AUTOMATED RECONSTRUCTIONS FOR THE DIGITAL FORENSIC EXAMINER WORKFLOW

THESIS

Ryan P. Montgomery, Captain, USAF

AFIT-ENG-MS-22-M-048

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-22-M-048

AUTOMATED RECONSTRUCTIONS FOR THE DIGITAL FORENSIC EXAMINER WORKFLOW

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Cyber Operations

Ryan P. Montgomery, B.S.

Captain, USAF

March 24, 2022

AFIT-ENG-MS-22-M-048

AUTOMATED RECONSTRUCTIONS FOR THE DIGITAL FORENSIC EXAMINER WORKFLOW

THESIS

Ryan P. Montgomery, B.S.
Captain, USAF

Committee Membership:

Gilbert L. Peterson, Ph.D
Chair

Douglas D. Hodson, Ph.D
Member

Robert F. Mills, Ph.D
Member

AFIT-ENG-MS-22-M-048

Abstract

One product of a digital forensics examination is a reconstruction of events recorded in the media. A reconstruction places all of the case relevant trace into temporal, identity and associative relationships. Creating this reconstruction is a manual and time consuming process for the examiner. This thesis presents Autopsy Integrated Event Reconstruction (AIER). AIER integrates automation, abstraction and visualization into the Autopsy forensic software to improve the reconstruction process. The integration utilizes a custom Autopsy ingest module to extract and abstract artifact data and an interactive graph-based timeline visualization module. These improvements to the forensic examiner workflow are evaluated through a series of use cases.

Acknowledgements

This research would not have been possible without the support of many people. I would like to thank my advisor, Dr. Gilbert Peterson, for his constant guidance, support, and most of all, his patience. I would also like to thank my mother, brother, and new found Ohio "family" for their continuous love, support, and motivation.

Ryan P. Montgomery

Table of Contents

Page

Page

List of Figures

AUTOMATED RECONSTRUCTIONS FOR THE DIGITAL FORENSIC EXAMINER WORKFLOW

I. Introduction

As technology continues to develop, data storage requirements are growing at an accelerated rate [1]. Modern consumer technology devices create, download, and utilize large amounts of data quickly [2]. Consumers also own multiple devices that perform a wide variety of tasks. The diversity of devices is increasing the workload a cyber examiner faces with each case [3].

Cyber related crime is the fastest growing crime in the United States [4]. Criminals are able to hide their crime efforts due to the sheer amount of data and number of criminals utilizing technology to break the law. The expansion of data sets coupled with increasing cybercrime rates, further increases the workload on investigators and analysts. Investigators are fighting a losing battle against an overload of data even with automated tools [5].

Much like the greater digital forensics field as a whole, the Department of Defense (DoD) and subsequently the United States Air Force (USAF) shares most of the same burdens. The DoD and USAF suffer from criminal case backlogs with the added burden of cyber defense investigations. Both criminal investigations and cyber incident investigations use tools like Autopsy. Digital forensics tools help with both investigation types but digital forensic examiners require more automated tools and techniques to alleviate case backlog.

Although developers have made major advances in digital forensic tools, there is still a heavy reliance on manual reconstructions completed by the examiner. Tools

such as EnCase Forensic and Autopsy do not aid the examiner during the event reconstruction process. Reconstruction takes a significant amount of input from the examiner. New automation, visualization, and abstraction methods must be employed within these tools to combat the slow, manual reconstruction processes.

1.1 Digital Forensics

Technological innovation has been growing so rapidly that technological development in digital forensics has been unable to keep up. Much of the forensics process still relies on manual searches by human investigators [6]. Additionally, a lack of automation capabilities makes the digital forensics process very time consuming. In the past, the expense and relatively small footprint of digital devices allowed for investigators to keep up with the influx of cases involving digital evidence [7]. In today's connected world, investigators are plagued by a backlog of cases and a time consuming process.

In addition to changes in general computing and the internet, the development of the Internet of Things (IoT) has changed the forensics landscape [3]. IoT devices have created new ways for criminals to conduct their business and try to remain hidden from law enforcement [3]. These new devices are usually inexpensive while still being able to complete complex tasks and store vast amounts of data. These devices include televisions, refrigerators, automobiles, Global Positioning System (GPS) devices, cell phones, and nearly any other device connected to the internet [3]. Many of these devices record data that could be a key piece of evidence necessary to prove the elements of a crime.

To keep up with this rapidly changing industry, new tools with greater automation and abstraction capabilities must be developed to deliver traces of relevant criminal activity to the investigator much faster.

The digital forensic process involves acquiring the target system image by making a forensic copy of the system storage hardware seized from the scene [8]. Examiners use the newly created copy in a digital forensic examination tool such as Autopsy. Using Autopsy, the examiner runs ingest modules that parse the data stored on the target image. Once the data is ingested, the examiner manually reviews it to look for significant evidentiary artifacts. When finished, the examiner manually reconstructs the steps the user took in order to create the artifacts. The examiner often creates a timeline of events that shows how an item or event was created and where it came from. This activity is the most time consuming aspect of the digital forensics process and the cause for an inefficient digital forensics workflow.

1.2 Hypothesis

The research hypothesis is that automated event reconstruction and timeline visualization integrated into Autopsy reduces time spent on reconstruction and improves the forensic examiner workflow. The improved workflow allows the examiner to spend more time working on decreasing case backlog.

1.3 Research Objectives

This research leverages the Property Graph Event Reconstruction (PGER) research by Schelkoph [9] and the Temporal Analysis Integration Management Application (TAIMA) research by Adderley [10]. PGER provided a method for event data abstraction and graph database storage for rapid querying [9]. TAIMA used the PGER database and abstraction rules to create a timeline visualization allowing examiners to quickly analyze and reconstruct forensic events [10].

To evaluate the hypothesis, first is the development of Autopsy Integrated Event Reconstruction (AIER). AIER integrates the ideas from Schelkoph and Adderley into

Autopsy, a common digital forensics software.

AIER automates event reconstructions and allows direct interaction between Autopsy and the abstracted timeline. AIER's improvements to the digital forensic examiner workflow are demonstrated using three use cases.

1.4 Methodology

The first component of AIER is the ingest module. The AIER ingest module utilizes the Autopsy database to extract events and temporarily stores them in Comma Separated Value (CSV) files. The ingest module also utilizes an open-source tool called Plaso to extract Windows event logs. Plaso creates text based timelines of system events and saves them to CSV files. AIER abstracts the collected events from the CSV files into five separate categories using a series of Neo4j cypher queries. The Neo4j cypher queries also load the abstracted data into a graph database for rapid searches. The examiner is only required to initiate the ingest module processing and then wait for it to complete. The ingest module incorporates the automation and abstraction portions of the AIER process.

The second part of the AIER system is the visualization and user interface. The visualization is made up of the GraphQL Application Program Interface (API) and the vis.js timeline from the React JavaScript library. The GraphQL API receives requests from the vis.js User Interface (UI) then queries the Neo4j database to retrieve the requested data. Next, the GraphQL API receives the requested data from Neo4j and passes the data to the UI for display to the examiner. The vis.js UI displays the abstracted artifacts from the Neo4j database such that the examiner can easily see the order of events within the five categories. The visualization of the events in one place eliminates the need for manual reconstructions.

There are three ways to interact with AIER timeline. If the examiner already

knows specific dates they want to view in AIER, they can select the timeline button in Autopsy, select the AIER button, then enter their dates to view the abstracted artifacts within their search. If the examiner starts their search by browsing the artifacts in the Autopsy UI, they can find other temporally related artifacts in AIER by right-clicking the artifact selecting "View Item in AIER...". While browsing artifacts in AIER an examiner may want to authenticate a specific artifact and can find more information about an artifact by double-clicking it in AIER. AIER opens a new content viewer pane containing more information about the artifact.

1.5 Results

AIER was evaluated using three different scenarios to assess its ability to improve the forensic examination workflow. Using the novel Autopsy ingest module, forensic artifacts were automatically discovered, abstracted, and made available for display in the AIER timeline. The integrated AIER timeline presented the abstracted data across five categories. The presentation of the data enabled an examiner to quickly identify key artifacts in the Autopsy user interface, display them on a timeline, and reconstruct each step a malicious actor took on a target system. Lastly, the integration of this visualization into the Autopsy software enabled the examiner to gather more detail about each reconstructed artifact with ease.

