CSF Tool Project – 4050.581.01

THE expandable tool for all of your timeline needs!

FLSmac

Brian Rowe

Contents

[Executive Summary 2](#_Toc355377264)

[Current condition overview 3](#_Toc355377265)

[Tool selection 4](#_Toc355377266)

[Tool development 5](#_Toc355377267)

[Any particularly interesting issues encountered along the way 6](#_Toc355377268)

[Conclusion 6](#_Toc355377269)

[Personal Conclusion 6](#_Toc355377270)

[To Use 6](#_Toc355377271)

[Appendix 7](#_Toc355377272)

[Contents of flsmac.pl 7](#_Toc355377273)

[Contents of flsmac.info 9](#_Toc355377274)

[References 10](#_Toc355377275)

# Executive Summary

After working on the Computer System Forensics lab 3, and the subsequent midterm, I realized a screaming need to write a simple program that could combine the complicated/confusing tasks of running FLS and mactime to prepare a simple file activity timeline.

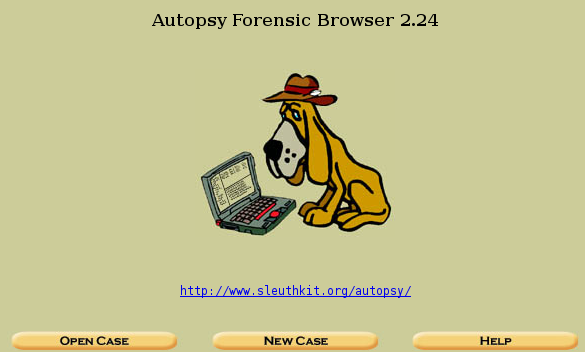
There are only a few front-ends for these utilities, but they are far from simple and convenient (see Current Condition Overview below). From this missing need, FLSmac was born. FLSmac is a simple file timeline creation utility that combines fls and mactime into a simple-to-use command-line-driven interface. It was designed with default parameters so that the average user could run it without any knowledge of how the underlying program actually works. This is not to say that it doesn’t clue the user into its operation, as it fully outputs all the commands it will execute interactively while the user is constructing the commands. This allows him to see what is actually going on in the background, while still providing a simple file timeline utility.

In summary, FLSmac is THE expandable tool for all of your timeline needs!

Get it now at <http://sourceforge.net/projects/flsmac/>!

# Current condition overview

There are not many tools that provide front-ends for fls and mactime. One such front-end is Autopsy [1] which is included with the Sleuth Kit, from which mactime and fls come. One problem with this solution is that it is a web-based Graphical User Interface (GUI) that cannot be run over a terminal connection. Another drawback is that it is designed to be a forensics case management tool, so one cannot quickly or simply create a file activity timeline. Instead, a case has to be created, evidence added, then evidence processed, etc... Autopsy’s mandatory case creating web GUI is shown in the image below.



Another option is PTK Forensics which is a tool published by DFLabs [2]. This package once again uses a GUI front-end. Even worse, PTK Forensics costs $1,258.80 for a single user license! Due to its price, I can only theorize as to its capabilities.



The objective then, was to write a tool that is command-line-driven but easy to use while still meeting all timeline needs.

# Tool selection

The tools improved upon and simplified by FLSmac are fls and mactime [1], hence the name. Fls is used to list file and directory names within a disk image, saving this information to a file to be read by mactime. Mactime is then used to create a timeline from the file created by fls. This is not an easy or user friendly task using these tools, as demonstrated in figures 1 & 2 below (from the CSF midterm practical given 20123).



Figure - midterm practical fls example



Figure - midterm practical mactime example

The objective for the new tool was to provide a simple user interface to not only simplify the task of entering parameters and options, but also allow the use of one rather than two different commands. At the same time, the tool was commented thoroughly and designed so that extensions could be added quite simply if another fls argument was needed for a specific project.

Another task to overcome, was tools calling commands discreetly, hiding the backend commands from the user. What good is a tool if you don’t know what it is actually doing on the backend? Instead, this tool attempts to be completely transparent to the user, going so far as to tell him the exact fls and mactime commands it is building and running (excerpt shown in figure 3 below).



Figure - ACTUAL commands

# Tool development

This tool was written in Perl because it is a powerful text manipulation language that is broadly understood, allowing users to modify the base program to their hearts’ content. This makes FLSmac an even more valuable tool.

The first step in developing the tool was deciding which fls and mactime features I (as author) felt were the most important. Then, a simple menu-driven system that met these needs was created. The resulting menu (figure 4) was created from those decided options.

