiner can see if the *HateLetter.txt* file falls within this window of suspicion. The following SQL statement was used with results to follow the query string.

```
SELECT Log2Timeline.Date, Log2Timeline.Time, Log2Timeline.MACB,
       Log2Timeline.source, Log2Timeline.sourcetype, Log2Timeline.type,
       Log2Timeline.short, Log2Timeline.desc, Log2Timeline.filename
FROM Log2Timeline
WHERE ((Log2Timeline.Date)=#4/27/2012#) AND
      ((Log2Timeline.Time)>=#12/30/1899 18:0:6# And
      (Log2Timeline.Time)<=#12/30/1899 18:16:1#))
ORDER BY Log2Timeline.Date DESC, Log2Timeline.Time DESC;
```

Date	Time	MACB	source	sourcetype	type	short
4/27/2012	18:16:01	MACB	EVT	Event Log	Time generated/writ	Security/593;Success;220 - C:/WINDOWS/system32/cmd.exe - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	18:16:01	MACB	EVT	Event Log	Time generated/writ	Security/593;Success;1824 - C:/DOCUME-1/MISSSC-1/LOCALS-1/Temp/eW_3.tmp/services2000.exe - MissScarlet
4/27/2012	18:15:41	A..	FILE	NTFS \$MFT	\$\$I [A..] time	/WINDOWS/system32/logon.scr
4/27/2012	18:15:41	MACB	EVT	Event Log	Time generated/writ	Security/600;Success;628 - C:/WINDOWS/system32/winlogon.exe - VICTIMS - WORKGROUP - (0x0-0x3E7) - 1780 -
4/27/2012	18:15:41	MACB	EVT	Event Log	Time generated/writ	Security/592;Success;1780 - C:/WINDOWS/system32/logon.scr - 628 - VICTIMS - WORKGROUP - (0x0-0x3E7)
4/27/2012	18:14:58	A..	FILE	NTFS \$MFT	\$\$I [A..] time	/WINDOWS/system32/ulib.dll
4/27/2012	18:14:58	A..	FILE	NTFS \$MFT	\$\$I [A..] time	/WINDOWS/system32/attrib.exe
4/27/2012	18:14:58	MACB	EVT	Event Log	Time generated/writ	Security/592;Success;1656 - C:/WINDOWS/system32/attrib.exe - 220 - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	18:14:58	MACB	EVT	Event Log	Time generated/writ	Security/593;Success;1656 - C:/WINDOWS/system32/attrib.exe - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	18:13:30	A..	FILE	NTFS \$MFT	\$\$I [A..] time	/RECYCLER/S-1-5-21-1390067357-1078145449-839522115-1005
4/27/2012	18:13:30	A..	FILE	NTFS \$MFT	\$\$I [A..] time	/RECYCLER
4/27/2012	18:12:02	MAC.	FILE	NTFS \$MFT	\$\$I [MAC.] time	/Documents and Settings/MissScarlet/My Documents
4/27/2012	18:11:00	MA..	File	Dir		\Documents and Settings\MissScarlet\My Documents\HateLetter.txt
4/27/2012	18:09:00	.C.	File	Dir		\Documents and Settings\MissScarlet\My Documents\HateLetter.txt
4/27/2012	18:06:04	MAC.	FILE	NTFS \$MFT	\$\$I [MAC.] time	/WINDOWS/Prefetch/CMD.EXE-087B4001.pf
4/27/2012	18:05:54	A..	FILE	NTFS \$MFT	\$\$I [A..] time	/WINDOWS/system32/cmd.exe
4/27/2012	18:05:54	MACB	PRE	XP Prefetch	Last run	CMD.EXE-087B4001.pf: CMD.EXE was executed
4/27/2012	18:05:54	MACB	EVT	Event Log	Time generated/wri	Security/592;Success;220 - C:/WINDOWS/system32/cmd.exe - 1824 - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	18:00:47	A..	FILE	NTFS \$MFT	\$\$I [A..] time	/WINDOWS/system32/sol.exe
4/27/2012	18:00:47	MACB	REG	NTUSER key	Last Written	Software/Microsoft/Solitaire
4/27/2012	18:00:47	MACB	EVT	Event Log	Time generated/writ	Security/592;Success;1992 - C:/WINDOWS/system32/sol.exe - 1928 - MissScarlet - VICTIM - (0x0-0x24DEF)
4/27/2012	18:00:47	MACB	REG	NTUSER key	Last Written	Software/Microsoft
4/27/2012	18:00:47	MACB	REG	UserAssist key	Time of Launch	UEME_RUNPIDL:%csidl2%\Games/Solitaire.Ink
4/27/2012	18:00:47	MACB	REG	NTUSER key	Last Written	Software/Microsoft/Windows/ShellNoRoam/MUICache
4/27/2012	18:00:47	MACB	REG	UserAssist key	Time of Launch	UEME_RUNPATH
4/27/2012	18:00:47	A..	FILE	NTFS \$MFT	\$\$I [A..] time	/WINDOWS/system32/cards.dll
4/27/2012	18:00:47	MACB	REG	UserAssist key	Time of Launch	UEME_RUNPIDL:%csidl2%\Games
4/27/2012	18:00:47	MACB	REG	UserAssist ke	Time of Launch	UEME_RUNPATH:C:/WINDOWS/system32/sol.exe
4/27/2012	18:00:47	MACB	REG	UserAssist key	Time of Launch	UEME_RUNPIDL
4/27/2012	18:00:42	MACB	REG	NTUSER key	Last Written	Software/Microsoft/Windows/CurrentVersion/Explorer/MenuOrder/StartMenu2/Programs
4/27/2012	18:00:08	A..	FILE	NTFS \$MFT	\$\$I [A..] time	/Documents and Settings/MissScarlet/Local Settings/Temp
4/27/2012	18:00:08	MACB	FILE	NTFS \$MFT	\$\$I [MACB] time	/Documents and Settings/MissScarlet/Local Settings/Temp/eW_3.tmp/services2000.exe
4/27/2012	18:00:08	MAC.	FILE	NTFS \$MFT	\$\$I [MAC.] time	/Documents and Settings/MissScarlet/Local Settings/Temp/eW_3.tmp
4/27/2012	18:00:08	.AC.	FILE	NTFS \$MFT	\$\$I [.AC.] time	/Documents and Settings/MissScarlet/Local Settings/Temp/eW_3.tmp/ProcessHacker.exe
4/27/2012	18:00:08	MACB	FILE	NTFS \$MFT	\$FN [MACB] time	/Documents and Settings/MissScarlet/Local Settings/Temp/eW_3.tmp/services2000.exe
4/27/2012	18:00:08	A..	FILE	NTFS \$MFT	\$\$I [A..] time	/WINDOWS/system32/drivers/etc/services
4/27/2012	18:00:08	MACB	EVT	Event Log	Time generated/writ	Security/592;Success;1612 - C:/DOCUME-1/MISSSC-1/LOCALS-1/Temp/eW_3.tmp/ProcessHacker.exe - 908 - Mis
4/27/2012	18:00:08	MACB	EVT	Event Log	Time generated/writ	Security/592;Success;1824 - C:/DOCUME-1/MISSSC-1/LOCALS-1/Temp/eW_3.tmp/services2000.exe - 908 - Miss
4/27/2012	18:00:07	M.B	FILE	NTFS \$MFT	\$\$I [M.B] time	/Documents and Settings/MissScarlet/Local Settings/Temp/eW_3.tmp/ProcessHacker.exe
4/27/2012	18:00:07	MACB	FILE	NTFS \$MFT	\$FN [MACB] time	/Documents and Settings/MissScarlet/Local Settings/Temp/eW_3.tmp/ProcessHacker.exe
4/27/2012	18:00:07	..B	FILE	NTFS \$MFT	\$\$I [..B] time	/Documents and Settings/MissScarlet/Local Settings/Temp/eW_3.tmp
4/27/2012	18:00:07	MACB	FILE	NTFS \$MFT	\$FN [MACB] time	/Documents and Settings/MissScarlet/Local Settings/Temp/eW_3.tmp
4/27/2012	18:00:07	A..	FILE	NTFS \$MFT	\$\$I [A..] time	/WINDOWS/system32/crtldll.dll
4/27/2012	18:00:07	M.C.	FILE	NTFS \$MFT	\$\$I [M.C.] time	/Documents and Settings/MissScarlet/Local Settings/Temp
4/27/2012	18:00:06	MACB	REG	UserAssist ke	Time of Launch	UEME_RUNPATH:E:/ProcessList.exe
4/27/2012	18:00:06	MACB	EVT	Event Log	Time generated/writ	Security/592;Success;908 - /Device/Harddisk1/DP(1)0-045/ProcessList.exe - 1928 - MissScarlet - VICTIM - (0x0-0x2