1.6 Summary

Digital forensics examiners are fighting a challenging battle against rapidly changing technology and growing data sets. State of the art tools such as Autopsy help the examiner, however these tools are still plagued with manual processes and time consuming searches that cause case backlogs. Using graph database and abstraction techniques from PGER and timeline visualization techniques from TAIMA, this re-

search seeks to automate reconstructions for the forensic digital examiner workflow to reduce data overload and case backlog.

II. Background and Literature Review

Creating a tool to improve the workflow for the forensic examiner must include a sound understanding of the technical and legal aspects of the cyber forensics process, an understanding of currently available tools, and techniques for improving cognition and pattern recognition. Enhancing forensics tools to improve cognition of forensic data is key to improving the examiner's workflow and a potential way to offset the increasing amount of data produced by cyber devices today. Additionally, automating the reconstruction phase of the investigation can aid in workflow efficiency.

The following sections discuss cyber forensics, tools currently used, and highlight the prior research efforts in support of an efficient forensics workflow. These sections also discuss the components that make up the resulting autopsy integrated event reconstruction process developed in this research.

2.1 The Forensic Process

Much like the standard procedures for any forensic investigation, cyber forensics shares many of the same investigation elements [11]. Cyber forensics follows the use of the scientific methods, collection and preservation, validation, identification, analysis and interpretation, and documentation or presentation [8]. The scientific method in the forensics process involves observing the scene, research potential forensic methods, establish a hypothesis, test the hypothesis through searches and examination, observe the collected evidence, analyze the evidentiary results and draw a conclusion, and finally present the findings through documentation [8]. Similar to this scientific approach in forensics, cyber forensics involves obtaining and imaging forensic data, receiving a forensic request, preparation and extraction of the data, identification of notable data, analysis of collected information, forensic reporting, and a case level

analysis [8]. This research focuses on the extraction of data, identification of events, and analysis of collected information using commercial and custom-built tools. Figure 1 shows the Department of Justice (DOJ) digital forensics investigation process.

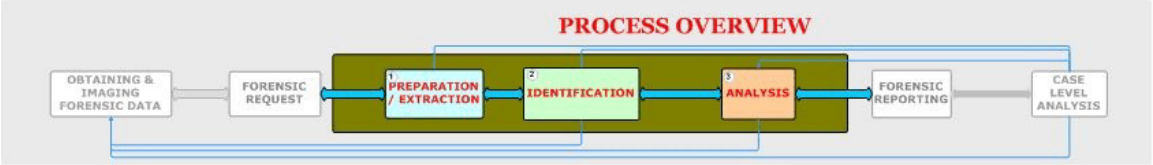

Figure 1: DOJ Investigative Process [8].

Dr. Edmond Locard, a forensic science pioneer, is credited with establishing Locard's exchange principle which states that every criminal act leaves a trace [12]. The Organization of Scientific Area Committees (OSAC) defines forensic science as the systematic and coherent study of traces to address questions of authentication, identification, classification, reconstruction, and evaluation for a legal context [13]. Locard's exchange principle applies to digital forensics as it applies to many other facets of forensic science. The OSAC questions guide digital forensic examiners during their discovery of traces left by users.

2.1.1 Authentication

Authentication is the decision process used to establish confidence in the truth of a claim [13]. This process involves evaluating a claim and deciding if that claim meets an established level of confidence [13]. In the context of digital forensics the same principles apply to the evaluation of an artifact of trace evidence from a target system. The trace is evaluated based on its integrity and provenance to establish the examiner's level of confidence that the trace artifact is authentic.

2.1.2 Identification

Identification is the process that attempts to establish confidence that identity information describes a specific trace artifact [13]. Identification attempts to apply specific defining characteristics to an artifact. The identification process is very often applied to humans, but the process can also be applied to nearly any object, physical or virtual [13]. The identification of objects in digital forensics can be the application of file names, timestamps, or other unique properties that identify that object.

2.1.3 Classification

Classification is the decision process that leads to the development of groups that similar trace artifacts belong to [13]. The process of classification is defined by two facets. One is the creation of the groups to which the artifacts belong. The groups are formed by the common identification characteristics among artifacts. The second is the process of evaluating the confidence that an artifact belongs to a specific group. In digital forensics this is an important process in order to organize like traces and enables the next process, reconstruction.

2.1.4 Reconstruction

Reconstruction is the organization of trace artifacts to find the most likely operational conditions or capabilities, patterns in time, and linkages between entities such as people, places, or objects [13]. Within the examination and analysis phase of the digital forensic process the examiner attempts to discover and reconstruct events in the order in which they occurred [11]. Reconstruction can account for a significant portion of the examiner's time during the analysis phase. The use of a timeline during the analysis phase is extremely helpful and allows the examiner to visualize the steps taken by a user of the target system.

2.1.5 Evaluation

The process of evaluation is very broad and sometimes controversial [13]. In a courtroom, the jury evaluates the truthfulness of the claims made by prosecution and defense. The evaluation of forensic evidence is the process by which the strength of the evidence is determined [13]. Evaluation occurs throughout the forensics process to make decisions about the trace.

2.2 Forensic Tools

Commercial and open-source tools are available for processing digital investigations. Each tool has its own way of analyzing data and displaying it to the investigator. Each of these tools offers varying levels of detail and data granularity, and most have varying methods of presenting data in a timeline. Some of the most common commercial forensics tools include:

- Cellebrite's Universal Forensic Extraction Device (UFED)
- EnCase Forensic by OpenText
- Forensic Tool Kit (FTK) by AccessData
- i2 Analysis Notebook by International Business Machines (IBM)
- Magnet Forensics

Two of the most popularly used commercial forensics tools are Cellebrite's UFED and EnCase Forensic by OpenText. Cellebrite's UFED is a widely used commercial forensic tool that enables investigators to access and extract data from a wide variety of modern devices [14]. UFED incorporates a substantial amount of proprietary analysis and abstraction software [14]. UFED, however, can be costly, putting it out of reach of many agencies. EnCase Forensics is another popular tool with enhanced

processes to increase efficiency and improve workflows [15]. Much like UFED, EnCase can be cost prohibitive to smaller agencies.

In addition to these commercial products, there are multiple open source and free forensic tools. Open-source software reduces costs and provides the level of flexibility required to implement custom tools to suit specific forensic needs. These open-source tools often provide similar capabilities as compared to the commercial products but allow for customization. Some of the most common open-source or free digital forensics tools include:

- Autopsy / Sleuth Kit
- SANS Investigative Forensic Toolkit (SIFT)
- Digital Evidence and Forensic Toolkit (DEFT)
- Plaso

Autopsy provides the most robust feature set compared to other open-source tools and is the closest open-source equivalent to other commercial tools. These forensic tools are very helpful at extracting and displaying information to the investigator. However, the investigator must still review and interpret the data to gain an understanding of what happened on the device in question. Using computer graphics and visual displays enables investigators to interpret data much faster. Autopsy Integrated Event Reconstruction (AIER) utilizes Autopsy and The Sleuth Kit's open-source framework to help the digital forensic investigator.

2.3 The Sleuth Kit

The Sleuth Kit (TSK) is a collection of command line tools that enables developers to build automated end-to-end digital forensics applications [16]. The TSK framework

contains an image database, pipelines and plug-in modules, blackboard, services, and a three phase analysis process [17]. All of the data produced by the TSK framework is stored in the image database as shown in Figure 2. The image database is used in Section 3.2.2 during AIER ingest module processing. In addition to its use in Autopsy, TSK tools are used in other open-source and commercial programs [18].

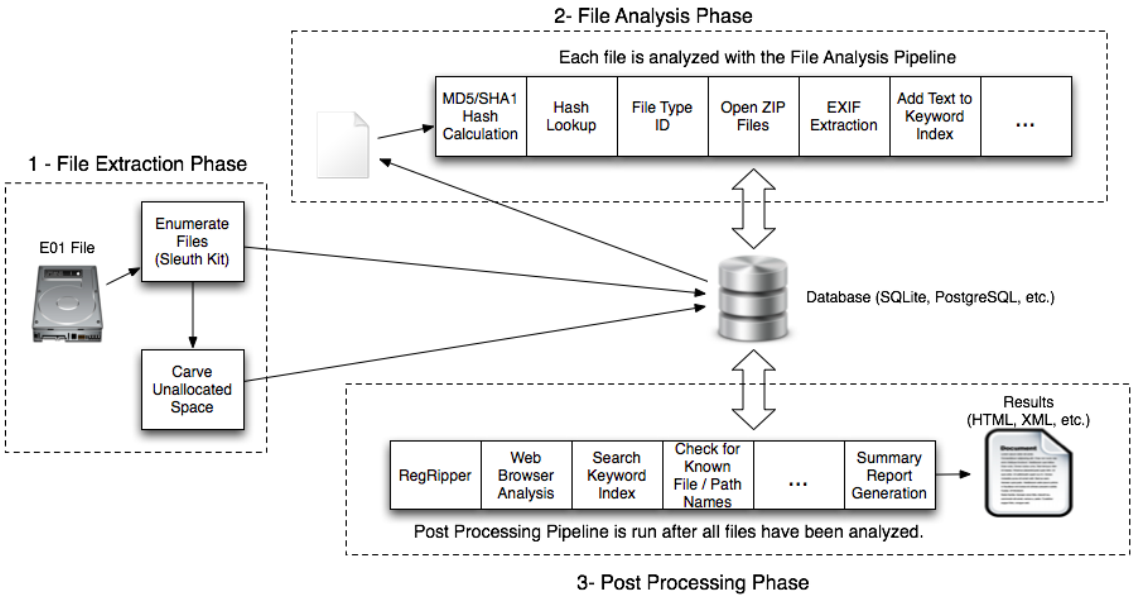

Figure 2: The Sleuth Kit Framework [17].

