ERNW RESEARCH
pursuing knowledge.

ERNW WHITE PAPER 79

INTEGRATING INCIDENT ANALYSIS AND DIGITAL FORENSICS TOOLING FOR AUTOMATED COMPROMISE DETECTION

Version:	1.0
Date:	August 12, 2026
Classification:	Public

Julian Suleder

Digital unterschrieben von Julian Suleder
Datum: 2026.08.11 16:46:23 +02'00'

ERNW RESEARCH
pursuing knowledge.

Table of Contents

ERNW RESEARCH
pursuing knowledge.

1 Handling

The present document is classified as *Public*. Any distribution or disclosure of this document REQUIRES the permission of the document owner as referred in Section *Document Status and Owner*.

1.1 Document Status and Owner

As the owner of this report, the document owner has exclusive authority to decide on the dissemination of this document and responsibility for the distribution of the applicable version in each case to the places.

The possible entries for the status of the document are *Initial Draft*, *Draft*, *Effective* and *Obsolete*.

Report Information	
Title:	ERNW White Paper 79 - Integrating Incident Analysis and Digital Forensics Tooling for Automated Compromise Detection
Document Owner:	ERNW Research GmbH
Version:	1.0
Status:	Effective
Classification:	Public
Author(s):	Ann-Marie Belz

Table 2: Document Status and Owner

1.2 Classification Levels

Classification Level	Audience
Public:	Everyone
Internal:	All employees and business partners
Confidential:	Only employees
Secret:	Only selected employees

Table 3: Classification Levels

1.3 Document Version History

Version	Date	Details
1.0	August 12, 2026	Initial version after quality assurance.

Table 4: Document Version History

2 Introduction

The security landscape in recent years has been marked by a significant number of cyber security incidents, with ransomware attacks emerging as one of the primary threats to organizations and critical infrastructure. The *BSI Lagebericht 2025* [Bun25] reports the highest average ransom payments since the beginning of data collection. In particular, Ransomware-as-a-Service (RaaS) models have enabled attackers to generate substantial profits, with one notable case involving the ransomware Dark Angels, which reportedly extorted more than USD 75,000,000. This trend was already evident in the *BSI Lagebericht 2024* [Bun24] for Germany, which highlighted the far-reaching impact of such cyber security incidents. A high-profile case involving an IT service provider affected 72 municipalities, disrupting essential citizen services, including registration offices and building authorities. Such incidents underline the critical need for organizations and government entities to rapidly regain operational capacity in the face of cyber threats, ensuring that essential services can be resumed as quickly as possible. A key measure to achieve this resilience is the effective implementation of Incident Response (IR) [Cic+12].

As the frequency of cyber security incidents continues to increase, the need for effective Incident Analysis (IA) has never been greater. Although incidents themselves may be unavoidable, organizations can certainly influence how they manage and respond to these events. The IA process seeks to understand how and why an incident occurred, often revisiting definitions of unlawful, unauthorized, or unacceptable actions to identify their origin. IA involves collecting evidence, reconstructing events, and assessing the impact on the organization's systems and data [Lut+14].

2.1 Motivation

IA offers a multitude of possibilities, but also presents various challenges that hinder its effectiveness. Some of the key issues are summarized below.

Garfinkel [Gar10] predicted in 2010 several challenges for the next 10 years for Digital Forensics (DF) that also apply to IA, including the growing size of storage devices, the insufficient time available to analyze all data, and the increasing complexity of analyzes due to a wider variety of operating systems and file formats, which necessitates the use of multiple analysis tools. As forensic analyses become more time-consuming and costly, the need for efficient tools becomes critical. For instance, Garfinkel noted that imaging a 2TB hard drive already took around seven hours, and storage capacities have since increased significantly. Furthermore, forensic tools must be constantly updated or integrated with others to remain relevant. Large vendors struggle to cover all use cases, making it necessary to rely on smaller, often open-source tools. However, these tools often suffer from issues such as being outdated, poorly documented, or abandoned due to lack of funding. A final issue is the lack of triage functionality in forensic tools. Given the length and complexity of forensic analyses, a more efficient approach would be to prioritize critical findings and immediately

return them to the analyst for further investigation. However, few existing tools implement such a system, highlighting a gap in current forensic methodologies.

These predictions have largely materialized, leading to the widespread need for multiple analysis tools, which is a challenge that this work seeks to address.

This development is also supported by a review from 2024 by Fakhouri et al. The review confirms several of Garfinkel's earlier observations. In particular, the authors highlight the exponential growth of storage capacities, resulting in vast volumes of data that forensic analysts must process, often exceeding the capabilities of existing tools. Additionally, the increasing diversity of data formats and protocols, especially in domains such as Internet of Things (IoT) forensics, has intensified the need for specialized tools that contribute to the growing complexity of the forensic tool landscape.

Although incident analysts now have access to a wide range of tools for IA to overcome the previously mentioned challenges, they still devote a large portion of their time to this phase of the IR lifecycle [Lut+14]. Two primary challenges arise in this phase: reliance on manual labor and time constraints that hinder the ability to perform a comprehensive analysis. Although numerous commercial and open-source tools are available, it is often difficult to extract meaningful insights from the vast amounts of detailed data. Additionally, effective use of these tools requires a certain level of technical expertise, as the underlying decision-making processes are often not well explained. Even more problematic is that these tools still need to be applied manually by the analyst, tailored to the specifics of each situation. This manual approach is not only time-consuming but also prone to human error and requires expertise in selecting the right tools and interpreting their results.

Another major challenge is the lack of integration between these tools. While many steps of IA remain relatively constant across different types of incidents, no single tool is capable of covering all analyzes. Therefore, connecting these tools becomes essential to maximize their effectiveness.

A 2021 review examines the human decision-making process during IA, particularly in the selection of analysis tools and the way information is reported. To identify existing gaps, the study analyzed 30 documents from the Internet Engineering Task Force (IETF), the International Organization for Standardization (ISO), and Forum of Incident Response and Security Teams (FIRST), revealing a lack of guidelines on how analysts should make decisions in these areas. One identified gap concerns the methodological approach to IA. While general recommendations exist for implementing structured methods, there are no specific guidelines on what such an approach should look like in practice. The available instructions tend to be either too broad or too narrow, offering little practical value. Additionally, although adequate tools and training materials are available, there are no established criteria for selecting the most suitable ones. A similar issue arises in reporting. While the standard demands a scientific approach, there is no clear definition of what this means for analysts in practice. Some documents imply that reporting in DF should follow the model of traditional forensic disciplines, while others highlight that reporting remains one of the most challenging aspects of IA [Spr+21].

This challenge is further emphasized by the ENISA Foresight Cybersecurity Threats for 2030 report, published by the European Union Agency for Cybersecurity (ENISA) [Eur24], which identifies automated decision-making as a key trend in response to the increasing complexity of cybersecurity threats. As decision-making processes become more complex, there is a growing need for automated approaches to support or replace manual analyst decisions. Consequently, the design of the PYRAMID Framework prioritizes automation, aiming to minimize the need for manual decision-making by analysts.

2.2 Contribution

Having outlined the challenges of IA and existing approaches, the primary objective of this work is to develop a framework that integrates existing IA tools and enables their automatic execution based on a predefined workflow. This integration will streamline the analysis of hard drive disk images. The framework will dynamically decide which tools should be executed next by processing the outputs of previously executed tools. This process aims to enhance the efficiency of IA by guiding the selection and application of appropriate tools tailored to specific scenarios.

Additionally, the framework will generate an initial report, compiling key findings and insights from the outputs of all launched tools. This report will include potential Indicators of Compromise (IoC) and a statement regarding whether the disk image appears to have been compromised. While the framework will automate much of the preparatory work, it will still rely on the expert knowledge of analysts, as it cannot fully replace human expertise. The workflow used within the framework is developed with the assistance of professional security incident analysts to ensure its relevance and accuracy.

This approach aims to address the following research questions:

- RQ1: To what extent can an automated analysis of a hard drive disk image determine if the system has been compromised?
- RQ2: How much time can security incident analysts save by using an automated hard drive disk image analysis framework?

2.3 Outline

In the following Section 3 foundational work in IA and DF is presented, as well as existing approaches to automate analysis processes. In Section 4 the workflow, developed in collaboration with experienced security analysts, and the automated analysis steps, including file system analysis, Windows artifact analysis, and malware analysis are outlined. All orchestrated tools to perform these analyses are presented in Section 5. To validate the framework, an evaluation was conducted using six disk images, enabling an assessment of its compromise detection capabilities and performance (see Section 6). Section 7 discusses whether the defined goals have been achieved and provides answers to the research questions. Finally, potential directions for future work aimed at improving the framework are presented in Section 8.

3 Related Work & Existing Frameworks

Luttgens et al. provide foundational work related to IR, outlining the structured process of handling computer security incidents. In their work, they describe the IR process including preparation, data collection, data analysis, and remediation. They go into more detail how an IA is conducted. Specifically, the authors describe methodologies for analyzing Windows-based systems, with an emphasis on file system analysis of the New Technology File System (NTFS). They explain key mechanisms and artifacts, including their purpose, storage locations within the system, and the tools required for their analysis. Among the crucial artifacts discussed are Prefetch files, Windows Event Logs, and the Windows Registry, which can provide valuable insights into system activity and potential security incidents. Additionally, Luttgens et al. emphasize the importance of reporting in the incident analysis process. They outline best practices for creating reports that are accurate, complete, and readable.

A related field of IA is DF, with Dewald and Freiling's [Dew+15] standard work providing key insights. They first outline traditional forensic sciences and then describe computer forensics as a scientific field that works with associations of digital traces. While IA focuses on questions about the compromise of a computer, DF addresses questions related to the legal system, such as whether computer A was used to access website B. This makes forensic work legally defensible, requiring scientifically accurate methods and clear reporting. Additionally, Dewald and Freiling explain various models, including the IR model by Luttgens et al., and also cover forensic analyses of storage media and their documentation.

Carrier [Car15] describes key concepts of file system analysis. He begins by outlining the fundamentals of the field of DF. After that, he discusses analysis techniques for volumes and file systems and introduces his self-developed tool, The Sleuth Kit (TSK). In file system analysis, Carrier presents a basic model with five data categories and details methods for examining the File Allocation Table (FAT) and its successor, NTFS. He introduces TSK and Autopsy, two open-source tools designed for forensic analysis. TSK is a collection of command-line tools for analyzing disk and file system images. Autopsy serves as a front-end for TSK, functioning as an HTML-based web server that parses TSK output. It provides multiple analysis modes, enabling users to extract various types of information from files and create timelines of file activity.

Contemporary works increasingly focus on implementing new tools and technologies to automate IA and generalizing workflows for IR. For example Akatosh. The framework was developed within the scope of a project by the U.S. Department of Homeland Security's Transition to Practice Program (TTP). Akatosh automates IA by capturing time snapshots of memory images. When an Intrusion Detection System (IDS) triggers an alert, corresponding snapshots are automatically analyzed, and a report is generated. The analysis is performed using Volatility and Rekall. However, Akatosh is limited to memory analysis and does not extend to disk image analysis [Smi+17].

The GRR Rapid Response Framework (GRR) is an open-source, multi-platform tool designed for continuous remote live forensic investigations within enterprise environments. It allows analysts to collaboratively manage investigations through a centralized web interface. GRR is built to support automated analysis and integrates additional forensic tools, such as TSK. The framework operates on a client-server model, where flows are initiated from the server to client machines. For example, an analyst can configure a flow to have client machines search for specific malware hashes, with any findings returned to the server for further analysis. However, a key limitation of the GRR framework is its real-time investigative focus, which may not align with workflows requiring non-live analysis [Coh+11].