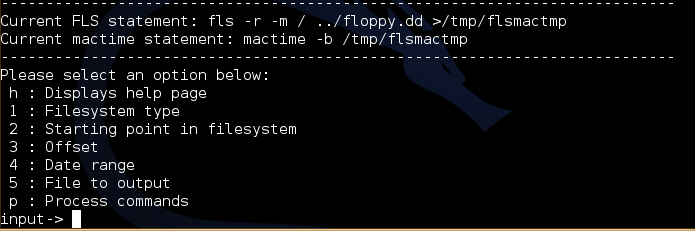


Figure - interactive menu

Next, the back-end was built in by receiving input from the user, then crawling the if/elsif statements to build the command execution statements appropriately. After each option is selected, the “Current” statements at the top of the display are dynamically updated.

Finally, user fault tolerance was added into the program. For starters, if the image filename input by the user doesn’t exist, the program will error and exit saying that the file doesn’t exist, asking the user to try again. Both the ‘p’ and ‘h’ commands accept both the capital and lowercase forms using basic pattern matching. Any yes or no questions, designated by (yes/no), are case insensitive and allow the full “yes” and “no” to be typed in by solely pattern matching the first character of the input. Input verification was also added to the program to make sure a user enters information in the format expected. For example, if a user enters the “Starting point in the file system” menu, but never enters anything (leaving a blank space in the input variable), it reinserts the default “/” referring to the root of the file system. Another example is the “Date range”: it asks for the start date and the end date in the format yyyy-mm-dd and has regex built in so that if the statement is built incorrectly, it does not add it to the command statement but instead states an error that the input was formatted incorrectly and requests that the user try again.

Thorough comments were also added during the creation process to ensure edit ability and readability for any user that wants to make modifications.

# Any particularly interesting issues encountered along the way

The only thing I could not seem to bypass was the need to actually output the results from fls to a file before they could be read into mactime. Regardless of the different methods attempted to skip the intermediary file, I could not get it to work. Methods tried:

1. Piped the input from fls to mactime
2. Called fls within back-ticks and nested it within the mactime command
3. Stored the fls output in an array then tried to pass it to mactime

I ended up having to create a temporary file, /tmp/flsmactmp, then delete it to transfer the output properly between the two programs.

# Conclusion

FLSmac is a user friendly timeline creation utility that is open source and modifiable to meet any timeline creation requirements an investigation may have. It is reliably built on the Sleuth Kit investigation package [1]. What are you waiting for? Go get it now at <http://sourceforge.net/projects/flsmac/>!

# Personal Conclusion

I found creating a utility that actually had real world value an extremely interesting project. Up until this point, all the programs I had written in college were solely to accomplish a personal need for an administration class or a \*problem\* in the lines of a project. It was interesting attempting to tackle an obstacle from the perspective of someone actually trying to solve a real world problem while making the tool robust and intuitive enough to make it user friendly. All in all, I really enjoyed this project.

# To Use

Installation  
Simply un-tar the folder to the location of your choosing (ex: tar -xvf flsmac.tar).  
  
Execution  
./flsmac.pl [image]

#### Usage

Navigate through the program selecting the appropriate number for the menu item requested to build your command.

When complete, hit ‘p’ to process the command.

#### OR visit <http://sourceforge.net/projects/flsmac/>

# Appendix

## Contents of flsmac.pl

#!/usr/bin/perl

#For support visit http://sourceforge.net/projects/flsmac/

#------------------

#Functions

#------------------

#function to print the interface menu

#takes the FLS command and the mactime command as input

sub mainMenu {

print "---------------------------------------------------------------------------\n";

print "Current FLS statement: $\_[0] \n";

print "Current mactime statement: $\_[1] \n";

print "---------------------------------------------------------------------------\n";

print "Please select an option below:\n";

print " h : Displays help page\n";

print " 1 : Filesystem type\n";

print " 2 : Starting point in filesystem\n";

print " 3 : Offset\n";

print " 4 : Date range\n";

print " 5 : File to output\n";

print " p : Process commands\ninput-> ";

chomp($input=<STDIN>);

return $input;

}

#------------------

#Variable default initialization

#------------------

($fileSysType,$selection,$fileSysType,$offset,$range,$start,$end,$outFile)='';

$startLoc='/ ';

#------------------

#Main Program

#------------------

#if an argument is passed, set it to the file name - else, query user for file name

#if -h argument is passed, display help contents then exit

#else query user for filename

#test if file name exists, if not end program and ask the user to try again

if($ARGV[0])