Figure 26 - Query Results of Activities around ProcessList.exe and cmd.exe processes.

Reading the results from the bottom, the blue arrow points to where the ProcessList.exe process began executing. The second blue arrow shows when Services2000.exe began executing. The red arrow points where the cmd.exe process began executing. The orange

arrow points to the key file called *HateLetter.txt*. This file was created at 18:09 and then modified again at 18:11. Since the file in question, *HateLetter.txt*, was created and modified during the window when the cmd.exe process was active, and the file has no owner attributes, then this file was likely created by the cmd.exe process and not Miss Scarlet. Additionally, Miss Scarlet did not run a command shell nor did she run notepad to create the *HateLetter.txt* file during this timeframe further creating suspicion of the cmd.exe process.

Highlighted in blue in the figure above are User Assist keys, which are located in the registry (`HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist`). This part of the registry logs when the user accesses objects. As seen in the above figure, the user ran two programs, the ProcessList program and the Solitaire program. The red highlighted rows are the start of the suspicious cmd.exe program. There is no User Assist entry meaning the user, Miss Scarlet, did not run the cmd.exe program. All three of these programs equally have an event log, “success/592” entry that denotes when the program began running, according to the Windows event logs. Looking closer at the successful event log entry will indicate the parent process. The ProcessList.exe logged 1928 as the parent process so this process came from the root process of Miss Scarlet, explorer.exe. The timeframe when ProcessList.exe ran was within the logon and logoff times of Miss Scarlet. Using multiple artifacts like User Assist, Event Logs, File MAC times, and process IDs can determine if the user actually ran a process or not. In this case, Miss Scarlet clearly ran Solitaire and ProcessList but not cmd.exe.

The Log2timeline tool worked well when parsing Windows XP artifacts. As seen above, it can easily correlate many different artifacts into a super timeline. The examiner can narrow down this super timeline with general and specialized queries that reduce the workload from analyzing 300,000 lines of artifacts down to looking at less than 100 lines of artifacts making the examiner's job of determining the user's activities during the incident much easier.

4.7 Summary

There are many different ways to create simple and complex queries based on evidence collected at the incident. The above queries demonstrates how evidence can be quickly discovered even using the simplest of queries and progressing to a more complex yet granular look at the artifacts. As shown above, using these queries when an incident occurs allows the examiner to narrow down hundreds of thousands of artifacts to roughly one hundred artifacts or less. This allows the examiner to eliminate artifacts not relevant to the incident and concentrate on the artifacts of interest. The super timeline demonstrated how various artifacts like audit logs, master file table entries, and last access time for a file can quickly connect evidentiary artifacts to understand what happened by whom during the incident. Additionally, these concise super timelines allowed the examiner to reduce the workload and time required to determine the activities during the incident since the number of artifacts required for interpretation was reduced from over 303,000 to less than one hundred.

V. Conclusion and Future Work

Computers and digital devices are proliferating in an exponential manner. Some of these devices are used to commit crimes, some devices are used to detect crimes, and some devices are used to store the evidence of crimes. Digital forensic devices help detect crimes and store the evidence. Digital forensics is often reactionary in nature to an incident, but it is a vital facet to agile network defense.

This paper developed queries that use the output from Log2timeline tool to produce super timelines of a Windows operating system to provide an overview of the events that occurred before, during, and after an incident of interest. The research goal is to identify reusable queries that extract concise timeline artifacts that allow the examiner to decisively determine the activities of computer system incident. This research goal is broken down into two requirements, temporal artifact collection and timeline artifact analysis. Chapter 3 focused on the collection of temporal artifacts, which is the first requirement of the research goal. Then Chapter 4 discussed the second requirement of timeline artifact analysis.