One of the most widely used tools in DF is X-Ways, a software based on the WinHex hex and disk editor. It enables collaboration between computer forensics analysts and offers a comprehensive set of features. Such features are for example disk imaging, reading partitioning and file system structures, detecting Alternate Data Streams (ADS), and performing hash calculations. X-Ways also supports the import of hash sets like the National Software Reference Library (NSRL) and allows users to view Windows event log files and tag interesting files for reporting. The software is GUI-based and licensed therefore requiring a paid subscription [X-W26].

The AUDIT toolkit integrates open-source tools within a Java Expert System Shell (Jess). It allows users with limited technical expertise to create rules for automated task execution during forensic investigations of disk images. It consists of three main components: a MySQL database, a knowledge base, and a core component. The database stores information about forensic tools, including their input requirements and expected output. Examples of integrated tools include bulk_extractor, which extracts artifacts such as emails, credit card numbers, and Internet history [sim26]. The knowledge base contains facts and rules that enable forward chaining, allowing the system to infer new information about the target disk based on previously collected data. The core component is responsible for executing forensic tools according to the rules defined in the knowledge base. While the system generates reports, these do not contain specific investigative findings but rather document which tools were executed, when, and why. The AUDIT toolkit primarily focuses on extracting personal data from suspect computers, such as images, emails, and documents, rather than detecting malware infections or other system compromises [Kar+16].

The SCAlable Realtime Forensics (SCARF) framework is a container-based software tool designed to support automated scalable forensic analysis. SCARF accepts disk images as input and generates a searchable database of forensic artifacts, which is maintained as a cluster of Elasticsearch nodes to enable efficient querying. The actual analysis is performed by workers, each operating as an independent Docker container, with every worker handling one specific task. Tasks can include operations such as SHA-1 hashing, running bulk_extractor, or using ExifTool to extract metadata from files. SCARF offers fast processing and scalability, as its containerized architecture allows for dynamic resource allocation. Additional containers can be deployed as needed, depending on the available hardware resources, making the system adaptable to different workloads. However, its primary focus is on evidence retrieval for DF rather than on understanding or reporting the root causes of a compromise, limiting its use in IA scenarios [Ste+17].

Farrell [Far09] presents the development of an automated reporting system built on top of the PyFlag forensic tool, an open-source solution implemented by the Australian government for forensic media and network analysis. PyFlag, short for Python Forensic and Log Analysis GUI, utilizes TSK for file system analysis and includes its own scripting language, PyFlash, which allows users to develop extensions. In this work, PyFlag was extended with two new plugins focusing on improved reporting and the integration of an AOL Instant Messenger file identifier and extractor. However, the study also highlights several limitations. The system suffers from usability issues, making installation and setup challenging. Additionally, PyFlag faces significant integration challenges, as every new tool or component must be specifically adapted to fit within its architecture, limiting its flexibility and extensibility.

4 Methodology

The foundation of the incident analysis framework is a workflow that models the investigative steps a human analyst would typically follow based on the results of previous tools. This structure was developed based on insights gained from interviews with ten experienced security analysts of ERNW.

The scenario that the workflow deals with involves a potentially compromised disk image from a Windows computer that should be analyzed. This scenario was selected because disk images are examined most frequently in forensic investigations, and Windows is the most widely used desktop operating system [Sta25]. Before the analysis can begin, it is essential to define the objectives and key questions to be addressed. In this work, the primary question is whether the disk image has been compromised. Consequently, analysis must focus on identifying malware and detecting suspicious events that indicate unauthorized manipulation by attackers rather than legitimate user actions. The analysis process is structured to first collect data comprehensively before examining individual files and events. Subsequently, the findings are evaluated and a report is generated. The framework is based on a file-driven structure, using YAML for decision control and Python for implementing analyses.

Figure 1 provides a summarized overview of the workflow. The full workflow can be found in Appendix A. The input is the disk image, followed by multiple analysis steps. Initially, fundamental data about the image is retrieved. The process then diverges into two main paths: event analysis and artifact analysis. From the artifact analysis, specific files are extracted and subjected to a detailed malware analysis in a subprocess. All results of the analysis are stored in a central TinyDB database [Sie21], ensuring that subsequent analysis steps can access and persistently store the findings. This structure enables a comprehensive evaluation of the results and facilitates the final report generation with Jinja2 templates.

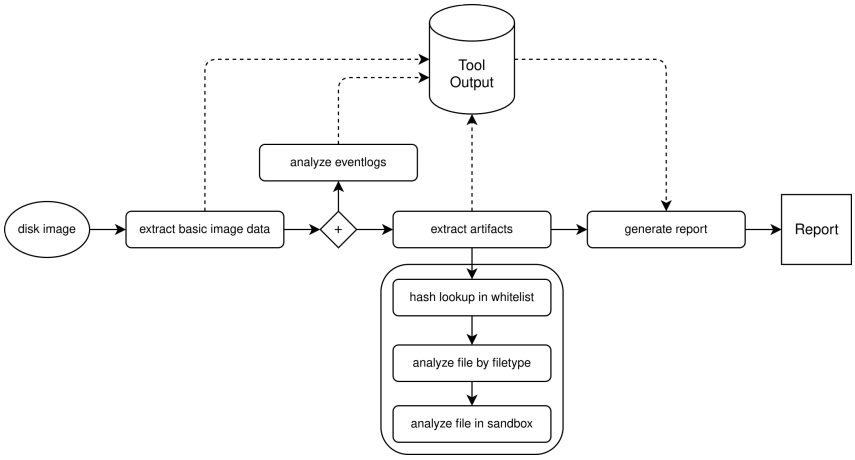


Figure 1: Simplified overview of the workflow illustrating seven analysis steps.

The analysis process described by the workflow proceeds from general to specific: first, file system analysis, then Windows-specific analysis, and finally, malware analysis. The initial step in file system analysis is the volume consistency check with mmls [Car04]. This serves two purposes: first, to exclude a corrupted partition table, which could lead to errors in further analyses, and second, to verify that no hidden data exists between partitions. This ensures that no relevant data is overlooked in subsequent steps. The output of this analysis is the partition table, which is stored for verification by analysts. Next, basic system data is extracted with dissect [Fox26] to provide context on system usage. This includes information such as user accounts, IP addresses, Windows version, and hostname. This data helps verify that the correct image is being analyzed and can later be used e.g., to distinguish legitimate users from potential attackers when comparing Indicators of Compromise (IoCs). All basic data is stored in a dedicated database table. Following this, the process branches into Windows analysis, which consists of two main components: the examination of Windows Event Logs and the analysis of artifacts such as registry data.

The second part of the workflow focuses on Windows analysis. Event logs, stored as `.evtx` files, must first be extracted from the disk image with Dissect's target-fs [Fox23]. Once extracted, these logs are uploaded into an Elastic Stack [Ela26] for analysis, where the events are structured for further examination. Given the large size of the event logs, they are too extensive to be stored in the central database. Instead, suspicious events are identified with Sigma rules [Sig26] and analyzed within the Elastic Stack, and relevant findings are included in the final report.

Following the event log analysis, the next step is artifact extraction and analysis. In this phase, kirby is used to extract forensic artifacts from the disk image, including Windows Registry hives, Prefetch files, browser-related artifacts, and files containing Alternate Data Streams (ADS) [BDO26]. Rather than performing the analysis itself, kirby invokes the appropriate Dissect plugins for each artifact type and exports the extracted information as CSV files. A custom workflow implemented in YAML and Python processes these CSV outputs and imports the relevant metadata, such as file names, file paths and associated timestamps, into a TinyDB database. The focus on these artifacts was chosen to limit the scope of the file analysis while capturing indicators associated with common attack techniques, such as persistence mechanisms and file execution activity, as described in the MITRE ATT&CK matrix [The26]. Storing all extracted metadata in the database ensures that the information remains available for the subsequent malware analysis phase.

The final part of the main workflow is the final assessment of whether the disk image is compromised and the generation of the final report. For the rating process, specific criteria have been established. If no suspicious files were identified in the previous analysis steps and no suspicious events were detected in the event logs, the system is likely not compromised. However, it is crucial to note that assessing a system as not compromised can never be made with absolute certainty. The report is generated by aggregating and processing all relevant information from the database, ensuring a clear and structured presentation of findings. This step is of particular importance as it represents the core outcome of the framework and includes the final rating. Screenshots of an example report can be found in Appendix A.

The subprocess for analyzing suspicious files is depicted in Figure 2. This subprocess handles all files identified during the Windows artifact analysis as potentially relevant. These include files associated with e.g., services, run keys, or files downloaded via a browser. The objective of this step is to determine whether any of these files are malicious. The analysis begins by generating a hash for each file and checking it against the NSRL whitelist [NIS26]. This step helps eliminate many false positives, thereby reducing the number of files requiring further analysis and conserving computational resources. The subsequent analysis is tailored to the file type, allowing for a more precise examination using the most appropriate tools. However, only files not found in the whitelist proceed to this step, which is determined by querying the database for not-whitelisted files. All results are stored in the same database used in the main process, now including additional information such as file hashes, whitelist status, and outputs from file-specific analyses. However, due to size constraints and formatting challenges, detailed file analysis outputs are stored separately as external files rather than being directly inserted into the database. This approach ensures that the original structure and formatting of the analysis results are preserved.

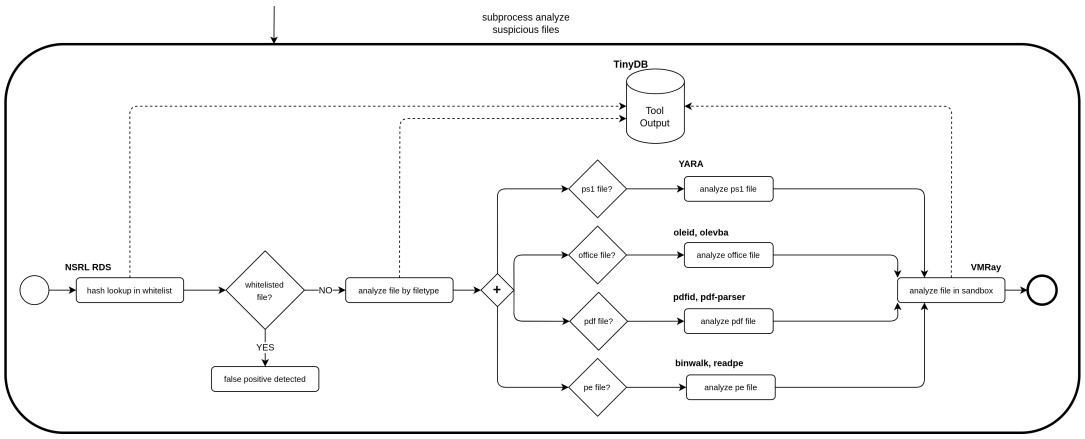


Figure 2: The subprocess in the workflow for analyzing suspicious files. Initially, the files are checked against a whitelist to filter out known benign files. Subsequently, a static analysis is performed based on the specific file type. After this, all files undergo dynamic analysis in a sandbox.

The second part of the subprocess decides which files undergo detailed analysis and which tools are used. The focus is on file types commonly exploited by attackers for malware distribution. These primarily include executable files containing code, such as Portable Executable (PE) files and PowerShell scripts, as well as widely used document formats like PDF and Microsoft Office. After the initial static analysis, which examines file characteristics without execution, a dynamic analysis is conducted. For security reasons, this process takes place in the sandbox environment of VMRay [VMR26] to prevent malware from compromising the analysis system. To optimize resource usage, a triage system is implemented, ensuring that higher-priority files are analyzed first. This prioritization allows analysts to obtain critical results as early as possible. With the completion of the sandbox analysis, the subprocess concludes, enabling the main process to proceed with generating the final report.

