Global Information Assurance Certification Paper

Automation of Report and Timeline-file based file and URL analysis

GIAC (GCFA) Gold Certification

Author: Florian Eichelberger, florian.eichelberger@cognosec.com
Advisor: Richard Carbone

Accepted: April 30th, 2014

Abstract

The objective of this paper is to describe the workings of a highly extensible, high-performance, automatic timeline report parsing tool. The proposed tool, Log:Mole, can process log2timeline CSV export files and mactime-based bodyfiles.

The tool can process these file types from both online and offline systems. This capability enables the tool to provide an automated, up-to-date mechanism for weeding out files that are "known good" and "known-bad. If the files that are referenced in the log2timeline report are accessible to Log:Mole, it is able to gather additional information on their nature by passing them to various analyzing modules and combining the results. Thus Log:Mole can provide additional information not available solely from the logfiles.

The proposed tool will greatly reduce the overwhelming amount of data requiring in-depth analysis and it runs atop both Windows and Linux. Moreover, it supports very large input-based report files while remaining high-performance making it suitable for large investigations. Finally, this paper examines use cases, examples and provides a conclusion and possible future work to be carried out including implementing data visualization and metadata extraction capabilities.

1. Objective

The objective of this paper is to provide a proposal for the tool's implementation, including a working proof-of-concept tool to be used as a means of automating the analysis of log2timeline and mactime bodyfiles.

2. Introduction

The proposed solution tries to lessen the burden of manually processing timeline-based logfiles and automating the classification of both files and URLs. What is termed "malware funneling" (Lee & Tilbury, 2013, p. 219) by SANS is implemented using log2timeline CSV reports and fls-based mactime bodyfiles. Log2timeline uses its own CSV-style log format for the artifacts collected on the machine. The older fls tool uses its own "mactime" log-based format for the gathered filesystem metadata. By parsing the logfiles and classifying the listed files, the tool presents the forensic investigator with additional information concerning possibly malicious files.

The tool can also be used to find open attack vectors related to various web services that might have be compromised even though they were considered secure by the end user. This can be accomplished by comparing the results of the available URL detection modules with the results of the file detection modules. Even though this has to be done manually it can be further automated in future releases to find these and other correlations.

There are both an increasing number of systems affected and amount of data to be investigated and analyzed during forensic incidents (Glisson & Tabona, 2011, p.2). A manually driven examination of timeline evidence typically requires a forensic investigator to look at, on an average Windows Server 2008 R2, 60,000-80,000 individual files, according to (Lee & Tilbury, 2013, p. 219).

However, a highly skilled investigator might use up to 10 or more different tools in order to reduce his analytic efforts. The problem with this, however, is the effort needed to process the data generated by these various stand-alone tools and interpreting their results. The benefit, however, is that a forensic investigator would only have to look at hundreds of files, instead of thousands to find potentially relevant evidence. Thus, there is a net benefit to forensic investigators if they automate various aspects of evidence analysis and processing using only one tool that presents a clear set of findings. Moreover, if the tool can present only the most relevant evidence and provide a reason for their selection and classification this would allow investigators to more accurately draw their own conclusions.

Florian Eichelberger, florian.eichelberger@cognosec.com

A good example is the automated collection of forensic timeline metadata analysis tools provided by the log2timeline suite of tools written by Kristinn Gudjonsson[1].

By collecting a wide variety of different time-based metadata automatically using one tool, error-prone repetitive manual tasks are avoided (Dhillon, 2009, p. 36). However, even this automatic solution only provides a means of collecting and normalizing a large amount of data, without providing any help in classifying it. This is a problem as more and more organizations fail to detect intrusions in a timely manner. The timeframe to conduct an investigation grows longer, thus more systems may become compromised, leading to an increased workload on the forensic investigator (Mandiant, 2013) (Trustwave, 2013).

The problem here, however, is that to speed up this process, enormous amounts of information need to be analyzed to find possible "smoking gun" evidence and possible infection vector.

The currently proposed solutions include running various scripts, stand-alone tools or importing large datasets into spreadsheet applications which have become unmanageable. Other solutions propose feeding the data and metadata into specialized big-data software solutions like Splunk to normalize the data and facilitate searching. These solutions leave the forensic investigator with possibly thousands of files and URLs to investigate[2] (Lee & Tilbury, 2013, p. 219) and are of no help in their classification.

3. Prior work

Currently, there are numerous tools available to collect artifacts including timestamps from file-systems, configuration information such as the Windows registry or various browser caches. Both commercial and free/open-source tools are available to collect these.

[1] https://code.google.com/p/log2timeline/, retrieved 26.02.2014.
[2] The Author's tests indicated approximately 60,000 unique files on a clean Windows Server 2008 R2 SP1.

The following is a short list of tools that generate timelines (as described by Bean & Carbone 2011) for various artifacts. As this paper is neither about artifact gathering nor timeline creation, this list is kept short, as tools that have been technologically superseded or are obsolete not listed. They include:

- EnCase , FTK
- Log2timeline
- The Sleuth Kit (ils, fls, mactime) , PTK, Autopsy, Fiwalk
- Zeitline
- DFF (Digital Forensic Framework)
- The Coroner's Toolkit (grave-robber, mactime)
- NFILabs Aftertime

Each of these tools has the capability to generate a timeline based on the information collected from a system. From the above list, log2timeline provides the most diverse assortment of artifact information collection currently available using an automated straightforward implementation. Upon further research, it appears that at the time of this writing there are no tools currently available to parse the overwhelming amount of information collected and generated by such tools. However, such a tool, if it existed, would provide increased information and knowledge based on the underlying nature of the artifacts and their metadata.

3.1. Motivation

Cognosec GmbH has chosen to undertake this project in the spirit of a quote from Edwin H. Land (Land, 1945, p. 81), scientist and inventor: "I believe quite simply that the small company of the future will be as much a research organization as it is a manufacturing company, and that this new company is the frontier for the next generation."

At Cognosec, an IT and Security startup, the word Research is an integral part of the strategic core of the company. Cognosec has decided to endeavor upon its first public

research project. There were frustrations with respect to the amount of effort required by the author to carry out the task of conducting in-depth forensic investigations of Windows servers under significant time constraints. The original objective was to automate some of those tasks but upon further study the author concluded that the effort required in coordinating multiple tools and their output could be better spent actually evaluating the results of a single tool. Thus, the tool has been designed to be not only flexible but to produce a quality set of defendable results with low levels of false positives.

The objective of this paper is to examine not only the author proposed tool in-depth but also to present additional ideas that could be taken up by fellow researchers.

The Lessons Learned was that in the world of growing storage, increased time and resource pressures, automation is the key to success, and this aptly applies to Security Projects too.

4. Log:Mole, the proposed timeline report processing tool

4.1. Availability

The author proposed tool, Log:Mole, is available upon request from the author. The reason the software is not being open-sourced is that proprietary technology has been included as well as certain data-classes developed during previous corporate projects.

4.2. Scope and Objective

The scope of this paper covers the design and implementation of the proposed tool which runs atop both Windows and Linux using a modular, extensible, high-performance approach. This approach includes using modules to identify potentially malicious files and URLs based on existing online systems and offline tools where the focus of the investigation is placed on Windows machines.