One of the developed queries for a Microsoft Access database list all user's logon and logoff times with all executable program's begin and end times. This permits the examiner to break the complete super timeline into chunks making the hundreds of thousands of line of artifacts much easier to consume and dissect. Then narrowing the incident down by a timeframe allowed the examiner to develop a follow-on query that used this timeframe to find more specifics about the incident. Another reusable query was the listing of all the process' begin and end times with its process identification number

and parent process identification number, which allowed the examiner to understand which child process came from what parent process. Each subsequent and more detailed query would narrow down the hundreds of thousands of artifacts to one hundred artifacts or less allowing the examiner to minimize the time used to analyze the incident and focus the analysis on the artifacts of interest. Using the queries discussed in this research will shorten the time required for analysis as well as narrow the sheer number of artifacts to sort out.

5.1 Future Work

There are some additional areas of research to improve the ability to create more thorough super timelines. Volume shadow copy, emails, chat logs, pagefile, hibernation file, alternate data streams, and RAM are not currently captured by the Log2timeline tool so research in these areas can produce more meaningful results. Additionally, this research showed the limited capability Log2timeline has with the Windows 7 operating system. Expansion of these research areas will improve evidence collection and super timelines.

According to Harlan Carvey's blog on *More VSCs*, he discusses how files like NTUSER.DAT can be extracted from volume shadow copies so the results from the extracted DAT file can be compared to the current DAT file for differences. If the extracted NTUSER.DAT file contains information not found in the current DAT file but predates the current date then this data was likely deleted from the system sometime between the creation of the volume shadow copy and the current time of collection. This can allow the examiner to fill-in more of the timeline.

One other area the Log2timeline tool did not parse was email. The Microsoft Outlook program stores each user's emails in an identity folder based on the user's profile. Emails can be very important artifacts for evidence collection. For the average home user this may not result in too much data as many users are reading their emails online, but company systems usually use an email client like Outlook, which means the user's emails are usually being stored on the system. This can provide an additional resource for filling in the "gaps" of the super timeline.

Another area that can provide additional benefit to the super timeline is chat history. With more users conversing over chat type applications, more evidence can be extracted from chat history logs. The Log2timeline tool does not parse chat history information so programming a module to capture history from an application like Google Chat would be an effective way to fill in the super timeline even more.

One last area to pursue would be creating modules for Log2timeline. As seen in this research, Log2timeline failed to produce a good conclusion from an incident on a Windows 7 system but produced excellent results from a Windows XP system. A future research area would be creating a parse module to incorporate into the Log2timeline tool framework. The Windows 7 prefetch data or more file attributes would be excellent additions for Log2timeline.

Digital forensics deals with the collection and assembly of temporal artifacts. When many of these different types of artifacts are assembled into one timeline, a super timeline of artifacts is produced. The super timeline cuts the amount of time an examiner needs to analyze the artifacts to determine what pieces of evidence are most crucial.

There are many different tools capable of collecting artifacts from a system. Each tool has its own procedure to collect the artifacts. To create a timeline from multiple tools can be an arduous task since each tool outputs differently and normalizing the data can be too labor-intensive. Instead of using many different tools to collect the various artifacts, the Perl script Log2timeline tool was used, which collected many different types of artifacts into one super timeline. All the artifacts were assembled into one normalized comma separated values file, which allowed for easy importation into a database. Once inside the database, simple query language expressions were executed to extract mini timelines to correlate temporal artifacts producing concise results that helped the examiner focus on specifics, reduce time and effort, and reduce the number of artifacts requiring investigation.

Appendix A - Log2Timeline (ver. 0.62) Input Module Listing

Name	Ver.	Description
altiris	0.1	Parse the content of an XeXAMInventory or AeXProcessList log file
analog_cache	0.1	Parse the content of an Analog cache file
apache2_access	0.3	Parse the content of a Apache2 access log file
apache2_error	0.2	Parse the content of a Apache2 error log file
chrome	0.3	Parse the content of a Chrome history file
encase_dirlisting	0.2	Parse the content of a CSV file that is exported from FTK Imager (dirlisting)
evt	0.2	Parse the content of a Windows 2k/XP/2k3 Event Log
evtx	0.5	Parse the content of a Windows Event Log File (EVTX)
exif	0.4	Extract metadata information from files using ExifTool
ff_bookmark	0.3	Parse the content of a Firefox bookmark file
ff_cache	0.2	Parse the content of a Firefox _CACHE_00[123]_ file
firefox2	0.3	Parse the content of a Firefox 2 browser history
firefox3	0.8	Parse the content of a Firefox 3 history file
ftk_dirlisting	0.3	Parse the content of a CSV file that is exported from FTK Imager (dirlisting)
generic_linux	0.3	Parse content of Generic Linux logs that start with MMM DD HH:MM:SS
iehistory	0.8	Parse the content of an index.dat file containing IE history
iis	0.5	Parse the content of a IIS W3C log file
isatxt	0.4	Parse the content of a ISA text export log file
jp_ntfs_change	0.1	Parse the content of a CSV output file from JP (NTFS Change log)
l2t_csv	0.1	Parse the content of a body file in the l2t CSV format
mactime	0.6	Parse the content of a body file in the mactime format
mcafee	0.3	Parse the content of log files from McAfee AV engine
mcafeefireup	0.1	Parse the content of an XeXAMInventory or AeXProcessList log file
mcafeehel	0.1	Parse the content of a McAfee HIPS event.log file
mcafeehs	0.1	Parse the content of a McAfee HIPShield log file
mft	0.1	Parse the content of a NTFS MFT file
mssql_errlog	0.2	Parse the content of an ERRORLOG file produced by MS SQL server
ntuser	1.0	Parses the NTUSER.DAT registry file
openvpn	0.1	Parse the content of an openVPN log file
opera	0.2	Parse the content of an Opera's global history file
oxml	0.4	Parse the content of an OpenXML document (Office 2007 documents)
pdf	0.3	Parse some of the available PDF document metadata
prefetch	0.7	Parse the content of the Prefetch directory
proftpd_xferlog	0.1	Parse the content of a ProFTPD xferlog log file
recycler	0.6	Parse the content of the recycle bin directory
restore	0.9	Parse the content of the restore point directory
safari	0.3	Parse the contents of a Safari History.plist file
sam	0.1	Parses the SAM registry file
security	0.1	Parses the SECURITY registry file
setupapi	0.5	Parse the content of the SetupAPI log file in Windows XP
skype_sql	0.1	Parse the content of a Skype database
software	0.1	Parses the SOFTWARE registry file
sol	0.5	Parse the content of a .sol (LSO) or a Flash cookie file
squid	0.5	Parse the content of a Squid access log (http_emulate off)
symantec	0.1	Parse the content of a Symantec log file
syslog	0.2	Parse the content of a Linux Syslog log file
system	0.1	Parses the SYSTEM registry file
tlm	0.5	Parse the content of a body file in the TLN format
volatility	0.2	Parse the content of a Volatility output files (psscan2, sockscan2, ...)
win_link	0.7	Parse the content of a Windows shortcut file (or a link file)
wmiproov	0.2	Parse the content of the wmiproov log file
xpfirewall	0.4	Parse the content of a XP Firewall log