5 Orchestrated Tools

The following section provides a comprehensive list of the tools currently orchestrated by the framework. Some internal tools are omitted for privacy and confidentiality reasons. All tools integrated into the framework are executed locally within the analysis environment, and no data is transmitted to third-party services. Components such as the Elastic Stack and the VMRay sandbox are hosted on-premises, ensuring that all forensic artifacts and analysis results remain within the infrastructure of ERNW.

`mmls` Introduced in version 1.63, mmls was the first tool within the media management category of TSK. It is primarily used to display the partition table and disk labels of a volume system. This functionality is essential to inspect the overall partition layout, verify the consistency of the structure, and identify hidden or unallocated areas on the disk that may contain suspicious data. [Car04]

`Dissect` Developed by Fox-IT [Fox26], the toolset Dissect is a powerful forensic analysis tool designed to extract artifacts from disk images and files. It provides both command-line tools, such as target-query and target-fs, as well as a Python API for integration into other projects. Dissect is capable of handling a wide range of disk image formats.

`Elastic Stack` The Elastic Stack, also widely known as the ELK Stack consists of Elasticsearch, Kibana, Beats, and Logstash. Elastic Stack can be used to collect and analyze Windows Event Logs. The typical workflow involves collecting log data using Beats, forwarding it either directly to Elasticsearch or via Logstash, and finally visualizing it using Kibana. [Ela26]

`Sigma` Sigma is a generic signature format designed to describe detection rules for identifying malicious events in log data. Sigma detection rules, which are often developed and shared by the community, are written in YAML format and support a wide range of log sources. [Sig26]

`kirby` The DFIR-DD project, developed by a team of incident responders and forensic analysts, provides a collection of open-source tools for forensic analysis of Windows systems including the tool kirby. Kirby is a triage tool based on Dissect, implemented in Python. It leverages Dissect plugins to parse forensic artifacts such as browser history, installed services, and run keys, outputting the extracted data in CSV format. [BDO26]

`NSRL RDS` A crucial resource for local hash lookups in malware analysis is the National Software Reference Library (NSRL). Supported by the National Institute of Standards and Technology (NIST), the U.S. Department of Homeland Security, and law enforcement agencies, the NSRL project collects software metadata and compiles the Reference Data Set (RDS), which catalogs known software files. [NIS26]

`YARA` YARA is a tool originally developed by VirusTotal to identify and classify malware. YARA can be used either as a command-line tool or integrated into Python scripts. YARA operates based on rules, which define patterns or conditions under which a file should be considered malicious similar to antivirus signatures. [Vir22]

`oletools` To facilitate the analysis of Office documents, Philippe Lagadec [Lag26] developed a collection of Python-based tools known as oletools. These tools support both the legacy CFB format and the modern OOXML format. Two key

tools within this suite are oleid and olevba. Oleid scans documents for indicators such as the presence of VBA macros. If macros are detected, olevba can be used for in-depth analysis.

DidierStevensSuite One widely used set of tools was developed by Didier Stevens [Ste26], including pdfid and pdf-parser. Pdfid scans PDF documents for suspicious keywords, such as /JavaScript and /OpenAction, and reports the number of occurrences for each. If a PDF document is deemed suspicious, pdf-parser can be used for a more detailed analysis.

binwalk Binwalk is a tool developed by ReFirm Labs [ReF26], which was acquired by Microsoft in 2021. Originally designed for analyzing firmware, it has expanded its capabilities to support many file types, including PE files. Binwalk is particularly useful for file identification and extraction of embedded files. It also includes entropy analysis, which helps identify interesting sections within files.

readpe Readpe is a command-line tool that originated from the pev Tool and was initially used to extract the version of PE files. Over time, it has evolved into a more comprehensive utility for binary analysis. Readpe parses PE files to reveal various details about their structure, such as determining the header, identifying the entry point, listing sections, and providing a list of imported functions. [Men26]

VMRay To automate malware analysis, sandbox environments such as VMRay are commonly used. VMRay ensures that malware cannot escape the sandbox, making it a reliable tool for safely observing malware behavior during dynamic analysis. [VMR26]

6 Evaluation

This chapter presents the evaluation of the incident analysis framework. The evaluation was conducted using six test images, which are introduced in Section 6.1. The primary objective was to assess whether the research questions could be answered. First, the framework's ability to determine whether a disk image is compromised is evaluated in Section 6.2. Subsequently, the completeness of the analysis is discussed in Section 6.3 – Section 6.7, including whether all files were analyzed and whether relevant IoCs were identified. Finally, Section 6.8 and Section 6.9 examine the execution times of the individual analysis modules and compare them to the timeline of human analysts.

6.1 Test Data Set

The evaluation of the incident analysis framework was conducted using six disk images. While the number may seem small, this is due to two main constraints: first, obtaining real-world images is challenging, and second, all images needed to have been previously analyzed by experienced analysts and documented as ground truth to allow for meaningful comparison. Despite the limited number, using these verified images provides higher practical relevance than a larger set of synthetically generated images, as it reflects realistic investigative challenges.

Table 5 provides an overview of the basic properties of each image. Four of the images are from real incident cases and were provided by security analysts. These real-case images stem from two separate incidents: Images 1.1, 1.2, and 1.3 are from the same case, while Image 2.1 is from a different incident. For all real-case images, ground truth data in the form of the analysts' original incident reports was made available, allowing for a direct comparison with the results produced by the framework. In addition, one image (T.1) was artificially created as part of an incident response training course. While the case was fictitious, it was designed to closely mimic a realistic compromise scenario. The final image (T.2) represents a clean Windows virtual machine used for penetration testing purposes. Due to the inclusion of real incident data, some information has been stripped or generalized to avoid identifying the affected organizations or the nature of the specific incidents.

The images were selected to represent a broad range of scenarios and file formats to thoroughly test the framework's capabilities. As the framework categorizes systems into three compromise rating levels, two images per category were selected to ensure a balanced comparison. Images 1.1 and T.1 were confirmed to be compromised; Images 1.2 and 1.3 were considered likely compromised; and Images 2.1 and T.2 were verified to be clean. All images were based on Windows operating systems but varied in version to assess whether the framework's behavior differs across OS versions. The oldest system was running Windows 8.1 Enterprise, installed in 2017. The remaining images used Windows 10 Pro or Enterprise, installed between 2020 and 2024.

A range of disk image formats was tested, including E01, VMware Virtual Machine Disk (VMDK), Virtual Disk Image (VDI) and RAW. While the tools used in the framework are designed to support a wide range of formats, two cases were identified where format conversion was necessary. One of the formats tested was E01 (also known as Expert Witness Format), commonly used with EnCase. Due to tool limitations (e.g., version incompatibility with Dissect), Image 2.1 had to be converted to RAW and was mounted using ewfmount. Similarly, since the mmls tool does not support VDI images, Image T.2 was converted to VMDK. Furthermore, different partitioning schemes were included to test the framework's compatibility with both Master Boot Record (MBR) and GUID Partition Table (GPT) partition tables. Images 1.1 and 2.1 used GPT, while the others used MBR. The disk sizes also varied to test the framework's performance. The real-case images were significantly larger than the test images. The largest image (1.1) was approximately 250 GB, while the test images ranged from 10 GB to 25 GB. As shown in the table below, Images 1.2 and 1.3 share identical specifications and originate from the same incident. They were included in the test dataset to evaluate whether the framework produces consistent results for similar disk images.

Image	Real Case	Compromised	Windows Version	Date	Format	Partition Table	Size
Image 1.1	yes	yes	10 Pro	2021	e01	GPT	250 GB
Image 1.2	yes	likely	8.1 Enterprise	2017	vmdk	MBR	100 GB
Image 1.3	yes	likely	8.1 Enterprise	2017	vmdk	MBR	100 GB
Image 2.1	yes	no	10 Pro	2020	e01/raw	GPT	~40 GB
Image T.1	no	yes	10 Enterprise	2022	e01	MBR	~10 GB
Image T.2	no	no	10 Pro	2024	vdi/vmdk	MBR	~25 GB

Table 5: Metainformation about the six disk images from the test data set.

6.2 Evaluation of Compromise Rating

To answer the first research question RQ1, a confusion matrix was created, as shown in Table 6. The matrix is structured according to the three rating categories defined by the framework: *compromised*, *likely compromised*, and *not compromised*. The columns represent the framework's predictions, while the rows indicate the ground truth as defined by analyst reports. The matrix shows that the two images that were actually compromised were correctly classified as such by the framework. However, the two images that were only likely to be compromised were also classified as fully compromised, resulting in false positives in this category. Additionally, the two images that were not compromised were rated as likely compromised, which also represents a deviation from the ground truth. Overall, the framework shows a consistent tendency to overestimate the threat level, favoring more severe ratings. This leads to only two fully correct classifications out of six, resulting in an overall accuracy of 0.33. Despite its tendency towards more cautious predictions, the framework demonstrates consistent behavior in its assessments, as all images within the same true category were rated in the same way.

Rating	compromised (pred.)	likely compromised (pred.)	not compromised (pred.)
compromised (true)	2	0	0
likely compromised (true)	2	0	0
not compromised (true)	0	2	0

Table 6: The confusion matrix shows the predicted ratings (compromised/likely compromised/not compromised) of the evaluated images by the analysis framework compared with the rating given by security analysts.

6.3 Number of Analyzed Files

The results are presented in Table 7. For each disk image, the total number of files and directories was determined using the extracted Master File Table (MFT) from Dissect. However, it should be noted that the MFT also contains entries for files that have already been deleted. Although all files and directories could potentially be relevant, their sheer number exceeds what is feasible to analyze within a limited time frame. Therefore, only selected artifacts and files deemed relevant were written to the internal database for further analysis. Out of a total of over 6 million files and directories, the amount was reduced so effectively that only 0.14 % ultimately required detailed analysis, significantly reducing processing time. On average, this amounted to 0.19 % per image. Some of these files were already deleted, for example in Image 2.1, which contained many downloaded files. In Image 1.1, numerous file paths were incomplete due to limitations in the artifact extraction process using Dissect; in some cases, such as with Windows service artifacts, the full file paths could not be extracted.

Overall, 46 % of all files referenced by extracted artifacts were found on disk. The average across all images was 56 %. This already reduced the number of files eligible for further analysis. The first filtering step was a hash-based whitelist lookup. Only files that were present on disk could be checked against the whitelist. In total, 34 % of these files matched the whitelist; on average, 39 % per image. Only the files that did not match the whitelist were subjected to further analysis steps. This means that only 64 % of the files found on disk were analyzed using file-specific tools and dynamic analysis in the VMRay sandbox. File-specific analysis was only performed on selected file types for which appropriate tools were available: Microsoft Office documents, PDF documents, PowerShell scripts, and PE files. Notably, for Image 1.2, all files could be analyzed using these tools. In contrast, other disk images included file types such as image files (e.g., PNG, JPG), VBS scripts, or CSV files, for which no specific analyzers were used. Furthermore, some files could not be uploaded to VMRay, presumably due to excessive size. Although the timeout for API-based upload requests was increased, errors still occurred. In total, 13 files were not uploaded due to timeouts.

