

Forensic Triage Kit

Bachelor Thesis

Department of Computer Science

University of Applied Sciences Rapperswil

Spring Term 2016

Authors:	Luca Tännler, Mathias Vetsch
Advisor:	Cyrill Brunschwiler
Project Partner:	Compass Security Schweiz AG
External Co-Examiner:	Dr. Benjamin Fehrensén
Internal Co-Examiner:	Dr. Daniel Keller

Contents

1. Abstract	5
2. Management Summary	6
2.1. Introduction	6
2.2. Approach / Technologies	6
2.3. Results	6
2.4. Outlook	7
I. Technical report	8
3. Introduction	9
3.1. Problem statement	9
3.2. Project	9
4. Standard Forensic Procedure	10
4.1. Introduction	10
4.2. Forensic procedure	11
4.3. Context	12
5. Analysis of existing forensic tool kits	13
5.1. Triage techniques	13
5.2. Evaluation of the framework	15
5.3. Conclusion	18
6. Requirements	19
6.1. Introduction	19
6.2. Uses Case Diagram	19
6.3. Actors	20
6.4. Functional Requirements	20
6.5. Non-Functional Requirements	23
6.6. Conclusion	25
7. Autopsy Architecture	26
7.1. System context	26
7.2. Architectural Targets and Decisions	27
7.3. Logical Architecture	28

7.4. Specific Procedures and Decisions	31
7.5. Libraries / Frameworks	39
7.6. Deployment	40
7.7. Data Management	41
7.8. Collaboration between Autopsy and Sleuth Kit	41
8. Results	43
8.1. Introduction	43
8.2. Bitlocker Decryption Provider	44
8.3. Automatic HashSet update	46
8.4. Virustotal Online Checker	46
8.5. Golden Image Module	48
8.6. AuthentiCode verification	53
8.7. Tag Filter Module	57
8.8. Repositories	62
8.9. Conclusion	63
9. Benchmarking	64
9.1. Benchmarking setup	64
9.2. AuthentiCode on raw Windows 8	65
9.3. GoldenImage Reboot check	66
9.4. HashSet check	67
9.5. Bitlocker Efficiency	68
10. Conclusion	69
10.1. Assessment of the results	69
10.2. Outlook	69
II. Appendices	71
A. Aufgabenstellung	72
A.1. Einführung	72
A.2. Aufgabe	72
A.3. Vorgehen	73
A.4. Randbedingungen	73
A.5. Infrastruktur	74
A.6. Erwartete Resultate	74
A.7. Termine	75
A.8. Betreuung	76
A.9. Referenzen	76
A.10. Unterschriften	77
B. Golden Image Documents	78
B.1. Module: Golden Image - Options	78

B.2. Module: Golden Image - Planning	82
C. Installation Guide	91
C.1. Plugin installation	91
D. User Guide	92
D.1. Add a Bitlocker Data Source to Autopsy	92
D.2. Update Hash Sets	94
D.3. Verify AuthentiCode Signatures	97
D.4. VirusTotal Online Checker	99
D.5. Golden Image	100
D.6. Tag Filter	101
E. Analysis of Forensic Tool-kits	105
E.1. Open Source Forensic Frameworks	105
E.2. Tools and Scripts	110
E.3. White- and Blacklists	111
F. Glossar	113
G. Bibliography	114

1. Abstract

The forensic analysis of Windows systems is usually extremely time consuming. Therefore, in computer forensics, it is important to automatically mark known files whenever possible. This automated process is called forensic triage. The base for forensic triage is a framework, that starts different triage techniques and aggregates all relevant results.

The aim of this project was to create a solution for such a forensic triage kit with a set of standard triage technique features.

In the evaluation phase, different analysis techniques and frameworks were examined. This lead to the use of the Autopsy project as a basis. Autopsy is an open source digital forensics platform already containing a set of forensic triage features.

The implementation was done in a way of contribution to Autopsy itself and the development of several modules. For example, a module performs a check against an uninfected copy of a system, while another module verifies code signing certificates.

The result of the project is a significant improvement of efficiency in the analysis of Windows images when using Autopsy. A huge part of standard Windows images can be automatically marked as known-good with minimal user interaction.

2. Management Summary

2.1. Introduction

Nowadays, it is a daily routine of a digital forensics analyst or a Computer Incident Response Team (CIRT) to analyse potentially infected workstations or server systems. The main task of a digital forensics analyst is the investigation of hard disks from such systems and to find out, if they contain any malware, spyware or other indicators of compromise. In most cases, this kind of triage is very time consuming. The main goal of this project is the automation of as many triage steps as possible and merge out known good data and deliver hints on possible malicious data. This way, the analyst can put his focus on relevant data and save a lot of valuable time.

2.2. Approach / Technologies

In the first step, existing triage techniques and existing open source forensic triage projects were evaluated. Based on the initial research, a list was created, with possible features which could be implemented within this project. The list contained the most important features which would optimize a forensic triage process. A decision was made, to build upon the existing open source project Autopsy. It provides a variety of fundamental features which were required to achieve the projects goal. Also, the project has a modular architecture, which makes it easy to implement additional features.

2.3. Results

The results of the development phase helped, to significantly improve the Autopsy platform.

The Bitlocker module allows adding volumes from encrypted Windows workstations into Autopsy. The module is designed in a way, which makes it simple, to implement further disk encryption providers.

The following modules were developed to automate detection of known-good and known-bad files.

- The Golden Image module excludes known-good files from a reference data source and gives hints on files which were changed or deleted on the suspicious data source.
- The AuthenticCode verification module allows to identify the publisher of a software through code signing certificates.
- The hash-database update feature automates the import of white- and blacklists from the internet.
- The VirusTotal online check module performs a hash lookup to a free online database.

The above listed modules mark files on the evidence image with tags. To visualise the results, the Tag Filter module was implemented. This extension provides a generic filter for the tagged files. It allows the analyst to focus his work on interesting content.

2.4. Outlook

The results of this project enables for an automated analysis of Windows systems. But, the modules still have potential to be developed further. Some ideas for further improvements on the developed modules are documented in this report.

To bring the forensic analysis with Autopsy to the next level, further modules are required, which look into unknown files for bad indicators. For example, there is a software which scans for viruses or observes the behaviour of files within a sandbox. It would be a powerful and useful extension for Autopsy, if it would be embedded as a module.

Another approach to reach this target is the development of a scanner which uses multiple indicators of compromise to find malicious software. Indicators of compromise can be represented by descriptive languages which can characterize malware much better than a file hash. An intermediate scanner can improve the identification of more advanced and dynamic threats.

Part I.

Technical report

3. Introduction

3.1. Problem statement

In a computer forensic case the analyst has a nearly unmanageable amount of data of the affected system to be investigated. There are many tools for the analysis of data. The problem is, these tools are being used separately and all of them have their own specific output. It can be very time consuming for the analyst to run all these tools and aggregate the results. There are plenty of standard methods for this triage. One possibility is to compare the data against a non-infected system. The analyst then only has to focus on the delta of these two systems. Another possibility is to use white- and blacklists to automatically exclude known files.

The aim of the project is to develop a framework for this triage. The framework should be extensible and already provide a set of standard triage functions. The main goal is to automate as many steps of the triage as possible. In the end, the analyst should get a dramatically reduced data set which he then can analyse further. More details about the assignment are attached in Appendix-A.

3.2. Project

The project's initial assignment was to do research on triage procedures. A part of this assignment was to find existing open source forensic tools and tool kits. This research also provided some information about triage methods. An additional research was conducted on how standard forensic procedures are done and what steps they contain.

The implementation is built upon the open source forensic platform Autopsy. Autopsy provides a framework for forensic triage and comes with a set of triage features. Furthermore, it provides a modular architecture which makes it easy to add additional forensic features to the tool kit. The results of the project are several triage techniques implemented as an Autopsy module. This report contains sections with research that is relevant for this project, such as the evaluation of the framework, the implementation of the features and an overview about the efficiency of these features.

4. Standard Forensic Procedure

4.1. Introduction

The goal of this section is to give a short introduction into the procedure of digital forensics. What is digital forensics? In digital forensics, the objective is to get insight on digital incidents. It deals with the analysis of data which could bring up a suspicion. Such examinations are usually made after an incident occurred or if there is a suspicion of an incident. In order to classify an incident, the so called CERT-Taxonomy is used. This taxonomy classifies the incident in three parts: Incident, attack and event.

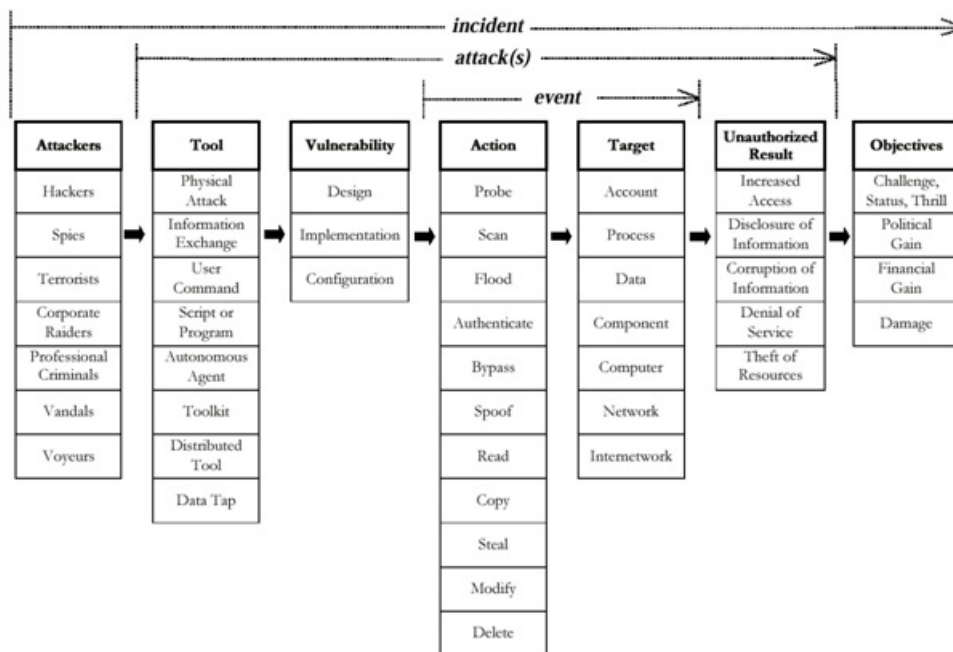


Figure 4.1.: CERT-Taxonomy, Source: [18]

Figure 4.1 illustrates that an IT forensic incident only exists, if there is an attacker with an intention. In between there is the event which contains an action with a target. The CERT-Taxonomy serves as a basis for the documentation.

4.2. Forensic procedure

Basically a forensic procedure is split up in three main parts:

1. Secure Data
2. Analysis
3. Documentation

4.2.1. Secure Data

In this step, the goal is to save the current state of the machine, more specific, secure all relevant data. This leads to the important question: Is it necessary to shut down the IT-System due to the hazard situation? If not, it might be necessary to save volatile data such as RAM, network connections or open files. While securing the hard disk, it is very common to use a write blocker. A write blocker is a physical device which prevents write operations to a hard disk and ensures the integrity of the data.

4.2.2. Analysis

After all relevant data is saved, a first analysis will be performed. The main goal is to investigate the data and find out, if a suspicion can be confirmed or disproved. Usually this analysis is not done on the original system, that is why the analyst has to look very carefully not to miss anything relevant.

4.2.3. Documentation

In this phase, the analyst documents the results from the analysis and their conclusion. Usually this report is very detailed and points out to all suspicious things. Often, the documentation also contains some of the following information:

- Identity of the offender
- Time frame of the act
- Scope of the act
- Motivation of the act
- Reason of the act

Although, in many cases a lot or sometimes even none of the above listed information can be provided.

4.3. Context

This chapter gave an overview of how a forensic procedure is done. This project only deals with the two last phases "Analysis" and "Documentation". The following chapter points to existing analysis methods, tools and tool kits.

5. Analysis of existing forensic tool kits

To identify possible triage techniques, multiple open source forensic tool kits were examined. Also, this step allowed to evaluate a possible framework to build upon. The characteristics of the examined tool kits are documented in Appendix-E.

5.1. Triage techniques

This section describes different techniques to identify known-good or known-bad files on a system.

5.1.1. Hash look-up

A Windows system consists of a lot of files which exist on every Windows system. These files are not interesting for the forensic analysis. It is possible to tag these files as known-good using a set of known hashes. Hashes are cryptographic fingerprints of files. This fingerprint can be used to compare a file on an evidence with another file fingerprint to prove their equality.

This method requires a hash set. The following hash sets can be used for forensic triage.

- NIST National Software Reference Library (NSRL). Hash database with known-good hashes of Windows operating system and user-software.
<http://www.nsrl.nist.gov/>
- VirusShare
Downloadable hash database of known-bad files.
<https://virusshare.com/>
- ClamAV Database
Downloadable hash database of known-bad files
<https://www.clamav.net/>

Online hash lookup

The hash sets do not necessarily have to be downloaded. A hash set can be provided over the network through a lookup API. This allows a software to send in hashes of files and get a response about the status of the specific files.

Virustotal [24] offers free but limited access to a hash database through their web service. Another free online service with limited access is the Kaspersky Whitelist project. [23]

5.1.2. Golden Image

Some forensic cases can provide a reference image of the system before the infection occurred. All files of the reference image can be considered as good. This method can dramatically reduce the amount of files to analyse.

5.1.3. Code Signing Verification

Microsoft recommends to sign software components with their code signing standard Authenticode. From a forensic viewpoint, this signatures can be used to authenticate the publisher of a file. The identity of a publisher does not directly help to decide if a file is good or bad. The analyst has to decide manually which publisher he trusts. Files from these trusted publishers can be marked as known-good.

5.1.4. Descriptive Languages for indicators of compromise

Advanced malware is often very dynamic. Their file signatures change fast over time or per infection. Such malware can not be identified using a file hash.

OpenIOC [15] is a standard to describe a malware on a abstract level. A description usually contains network targets, file names, process names and much more.

For a forensic triage we need two components. A database of OpenIOC descriptions and a scanner which understands the descriptions and is able to find out if the system is infected by such a description.

The OpenIOC project has launched the website <https://www.iocbucket.com/>. This is a database of OpenIOC files.

5.2. Evaluation of the framework

Among the examined software only one framework is open-source, actively developed and has a modular architecture. This Java based framework is called the Autopsy forensic platform.

The second option is to start a framework from scratch. This section compares these two options.

5.2.1. Autopsy Forensic Platform

Autopsy is a forensic tool kit providing a graphical user interface, infrastructure to install and run modules for forensic analysis. The framework provides functionality to add system images and run existing analysis modules. Autopsy strongly depends on the Sleuth Kit framework [13].

The Sleuth Kit

The Sleuth Kit is a standalone software for forensic analysis. It provides a command line interface consisting of several different binaries. The Sleuth Kit workflow has three phases shown in figure 5.1. In the File Extraction phase all meta information is collected and stored in a database. In the File Analysis phase modules are executed and their results are stored in the case database. The Post-Processing phase provides modules which use the results from the case database.

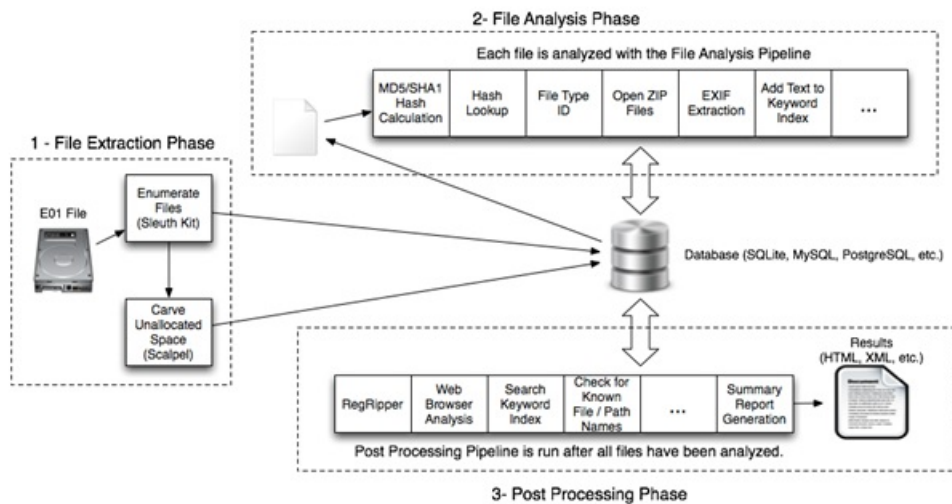


Figure 5.1.: Sleuth Kit analysis phases, Source: [13]

Autopsy extension points

Autopsy provides the following four extension points to develop third-party modules [6].

Ingest Module[9] Ingest modules provide an extension point for custom analysis techniques. The modules can be executed when the data source is added or at any point after the import. An Ingest Module can be one of the two following specialisations:

1. **File Ingest Module:** the module is being called for every file on the data source. This is mainly used, if you want to process all files on a data source.
(For example hash calculation, Hash lookup and file-type identification)
2. **Datasource Ingest Module:** The module is passed a reference to the data source. This type can be used when only specific parts of a data source are interesting or the analysis requires multiple files for a result
(For example Web artifact analysis, search for file types and more)

Report Module[10] These kind of module can be used for post-analysis but mainly for generating a report.

Content Viewers[5] To visualise the results of an analysis module for a specific file a content viewer module can be implemented.

Result Viewers[11] Result viewers provide a table view of a specific subset of the analysed data.

Overview

Autopsy represents a base for a forensic triage kit. The following characteristics need to be considered :

Advantages

- Provides basic functionality for forensic analysis
- Installable third party modules extend functionality [4]
- Extensible through the provided extension points
- Provides a graphical user interface
- Project is actively developed

Disadvantages

- No support or extension point for encrypted data sources
- The framework is developed to run on Windows. But the used technology allows to also run the framework on Linux.
- Complicated architecture: multiple programming languages are involved
- User interface is implemented with Java Swing - leads to portability problems on some systems.
- Tool kit does not support a fully automated procedure from analysis to report

5.2.2. Forensic Triage Kit from scratch

The development of a forensic triage tool kit provides complete freedom in architecture, technology and functionality. The design phase requires a lot of time because it is important to provide useful extension points. The design and implementation of the basis of such a tool kit would consume the most of the available time in this project. The analysis functionality would be very limited. Further on the risk is high, that the development would not be continued after the end of this project.

5.2.3. Decision

Since the option "Autopsy contribution" promises a massively better feature result we decided to take this option rather than the option "Forensic Triage Kit from scratch". We accept that development environment and architecture are given and can not be changed. Because of this, there could be some limitations in development. We take advantage from already existing components, which can be used or extended. Further on the chance is much higher that our work will be used from others since Autopsy is already well known in the digital forensic industry.

5.3. Conclusion

The analysis of the existing forensic tool kits showed different analysis techniques. Furthermore, a base framework has been evaluated. Based on the existing functionality of the Autopsy forensic platform and the possible analysis techniques the requirements for the project can now be defined.

6. Requirements

6.1. Introduction

Based on the previous research about forensic tools and tool kits, important information was gained about what kind of standard processes are common in digital forensics. During the research, a tool kit was found which meets most of the initial requirements from the assignment and a decision was made to build upon it. This chapter describes the requirements regarding the development with Autopsy and Sleuth Kit. The requirements described in this chapter are partially based on the initial requirements set in the assignment of this bachelor thesis.

Some initial requirements set in the assignment could not be met due to limitations of existing tool kits. The decision to throw some of these requirements over board is based on an agreement between the developers and the supervisor.

More details on the initial requirements of the assignment can be found in Appendix-A.

6.2. Uses Case Diagram

Figure 6.1 describes the fundamental use cases for the forensic triage tool kit Autopsy.

Most of these functionalities are already implemented in Autopsy / Sleuth Kit. The orange marked Use Cases are not implemented yet and will be part of this Bachelor Thesis.

More Use Cases for modules that could be developed within this project can be found below. Detailed descriptions of the Use Cases are located in section 6.4.

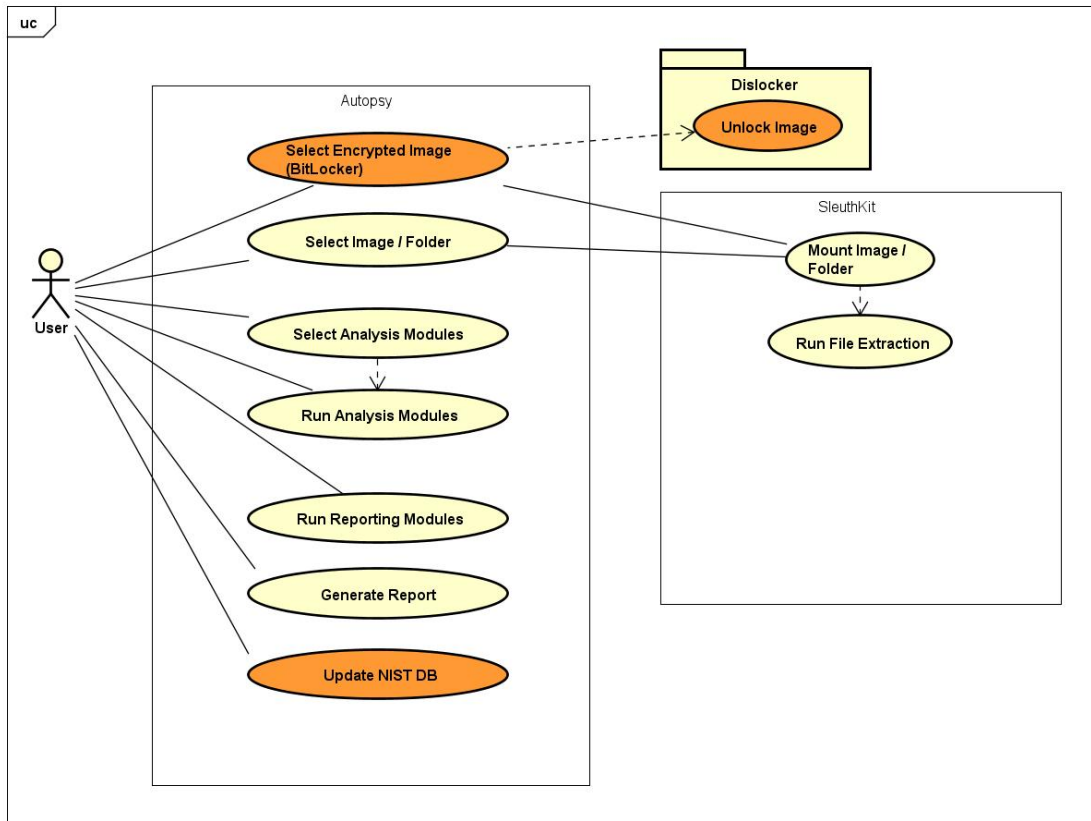


Figure 6.1.: Autopsy: Use Case Diagram

6.3. Actors

The main actor of this project is the user of the software, referenced as 'User'. He is usually an experienced computer user, who is skilled in scripting, programming, reversing and flow analysis.

6.4. Functional Requirements

This chapter describes the use cases defined for this project.

Since the project is being developed in an agile method, it is possible that use cases will change during the development process.