2.4 Autopsy

Autopsy is an open-source, Java based, Graphic User Interface (GUI) that allows a user to examine hard drives or mobile devices for the purposes of evidence recovery [19]. Autopsy uses various modules to process forensic data for analysis and review [16]. Some of these modules utilize TSKs command line tools to analyze disk images and recover files [16]. Autopsy's most popular features include multi-user cases, timeline analysis, web artifacts, registry analysis, and email analysis [20]. Autopsy's timeline feature, however, does not incorporate abstraction or automation to make it

easy for an examiner to reconstruct events. The Autopsy timeline shows data in two modes [21]. The first view mode is a bar graph that shows the frequency of events along a date line as shown in Figure 3 [21].

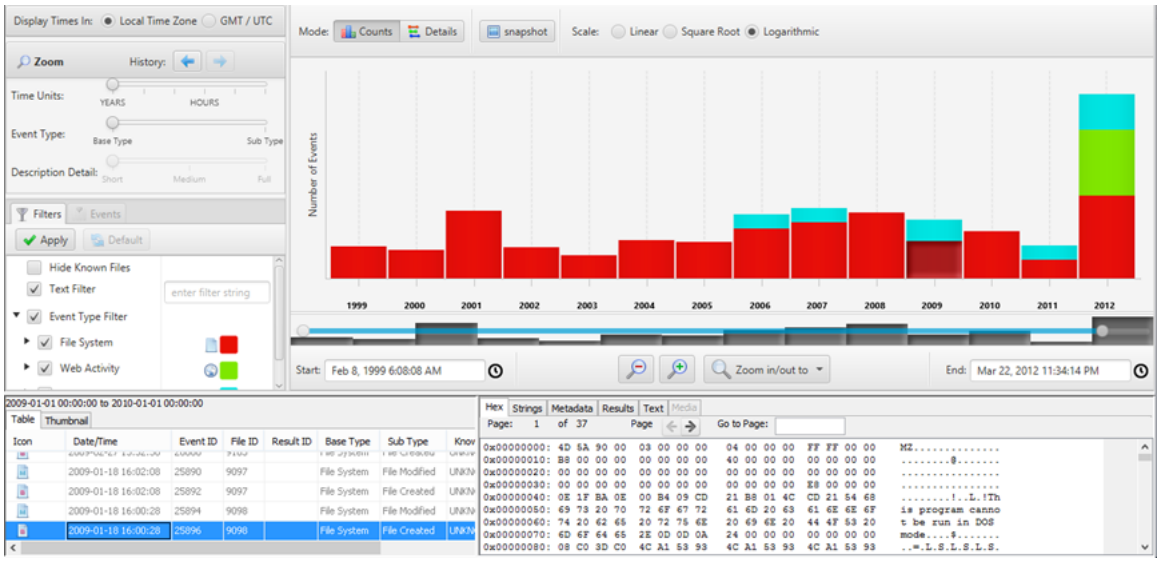

Figure 3: Autopsy Timeline [21].

The second view mode shows event details about specific occurrences from the extracted data as shown in Figure 4 [21]. The second view mode is still cluttered and overloaded with all of the extracted data from the source image. Autopsy offers no abstraction of the ingested data and simply displays all data to the examiner. Additionally, the Autopsy design suggests the user still manually search for data, tag relevant events, and then open the timeline to see when they occurred. This research aims improve the examiner's workflow by using the AIER timeline to find the relevant events instead of the other way around.

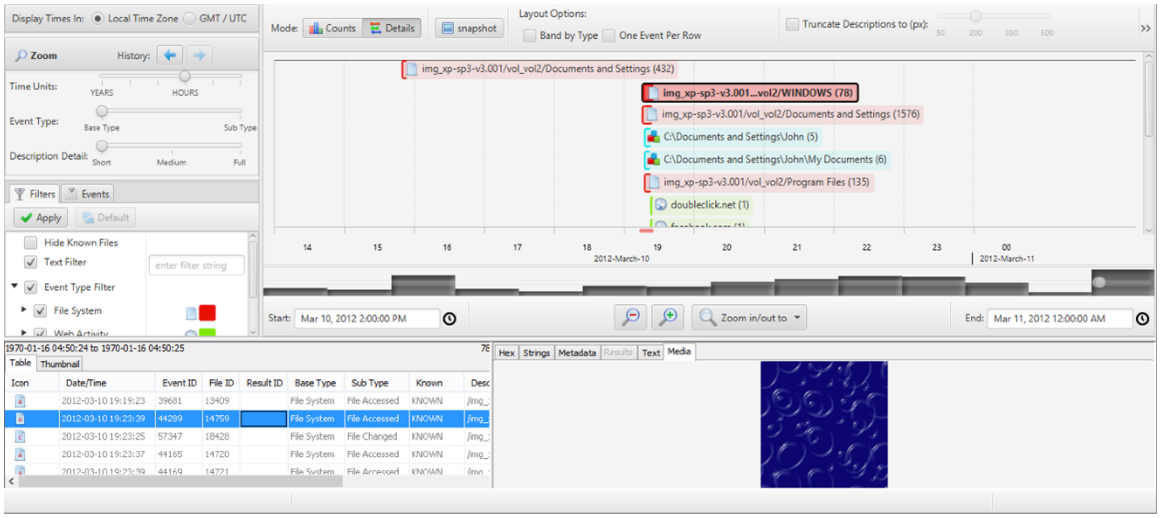

Figure 4: Autopsy Timeline Event Details [21].

2.4.1 Autopsy Modules

Autopsy's design is modular and uses a series of modules to parse target images and create a database of significant artifacts [16]. The modular design allows developers to create new modules to extend Autopsy in new ways. There are two types of modules used in Autopsy [22]. Both module types are for parsing artifacts and ingesting data into the Autopsy database. The modules are run on a new data source when it is added to a case. The two types of ingest modules include, file and data source modules. File modules are used on every file within an image. These modules are used to inspect individual files and gather information about those files. Data source modules are used to gather information about entire data sources, usually system images. AIER includes a novel data source module called the AIER ingest module.

2.5 Information Visualization

Visualizing information in a graphical format can significantly improve cognition [23]. The visual representation of information allows users to process and recall information much faster [23]. Visualization also allows users to easily detect patterns and infer data from the visualizations [23]. Using visualization in the identification phase of the forensic process can increase the speed at which investigators find digital evidence.

The visualization of digital forensic artifacts helps investigators see events, their chronological order, and make determinations based on the patterns of behavior [24]. However, investigators can become overwhelmed with an overload of forensic events produced by the various forensic tools. Chapter I highlighted the problems with the overload of data provided by today's modern devices. One method to help alleviate data overload problem is to 1) abstract lower-level event data into higher-level events and 2) reduce the number of significant artifacts that are visually displayed to the user.

2.6 Data Abstraction

Visualizing the data produced by the forensic tools improves the efficiency of the investigator's workflow. However, visualization solves only part of the problems associated with digital data overload. To further help the workflow problem we must also reduce the amount of data presented to the investigator [25]. Data reduction must be done carefully to ensure critical artifacts are not omitted from the investigator's review [25]. The use of filters and the abstraction of forensic events can significantly reduce the data overload. Data abstraction can be used to find patterns and groups of data that indicate a higher-level event [25]. Investigators do not often need to see every detail about every event produced by an item of digital evidence.