Available lists of modules

Use the -f LISTNAME to use only the modules included in the list

linux

apache2_access, apache2_error, pcap, syslog, generic_linux, proftpd_xferlog,

webhist

chrome, firefox3, firefox2, ff_bookmark, opera, iehistory, iis, safari, sol,

win7

chrome, evt, exif, ff_bookmark, firefox3, iehistory, iis, mcafee, opera, oxml, pdf,
prefetch, recycler, restore, sol, win_link, xpfirewall, wmipro, ntuser, software, system,
sam, mft,

win7_no_reg

chrome, evt, exif, ff_bookmark, firefox3, iehistory, iis, mcafee, opera, oxml, pdf,
prefetch, recycler, restore, sol, ntuser, win_link, xpfirewall, wmipro, mft, winsrv evt,
exif, iis, isatxt, mcafee, pdf, prefetch, recycler, restore, setupapi, win_link,
xpfirewall, wmipro, ntuser, software, system, apache2_access, apache2_error, mft,
mssql_errlog,

winxp

chrome, evt, exif, ff_bookmark, firefox3, iehistory, iis, mcafee, opera, oxml, pdf,
prefetch, recycler, restore, setupapi, sol, win_link, xpfirewall, wmipro, ntuser,
software, system, sam, mft,

winxp_no_reg

chrome, evt, exif, ff_bookmark, firefox3, iehistory, iis, mcafee, opera, oxml, pdf,
prefetch, recycler, restore, setupapi, sol, ntuser, win_link, xpfirewall, wmipro, mft,

Appendix B - Log2Timeline Quick Reference Guide



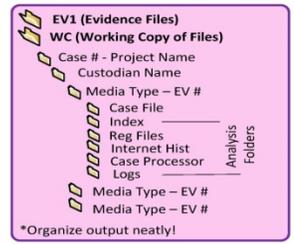
1. ADMINISTRATIVE STUFF

- Review Policies and Laws
- Chain of Custody form
- Digital Evidence Collection Form
- Consent form (if needed)
- Evidence Tracked and Stored

2. WORK PLAN (docs)

- Review Policies and Laws (if needed)
- Gain understanding of:
 - Background
 - If applicable, previous work
 - Requirements/Goal of analysis
 - Deliverable
- Create Analysis Work Plan
- Create Investigative Plan

3. SETUP CASE FOLDER (example)



4. CONFIRM IMAGE INTEGRITY

- Compare Acquisition and Verification Hash values (MD5, SHA)
- Save Verification Reports

GENERAL FORENSIC ANALYSIS CHECKLIST V.1.1

THE PURPOSE OF THIS REFERENCE GUIDE IS TO PROVIDE AN OVERVIEW AND OUTLINE OF COMMON PROCESSES, SOFTWARE, AND BEST PRACTICES FOLLOWED BY PROFESSIONALS CONDUCTING COMPUTER FORENSIC ANALYSIS

BY DAVID NIDES (12/16/2011)
 TWITTER: @DAVWADS
 BLOG: DAVNADS.BLOGSPOT.COM
 EMAIL: DNIDES@KPMG.COM
 CREDITS TO: ED GOINGS, ROB LEE & SANS
 QUESTIONS/FEEDBACK-CONTACT US!



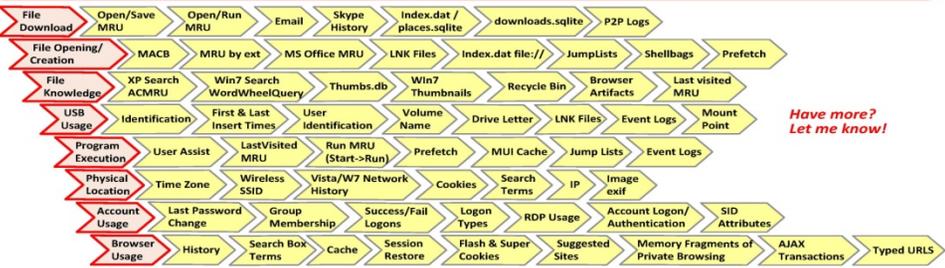
5. BEFORE YOU GET STARTED..

- Check physical size of drive and compare to physical label accounting for all drive space (Check for DCA/ HPA).
- Identify & compare logical partition size(s) to physical drive size to identify any deleted partitions or unused disk space
- Retrieve time zone settings for each disk and apply correct time zone, if applicable
- Rename hard disk volumes as necessary to "Recovery", "C", etc.
- Determine OS, service pack, OS install date, application list, owner, machine name, and other basic information.
- Retrieve user profile information (names, SIDs, create and last login dates)