The tool is neither meant to replace existing forensic software nor is it meant to introduce a new timeline format or generate a timeline on its own. Instead, it is intended as a resource for additional timeline-based information.

Covering the analysis of all available information from a disk image or live system (e.g. registry keys, cookies, EXIF-information and others sources of timeline metadata) is outside the scope of this paper, although implementing additional modules is possible.

4.3. Requirements and Setup

Log:Mole achieves platform independence without sacrificing high-performance by using the .Net framework on Windows and the Mono framework on Linux. In order to use the software, the following requirements must be met:

- A system with at least 2 GB of RAM
- Windows XP SP2 or later and Windows version: .NET 4.0 or later
- Linux kernel version 2.6.x or higher and Mono 3.2.5 or later (needs to support .NET 4.0)

As most of the modules work directly with files referenced in the logfiles, those files should be made available by either manually mounting the physical device under investigation using a write-blocker or a disk/partition image file as read-only. Mounting the proper volume must be done prior to running Log:Mole

To mount the physical device under Linux, the following command, for example, can be used:

mount /dev/sdb1 /mnt/mount_point -o ro

To mount a disk image-based partition using the loopback device, two commands can be used:

1) *fdisk -lu example.img*

Which results in output similar to the following:

```
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes ...
Device Boot      Start         End      Blocks   Id  System
example.img   *        56     6400000     3199972+   c  W95 FAT32 (LBA)
```

The "Start" offset, 56 in this example, is multiplied by 512 (the sector size) which results in a value of 28672 to be used as the offset. Then, the following command is used to mount the disk image at the appropriate byte offset:

2) *mount -o loop,ro,offset=28672 example.img /mnt/mount_point*

Under Windows, third party software is required as this capability is not native to it. Free tools including FTK Imager (registration required)[3] and PassMark OSFMount[4], both of which support the read-only mounting of a various disk image formats.

4.4. Design and architecture

The tool has been designed to accept the aforementioned log formats as specified in the "Scope and objectives" (see Section 4.2) and process them using the subsequent workflow as shown in the following diagram:

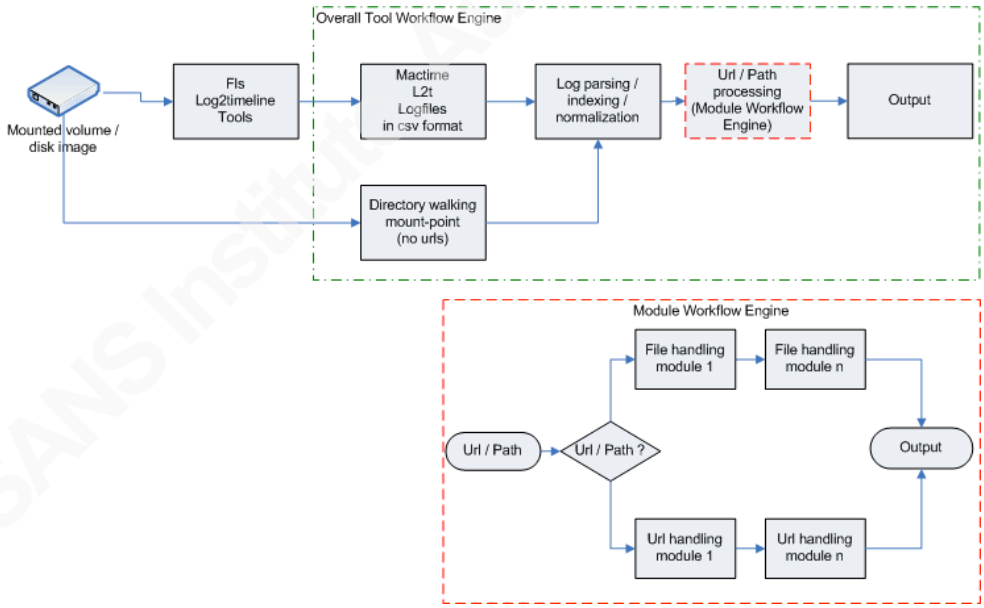

Figure 1. Overall process workflow

[3] http://marketing.accessdata.com/acton/formfd/4390/0206:d-0002, retrieved 11/03/2014.
[4] http://www.osforensics.com/tools/mount-disk-images.html, retrieved 11/03/2014.

4.4.1. Software components and application design

The proposed software consists of two executables and one configuration file. The executables are implemented as CLI (command-line-interface) applications named "LogMole32.exe for 32bit OS" and "LogMole.exe for 64bit OS". They were implemented as CLI applications to facilitate their integration into existing workflows or other scripting systems. For those requiring high performance or the ability to process large amounts of data (e.g. log2timeline reports with a size of > 600 MB per server), the CLI executable was developed in C# using the .NET framework with a highly optimized binary data structure. Depending on available resources, the timeline file to be analyzed can be very large. However, at the discretion of the investigator the timeline file can be broken into multiple pieces to facilitate batch or parallel processing.

Only one command line parameter is supported by the program, specifically the name of the configuration file to use. Using a custom format, it is a text file which contains the configuration information for one of the two workflow engines built into the tool. This workflow engine allows for the flexible configuration of the tool's inner workings including parameters such as files, options, programs to be executed, Application Programming Interface (API)[5] keys and others which can be readily adjusted to the user's needs. API keys in this context refer to a way of tracking the usage of a service as it is being used. The user has to sign up, in this case for free, to receive a personal and unique API key for VirusTotal that the user can use to authenticate against Log:Mole. In this way the user can be uniquely identified by VirusTotal without having to save the password in the configuration file, which helps to prevent credential abuse as the API key can only be used for certain actions offered by VirusTotal.

Each executed entity is called a "step" that can contain various commands and parameters. As shown in the example below, each "step" has to start with a "--Step = <Description>" line which is used solely for readability purposes and to help structure the configuration file. This step is then followed by a "_cmd=<cmd_name>" line, used to denote the module to be executed. Each module expects a certain number of parameters

[5] API (Application Programming Interface) specifies how software components interact with each one another.

to process the data in an expected way. New steps can be added to the configuration file in so long as existing "--Step" blocks are called in the order the user intends to execute them with the appropriate parameters provided. In this way, a simple yet configurable workflow can be easily created for the processing needs of the data to be fed into the system. Each step currently requires its own "--Step" line.

The current implementation implements the modules as classes that are compiled into a monolithic executable. As of this writing, the application does yet not support externally developed modules for seamless integration into the product. This will be fixed in a future version provided there is sufficient demand.

The following is a short snippet of a working configuration file for illustration purposes:

```
# Config
# this is a comment
Name = Config001

--Step = Read Log2Timeline
_Cmd = LoadLog
Filename = "Log\log2timeline.csv"

--Step = Check Data in Logfile
#Module name
_Cmd = LogCheck
#Parameters expected by the module
Mountpoint = "C:\Data\Test"
MalwareList = "Log\export.csv"
```

Command Line for AV Scanner, Mountpoint is appended to the call of the "AV" #executable

AV = "C:\Util\Portable\ClamWin\App\clamwin\bin\clamscan.exe --nocerts -o -i --no-summary -r "

See Appendix A for a full working example of the configuration file setup in order to be able to use all modules and parameters.