6.4.1. Add Bitlocker encrypted data source

Main Actor User
Priority MUST

The user wants to import an image to Autopsy which is encrypted with Bitlocker [26]. The user selects the image in the file-system and enters the decryption password or key. The program provides unencrypted access to the data source. After this process, the image can be used and analysed in the software.

6.4.2. Unlock Image

Main Actor -
Priority MUST

During the process described in section 6.4.1, the image must be decrypted. The included image must be forwarded to a decryption tool such as Dislocker on Linux systems. The tool decrypts the partition, so that Autopsy and Sleuth Kit have access to its data.

6.4.3. Update NIST NSRL Database

Main Actor User
Priority HIGH

The user always wants a local up-to-date NIST-NSRL database. To maintain this, the user can either update the database through a simple button click or the software automatically updates the database (for example on startup). The tool checks if there are any updates and includes the latest version locally.

6.4.4. Compare against Golden Image Data

Main Actor User
Priority MUST

The user wants to compare a Golden Image with an infected image. Then he loads both images into Autopsy and marks the golden image. This will allow the user, to compare specific parts of the image and exclude clean parts so the analysis scope will be reduced.

6.4.5. Analyse Registry Data

Main Actor User
Priority MUST

The User wants to analyse the Windows Registry. A Registry tree will be shown to the user, so the navigation process and overview is more simple for the analyst. If a Golden Image is used, a Registry Delta can be created which will exclude default and clean Registry entries - this makes it much easier for the analyst to concentrate on the relevant parts.

6.4.6. Analyse Meta Information

Main Actor User
Priority MUST

The user wants general information about the included Windows system. The tool shows various meta data such as information about the operating system, the configured users and installed software packages

6.4.7. Virustotal Online Hash-Check

Main Actor User
Priority MUST

The user wants to search for hashes in the Virustotal online database to find out if a file is already known as a bad file.

6.4.8. Analyse Event Log

Main Actor User
Priority MEDIUM

The user wants to examine the Windows Event Log. The tool displays the newest events based on a graphical timeline.

6.4.9. Run OpenIOC scanner

Main Actor User
Priority MEDIUM

The user wants to scan a system against an OpenIOC report. So he specifies the data source that should be scanned. The scanner then processes the input OpenIOC report and search for indicators on the selected data source.

6.5. Non-Functional Requirements

6.5.1. Reliability

Synopsis	Our software creates reproducible results. Tasks can be reverted or repeated in case of a failure.
Relevant QA	Reliability
Measurability	Can be measured by manually revert a task or cancellation of a task during runtime.
Open issues	Success of this NFR strongly depends on the reliability of the underlying frameworks.

6.5.2. Usability

Synopsis	The program can be used by an analyst without prior experience. The basic workflow of the software is self-explaining and easy to use. These include for example: <ul style="list-style-type: none">• Create an Autopsy case• Import a data source• Select, configure and run ingest modules• View Results
Relevant QA	Usability, Comprehensibility
Measurability	A user acceptance test will be performed.
Open issues	The existing project already has defined the user interface. We can not influence all parts of the interface.

6.5.3. Performance and Efficiency

Synopsis	The tool processes big data sources which usually takes some hours. Usage of efficient algorithms is therefore important.
Relevant QA	Performance, Efficiency
Measurability	The processing time of a data source must be approximately linear to its size. We test a 100 GB and a 200 GB data source and expect a runtime less than 210 percent of the smaller image.
Open issues	Not all modules can be implemented with a runtime linear to the data size, because some modules depend on network resources. Thus the runtime is not really predictable nor deterministic.

6.5.4. Portability and Transferability

Synopsis	The installation of the tool should be simple and the generated forensic data can be reviewed on every other installation of the same software.
Relevant QA	Portability, Transferability
Measurability	An autopsy project created on computer A can be opened on computer B without losing functionality.
Open issues	There is no existing mechanism to create a Linux package for this software project.

6.5.5. Correctness

There are not any special requirements to the correctness of the software. Specially because many of the processed data and results can not be declared as correct or corrupt by the software.

6.5.6. Changeability / Extensibility

Synopsis	Our Features often solve a specific problem. Through the definition of general interface, implementations of other similar problems should be simple. (e.g. different Hash Set or different disk encryption software)
Relevant QA	Changeability, Extensibility
Measurability	The implementation of a second Hash Set update should be planned and implemented.
Open issues	-

6.6. Conclusion

The definition of the requirements has set a milestone in this project and enabled the transition into the next phase. With this milestone reached, the planning and beginning of the development phase can be initiated.

The next chapter gives a detailed introduction into Autopsy and its architecture. Furthermore, it introduces into specific procedures and decisions of the features implemented within this project.

7. Autopsy Architecture

This chapter gives an overview on how the Autopsy forensic platform is designed. It also includes the extensions developed in this project. Furthermore, it contains information about specific procedures and decisions of the implemented features.

7.1. System context

Autopsy is a standalone platform that runs on a single computer. A few modules communicate with online services to get analysis results. The System Context Diagram illustrates the involved components (fig. 7.1).

The client platform consists of four main parts:

Sleuth Kit is an interface to the case database and the filesystem on the data source. It is written in C / C++ and provides a Java Native Interface for the Autopsy platform.

Autopsy is a user interface for Sleuth Kit. Additionally, it contains modules, that use the data provided by Sleuth Kit. The modules provide analysis functionality, user interface components or report generation types.

Dislocker provides functionality to encrypt and decrypt Bitlocker encrypted volumes. This library is used by the Decryption provider module.

Images are the data sources to be processed by the forensic platform.

The system context diagram contains two further components outside of the client:

NIST NSRL Database The NIST NSRL database is a database containing hashes of Windows components and common user software. This database can be downloaded and used offline. In order to maintain an automated up-to-date NIST NSRL database, Autopsy needs to update its local copy regularly. The hash lookup module uses this hashes to identify this known components.

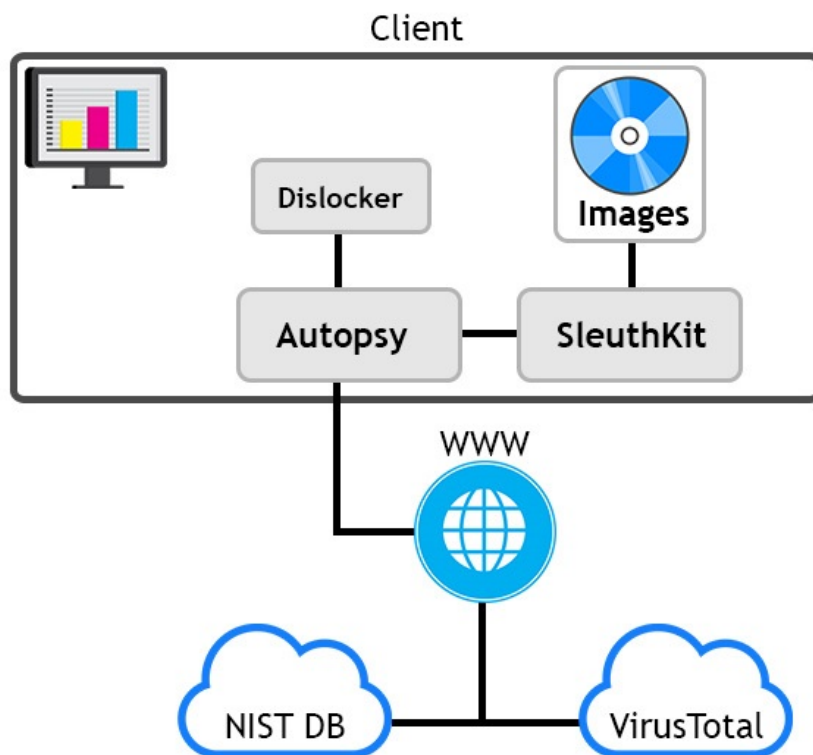


Figure 7.1.: System Context Diagram

Virustotal Virustotal is an online hash lookup service. This service is requested by a module to identify known-bad files.

7.2. Architectural Targets and Decisions

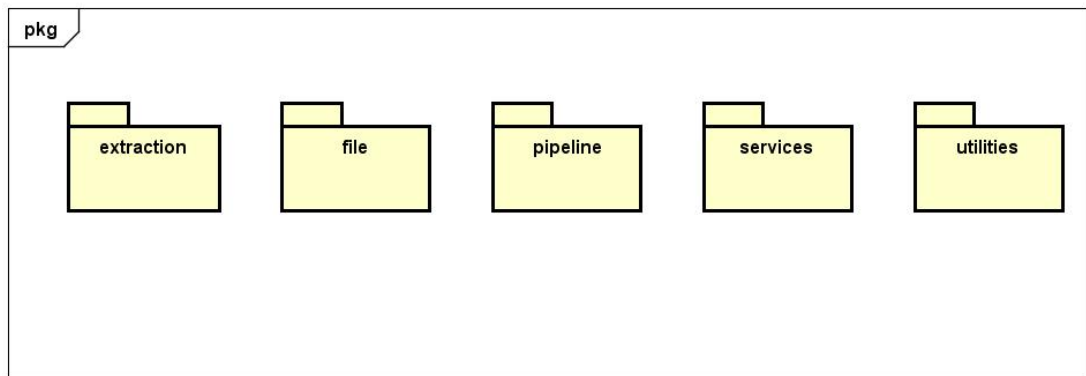
The software has to run on Kali Linux [32] and optionally runs on Windows.

For the analysis, it is expected, that the provided data sources are Windows systems. The ability to analyse images of other operating systems is optional.

It is required, that the user can import encrypted data sources into the Autopsy and that it will be decrypted within the software. To allow implementations for different disk encryption software a general interface is essential. The reference implementation supports Microsoft's disk encryption software Bitlocker.

7.3. Logical Architecture

7.3.1. Sleuth Kit Framework



powered by Astah

Figure 7.2.: Sleuth Kit Framework

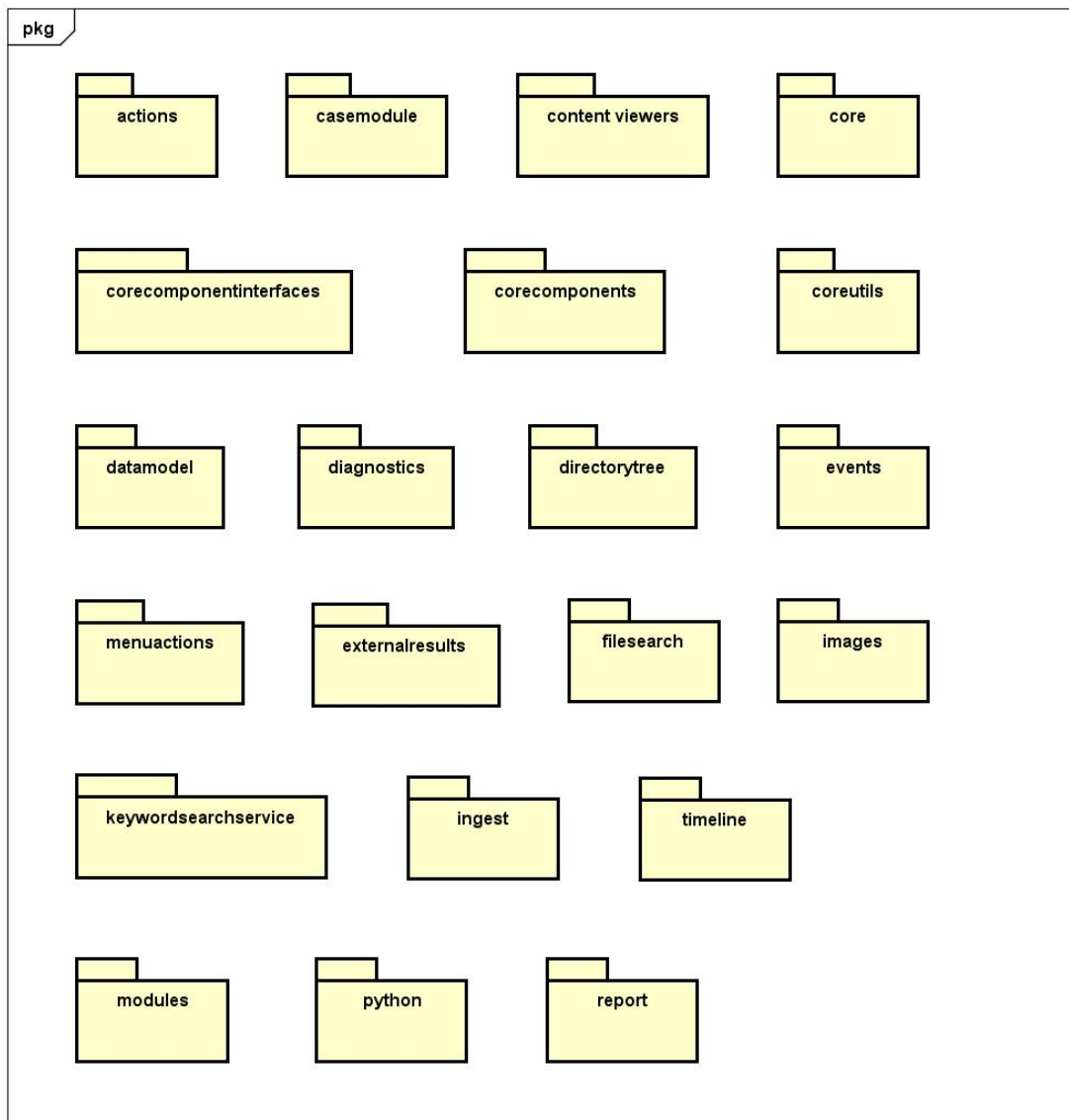
Above diagram shows the Structure of the Sleuth Kit Core Framework. Descriptions can be found on the official website of the Sleuth Kit project.

The logical architecture of this framework is not described in detail here since it is irrelevant for this project.

7.3.2. Autopsy Core

The Autopsy Core is based on the Netbeans Platform [16]. This basis is a framework for Java client applications. The modular architecture of Autopsy is based on functionality of the Netbeans Platform.

The packages mentioned in figure 7.3 are sub-packages from `org.sleuthkit.autopsy`. Below will be a description only of the most important packages which are relevant for this project.



powered by Astah

Figure 7.3.: Autopsy Logical View

7.3.3. Module

The Module-package contains all third-party modules.

7.3.4. Data model

The package "datamodel" contains important classes for working with images, files, file types etc.

7.3.5. Ingest

The Ingest-package contains relevant classes which are important for Ingest-Modules. This are actions which are executed after the image has been added to Autopsy and is being scanned.

7.3.6. Report

The Report-package contains important features for the Report generation. These are features like creating HTML-Reports, PDF-Reports or other forms of reports. But also functionality to merge results and prepare them for a report.

7.4. Specific Procedures and Decisions

7.4.1. Sleuth Kit - General Work Flow

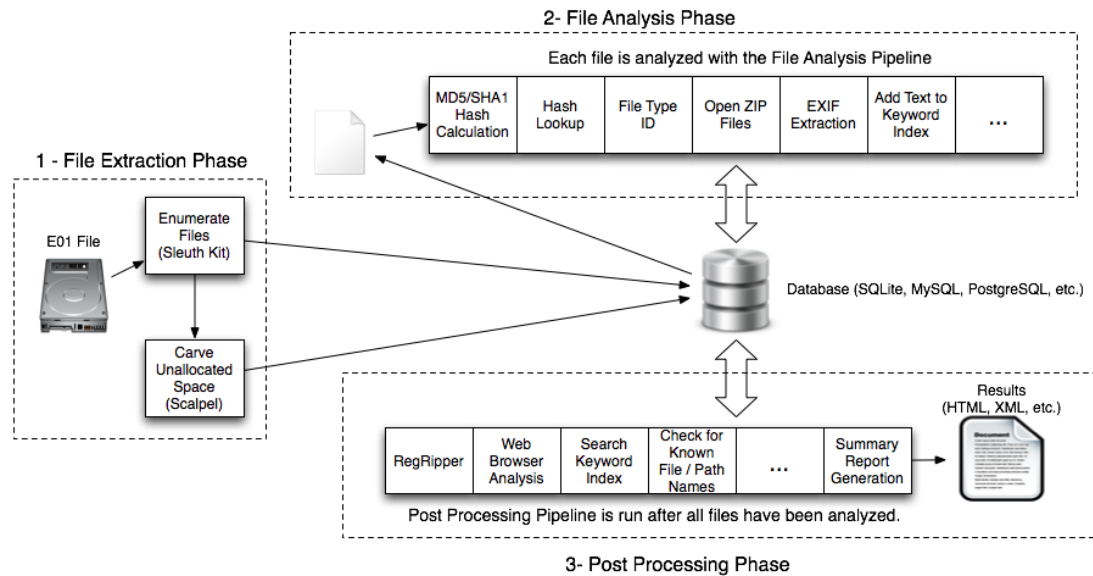


Figure 7.4.: Sleuth Kit- General Work Flow

The Sleuth Kits basic workflow 7.4 can be divided into three phases:

1. File Extraction Phase

Analyse disk images and identify files. Metadata about each file will be added to a central database.

2. File Analysis Phase

Each file will be analysed by running it through a series of modules. Each module has its specific task such as hash-value-lookup, calculating entropy etc. All of the results will also be saved in the database.

3. Post Processing / Reporting Phase

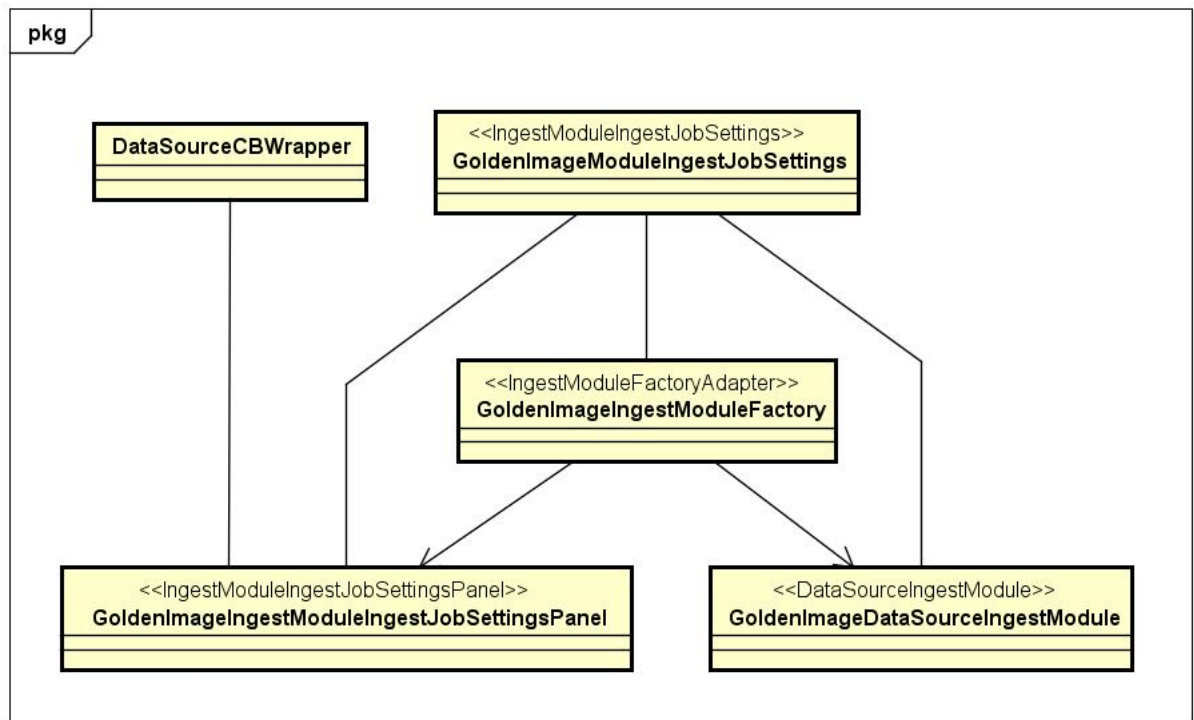
After the files have been analysed, some other modules are run. These modules may merge results, prepare them or make final reports.

For more detailed information about Sleuth Kit workflow, visit the official Website [13]

7.4.2. Module: Golden Image

The Golden Image feature is packed within a Netbeans Module package which can be imported and installed in Autopsy while its running.

Classes



powered by Astah

Figure 7.5.: Golden Image - Class Diagram

GoldenImageIngestModuleFactory

extends IngestModuleFactoryAdapter

This class is the core controller of this module. It is being called by Autopsy when it loads all the modules. The module factory is responsible for creating instances of all important module classes such as settings and the ingest module.

GoldenImageIngestModuleIngestJobSettingsPanel

extends IngestModuleIngestJobSettingsPanel

This class contains a panel with settings for the module. It is being showed when the user configures the ingest modules before running them in Autopsy. The instance of this class is being created from the module factory.

DataSourceCBWrapper

This class is a simple wrapper for a Content (data source). It is needed within the class *GoldenImageIngestModuleIngestJobSettingsPanel* as an object passed to the *JComboBox*. This is necessary since the *toString()* method of the Content isn't satisfying enough to display in a combo box.

GoldenImageDataSourceIngestModule

extends DataSourceIngestModule

This class is responsible for running the golden image process. It compares, hashes and tags the files. The instance of this class is being created by the module factory.

GoldenImageModuleIngestJobSettings

extends IngestModuleIngestJobSettings

This class contains all the settings for this module. The only main settings this class is about, is the reference to the golden image data source. Autopsy saves these settings so they will still be there after a restart.

Multi-threading

This module supports multi-threading for the process of hashing, comparing and tagging files. The multi-threading happens within the class *GoldenImageDataSourceIngestModule*. For the handling of the threads, an *ExecutorService* is used - more specifically a *FixedThreadPool*. The maximum concurrent threads that will be used is calculated by following equation: amount of available processors * 25. The minimum is 25 threads.

There is an inner class *FileWorkerThread* implements *Runnable* which is used as worker thread. Each worker threads processes one file.

File processing and Tag

The file processing happens within a worker thread. The sequence looks as following:

1. Find file on dirty image

2. Hashing: Hash both files
3. Comparison: Compare the hashes of both files
4. Tag File

There are 3 different tags:

- Good
- Changed
- Deleted

The **Good**-tag is set on files contained in the dirty image which haven't changed compared to the one in the golden image. This tag is defined in the class *GoldenImageIngestModuleFactory* as "DI_Good".

The **Changed**-tag is also set on files contained in the dirty image which have changed compared to the one in the golden image. This tag is defined in the class *GoldenImageIngestModuleFactory* as "DI_Changed".

The **Deleted**-tag is set on file contained in the golden image which weren't found on the dirty image. This tag is defined in the class *GoldenImageDataSourceIngestModule* since it has its tag name defined by the name of dirty image. The tag name has following structure: *DI_DELETED_[NAME OF DIRTY IMAGE]*. This makes sure the analyst knows on which dirty image the file doesn't exist. This can be very helpful if you have several dirty images within one Autopsy case.

New files, which are not on the golden image but on the dirty image can be viewed by using the Tag Filter module in Autopsy.