6. EQUATION FOR SUCCESSFUL ANALYSIS: (TIMELINES + MANUAL ANALYSIS) x (PASSION + TIME + RESEARCH + RESOURCES) = "WINNING"

GENERAL AREAS	REGISTRY ANALYSIS (Win only)	SYSTEM ARTIFACTS	MEMORY RESIDUE	INTERNET
Analyze general folder locations for artifacts or data of interest. For example: >Desktop >User folders >Documents >Desktop >Network Shortcuts >Recently Opened File folders >System Temp folders >Browser Temp folders >System Folders (malware) >Autorun	The system registry hives (NTUSER.DAT, SYSTEM, SOFTWARE, SECURITY, SAM...) contain a vast amount of information that can be imperative to your analysis. For Examples: -OS Version (SOFTWARE) -Last logged on user (SAM) -Last Failed Logon (SAM) -Username & SID (SAM) -Shutdown (SYSTEM) -TimeZone (SYSTEM) -Drives Mounted by User (NTUSER.DAT) -File Ext Associations (NTUSER.DAT) -Installed application list (SOFTWARE) -Search History (SOFTWARE) -Removable Storage Devices (SYSTEM) Using Registry analysis tools, such as Reg Ripper, can aid your analysis..	There are many system related artifacts that may contain potentially relevant information including: -Backups (RP, VSC, others) -Event Logs (.evt, evtx) -Shell bags (Registry) -Jump Lists -Misc. Logs (firewall, AV, Apps, etc) -Removable Media Connections -LNK files -Prefetch -Pagefile	If applicable, perform memory analysis to gather volatile information including: -Handles, files, directories, processes, registry keys, Semaphores, events and sections -Open network ports -Hooks: Driver IRP, SSDT and IDT driver tree	Identify installed browsers and perform analysis on artifact such as: -Parse Internet history files (index.dat, sqlite, etc) -Check temp folders -Parse cookies -Cached pages -Form History/Auto complete files -Favorites/bookmarks -Toolbars -WebSlices -Browser plug ins -Registry analysis -Carve unallocated for deleted history artifacts
TIE IT ALL TOGETHER USING TIMELINE ANALYSIS - FLIP SIDE!		SOFTWARE RESIDUE Identify software (i.e Wiping tools, P2P, Sticky Notes, hacker tools, etc) and perform analysis on associated files (binary-malware analysis), logs, settings, registry and etc	E-MAIL/IM/SOCIAL ARTIFACTS Identify email clients or web access on system and perform analysis on associated data stores or application residue/settings: Client based (file examples): Windows -.OST, .PST, .MSG, Temp folder for Outlook attachments, Lotus Notes -.NSF, Mac -.EML, .EMLX, .MBOX Web based (OWA, Facebook, Twitter, etc.): -Logs -Internet History reconstruction -Cache	...BUCKETING ANALYSIS "TO DO" ITEMS LIKE THIS CAN HELP.

7. INTERPRETATION/REVIEW OF ARTIFACTS (examples) ..



Have more? Let me know!

EXPORT

- Export files from case for independent analysis with specialty tools. For example:
- | | |
|---------------------------|-----------------------|
| Memory | Email |
| Memorize | NUJK |
| Redline | Clearwell |
| Volatility (SIFT) | Recover My Email |
| AD PRTK | Bulk extractor (SIFT) |
| Passware | Image Mounting |
| Ophcrack | FTK Imager |
| Shellbags | ImDisk |
| Shellbags.py (SIFT) | Live View |
| LNK Files | OSFMount |
| Internet History | Virtual Box |
| WebHistorian | Stego |
| Outguess | Hashing |
| Tzworks | Md5deep (SIFT) |
| Lslink (SIFT) | Sha25deep (SIFT) |
| Event Logs (.evt & .evtx) | Hashdeep (SIFT) |
| Reg Ripper (SIFT) | Registry Decoder |
| Registry Viewer | AD Registry Viewer |
| Reglookup (SIFT) | YARU (SIFT) |
| Windows Journal | Parser |
| Tzworks | |
- SIFT Workstation is a great resource for tons of tools!**

8. REPORTING

- Document findings comprehensively
- Fact based Interpretation
- Remember who the audience is
- Remember requirements/expectations

SIFT REFERENCE GUIDE (V.1.1) – CREATING TIMELINES WITH THE SIFT WORKSTATION

1. VISIT: <http://computer-forensics11.sans.org/community/downloads>

- Download: SIFT Workstation VM Appliance
- Download: SIFT Workstation Installation

2. BOOT SIFT VM
Login: *sansforensics*
Password: *forensics*

3. ELEVATE PRIVS
\$ sudo su

4. CONNECT IMAGE TO SIFT
Plug hard drive to physical host and attach to SIFT VM

5. HARD DRIVE MOUNTING (if you are using log2timeline-sift and Single DD you can skip to 7-A)

SINGLE OR SPLIT IMAGE (2 options):

mount_ewf.py image.E01 /mnt/ewf
OR
ewfmount image.E01 /mnt/ewf/

mount -t ntfs -o ro,loop,show_sys_files,streams_interface=windows,offset=#### /mnt/ewf/<image> /mnt/windows_mount/

MOUNT TO MOUNT POINT

SINGLE IMAGE

mount -t ntfs -o ro,loop,show_sys_files,streams_interface=windows,offset=#### image.dd /mnt/windows_mount/

SPLIT IMAGE (2 step process)

affuse image.001 /mnt/aff

mount -t ntfs-3g -o loop,ro,show_sys_files /mnt/aff/<image> /mnt/windows_mount/

6. log2timeline default timezone is set to examiner local host. To change use -z [TIMEZONE] option. To list all available timezones: # log2timeline -z list



HOW TO CALCULATE THE OFFSET FOR MOUNTING

- Run mmls to query partition layout
mmls image.E01
- Identify partition and byte offset
- (Partition byte offset) x (bytes per sector) = offset #### to use!
Example: 63 X 512 = 32256

Note: If needed, repeat for each partition. Make new mount point:
mkdir /mnt/windows_mount2/

- log2timeline PARSING PLUGINS
- apache2_error - Apache2 error log file
 - chrome - Chrome history file
 - encase_dirlisting - CSV file that is exported from encase
 - evt - Windows 2k/XP/2k3 Event Log
 - evtx - Windows Event Log File (EVTX)
 - exif - Metadata information from files using ExifTool
 - ff_bookmark - Firefox bookmark file
 - firefox2 - Firefox 2 browser history
 - firefox3 - Firefox 3 history file
 - ftk_dirlisting - CSV file that is exported from FTK Imager (dirlisting)
 - generic_linux - Generic Linux logs that start with MMM DD HH:MM:SS
 - iehistory - index.dat file containing IE history
 - ils - IIS W3C log file
 - isatt - ISA text export log file
 - jp_ntfs_change - CSV output file from JP (NTFS Change log)
 - mactime - Body file in the mactime format
 - mcafee - Log file
 - mft - NTFS MFT file
 - mssql_errlog - ERRORLOG file produced by MS SQL server
 - ntuser - NTUSER.DAT registry file
 - opera - Opera's global history file
 - oxml - OpenXML document pcap
 - pcap - PCAP file
 - pdf - Available PDF document metadata
 - prefetch - Prefetch directory
 - recycler - Recycle bin directory
 - restore_0-9 - Restore point directory
 - safari - Safari History.plist file
 - sam - SAM registry file
 - security - SECURITY registry file
 - setupapi - SetupAPI log file in Windows XP
 - skype_sql - Skype database
 - software - SOFTWARE registry file
 - sol - sol (LSO) or a Flash cookie file
 - squid - Squid access log (http_emulate off)
 - syslog - Linux Syslog log file
 - system - SYSTEM registry file
 - tl - Body file in the TLN format
 - volatility - Volatility output files (psscans2, sockscan2, ...)
 - win_link - Windows shortcut file (or a link file)
 - wmiprov - WMI log file
 - xpfirewall - XP Firewall log