4.4.2. Data acquisition, processing and output

Currently, the tool provides three implementations for loading the required information necessary for additional processing. The main sources of information analyzed by the software are the log2timeline CSV file[6] and the mactime-based files as provided by the fls utility from TSK[7] (The Sleuth Kit). The third approach is a direct recursive scan of the mounted file-system which collects MD5 hashes. The log2timeline CSV output was chosen as it is a relatively simple format, well documented, and similar to the mactime format making it easy to parse and analyze. By default, Log:Mole does not produce any output. Instead, the output format must be specified in the configuration file. See the "SaveCSV" command in Section 7 of the configuration file.

4.4.2.1 Acquisition

The data used as the basis for processing is typically collected by making a bitcopy image of the suspect drive using either third-party Windows software like FTK Imager[8] or Linux tools including dd or it's much improved successor dc3dd[9]. In addition to making a bitcopy image of the original drive, a hash should be generated for both the original drive or device and the disk/device bitcopy image. After verifying that they match one another the disk image should be mounted read-only as previously discussed (see Section 4.3).

---

[6] http://code.google.com/p/log2timeline/wiki/l2t_csv, retrieved 03/02/2014.
[7] http://wiki.sleuthkit.org/index.php?title=Body_file, retrieved 03/02/2014.
[8] http://marketing.accessdata.com/acton/formfd/4390/0206:d-0002, retrieved 11/03/2014.
[9] http://sourceforge.net/projects/dc3dd/, retrieved 26/03/2014.

As for log2timeline, the process of generating the logfile is straightforward, as shown below. In general terms the log2timeline command, shown on the following page, recursively analyzes the specified directory, /mnt/usb_stick/, by using the -r parameter. To specify a target operating system an appropriate listfile must be specified, which is done using the -f parameter. The timezone is set to ensure correct timezone artifact conversion carried out using the -z parameter. The -c parameter requests log2timeline to calculate MD5 hashes and the -p parameter is required to enable preprocessing which can speed the whole process up. A full list of parameters and their meaning is provided on the log2timeline manual page[10].

The mount point to be used for file access required by each module needs to be specified in the Log:Mole configuration file (see Appendix A). The "-f" parameter corresponds to either a single log2timeline input module[11] or a log2timeline listfile as per the following command:

./log2timeline -f list

This command provides an output similar to this:

win7

chrome, evtx, exif, ff_bookmark, firefox3, iehistory, iis, mcafee, opera, oxml, pdf, prefetch, recycler, restore, sol, win_link, xpfirewall, wmiprov, ntuser, software, system, sam, mft, ff_cache, mcafeefireup, mcafeehel, mcafeehs, openvpn, skype_sql, security, symantec, firefox2, safari

By specifying the listfile name, all the modules contained in the listfile will be executed to collect artifacts. It is advisable to use the appropriate package name for the operating system under investigation. The output filename is specified using "-w" parameters.

A typical log2timline-based command which implements the various aforementioned parameters is as follows:

---

[10] http://log2timeline.net/man.html, retrieved 26/03/2014.

[11] These modules are responsible for extracting and collecting the artifacts from the mounted image under investigation.

./log2timeline -z UTC -p -c -r /mnt/usb_stick/ -w output_ntfs.csv -f winxp -c -x -log log.txt >/dev/null

To generate a mactime-based timeline, the fls command is used. Consider the following example:

./fls -F -u -m /mnt/usb_stick/ -r -z UTC -o 32 -p /opt/usb_stick_ntfs.dmg > bodyfile_ntfs_1.txt

In general terms the fls command recursively parses the filesystem within the image file as specified by the -p parameter. The timeline output of this tool is saved to ASCII-based text file bodyfile_ntfs1.txt. The -F parameters reduces the output to contain only files and the -u output ensures that only files accessible without further data-recovery are included in the output file. The -o parameter is crucial as this denotes the sector-offset where the filesystem is found to start within the disk image. To get the sector offset, proceed according to the example found Section 4.2. A full list of commands and their descriptions can be found on the webpage for the TSK suite.[12]

It is important is to ensure that the "-m" parameter represents the mount point where the disk image will be mounted on to provide access to the individual files. The string passed to this option should be carefully verified as it will be prepended to every file/directory entry in the output text file and thus will represent the path Log:Mole will use to access the files.

4.4.2.2 Log2timeline file data processing

The log2timeline format provides a wealth of information concerning the various artifacts collected from an investigated machine (Guðjónsson, 2010). This information is grouped based on the "source" CSV field. The values of the "source" CSV fields currently evaluated are "File" for files collected by the MFT log2timeline module and "WEBHIST" for URLs extracted from the local browser cache or history. In the event that the MD5 checksum is already provided this checksum is used, otherwise Log:Mole will calculate them. Even though the MD5 hash is currently considered cryptographically

[12] http://www.sleuthkit.org/sleuthkit/man/fls.html, retrievd 26/30/2014.

insecure (Joux, 2004) (Wang, Feng, Lai, & Yu, 2004) (Kaminsky, 2004), it is used because of wide industry acceptance (e.g. VirusTotal.com, National Software Reference Library (NSRL)). Using the l2t_process tool from the log2timeline package, it is possible to filter out noise and greatly reduce the amount of data to be analyzed by specifying a timeframe. Consider the following example:

l2t_process -b timeline_input.csv -w whitelist.txt -e MM-DD-YYYY

Or

l2t_process -b timeline_input.csv -w whitelist.txt -e MM-DD-YYYY..MM-DD-YYYY

Where MM-DD-YYYY signifies the starting point of the events kept until the specified date/time. Consider that MM-DD-YYYY..MM-DD-YYYY indicates the time range for which the events are to be kept. Finally, *whitelist.txt* should be an ASCII file containing keywords to be filtered from the results.

4.4.2.3 Mactime file data processing

As log2timeline does not provide a direct parsing module for FAT-based file-systems, the mactime output from the "fls" tool is used. From this output, the values of the second field, "name", are extracted. The fls tool does not provide CSV headers by default. Log:Mole uses the documented fls file header information currently hard-coded into Log:Mole to process files in mactime format. This works even without the CSV headers being present in the log files as long as the file format does not change and the order of the columns adheres to the documented format[13].

4.4.2.4 Direct processing

A third, quicker method of verifying that the evidence collected has been carried out correctly is by directly analyzing the read-only mounted disk image and iterating through its directory structure. The collected file-system information is then processed and used as input for other modules as if it would have originated from a logfile. This intermediate information is not saved, only the final output is.

---

[13] http://wiki.sleuthkit.org/index.php?title=Body_file, retrieved 03/02/2014.