Known Issues

The Golden Image module takes a long time to process. One of the major reasons is the *findFile*-method in the class *GoldenImageDataSourceIngestModule*. For each file on the golden image this method is called. The method searches a file on another image by its file name and path through the *FileManager* from Autopsy. This method can take plenty of time to process under certain circumstances until it returns the wanted file. It often slows down when it is called too often in a short amount of time (For example if you iterate through a data source and call the method for each file). This is the main factor what makes the process very slow. Currently this can't be easily fixed. There might be a fix in the future when the Autopsy developers update their *FileManager*. **Important:** Making an own SQL-query to find the file won't help. It won't run faster and it might crash. Further on you still have to use the *FileManager* to get the *AbstractFile*-object from the file.

7.4.3. Module: Tag Filter

The module Tag Filter is wrapped in a Netbeans Module which can be imported and installed in Autopsy while it is running.

Autopsy Toolbar

To open the tag filter, there is a button in the toolbar in the upper right corner of Autopsy as seen in the screen shot beneath.

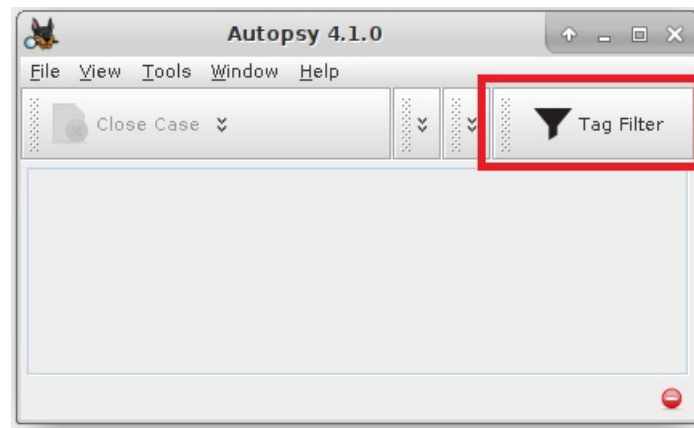


Figure 7.6.: Tag Filter -Toolbar

In order to add a section in the Autopsy toolbar within a Netbeans Module, you need following:

- JPanel with the contents of the toolbar section
- *AbstractAction*-class
- An entry in the *layer.xml*-file from the Netbeans Module

JPanel

The JPanel is implemented in the class *TagFilterToolbar.java*. It simply contains a button with an action (Open a new JFrame and display the tag filter [*TagFilterConfiguration.java*]).

AbstractAction

The AbstractAction is implemented in the class *TagFilterToolbarAction.java*. It extends *AbstractAction* and implements *Presenter.Toolbar*. This class contains two methods: *actionPer-*

formed and `getToolBarPresenter` which returns an instance of `TagFilterToolBar.java`.

layer.xml

In the layer.xml of the module, you have to add following two entries:

Definition of the tool:

```
1 <folder name="Actions">
2   <folder name="Tools">
3     <file name="org-sleuthkit-autopsy-modules-tagfilter-TagFilterConfigurationAction
4       .instance" />
5     <file name="org-sleuthkit-autopsy-modules-tagfilter-TagFilterToolBarAction.
6       instance">
7       <attr name="delegate" newvalue="org.sleuthkit.autopsy.modules.tagfilter.
8         TagFilterToolBarAction" />
9       <attr name="displayName" bundlevalue="org.sleuthkit.autopsy.modules.
10        tagfilter.Bundle#CTL_TagFilterToolBar.title" />
11     </file>
12   </folder>
13 </folder>
```

Listing 7.1: Definition of the tool

Definition for the toolbar:

```
1 <folder name="Toolbars">
2   <folder name="TagFilter">
3     <attr name="position" intvalue="200" />
4     <file name="org-sleuthkit-autopsy-modules-tagfilter-TagFilterToolBarAction.
5       shadow">
6       <attr name="originalFile" stringvalue="Actions/Tools/org-sleuthkit-autopsy-
7         modules-tagfilter-TagFilterToolBarAction.instance" />
8     </file>
9   </folder>
10 </folder>
```

Listing 7.2: Definition for the toolbar

Tag Filter Configuration Panel

The tag filter configuration panel will open in a new window after the user clicked on the tag filter button in the Autopsy toolbar in the upper right corner. In this panel the user defines the filters. Following classes and (sub-)panels are used to create the filter panel:

- `TagFilterConfiguration.java`
This is the main panel that is shown in the filter window.
- `TagFilterConfigurationFilter.java`
This is a sub-panel which contains one filter entry for the list in the main panel. This class extends the `TagFilterConfigurationFilterComp.java`

- `TagFilterConfigurationFilterGroup.java`
This is a sub-panel which contains one filter group entry for the list in the main panel. This class extends the `TagFilterConfigurationFilterComp.java`. A filter-group panel has a list which can contain filter-panels.
- `TagFilterConfigurationFilterComp.java`
This is the base class for the filter- and filter-group sub-panel. This is needed to have a unique type within the filters-list in the main panel.

Display Results

When the tag filter is executed and the search is done, a new result-viewer tab is created in the Autopsy main window which contains a table with all matching files. This action is created in the class `TagFilterSearch.java` in the method `createTopComponentPanel()` which is called after the search is done. The creation and filling of the result-table is implemented upon the OpenIDE framework with Nodes and ChildFactories. Below listing shows the method `createTopComponentPanel()` from the class `TagFilterSearch.java`:

```

1 private void createTopComponentPanel () {
2     DataResultTopComponent searchResultWin = DataResultTopComponent.createInstance ("Tag
3     Filter");
4     Node rootNode;
5     if (results.size () > 0) {
6         TagFilterResultFactory tfrFactory = new TagFilterResultFactory (results);
7         Children childNodes
8             = Children.create (tfrFactory , true);
9
10        rootNode = new AbstractNode (childNodes);
11    } else {
12        rootNode = Node.EMPTY;
13    }
14
15    DataResultTopComponent.initInstance ("Filter: "+createPathText (), rootNode , 10,
16    searchResultWin);
17 }

```

Listing 7.3: TagFilter: Display Results

The `DataResultTopComponent.class` is a class from the Autopsy core. It can be used to create tables with files as content.

In order to fill the table with elements, you need a child factory which creates nodes for the listing. The child factory class is called `TagFilterResultFactory.class` and extends `ChildFactory<AbstractFile>`.

7.4.4. Decryption Provider

The interface DecryptionProvider is designed to allow implementations for multiple disk encryption software. To implement a Decryption Provider, all methods visible in figure 7.7 have to be implemented:

getInstance	Factory method
start	The instance must enable access to unencrypted data
stop	The instance can deactivate the access to unencrypted data
matchesVolume	Decides if the given volume is encrypted with the specific implementation
getPanel	Provide a JPanel to configure key material for the given volume
run	initiate AddImageTask to load file metadata into case db

In addition, a DecryptionProvider must persist the key material. It must be able to unlock the disk after the case was closed and reopened.

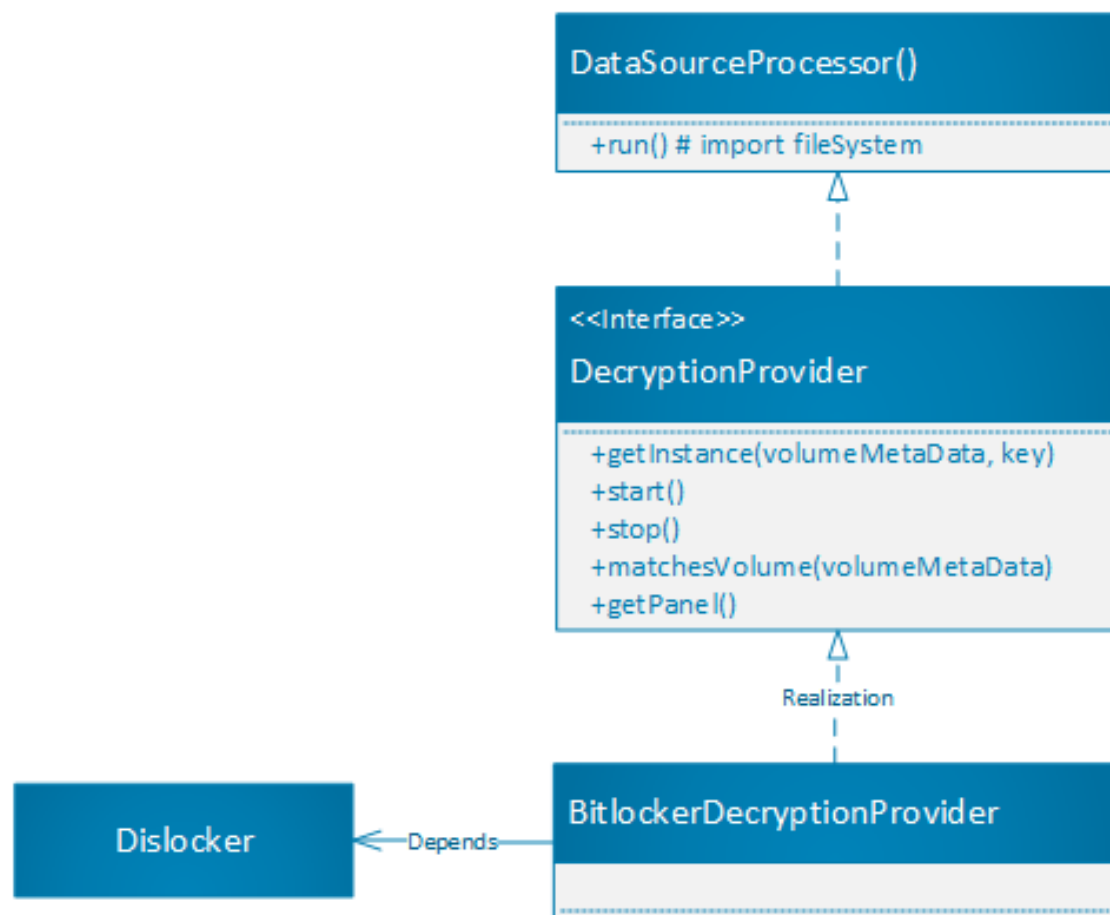


Figure 7.7.: Simplified Class Diagram of Decryption Module

Detecting volumes

During the “Add Data Source” process Decryption Provider instances are assigned to volumes. This requires the software to know the existing volumes on a disk. The Sleuth Kit is able to read the partition table. During this process it adds the relevant information to the case database.

To assign decryption provider instances to volumes, we need the partition table before the process is started. Therefore, a method on the SleuthkitJNI interface was implemented to get the offset to the partition start.

7.5. Libraries / Frameworks

7.5.1. Sleuth Kit

Sleuth Kit is a Forensic Triage Framework which provides several basic features such as loading and reading images, volume and file-system analysis and creating a meta-data database of all files on the image.

This Framework is the Basis of our project and provides the basic functionality which is needed for analysing images and file-systems.

For more information regarding Sleuth Kit, visit the official website <http://www.sleuthkit.org>

7.5.2. Dislocker

Dislocker is a library to use Bitlocker encrypted volumes on a Linux system. The library supports all version of Bitlocker including the current implementation for Windows 10. [1]

7.5.3. Jsign

This Java based implementation is able to create AuthenticCode signatures for Windows executables. The library was extended to perform signature verification in the AuthenticCode verification module. [2]

7.6. Deployment

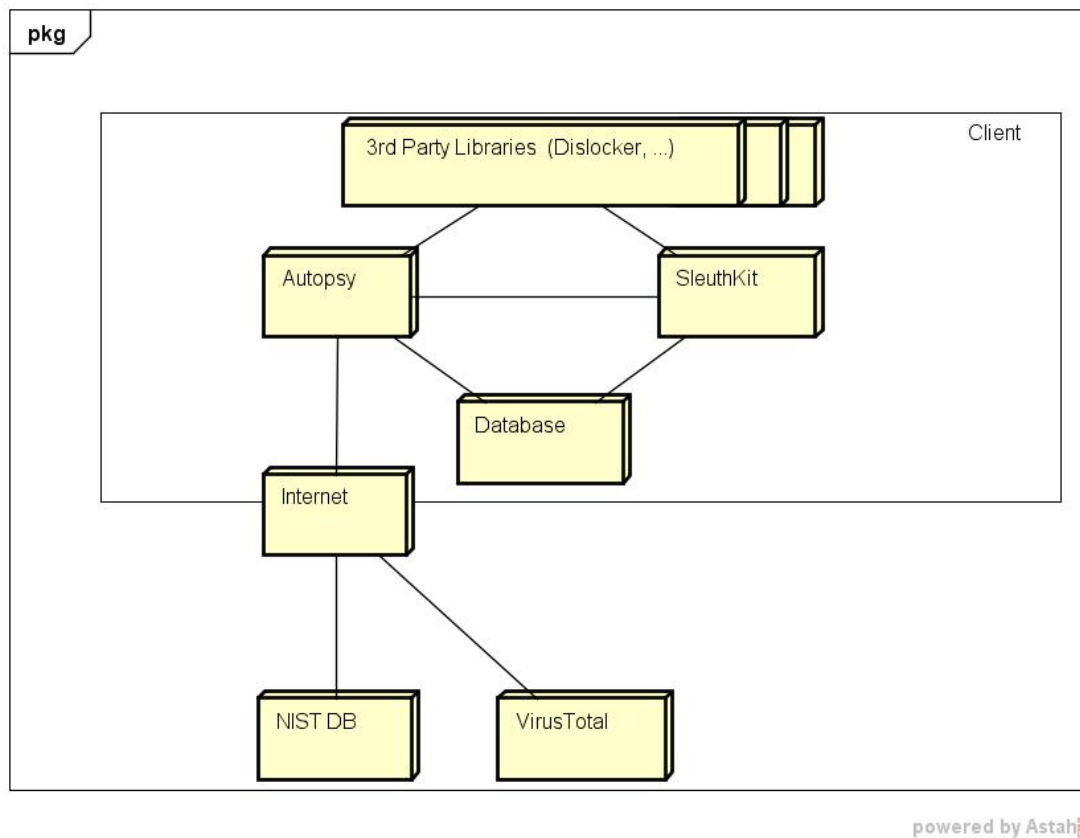


Figure 7.8.: Deployment diagram

The software will be run on a local machine. All frameworks, libraries and external tools are on the same local machine.

The database is automatically created by the Sleuth Kit during the case creation process.

Some modules might need an Internet connection in order to access online resources.

7.7. Data Management

7.7.1. Case related data

Sleuth Kit uses a case directory for each project. The directory contains the following elements:

- **case.aut** Stores meta information for the case
- **case.db** The case database
- **Cache Directory**
- **Export Directory** The target directory for exported files
- **Log Directory** Case related log files
- **ModuleOutput Directory** A place where modules can store data
- **Reports Directory** Location for all generated reports
- **Temp Directory**

Files from the data source are not copied into the case database, but all meta information about the files and folders are stored in the case database. The case database also handles results of the standard modules.

7.7.2. Configuration

Autopsy stores all user specific configuration in the users home directory in a sub directory “.autopsy”. This options are accessible over different Autopsy cases.

7.8. Collaboration between Autopsy and Sleuth Kit

The Sleuth Kit project is mainly written in C and C++ and native compiled. Autopsy is a Java software. To make these two components work together, an additional component is required. This interface is built on the Sleuth Kit project as Java Native Interface (JNI) [30]. JNI allows Java applications to load and use native compiled libraries.

7.8.1. Sleuth Kit JNI Interface

The Sleuth Kit JNI interface has two main tasks:

- provide access to the data source
- provide access to the case database

The interface also provides some modules which are used by autopsy.

Figure 7.9 illustrates the call hierarchy. As we can see, there is no direct access from Autopsy to the data source or the case database possible.

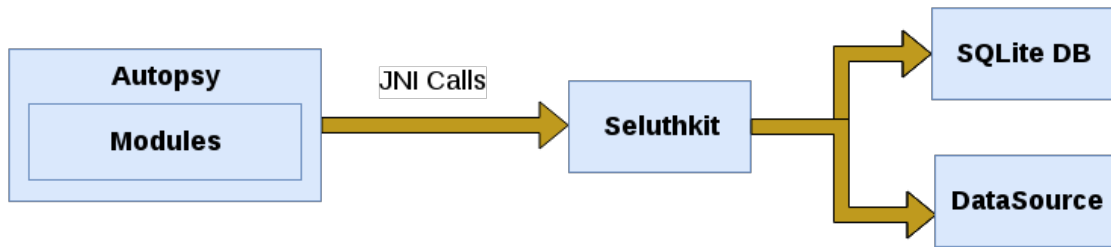


Figure 7.9.: Collaboration between Autopsy and Sleuth Kit

8. Results

8.1. Introduction

This chapter contains the documentation of the features which were implemented in this project. The primary goal is, to present the achieved results and explain their use. Besides the implemented results, the conclusion of this chapter will also talk about features that were not implemented although they were planned and the reason for their dismissal.

8.2. Bitlocker Decryption Provider

Autopsy currently supports three types of data sources. Image files, local disks and Files and Folders. Image files and local disks of enterprise computers often contain encrypted volumes. To allow autopsy to analyse such volumes, we have designed a generic extension for autopsy, which allows to implement multiple decryption providers.

For the most widespread encryption tools exists a Linux implementation which allows to mount an encrypted volume to an unencrypted volume. If we want to reuse this functionality, we only have to provide a start and a stop method for unlocking and locking encrypted volumes.

There is already functionality to add a raw image to an autopsy case, searching for deleted files and run ingest modules against the new Content. We can reuse this functionality through instantiate an AddImageTask which only knows the dislocked image location.

8.2.1. Recognize encrypted volumes

To minimize the user-interaction, we automatically recognize encrypted Volumes. The data source wizard creates an VolumeMetaData object for each volume on the data source. This object contains name, volume number, size and the first sector of the volume. Every DecryptionProvider has to implement the method matchesVolume which takes an VolumeMetaData object and decides if it can unlock this volume.

For the Bitlocker implementation, we can simply look for a magic string at the beginning of the volume [17] (fig. 8.1).

```
--
15f00000: eb58 902d 4656 452d 4653 2d00 0208 0000  .X.-FVE-FS-.....
15f00010: 0000 0000 00f8 0000 3f00 ff00 00f8 0a00  .....?.....
15f00020: 0000 0000 e01f 0000 0000 0000 0000 0000  .....
15f00030: 0100 0600 0000 0000 0000 0000 0000 0000  .....
15f00040: 8000 2900 0000 004e 4f20 4e41 4d45 2020  ..)....NO NAME
15f00050: 2020 4641 5433 3220 2020 33c9 8ed1 bcf4  FAT32  3.....
15f00060: 7b8e c18e d9bd 007c a0fb 7db4 7d8b f0ac  {.....|..}...
15f00070: 9840 740c 4874 0eb4 0ebb 0700 cd10 ebef  .@t.Ht.....
15f00080: a0fd 7deb e6cd 16cd 1900 0000 0000 0000  ..}.....
15f00090: 0000 0000 0000 0000 0000 0000 0000 0000  .....
15f000a0: 3bd6 6749 292e d84a 8399 f6a3 39e3 d001  ;.gI)..J...9...
--
```

Figure 8.1.: Bitlocker volume identification

Dislocker

The autopsy implementation to decrypt Microsoft Bitlocker volumes uses the dislocker library. Dislocker is an open-source fuse driver for Microsoft Bitlocker volumes.[1]

User interface

An encrypted volume normally requires some kind of key. A key can be any kind of data or device (String, File, SmartCard). Therefore every DecryptionProvider must implement the getPanel Method which returns an instance of a graphical configuration element. Our BitlockerDecryptionProvider allows three different Key Types (fig. 8.2).

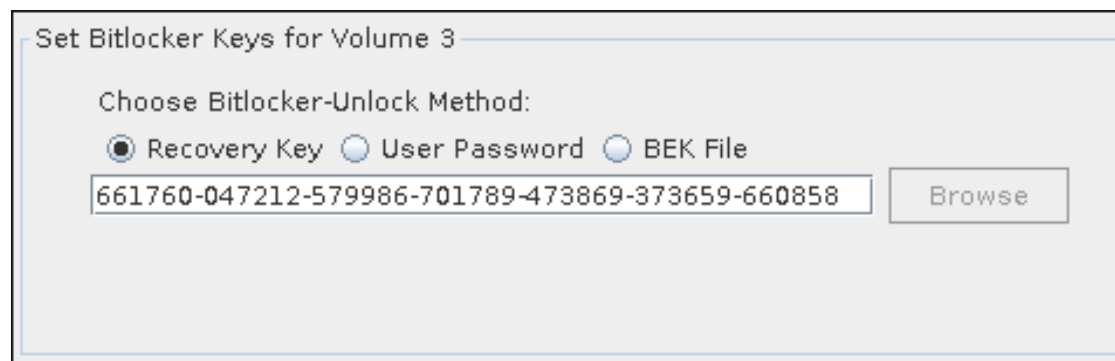


Figure 8.2.: Bitlocker key configuration options

Successfully added volumes through BitlockerDecryptionProvider appear in the list of data sources with the name dislocker. Autopsy extracts the name of a data source from the image-name. In case of dislocker, this name is a constant and cannot be manipulated.

8.3. Automatic HashSet update

An existing Autopsy feature allows to import MD5-Hash-databases. Every imported database must be marked as known-good or known-bad. The database can be provided in different formats. The simplest format is list of hex encoded MD5-hashes which are newline-separated.

The hashset lookup ingest module allows to compute hashes for files on a data source and find them in the imported hash-databases. The module sets the “Known” flag to files found in a known-good database or a content tag for known-bad files.

Many manual preparation steps are required to use this feature. The automatic hashset update extension automates this steps.

The wizard asks the user which hashsets he want to download, after pressing the start-button, the hashsets are downloaded, extracted and indexed automatically.

The following table displays the hashsets you can import into autopsy through this module:

Name	Type	Size
NIST NSRL	Known Good	5.3 GB
VirusShare.com	Known Bad	779 MB
ClamAV DB	Known Bad	355 MB

8.4. Virustotal Online Checker

Virustotal is a free online service that takes a file or a file hash and delivers a report if the hash is known.

The Virustotal module is implemented as file ingest module. For each file of a data source the hash is calculated. This file hash is used to request a report from Virustotal. If the file is reported as known-bad, the module creates a "Virustotal" file tag. The full Virustotal report is linked in the comment field of the file tag.

8.4.1. Limitations

The public API is limited to four requests per minute. This means it can only be applied to small data sources or single directories. To run this module over an entire workstation image takes a lot of time.

Virustotal offers unlimited requests for premium users. The prices for premium users can be requested inquired from Virustotal.

8.4.2. Obtaining an API key

The usage of this module requires an API key. An API key can be obtained on <http://www.virustotal.com> after successful registration with an username and a mail address.

8.5. Golden Image Module

8.5.1. Introduction

This document describes the Module "Golden Image". Several parts will be described such as the workflow, design decisions and process descriptions.

The Golden Image

The golden image is a regular image, like any other. It does not contain any special features or characteristics. The reason why it is called "golden image" is, because it is a fresh installation of the operating system (in our project it will be a fresh Windows installation). That means, the image will not contain any malware and also no user-installed software. Now imagine you have an image of a Windows-partition which might be contaminated. In the real world, such an image can easily have a size of 200GB, 500GB, 1TB or even more. The larger the partition is and the more files and software is saved on it, the longer it takes to analyse it. What we want to achieve with the golden image module is, to reduce the scope of the potentially contaminated image which the analyst has to analyse. The golden image will compare its "good" files, hashes and more with the equivalents from the other image and exclude all known good parts from the contaminated image. This will reduce the workload of parts which the analyst or software has to scan for malware and can save a lot of time. Before this documentation, several ideas and options on how to implement the "Golden Image" feature were created and described in the Appendix-B. We decided to do the implementation based on Option 2 which is described in the before mentioned document, although, some features had to be discarded from the original idea due to architectural blockades of Autopsy. For detailed information about the original idea, see the before mentioned documents or the chapter "Discarded Features".