{

chomp($file=$ARGV[0]);

if($file =~ m/^-h$/) { system("cat flsmac.info | more"); exit; }

}

else { print "Enter file location and name: "; chomp($file=<STDIN>); }

unless(-e "$file") { print "The file \"$file\" does NOT exist!\nPlease run the program again with a valid file for input.\n"; exit; }

#call function to ouput current command strings

$selection=mainMenu("fls " . $fileSysType . $offset . "-r -m " . $startLoc . $file . " >/tmp/flsmactmp", "mactime -b /tmp/flsmactmp $range" . $outFile);

while($selection)

{

#if selection is h, display help contents

if($selection =~ m/[hH]/)

{

system("cat flsmac.info | more");

print "\nPlease press [enter/return] to continue.";

<STDIN>;

}

#if selection is 1, query user for file system type then build command

elsif($selection eq '1')

{

print "Please enter file system type: ";

chomp($fileSysType=<STDIN>);

unless($fileSysType eq '') { $fileSysType = "-f $fileSysType "; }

}

#if selection is 2, query user where in the file structure they want us to start then build command

elsif($selection eq '2')

{

print "Where would you like us to start? ";

chomp($startLoc=<STDIN>);

if($startLoc eq '') { $startLoc='/ '; }

else{ $startLoc="$startLoc "; }

}

#if selection is 3, query user for the starting sector then build the command

elsif($selection eq '3')

{

print "What is the starting sector? ";

chomp($offset=<STDIN>);

unless($offset eq '') { $offset = "-o $offset "; }

}

#if selection is 4, query user for the start & end dates, then test for valid input format

elsif($selection eq '4')

{

print "What is the starting date (yyyy-mm-dd)? ";

chomp($start=<STDIN>);

print "What is the ending date (yyyy-mm-dd)? ";

chomp($end=<STDIN>);

$range="$start..$end ";

unless($range =~ m/^\d{4}\-\d{2}\-\d{2}\.\.\d{4}\-\d{2}\-\d{2}\s$/)

{

print "Bad format entered! Please press [enter/return] to return to the main menu.";

<STDIN>;

$range='';

}

}

#if selection is 5, ask file name for output, if they also want it put to the screen, if they want it as a .csv

elsif($selection eq '5')

{

print "Enter the file name for output: ";

chomp($outFile=<STDIN>);

print "Would you still like the results output to the screen? (y/n) ";

chomp($screenOut=<STDIN>);

if($screenOut =~ m/[yY]/) { $outFile= "| tee $outFile"; }

else{ $outFile= ">$outFile"; }

print "Would you like to output this file as a .csv to be opened in a spreadsheet? (y/n) ";

chomp($csv=<STDIN>);

if($csv =~ m/[yY]/) { $outFile= "-d $outFile"; }

}

#if selection eq p, call and execute commands

#creates then deletes a temp file located at /tmp/flsmactmp

elsif($selection =~ m/[pP]/)

{

system("fls $fileSysType $offset -r -m $startLoc $file >/tmp/flsmactmp");

system("mactime -b /tmp/flsmactmp $range" . $outFile);

#cleanup temp file and exit

system("rm -f /tmp/flsmactmp");

exit;

}

else { print "You entered an invalid selection, press [enter/return] to try again."; <STDIN>; }

$selection=mainMenu("fls " . $fileSysType . $offset . "-r -m " . $startLoc . $file . " >/tmp/flsmactmp", "mactime -b /tmp/flsmactmp $range" . $outFile);

}

## Contents of flsmac.info

FLSmac

Usage

flsmac [-h] [image\_name]

Description

FLSmac provides a simple to use interface to integrate the most commonly

used FLS and mactime commands to create a forensics timeline from a disk image.

In addition, it shows you the command as it is being built so that you can

verify its accuracy AND learn how the commands were assembled.

Parameters

-h Display this help file

[image\_name] Send the file name as input image

Requirements

SleuthKit must be installed and properly aliased to allow calls from

the command line. The SleuthKit is available from:

http://www.sleuthkit.org/sleuthkit/download.php

Examples:

flsmac

Calls the command line user interface which will request you

enter the image file.

flsmac -h

Shows this help file.

flsmac image.dd

Calls the command line user interface sending the image as the

input file.

Limitations

- Has to create a temporary file to catch the output of FLS at

/tmp/flsmactmp which is deleted after mactime execution.

- Only includes what the author of flsmac deemed the

"most commonly" used commands - although more can be easily added.

For support visit http://sourceforge.net/projects/flsmac/

# References

1 - <http://www.sleuthkit.org/sleuthkit/>

2 - <http://www.cyberforensicstore.com/en/index.php>