4.4.2.5 Data Output

Log:Mole provides configurable output modules. The main output module saves the processed data to CSV format. Below is a select portion of the configuration file used to illustrate the usage of the main output module. Some important configuration options are related to the output of data. The amount of data that is outputted is configurable through two options. By default, with the FilterValue parameter commented out, Log:Mole outputs all lines from the original input file with two columns added. These two columns contain the classifications "results" and "reason" , both of which are succinctly named. At the time of this writing, the investigator can rename the column containing the result by specifying an appropriate value for the "FilterProperty" setting. This will be changed in a future version to allow the column containing the reason to be renamed as well. Consider the following sample configuration file:

```
_Cmd = SaveCSV

Seperator = ;

Encoding = ASCII

#Encoding = Unicode

#Default Encoding is UTF-8

#If the "FilterValue" parameters is commented out, the whole input file is

#output with the result added to a column named by the value of "FilterProperty".

# If FilterValue is set to "*", only lines process by a Log:Mole module are output.

FilterProperty = "checkresult"   (output csv field name)

FilterValue = "*"

Filename = C:\temp\test_out.csv
```

The FilterValue setting can be used to limit the output to contain only specific lines. For example, to filter the output to contain only lines referencing malicious files, the FilterValue setting should be set to "malicious". If the FilterProperty setting is set to "*" (without the quotes), the wildcard tells Log:Mole to output all the lines that have

been classified by the tool no matter how they were classified. This option excludes all lines from being written to the output that have not been processed by Log:Mole.

4.4.3. Analysis Workflow

The analysis workflow engine is similar to the configuration parsing engine described in Section 1.3.1. At the time of this writing, each module can be parameterized according to the user's needs through appropriate configuration file settings. However, these settings do not allow the order of the analysis steps to be changed. Changes to this order are planned for a future version of the program that will enable the full analysis of workflows from all modules. This will allow individual modules to be enabled or disabled making the program adjustable to the forensic investigator's needs and local constraints.

A module has been implemented that processes only the data from the logfile without trying to access the files from a mounted device or disk image directly in the event the image or disk is unavailable for mounting. However, this module is severely limited in its capability and should only be used a last-resort when the other modules do not work. Other modules require Internet availability although most modules require access to the disk image and the files contained therein in order to function correctly. See figures 2 and 3 for the different workflows currently implemented as proofs-of-concept in this tool.

Log:Mole provides the field values of "Good" for whitelisted files with no further indications of maliciousness, "malicious" for direct detection by a module and "suspicious" for suspicious files that might be a false positive but should be further examined by an expert. As for URLs, a score is added to the output report as based on the various characteristics of the URL or by a detection module (e.g. suspicious [2]). This specific workflow is shown in Figure 3 below.