Files	Image 1.1	Image 1.2	Image 1.3	Image 2.1	Image T.1	Image T.2
All Files/Dirs.	>3M	>600K	>600K	>1M	>200K	>200K
Extracted Files	4,427	926	971	2,015	619	815
Found on Disk	1,505	547	620	835	420	570
Not Whitelisted	1,115	401	460	649	141	190
File Analysis	1,043	401	457	599	133	190
VMRay Analysis	1,113	401	459	642	141	188

Table 7: The table shows how many files of the test disk images were analyzed.

6.4 VMRay Verdicts

All files uploaded to VMRay were subjected to a three-step analysis process: reputation analysis, static analysis using YARA rules and antivirus signatures, and dynamic execution within the sandbox environment.

In total, 2,944 files were analyzed using VMRay. The resulting verdicts from the sandbox analysis had a substantial impact on the final compromise rating assigned by the framework. VMRay categorizes analysis results into four verdicts: *N/A* (or *null*) in cases where errors occurred during analysis, *clean* if the file is known or no suspicious behavior was detected, *suspicious* if the behavior is indicative of potentially harmful activity, and *malicious* if the file is classified as malware. The distribution of these verdicts is illustrated in Figure 3. As expected, the majority of files (73 %) were rated as *clean*. The smallest category was *malicious*, comprising only 12 files. Among these, 9 files originated from images 1.1 and T.1, where all samples identified as malicious by VMRay were independently confirmed as such by security analysts. Three files were classified as false positives following consultation with analysts.

Of the 668 files rated as *suspicious* no files with malicious behavior could be confirmed. Most of these files were legitimate system files that were rated *clean* in the reputation and static analyses but flagged as *suspicious* during dynamic analysis due to behavior such as spawning an unusually high number of processes. Additional false positives originated from image T.2, which was a virtual machine used for penetration testing and contained a large number of security tools that influenced the dynamic analysis. For 121 files, no final verdict could be produced. This occurred either because the sandbox analysis encountered errors or because the files could not be executed during the dynamic analysis.

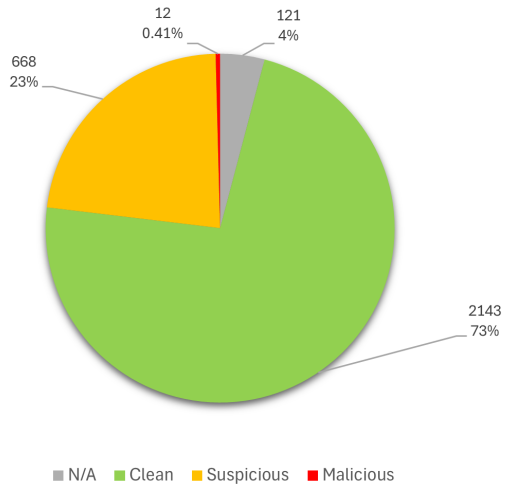


Figure 3: The pie chart shows the distribution between the VMRay verdicts N/A, clean, suspicious and malicious.

6.5 Detected IoCs

A critical aspect in answering RQ1 is determining whether all IoCs present in the compromised images could be identified by the framework. Table 8 presents an overview of the IoCs from the first incident on images 1.1, 1.2, and 1.3, comparing detection by the analysts and the framework. The compromise involved attackers deploying several tools on the systems, including a cryptominer, a backdoor, and a keylogger, each consisting of multiple associated files. The framework was able to extract and analyze at least one file per tool, thereby confirming the presence of each attack component, although not all associated files were recovered. All extracted files were correctly flagged as malicious. Interestingly, in the case of the keylogger, the specific file was not identified by the analysts but was detected by the framework. However, the analysts were still able to confirm the presence of the keylogger, as the backdoor component included a file that contained keylogging functionality.

The attackers also employed several auxiliary tools to facilitate the attack. While these tools themselves were not inherently malicious, they were not detected by the framework. One of the attacker tools downloaded a file for the cryptominer. In this context the analysts were able to extract the download URLs. While the framework was not able to find the tool, through Sigma rule detections the URLs were found as IoCs. Both the analysts and the framework identified that the attackers had excluded certain directories from Windows Defender scans. In the event logs, two suspicious scheduled tasks and corresponding services were identified by the analysts. The framework also detected the services through its event log analysis, specifically via the Sigma rule `win_system_service_install_paexec`. Although no Sigma rules matched the creation of the scheduled tasks, the relevant events were ingested into the Elastic Stack, where they could be identified manually.

Indicator of Compromise	Analysts	Framework
Cryptominer	yes	yes
Backdoor	yes	yes
Keylogger	no	yes
Other Attacker Tools	yes	no
Cryptominer Download URLs	yes	yes
Defender Exclusions	yes	yes
Scheduled Tasks	yes	no
Suspicious Services	yes	yes

Table 8: The table shows the comparison between the IoC detections of analysts and the framework of the real-world incident.

Table 9 presents an overview of the IoCs from the second incident. The incident was a controlled training scenario. Consequently, no actual report or real-world analysis existed; all IoCs were known to the analysts, as they had designed the scenario. However, the framework was not able to identify three out of eight IoCs. The scenario began with an initial compromise through the download of a malicious Microsoft Office document containing Visual Basic for Applications (VBA) macros. This document was successfully extracted by the framework, including the download URL. The embedded macro code was analyzed using the olevba tool, revealing that the macro downloaded additional malware intended for reflective DLL injection. The downloaded file was identified both via artifact analysis and through Sigma rule matches. This file subsequently initiated a Meterpreter reverse shell, which was also detected and analyzed by the framework. Subsequently, the scenario involved the download and execution of Mimikatz. Although the corresponding file was not recovered, the Sigma rule `win_alert_mimikatz_keywords` was triggered, indicating its presence. Later stages of the attack included the deployment of ransomware and process hollowing tools. These files were discovered in the Recycle Bin and documented in the report; however, they could not be analyzed further as they had been deleted from disk. The final malicious file, responsible for shellcode injection, was successfully extracted and analyzed, and its results were included in the report. The last action performed by simulated attackers was to clear the event logs which was also detected via corresponding Sigma rules.

Indicator of Compromise	Analysts	Framework
Office Document with Macros	yes	yes
Reflective DLL Injection	yes	yes
Meterpreter	yes	yes
Mimikatz	yes	no
Ransomware	yes	no
Process Hollowing	yes	no
Shell Code Injection	yes	yes
Event Logs Cleared	yes	yes

Table 9: The table shows the comparison between the IoC detections of analysts and the framework of the simulated incident.

6.6 Artifacts Distribution

The files were analyzed across a variety of Windows artifacts. Figure 4 presents a bar chart showing the distribution of analyzed artifacts, indicating how many files were examined per artifact and where malware was identified. The total number of files exceeds the overall number of analyzed files because a single file can appear in multiple artifacts. This also applies to the identified malware.

The majority of files were found in the AmCache application files and the Windows Services artifact. A significant number of files were also located in download artifacts of browsers, particularly in Image 2.1, which accounted for over 80 % of all downloaded files. Some images lacked specific artifacts entirely. For example, Image 1.1 contained no StartupInfo, while Prefetch files and Background Activity Moderator (BAM) were absent in Image 1.2 and 1.3. This can be attributed to the fact that BAM was introduced with Windows 10, whereas Images 1.2 and 1.3 are based on Windows 8. Moreover, Prefetching can be disabled via the Windows Registry, which may explain its absence. Most Recently Used (MRU) entries for Microsoft Office were missing in Images 2.1 and T.2, and no download artifacts were present in Image T.2. Most malware was identified via the Prefetch files and UserAssist artifact. Additional detections occurred through BAM, Services, and the AmCache application files but the malware found there was also identified through the Prefetch files. Notably, three of the malware samples found in UserAssist were also identified in the Prefetch files. Malware located in download artifacts, MRU Microsoft Office, and Runkeys was exclusively identified through these respective artifacts.

Kirby was used to extract artifacts which by default invokes 57 Dissect plugins. Of these, the 10 most relevant artifacts were incorporated into the automated analysis by the PYRAMID Framework, while the remaining artifacts were manually reviewed for malware and previously reported IoCs. No additional malware was identified during this manual

verification, indicating that the framework's artifact analysis successfully included all relevant artifacts that could be extracted using the employed tools.

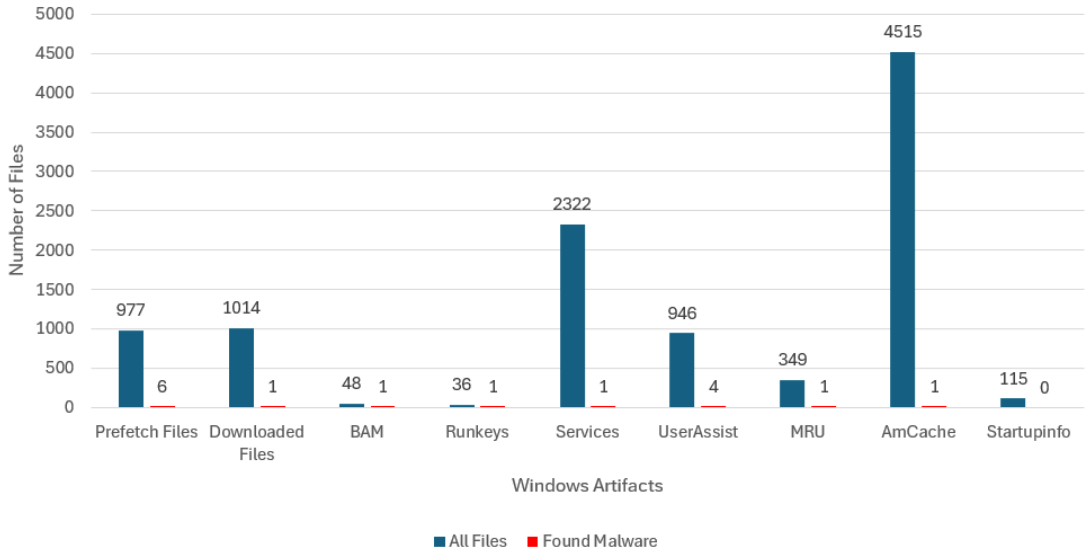


Figure 4: The bar chart shows the distribution from which artifacts the analyzed files originated and through which artifact malware could be identified.

6.7 Sigma Rule Detections

IoCs were not only identified through artifact analysis and the presence of malware, but also through the evaluation of event logs using Sigma rules. The results of this analysis are presented in Figure 5, which displays all Sigma rule detections across all analyzed images. In total, over 1.3 million events were analyzed across all examined images using Sigma. Approximately 41,000 events were flagged as potentially malicious. Of these, 1,565 events could be correlated with confirmed malicious activity in the compromised images, with 1,527 of them originating from a single malicious campaign. The bar chart highlights those Sigma rules that produced detections in all images and/or were associated with events deemed malicious. All other rules with detections are grouped under the category *Other Rules*. In total, 39 distinct Sigma rules triggered detections, eight of which were able to identify actual malicious activity. However, 96 % of all events flagged by Sigma rules could not be linked to verified malicious behavior as determined by the analyst reports. For instance, the rules `win_defender_real_time_protection_errors` (R1), `win_applocker_file_was_not_allowed_to_run` (R2), `win_defender_restored_quarantine_file` (R7) and `sysmon_wmi_event_subscription` (R9) generated detections across all disk images, yet none of these detections corresponded to confirmed malicious activity.

