CHAMPLAIN COLLEGE

LCDI

The Senator Patrick Leahy Center for Digital Investigation

Plaso

Written by
Nick Aspinwall
Chapin Bryce
Researched by
Nick Aspinwall
Chapin Bryce

175 Lakeside Ave, Room 300A
Phone: 802/865-5744
Fax: 802/865-6446
http://www.lcdi.champlin.edu

Published Date 12/02/2013

Disclaimer:

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

Contents

Introduction 3
Background: 3
Purpose and Scope: 3
Research Questions: 3
Terminology: 3
Methodology and Methods 4
Equipment Used 4
Data Collection: 4
Analysis 5
Results 6
Plaso on Windows 6
NTFS Partition 6
HFS Partition 6
FAT32 Partition 6
Plaso on Mac 6
NTFS Partition 6
HFS Partition 7
FAT32 7
Plaso on Linux 7
NTFS Partition 7
HFS Partition 7

FAT32 Partition 7
Conclusion 7
Further Work 8
References 9

Introduction

Timeline analysis has become an increasingly popular method of analysis in Digital Forensics. . Plaso is a rework of the original Log2timeline, created by Kristinn Gudjonsson. Currently, Plaso is in the alpha stage of testing and development. Plaso is a backend written in python for Log2timeline. This project's is intended to test and compare Log2timeline against Plaso. We are going to be conducting testing on an image across multiple host platforms and comparing the results. It is important to note that whenever Log2timeline is referenced, we are referring to the original Perl version 0.65. Whenever Plaso is referenced, we are referring to either Log2timeline.py version 1.0.2 alpha (in Linux) or 1.0.1 (in Windows and Mac).

Background:

Plaso is the framework that log2timeline operates on. While log2timeline still runs in a similar manner, it now has a new framework and new post-processing featuring. A major change in the framework is how log2timeline stores certain information. Log2timeline has the potential to create the data directly into csv format. With the Plaso framework, log2timline instead puts the data into an archive file. This archive file then has post-processing tools ran against it. The tool that is used for the pre-processing is called psort. Psort dictates how the csv file is generated. It also allows for different post-processing steps to be performed on the timeline, such as narrowing down all the results to just events that contain a specific keyword, or events only about Chrome history. We focused our research on seeing how the results differed from data from the older log2timeline. For this we ran Psort with the minimum amount of flags and conditions, essentially trying to grab everything possible.

For Log2timeline to run against an image, the desired image has to be mounted. This can create a problem if Log2timeline is being used within a windows environment. Windows does not recognize certain file systems such as HFS and HFS+, so the tool cannot be ran against an unrecognized drive. If a drive was formatted with these file systems, the timeline creation tool would have to be used on another system. Plaso can be run against a raw image or a mounted image. As long as the image is a single raw file, Plaso is able to run. Plaso can be run against a drive that has multiple partitions, as long as an offset is given so that Plaso has a set starting point. This is similar to mounting the image, where each mount point would have to be a single partition.

Purpose and Scope:

The purpose of this project is to compare the results of Plaso against those from Log2timeline. It is a valuable way to see if Plaso or Log2timeline could gather events that other may have missed. We want to compare the number of events that each tool can gather and the types of events.

Research Questions:

Our research question is: what are the differences between Plaso and Log2timeline when ran across different platforms and against different file systems? We also want to see if there are differences between Plaso's data on multiple host operating systems.

Terminology:

Plaso: the reworked backend to log2timeline v0.65
File system offset: the starting point on the physical drive where the partition begins.
NTFS: File System used by Windows 7 partitions
HFS+: File System used by Mac OS X

Fat32: common file system for smaller drives such as flash and thumb drives.
OS: Operating System
VSS: Volume Shadow Copies (a checkpoint backup system)
VM: Virtual Machine

Methodology and Methods

Our research was conducted on three separate machines. The first machine was a 64bit Windows 7 Professional machine. The Linux machine we used was a prebuilt distro with plaso 1.0.2 already built by Greg Freemyer. It is important to note that the version of Plaso on our linux machine was one version newer then the version used for Mac and Windows. Our Windows and Mac machines were running alpha version 1.0.1. This machine was a virtual machine on workstation version 10. This VM can be downloaded from the DIFR (Digital Forensics, Incident Response) Open Suse at http://susestudio.com/a/eD1wrT/dfir-opensuse-gnome-desktop-32bit. Additionally, we installed and ran Plaso on Mac OS X 10.8.4. The image we used wass a single raw image format that contained three separate partitions. It had a Windows NTFS partition with VSS enabled, a Fat32 partition, and a HFS+ partition. The image used was from the LCDI Fire project. This image was created with care to note everything that was done on the system in an excel sheet. This allowed for us to check Plaso and Log2timeline's results against what was recorded when the image was created. The image was originally in E01 format, but using FTK imager we were able to transfer the image into raw format.