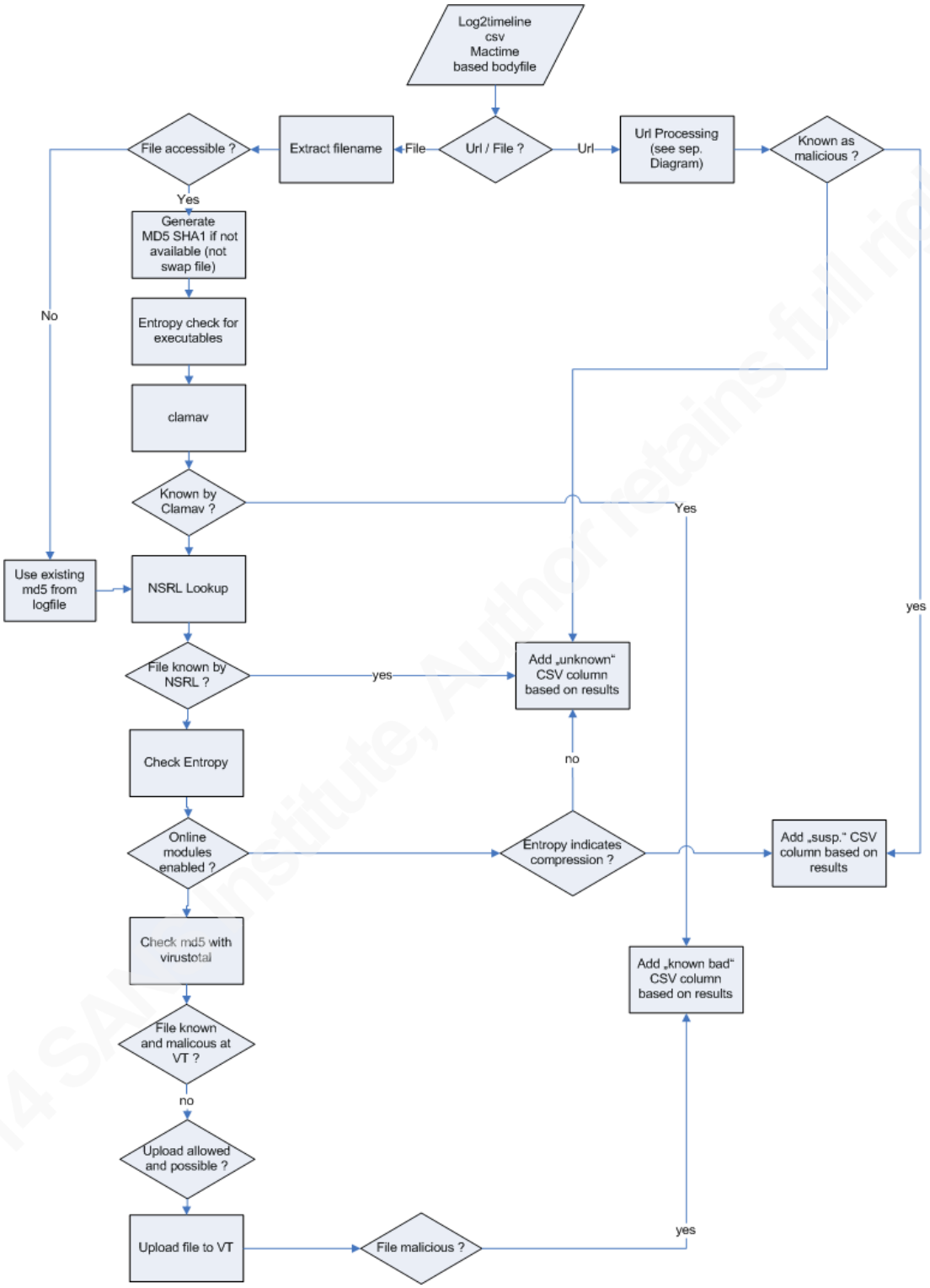

Figure 2. Proof-of-Concept File Workflow

Florian Eichelberger, florian.eichelberger@cognosec.com

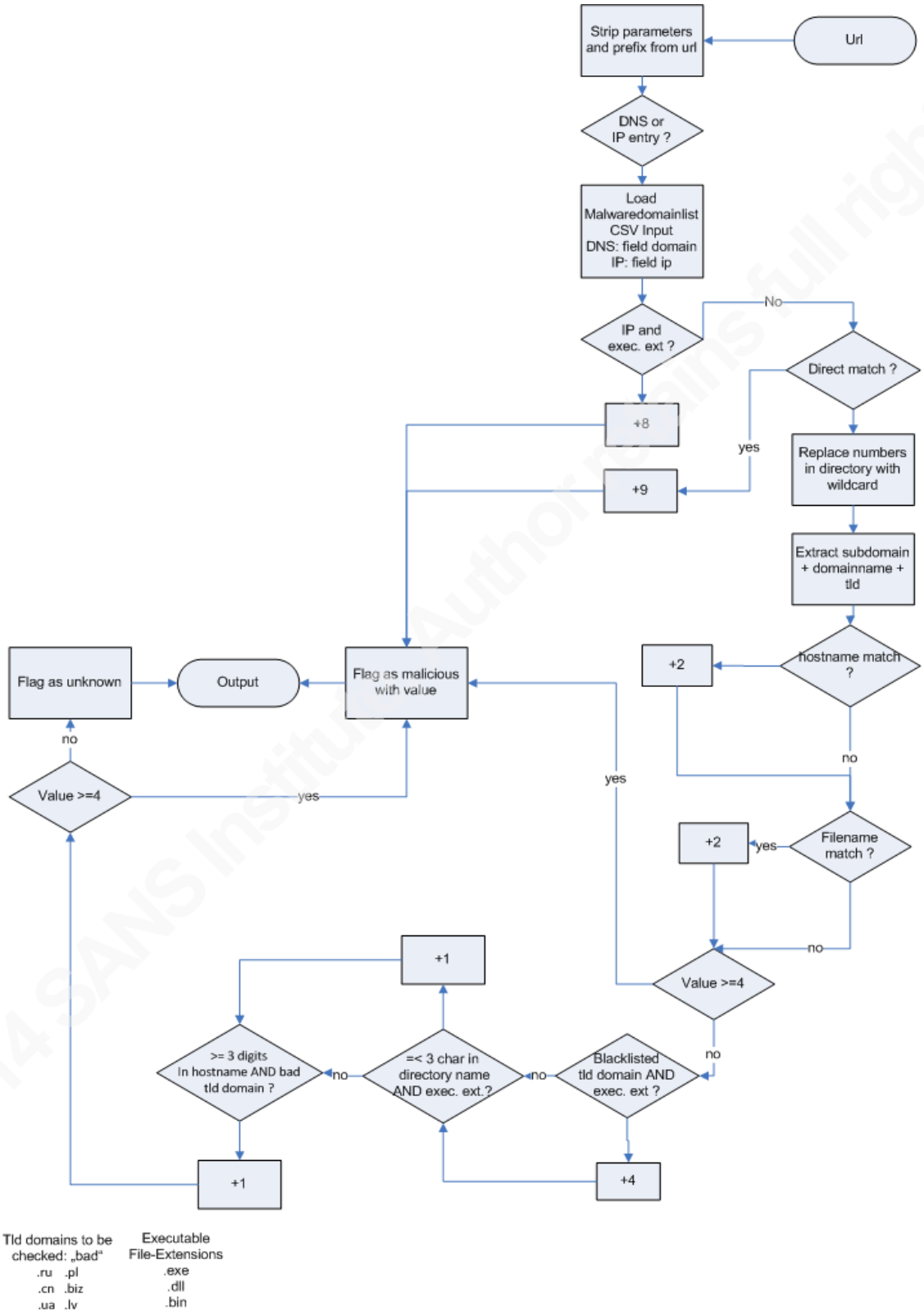

Figure 3. Proof-of-Concept URL Workflow

Florian Eichelberger, florian.eichelberger@cognosec.com

4.5. Analysis Modules

Log:Mole features an extensible approach which enables the checking of a file or URL by the aforementioned modules.

4.5.1. ClamAV

As the most widely used open-source virus scanner[14], ClamAV is readily available for a large number of systems and contains a virus database that is updated regularly[15]. This makes it an ideal choice for inclusion into automated tools. Moreover, due to the nature of Log:Mole, additional virus scanners can be easily added into the workflow without having to re-write any code.

ClamAV is called based on the file and path parameters extracted from the log files and the result is read back and parsed by Log:Mole. For performance reasons, the whole directory is scanned rather than individual files.

Clamscan, the command line ClamAV Scanner, is called with appropriate parameters to reduce additional output and to make parsing easier with the result read back into the tool and processed therein. For example, consider:

./clamscan --nocerts -o -i --no-summary -r /mountpoint/plaso/

Which results in the following sample output:

```
/mountpoint/plaso/somefile.exe: Eicar-Test-Signature FOUND
/mountpoint/plaso/eicar.com: Eicar-Test-Signature FOUND
```

4.5.2. NSRL Lookup

The average desktop or server computer contains around 80,000 files (Lee & Tilbury, 2013, p. 219). Tests by the author using a standard Windows server 2008 R2 SP1 machine resulted in around 60,000 files on the machine. By having to check all existing files, much time is wasted on files that are considered as "known-good" with only a few files potentially found as infected on a given machine. Moreover, 60-80% of

[14] http://www.clamav.net/lang/en/about/who-use-clamav/, retrieved 26/02/2014.
[15] https://github.com/vrtadmin/clamav-faq/blob/master/faq/faq-cvd.md, retrieved 26/02/2014.

the files remain the same on the installation media and the system they were installed to[16].

The NSRL RDS contains metadata on computer files that can be used to uniquely identify the files and their origins. For each file in the NSRL collection, the following data are published:

- Cryptographic hash values (MD5 and SHA-1) of the file's content. These uniquely identify the file even if, for example, it has been renamed.
- Data about the file's origin, including the software package(s) containing it and the manufacturer of the package.
- Other data about the file, including its original name and size.
- The list provided by the NSRL is a text-based ASCII formatted file and is downloadable using a single package in an ISO disk image format.

Investigators typically use the NSRL to weed out known-good files[17] [18]. It should be taken into account that the NSRL database not only contains "known-good" files but also possibly malicious files. According to the NSRL documentation, files can possible be flagged as malicious by using a special flag[19]. At the time of writing, this flag was not in use but based on the application type information provided in the NSRLProd.txt file, files could be possibly identified that are most likely related to "malicious" applications. The current implementation of Log:Mole takes this into account. The application types as listed in Appendix A.1 are flagged as "suspicious" by Log:Mole in the output to assist the investigator in deciding if the files identified are to be considered malicious.

To be able to use the NSRL database with Log:Mole, the file downloaded from the NIST website, typically NSRLFile.txt, needs to be converted before first use to the

[16] http://www.nsrl.nist.gov/Documents/yapc2004/index.html, retrieved 20/02/2014.
[17] http://www.nsrl.nist.gov/new.html, retrieved 20/02/2014.
[18] http://www.nsrl.nist.gov/Downloads.htm#isos, retrieved 20/02/2014.
[19] http://www.nsrl.nist.gov/Documents/Data-Formats-of-the-NSRL-Reference-Data-Set-16.pdf , retrieved 01/04/2014.

binary format used by Log:Mole. This conversion is triggered by the following configuration settings:

" KnownGoodCheckText = "C:\NSRLFile.txt" "

" KnownGoodCheck = "C:\ \NSRLFile.bin" "

If the first configuration option is not commented out, Log:Mole processes file NSRLFile.txt and generates the NSRLFile.bin file, which has been optimized for reading and searching as specified by the KnownGoodCheck option.

After initial conversion the first configuration setting should be commented out until the NSRLFile.txt is updated or modified. The current implementation of Log:Mole requires this text file to have Windows line endings (CR/LF endings). Files found in the NSRLFile.txt that are considered known-good and are not processed further by other modules.

4.5.3. Entropy

In 1948, Claude E. Shannon described entropy as the amount of randomness in an information stream, specifically the difficulty of predicting bytes in sequences or numbers in series (Shannon, 1948). The concept was named "entropy" by Shannon after Johann von Neumann, an Austrian mathematician, alerted him that the mathematical concepts of entropy in information theory and physics are the same (Eimann, 2008, p. 22). In this paper, the term "entropy" denotes "Shannon entropy", the function of the probability distribution of an information source (Eimann, 2008, p. 35). For simplicity's sake, a statistical and mathematical treatise of information entropy is not examined herein.