In contrast, the `rules win_security_audit_log_cleared` (R3) and `win_system_susp_eventlog_cleared` (R6) successfully identified `win_alert_mimikatz_keywords` (R4) were linked to the installation of Mimikatz on Image T.1. Two of the nine events detected by `proc_tampering_susp_process_hollowing` (R11) corresponded to the use of process hollowing tools on Image T.1. Additionally, the rule `win_defender_threat` (R5) detected two events, and `posh_pm_alternate_powershell_hosts` (R10) also identified two events related to the installation of Meterpreter on Image T.1. The remaining 1,527 malicious events detected by `posh_pm_alternate_powershell_hosts` (R10) were associated with the download of a cryptominer via specific URLs on Image 1.1. The rule `win_system_meterpreter_or_cobaltstrike_getsystem_service_installation` (R8) detected the cryptominer installation. Concluding the rule `win_system_service_install_paexec` (R12) identified two suspicious service installations on Images 1.2 and 1.3.

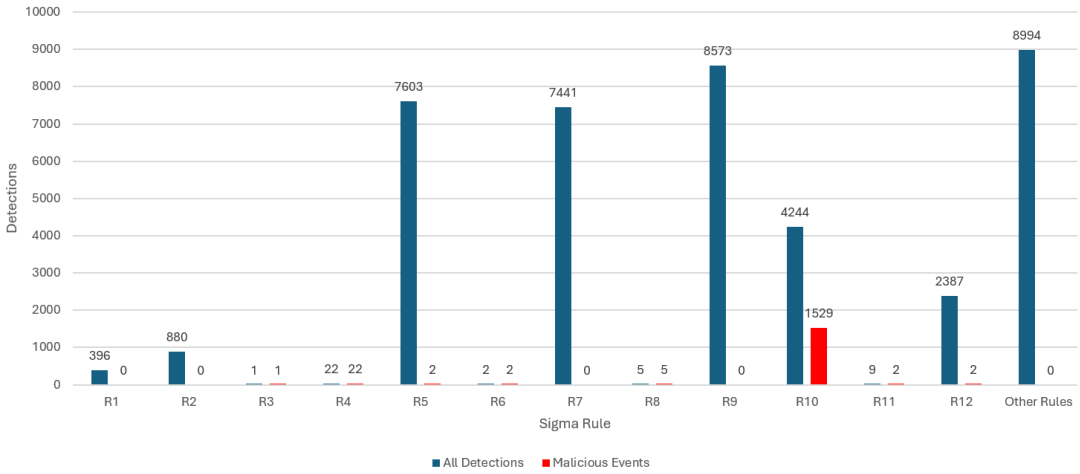


Figure 5: The bar chart shows all detected events with Sigma rules and which rules detected actual malicious events. For readability, rules are labeled as R1, R2, etc., the corresponding mapping to the full Sigma rule names is provided in the text.

6.8 Execution Time

The evaluation was conducted on a ThinkPad T14 with 8 CPU cores, 32 GB of RAM and a 500 GB SSD. To analyze the overall execution time of the framework and its analyses, box plots were generated, as shown in Figure 6. Due to the significant differences in execution times between the various analysis steps, different time scales were applied for each plot. The shortest execution times were observed in the basic analysis and the report generation phases, with median durations of 0.089 and 0.5095 seconds, respectively. The report generation exhibited one outlier associated with Image 1.1; however, this outlier still remained below approximately one second. The outlier of the basic analysis stemmed from the same image. These short execution times can be attributed to the relatively consistent amount of data extracted

during the basic analysis across all disk images, as well as to the efficient implementation of the reporting component. The reports were generated using Jinja templates, which allowed for rapid creation of the HTML output.

The hash lookup in the whitelist was also executed very quickly. Due to the use of a binary search algorithm, the median execution time was only 18.5 seconds. Image 1.1 required more time for this step, which can be explained by its larger size (250 GB) and the significantly higher number of files to be verified. Both the log and file type analyses were completed within a few minutes. Execution times were relatively consistent across all images, with no significant outliers. For log analysis, the execution time ranged predominantly between 10.25 and 14.75 minutes, with the majority of the time spent on Sigma rule evaluation. File type analysis was predominantly completed within a range of 5.25 to 12.5 minutes. The artifact analysis step included the extraction and evaluation of various forensic artifacts and also the extraction of ADS and the MFT. The median execution time for this step was higher, at 26.5 minutes. Image 1.1 again required significantly more time, taking over two hours to extract and analyze all artifacts. The most time-consuming part of this phase was the extraction of the MFT. When combining the execution times of all previously described analyses, the total median was 43.5 minutes. In contrast, Image 1.1 required a total of 174 minutes for the same steps.

The final analysis step was the dynamic malware analysis using the VMRay sandbox. This was by far the most time-intensive phase, with a median execution time of 22.5 hours. The analysis times typically ranged between 7.5 and 24 hours. Image 1.1, however, exceeded 72 hours because dynamic malware analysis requires each file to be executed and observed within an isolated virtual environment, making the process computationally expensive and inherently time-consuming. In contrast, images T.1 and T.2, each containing fewer files, were processed in approximately three hours. This difference illustrates that the total analysis time is strongly influenced by the number of files requiring dynamic execution, even though VMRay can perform multiple analyses in parallel. Taking the sandbox analysis into account, the overall median execution time of the full analysis pipeline was 22.5 hours.

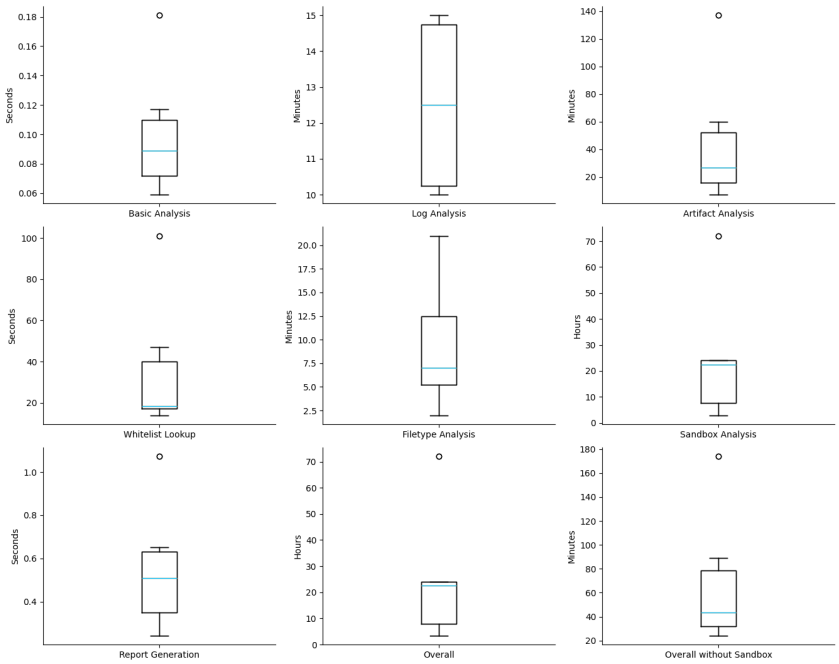


Figure 6: The box plots show the execution times of analysis steps and the overall execution time of the analysis framework.

6.9 Analysis Timeline of Analysts

To enable a comparison between manual analysis and the automated framework, a timeline was constructed based on the real activities of analysts during the investigation of the first incident, from which Images 1.1, 1.2, and 1.3 originate (see Figure 7). The analysts worked on the overall incident for a total of 40 days. The three selected disk images, which were also analyzed by the framework, were not the only ones they investigated. Relevant segments of their work are visualized in the timeline. The analysts dedicated four days in total to the analysis of the three images in question. However, it cannot be ruled out that additional time was spent beyond what is documented, nor that the analysts worked full days on each occasion.

On the first day, they examined the file system and event logs of Images 1.2 and 1.3, where they identified the suspicious services and scheduled tasks. From day 2 to day 6, their focus shifted to other disk images. On day 7, they conducted malware and event log analysis on Image 1.1, successfully identifying the cryptominer. Additional file system analysis was performed on days 12 and 13, during which further malicious artifacts were discovered, including the backdoor and keylogger.

Compared to this timeline, the execution time of the automated analysis framework allows for a more efficient workflow. The framework can run continuously until all analysis tasks are completed, which would take approximately five days for a comprehensive assessment of all three images. On the first day, both event log and file system analyses can be fully completed, along with the initial phase of malware analysis on Image 1.1 using VMRay. The analysis of Image 1.1 would be finalized by day 3, with the malware analyses of the remaining two images conducted on days 4 and 5.

While the overall duration is one day longer than the documented time spent by analysts, this estimate assumes that all files are subjected to VMRay analysis. Importantly, all event log and file system results would already be available on day 1. Furthermore, since the VMRay module operates in a triage mode, the most suspicious files are prioritized, and preliminary results are available after each scan. As a result, the framework can provide valuable early indicators for the analysts, guiding their attention to relevant files and events. In parallel with the framework's execution, analysts can continue working on other parts of the incident, maximizing overall efficiency.

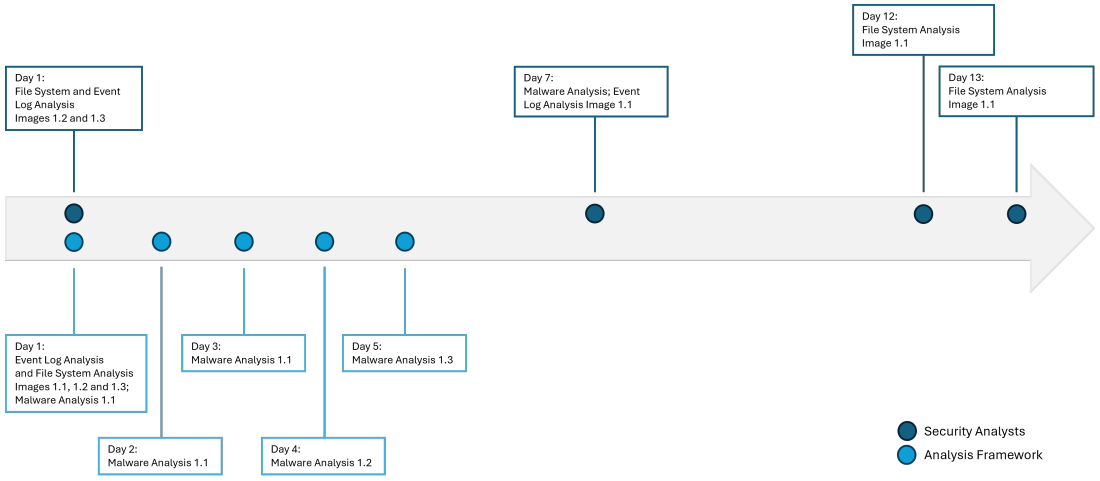


Figure 7: The figure shows the timeline of the first incident which was analyzed by security analysts and the automated incident framework.

7 Discussion and Limitations

This chapter discusses whether the defined research questions RQ1 and RQ2 could be answered and outlines the limitations of the framework.

7.1 Discussion

With respect to the first research question RQ1 evaluations regarding the performance of the framework's compromise rating system and the overall analysis results were conducted. The findings demonstrated that the framework was capable of correctly identifying the genuinely compromised disk images, assigning them an appropriate compromise rating. In total, 11 out of 16 IoCs were successfully detected. Missed indicators were primarily due to deleted files. However, the integration with event log analysis proved effective, as associated event traces could still point to malicious activity even in the absence of the original files. Nonetheless, the overall classification accuracy of the framework was relatively low, at 0.33, primarily due to a tendency to overestimate compromise on clean images. This was largely attributed to high false positive rates in third-party tools used in the analysis, particularly VMRay (23 %) and Sigma (96 %). Despite this limitation, the framework demonstrated strong potential in data reduction, managing to exclude 99.86 % of all files from further analysis and focusing only on relevant artifacts. This represents a significant advantage both in terms of execution time and in reducing the workload for analysts reviewing the results. In conclusion, while the compromise rating mechanism requires further refinement to improve accuracy, the analysis results highlight the framework's value as a preliminary triage tool. It supports analysts by filtering out large volumes of irrelevant data, identifying potential IoCs, and offering an initial assessment of disk images. However, the final determination of whether a system has been compromised must ultimately be made by experienced security analysts.