8.5.2. General Workflow

Following is a workflow-description of the use of the "Golden Image" module:

1. The user adds the golden image:
He adds the golden image as a normal image, like any other.
2. The user adds the dirty image or wants to re-run the ingest-modules on an existing dirty image.

3. In the selection of the ingest modules that the user wants to run, he selects the module "Golden Image". In the configuration box, the user selects the golden image (There will be a dropdown menu in which all images which were added to the case are listed - the user needs to know, which of them the golden image is.)
4. After the configuration, the Golden Image module will be run. First it compares and tags the files against the dirty image.
5. After the comparison and tagging is done, the analyst can view in a filtered list, all files with either the "Good" or "Changed" tag and additionally, the "Deleted Files" will also be listed.

8.5.3. Module Type

The module is a "Data-source-level Ingest Module" according to the official description of the Autopsy documentation. More information can be found on following URL:

http://sleuthkit.org/autopsy/docs/api-docs/4.0/mod_ingest_page.html

The reason for this decision is, because in this module type a reference to the dirty image will be passed. This allows us to search the meta-data database of the dirty image, for the files on the golden image. Further information about this process can be found in the following chapters.

8.5.4. Initial Workflow Steps

In order to compare a dirty image with a golden image, the user has to import a golden image as a regular data source in Autopsy. The user does not have to run any ingest modules over this image (unless he wants to). After he added the golden image, he either has to import the dirty image to Autopsy, or open the panel where he can re-run ingest modules over the dirty image if he already added a data source. In the following chapter "Module Settings" we will talk about the settings of the "Golden Image" module which the user has to/can set before running the module.

8.5.5. Module Settings

The "Golden Image" module will enable the user to set following settings:

- The Golden Image: The user has to specify the golden image. There will be a dropdown in which the user can select the golden image out of all images which were imported to the current Autopsy-case. MUST-Setting.

The selection of the Golden Image will be placed in the general settings panel of the Golden Image. (See following wireframe "Wireframe: Ingest Module Selection").

Wireframe: Ingest Module Selection

Following wireframe shows the ingest module selection which is shown when adding a new data source or when re-running ingest modules of already imported data sources. This panel already exists in the Autopsy core. The interesting part is on the right side where the general settings of the "Golden Image" module is displayed. When the "Golden Image" module entry is focused, the general settings panel of this module is shown. It provides a short description about module and also a Dropdown-menu from which the user can select a golden image.

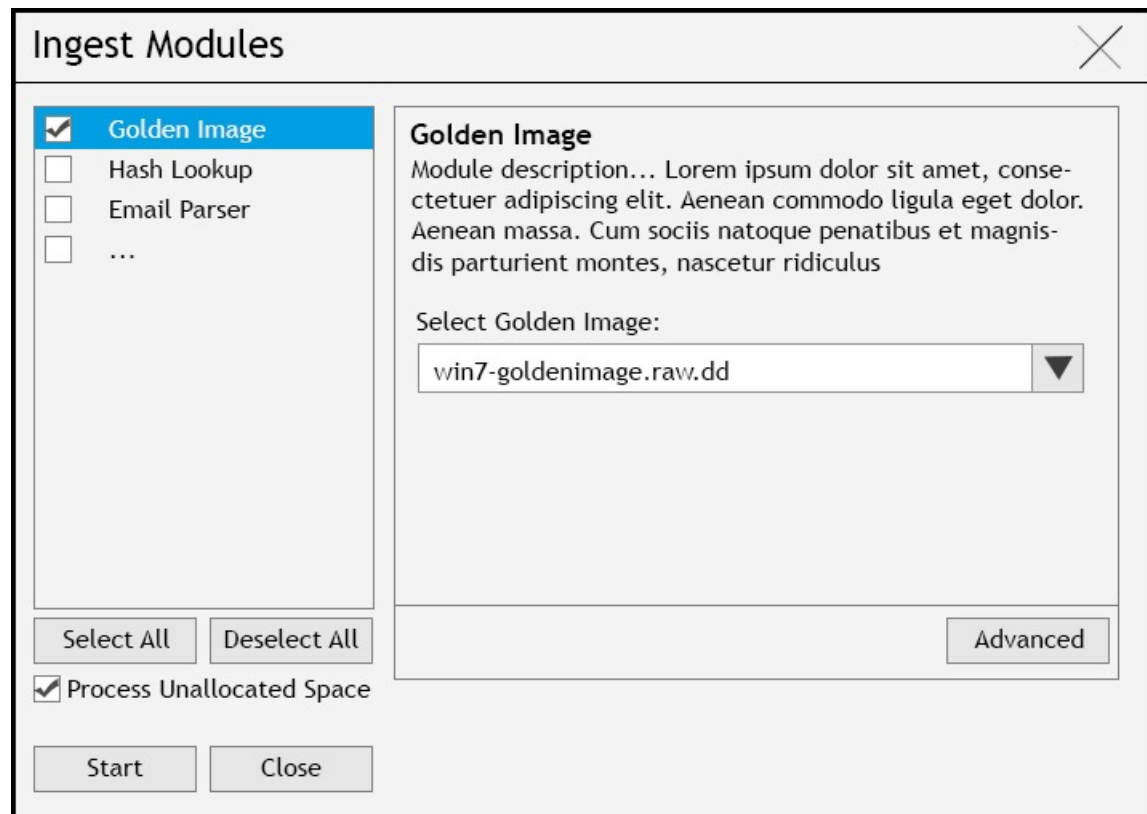


Figure 8.3.: Wireframe - Ingest Module Selection

8.5.6. File Tagging and "Deleted Files"

During the comparison process in this module, files will be tagged. All file-tagging will happen on the dirty image. There are 4 types of files:

- Untagged files
- Changed files
- Good files
- Deleted files

Untagged Files are files which were not found on the golden image. They will not be tagged. **Changed Files** are files which are found on the golden image but differ from the one on the dirty image. These files will be marked on the dirty image as "Changed". **Good Files** are files which are found on the golden image and were not changed. These files will be marked on the dirty image as "Good". **Deleted Files** are files which are found on the golden image but were not found on the dirty image. These files will be tagged on the golden image with a special tag for the dirty image data source.

8.5.7. Golden Image Workflow

This chapter describes the workflow of the "Golden Image" module when it is run. There is one main-steps in this process:

- Comparison and Tagging

Comparison and Tagging

In this step, the module runs through all files in the golden image. Each file will be checked against the dirty image. It will be checked if it exists, got changed or is the same. Following flow chart describes the process of each file.

8.5.8. Discarded Features

The "Golden Image" module was originally planned with more features than it comes with now. Following Features have been discarded from the Project.

DataSet Filter / Run Ingest Modules

In order to run an Ingest Module on a filtered dataset, a lot of changes have to be done in the Autopsy core. Since we planned to create this feature as a 3rd-party module, we discarded this feature. This feature can be implemented in the future, when the Autopsy core is extended with this functionality.

Golden Image View

Originally it was planned to also implement a view for the tagged files with an easy-to-use filter. We have decided to not implement this feature in the "Golden Image" module, but instead, to implement a new, generic filtering-module which can then be used also for other tagged files.

8.5.9. Features: Outlook

The golden image has potential to be extended with further features. Following are some ideas:

- File comparison regardless of path and file name
- Meta-data comparison
- Data set filter: Exclude files from the whole comparison and tagging process

Some of the above mentioned features may require updates in the Autopsy Core.

8.6. AuthentiCode verification

Microsoft developed AuthentiCode technology for code signing and verification on Windows operating systems. The goal of code signing is to provide trust into software packages. Cryptographically signed code ensures that the code was not altered from the time the publisher signed the code until it gets executed on a user's computer. The second purpose is to authenticate the publisher. But the existence of a valid AuthentiCode signature is not a proof that the file has not a bad intention. Neither the signature provides quality information.

8.6.1. Module

The Autopsy AuthentiCode Module uses signatures to identify the publisher of software on the system. The module creates a view which groups files by its publisher.

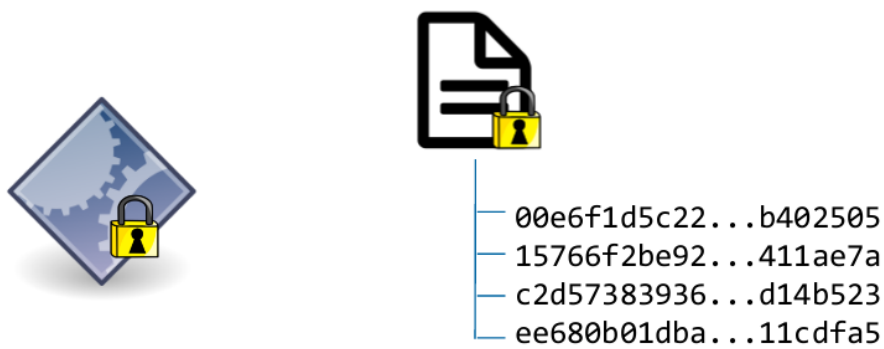
The analyst must decide which publishers are trustworthy and which are not. The main part of signed software usually is code from the operating system vendor or from installed 3rd party software. If there is an unknown publisher in the list, the analyst can investigate into this files.

8.6.2. Types of signatures

AuthentiCode supports two different types of cryptographic signatures (fig. 8.4). Embedded signatures are self contained, that means the signature is a part of the executable file. The file format for executable files is the "portable executable" [27] which provides an optional field for the signature [29].

Detached signatures are stored in catalog files. Catalog files are PKCS # 7 files, which is a format to store certificates, signatures and signed content. [19]

A detached signature is a signed collection of file hashes. The signature is valid for the files the hashes are computed from.



Embedded Signatures

Detached Signatures

Figure 8.4.: different types of Authenticode signatures

8.6.3. Existing Authenticode tools

Microsoft offers a bunch of tools [28] to create and verify signatures. These tools use the Windows CryptoAPI and run exclusively on Windows.

Another tool provided by Windows Sysinternals is sigcheck.exe [31]. This tool is developed for forensic investigation. It verifies the signature.

Jsign

Jsign is an open Java project which allows to sign Windows executables on a non-Windows build server. The tool contains classes which understand the PE File format and the Authenticode signatures. There is no built-in support to verify such a signature.

8.6.4. Implementation

Through the implementation of some additional features on the existing Jsign library (section 8.6.3), the module can use this library to perform signature verification for Windows software.

The module verifies the signature, and creates a file tag with the subject of the signer certificate.

Verification of embedded signatures

Embedded signatures can be verified file by file. This can be implemented as file ingest module. The pseudo code 8.1 illustrates the process.

```
1  @Override
2  public ProcessResult proces(AbstractFile file) {
3      if (!fileHasPEHeader(file)) {
4          return ProcessResult.OK
5      }
6
7      if (fileHasValidSignature(file)) {
8          Certificate certificate = getSignature(file);
9          createContentTag(certificate.getSubject, file);
10         return ProcessResult.OK
11     }
12 }
```

Listing 8.1: AutheniCode File Ingest Module Pseudo Code

Verification of detached signatures

It is not possible to verify detached signatures by just looking at a single file. The verification requires an analysis of the entire data source. This requires a data source ingest module.

The module performs the following steps over a whole data source:

1. Find all Catalog Files on the data source
2. Validate the signature of the catalog files.
3. Collect all correctly signed hashes from this catalog files
4. Create file hashes for each file on the disk
5. Look for the file hashes in the list of correctly signed hashes.
6. Create a content Tag for all matches from the previous step.

The hashes in the catalog files are computed by different digest algorithms. Because we do not have a link between a potentially signed file and the catalog file, we are forced to compute every hash type for each file.

The Ingest job settings allow a user to select which digest algorithms he want to compute. The default settings have SHA-1 and SHA-256 enabled.

8.6.5. AuthentiCode Data Content Viewer

The AuthentiCode Data Content Viewer displays the result of the AuthentiCode Ingest Module for a single File. The limited view in fig. 8.5 displays where the according signature was found and some information about the publisher certificate.

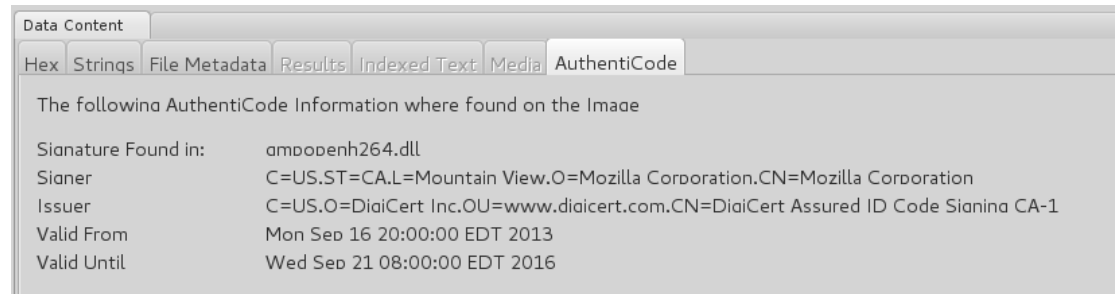


Figure 8.5.: AuthentiCode Data Viewer Content

8.6.6. Outlook

The data content viewer displays just a few lines of meta information. This view can be extended to show the full certificate chain with all certificate flags.

To fully validate the AuthentiCode signatures, it is required to check the time stamping signature too. The result of the additional verification step should be integrated into the AuthentiCode data content viewer.

8.7. Tag Filter Module

8.7.1. Introduction

Problem

In Autopsy, each tag has its own view with a table containing all files which are tagged by this specific tag. Since there are many different modules with their own tag, some tags might specify a file for the same reason. For example, the Golden Image module has its "Safe" tag while the HashSet comparison module also has its own tag for safe files. An analyst might want a list with all files that are safe, no matter which module tagged it as safe - or more likely a list containing all files, which were not tagged as safe. Autopsy only comes with limited filter options offering only a few, hard-coded tags to filter with. The module "Tag Filter" enables the user to create custom filters with all available tags contained in the Autopsy case.

8.7.2. Features

- Tag Filter
- Content Viewer: Tags

Tag Filter

The user can specify an unlimited amount of filters to generate a list of files as he desires. The filter is based on the idea of the SQL where-statements.

Example: `SELECT file WHERE file contains TAG_X AND file does not contain TAG_Y`

For each filter the user can specify:

- **The combination-operator:** AND / OR
 - This defines, how the filter should be combined with the previously defined filter.
- **The negation-operator:** Contains / Does not contain
 - This part defines, if the file contains the specified tag or not
- **The tag** which is an elementary part of the filter

Further on, the user has the possibility, to create filter groups. This enables the user to create more specific filters. Each filter group, contains one or more filters.

Example: `SELECT file WHERE file contains TAG_X AND (file contains TAG_Y OR file contains TAG_Z)`

Content Viewer: Tags

The Content Viewer for tags, adds a content viewer to Autopsy to view all assigned tags of a file.

8.7.3. Screenshots

Opening The Tag Filter

In the upper right corner, there will be a button called "Tag Filter". By clicking on it, a new window will open with the filter configuration panel.

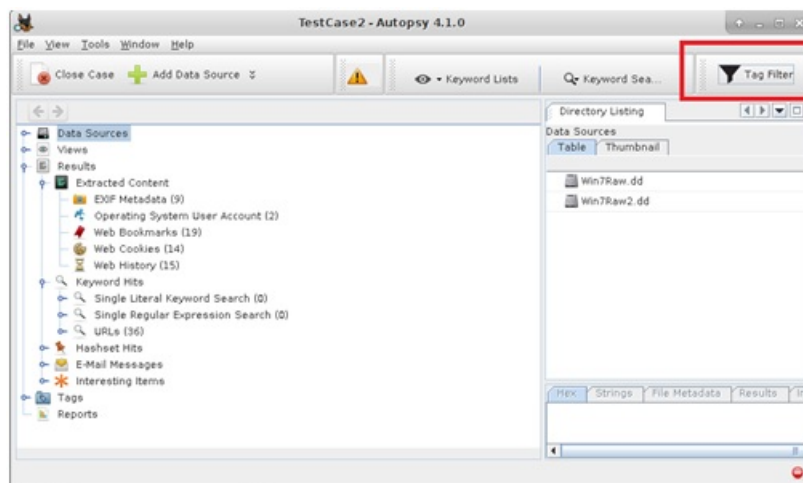



Figure 8.6.: How to open the Tag Filter

Tag Filter Configuration Panel

Following shows a screenshot of the Tag Filter configuration panel. The filters / filter groups are meant to be read top-down. The combination-operator (AND / OR) is connected to the previous filter in the list.

- **Button "Add Filter"**: By clicking on this button, a new filter will be added to the end of the list.
- **Button "Add Filter Group"**: By clicking on this button, a new filter group will be added to the end of the list. By clicking on the plus-icon () on the right side of the filter

group, a new filter can be added to this filter group.

- **Dropdown Datasource:** The user can specify, if he only wants to search for files on a specific datasource. If none is selected, the filter will search for files in all datasources contained in the current Autopsy case.
- **Button "Filter":** By clicking on this button, the filter will be applied and matching files are being searched. The Tag Filter configuration window will close and a list with the matched files will be displayed to the user.

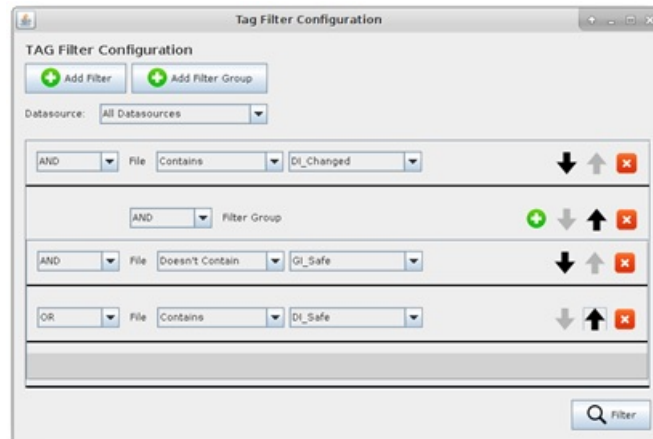


Figure 8.7.: Tag Filter Configuration Panel

Filter Entry




Following screenshot shows a filter entry within the Tag Filter configuration panel:



Figure 8.8.: Filter Entry

- **Dropdown "Combination-operator":** In the first dropdown on the left, the user can specify the combination-operator AND / OR.
- **Dropdown "Negation-operator":** In the second dropdown on the left, the user can specify the negation-operator Contains / Does not contain.
- **Dropdown "Tag":** In the third dropdown on the left, the user can specify the tag of

which this filter is about. This dropdown dynamically shows all tags contained in the current Autopsy case.

-  This button enables the user to move the filter down in the list.
-  This button enables the user to move the filter up in the list.
-  This button enables the user to delete the filter from the list.

Tag Filter File List

After creating and applying a filter, all matched files will be displayed to the user within a table as following screenshot shows:

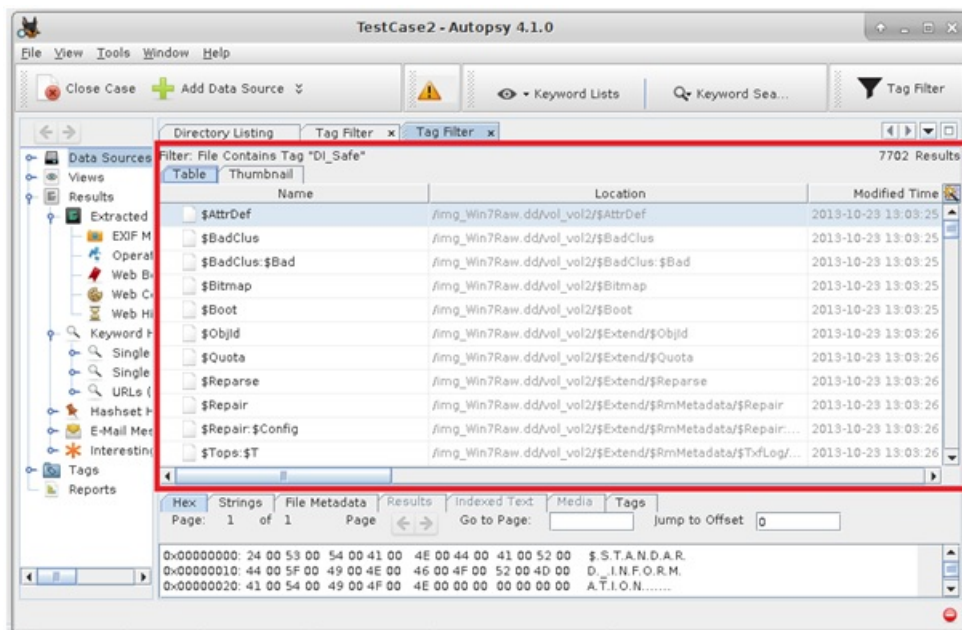
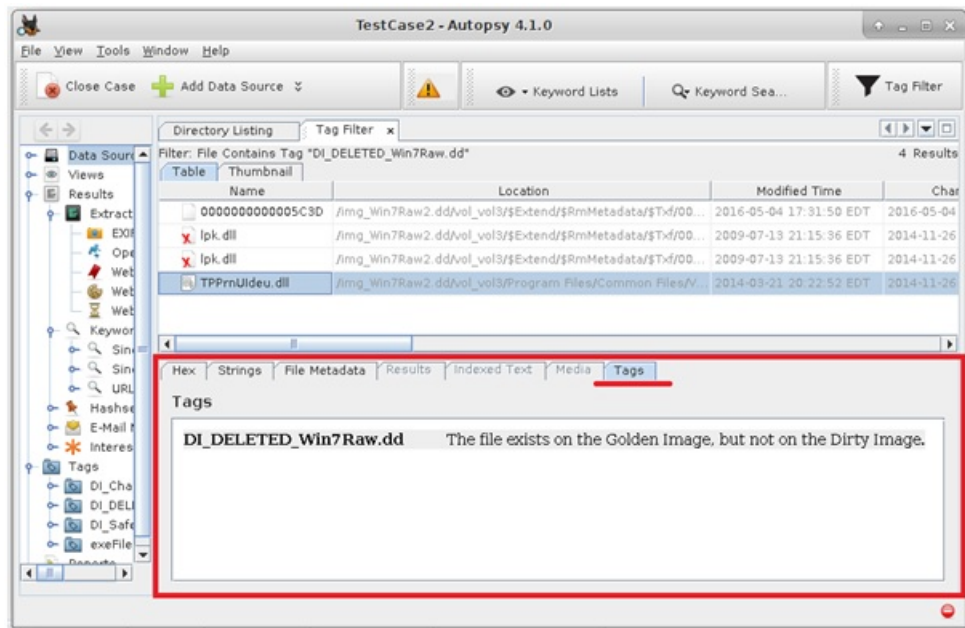


Figure 8.9.: Tag Filter File List

Content Viewer: Tags

The Content Viewer Tags is a tab which is shown in the lower area of Autopsy. This view is not only available for the Tag Filter results list. It is also available for any other file-list within Autopsy. The Content viewer for tags shows all tags which are assigned to the selected file.