As compression algorithms remove redundancy the entropy of compressed information rises. In this case, a file is analyzed and the frequency of encountered byte values is recorded. Example implementations of this are bintropy (Hamrock & Lyda,

2007, p. 41-42) and PeiD[20] where bintropy and other systems use a grading between 0 for ordered and 8 for random data (Hamrock & Lyda, 2007, p. 41).

This approach has been used for some years to detect malware files, as malware tends to use run-time packers to hinder reverse engineering and analysis. According to literature and tests, 80-90 percent of today's malware is packed (Brosch & Morgenstern, 2006). Log:Mole features an implementation which calculates the entropy of all files that are identified as PE files or Linux ELF files.

The main limitation of this method is the fact that identified executables could contain large resource sections or significant portions of the file could contain unencrypted data, whereas the executable code itself could still remain encrypted. However, the resulting entropy value would not appropriately reflect this. This attempt could also be thwarted by adding non-random padding between the encrypted payload, therefore lowering the entropy value. (Hamrock & Lyda, 2007, p. 42). The current implementation featured by Log:Mole provides an entropy value based on the byte distribution in 256 byte blocks where at least half of the bytes need to be non-zero, an approach tested and verified by Lyda and Hamrock (Hamrock & Lyda, 2007, p. 42.) .

4.5.4. Malwaredomainlist

With many thousands of newly infected websites each month[21], services like Malwaredomainlist or Google safe-browsing provide the necessary information for classifying URLs found in the collected artifacts gathered by Log:Mole. The Malwaredomainlist module helps the forensics investigator sift through the various URLs a user visited including intentionally malicious webpages[22], drive-by-downloads and hacker attacks[23] that pose a great risk of malware infection. Even the top search results of

---

[20] Original page is no longer available, retrieved from http://www.heise.de/download/peid-1167913.html, retrieved 20/02/2014.
[21] https://blog.commtouch.com/cafe/miscellaneous/343927-new-malicious-sites-commtouch-security-number-of-the-month-for-november/, retrieved 24/02/2014.
[22] https://www.securelist.com/en/analysis/204792299/IT_Threat_Evolution_Q2_2013#24, retrieved 24/02/2014.
[23] http://www.computerworld.com/s/article/9238842/U.S._Department_of_Labor_website_infected_with_malware, retrieved 03/02/2014.

the largest search engines can contain malicious URLs a user could be tricked into surfing to[24]. By detecting those malicious URLs in the users surfing history or cache, additional evidence can be found and sometimes linked with the malware detected by other modules. The current proof-of-concept implementation implements the well-known Malwaredomainlist.com blacklist. Log:Mole uses the "WEBHIST" type of fields of the log2timeline report and extracts the URL from the "desc" field.

For example, consider the following line from a sample log2timeline CSV-based report:

> URL:http://badcorp.com/about_us.php cache stored in: Q75VRIK0/1360143754[1].php - HTTP/1.1 200 OK - - Content-Type: application/x-httpd-php- - ETag: "ffffffff992a5ec3-1f4b-4d50b254cf680" - - Content-Length: 8011 - - - - ~U:user1 -

Then consider the following line showing an example from the Malwaredomain.com CSV report:

> "2009/01/17_00:00","969696.ru/696969/ldr.exe","70.84.195.170","aa.c3.5446.static.theplanet.com","zeus v1 (non-RC4) trojan","abdulov@gmail.com","21844","1","1"

URL badcorp.com is then extracted without additional parameters passed to it as those are typically random or host-specific and therefore of no value when checking against blacklists. As the Malwaredomain.con blacklist is available as a CSV download, two fields named "domain" and "ip" contain either the full hostname or the IP if no DNS was used. Below is a table showing portions of a URL from the Malwaredomainlist CSV file with an explanation of the extracted information as used in Log:Mole or shown in Figure 3.

[24] http://www.av-test.org/fileadmin/pdf/avtest_2013-03_search_engines_malware_english.pdf , retrieved 03/02/2014.

Table 1: Example of URL portions of a malware-based URL from the Malwaredomainlist CSV report

CSV Item	Names used in Figure
969696.ru	Hostname
.ru	Top Level Domain
696969	Directory Name
Ldr.exe	Filename
.exe	Executable extension
696969/ldr.exe	Path

Besides receiving a direct hit in the blacklist, some "heuristics" are applied to grade the trustworthiness of a URL, to be able to identify potentially malicious unknown pages. Figure 3 shows the proof-of-concept URL heuristics.

The limitation of this method is obvious as thousands and thousands of new malicious pages are setup every month, yet by implementing various URL-scanning systems and heuristics a reasonable detection rate can be accomplished.

4.5.5. VirusTotal

Founded in 2004[25], VirusTotal is likely the largest free multi-scanner system on the market. At the time of writing, VirusTotal consisted of 39 different anti-virus systems used to analyze around 500.000 uploaded files per week, where around 260.000 are detected by at least one system[26]. By providing a public json-based API it is possible to automatically query the VirusTotal DB for MD5 / SHA1 / SHA256 hashes or to upload a file and have it checked[27].

Besides its file scanning capabilities, VirusTotal consists of several URL checking tools to provide malicious URL detection. Our implementation requires the user

[25] https://www.VirusTotal.com/en/about/team/, retrieved 24/02/2014.
[26] https://www.VirusTotal.com/en/statistics/, retrieved 06/03/2014.
[27] https://www.VirusTotal.com/en/documentation/, retrieved 24/02/2014.

to sign-up with VirusTotal to acquire a personal API key that can then be used for 4 queries or uploads per minute. Upon request this restriction can be lifted by VirusTotal[28].

By using the available .NET module for VirusTotal, a seamless integration into our tool is possible.

4.6. Usage example

In order to demonstrate the actual usage of Log:Mole, the "Stark Research Lab Intrusion" example as provided by SANS was used as a use case. As part of the exercise, the computer of a user called Tony Dungan is infected via the means of a social engineering attack and his computer used as a bridgehead. The system is imaged and the image file made available together with the SANS training material. For this example, a VMware instance running CentOS 6.0 Final was used.

4.6.1. Mounting the image

The ewftools package needs to be installed in order to be able to mount the provided EnCase-format images. If the SANS SIFT workstation is used, this is already installed. To Mount the EnCase (ewf) disk image file as read-only:

mount.ewf -o ro xp-tdungan-c-drive.E01 /mnt/evidence

To mount the disk image file contained within the EnCase image in order to show additional NTFS $MFT files, as read-only:

mount -o ro,loop,show_sys_files,streams_interface=Windows /mnt/evidence/xp-tdungan-c-drive /mnt/xp-tduncan-c/

4.6.2. Generation of the timeline

To generate the timeline execute the log2timeline tool using the command listed below. It uses the Windows XP listfile and recursively collects artifacts on the filesystem mounted and calculates md5 hashes of the files contained within the mounted filesystem. The name of the output file that needs to be put into the Log:Mole configuration is specified as output_timeline.csv. For example, consider the following command:

[28] https://www.VirusTotal.com/en/documentation/public-api/, retrieved 24/02/2014.