Windows based computers generate many thousands of events which are recorded. The creation of events makes it difficult for investigators to see the ones that have significant evidentiary value. Data abstraction combines numerous low-level events into a high-level event [25]. The creation of a high-level event indicates to the investigator that something significant likely occurred warranting further analysis.

2.7 Timelines

Criminal investigations often rely on timelines to create a chronological order of events [26]. Humans naturally want to order events in order to understand the process or order in which they were created. Digital forensic investigations are no different in this respect. Additionally, a timeline assists the investigator in the reconstruction phase of an investigation [26]. Timelines can be presented to the examiner in multiple ways such as text-based timelines and visual timelines. Text-based timelines can lead to an information overload problem and can make cognition difficult. Plaso, for example, creates text-based timelines from system images and outputs the data in the form of a Comma Separated Value (CSV) file. The CSV file is normally processed further into a more useful format. AIER uses Plaso to gather system event log data, but processes the CSV file into a visual timeline. Visual timelines make it easier for examiners to identify patterns and relationships among trace artifacts. Autopsy's histogram style timeline and Temporal Analysis Integration Management Application (TAIMA)'s timeline are examples of a visual timeline.

2.7.1 Property Graph Event Reconstruction

Property Graph Event Reconstruction (PGER) is a tool developed to reduce the amount of data the investigator has to search through and reduces the amount of computing power required by the forensic computer [9]. PGER processes events

from a disk image using two applications. The first application is called Temporal Event Abstraction and Reconstruction (TEAR). TEAR identifies high-level events using algorithms and pattern matching [27]. These events are stored in CSV files for further processing via a custom python script. The second application called Plaso is a python based tool that extracts timestamps from system images [28]. Plaso consists of two parts, log2timeline and psort [28]. Log2timeline extracts the timestamps from the disk image and stores them in a database [28]. Psort parses the newly created database and loads that data into a another database called Elasticsearch for further processing. Next, PGER further processes the image for storage in a graph database called Neo4j that allows for rapid searching [9]. PGER processes the disk image using the ELK Stack [9]. The ELK stack consists of the Elasticsearch, Logstash, and Kibana [29]. Elasticsearch is a JavaScript Object Notation (JSON) based database that allows for fast searching and data delivery [29]. Logstash is a tool that processes data from the ElasticDB, filters it, and sends it to another source, and in Schelkoph's research, Neo4j. Lastly, Kibana is the GUI for the ElasticDB that allows users to view and manage elastic data [29]. PGER uses these tools to import, filter, and export the collected image data to Neo4j [9].

2.7.2 Temporal Analysis Integration Management Application

TAIMA uses the GRANDstack suite of applications to query the data abstracted by PGER and present it to the user in a simplified timeline [10]. GRANDstack is made up of GraphQL, React, Apollo, and Neo4j Database [30]. GraphQL is an Application Program Interface (API) responsible for querying the Neo4j database and passing the information along to the User Interface (UI) for presentation [31]. Neo4j uses a native query language called Cypher, and is similar to Structured Query Language (SQL) but for graph databases [32]. React is a JavaScript library for building reusable user

interfaces [33]. Within this research, vis.js from the React library is used to present the visual timeline to the end user. Apollo interacts with GraphQL to handle the queries that are sent from the timeline UI [10]. Finally, Neo4j is a native graph database that shows object relationships in a graph form [34]. The use of a graph database configuration allows for rapid searching and data retrieval.

TAIMA utilizes these applications by first showing a blank timeline and waits for a date input from the user. The user enters a date and the timeline sends the date query to the GraphQL API via the Apollo client [10]. The GraphQL API creates a Cypher query that is readable by the Neo4j database [10]. The cypher query is received by the Neo4j database and Neo4j finds the appropriate data [10]. The data is sent back to GraphQL and is then sent to the timeline UI via the Apollo client [10]. Figure 5, is a visual representation of the process for a movie database.

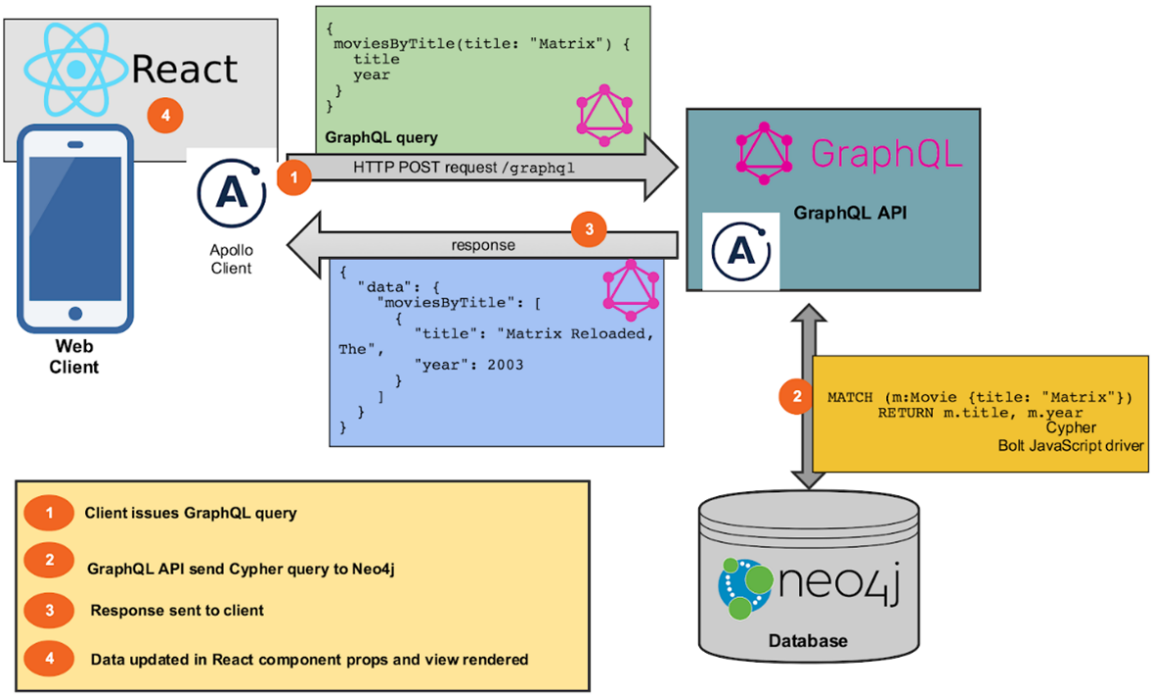

Figure 5: GRANDStack Framework [30].

2.8 Summary

The digital forensic and traditional forensic process are nearly the same. Both utilize the scientific method, collection and preservation techniques, and include the formation and tests of hypotheses. Digital forensic investigations also include the extraction of data, identification of artifacts, and analysis of the artifacts [13]. The digital forensic examiner knows that every action on a target system leaves a trace behind and the collection of those traces involves their authentication, identification, classification, reconstruction, and evaluation. The reconstruction of artifacts is a time consuming task for the examiner, but tools such as Autopsy and timeline visualizations help the examiner workflow to overcome information overload and the burden of large data sets.

III. Autopsy Integrated Event Reconstruction

The goal of this research is to enhance the forensic examiner workflow through automation, visualization, and integration into the popular forensic software, Autopsy. Currently, the examiner workflow is plagued with string searches and manual event reconstruction. Manual reconstruction takes a considerable amount of time and attention from the examiner. Autopsy Integrated Event Reconstruction (AIER) lessens the burden on the examiner by integrating Autopsy, Plaso or super timeline all the things, Neo4j graph database, GraphQL Application Program Interface (API), a JavaScript based user interface, and a custom Autopsy ingest module called AIER ingest module. AIER abstracts data extracted from a target image and produces a visual timeline as an alternative to the Autopsy timeline. AIER's custom timeline provides the examiner with information from five categories including program installations, power events, program executions, file downloads, and web history.

The following sections overview each component, how the component was created, and its purpose in the creation of the final timeline. AIER requires a Windows based system image as used in the prior research by Schelkoph [9] and Adderley [10]. AIER is based on some of the ideas and techniques proposed by Schelkoph and Adderly, but integrates these ideas into Autopsy, abstracts the data, and provides a visual timeline in the style of Adderley's work.

3.1 Autopsy Ingest

The first step in the AIER process is the standard Autopsy data ingest. The ingest modules extract events from the provided image and stores them in a relational database. Later, the Autopsy database is utilized to build the graph database required to store the abstracted artifacts for use on the timeline.