7-A: AUTOMATED SUPER TIMELINE CREATION

log2timeline-sift -o -z [TIMEZONE] -p [PARTITION #] -i [IMAGE FILE]

DISK IMAGE (prompt for partition, mount, and run):
XP # log2timeline-sift -z ESTSEDT -i image
WIN7 # log2timeline-sift -win7 -z ESTSEDT -i image

FOR PARTITION (mount and run using all applicable plugins):
XP # log2timeline-sift -z ESTSEDT -p 0 -i partition
WIN7 # log2timeline-sift -win7 -z ESTSEDT -p 0 -i partition

OTHER USAGE EXAMPLES:
Display list of available plugins:
log2timeline -f list
Run log2timeline use -o flag to use only specific plugins:
log2timeline-sift -o evtx,prefetch -z ESTSEDT -i image.dd
Help (man page):
log2timeline-sift -h

8. CSV FILE OUTPUT (/cases/timeline-output-folder)

-date: Date of the event, in the format of MM/DD/YYYY
-time: Time of day, expressed in a 24h format, HH:MM:SS
-timezone: the timezone that was used to call the tool with.
-MACB: MACB meaning of the fields: comp w/ mactime format.
-source: Source short name (i.e. registry entries are REG)
-sourcetype: Desc of the source ("Internet Explorer" instead of WEBHIST)
-type: Timestamp type (i.e. "Last Accessed", "Last Written")
-user: Username associated with the entry, if one is available.
-host: Hostname associated with the entry, if one is available.
-short: Contains less text than the full description field.
-desc: where majority info is stored, the actual parsed desc of the entry.
-version: Version number of the timestamp object.
-filename: Filename with the full path that contained the entry
-inode: inode number of the file being parsed.
-notes: Some input modules insert additional information in the form of a note, which comes here. Or it can be used during the review.
-format: input module name used to parse the file.
-extra: Additional information parsed is joined together and put here.

7-B: MANUAL "MICRO" TIMELINE CREATION

log2timeline [OPTIONS] [-f FORMAT] [-z TIMEZONE] [-o OUTPUT MODULE] [-w BODYFILE] LOG_FILE/LOG_DIR [-] [FORMAT FILE OPTIONS]

FILE SYSTEM METADATA (using log2timeline or fls)
Parse file system data w/log2timeline from mounted file system:
log2timeline -f mft -o mactime -r -z ESTSEDT -w mft.body /mnt/volume/
OR Extract MFT from image using Sleuthkit:
fls -m "" -o offset -r image.dd > fls.body
Convert body file format to CSV format w/ mactime:
mactime -b fls.body -d > fls.csv

ARTIFACTS (run I2L on mounted file system with plugins recursively)
Extract artifacts w/ log2timeline and run on mounted file system:
log2timeline -f firefox3,chrome -o mactime -r -z ESTSEDT -w web.body /mnt/volume/
Convert body file format to CSV format w/ mactime:
mactime -b log2timeline.body -d > log2timeline.csv

9. FILTER TIMELINE

Filter timeline with date range to include only:
I2t_process -b timeline.csv MM-DD-YYYY..MM-DD-YYYY > filtered.csv
Filter timeline with keyword list (one term per line in keywords.txt):
I2t_process -b timeline.csv -k keywords.txt > filtered.csv
What sources are in your timeline?
awk -F , '{print \$6;} timeline.csv | grep -v sourcetype | sort | uniq
Find all LNK files that reference E Drive
grep "Shortcut LNK" timeline.csv | grep "E"
FindMountPoints2 entries that reference E Drive
grep "MountPoints2 key" timeline.csv | grep "E drive"
grep USB timeline.csv | grep "SetupAPIlog"

File System	M	A	C	B
Ext2/3	Modified	Accessed	Changed	N/A
FAT	Written	Accessed	N/A	Created
NTFS	File Modified	Accessed	MFT Modified	Created
UFS	Modified	Accessed	Changed	N/A

HELP? OPTIONS? USAGE?
log2timeline -help
Log2timeline-sift -help
L2t_process -help

OTHER log2timeline OUTPUT FORMATS
Note: CSV is Default Output
-BeeDocs - Mac OS X visualization tool
-CEF - Common Event Format - ArcSight
-CFTL - XML file- CyberForensics TimeLab visualization tool
-CSV - comma separated value file
-Mactime - Both older and newer version of the format supported for use by TSK's mactime
-SIMILE - XML file - SIMILE timeline visualization widget
-SQLite - SQLite database
-TLN - Tab Delimited File
-TLN - Format used by some of H Carvey tools, expressed as a ASCII output
-TLNX - Format used by some of H Carvey tools, expressed as a XML document

10. CONNECT TO SIFT

- 1. VM -> SETTINGS -> OPTIONS -> Shared Folders -> Always Enabled (Check)
- 2. SIFT Desktop -> VMware-Shared-Drive
- Access from a Win Machine
\\SIFTWORKSTATION

11. REVIEW TIMELINE

- Review timelines using:
- Open, Soft, Filter with Excel
 - Import into SPLUNK
 - SIMILE
 - Tapestry

BY DAVID NIDES (12/16/2011)
TWITTER: @DAVNAIDS
BLOG: DAVNAIDS.BLOGSPOT.COM
EMAIL: DVIDES@KPMG.COM
CREDITS TO: ED GOINGS, ROB LEE, KRISTIN GUIDONSON, KPMG & SANS!!
QUESTIONS/FEEDBACK-CONTACT US!