To address the second research question RQ2, the execution times of the framework's analyses were evaluated and compared to the timeline of security analysts working on a real-world incident involving three compromised disk images. The analyst required a total of four days to complete the investigation and identify the relevant IoCs, whereas the framework would have required approximately five days. However, in the actual incident, analysts were responsible for reviewing multiple images, and the final analyses were not completed until day 13 of the investigation. In contrast, the automated framework which is capable of continuous operation, would have produced complete results by day five. Notably, file system and event log analyses for all three images would have been available as early as the first day. The framework exhibited solid performance across seven out of eight analysis modules. While execution time naturally increased with the size of the disk image, even the largest image in the evaluation (250 GB) was processed in under three hours, excluding the dynamic malware analysis conducted with VMRay. The dynamic malware analysis component, although highly valuable, presented a performance bottleneck. This challenge was mitigated through a triage mechanism and the generation of preliminary reports, ensuring that the most relevant findings were prioritized for analysts.

In summary, while it remains difficult to precisely quantify the amount of time that analysts could save using the framework, it is evident that the automation significantly enhances efficiency. By operating in parallel and without interruption, the framework is capable of accelerating the investigative process and supporting analysts with timely, prioritized insights.

7.2 Limitations

Certain types of analyses were not integrated into the framework due to their high complexity or limited feasibility for automation. For instance, the incorporation of antivirus scanning tools or commercial forensic software such as X-Ways [X-W26] would require advanced GUI automation techniques, including the emulation of keystrokes and mouse actions. However, such methods are not only time-consuming to implement but also inherently unreliable, as graphical interfaces may change or behave inconsistently across systems. Additionally, limitations arose in the completeness of artifact extraction. Some files could not be analyzed due to constraints in how Dissect handles data extraction. Specifically, certain artifacts identified by Dissect lacked usable file path information, making it difficult or impossible to locate the corresponding files on the disk image. In some cases, even when paths were extracted, further parsing was required for Dissect to utilize them effectively. Despite being generated by Dissect itself, not all paths could be re-parsed or mapped back to accessible file locations, resulting in gaps in the analysis.

Although several analysis modules were successfully automated, integrating their results into a unified and reliable compromise rating remains challenging. The interpretation of tool outputs often depends heavily on the specific context of the incident and the environment in which the system operated. What is flagged as malicious in one case may be considered legitimate in another. As a result, automated evaluation for rating purposes can lead to inaccuracies. Nevertheless, all tool outputs are included in the final report, allowing analysts to conduct their own assessments based on the full set of available data.

One significant issue observed was the high rate of false positive *suspicious* verdicts resulting from VMRay outputs. In numerous cases, VMRay assigned an overall rating of *suspicious* even when both the reputation and static analysis components indicated a clean result, with only the dynamic analysis suggesting potential suspicious behavior. This discrepancy contributed to incorrect compromise classifications. A potential solution would be to investigate whether VMRay can be configured to skip dynamic analysis or override its results when reputation analysis confirms that a file is known to be clean. Such a configuration could not only improve rating accuracy but also reduce analysis time by avoiding unnecessary scans.

During the development phase, example data from one disk image was used to implement and test the framework. As a result, not all potential states and variations of artifacts could be accounted for during implementation. To enhance the robustness and reliability of artifact processing, a broader range of sample images should be utilized. This would help verify whether all types of artifacts are correctly extracted and interpreted under diverse conditions.

The final evaluation of the framework was conducted using six test disk images. While these provided valuable insights, the number of test cases is insufficient for a comprehensive statistical analysis. To draw more generalizable conclusions about the framework's performance, further testing with a larger dataset is necessary. Ideally, this should include disk images from real-world security incidents to ensure realistic and representative evaluation scenarios.

8 Future Work and Conclusion

The implementation and evaluation of the framework offer several opportunities for future extension and improvement. Additional modules with extended functionality could be developed to enhance the analytical capabilities of the framework. For example, the integration of file carving techniques could allow for the recovery of deleted files, thereby reducing the number of artifacts that are currently missed due to their absence from the file system.

Further, new analysis categories such as the examination of email artifacts could broaden the applicability of the framework beyond IA and make it more suitable for use in DF investigations. Support for additional operating systems, such as Linux, would also increase the framework's flexibility and usability in diverse environments. Enhancements to input flexibility, such as the inclusion of specific timestamps, file signatures, or other contextual indicators, could contribute to more precise and incident-specific analyses. To improve performance and scalability, particularly with large disk images, future versions of the framework could implement parallelized processing of independent modules.

Most importantly, the framework should be validated and refined through real-world use by experienced analysts during live incident response scenarios. This would provide valuable feedback regarding its practical utility. However, it must be emphasized that no automated system can replace the expertise and judgment of skilled security analysts. The framework should be viewed as a supportive tool designed to augment, not substitute, human decision-making in security investigations.

Overall, this paper demonstrated that the proposed framework is capable of automatically detecting system compromises, thereby significantly reducing the workload of human analysts and providing a practical solution to a real-world problem. While the framework did not identify analysis results in exactly the same manner as human analysts, it successfully detected relevant IoCs and proved its operational value. These differences highlight opportunities for further refinement. In particular, the integration of artificial intelligence techniques offers substantial potential to enhance detection accuracy and further narrow the gap between automated and human-driven analysis in future work.

ERNW RESEARCH pursuing knowledge.

9 References

[BDO26] BDO Cyber Security. *dfir-dd*. 2026. URL: https://github.com/dfir-dd (visited on Aug. 4, 2026).

[Bun24] Bundesamt für Sicherheit in der Informationstechnik. *Die Lage der IT-Sicherheit in Deutschland 2024*. Bonn, North Rhine-Westphalia, Germany: Bundesamt für Sicherheit in der Informationstechnik, 2024. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Lagebericht2024.pdf?__blob=publicationFile&v=5 (visited on Aug. 4, 2026).

[Bun25] Bundesamt für Sicherheit in der Informationstechnik. *Die Lage der IT-Sicherheit in Deutschland 2025*. Bonn, North Rhine-Westphalia, Germany: Bundesamt für Sicherheit in der Informationstechnik, 2025. URL: https://medien.bsi.bund.de/lagebericht/de/index.html (visited on Aug. 4, 2026).

[Car04] Brian Carrier. *The Sleuth Kit Informer*. 2004. URL: https://www.sleuthkit.org/informer/sleuthkit-informer-12.txt.

[Car15] Brian Carrier. *File system forensic analysis*. Upper Saddle River, NJ, USA: Addison-Wesely, 2015.

[Cic+12] Paul Cichonski, Tom Millar, Tim Grance, et al. *Computer security incident handling guide : Recommendations of the national institute of standards and technology*. Tech. rep. National Institute of Standards and Technology, Aug. 2012.

[Coh+11] M.I. Cohen, D. Bilby, and G. Caronni. "Distributed forensics and incident response in the enterprise." In: *Digital Investigation* 8 (2011). The Proceedings of the Eleventh Annual DFRWS Conference, S101–S110. ISSN: 1742-2876. DOI: https://doi.org/10.1016/j.diin.2011.05.012. URL: https://www.sciencedirect.com/science/article/pii/S1742287611000363.

[Dew+15] Andreas Dewald and Felix C. Freiling. *Forensische Informatik*. Norderstedt, Schleswig-Holstein, Germany: BoD - Books on Demand, 2015.

[Ela26] Elasticsearch B.V. *Elastic Stack*. 2026. URL: https://www.elastic.co/de/elastic-stack (visited on Aug. 4, 2026).

[Eur24] European Union Agency for Cybersecurity (ENISA). *Foresight Cybersecurity Threats for 2030 - Update*. Chalandri, Attiki, Greece: European Union Agency for Cybersecurity (ENISA), 2024. URL: https://www.enisa.europa.eu/sites/default/files/2024-11/Foresight%20Cybersecurity%20Threats%20for%202030-Update-fullreport_en_0.pdf (visited on Aug. 4, 2026).

[Fak+24] Hussam Fakhouri, Mohammad Alsharaiah, Ahmad Al Hwaitat, et al. "Overview of Challenges Faced by Digital Forensic." In: *2024 2nd International Conference on Cyber Resilience (ICCR)*. Dubai, Emirate of Dubai, United Arab Emirates: IEEE, Feb. 2024, pp. 1–8. DOI: 10.1109/ICCR61006.2024.10532850.

[Far09] Paul F. Farrell Jr. "A Framework for Automated Digital Forensic Reporting." MA thesis. Naval Postgraduate School, 2009. URL: https://apps.dtic.mil/sti/citations/ADA496800.

[Fox23] Fox-IT. *Dissect target-fs*. 2023. URL: https://docs.dissect.tools/en/stable/tools/target-fs.html (visited on Aug. 4, 2026).

[Fox26] Fox-IT. *Dissect GitHub Reposity*. 2026. URL: https://github.com/fox-it/dissect/ (visited on Aug. 4, 2026).

[Gar10] Simson Garfinkel. "Garfinkel, S.L.: Digital Forensics Research: The Next 10 Years. Digital Investigation 7(suppl.), 64-73." In: *Digital Investigation* 7 (Aug. 2010). DOI: 10.1016/j.diin.2010.05.009.

[Kar+16] Umit Karabiyik and Sudhir Aggarwal. "Advanced Automated Disk Investigation Toolkit." In: *Advances in Digital Forensics XII*. Ed. by Gilbert Peterson and Sujeet Shenoi. Cham: Springer International Publishing, 2016, pp. 379–396. ISBN: 978-3-319-46279-0.

[Lag26] Philippe Lagadec. *python-oletools*. 2026. URL: https://github.com/decalage2/oletools (visited on Aug. 4, 2026).

[Lut+14] Jason T. Luttgens, Matthew Pepe, Ryan Kazanciyan, et al. *Incident response and computer forensics, third edition: Jason T. Luttgens, Matthew Pepe and Kevin Mandia*. New York, NY, USA: McGraw-Hill Education, 2014.

[Men26] Mente Binária. *readpe*. 2026. URL: https://github.com/mentebinaria/readpe (visited on Aug. 4, 2026).

[NIS26] NIST. *National Software Reference Library (NSRL)*. 2026. URL: https://www.nist.gov/itl/ssd/software-quality-group/national-software-reference-library-nsrl (visited on Aug. 4, 2026).

[ReF26] ReFirm Labs, Inc. *binwalk*. 2026. URL: https://github.com/ReFirmLabs/binwalk (visited on Aug. 4, 2026).

[Sie21] Markus Siemens. *TinyDB*. 2021. URL: https://tinydb.readthedocs.io/en/latest/index.html (visited on Aug. 4, 2026).

[Sig26] SigmaHQ. *Sigma*. 2026. URL: https://github.com/SigmaHQ/sigma (visited on Aug. 3, 2026).

[sim26] simsong. *bulk_extractor*. 2026. URL: https://github.com/simsong/bulk_extractor (visited on Aug. 4, 2025).