Figure 6 illustrates the ingest process for a Windows system image in Autopsy. Starting with the standard Autopsy ingest modules, the image is processed and a database of artifacts found during the processing is created for the case. Next, the examiner initiates a new image ingest but only selects the AIER ingest module for processing. The AIER ingest module uses data from the Autopsy database and the Plaso tool to create an abstracted timeline. Lastly, the examiner opens the AIER timeline for exploration with a new start and end time. The entered timestamps are submitted through an API called GraphQL that queries the Neo4j database and returns the data to the timeline User Interface (UI) for view by the examiner. The following sections describe each step of the ingest process in more detail.

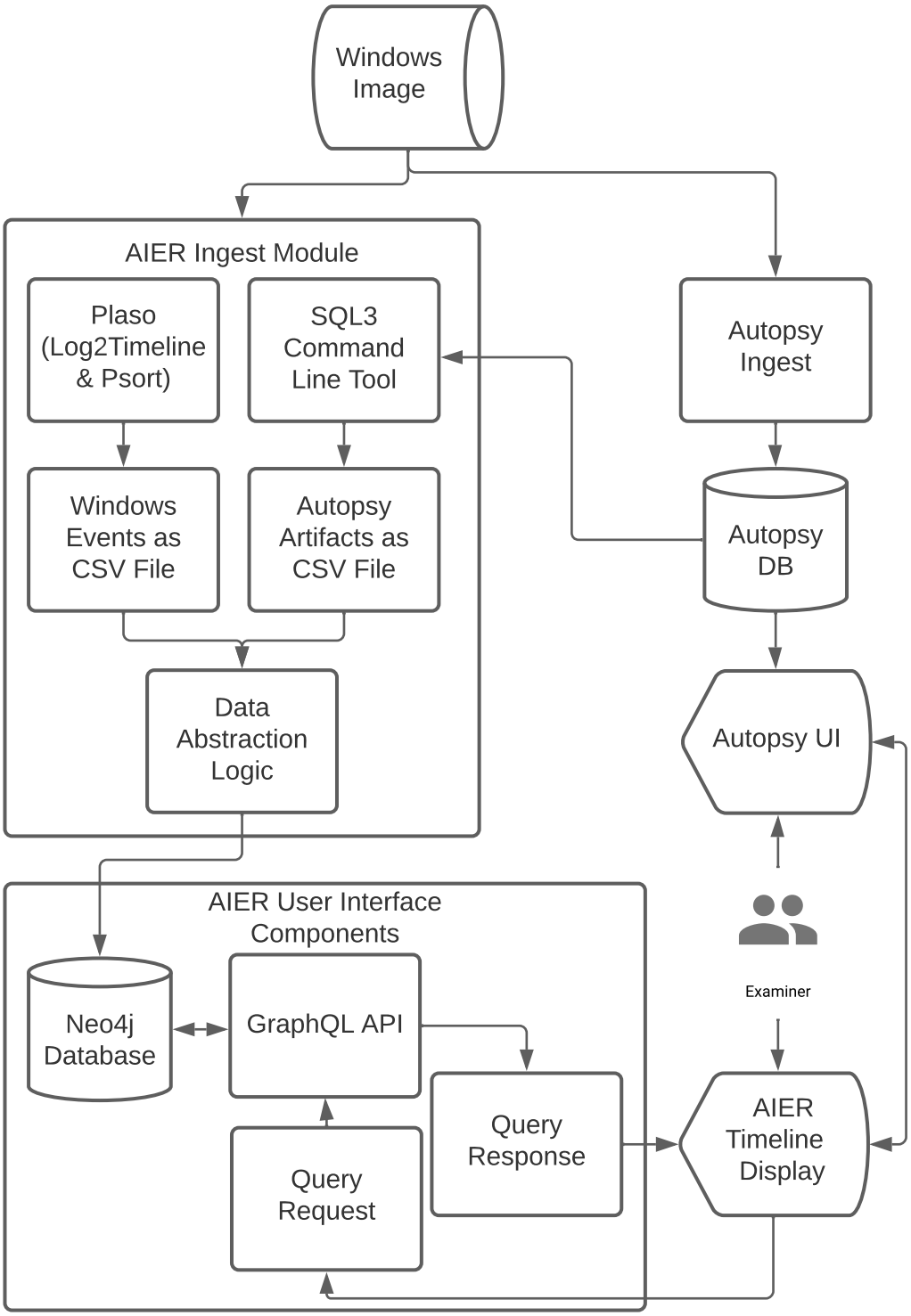

Figure 6: AIER Integration Process.

3.1.1 Autopsy Database

The Autopsy database is made up of many tables, but the AIER ingest module utilizes the blackboard_artifact_types, blackboard_artifacts, and blackboard_attributes tables created by the Autopsy ingest modules. The blackboard artifact types table describes the type of data artifact listed within the blackboard artifacts table and assigns each type a respective Identification (ID). The blackboard artifacts table lists all the artifacts generated by the ingest modules. These artifacts have a respective artifact ID, object ID, artifact object ID, and an artifact type ID from the blackboard artifact types table. The blackboard attributes table contains the core data generated by the ingest modules and contains the artifact IDs, artifact types, the data source, timestamps, and other valuable data such as the artifact's file path.

Figure 7 shows an example of the data stored in the Autopsy database tables. The artifact ID is automatically generated by the ingest module. The artifact type ID 32 indicates that the artifact is showing a program that has run on the target Windows image. The source column indicates what part of the ingest module found the artifact. The value text column provides additional information about the artifact, and in this case the value text column shows the file name of the run program. Lastly, the value int64 column provides the timestamp indicating precisely when the program was run. AIER utilizes these fields to abstract the data and create events for visualization within the abstracted timeline.

artifact_id ▾1	artifact_type_id	source	:ontex	attribute_type_id	ue_ty	ue_by	value_text	ue_int	value_int64
Filter	32	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
-9223372036854774786	32	Windows Prefetch Analyzer		4	0	NULL	123WASP_SETUP.EXE	NULL	NULL
-9223372036854774786	32	Windows Prefetch Analyzer		8	0	NULL		NULL	NULL
-9223372036854774786	32	Windows Prefetch Analyzer		2	5	NULL	NULL	NULL	1093014774
-9223372036854774786	32	Windows Prefetch Analyzer		92	1	NULL	NULL	1	NULL
-9223372036854774786	32	Windows Prefetch Analyzer		66	0	NULL	Prefetch File	NULL	NULL

Figure 7: Blackboard Attributes Table.

3.2 AIER Ingest Module

The core of the automation, abstraction, and data transformation occurs in the AIER ingest module. AIER is a custom Autopsy module that utilizes open-source forensics tools to extract, parse, transform, organize, and import data to a Neo4j graph database. The Neo4j database is used to host the abstracted timeline events for the rapid retrieval and display to the investigator on the AIER integrated timeline. Figure 6 shows a hierarchical view of the ingest process.

3.2.1 Plaso

The Autopsy database generated by the standard Autopsy ingest modules provides the data required to populate four of the five categories shown on the AIER timeline. Additional information from the Windows target image is required to display results for the power events category. The open-source tool Plaso also known as super timeline all the things retrieves the data that is required to determine power events. Plaso uses two tools, Log2Timeline and PSort, to generate a Comma Separated Value (CSV) file of selected events.

3.2.1.1 Log2Timeline and PSort

Log2Timeline parses the Windows image and collects selected events from Windows logs and other various sources. AIER focuses on the Windows Event Log parser built into Log2Timeline. The event log parser collects events and timestamps from the Windows event logs.

After Log2Timeline parses the Windows image, PSort collects the data output, organizes the data, applies headings, and outputs the data to a CSV file. The CSV file contains the low-level event data used to abstract information into higher-level data. The higher-level data is used determine major events that are otherwise not

listed in the Windows event logs. The abstracted data is used to indicate a system power event on the timeline and is further discussed in Section 3.2.3.2. Figure 8 shows and example of the output provided by PSort.

```
datetime,timestamp_desc,source,source_long,message,parser,display_name,tag
2019-01-28T19:09:06.546508+00:00,Content Modification Time,EVT,WinEVTX,[12 / 0x000c]
2019-01-28T19:09:06.546565+00:00,Content Modification Time,EVT,WinEVTX,[153 / 0x0099]
2019-01-28T19:09:06.546618+00:00,Content Modification Time,EVT,WinEVTX,[157 / 0x009d]
2019-01-28T19:09:06.546619+00:00,Content Modification Time,EVT,WinEVTX,[157 / 0x009d]
2019-01-28T19:09:06.546619+00:00,Content Modification Time,EVT,WinEVTX,[157 / 0x009d]
```

Figure 8: Plaso PSort CSV Output.