KEY
Red text - image/source
Blue text - mount point
Purple text - output file
Green text - log2timeline plugins
Brown text - TimeZone

Appendix C – Script Used to Create Incident

Windows 7 Incident Script

- User logs on
- User inserts USB device (mounts as E:\)
- User clicks Open Folder view from Autorun pop-up menu
- User navigates to USB device and runs ProcessList.exe
 - Program run Process Hacker Program
 - Program Seviles2000.exe runs which is renamed Netcat
- Hacker starts Netcat and connects to User's system
- Command shell is given to Hacker
- Hacker navigates to User's Documents folder
- Hacker creates file KillBoddy.txt using Echo command
- Hacker disconnects from User's system
- User navigates to Google News
- User logs off

Windows XP Incident Script

- User logs on
- User closes "Take Windows Tour" pop-up
- User opened Notepad and created LoveLetter.txt
- User closed Notepad
- User opened Calculator and performed 1+1=
- User closed Calculator
- User Inserts USB device (mounts as E:\)
- User clicks Open Folder view from Autorun pop-up menu
- User navigates to USB device and runs ProcessList.exe
 - Program run Process Hacker Program
 - Program Seviles2000.exe runs which is renamed Netcat
- User runs Solitaire
- Hacker starts Netcat and connects to User's system
- Command shell is given to Hacker
- Hacker navigates to User's Documents folder
- Hacker performs Dir command
- Hacker deletes LoveLetter.txt file

- Hacker creates file HateLetter.txt using Echo command
- Hacker performs command, %systemdrive%\recycler* /F /S
- Hacker uses Attrib.exe command
- Hacker disconnects from User's system
- User clicks off Screen Saver
- User closes Solitaire program
- User logs off system

Bibliography

- Accessdata. (2008). *Accessdata Supplemental Appendix*. July 22, 2008. Retrieved from <http://accessdata.com/downloads/media/Registry%20Quick%20Find%20Chart%20%207-22-08.pdf>.
- Altheide, C. & Carvey, H. (2011). *Digital Forensics with Open Source Tools*. Syngress Publication. Waltham, MA. p 231-239.
- Availability and description of the File Checksum Integrity Verifier utility*, September 10, 2011, Microsoft Support. Retrieved from <http://support.microsoft.com/kb/841290>.
- Carvey, H. (Feb 2012). *How To: USB Thumb Drives*. Feb 4, 2012. Retrieved from <http://windowsir.blogspot.com/2012/02/howto-usb-thumb-drives.html>.
- Carvey, H. (Jan 2012). *Timeline Analysis*. January 13, 2012. Retrieved from <http://windowsir.blogspot.com/2012/01/timeline-analysis.html>.
- Carvey, H. (Dec 2011). *Even More Stuff*. December 19, 2011. Retrieved from <http://windowsir.blogspot.com/2011/12/even-more-stuff.html>.
- Carvey, H. (Jan 2011). *More VSCs*. January 18, 2011. Retrieved from <http://windowsir.blogspot.com/2011/01/more-vscs.html>.
- Carvey, H. (Nov 2010). *Updates*. November 17, 2010. Retrieved from <http://windowsir.blogspot.com/2010/11/updates.html>.
- Carvey, H. (July 2010). *Registry and Timeline Analysis*. July 8, 2010. Retrieved from the SANS US Digital Forensic and Incident Response Summit 2010 archive at <http://computer-forensics.sans.org/summit-archives/2010/files/4-carvey-wkshp.pdf>.
- Carvey, H. (Feb 2010). *MFT Analysis*. February 8, 2010. Retrieved from <http://windowsir.blogspot.com/2010/02/mft-analysis.html>.
- Carvey, H. (Oct 2009). *Free Tools*. October 23, 2009. Retrieved from <http://windowsir.blogspot.com/2009/10/free-tools.html>.
- DoD CIO Website (2012). Retrieved from <http://dodcio.defense.gov>.
- Edwards, D (Nov 2011). *Computer Forensic Timeline Analysis with Tapestry*. SANS Gold Paper accepted November 12, 2011. Retrieved from http://www.sans.org/reading_room/whitepapers/incident/computer-forensic-timeline-analysis-tapestry_33836.

- Guðjónsson, K. (2010). *Mastering the super timeline with log2timeline*. SANS Gold Paper accepted June 29,2010. Retrieved from http://www.sans.org/reading_room/whitepapers/logging/mastering-super-timeline-log2timeline_33438.
- Harrell, C. (Dec 2011). *Ripping Volume Shadow Copies Sneak Peek*. December 19, 2011. Retrieved from <http://journeyintoir.blogspot.com/2011/12/ripping-volume-shadow-copies-sneak-peek.html>.
- Harell, C. (Jan 2012). *Dual Purpose Volatile Data Collection Script*. January 2, 2012. Retrieved from <http://journeyintoir.blogspot.com/2012/01/dual-purpose-volatile-data-collection.html>.
- Harrel, C. (Nov 2011). *What's a Timeline*. September 7, 2011. Retrieved from <http://journeyintoir.blogspot.com/2011/09/whats-timeline.html>.
- Harrell, C. (Apr 2011). *A Little Help with Volume Shadow Copies*. April 20, 2011. Retrieved from <http://journeyintoir.blogspot.com/2011/04/little-help-with-volume-shadow-copies.html>.
- Jones, K. (2011). *Casey Anthony Murder Trial: The Computer Evidence (Part 2)*. June 14, 2011. Retrieved from <http://www.jonesdykstra.com/blog/201-%C2%AD%E2%80%90caseyanthony-%C2%AD%E2%80%90part2>.
- Jones, K. J. & Belani, R. (2010). *Web Browser Forensics, Part 1*. Symantec, November 2, 2010. Retrieved from <http://www.symantec.com/connect/articles/web-browser-forensics-part-1>.
- Keizer, G. (2012). *Adobe Plugs 6 Critical Holes in Reader*. Computerworld, January 11, 2012. Retrieved from http://www.computerworld.com/s/article/9223344/Adobe_plugs_6_critical_holes_in_Reader.
- Kornblum, J. (2012). *Md5deep version 4.0.1, Critical Bug Fixes*. Live Journal, January 22, 2012. Retrieved from <http://jessekornblum.livejournal.com/278069.html>.
- Lloyd, J., Healy, P., & Klemack, J. C. (2011). *Phone Calls from Michael Jackson's Doctor Take Spotlight*. ABC Los Angeles. October 4, 2011. Retrieved from <http://www.nbclosangeles.com/news/local/Conrad-Murray-Trial-Michael-Jackson-Day-6-131046673.html>.
- Mandia, K., Proise, C., & Pepe, M. (2003). *Incident response & computer forensics*. (2nd ed.). Emeryville, CA: McGraw-Hill.
- National Institute of Justice, (2008). *Electronic Crime Scene Investigation: A Guide for First Responders, Second Edition*. Department of Justice. April 2008.