[Smi+17] Jared M. Smith, Elliot Greenlee, and Aaron Ferber. "DEMO: Akatosh: Automated Cyber Incident Verification and Impact Analysis." In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS '17. Dallas, Texas, USA: Association for Computing Machinery, 2017, pp. 2463–2465. ISBN: 9781450349468. DOI: 10.1145/3133956.3138854. URL: https://doi.org/10.1145/3133956.3138854.

[Spr+21] Jonathan Spring and Phyllis Illari. "Review of human decision-making during computer security incident analysis." In: *Digital Threats: Research and Practice* 2.2 (Apr. 2021), pp. 1–47. DOI: 10.1145/3427787.

[Sta25] StatCounter. *Operating System Market Share Worldwide*. 2025. URL: https://gs.statcounter.com/os-market-share#monthly-202502-202502-bar (visited on Aug. 4, 2026).

[Ste+17] Christopher Stelly and Vassil Roussev. "SCARF: A container-based approach to cloud-scale digital forensic processing." In: *Digital Investigation* 22 (2017), S39–S47. ISSN: 1742-2876. DOI: https://doi.org/10.1016/j.diin.2017.06.008. URL: https://www.sciencedirect.com/science/article/pii/S1742287617301950.

[Ste26] Didier Stevens. *DidierStevensSuite PDF Tools*. 2026. URL: https://github.com/DidierStevens/DidierStevensSuite (visited on Aug. 4, 2026).

[The26] The MITRE Corporation. *MITRE ATT&CK framework*. 2026. URL: https://attack.mitre.org/ (visited on Aug. 4, 2026).

[Vir22] VirusTotal. *YARA Documentation*. 2022. URL: https://yara.readthedocs.io/en/latest/index.html (visited on Aug. 4, 2026).

[VMR26] VMRay. *VMRay Technologies*. 2026. URL: https://www.vmray.com/why-vmray/technology/ (visited on Aug. 4, 2026).

[X-W26] X-Ways AG. *X-Ways*. 2026. URL: https://www.x-ways.net/forensics/ (visited on Aug. 4, 2026).

A Supplementary Material

Figure 8 shows the complete workflow of the PYRAMID Framework, which is referenced in Section 4. It illustrates the full structure of the analysis workflow, including all integrated tools.

Figure 9 – Figure 13 show an example report generated from the test disk image T.1 of the simulated incident using synthetic data.

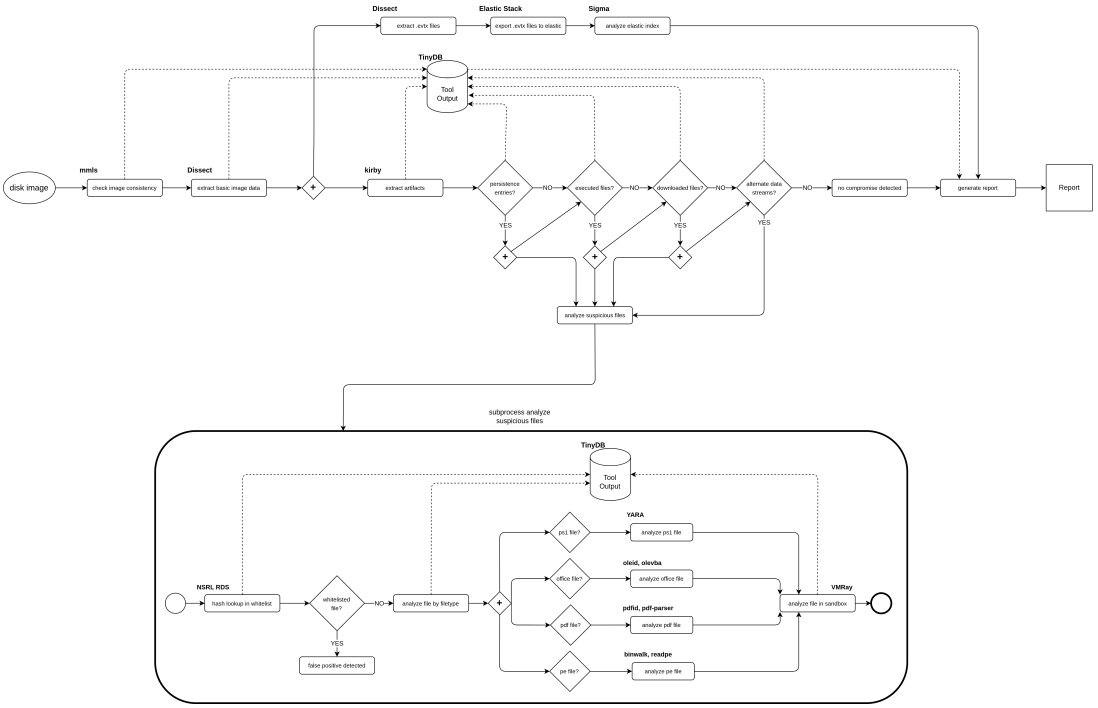
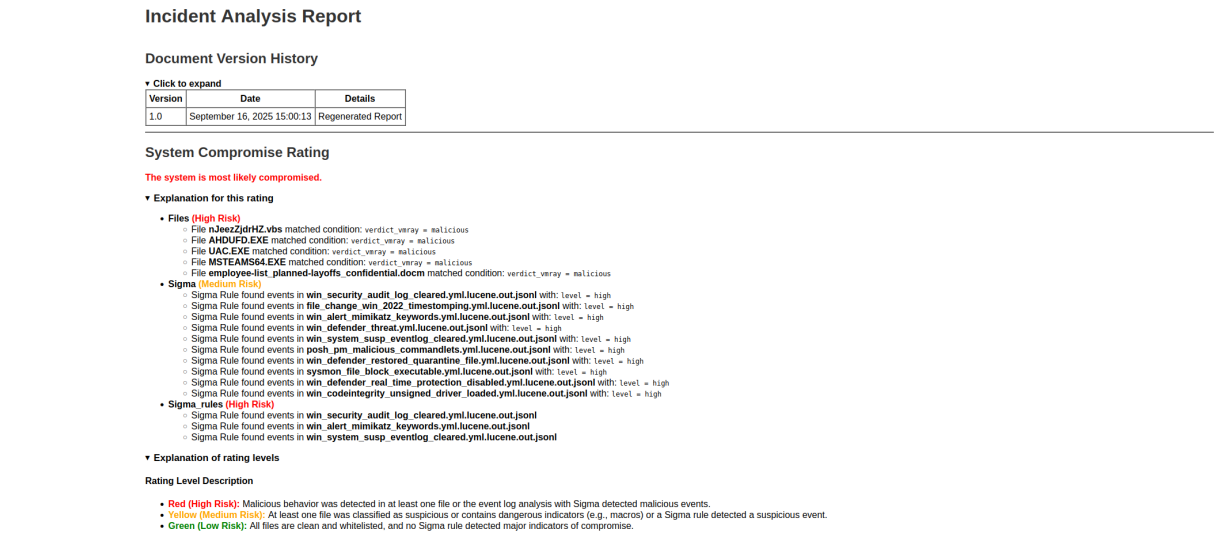


Figure 8: The full workflow with all analysis steps and the subprocess for malware analysis.



Incident Analysis Report

Document Version History

▾ Click to expand

Version	Date	Details
1.0	September 16, 2025 15:00:13	Regenerated Report

System Compromise Rating

The system is most likely compromised.

▾ Explanation for this rating

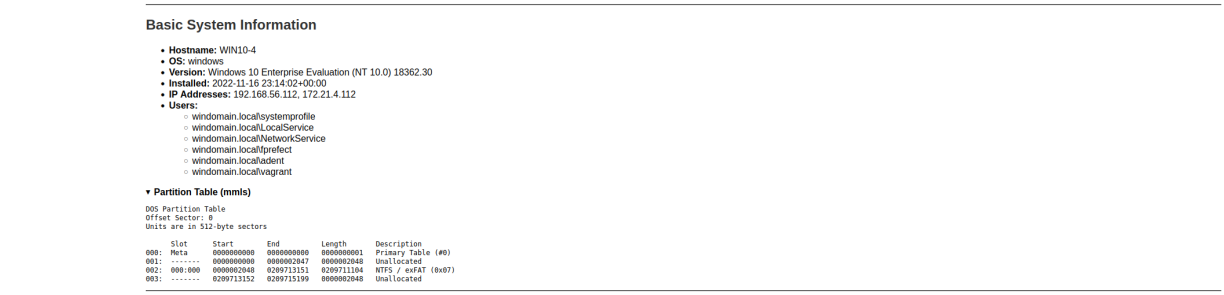
- **Files** (High Risk)
  - File **nJeezZjdrHZ.vbs** matched condition: `verdict_vmray = malicious`
  - File **AHDUFD.EXE** matched condition: `verdict_vmray = malicious`
  - File **UAC.EXE** matched condition: `verdict_vmray = malicious`
  - File **MSTEAMS64.EXE** matched condition: `verdict_vmray = malicious`
  - File **employee-list_planned-layoffs_confidential.docm** matched condition: `verdict_vmray = malicious`
- **Sigma** (Medium Risk)
  - Sigma Rule found events in **win_security_audit_log_cleared.yml.lucene.out.jsonl** with: `level = high`
  - Sigma Rule found events in **file_change_win_2022_timestomping.yml.lucene.out.jsonl** with: `level = high`
  - Sigma Rule found events in **win_alert_mimikatz_keywords.yml.lucene.out.jsonl** with: `level = high`
  - Sigma Rule found events in **win_defender_threat.yml.lucene.out.jsonl** with: `level = high`
  - Sigma Rule found events in **win_system_susp_eventlog_cleared.yml.lucene.out.jsonl** with: `level = high`
  - Sigma Rule found events in **posh_pm_malicious_commandlets.yml.lucene.out.jsonl** with: `level = high`
  - Sigma Rule found events in **win_defender_restored_quarantine_file.yml.lucene.out.jsonl** with: `level = high`
  - Sigma Rule found events in **sysmon_file_block_executable.yml.lucene.out.jsonl** with: `level = high`
  - Sigma Rule found events in **win_defender_real_time_protection_disabled.yml.lucene.out.jsonl** with: `level = high`
  - Sigma Rule found events in **win_codeintegrity_unsigned_driver_loaded.yml.lucene.out.jsonl** with: `level = high`
- **Sigma_rules** (High Risk)
  - Sigma Rule found events in **win_security_audit_log_cleared.yml.lucene.out.jsonl**
  - Sigma Rule found events in **win_alert_mimikatz_keywords.yml.lucene.out.jsonl**
  - Sigma Rule found events in **win_system_susp_eventlog_cleared.yml.lucene.out.jsonl**

▾ Explanation of rating levels

Rating Level Description

- **Red (High Risk):** Malicious behavior was detected in at least one file or the event log analysis with Sigma detected malicious events.
- **Yellow (Medium Risk):** At least one file was classified as suspicious or contains dangerous indicators (e.g., macros) or a Sigma rule detected a suspicious event.
- **Green (Low Risk):** All files are clean and whitelisted, and no Sigma rule detected major indicators of compromise.

Figure 9: The screenshot shows the document version history and the compromise rating.