3.2.2 Autopsy Database Extraction

Data represented in the Autopsy database is created from the standard Autopsy ingest modules. The Autopsy database information must be extracted from the database file and organized for the abstraction phase as discussed in Section 3.2.3.2. AIER uses an sqlite3 command line tool to extract the data from the Autopsy database. AIER extracts information from the blackboard_artifact_types, blackboard_artifacts, and blackboard_attributes tables and writes them to a CSV file for later abstraction and import into Neo4j. Figure 7 shows the CSV output of the sqlite3 command line tool.

```
artifact_id,artifact_type_id,source,context,attribute_type_id,value_type,value_byte,v
-9223372036854775807,10001,"Recycle Bin Analyzer","",8,0,,C:\\Users\\Steve\\Pictures\
-9223372036854775807,10001,"Recycle Bin Analyzer","",133,5,,,,1548989321,
-9223372036854775807,10001,"Recycle Bin Analyzer","",14,0,,"",,,
```

Figure 9: Structured Query Language (SQL) Command Line Tool Output.

3.2.3 Neo4j

The core of the AIER timeline integration and ingest module is the Neo4j native graph database. Neo4j allows for rapid query speeds and data storage after abstraction. The CSV files generated in the previous ingest steps provide the data required

for abstraction. The following section describe the processes required to abstract and organize the data for use by GraphQL.

3.2.3.1 CSV Import

The first step in the data transformation process imports the CSV data from the prior ingest steps. Importing uses the cypher shell command line tool. Cypher is the query language used by Neo4j for all of its data operations [32]. The initial import populates the database with the Autopsy artifacts and no relationships as shown in Figure 10. Section 3.2.3.2 discusses the import of the blackboard attributes CSV output and the Plaso CSV output as well as abstraction of both data sets into higher-level events.

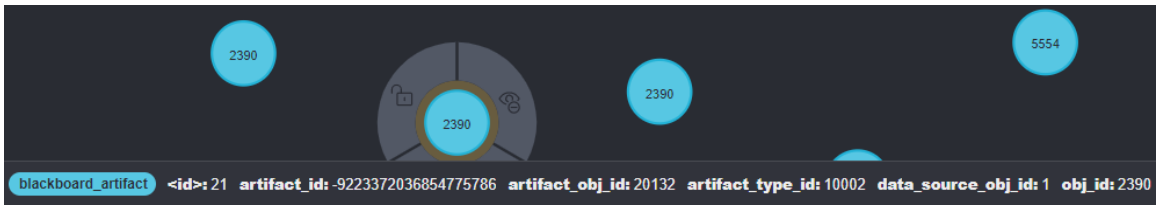

Figure 10: Neo4j CSV Import.

3.2.3.2 Data Abstraction

With the CSV data imported into the Neo4j database, the AIER ingest module abstracts and organizes the imported data into higher-level events for use by the AIER timeline. The data is transformed using a series of cypher shell commands that first create relationships between the artifacts and their types. Next, the remaining data is imported, assigned properties for each node, and those properties are used for abstraction and subsequently assigned a timeline group value. Finally, all of the abstracted data is combined into a single Abstraction node. The abstraction nodes are all connected to a time tree based on their timestamp values for rapid querying.

The following sections describe each data abstraction in more detail.

Artifact and Type Relationships

The abstraction begins by creating a relationship between the artifacts shown in Figure 10 and the artifact types that describe those artifacts. Figure 11 shows an example graphical representation of the relationships between the artifact type TSK_PROG_RUN (red), that indicates a program has run, and the artifacts (blue) produced by Autopsy that provide the specific artifact IDs. These artifact IDs are the link to more detailed information about that artifact in later abstraction steps.

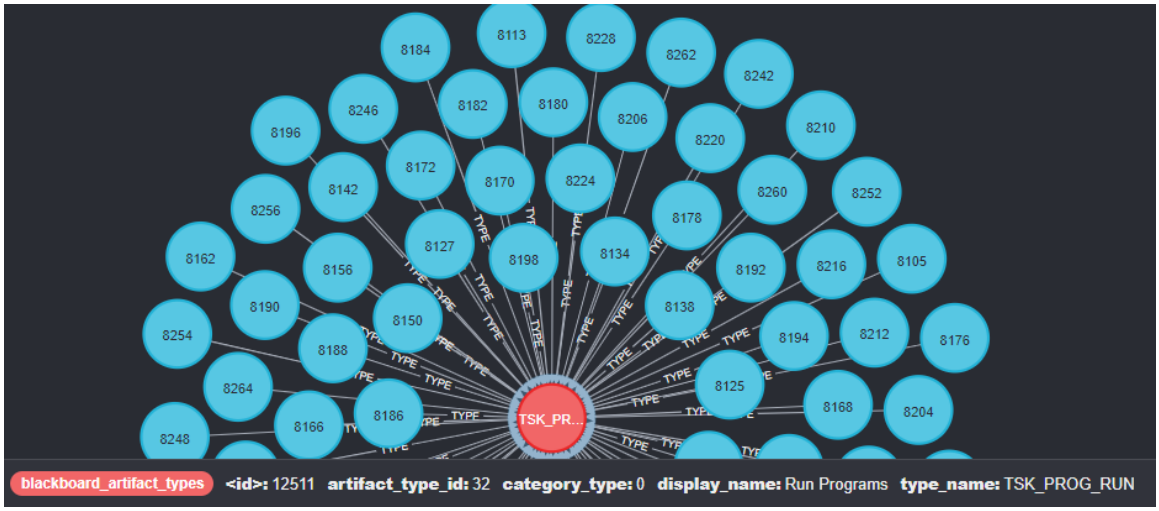

Figure 11: Artifacts Associated with Types.

Program Installation Abstraction

To import and abstract data indicating the installation of a program on the target Windows image we first import the blackboard attributes CSV data and select the specific headers that are useful to the abstraction. The artifact IDs from the previous abstraction step are used to create a relationship to the detailed blackboard attributes CSV data. Figure 12 shows an example of a single artifact (blue), its associated artifact type (red), and the newly created attributes that describe that artifact.

These newly created attributes contain the data required to create the final program installation abstraction node.

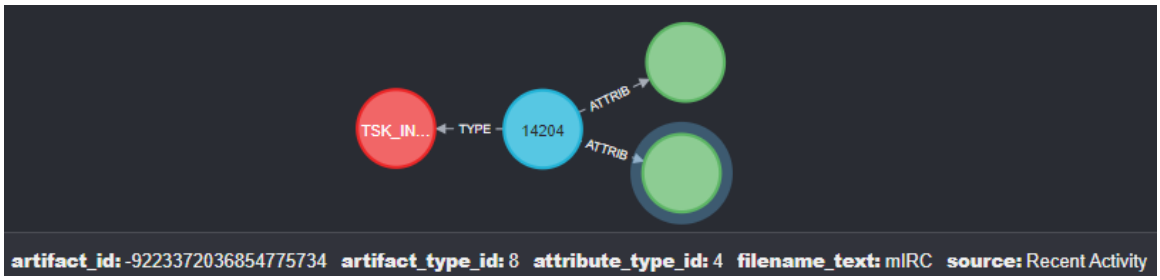

Figure 12: Attributes Related to Artifacts.

With all of the data required for the abstraction now imported and organized, the final abstraction node is created using the cypher query shown below.

```
MATCH (a:blackboard_artifact)
WHERE a.artifact_type_id = "8"
MATCH (a)-[b:ATTRIB]->(c:blackboard_attributes)
WHERE c.attribute_type_id = "4"
WITH *, c.filename_text AS Event
MATCH (a)-[d:ATTRIB]->(e:blackboard_attributes)
WHERE e.attribute_type_id = "2"
WITH *, e.timestamp AS Timestamp
MATCH (a)-[f:ATTRIB]->(g:blackboard_attributes)
WHERE g.attribute_type_id = "4"
WITH *, g.source AS Trace
CREATE (Abstraction:Abstraction{artifact_id:a.artifact_id,
    Event:Event,Trace:("Source: ")+Trace,Description:"
   Installation operation completed successfully",
   Timestamp:datetime({epochSeconds:(toInteger(Timestamp)
   -18000)}).epochMillis,Group:"1"})
```

```
WITH Abstraction
CALL ga.timetree.events.attach({node: Abstraction, time:
   Abstraction.Timestamp, relationshipType: "AT_TIME",
   create: true, resolution: "Second"})
YIELD node RETURN node;
MATCH (a:Abstraction),(b:blackboard_artifact)
WHERE a.artifact_id = b.artifact_id
CREATE (b)-[:ABSTRACTION_LINK]->(a);
```

The above cypher query collects the artifacts by their ID number and their attribute types then assigns them to a property in a new abstraction node. The abstraction node is created with the collected properties and assigned a group number, then is connected to the time tree and the original artifact that composed the abstraction. The group number for program installations is 1. The time tree is created by a Neo4j plugin that creates a tree of connected nodes that represent each part of the associated timestamp. The time tree creates a node for the year, month, day, hour, minute and second. The abstraction nodes have a relationship with the seconds node. The creation of these relationships maintains the integrity of the graph database allowing for rapid queries. Figure 13 shows the newly created abstraction node connected to its associated time tree. The abstraction node (orange) example shown here indicates that "mIRC" was installed successfully. The time tree indicates the installation occurred on 20 August 2004 at 14:10:04. Additionally, the timestamp property of the abstracted node indicates in milliseconds the amount of time since the UNIX epoch on January 1, 1970.