H]

Figure 8.10.: Content Viewer for Tags

8.7.4. Features: Outlook

This module has potential to be extended with some additional functionality. Following are some ideas:

- Advanced filter possibilities: Add the possibility to also filter files by dates, owner etc.
- Export: Add functionality to export the filtered file list. Either as a table list such as Excel or save the list within Autopsy so the user can call it again in the future without having to use the filter again.
- Save Filters: Add the option to save filters

8.8. Repositories

The listing below contains the links to the GitHub repositories for each feature. The repositories are publicly available for everyone.

- Sleuth Kit
<https://github.com/mvetsch/sleuthkit>
- Autopsy
<https://github.com/mvetsch/autopsy>
- VirusTotal Online Checker
<https://github.com/mvetsch/VirusTotalOnlineChecker>
- Golden Image Module
<https://github.com/colapse/Autopsy-GoldenImage>
- AuthentiCode verification
<https://github.com/mvetsch/Autopsy-AuthentiCodeVerification>
- Tag Filter Module
<https://github.com/colapse/Autopsy-TagFilter>

8.9. Conclusion

8.9.1. Results

The results of the implemented features within this project are powerful extensions for Autopsy. They bring great benefits to digital forensic analysts and can make them save a lot of valuable time by partially automating analysis steps.

8.9.2. Dismissed Features

Some of the initially planned features, were not implemented within this project due to different reasons. Following listing presents the planned features that got dismissed.

- Analyse Registry Data (See 6.4.5)
- Analyse Meta Information (See 6.4.6)
- Analyse Event Log (See 6.4.8)
- OpenIoC Scanner (See 6.4.9)

Some of the implemented features took more time than expected. Further on, due to the agile implementation method, the decision to implement the feature "AuthentiCode verification" was made in the middle of the development phase. Due to the above mentioned reasons, some of the other planned features had to be dismissed, because there was not enough time to implement everything.

The feature "Analyse Registry Data" (See 6.4.5) was dismissed, because the library contained in Autopsy, which enables access to the registry data of a windows image is faulty and throws a lot of exceptions. This made it impossible to implement this feature without exceptional high effort.

9. Benchmarking

The benchmarking was performed to test the practical suitability of the developed features. On one hand the consumed time per feature was measured, on the other hand the results, for example the amount of hits per module. The goal is to make a statement about the time consumption, the efficiency and the quality of the results.

9.1. Benchmarking setup

The specific tests measure the runtime of a specific module. They do not measure the import of the data source into autopsy or other modules which may have been executed on the data source.

9.1.1. Hardware

All tests have been performed on the same Hardware. Table 9.1 shows the hardware specification of the used computers in this benchmarking.

Manufacturer	Lenovo
Model	T410 Notebook
RAM	8 GB
Solid State Disk	Samsung SSD 256 GB
CPU	Intel Core i5 CPU M520 2,4 GHz

Table 9.1.: Benchmarking: Hardware

9.1.2. Data sources

The benchmarking was accomplished with four different images. All images have Windows 8 installed with the latest patches (28th May 2016).

Table 9.2 shows a list of the images and describes their characteristics.

1	win8-GoldenImage_Bitlocker	Golden Image, a lot of 3rd party software installed, size is 60GB
2	win8-GoldenImage_Reboot_Bitlocker	Golden Image after a single reboot, size is 60GB
3	win8-GoldenImage_No_Bitlocker	Golden Image without Bitlocker enabled, size is 60GB
4	win8_Raw_Bitlocker	Windows 8 installed without any third party software, size is 25 GB

Table 9.2.: Benchmarking: Images

The images have been copied to the SSD on the benchmarking device before the test was started.

9.2. AuthentiCode on raw Windows 8

This test measures the AuthentiCode module on a Windows 8 installation without additional software installed (Image No. 4 as listed in table 9.2).

Analysed Image	4
Duration	19 minutes 46 seconds
Embedded Signatures found	8'250
Amount of catalog files	23'010
Amount of files signed through detached files	125'829
Amount of different Publishers	27

Table 9.3.: Benchmarking: AuthentiCode on raw Windows 8

9.2.1. Conclusion

This test displays, that about four out of five files on a raw Windows 8 installation are signed with an AuthentiCode Signature. (fig. 9.1). The majority of the content of the installation has a detached signature. The majority of the signatures have been issued by Microsoft. The remaining five percent are drivers signed by the hardware manufacturer.

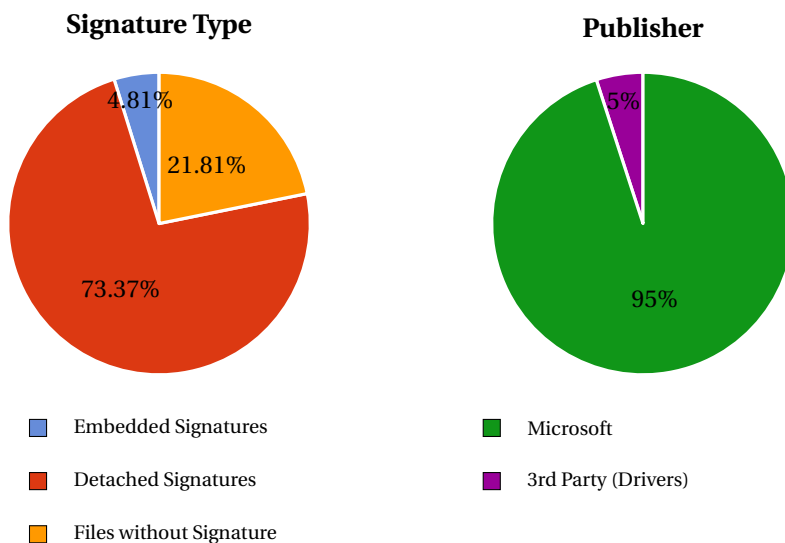


Figure 9.1.: Signature information of a raw Windows 8 Image

9.3. GoldenImage Reboot check

In this test the golden image module was run over two nearly equal system images. The difference between the two images is simply a restart of one of the images to see what happened on the system during a restart of the system.

Table 9.4 shows the results of this benchmarking.

Analysed Image	1 and 2
Duration	22 hours 17 minutes
Good files	246857
Changed files	3587
Deleted files	3459
New files	429

Table 9.4.: Benchmarking: Golden Image - reboot

9.3.1. Conclusion

The runtime of the module is unexpected high. The expected duration is about 10 percent of the actual consumed time. The difference is caused by a Java method which is looking for

a file on the dirty image by using the path and file name. This method takes a lot of time to deliver a result. Since this method is part of the Autopsy core (connected with functionality in Sleuth Kit), this issue cannot be resolved within the module.

9.4. HashSet check

The following hash sets have been downloaded and indexed to the Autopsy forensic platform.

- NIST NSRL 2.51
- VirusShare.com (accessed at 5th June 2016)
- ClamAV Database (accessed at 5th June 2016)

The selected image is a fresh and untouched Windows 8 installation. The image is considered to be uninfected. All matches on a Known Bad hash set are considered to be false positives.

Table 9.5 shows the results of this benchmarking.

Analysed Image	4
Duration	28 minutes 54 seconds
NIST NSRL matches	18968
VirusShare false positives	91
ClamAV DB false positives	2
False positives through empty files	22685
Unidentified Files	137 052

Table 9.5.: Benchmarking: HashSet check

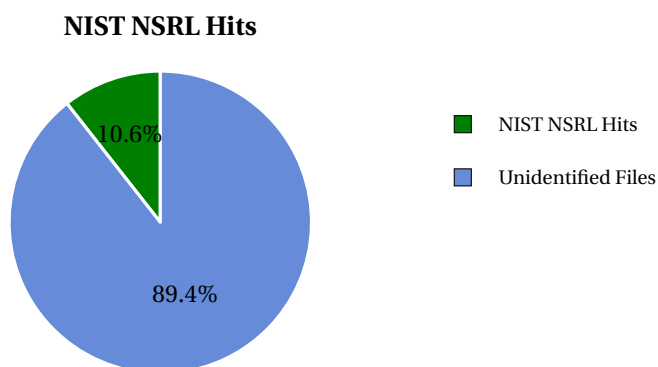


Figure 9.2.: Signature information of a raw Windows 8 Image

9.4.1. Conclusion

The above mentioned results of this benchmarking (Table 9.5 and figure 9.2) clearly show, that this feature is not very effective since it delivers a load of false positives and a large amount of the files on a fresh Windows 8 installation are not recognized. The NIST NSRL database only identified 10.6% of all the files on the image which is a very low amount.

9.5. Bitlocker Efficiency

In order to test the efficiency of the Bitlocker feature, we have tested two similar scenarios. In one of the scenarios we tested some modules on a Bitlocker encrypted image and in another scenario one without. This test indicates the additional overhead by using Bitlocker encrypted partitions.

This test is meaningful when running a module over the content with a high reading rate. The benchmarking test from section 9.4 has been performed on the same image without Bitlocker encryption.

Image 1, Bitlocker enabled	28 minutes 54 seconds
Image 3, Bitlocker disabled	12 minutes 7 seconds

9.5.1. Conclusion

This test illustrates a performance overhead for encrypted data sources. This specific test took a factor of 2.39 longer on the encrypted data source. In cases with a lot of different modules running through a large amount of files, it could save a lot of time if the data source would be decrypted in a first step and then import the decrypted data source into Autopsy.

10. Conclusion

10.1. Assessment of the results

The result of the project is a feasible Forensic Triage Kit. The strength of the tool kit lies in the separation of known-good files. Nevertheless, due to the troubles occurred with understanding the Autopsy forensic platform, not all planned features could be implemented during the project.

10.1.1. Summary of the implemented Features

We enhanced the forensic triage with Autopsy with the following features:

- Support for Bitlocker encrypted volumes
- Automated download for a several hash-sets
- Golden Image module
- AuthentiCode verification module
- VirusTotal online hash check module
- Tag Filter module

The features allow a forensic triage, targeting Windows systems, although some modules can also be used for other operating systems. More analysis modules can be used in the same projects using Autopsy or third party modules [4].

10.2. Outlook

In order to allow further development on our features for everyone, the code is publicly published on GitHub. (See chapter 8.8 for further information)

We are convinced that our features will be successfully applied in practice. The industry

partner of this Bachelor Thesis already planned to use the Forensic Triage Kit in a customer project.

This test run will decide whether it is useful in the industry and if it is worth to start further development and implement new features.

To get feedback from the Autopsy community, we will participate in the Autopsy module contest this year. The response from the community will give us besides the test run in the industry if our features are useful and if it makes sense to extend the existing features.

The possibilities for forensic triage are way not exhausted yet. Based on Autopsy and our contribution, it is possible to implement more analysis techniques. Most of our implemented features have potential to be extended with additional functionality to improve analysis and output. Further on there a lot more forensic methods which have not been implemented yet for Autopsy, so there is a lot more that can be done to improve digital forensic processes with this tool kit.

Part II.
Appendices

A. Aufgabenstellung

A.1. Einführung

Je länger je mehr wird die Untersuchung von potentiell verseuchten Workstations und Serversystemen zur täglichen Routine eines IT Forensikers resp. Mitarbeitenden in Computer Incident Response Teams (CIRTs). Bei der Triage von potentiell verseuchten Inhalten werden typischerweise Blacklists von Malware und Whitelists von bekannten Softwarekomponenten herangezogen. Im Enterprise-Umfeld können in der Regel sogar Festplattenabbilder von Neuinstallationen als Referenz verwendet werden.

A.2. Aufgabe

Im Rahmen dieser Arbeit sollen ein Framework sowie eine Hand voll Basis-Plug-Ins erstellt werden, welche die Triage automatisieren und bekannte Dateien aus der Verarbeitung ausscheiden. Zudem soll eine Zusammenstellung der Resultate präsentiert werden. Die übrigen Dateien gilt es mit weiteren Plug-Ins zu analysieren.

Es sollen Mechanismen für die Anbindung von Listen und Abbildern bereitgestellt werden. Zudem muss für spezifische Dateitypen die nachgelagerte Detailanalyse mittels Plug-In möglich sein.

Als Basis dient auch eine Master-Thesis in Forensik, welche unter Anderem den Vergleich von virtuellen Datenträgern beleuchtet.

Es geht im Rahmen dieser Arbeit weder um die formal korrekte Behandlung von Datenträgern (chain of custody) noch geht es um die Analyse von Malware oder das Wiederherstellen von gelöschten Daten.

A.2.1. Mögliche Tätigkeiten

- Studium der verfügbaren Tools und White- resp. Blacklists

- Studium vorhandener Unterlagen zu Triage-Prozessen und Vergleich von VM-Disks
- Erarbeitung einer Übersicht und Erklärung von bekannten Vorgehensweisen für Untersuchungen
- Übersicht von möglichen Infektionsarten und wie man diese detektieren könnte
- Erarbeitung von Visuals für mögliche Reports und Teilreports
- Design eines offenen Frameworks für die Untersuchung und Vergleich von Disks evt. Memory
- Definition des Mindestumfang im Rahmen der Arbeit
- Programmierung eines Frameworks/Toolkit

A.2.2. Benötigtes Skillset

- Linux Kenntnisse (Kali Distribution, mount, fdisk, dd, libguestfs, dmesg)
- Kenntnis über Dateisysteme (physisch und virtuell, NTFS, HFS+, LVM, RAID, NFS, CIFS)
- Kenntnis zu Festplattenverschlüsselung
- Python Kenntnisse, Software Engineering Prozesse und Patterns, RUP, SCRUM, GIT
- Malware Analyse (clamav, ICAP, volatility, yara, IOCs)
- Blacklist, Whitelist verfahren

A.3. Vorgehen

Im Rahmen der allgemeinen Richtlinien zur Durchführung von Studien- und Bachelorarbeiten gemäss eigenem Projektmanagementplan. Dieser Projektmanagementplan ist als Erstes von den Studenten zu erstellen und enthält insbesondere:

- Die Beschreibung des dem Projektcharakter angepassten Vorgehensmodells.
- Eine erste Aufteilung der Aufgabe in gemeinsam und einzeln zu bearbeitende Teile unter Berücksichtigung der vorgegebenen Teilaspekte. Die genaue Aufteilung muss spätestens nach der Technologiestudie erfolgen.
- Den Projektplan (Zeitplan) und die Meilensteine.

A.4. Randbedingungen

- Wo möglich sollten Open Source Produkte eingesetzt werden
- Als Betriebssystem soll die Kali Linux Distribution zum Einsatz kommen.
- Das Framework muss in Python geschrieben werden.
- Der Einsatz des Toolkit muss offline (ohne Netzwerkverbindung) möglich sein

- Black und Whitelists sollten mit einem Updatemechanismus aktualisiert werden können
- Die Dokumentation wird in Englisch erwartet.

A.5. Infrastruktur

Die Arbeiten werden auf den Rechnern der Studenten durchgeführt. Zusätzlich benötigte Software oder Hardware wird bei Bedarf und nach Rücksprache mit Compass Security zur Verfügung gestellt.

A.6. Erwartete Resultate

A.6.1. In elektronischer Form

- Installationskit (alle Dateien für eine Installation und Installationsanweisung)
- kompletter Source Code
- komplette Software Dokumentation (Use Cases, Klassenmodell, Sequenzdiagramme usw. in UML)
- komplettes Use Cases und Success Scenarien resp. Musterlösungen
- alle Dokumente
- alle Protokolle

A.6.2. Auf Papier

Gemäss der Anleitung der HSR:

<https://www.hsr.ch/Allgemeine-Infos-Diplom-Bach.4418.0.html>

Es muss aus den abgegebenen Dokumenten klar hervorgehen, wer für welchen Teil der Arbeit und der Dokumentation verantwortlich war (detaillierte Zeiterfassung).

A.7. Termine

A.7.1. Start / Ende

Gemäss Vorgabe der HSR:

<https://www.hsr.ch/Termine-Bachelor-und-Studiena.5142.0.html>

- | | |
|---------------|---|
| 22.2.16 | Beginn der Bachelorarbeit, Ausgabe der Aufgabenstellung durch die Betreuer. |
| xx.5.16 | Fotoshooting. Genauere Angaben erteilt die Kommunikationsstelle rechtzeitig. |
| 08.6.16 | Die Studierenden geben den Abstract für die Diplomarbeit zur Kontrolle an ihren Betreuer/Examinator frei. Die Studierenden erhalten vorgängig vom Studiengangsekretariat die Aufforderung und die Zugangsdaten zur Online-Erfassung des Abstracts für die Broschüre. Die Studierenden senden per Email das A0-Poster zur Prüfung an ihren Examinator/Betreuer. Vorlagen sowie eine ausführliche Anleitung betreffend Dokumentation stehen unter den allgemeinen Infos Diplom-, Bachelor- und Studienarbeiten zur Verfügung. |
| 15.6.16 | Der Betreuer/Examinator gibt das Dokument mit dem korrekten und vollständigen Abstract für die Broschüre zur Weiterverarbeitung an das Studiengangsekretariat frei. Fertigstellung des A0-Posters bis 10.00 Uhr und Weiterleitung als PDF-Dokument an das Studiengangsekretariat. |
| 17.6.16 | Abgabe des Berichtes an den Betreuer bis 12.00 Uhr. |
| 17.6.16 | Präsentation der Bachelorarbeiten, 16 bis 20 Uhr |
| 08. - 26.8.16 | Mündliche BA-Prüfung |
| 30.9.16 | Bachelorfeier und Ausstellung der Bachelorarbeiten, ab 09.00 Uhr |

A.7.2. Zeitplan und Meilensteine

Zeitplan und Meilensteine für das Projekt sind von den Studenten selber zu erarbeiten und zusammen mit dem Projektmanagementplan abzuliefern. Die Meilensteine sind bindend. Der erste Meilenstein ist vorgegeben. Mit den Betreuern werden regelmässige Sitzungen zur Fortschrittskontrolle durchgeführt.

Abgabetermin Projektmanagementplan in der Woche vom 7. März 2015

A.8. Betreuung

Die Arbeiten werden durch Cyrill Brunschwiler betreut. Der Gegenleser ist noch nicht bestimmt.

A.8.1. Kontakt

Cyrill Brunschwiler, Managing Director, Compass Security Schweiz AG Werkstrasse 20, 8645 Jona, Switzerland Tel: +41 55 214 41 73 <http://www.csnc.ch/>

Please submit files securely using Filebox

<https://secure.csnc.ch/inbox/DshpdeDVVP68Pp>

A.9. Referenzen

- Vorgaben HSR
<https://www.hsr.ch/Allgemeine-Infos-Diplom-Bach.4418.0.html>
- Computer Forensics Field Triage Process Model
<http://www.digitalforensics-conference.org/CFFTPM/CDFSL-proceedings2006-CFFTPM.pdf>
- Forensic Tools List
<http://forensicswiki.org/wiki/Tools>
- YARA - the pattern matching swiss knife for malware researchers
<https://plusvic.github.io/yara/>
- LOKI - Indicators Of Compromise Scanner
<http://www.darknet.org.uk/2016/01/loki-indicators-compromise-scanner/>
- THOR Scanner (LOKI License Version)
<https://www.bsk-consulting.de/apt-scanner-thor/>
- The OpenIOC (Open Indicators of Compromise)
<http://www.openioc.org/>
- Using IOC (Indicators of Compromise) in Malware Forensics
<https://www.sans.org/reading-room/whitepapers/forensics/ioc-indicators-compromise-malware-forensics-34200>
- Forensic Triage for Mobile Phones with DECODE
https://www.usenix.org/legacy/event/sec11/tech/full_papers/Walls.pdf
- Windows Sysinternals Sigcheck
<https://technet.microsoft.com/de-ch/sysinternals/bb897441.aspx>
- SANS Blog on Triage
<https://digital-forensics.sans.org/blog/2011/05/26/digital-forensics-case-leads-triage-live-incident-response-memory-forensics>

A.10. Unterschriften

Jona, 25. Februar 2015



Cyrill Brunschwiler



Luca Tännler



Mathias Vetsch

B. Golden Image Documents

B.1. Module: Golden Image - Options

B.1.1. Introduction

This document shows the research about possible options on how to implement the "Golden Image"-Feature in Autopsy. In the end there will be a conclusion with the best option from the view of the Author.

The Golden Image

In this chapter I will give you a short introduction about the golden image: What is it? Why do we need it? What benefits will it bring? The golden image is a regular image, like any other. It doesn't contain any special features or characteristics. The reason why it's called "golden image" is, because it is a fresh installation of the operating system (in our project it will be a fresh Windows installation). That means, the image won't contain any malware and also no user-installed software. Now imagine you have an image of a Windows-partition which might be contaminated. In the real world, such an image can easily have a size of 200GB, 500GB, 1TB or even more. The larger the partition is and the more files and software is saved on it, the longer it takes to analyse it. What we want to achieve with the golden image module is, to reduce the scope of the potentially contaminated image which the analyst has to analyse. The golden image will compare its "safe" files, hashes and more with the equivalents from the other image and exclude all known safe parts from the contaminated image. This will reduce the workload of parts which the analyst or software has to scan for malware and can save a lot of time.

B.1.2. Workflow: The Idea

The workflow in Autopsy looks basically like this:

1. Create a case

2. Select Image(s)
3. Run Ingest modules
4. Run analysis modules
5. Run post-processing / reporting modules

Now the initial idea is, that the analyst can add the golden image during **Step 2** and somehow "mark" it as golden image (so the software knows, which of the images the clean one is). Now in **step 3** and **step 4**, the modules can make use of the golden image.

B.1.3. Option 1

One option for the implementation of the Golden-Image feature is, that while the user adds a new data source, he has the possibility to also add a golden image. The golden image will be tagged as golden image and has its own data source type. In the selection and configuration of the ingest modules, there will be a module called "Golden Image". The module "Golden Image" will run through all files of the golden image, compare them with the normal image and tag them as "changed" or as "safe". Additionally a list will be created with all the missing files - files which are on the golden image, but not on the normal image. This module must be run before the others so they either have access to the tags or that a data-filter can be applied.

Pros:

- The user can add the golden image while adding the dirty image - all in one step

Cons:

- Changes in the Autopsy Core must be done
- The user needs an Autopsy version with the changes in the core
- It's only possible to add a golden image while importing a dirty image
- File-Hash generation must be done with every import (for the golden image)

B.1.4. Option 2

Another option on how to implement the Golden-Image feature is following. The following enumeration describes the workflow:

1. The user adds the golden image:
He adds the golden image as a normal image, like any other.
2. The user adds the dirty image OR wants to re-run the ingest-modules on an existing dirty image.

3. In the selection of the ingest modules that the user wants to run, he selects the module "Golden Image". In the configuration box, the user can select the golden image (There will be a dropdown menu in which all other images which were added to the case are listed - the user needs to know, which of them the golden image is.) There also will be a button called "Advanced.." which will open a new panel with advanced configurations for the module (See Step 3.a for further information).
 - a) A list with all ingest-modules is shown. The user selects the modules which he wants to run over the filtered data after the comparison and tagging between golden image and dirty image is done. Further on, the user can also select, which files should be used for the ingest modules. He has following options: "untagged files", "changed files" and "safe files".
4. After the configuration, the Golden Image module will be run. First it compares and tags the files against the dirty image and after that's done, the ingest module will be run over the filtered data.
5. In Autopsy, there will be a new panel, where the user can see and filter the files by the tags or untagged files.