- Nides, D. (Jan 2012). *Timeline Analysis: The Hybrid Approach*. January 16, 2012. Retrieved from <http://davnads.blogspot.com/2012/01/timeline-analysis-hybrid-approach.html>.
- Nides, D. (Dec 2011). *Digital Forensics SIFT'ing: Cheating Timelines with log2timeline*. Retrieved from <http://computer-forensics.sans.org/blog/2011/12/16/digital-forensics-sifting-cheating-timelines-with-log2timeline>.
- NTFS.com, *NTFS vs. FAT*, retrieved from http://www.ntfs.com/ntfs_vs_fat.htm.
- NtfsDisableLastAccessUpdate (2010). Retrieved from <http://technet.microsoft.com/en-us/library/cc959914.aspx>.
- Olsson, J. & Boldt, M. (2009), *Computer forensic timeline visualization tool*, digital investigation, vol. 6, pp. S78-S87, 2009. Retrieved from <http://www.dfrws.org/2009/proceedings/p78-olsson.pdf>.
- Pipkin, D. L. (2000). *Information security: Protecting the global enterprise*. Upper Saddle River, New Jersey: Prentice Hall PTR.
- Phillip, A., Cowen, D., & Davis, C. (2009). *Hacking exposed computer forensics, second edition: Computer forensics secrets & solutions*. (2 ed.). New York, NY: McGraw Hill.
- PPM-Perl Package Manager (2010), Retrieved from <http://docs.activestate.com/activeperl/5.10/bin/ppm.html>.
- Roger's Access Blog (2008). *Ambiguous Outer Joins*. September 22, 2008. Retrieved from <http://rogersaccessblog.blogspot.com/2008/09/ambiguous-outer-joins.html>.
- SANS360 (2011). *Digital Forensics and Incident Response Lightning Talk*. December 13, 2011. Retrieved from <https://computer-forensics.sans.org/sans360/dec2011>.
- SANS Computer Forensics and Incident Response (2011). *Log2Timeline Cheatsheets*. Retrieved from <http://blogs.sans.org/computer-forensics/files/2011/12/digital-forensics-incident-response-log2timeline-timeline-cheatsheet.pdf> on January 2, 2012.
- SANS Computer Forensics and Incident Response (2011). *SANS Investigate Forensic Toolkit (SIFT) Workstation Version 2.12*. Retrieved from <http://computer-forensics.sans.org/community/downloads> on February 19, 2012.

- Security Technical Information Guide (STIG), retrieved from <http://iase.disa.mil/stigs>.
- Simile Widget Example (2009), *JFK Assassination*, retrieved from <http://www.simile-widgets.org/timeline/examples/jfk/jfk.html>.
- Stephenson, P. (2002), *The forensic investigation steps*, Computer Fraud & Security, vol. 2002, no. 10, pp. 17-19, 2002.
- Timestomp (2012), *Timestomp: Metasploit Unleashed*, retrieved from <http://www.offensive-security.com/metasploit-unleashed/Timestomp>
- Timestomp (2010). Retrieved from <http://www.forensicswiki.org/wiki/Timestomp>.
- UltimateWindowsSecurity.com, *Windows Security Log Events*, retrieved from <http://www.ultimatewindowssecurity.com/securitylog/encyclopedia/Default.aspx>.

REPORT DOCUMENTATION PAGE				<i>Form Approved OMB No. 074-0188</i>	
<p>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</p>					
1. REPORT DATE (DD-MM-YYYY) 14 June 2012		2. REPORT TYPE Graduate Research Project		3. DATES COVERED (From – To) 15 June 2011 – 14 June 2012	
4. TITLE AND SUBTITLE Analysis of Forensic Super Timelines				5a. CONTRACT NUMBER N/A	
				5b. GRANT NUMBER N/A	
				5c. PROGRAM ELEMENT NUMBER N/A	
6. AUTHOR(S) Esposito, Stephen J. Major				5d. PROJECT NUMBER N/A	
				5e. TASK NUMBER N/A	
				5f. WORK UNIT NUMBER N/A	
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way, Building 640 WPAFB OH 45433-7765				8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/ICW/ENG/12-04	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Intentionally Left Blank				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.					
13. SUPPLEMENTARY NOTES This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.					
14. ABSTRACT As the use and adoption of networked electronic devices grows, their use in conjunction with crimes also increases. Extracting the probative evidence from these devices requires experienced digital forensics examiners. These examiners use several specialized tools that interpret the raw binary data present in digital media. Once the evidentiary artifacts are collected, one of the examiners goals is to assemble a narrative that describes when events occurred based on the time associated with the artifacts. Unfortunately, generating and evaluating these narrative super timelines is a manual and labor intensive process. This research focuses on aiding the examiner in evaluation through the generation of several queries that can extract and connect the temporal artifacts, and produce concise timelines. Extracting and analyzing these concise timelines allows the examiner to decrease the number artifacts to search through from hundreds of thousands of artifacts to only a hundred artifacts or less. Additionally, queries that correlate various artifacts allow the examiner to confirm or deny attribution of the user's actions. Application of the queries presented on a fictitious event demonstrates their ability to reduce the number of artifacts and facilitate the understanding of the activities surrounding the incident.					
15. SUBJECT TERMS Digital Forensics, Log2timeline, Super Timeline, Artifact Analysis, Incident Collection					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Gilbert Peterson, PhD (ENG)
U	U	U	UU	87	19b. TELEPHONE NUMBER (Include area code) (937)257-3636x4581; gilbert.peterson@afit.edu