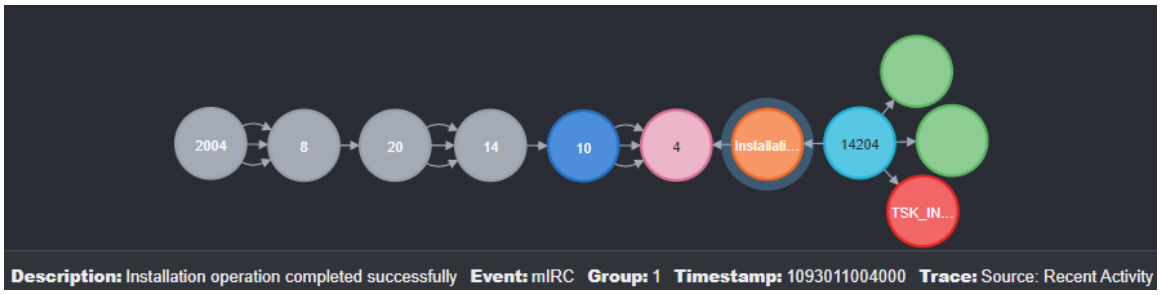

Figure 13: Abstracted Node and Time Tree.

Power Event Abstraction

Power events provide additional context to the examiner and help validate items within the timeline. Power events require a slightly different strategy than other events. There are a few indications within the windows event logs that typically only occur during system startup. The Windows event log data provided by the Plaso output helps determine system power events. Specifically, Windows event ID 6005 indicates that the event log service was started. Event ID 6005 typically only happens at system startup and is a reasonable indicator that the target system just booted. Additionally, the event log ID 6006 indicates the event log service stopped. Event ID 6006 generally only happens at system shutdown and is a reasonable indicator that the system was turned off.

The power event abstraction starts by importing the data from the PSort CSV output and storing it in a new node called plaso_attributes. The plaso_attributes node contains the data necessary to create the final abstraction node similar to the one created for program installations. Figure 14 shows an example of a plaso_attributes node. The following example indicates the system event log was started via the event ID 6005.

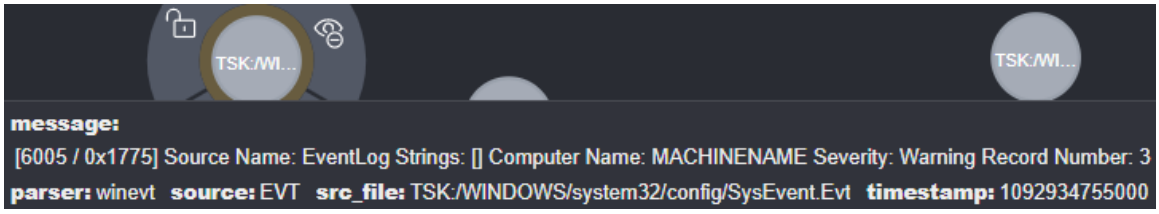

Figure 14: Plaso Attributes Node.

The final abstraction node is created is created using the following query. The properties of the plaso_attributes node are matched and sorted based on the event log ID 6005 or 6006. The collected data is compiled into the abstraction node with the group ID 2, then is assigned a relationship back to the original plaso_attributes node and to the time tree.

```
MATCH (a:plaso_attributes)
WHERE a.source = "EVT" AND a.message STARTS WITH "[6005 /"
WITH a, a.timestamp AS Timestamp, "System Startup<br>Event
    Log Service Started" AS Event, a.src_file AS
   Description, a.parser AS Trace
MERGE (Abstraction:Abstraction{Event:Event, Trace:Trace,
   Description:Description, Timestamp:Timestamp, Group
   :"2"})
WITH Abstraction
CALL ga.timetree.events.attach({node: Abstraction, time:
   Abstraction.Timestamp, relationshipType: "AT_TIME",
   create: true, resolution: "Second"})
YIELD node RETURN node;
MATCH (b:plaso_attributes)
WHERE b.source = "EVT" AND b.message STARTS WITH "[6006 /"
WITH b, b.timestamp AS Timestamp, "System Shutdown<br>
```

```
    Event Log Service Stopped" AS Event, b.src_file AS
    Description, b.parser AS Trace

MERGE (Abstraction:Abstraction{Event:Event, Trace:Trace,
    Description:Description, Timestamp:Timestamp, Group
    :"2"})
WITH Abstraction
CALL ga.timetree.events.attach({node: Abstraction, time:
    Abstraction.Timestamp, relationshipType: "AT_TIME",
    create: true, resolution: "Second"})
YIELD node RETURN node;
```

Program Execution Abstraction

It is important for the forensic examiner to know what programs were executed on a target system. Program execution abstractions show the origin of an exploit or other critical software information. The program execution abstraction is similar to the program installation abstraction. The abstraction imports the blackboard attributes data but only where the artifact type ID is 32. Figure 15 shows the attribute nodes (green) created after the import of the blackboard attributes data. The selected attribute node shows some, but not all of the information needed to create the abstraction node. The sample selected node shows its properties at the bottom of Figure 15. The filepath_text property in following example shows that the program Cain, a popular password cracking tool, was run. The attribute nodes are related to the original blackboard artifact node (blue) with the label "ATTRIB". The relationships and nodes are used to create the final abstraction node.

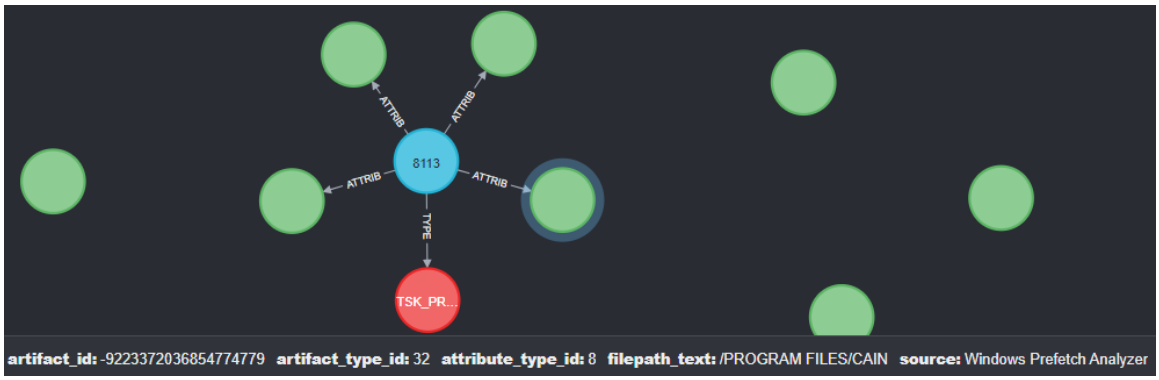

Figure 15: Program Execution Attributes.