Pros:

- The Golden Image feature is a simple module
 - Easy to install
 - No special Autopsy version required
- The imported golden image can be used with several dirty images
- The golden image feature can be used with already imported dirty images
- File-Hashes must be created only once for the golden image

Cons:

- This process requires multiple steps:
 - Import golden image
 - Import dirty image / Configure golden image module

B.1.5. Option 3

Option 3 has similarities to option 2 regarding the first steps:

1. The user adds a golden image as data source in Autopsy

2. The user adds an image which he would like to analyse.

Now in the step where the user selects the ingest modules which he likes to run over the image. In the selection, he selects the module "Golden Image" and in the settings, he selects the golden image from a list. Further on, he selects all the other ingest modules from the module-list which he would like to run over the reduced dataset from after the golden-image comparison. At first, the golden image module will be run, which will tag the files in the other image. After the execution, all the other selected ingest modules will be run, but only on the reduced dataset.

Pros:

- The imported golden image can be used with several dirty images
- The golden image feature can be used with already imported dirty images
- File-Hashes must be created only once for the golden image

Cons:

- This process requires multiple steps:
 - Import golden image
 - Import dirty image / Configure golden image module
- Changes in the Autopsy core need to be done
- The user requires a special version of Autopsy with the golden image features

B.1.6. Conclusion

We have decided, that Option 2 is the best option. It has several benefits such as:

- The data source / run-ingest-modules wizard must not be changed
- The golden image feature will be an ingest module, no changes in the core need to be made
- It's not necessary to implement a special data source type for golden images
- Compared to the other options it has the most pros and the fewest cons

Option 3 is very similar to option 2, but the reason why we decided not to take option 3 is, because we would have to make changes in the integration wizard (a core-component of Autopsy) while in option 2 we only have to implement a simple-to-integrate module. That is a clear benefit of option 2 against option 3.

Option 1 is the most time consuming option and also there we would have to make changes in the Autopsy-core. Additionally we would have to add 2 data sources at the same time. Fur-

ther on it could also increase the overall processing time if the user wants to run the golden image feature over several dirty images (Because of the repeating hash calculation).

B.2. Module: Golden Image - Planning

B.2.1. Introduction

This document describes, how the module "Golden Image" will be implemented. Several parts will be described such as the workflow, design decisions and process descriptions.

The Golden Image

The golden image is a regular image, like any other. It doesn't contain any special features or characteristics. The reason why it's called "golden image" is, because it is a fresh installation of the operating system (in our project it will be a fresh Windows installation). That means, the image won't contain any malware and also no user-installed software. Now imagine you have an image of a Windows-partition which might be contaminated. In the real world, such an image can easily have a size of 200GB, 500GB, 1TB or even more. The larger the partition is and the more files and software is saved on it, the longer it takes to analyse it. What we want to achieve with the golden image module is, to reduce the scope of the potentially contaminated image which the analyst has to analyse. The golden image will compare its "safe" files, hashes and more with the equivalents from the other image and exclude all known safe parts from the contaminated image. This will reduce the workload of parts which the analyst or software has to scan for malware and can save a lot of time. Before this documentation, several ideas and options on how to implement the "Golden Image" feature were created and described in the document "GoldenImageOptions_EN.docx". We decided to do the implementation based on **Option 2** which is described in the before mentioned document.

B.2.2. General Workflow

Following is a superficial workflow-description of the use of the "Golden Image" module:

1. The user adds the golden image:
He adds the golden image as a normal image, like any other.
2. The user adds the dirty image OR wants to re-run the ingest-modules on an existing dirty image.

3. In the selection of the ingest modules that the user wants to run, he selects the module "Golden Image". In the configuration box, the user selects the golden image (There will be a dropdown menu in which all other images which were added to the case are listed - the user needs to know, which of them the golden image is.) There also will be a button called "**Advanced.**" which will open a new panel with advanced configurations for the module (See Step 3.a for further information).
 - a) A list with all ingest-modules is shown. The user selects the modules which he wants to run over the filtered data after the comparison and tagging between golden image and dirty image is done. Further on, the user can also select, which files should be used for the ingest modules. He has following options: "untagged files", "changed files" and "safe files".
4. After the configuration, the Golden Image module will be run. First it compares and tags the files against the dirty image and after that's done, the selected ingest modules will be run over the filtered data.
5. In Autopsy, there will be a new panel, where the user can see and filter the files by the tags or untagged files. All ingest module results will be displayed according to the modules implementation. This module won't have any influence on these results.

B.2.3. Module Type

The module will mainly be a "**Data-source-level Ingest Module**" according to the official description of the Autopsy documentation. More information can be found on following URL:

http://sleuthkit.org/autopsy/docs/api-docs/4.0/mod_ingest_page.html

The reason for this decision is, because in this module type a reference to the dirty image will be passed. This allows us to search the meta-data database of the dirty image, for the files on the golden image. Further information about this process can be found in the following chapters. Additionally the "Golden Image" module will contain a **result-viewer** module and a **data-viewer** module. More information about these Autopsy module types can be found on following URL:

<http://sleuthkit.org/autopsy/docs/api-docs/4.0/index.html>

Initial Workflow Steps

In order to compare a dirty image with a golden image, the user has to import a golden image as a regular data source in Autopsy. The user doesn't have to run any ingest modules over this image (unless he wants to). After he added the golden image, he either has to import the dirty image to Autopsy, or open the panel where he can re-run ingest modules over the dirty image if he already added a data source. In the following chapter "**Module Settings**" we will talk about the settings of the "Golden Image" module which the user has to/can set before running the module.

B.2.4. File Tagging and "Deleted Files"

During the comparison process in this module, files will be tagged. All file-tagging will happen on the dirty image. There are 4 types of files:

- Untagged files
- Changed files
- Safe files
- Deleted files

Untagged Files are files which weren't found on the golden image. **Changed Files** are files which are found on the golden image but differ from the one on the dirty image. These files will be marked on the dirty image as "Changed". **Safe Files** are files which are found on the golden image and weren't changed. These files will be marked on the dirty image as "Safe". **Deleted Files** are files which are found on the golden image but weren't found on the dirty image. These files will be added to the list "Deleted Files".

B.2.5. Module Settings

The "Golden Image" module will enable the user to set following settings. Some of the settings are optional, others are required to be set by the user.

- **The Golden Image:** The user **has to** specify the golden image. There will be a dropdown in which the user can select the golden image out of all images which were imported to the current Autopsy-case. The dirty image won't be in the list to make sure that the dirty image won't be compared with itself. **MUST-Setting.**
- **Ingest Modules:** The user can select out of a list the ingest modules that he wants to be run on the resulting dataset after the comparison and tagging. The module "Golden Image" won't be shown in the selection to avoid loops. **OPTIONAL-Setting.**

- **Dataset Filter:** The user can select, which data he wants to be scanned/processed by the Ingest Modules. He will have the following options:
 - "Untagged Files": These are all files, which are not existing on the golden image.
 - "Changed Files": These are all files, which are existing on the golden image, but which were changed.
 - "Safe Files": These are all files, which are existing on the golden image and weren't changed. (Usually these files don't have to be analysed since they should be safe)

The selection of the Golden Image will be placed in the general settings panel of the Golden Image. (See following wireframe "Wireframe: Ingest Module Selection"). The settings for the Ingest Modules and Dataset Filter will be in a separate panel which is callable through opening the advanced-setting panel. (See following wireframe "Wireframe: Advanced Settings").

Wireframe: Ingest Module Selection

Following wireframe shows the ingest module selection which is shown when adding a new data source or when re-running ingest modules of already imported data sources. This panel already exists in the Autopsy core, the interesting part is on the right side where the general settings of the "Golden Image" module is displayed. When the "Golden Image" module entry is focused, the general settings panel of this module is shown. It provides a short description about module and also a Dropdown-menu from which the user can select a golden image. On the lower right side there is a button called "Advanced". This will open a new panel with advanced settings for the "Golden Image" module. See the following chapter "Wireframe: Advanced Settings" for more information.

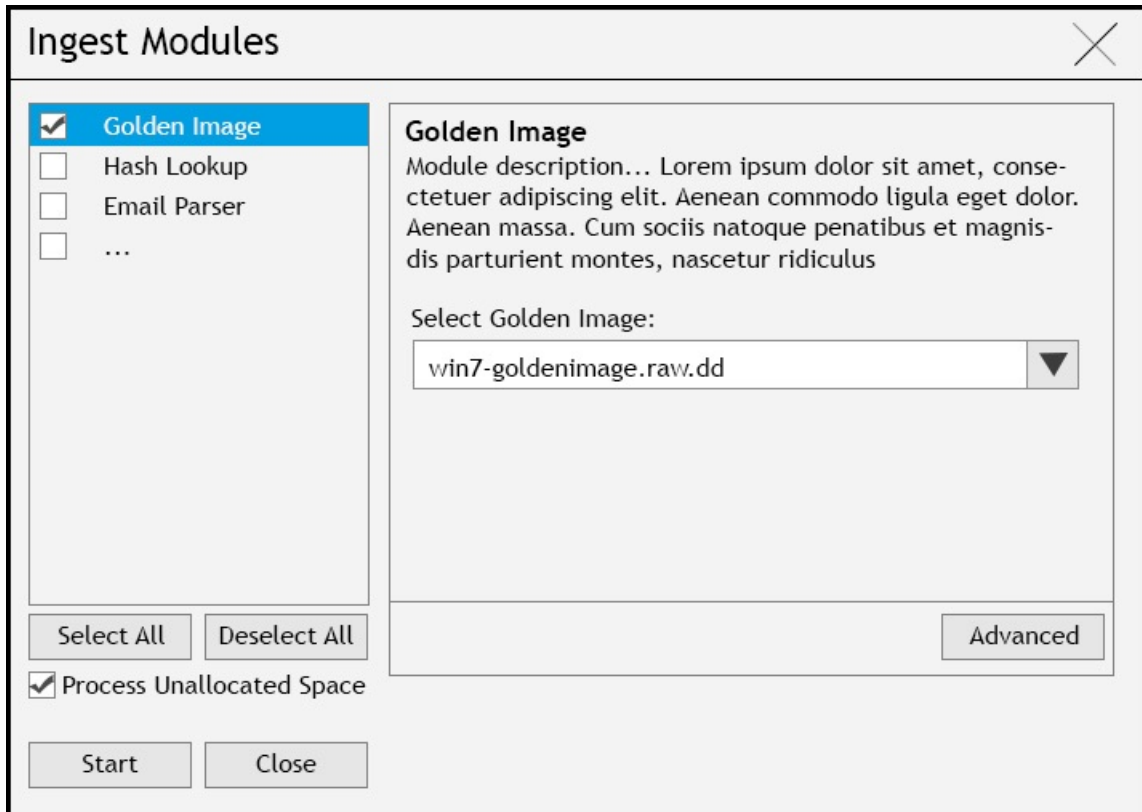


Figure B.1.: Wireframe - Ingest Module Selection

Wireframe: Advanced Settings

Following is a wireframe from the panel with advanced settings for the "Golden Image" module. **Ingest Modules:** The user can select the ingest modules which he wants to be run when the comparison and tagging process is done. **Dataset Filter:** The user can select which files he wants to be analysed by the ingest modules.

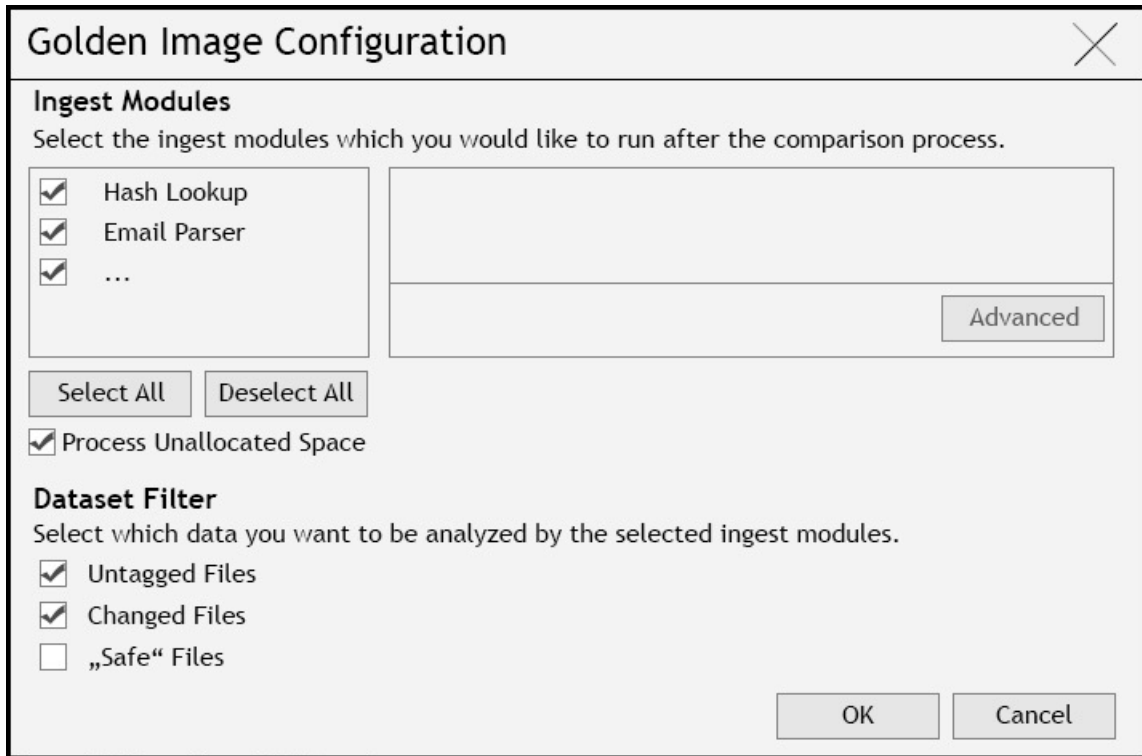


Figure B.2.: Wireframe - Advanced Settings

B.2.6. Golden Image View

There will be a new view in Autopsy for the Golden Image module. In this view, the user can search for the filtered files. The user can filter the files by the tags "safe", "changed" and also display the untagged files. Additionally there will be a list of the deleted files.

B.2.7. Golden Image Workflow

This chapter describes the workflow of the "Golden Image" module when it's run. There are two main-steps in this process:

- Comparison and Tagging
- Run Ingest Modules

Comparison and Tagging

In this step, the module runs through all files in the golden image. Each file will be checked against the dirty image. It will be checked if it exists, got changed or is the same. Following flow chart describes the process of each file.

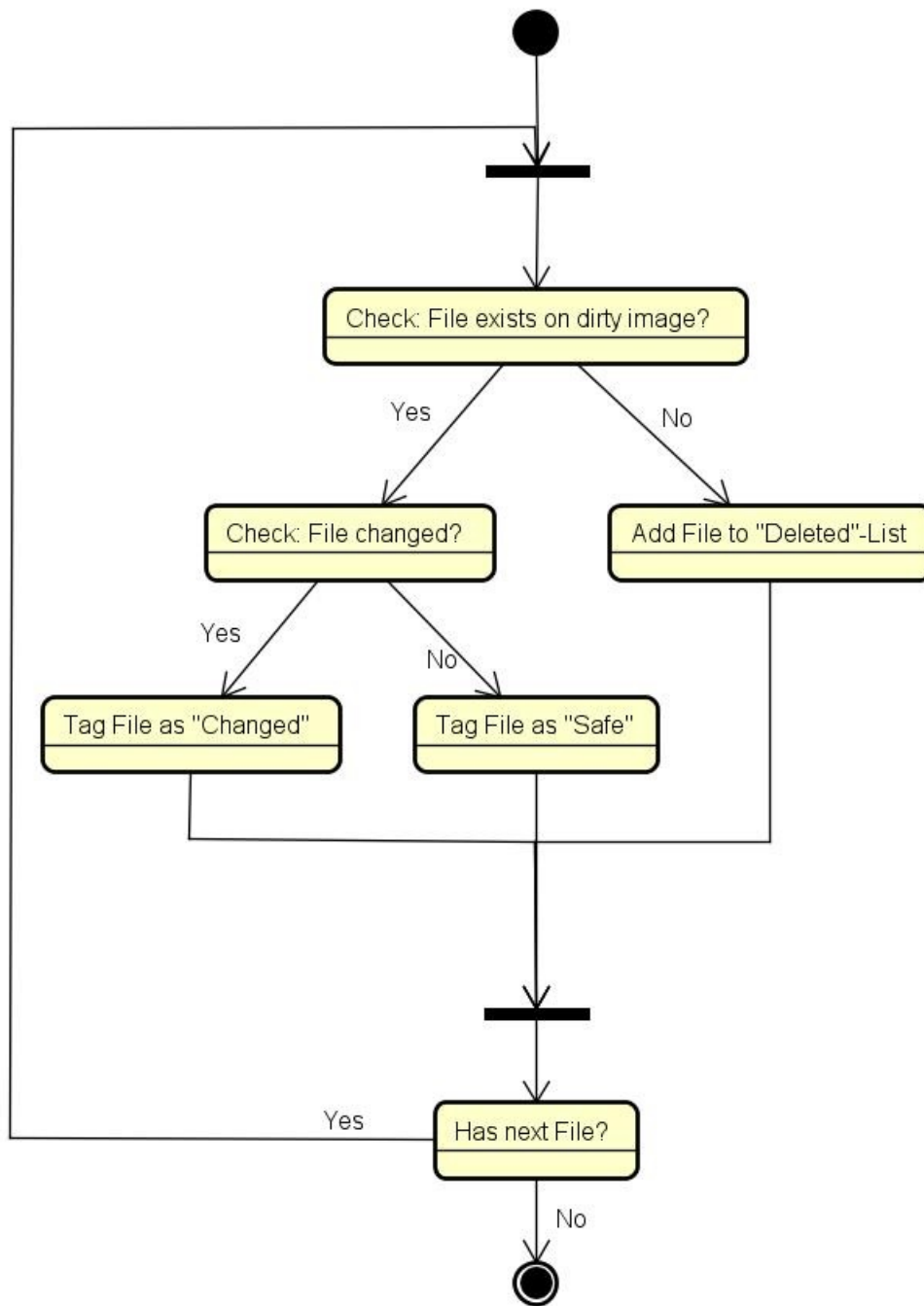


Figure B.3.: Golden Image - Flow Chart

Run Ingest Modules

After the golden image comparison and file tagging is done, the initially selected and configured ingest modules will be run. The dataset which will be processed by the ingest modules, depends on the initial configuration made by the user.

C. Installation Guide

The target platform is Kali Linux. You have to install Oracle JDK 8 before you can use autopsy properly. If you run autopsy with the preinstalled OpenJDK, the software will not work properly.

To install autopsy on Kali Linux, the delivered Debian package can be used. This package contains all features including the developed Modules. Autopsy can be installed and started with the following commands:

```
1 $ apt-get install -y ./autopsy4.deb
2 $ # run autopsy:
3 $ autopsy4
```

Listing C.1: Installation of autopsy on Kali linux

C.1. Plugin installation

To install the Autopsy plugins on a different Installation, you have to open Autopsy. In Options -> Plugins -> Downloaded you can select the *.nbm files on your disk.

Select all plugins you want to install, and press install. Now you can use the selected plugins in your Autopsy projects.

D. User Guide

D.1. Add a Bitlocker Data Source to Autopsy

To add a Bitlocker encrypted volume to an autopsy case use the standard add data source wizard. Select a disk image or a local disk containing a Bitlocker encrypted volume (fig. D.1).

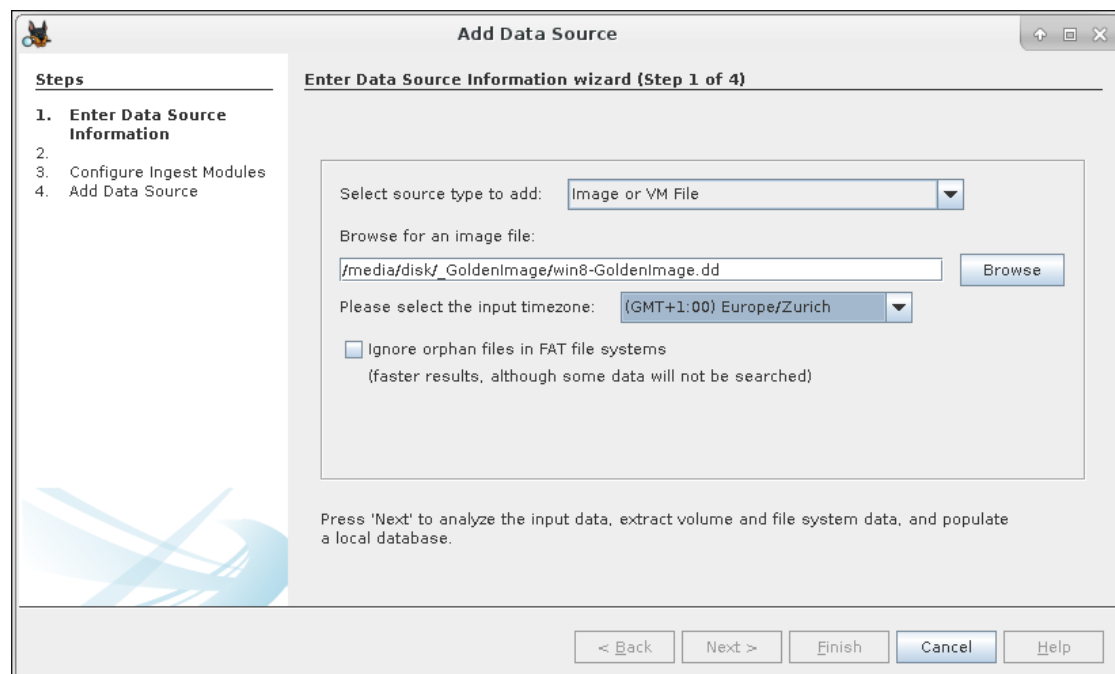


Figure D.1.: Screenshot of add data source wizard

The decryption module detects all Bitlocker encrypted volumes and adds a configuration panel to the wizard. Choose one of the three key types and enter the key. Press next (fig. D.2).