Our initial goal was to run Plaso on SIFT 2.14. However, because of the difference in the Linux version on Sift 2.14 and the version required by Plaso, this proved to be problematic. Kristinn Gudjonsson explained why this was a problem: SIFT 2.14 runs on Lucid and when trying to compile some of the libraries there are syntax errors because Plaso is built for Precise. Because of this difference, we used Greg Freemyer's DFIR Open Suse Linux distro with Plaso already installed.

Equipment Used

The hardware used in this project was a Windows machine and Mac OS X machine. The Mac OS X machine contained a 2.93 GHz Intel core 2 duo processor with 4 GB of Ram, running OS X 10.8.4. The Windows machine was a 64-bit windows 7 professional with 16GB of Ram and an Intel i7 processor. The Linux distro was installed in a virtual machine. The software used was log2timeline version .65 (the Perl version) and Plaso backend versions 1.0.2 and 1.0.1 (alpha). For Windows, Plaso contained a standalone .exe, Linux had log2timeline.py, and Mac ran log2timeline. It should be noted that log2timeline in Perl form is sometimes referenced as log2timeline.pl.

Data Collection:

All data was collected using the Plaso and Log2timeline tools. Plaso was run against an NTFS, FAT32, and HFS+ file system on each operating system. The steps taken were documented on each OS. Screenshots of commands and errors were logged in word documents, and the dump files were saved. Psort is the Plaso post-processing tool. This was used on the dump files to create a csv file that contained all the timeline events. This csv file will be used to make the comparisons between Plaso on different operating systems, as well as between Plaso and log2timeline. Table 1 shows the offsets for each file system that was used.

The commands to start Plaso are similar across multiple systems, with only small differences for each. The command on the Linux machine was " log2timeline.py –o 65443840 –vss /mnt/disk2/output/plaso_ntfs.dump /mnt/disk2/fire_raw_plaso.001." The .py at the end of log2timeline specifies the differences between Perl log2timeline and the Plaso backend. The only flag that is required for this is the –o flag, or offset flag. If Plaso was run against a system with only one partition, the –o flag would not be needed. Because this is a windows partition, the –vss flag is used to include the shadow copies in the events. If vss is not enabled on the windows computer, or the investigator does not want shadow copies to be included in the events, then this can be left out. The Windows 7 command is almost identical to the Linux command, with the exception of the .exe extension. The mac command requires additional flags. With both Linux and Windows, the first location and file is the dump file, or file that is created by Plaso. The second file and location is the image or mount point to be processed. Mac requires that this is specified with a –w (write) flag, specifying where the dump file will be written. The source does not require a flag and is assumed to be the last section in the command.

Psort also follows similar rules. The command for Linux is "psort.py –w /mnt/output/name.csv /mnt/disk2/plaso_ntfs.dump." The –w flag is used to specify where the csv will be written to. The last section in the Psort command is the dump file to be processed. This example of the command is essentially the minimum needed to have the tool run successfully. The Windows command for Psort is identical, except that the tool is called psort.exe. Mac has one difference from Linux and Windows, other than the file extension. Mac requires the –s (source) flag to be specified as the last segment in the command.

Table 1: File System Offset

File System	Image offset
NTFS	65443840
HFS	409640
FAT32	38029312

Analysis

All activity was recorded within spreadsheets on the system, providing a direct comparison to when events took place. This provides a better understanding of the events shown by our programs. Every search, click, and command was recorded. In general, the results of Plaso and Log2timeline are all similar with minor variations. It would be very difficult to compare every individual event, so instead each situation and configuration is generalized when comparing Log2timeline and Plaso. The main focus is the results retrieved by Plaso, so this is where the detail is reported. Because Plaso requires a post processing tool to get the results to a csv format, the same format command was used with all data. We used the most basic of the psort commands. The commands specified the location and file to write the results to, as well as the location of the dump file. This should grab everything inside that dump file read by Plaso.

Results

Plaso on Windows

NTFS Partition

Running Plaso on Windows offered a host of problems. The encountered the first notable problem when Plaso was run against the NTFS partition in the image, which is about 23.3 GB in size. We used the shadow copies flag, but the program was hung-up and ran for a little under two days. At this point, we quit the program. We then ran the post processing command (psort) on the incomplete dump file to see if there was any data and located a small number of entries. We tried running the Plaso command again, both with and without the vss flag. The program was able to complete the second time without the vss flag, but could not run with the vss flag. For the completed run, we retrieved close to 843,137 rows of data and events inside the csv file, after post processing. Unfortunately, at this point, an error occurred when trying to read the vss information and Plaso was not able to open the volume. The exact error was "WARNING:root:Error while trying to read VSS information: pyvshadow_volume_open_file_object: unable to open volume." Even without the vss information, useful timeline data was collected during our testing. The initial dates and times at the top of the csv file caught our attention immediately. The first entry in the time line is set to January 1, 1970 and the date goes through a range up to 2013. These are showing the creation of NTUser.dat and other registry events and are just from the installation. The last date and time on the document is around the year 2038. These appear to expiration times from web history and cookies.