Basic System Information

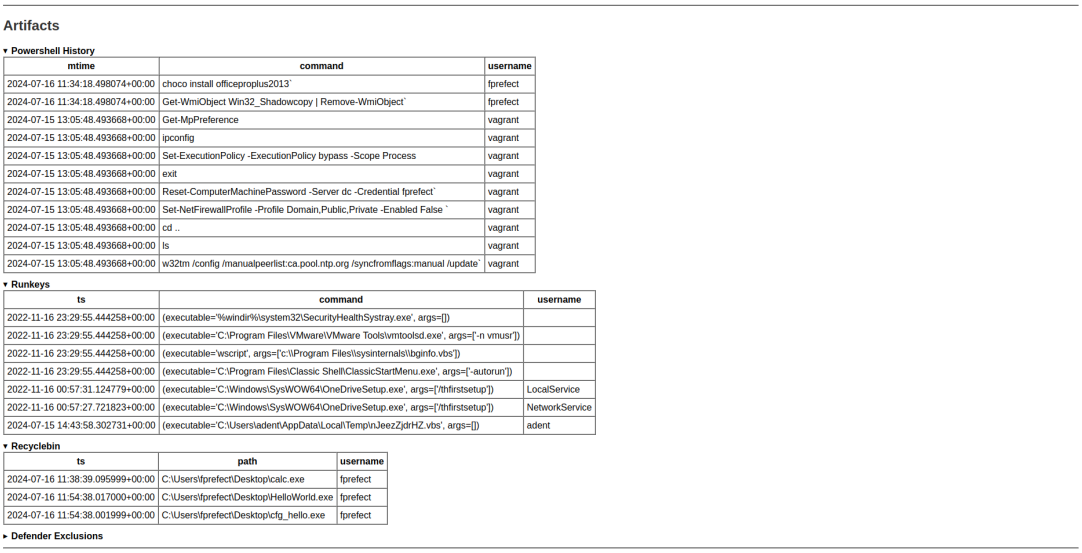
- **Hostname:** WIN10-4
- **OS:** windows
- **Version:** Windows 10 Enterprise Evaluation (NT 10.0) 18362.30
- **Installed:** 2022-11-16 23:14:02+00:00
- **IP Addresses:** 192.168.56.112, 172.21.4.112
- **Users:**
  - windomain.local\systemprofile
  - windomain.local\LocalService
  - windomain.local\NetworkService
  - windomain.local\fprefect
  - windomain.local\adent
  - windomain.local\vagrant

▾ Partition Table (mmls)

```
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

      Slot      Start        End          Length       Description
000:  Meta      0000000000   0000000000   0000000001   Primary Table (#0)
001:  -------   0000000000   0000002047   0000002048   Unallocated
002:  000:000   0000002048   0209713151   0209711104   NTFS / exFAT (0x07)
003:  -------   0209713152   0209715199   0000002048   Unallocated
```

Figure 10: The screenshot shows basic information about the analyzed disk image.



Artifacts

▾ Powershell History

mtime	command	username
2024-07-16 11:34:18.498074+00:00	choco install officeproplus2013`	fprefect
2024-07-16 11:34:18.498074+00:00	Get-WmiObject Win32_Shadowcopy | Remove-WmiObject`	fprefect
2024-07-15 13:05:48.493668+00:00	Get-MpPreference	vagrant
2024-07-15 13:05:48.493668+00:00	ipconfig	vagrant
2024-07-15 13:05:48.493668+00:00	Set-ExecutionPolicy -ExecutionPolicy bypass -Scope Process	vagrant
2024-07-15 13:05:48.493668+00:00	exit	vagrant
2024-07-15 13:05:48.493668+00:00	Reset-ComputerMachinePassword -Server dc -Credential fprefect`	vagrant
2024-07-15 13:05:48.493668+00:00	Set-NetFirewallProfile -Profile Domain,Public,Private -Enabled False `	vagrant
2024-07-15 13:05:48.493668+00:00	cd ..	vagrant
2024-07-15 13:05:48.493668+00:00	ls	vagrant
2024-07-15 13:05:48.493668+00:00	w32tm /config /manualpeerlist:ca.pool.ntp.org /syncfromflags:manual /update`	vagrant

▾ Runkeys

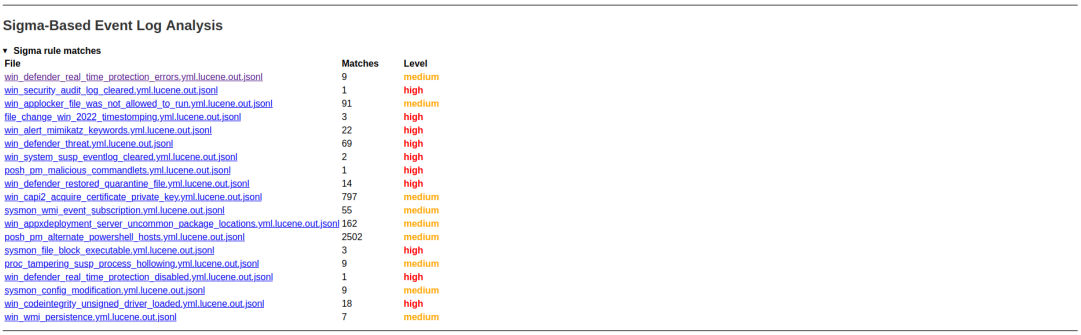
ts	command	username
2022-11-16 23:29:55.444258+00:00	(executable='%windir%\system32\SecurityHealthSystray.exe', args=[])	
2022-11-16 23:29:55.444258+00:00	(executable='C:\Program Files\VMware\VMware Tools\vmtoolsd.exe', args=['-n vmusr'])	
2022-11-16 23:29:55.444258+00:00	(executable='wscript', args=['c:\\Program Files\\sysinternals\\bginfo.vbs'])	
2022-11-16 23:29:55.444258+00:00	(executable='C:\Program Files\Classic Shell\ClassicStartMenu.exe', args=['-autorun'])	
2022-11-16 00:57:31.124779+00:00	(executable='C:\Windows\SysWOW64\OneDriveSetup.exe', args=['/thfirstsetup'])	LocalService
2022-11-16 00:57:27.721823+00:00	(executable='C:\Windows\SysWOW64\OneDriveSetup.exe', args=['/thfirstsetup'])	NetworkService
2024-07-15 14:43:58.302731+00:00	(executable='C:\Users\adent\AppData\Local\Temp\nJeezZjdrHZ.vbs', args=[])	adent

▾ Recyclebin

ts	path	username
2024-07-16 11:38:39.095999+00:00	C:\Users\fprefect\Desktop\calc.exe	fprefect
2024-07-16 11:54:38.017000+00:00	C:\Users\fprefect\Desktop\HelloWorld.exe	fprefect
2024-07-16 11:54:38.001999+00:00	C:\Users\fprefect\Desktop\cfg_hello.exe	fprefect

▸ Defender Exclusions

Figure 11: The screenshot shows an excerpt of analyzed artifacts.

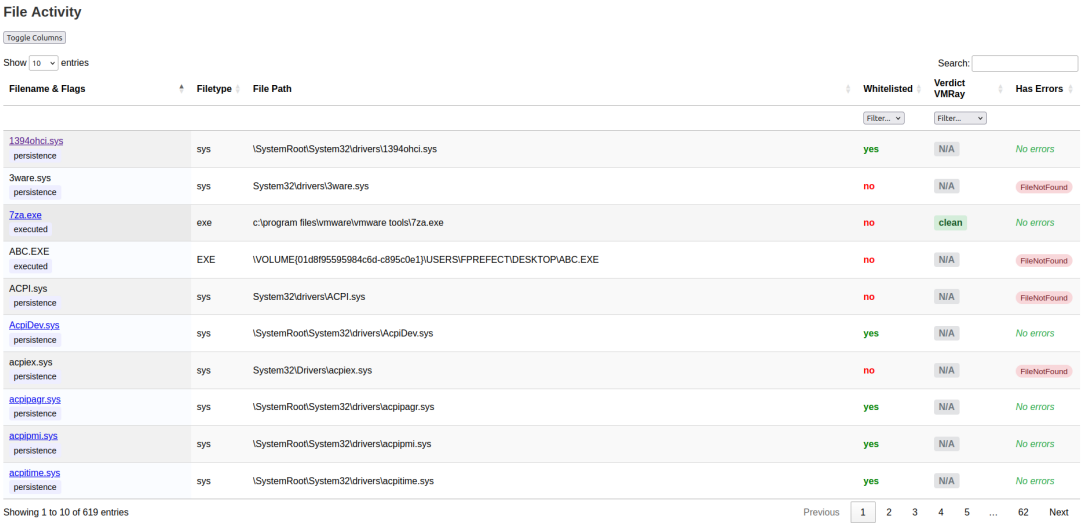


Sigma-Based Event Log Analysis

▾ Sigma rule matches

File	Matches	Level
win_defender_real_time_protection_errors.yml.lucene.out.jsonl	9	medium
win_security_audit_log_cleared.yml.lucene.out.jsonl	1	high
win_applocker_file_was_not_allowed_to_run.yml.lucene.out.jsonl	91	medium
file_change_win_2022_timestomping.yml.lucene.out.jsonl	3	high
win_alert_mimikatz_keywords.yml.lucene.out.jsonl	22	high
win_defender_threat.yml.lucene.out.jsonl	69	high
win_system_susp_eventlog_cleared.yml.lucene.out.jsonl	2	high
posh_pm_malicious_commandlets.yml.lucene.out.jsonl	1	high
win_defender_restored_quarantine_file.yml.lucene.out.jsonl	14	high
win_capi2_acquire_certificate_private_key.yml.lucene.out.jsonl	797	medium
sysmon_wmi_event_subscription.yml.lucene.out.jsonl	55	medium
win_appxdeployment_server_uncommon_package_locations.yml.lucene.out.jsonl	162	medium
posh_pm_alternate_powershell_hosts.yml.lucene.out.jsonl	2502	medium
sysmon_file_block_executable.yml.lucene.out.jsonl	3	high
proc_tampering_susp_process_hollowing.yml.lucene.out.jsonl	9	medium
win_defender_real_time_protection_disabled.yml.lucene.out.jsonl	1	high
sysmon_config_modification.yml.lucene.out.jsonl	9	medium
win_codeintegrity_unsigned_driver_loaded.yml.lucene.out.jsonl	18	high
win_wmi_persistence.yml.lucene.out.jsonl	7	medium

Figure 12: The screenshot shows results of the event log analysis with Sigma rules.



File Activity

Toggle Columns

Show 10 entries Search:

Filename & Flags	Filetype	File Path	Whitelisted	Verdict VMRay	Has Errors
			Filter...	Filter...	
1394ohci.sys persistence	sys	\SystemRoot\System32\drivers\1394ohci.sys	yes	N/A	No errors
3ware.sys persistence	sys	System32\drivers\3ware.sys	no	N/A	FileNotFound
7za.exe executed	exe	c:\program files\vmware\vmware tools\7za.exe	no	clean	No errors
ABC.EXE executed	EXE	\VOLUME{01d8f95595984c6d-c895c0e1}\USERS\FPREFECT\DESKTOP\ABC.EXE	no	N/A	FileNotFound
ACPI.sys persistence	sys	System32\drivers\ACPI.sys	no	N/A	FileNotFound
AcpiDev.sys persistence	sys	\SystemRoot\System32\drivers\AcpiDev.sys	yes	N/A	No errors
acpiex.sys persistence	sys	System32\Drivers\acpiex.sys	no	N/A	FileNotFound
acpipagr.sys persistence	sys	\SystemRoot\System32\drivers\acpipagr.sys	yes	N/A	No errors
acpipmi.sys persistence	sys	\SystemRoot\System32\drivers\acpipmi.sys	yes	N/A	No errors
acpitime.sys persistence	sys	\SystemRoot\System32\drivers\acpitime.sys	yes	N/A	No errors

Showing 1 to 10 of 619 entries Previous 1 2 3 4 5 ... 62 Next

Figure 13: The screenshot shows the table which contains an overview over all analyzed files.