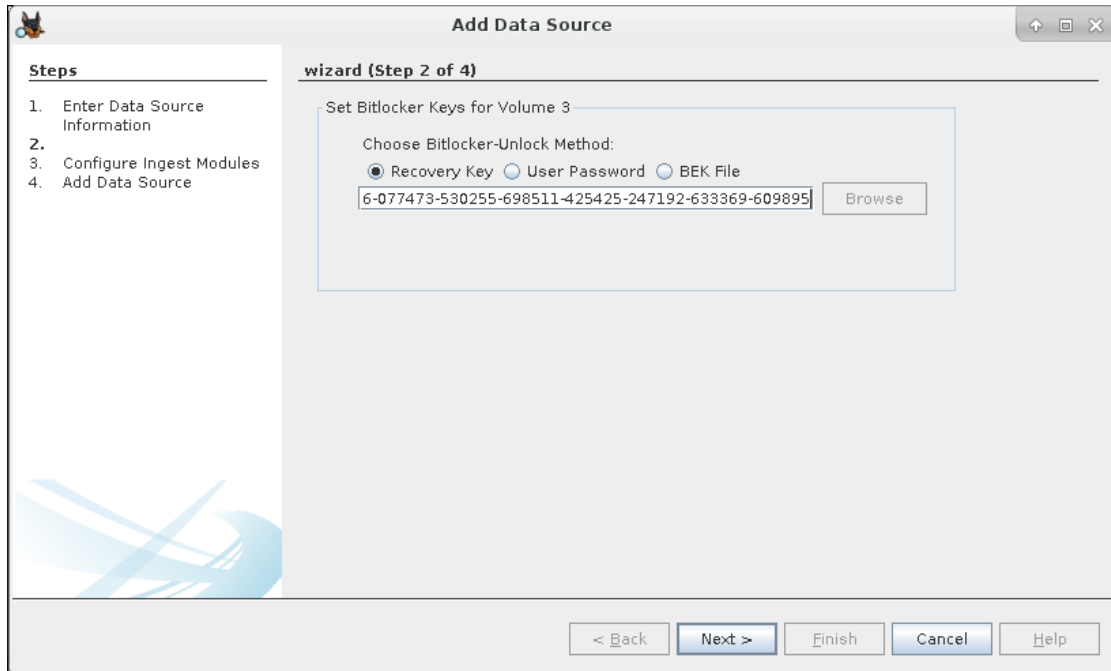


Figure D.2.: Configure key for encrypted volume in Data Source Wizard

The wizard requires some time to add all meta-data information from the data source to the case database. After this step, the encrypted volume is usable as additional data source in the autopsy case (fig. D.3).

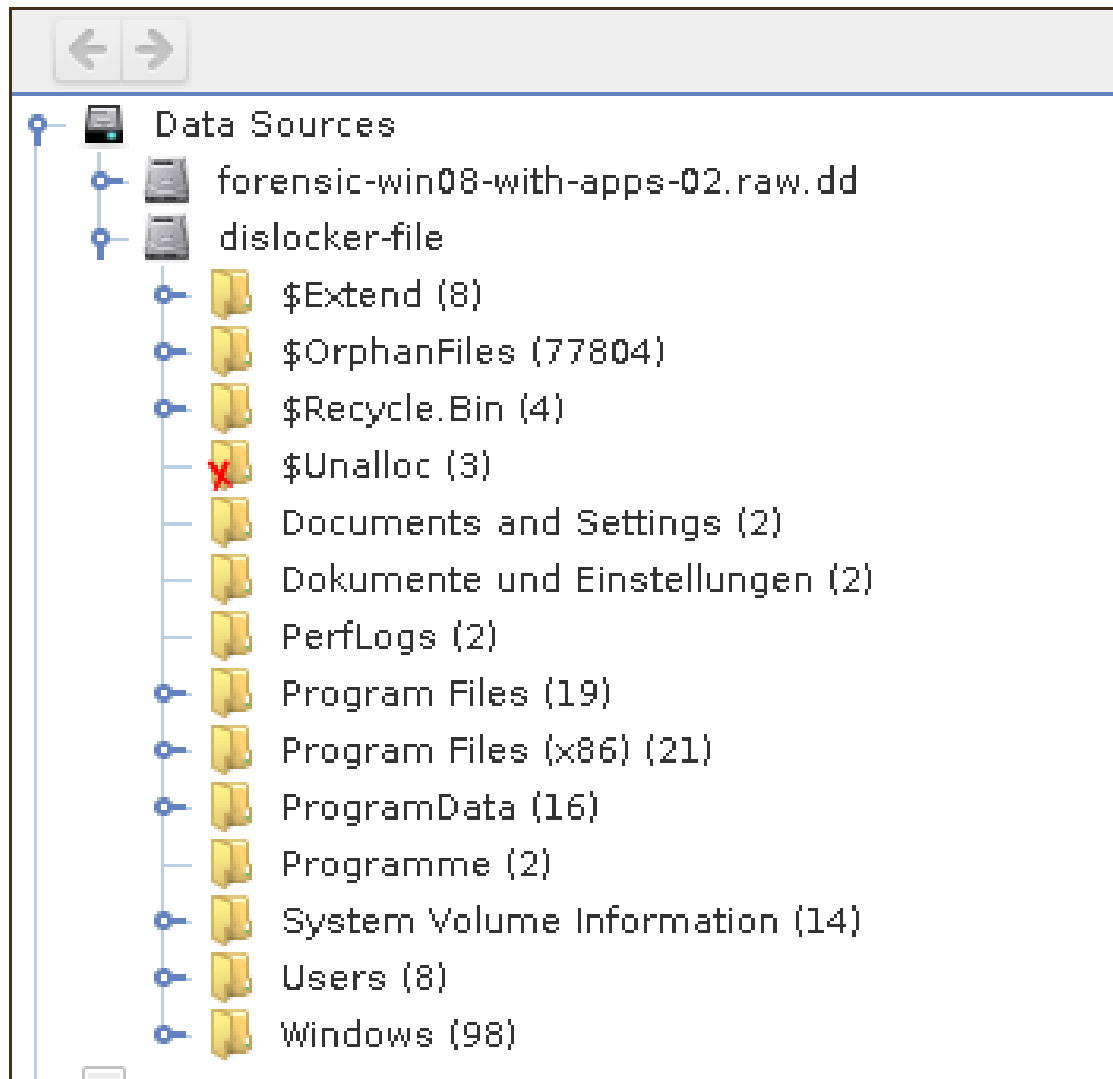


Figure D.3.: Autopsy project including a Bitlocker encrypted volume

D.2. Update Hash Sets

The Autopsy options panel (fig. D.4) contains a register for Hash Databases. Using the "update hashset" button, you can choose which hash sets you want to download (fig. D.5).

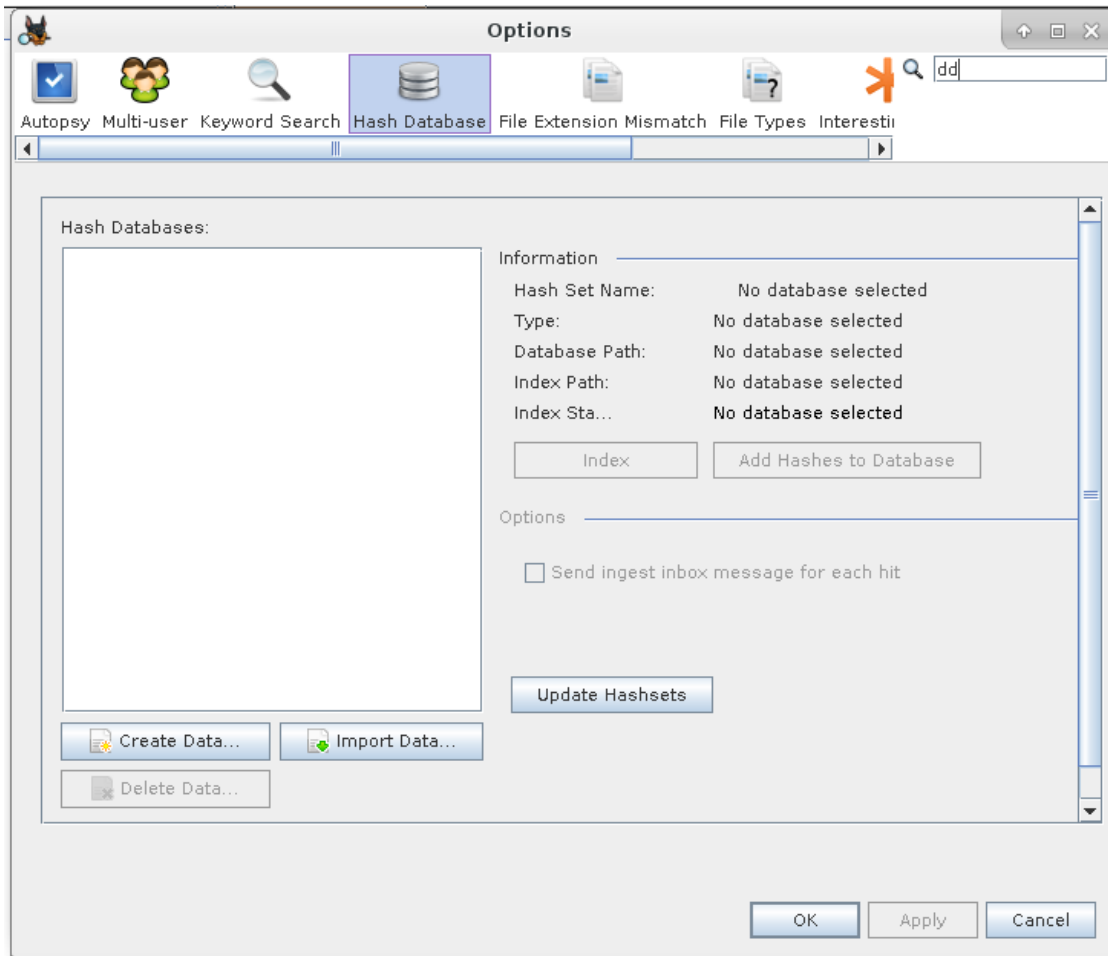


Figure D.4.: Hash Database configuration panel

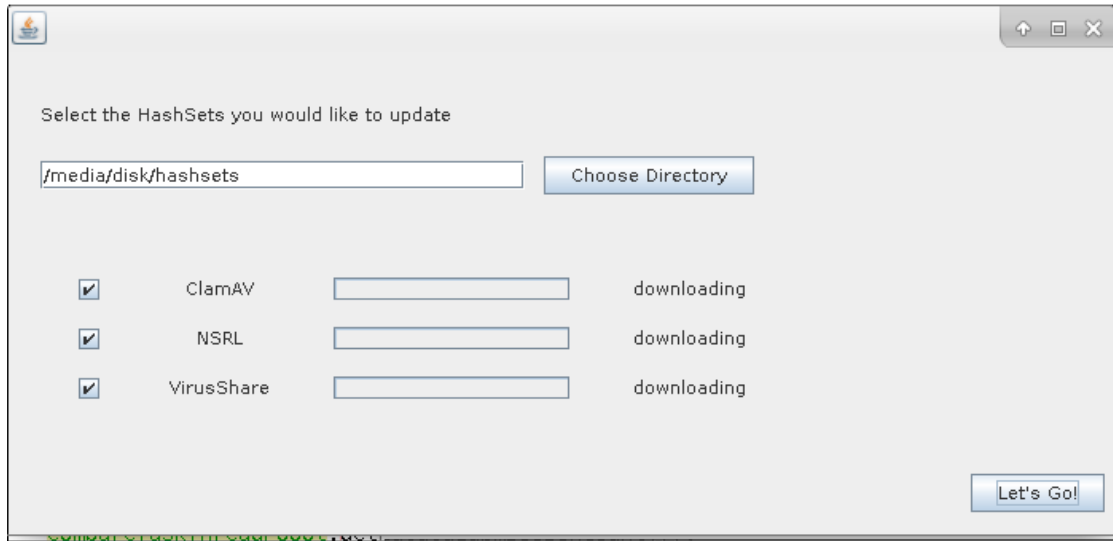


Figure D.5.: Wizard to automatically download hash sets

Figure D.6 shows the the hash database panel with the included hash sets.

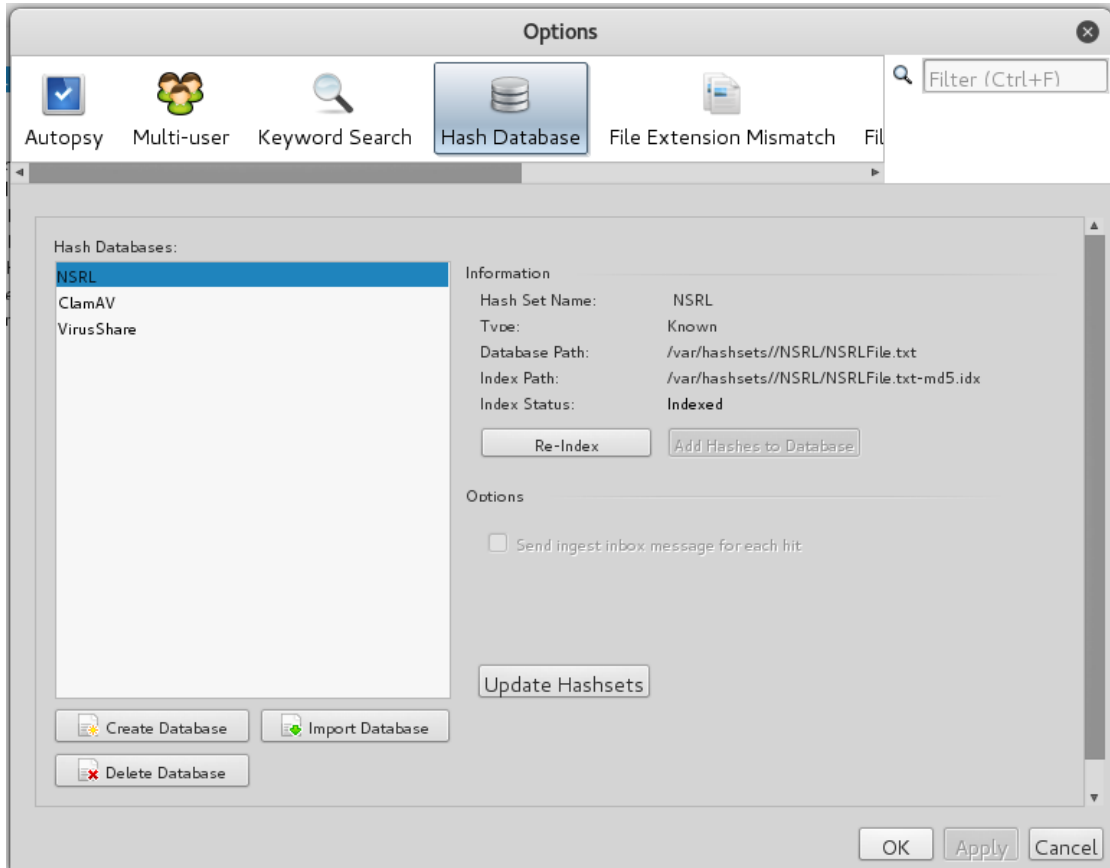


Figure D.6.: Hash Database configuration panel with downloaded hash sets

D.3. Verify AuthentiCode Signatures

The AuthentiCode Ingest Module is launched like every other ingest module in Autopsy. The user configures with which hash types he wants the module to compute the files on the data source to. (fig. D.7). The modules default is to compute SHA-1 and SHA-256 hashes. This delivered enough information from the images for the benchmarking.

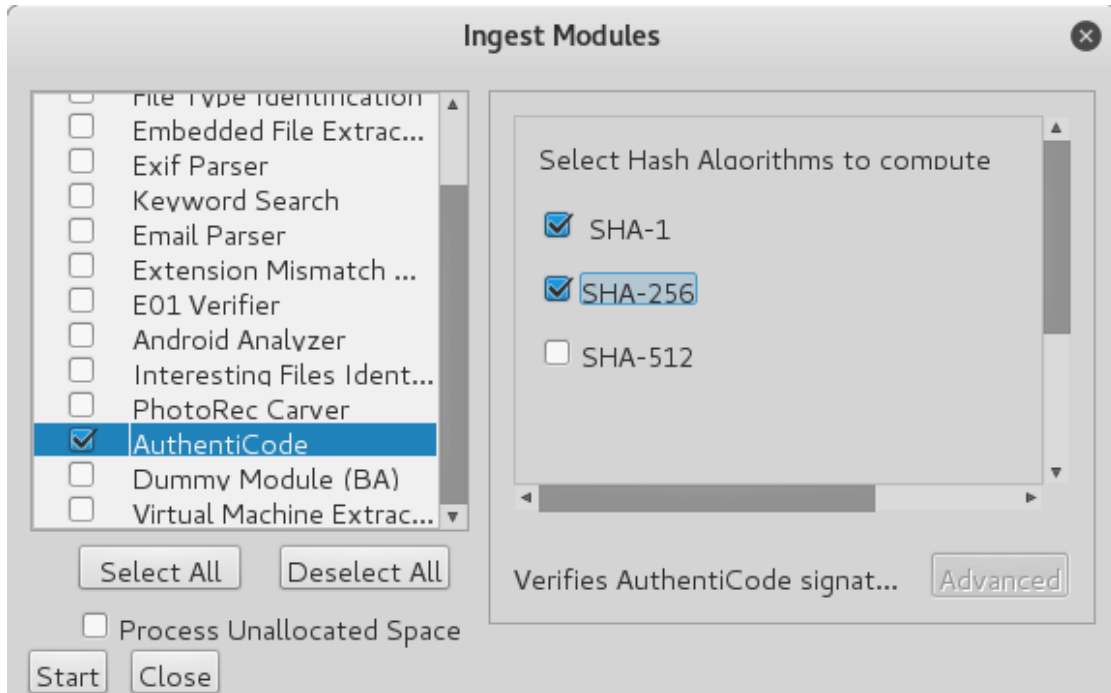


Figure D.7.: AuthentiCode configuration options

After the module has finished, the tagged files are visible in the autopsy project. The tag name is the subject of the signer certificate (fig. D.8).

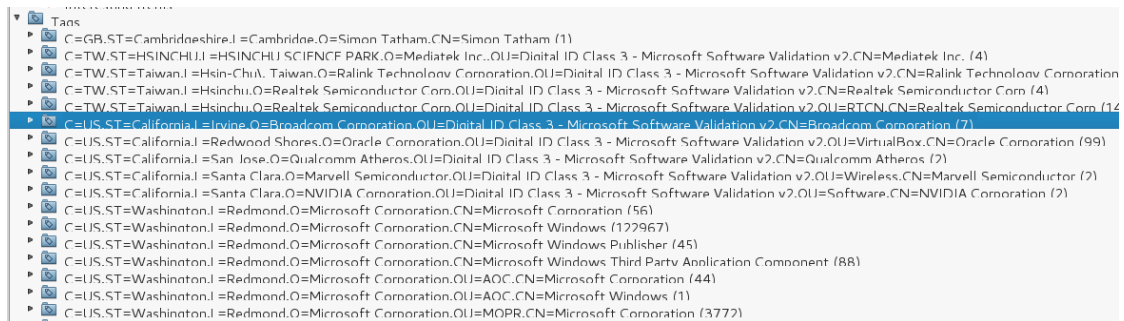


Figure D.8.: File tags created by AuthentiCode ingest module

To see more information about the AuthentiCode signature, you can use the data content viewer tab (fig. D.9).

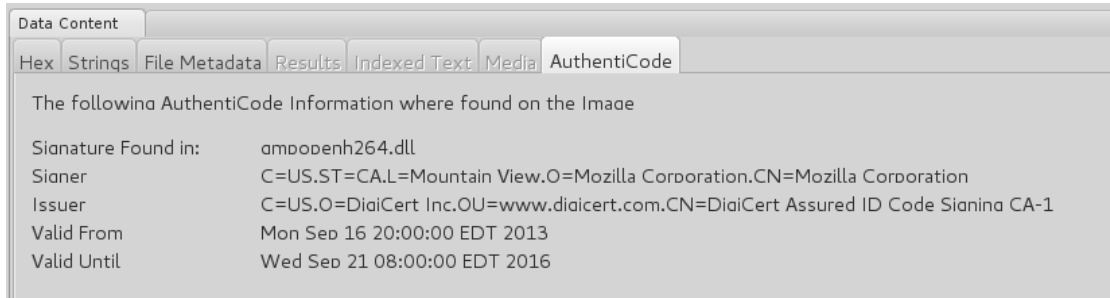


Figure D.9.: AuthentiCode Data Content Viewer Tab

D.4. VirusTotal Online Checker

This is a standard file ingest module. To use this module, you have to obtain a personal API-key for the VirusTotal API. After successful registration on <http://www.virustotal.com> you automatically receive the key.

Figure D.10 displays the API-key configuration panel. The module will not work if the

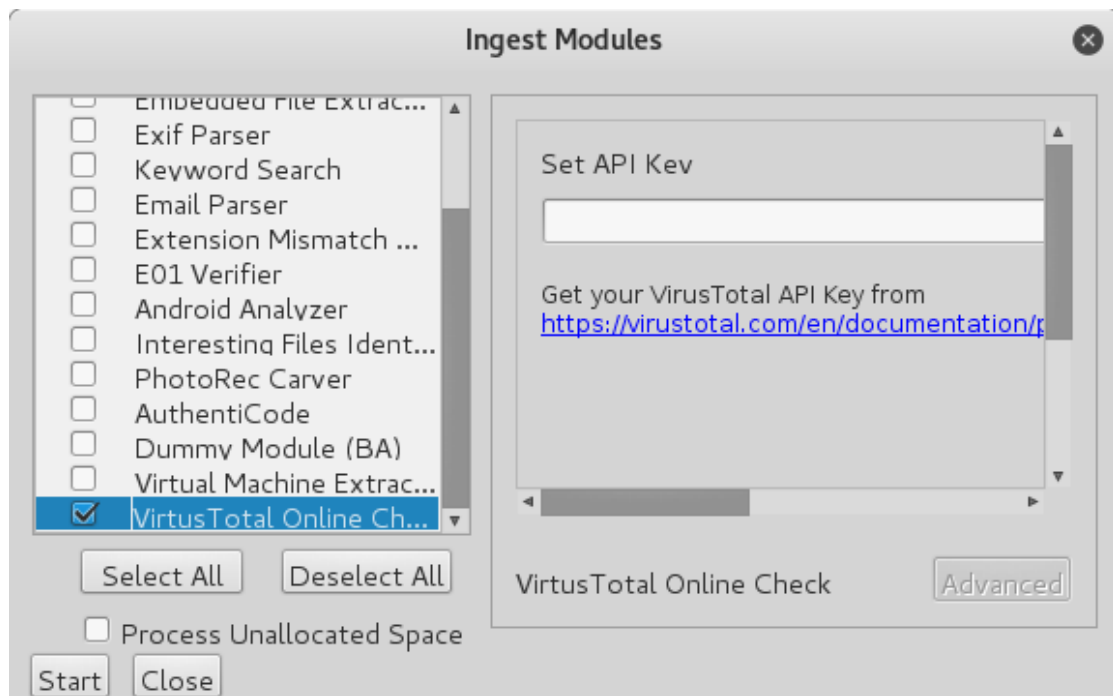


Figure D.10.: AuthentiCode Data Content Viewer Tab

D.5. Golden Image

In order to run the Golden Image ingest module, you need to have two data sources imported to Autopsy:

- **The dirty image:** The image with possible infection
- **The golden image:** The image which is known not to be infected by any kind of malware

To run the module, you need to open the run-ingest-modules panel on the dirty image. Select the module **Golden Image**.

When you click on the Golden Image entry in the ingest module list, the settings panel will appear on the right side of the window. Select the golden image - which you want to run the dirty image against - from the dropdown in the settings panel as shown in the screenshot below.