HFS Partition

We were able to successfully run Plaso on Windows 7 against the HFS partition. Please note that when we are running these commands, we are not filtering anything. We want to gather as much data as we possibly can. Originally, there were run errors that caused the program to break, but it was able to complete and report back 1,219,360 events. As with Windows, there are dates that are completely out of range. This is due again to the instillation time and expiration of some files. Unlike HFS, NTFS was able to tell us when the type was web history or expiration time. All the events are accurate to what was recorded at the time the original action took place. We were not able to compare Plaso to log2timeline on windows for the HFS partition, because the partition could not be mounted in the Windows machine.

FAT32 Partition

Running Plaso against the FAT 32 partition was the fastest of all our tests. This also returned the smallest number of events, at 197. Each one of these events comes from the .Spotlight-v100 directory, created by Mac OS X and dealing with mounted volumes. This is a great place to look at the times for the index of the volumes when mounted. The run completed successfully.

Plaso on Mac

NTFS Partition

As with Windows, we encountered difficulties going against the NTFS partition with vss flag enabled. However, we removed the flag and the program still appeared to hang-up. The first run lasted just over two days. Because both runs appeared to hang-up, we quit the program and ran post processing. We were able to retrieve 45,559 events. It is important to note that we were not working with a complete, successful run of

Plaso, leaving us with many discrepancies This run gathered a considerable amount less than the Windows host OS, and the beginning date was not 1970 but 6/10/2009. This is from the installation of Windows.

HFS Partition

Our run against the HFS partition finished successfully and was able to retrieve 58,530 events. The events are stemming from the Applications, spotlight, and .fseventsd locations. When using windows as the host OS, Plaso returned events from the same locations plus a number of others, such as Library, System, Private, and users. A considerable amount of information was left out with OS X. During the run, many errors occurred when retrieving information from the Users directory.

FAT32

Running on the spotlight partition returned 311 events. These events are similar to the results from Windows. The beginning time stamp is 2/7/2013, and the last event has a time stamp of 3/21/2013.

Plaso on Linux

NTFS Partition

When using Linux as a host, we used a pre-configured Linux machine. The exact distribution can be downloaded from the link provided in our References section. We again encountered difficulties with a few parsers, including WinEvtx Parser and Winreg.py. Even with these errors, Plaso was still able to retrieve 1,180,707 events. The starting date from installation was again from 1970. The last event was an expiration time for a web history file on January, 19 2038. We included the VSS flag, and Linux was able to parse and read through the VSS files. Our Linux machine was the only one able to read the VSS files.

HFS Partition

Running Plaso against an HFS partition provided us with interesting results. The first run encountered a number of different of parser errors, and appeared to get hung-up. At this point, we quit the program and ran the psort tool to see if any events could be extracted, retrieving 397,712. Most of the events that we looked included the correct time of the event. However, we let Plaso run again unaided against HFS. We let the tool run for a little over 48 hours, and after this time, the program appeared to not make any more progress and kept spitting out errors. We quit the tool and again ran psort against the dump file. This time, no data was able to be parsed out of the dump file. This could be due to quitting at a later time or a combination of other factors.

FAT32 Partition

Running against the Spotlight partition on the Linux machine, Plaso was able to read 217 events. This was the only event that did not provide any errors on the run. This version of Plaso retrieved fewer events than version 1.0.1. For events that matched with the other runs, the times and dates were the same, but Linux retrieved less events than the Mac and more than the Windows machine.

Conclusion

With the program in the Alpha stage, there is still room for improvement. Overall, the times and dates were all accurate for the actions that took place on the machine. The differences in the results from each host OS are

most likely due to the individual errors that occurred on each system. Our team did not have a single run that did not produce errors. Each host and target partition presented unique errors during the processing. Most of these errors were due to certain parsers running into problems.

Further Work

Once a more stable version is released, we would be interested in retrying this project and comparing the results. We would also like to be able to have Plaso run on SIFT.

References

Freemyer, G. (n.d.). DFIR openSUSE GNOME desktop. *SUSE Gallery*. Retrieved from http://susestudio.com/a/eD1wrT/dfir-opensuse-gnome-desktop-32bit

Plaso - home of the super timeline. (n.d.). *Plaso - Home of the Super Timeline*. Retrieved from https://sites.google.com/a/kiddaland.net/plaso/