./log2timeline -z EST5EDT -p -r -c -f winxp -w output_timeline.csv -log output_timeline.log /mnt/xp-tduncan-c/

4.6.3. Analyzing the timeline

See Appendix A for a working and useable configuration file with explained settings. Adjust the configuration file settings copied from Appendix A and save them to a file called "config.txt".

Then execute Log:Mole by calling the executable directly with the configuration file as its only parameter.

Depending on the size of the imported log2timeline report and the performance of an investigator's system it may anywhere from a few minutes to several hours to process the information from the log2timeline logfile and generate the desired output.

The results of the Log:Mole analysis modules are written to two columns that are added to the CSV output report. Those columns are named "checkresult" in this example as specified by the FilterProperty value from the configuration file and "checkreason", a column that includes information about the reason of the classification. The name of the "checkreason" column is currently hardcoded but this will be changed in a future version.

Provided below is a short snippet of an output report file in log2timeline format (some fields have been omitted from the data and the provided CSV header for the sake of readability):

```
date;time;timezone;source;sourcetype;desc;extra;checkresult;checkreason

02.03.2012;15:42:30;EST5EDT;WEBHIST;Internet
Explorer;URL:ad.yieldmanager.com/;;suspicious [2];URLHeuristics

02.03.2012;15:40:11;EST5EDT;WEBHIST;Internet
Explorer;URL:http://dl5.iq6download.com/lm/cdn2/freefileviewer_2_1283.exe;;suspicio
us [2];URLHeuristics

03.04.2012;12:29:56;EST5EDT;WEBHIST;Firefox3 history;
http://199.73.28.114:443/HYvy.exe (HYvy.exe) [count: 0] Host: 199.73.28.114
visited from: http://199.73.28.114:443/ (URL not typed directly) type: DOWNLOAD;;
malicious [8];Executable_download_from_IP
```

An investigator can, if so desired, import the resulting report file from Log:Mole into a spreadsheet application and sort the information therein based on the checkresult column in order to quickly identify which information may be the most relevant to his case. The score provided after the classification can provide another indication for the maliciousness of a file. Based on certain criteria, points are added for each matching test done internally by Log:Mole.

At the time of this writing, the criteria used for the URL and file heuristics where hard-coded into the application. In this example, a score of 2 indicates one heuristic rule with an assigned score of 2 matched the URL or file, indicating that the suspicious artifact is likely a false positive. On the other hand, a score of 8 indicates that several rules with lower scores are matching the URL/file or one rule with a higher score is matching the URL/file and thus the file is more likely to be malicious. From the above snippet, it is likely that a file was downloaded directory from an IP address, in this example from 199.73.28.114, something commonly seen in malware downloaders. Then the investigator could look at the filename itself and conduct further research to confirm Log:Mole's classification.

4.7. Future Work

Due to time and resource constraints, not every possible module has been developed for this paper. The module workflow engine could be extended for "sub-workflows" (e.g. which tests should occur in which order for URL scanning for example). These workflow engines could be extended and used in all available modules to allow investigators to readily add new tests and tools. Additional URL-checking modules could be included to increase the detection rate of the ever-growing number of malicious URLs. Moreover, additional scanning or heuristics modules could be implemented to further increase the detection rate of unknown malware. One obvious module addition would an implementation of the Google safe browsing API [29] or an IOC tool as Redline[30] to assist in more readily identifying malware.

[29] https://developers.google.com/safe-browsing/developers_guide_v2, retrieved 12/03/2014.
[30] https://www.mandiant.com/resources/download/redline, retrieved 12/03/2014.

4.8. Methodical limitations

Even though the automatic analysis of the information contained in the logfiles provides significant timesavings and reduces the chance of human error it does contain some limitations. The main limitation is that it is impossible to detect every possible (even unknown) malware by the time a given analysis has completed[31]. The same applies for URLs as every day thousands and thousands of malicious URLs are detected[32] and can be setup easily by mischievous individuals or groups.

Some of the modules from the proposed tool require a stable Internet connection and access to VirusTotal.com. However, this comes at the expense of time as the analysis of a heavily used system may take considerable time even after eliminating all known-good files when leveraging online resources such as VirusTotal due to their load or usage restrictions. Moreover, the entropy method in this paper is prone to interpretation errors based on the nature of the files processed and does not allow a precise conclusion to be drawn.

5. Conclusion

The automation of timeline report processing through a module driven approach provides many benefits. These include an easy and configurable manner of analyzing two of the most commonly used timeline log-based formats, log2timeline and mactime. Log:Mole can help speed up the process of malware funneling, assist in detecting the smoking gun, especially in a case where many URLs and files require additional verification.

Even though the up-front effort for the development of this prototype was high, the implementation of workflow engines and the modular system used therein can be considered as state-of-the-art. Moreover, the prototype's architecture can save significant development time throughout its life cycle. This is particularly true should additional

[31] See https://www.VirusTotal.com/en/statistics/, retrieved 06/03/2014.
[32] See http://www.google.com/transparencyreport/safebrowsing/?hl=en, retrieved 11/04/2014.

Florian Eichelberger, florian.eichelberger@cognosec.com

modules be included to further analyze information within the logfiles (e.g. registry keys) or to support more logfile formats.

As more and more systems with an ever growing amount of data are analyzed, automation or semi-automation-based anomaly detection and file classification systems such as Log:Mole will be necessary to keep up with the increasing number of files and URLs found on each system under investigation. At this time, Log:Mole is the only tool known to provide additional information value based on the underlying nature of the analyzed information in order to reduce the "time-to-evidence", the timespan from generating the timeline to either the conclusion that a file is malicious or the conclusion that further investigation is necessary.

The flexibility and performance of the optimized .NET code, in conjunction with optimized indexing, parsing and normalization leads to a multi-platform tool capable of processing large reports without excessively high memory consumption and is resistant configuration and data errors. The examiner is provided with an output report containing the information analyzed and a conclusion and in addition the data the conclusion is based on, to allow the investigator to make own judgments. This information is per default provided as a CSV file in the log2timeline format.

The ease of use of Log:Mole also makes this a feasible option for examiners that have no deep experience with malware detection or malware analysis.

Florian Eichelberger, florian.eichelberger@cognosec.com

6. References

Bean, C., & Carbone, R. Defence R&D Canada - Valcartier, (2011). Generating computer forensic super-timelines under Linux: A comprehensive guide for Windows-based disk images (TM 2011-216)

Brosch, T., & Morgenstern, M. (2006, 08). Runtime packers: The hidden problem ?. Black Hat USA, Las Vegas. Retrieved February 13, 2014, from http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Morgenstern.pdf

Dhillon, B. S. (2009). Human reliability, error, and human factors in engineering maintenance: with reference to aviation and power generation. Boca Raton: CRC Press. 36.

Eimann, R. (2008). Network event detection with entropy measures. (Doctoral dissertation)Retrieved from https://researchspace.auckland.ac.nz/handle/2292/3427

Glisson, W.B., & Tabona A.Z. (2011, 07). Exploring solutions put forth to solve computer forensic investigations of large storage media. Proceedings of the Sixth International Workshop on digital forensics & incident analysis (WDFIA 201) (pp. 1-16).