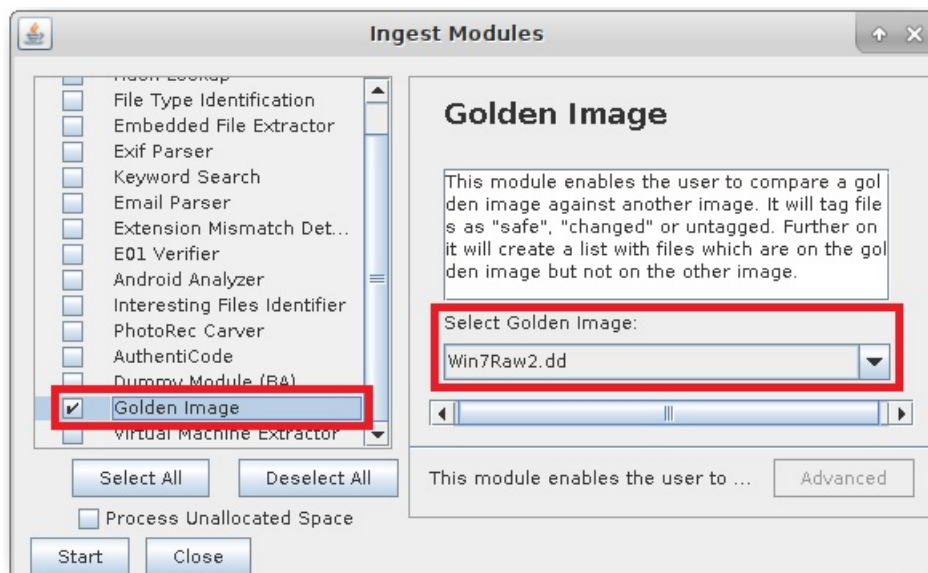


Figure D.11.: Golden Image - Configuration

After you start the ingest process, you will see its progress on the lower corner of the main window of Autopsy.

When the Golden Image module is done processing you can view the tagged files. To open the file lists of the tagged files, open the navigation "Tags -> [Tagname] -> File Tags" in the left navigation panel.

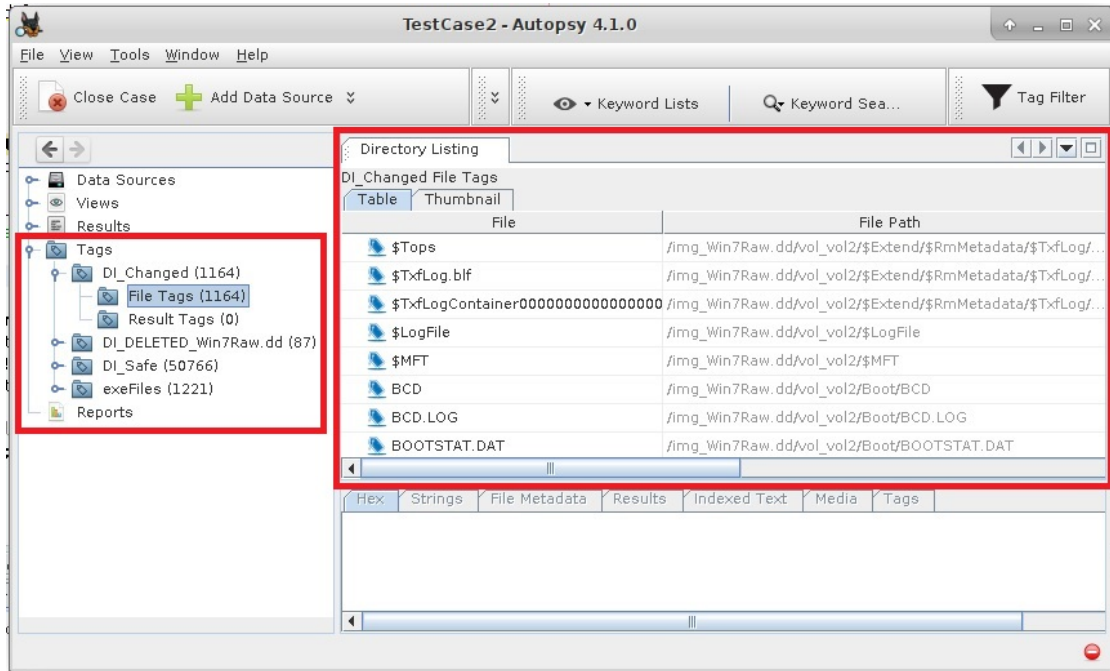


Figure D.12.: Golden Image - Show tagged files

D.6. Tag Filter

Opening The Tag Filter

In the upper right corner, there is a button called "Tag Filter" as shown in the screenshot below. By clicking on it, a new window will open with the filter configuration panel.

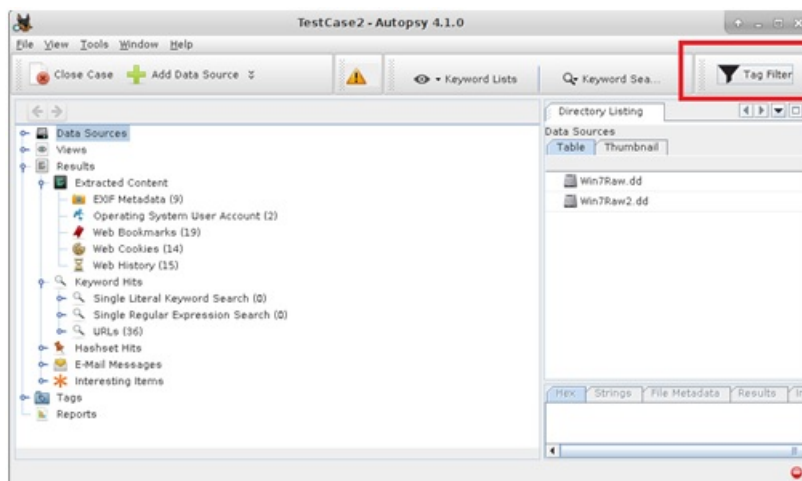



Figure D.13.: Tag Filter - Open

Tag Filter Configuration Panel

Following screenshot shows the Tag Filter configuration panel.

- **Button "Add Filter"**: By clicking on this button, a new filter will be added to the end of the list.
- **Button "Add Filter Group"**: By clicking on this button, a new filter group will be added to the end of the list. By clicking on the plus-icon () on the right side of the filter group, a new filter can be added to this filter group.
- **Dropdown Data Source**: The user can specify, if he only wants to search for files on a specific data source. If none is selected, the filter will search for files in all data sources contained in the current Autopsy case.
- **Button "Filter"**: By clicking on this button, the filter will be applied and matching files are being searched. The Tag Filter configuration window will close and a list with the matched files will be displayed to the user.

The filters / filter groups are meant to be read top-down. The combination-operator (AND / OR) is connected to the previous filter in the list.

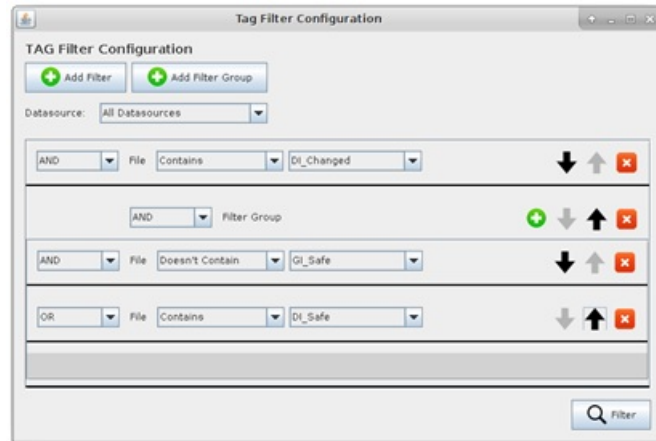





Figure D.14.: Tag Filter - Configuration Panel

Filter Entry

Following screenshot shows a filter entry within the Tag Filter configuration panel:



Figure D.15.: Filter Entry

- **Dropdown "Combination-operator":** In the first dropdown on the left, you can specify the combination-operator AND / OR.
- **Dropdown "Negation-operator":** In the second dropdown on the left, you can specify the negation-operator Contains / Doesn't contain.
- **Dropdown "Tag":** In the third dropdown on the left, you can specify the tag of which this filter is about. This dropdown dynamically shows all tags contained in the current Autopsy case.
-  This button enables you to move the filter down in the list.
-  This button enables you to move the filter up in the list.
-  This button enables you to delete the filter from the list.

When you are done with setting up your filter, click on the button "**Filter**" in the lower right corner.

Tag Filter File List

After creating and applying a filter, all matched files will be displayed within a table as following screenshot shows:

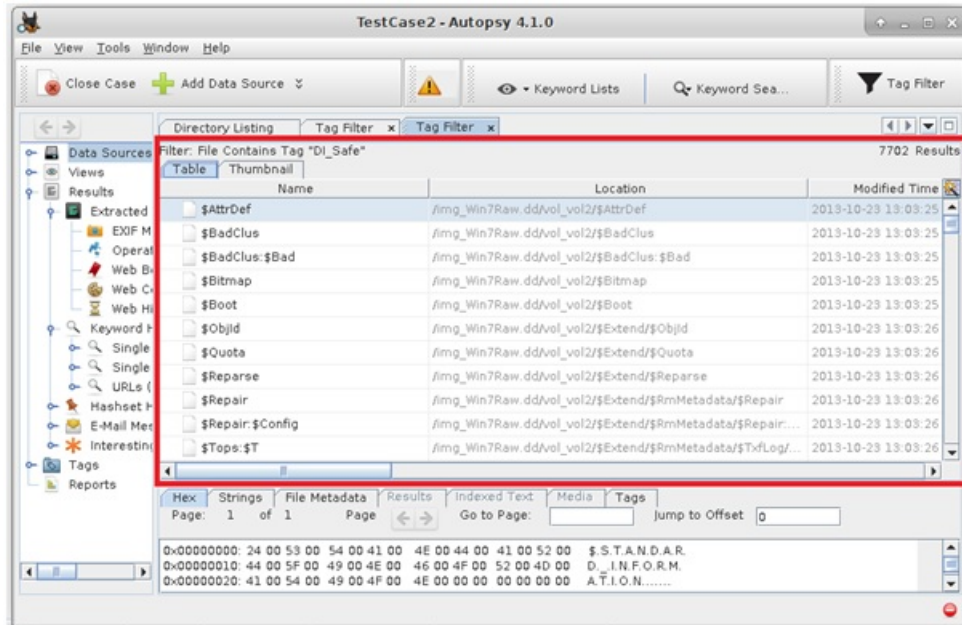


Figure D.16.: Tag Filter - File List

E. Analysis of Forensic Tool-kits

This chapter gives an overview about existing open source forensic tool kits.

E.1. Open Source Forensic Frameworks

E.1.1. MantaRay



Figure E.1.: MantaRay Forensics

Project website: <http://mantarayforensics.com/category/mantaray/>

MantaRay[21] is a forensic tool designed to automate the forensic processing of images with existing open source tools. This tool kit comes with a graphical wizard. The wizard allows you to run several modules/tools on an Image. Every Module creates an output file in a output directory with detailed information about the results of the processing.

This tool meets plenty of our requirements for our forensic tool kit:

- Open Source
- Modular architecture
- Supports several image formats

There are also some downsides which will have a tremendous effect on our decision:

- Not actively developed anymore
- Installers do not work due to dependency problems.

The project is hosted on GitHub[22]. A compiled version can be downloaded directly on their website[20].

E.1.2. The Sleuth Kit



Figure E.2.: The Sleuth Kit logo

Project website: <http://sleuthkit.org/sleuthkit/>

The Sleuth Kit[13] (TSK) is an open-source library and collection of several command-line tools for analysing disk-images. Its main features is the analysis of volume and filesystem data. This framework doesn't come with a graphical user interface.

This open-source project allows to analyse disks in 3 Steps:

1. File extraction Phase
2. File Analysis Phase
3. Post Processing Phase

Some of the benefits of the Sleuth Kit are:

- You can integrate it in other forensic tools
- You can implement new modules
- It handles the import of several disk-image types
- Active community
- Open Source

The project is hosted on GitHub[14] and it is written in c++. The Sleuth Kit framework can be downloaded[12] directly on their website in different versions. The contribution of Modules is well documented and welcome.

This framework comes with many benefits and a rich set of features. It might be a good choice to build upon.

E.1.3. Autopsy



Figure E.3.: Autopsy

Project website: <http://sleuthkit.org/autopsy/>

Autopsy[3] was developed by the same company as the Sleuth Kit[13]. It is a digital forensic platform with a graphical user interface for the Sleuth Kit framework and other forensic tools. Like the Sleuth Kit, Autopsy is open source too. It has a modular architecture which makes it easy to implement further features. Modules can be integrated into Autopsy while the software is running and there are already several 3rd party modules.

Autopsy is written in Java and uses several Netbeans libraries of the Netbeans Platform[**netbeansplatform**]. Further on, you can also implement your modules in Python, although they will be compiled to Java, which comes with some limitations.

Some of the main benefits are:

- It uses the Sleuth Kit[13], which is a powerful framework
- Modular architecture
- Open Source
- Active community
- Ongoing development
- Java (Since we are good at Java development)

The autopsy project is hosted on GitHub[8] and it is mainly written in Java. The most recent versions of Autopsy for Windows systems can be downloaded[7] on their website.

Due to its several benefits, it has a big potential for being our number one choice of software to build upon.

E.1.4. LOKI Free IOC Scanner



Figure E.4.: LOKI

Project website: <https://github.com/Neo23x0/Loki>

LOKI is a scanner for simple indicators of compromise (IoC)[25]. It comes with some interesting Features:

- MD5 / SHA1 / SHA256 hashes
- Yara Rules (applied to file data and process memory)
- Hard Indicator Filenames based on Regular Expression (e.g. \\pwdump\\.exe)
- Soft Indicator Filenames based on Regular Expressions (e.g. Windows\\[\\w]\\.exe)

This tool comes with some disadvantages:

- The project has no documented extension points
- The project is designed to run on the target host itself.
- It's not a tool kit, rather a tool

Since LOKI isn't a tool kit, we won't be able to build upon it. But it's an interesting tool and we might be able to integrate it in our future tool kit.

E.1.5. PlainSight

Project website: <http://www.plainsight.info/>

PlainSight is a forensic tool kit which combines several open source forensic tools. It comes with a graphical user interface.

This tool kit provides several features such as:

- Get hard-disk & partition information
- Importing DD-Images
- Several data analysis tools for gathering information

There are some downsides regarding this project:

- They don't provide the source code for download - you have to contact them to get it
- No active community
- No documentation found

- No information about active development

E.1.6. Digital Forensics Framework

Project website: <http://www.digital-forensic.org/>

This is an open source digital forensic framework. It has a modular architecture and already comes with some features. The downside of this project is, that many important features aren't freely available - you have to buy a pro version of the software.

This project isn't actively extended - no commits in the last 2 years. Further on there is no active community.

E.2. Tools and Scripts

Following is a list of some interesting scripts and tools:

- Bulk Extractor
Takes an image, file or folder and delivers data by keyword search.
http://www.forensicswiki.org/wiki/Bulk_extractor
- Scalpel
Tool to recover deleted files. (File carving)
<https://github.com/sleuthkit/scalpel>
- Fireeye's Redline
Windows tool for live forensics. Many features.
<https://www.fireeye.com/services/freeware/redline.html>
- Volatility
Memory forensics
<http://www.volatilityfoundation.org/>
- RegRipper
Registry analysis
<https://github.com/keydet89/RegRipper2.8>
- NSRLLookup
NIST NSRL DB Query Tool
<http://rjhansen.github.io/nsrlllookup/>

- OSSEC
Host-based Intrusion Detection System (HIDS)
<http://ossec.github.io/about.html>
- log2timeline
Generate Timelines
<https://github.com/log2timeline>
- Open IOC Scanner
IOC Database
<http://www.openioc.org/>

Lists of further forensic tools can be found on following websites:

https://en.wikipedia.org/wiki/List_of_digital_forensics_tools#Computer_forensics
<http://www.computer-forensik.org/tools/>

E.3. White- and Blacklists

An important part in digital forensics is the use of white- and blacklists. These are lists containing hashes of file which are known good or known bad.

Following is a list with databases that can be used for forensic tasks (mainly for white-/blacklisting):

- NIST National Software Reference Library (NSRL). Hash database with known-good hashes of Windows operating system and user-software
<http://www.nsrll.nist.gov/>
- OWASP File Hash Repository (FHR) aggregates several existing hash databases and provides access via a DNS queries. The project is not online at the time of evaluation.
https://www.owasp.org/index.php/OWASP_File_Hash_Repository
- Kaspersky Whitelist
Online hash database with known-good and known-bad hashes. Public access is limited to a few requests per minute
<http://whitelist.kaspersky.com/>
- VirusShare
Downloadable hash database of known-bad files.
<https://virusshare.com/>
- Virustotal

Online hash database of known-bad files. Public access is limited to a few requests per minute.

<https://www.virustotal.com/>

F. Glossar

AuthentiCode code signing standard by Microsoft

Autopsy Open source forensic analysis platform

Bitlocker Disk encryption software from Microsoft

Blacklist A list containing hashes of known bad files.

CERT Computer Emergency Response Team

Data Source A data source can be a system image, hard disk or a set of files.

Digital Forensics Recovery and investigation of material found in digital devices.

GitHub Provider for Git repositories

Git Version control system

Golden Image An Image which is considered to be uninfected.

Hashset A list of hashes of software artifacts.

Image A (system) image is an exact copy of a hard disk.

Ingest Module A type of autopsy module performing a specific analysis method.

Java Swing Framework for building user interfaces in Java.

Java A programming language.

Kali Linux Security Linux distribution

Kaspersky Digital security organization

Known bad files Files that are known to be (partially) malicious code

Known good files Files that are known to be safe, and which do not contain any kind of malicious code

Meta data Data that provide information about other data.

module A plug-In or attachable part for a software.

NIST National Institute of Standards and Technology

NSRL National Software Reference Library by NIST.

OpenIOC Open indicators of compromise, a description format of technical characteristics that identify a known threat.

TSK the Sleuth Kit abbreviation

VirusTotal Online hash lookup

Whitelist A list containing hashes of known good files.

G. Bibliography

- [1] Aorimn. *Dislocker*. <https://github.com/Aorimn/dislocker>. last accessed at 11th April 2016.
- [2] Emmanuel Bourg. *jsign library*. <https://ebourg.github.io/jsign/>. last accessed at 1th June 2016.
- [3] Brian Carrier. *Autopsy*. <http://sleuthkit.org/autopsy/>. last accessed at 9th April 2016.
- [4] Brian Carrier. *Autopsy 3rd-Party Modules*. http://wiki.sleuthkit.org/index.php?title=Autopsy_3rd_Party_Modules. last accessed at 28th May 2016.
- [5] Brian Carrier. *Autopsy Content Viewer Modules*. http://sleuthkit.org/autopsy/docs/api-docs/4.0/mod_content_page.html. last accessed at 23th mai 2016.
- [6] Brian Carrier. *Autopsy Developers Guide*. <http://sleuthkit.org/autopsy/docs/api-docs/4.0/>. last accessed at 23th May 2016.
- [7] Brian Carrier. *Autopsy Downloads*. <http://sleuthkit.org/autopsy/download.php>. last accessed at 9th April 2016.
- [8] Brian Carrier. *Autopsy GIT Repository*. <https://github.com/sleuthkit/autopsy>. last accessed at 9th April 2016.
- [9] Brian Carrier. *Autopsy Ingest Modules*. http://sleuthkit.org/autopsy/docs/api-docs/4.0/mod_ingest_page.html. last accessed at 23th May 2016.
- [10] Brian Carrier. *Autopsy Report Modules*. http://sleuthkit.org/autopsy/docs/api-docs/4.0/mod_report_page.html. last accessed at 23th May 2016.
- [11] Brian Carrier. *Autopsy Result Viewer Modules*. http://sleuthkit.org/autopsy/docs/api-docs/4.0/mod_result_page.html. last accessed at 23th May 2016.
- [12] Brian Carrier. *SleuthKit Downloads*. <http://sleuthkit.org/sleuthkit/download.php>. last accessed at 9th April 2016.
- [13] Brian Carrier. *Sleuthkit framework documentation*. <http://sleuthkit.org/sleuthkit/framework.php>. last accessed at 9th April 2016.
- [14] Brian Carrier. *SleuthKit GIT Repository*. <https://github.com/sleuthkit/sleuthkit/>. last accessed at 9th April 2016.
- [15] Mandiant Cooperation. *OpenIOC*. <http://openioc.org/>. last accessed at 15th June 2016.
- [16] Oracle Corporation. *The NetBeans Platform*. <http://www.netbeans.org/features/platform>. last accessed at 16th June 2016.

- [17] Simon Garfinkel. *BitLocker Disk Encryption*. http://www.forensicswiki.org/wiki/BitLocker_Disk_Encryption. last accessed at 11th April 2016.
- [18] Telia CERT Jimmy Arvidsson. *Taxonomy of the Computer Security Incident related terminology*. https://www.terena.org/activities/tf-csirt/iodef/docs/i-taxonomy_terms.html. last accessed at 15th June 2016.
- [19] B. Kaliski. *PKCS #7: Cryptographic Message Syntax*. RFC 2315. RFC Editor, Mar. 1998. URL: <http://www.rfc-editor.org/rfc/rfc2315.txt>.
- [20] Doug Koster and Kevin Murphy. *MantaRay Download*. <http://mantarayforensics.com/downloads/>. last accessed at 15th June 2016.
- [21] Doug Koster and Kevin Murphy. *MantaRay Forensics*. <http://mantarayforensics.com/category/mantaray/>. last accessed at 15th June 2016.
- [22] Doug Koster and Kevin Murphy. *MantaRay GIT Repository*. <https://github.com/mantarayforensics/mantaray>. last accessed at 15th June 2016.
- [23] Kaspersky Lab. *Whitelist Security Approach*. <http://whitelist.kaspersky.com/>. last accessed at 16th June 2016.
- [24] Rotarua Limited. *About VirusTotal*. <https://www.virustotal.com/en/about>. last accessed at 16th June 2016.
- [25] Hun-Ya Lock. *IoC Article*. <https://www.sans.org/reading-room/whitepapers/forensics/ioc-indicators-compromise-malware-forensics-34200>. last accessed at 15th June 2016.
- [26] Microsoft. *BitLocker Drive Encryption Overview*. <http://windows.microsoft.com/en-us/windows-vista/bitlocker-drive-encryption-overview>. last accessed at 16th June 2016.
- [27] Microsoft. *Microsoft PE and COFF Specification*. <https://msdn.microsoft.com/en-us/windows/hardware/gg463119.aspx>. last accessed at 9th June 2016.
- [28] Microsoft. *Tools to Sign Files and Check Signatures*. [https://msdn.microsoft.com/en-us/library/windows/desktop/aa388151\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/aa388151(v=vs.85).aspx). last accessed at 2th June 2016.
- [29] Microsoft. *Windows Authenticode Portable Executable Signature Format*. http://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/authenticode_pe.docx. last accessed at 15th June 2016.
- [30] Oracle. *Java™ Native Interface*. <http://docs.oracle.com/javase/7/docs/technotes/guides/jni/>. last accessed at 17th April 2016.
- [31] Mark Russinovich. *sigcheck.exe utility*. <https://technet.microsoft.com/en-us/sysinternals/bb897441.aspx>. last accessed at 1th June 2016.
- [32] Offensive Security. *Kali Linux*. <https://www.kali.org/>. last accessed at 9th April 2016.