Guðjónsson, K. (2010). Mastering the Super Timeline With log2timeline. Retrieved February 15, 2014, from http://www.sans.org/reading-room/whitepapers/logging/mastering-super-timeline-log2timeline-33438

Hamrock, J., & Lyda, R. (2007). Using entropy analysis to find encrypted and packed malware. Using Entropy Analysis to Find Encrypted and Packed Malware, 5(2), 40-45. doi: 10.1109/MSP.2007.48

Joux, A. (2004, 08). Multicollisions in iterated hash functions. application to cascaded constructions.. 24th annual international cryptology conference, Santa Barbara. doi: 10.1007/978-3-540-28628-8_19.

Kaminsky, D. (2004). MD5 To Be Considered Harmful Someday. IACR Cryptology ePrint Archive, 2004, 357.

Florian Eichelberger, florian.eichelberger@cognosec.com

Land, E.H., (1945). Research by the business itself. The future of Industrial Research Papers and Discussions. New York: Standard Oil Development Company. (pp. 81-86).

Lee, R., & Tilbury, C. (2013). Forensics 508 Advanced computer forensic analysis and incident response.(Vol. 1, p. 219).

Mandiant (2013). 2013 Threat report. Retrieved February 12, 2014, from http://www.mandiant.com/library/M-Trends_2013.pdf .

Ponemon Institute (2013). The post breach boom. Retrieved February 13, 2014, from http://www.ponemon.org/local/upload/file/Post%20Breach%20Boom%20FINAL%201.pdf

Shannon, C. E. (1948), A Mathematical Theory of Communication. Bell System Technical Journal, 27: 379-423. doi: 10.1002/j.1538-7305.1948.tb01338.x

Trustwave (2013). 2013 Global Security Report. Retrieved February 12, 2014, from http://www2.trustwave.com/rs/trustwave/images/2013-Global-Security-Report.pdf

Wang, X., Feng, D., Lai, X., & Yu, H. Collisions for hash functions md4, md5, haval-128 and ripemd. IACR Cryptology ePrint Archive, 2004.

Florian Eichelberger, florian.eichelberger@cognosec.com

7. APPENDIX A

Appendix A provides a working configuration file with explanations. This configuration information can be copied to a configuration file (e.g. config.txt) and used with Log:Mole.

```
# denotes a commentary field that is ignored by the parser.
# invalid / unknown parameters are ignored
# paths can be absolute or relative , based on the path of the executable.
#Module names are case sensitive


# The "Name" settings has to be a unique identifier for this logfile
# Configuration file header
Name = Config001
using System.Globalization;
#
# Data acqusition step
#
--Step = Read Log2Timeline
#Enable the LoadLog module and set the Filename parameter to parse the
# specified log2timeline logfile. Specify either LoadLog or SearchDirectory.
#_Cmd = LoadLog
# SearchDirectory tells Log:Mole to iterate through the directory specified in the
# Directory directive.
_Cmd = SearchDirectory
```

Florian Eichelberger, florian.eichelberger@cognosec.com

```
# Uncomment the following line if the logfile you want to parse is in mactime
#if commented out, log2timeline format is assumed, otherwise set it to mactime
#Format=mactime
# Add the csv header based on Format setting
#AddHeader=1
# Directory is used if _Cmd=SearchDirectory and the Directory Walker should be
# used in case no logfile is available or a quick analysis of the mounted evidence
# is intended.
Directory = "C:\"
# If the "_Cmd = LoadLog" line is uncommented, specify the log2timeline or
# mactime format report file in the Filename setting (either with full or relative
# path)
#
#Filename = Log\xp-tdungan-c-drive_bodyfile.txt
#Exclude those columns from being read into Log:Mole while parsing the file
ExcludeColumn = "date"
ExcludeColumn = "time"
ExcludeColumn = "timezone"
ExcludeColumn = "MACB"
ExcludeColumn = "sourcetype"
ExcludeColumn = "type"
ExcludeColumn = "user"
ExcludeColumn = "host"
ExcludeColumn = "short"
```

```
ExcludeColumn = "version"
ExcludeColumn = "inode"
ExcludeColumn = "notes"
ExcludeColumn = "format"
# all the config settings includig ExcludeColumn are related to the LoadLog
# module
#
# Data processing step
#
--Step = Check Data in Logfile
_Cmd = LogCheck
# mountpoint where the AV scanner should look for mounted evidence
Mountpoint = "C:\"
#malwaredomainlist.com downloaded csv list.
MalwareList = "C:\temp\export.csv"
# Command Line for AV Scanner, Mountpoint is appended
AV = "C:\tools\Portable\ClamWin\App\clamwin\bin\clamscan.exe --nocerts -o -i
--no-summary -r "
# Convert NSRLFile with MD5 checksum to binary format - only needed once
#KnownGoodCheckText = "Z:\unique\NSRLFile.txt"
#Binary outputfile used by Log:Mole
KnownGoodCheck = "Z:\unique\NSRLFile.bin"
URLCheckHeuristics = 1
FileExtensionExe = ".exe"
```

Florian Eichelberger, florian.eichelberger@cognosec.com

```
FileExtensionExe = ".dll"

FileExtensionExe = ".bin"

BlacklistedTLD = ".ru"

BlacklistedTLD = ".cn"

BlacklistedTLD = ".ua"

BlacklistedTLD = ".pl"

BlacklistedTLD = ".biz"

BlacklistedTLD = ".lv"

EntropyThreshold = 6.8

#

# Data Ouput step

#

--Step = Save all Data CSV

_Cmd = SaveCSV

Seperator = ;

Encoding = ASCII

#Encoding = Unicode

#Default Encoding is UTF-8

# If this setting is not commented out ,only the corresponding log lines are output
# if the Value of the FilterProperty field is with “FilterValue”  (* means

# string length>0 )

#Modules may write various information to this field (to output all findings, use
#*)

#If those two “Filter*” parameters are commented out, the whole input file is

#output with two result columns added.
```

Florian Eichelberger, florian.eichelberger@cognosec.com

FilterProperty = "checkresult" (output csv field name)

FilterValue = "*"

Filename = C:\temp\test_out.csv

#

Various modules used for internal debugging only, they might be changed

without further notice.

The following two modules are for debugging only

#--Step = Save all Data to Text file (readable form of binary format)

#_Cmd = SaveDataText

#Filename = C:\temp\test_out.txt

#--Step = Statistics on Data

#_Cmd = Statistics

#Filename = Log\output_ntfs_new_stats.txt

#Filename = Log\mactime_ntfs_new_c_stats.txt

#Filename = Log\output_dir_stats.txt

#AppendDate = 1

7.1. APPENDIX A.1

Application types considered suspicious as found in the NSRLProd.txt in column "ApplicationType" include the following "products":

Hacker Tool

Data security

employee monitoring

Encryption

Encryption, Security, identity protection

Internet Security, Keyboard Logger

Internet Security,privacy tool

Keyboard Logger

Network Scanner related

Parental Control

password recovery

privacy tool

remote access

remote monitoring

Security tool

Security, privacy tool

Spyware

Steganography

Florian Eichelberger, florian.eichelberger@cognosec.com