Using the artifact type ID to indicate the type of artifact to be collected, and using the attribute type IDs all of the data required to create the final abstraction node is gathered. The query below shows how the data is matched and compiled to form the final abstraction node with the group ID 3.

```
MATCH (a:blackboard_artifact)
WHERE a.artifact_type_id = "32"
MATCH (a)-[b:ATTRIB]->(c:blackboard_attributes)
WHERE c.attribute_type_id = "4"
WITH *, c.filename_text AS Event
MATCH (a)-[d:ATTRIB]->(e:blackboard_attributes)
WHERE e.attribute_type_id = "66"
WITH *, e.trace_text AS Trace
MATCH (a)-[f:ATTRIB]->(g:blackboard_attributes)
WHERE g.attribute_type_id = "8"
WITH *, g.filepath_text AS Description
MATCH (a)-[h:ATTRIB]->(i:blackboard_attributes)
WHERE i.attribute_type_id = "2"
WITH *, i.timestamp AS Timestamp
```

```
CREATE (Abstraction:Abstraction{artifact_id:a.artifact_id,
    Event:Event,Trace:("Trace: ")+Trace,Description:("
   Filepath: ")+Description,Timestamp:datetime({
   epochSeconds:toInteger(Timestamp)}).epochMillis,Group
   :"3"})
WITH Abstraction
CALL ga.timetree.events.attach({node: Abstraction, time:
   Abstraction.Timestamp, relationshipType: "AT_TIME",
   create: true, resolution: "Second"})
YIELD node RETURN node;

MATCH (a:Abstraction),(b:blackboard_artifact)
WHERE a.artifact_id = b.artifact_id
CREATE (b)-[:ABSTRACTION_LINK]->(a);
```

With the final abstraction node created and the group ID assigned it is then connected to the time tree along side the other abstractions. Figure 16 shows the final abstraction, with its properties, and the connection to the time tree. The final abstraction indicates that "CAIN.EXE" was run at 15:33:03 on 27 August 2004.

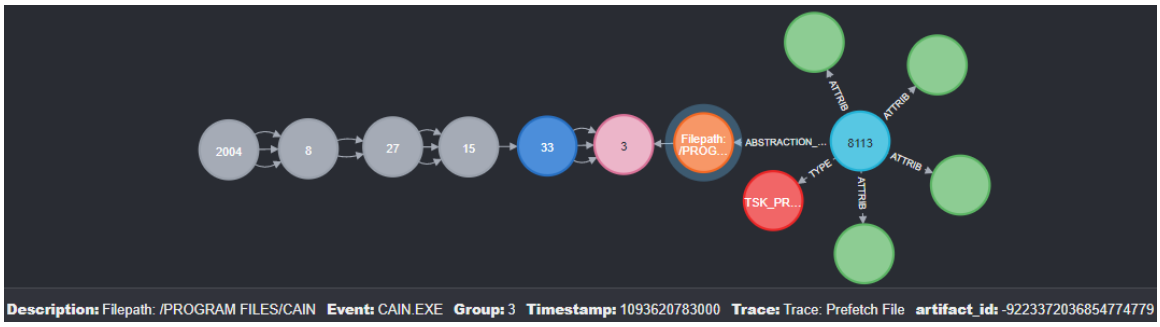

Figure 16: Program Execution Abstraction.

File Downloads Abstraction

In today's internet based world, web download information is critical for determining where a file or executable came from. The file downloads group shows the examiner the origin of items introduced to the target system. The file downloads abstraction is created by importing timestamps, browser data and Uniform Resource Locator (URL) data from the blackboard attributes nodes. These blackboard attributes are created by importing the required properties only where the artifact type ID is 5. Artifact type ID indicates that this artifact is from the file downloads category. Figure 17 shows the artifact node (blue) and the associated attributes (green) of that artifact. The selected attribute indicates that "Image Stenography 1.5.2 Setup.exe" was downloaded. The selected attribute is only one of the many properties required to create the final abstraction node. All the attribute data is collected and used to create the final abstraction node.

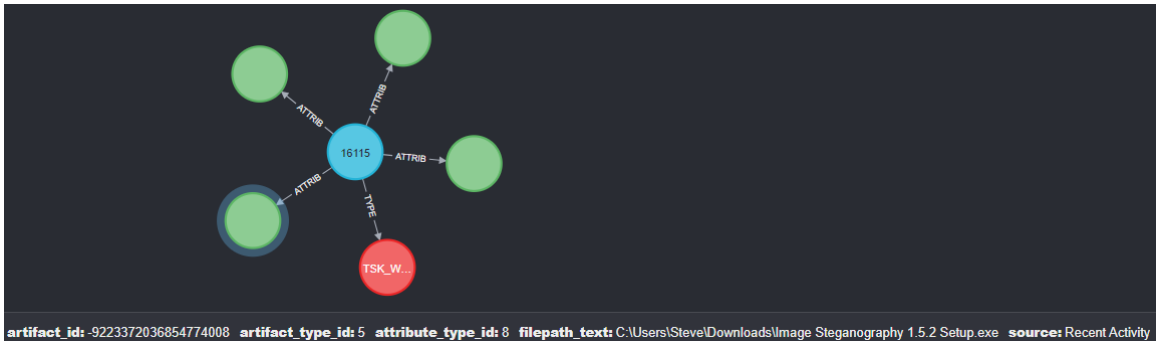

Figure 17: File Downloads Attributes.

The query below shows how the attribute data is abstracted to form the final abstraction node. Once created, the abstraction node provides the original artifact ID, event type, URL from which the file was downloaded, browser used to download the file, timestamp, and the group ID 4.

```
MATCH (a:blackboard_artifact)
```

```
WHERE a.artifact_type_id = "5"
MATCH (a)-[b:ATTRIB]->(c:blackboard_attributes)
WHERE c.attribute_type_id = "8"
WITH *, c.filepath_text AS Event
MATCH (a)-[d:ATTRIB]->(e:blackboard_attributes)
WHERE e.attribute_type_id = "1"
WITH *, e.url_text AS Trace
MATCH (a)-[f:ATTRIB]->(g:blackboard_attributes)
WHERE g.attribute_type_id = "4"
WITH *, g.program_name_text AS Description
MATCH (a)-[h:ATTRIB]->(i:blackboard_attributes)
WHERE i.attribute_type_id = "33"
WITH *, i.timestamp AS Timestamp
MERGE (Abstraction:Abstraction{artifact_id:a.artifact_id,
   Event:Event,Trace:("Source: ")+Trace,Description:("
   Browser: ")+Description,Timestamp:datetime({
   epochSeconds:toInteger(Timestamp)}).epochMillis,Group
   :"4"})
WITH Abstraction
CALL ga.timetree.events.attach({node: Abstraction, time:
   Abstraction.Timestamp, relationshipType: "AT_TIME",
   create: true, resolution: "Second"})
YIELD node RETURN node;
MATCH (a:Abstraction),(b:blackboard_artifact)
WHERE a.artifact_id = b.artifact_id
CREATE (b)-[:ABSTRACTION_LINK]->(a);
```

The final abstracted node is assigned a relationship to the same time tree as the other abstractions and to the original artifact. Figure 18 shows the final abstraction node (orange) with its properties and relationship to the time tree and artifact (blue). The example shown in Figure 18 shows that "Image Stenography 1.5.2 Setup.exe" was downloaded using Google Chrome at 00:16:32 on 01 February 2019.

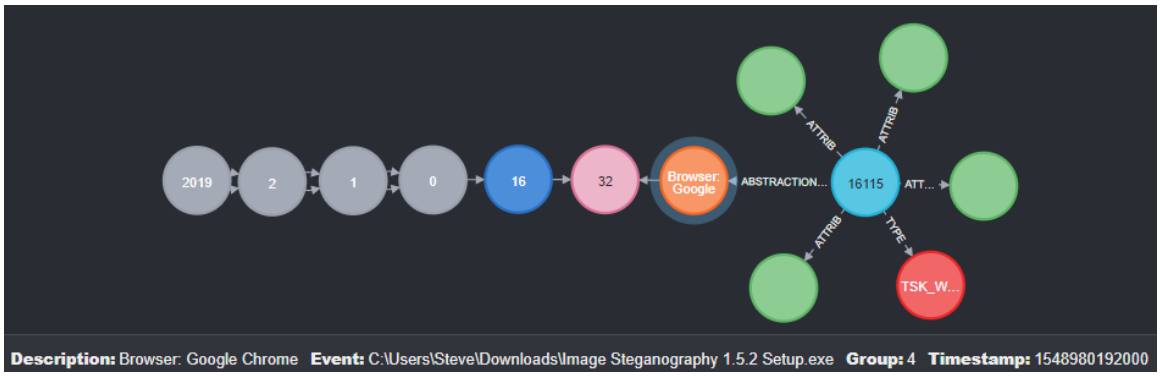

Figure 18: File Downloads Abstraction.

Web History Abstraction

Web history is another important category for forensic examiners. Web history abstraction must be handled differently than the previous abstractions due to the nature of web browsing. When a web page is loaded, many assets are downloaded and appear as individual web history artifacts. The numerous web assets could create many artifacts when only a single web page or web session was created. Using abstraction rules, AIER shows only the most important parts of web history data to the examiner and eliminates data overload. To start the abstraction process, the blackboard attributes data is imported and attribute nodes are created with relationships to the artifact node as shown in Figure 19.

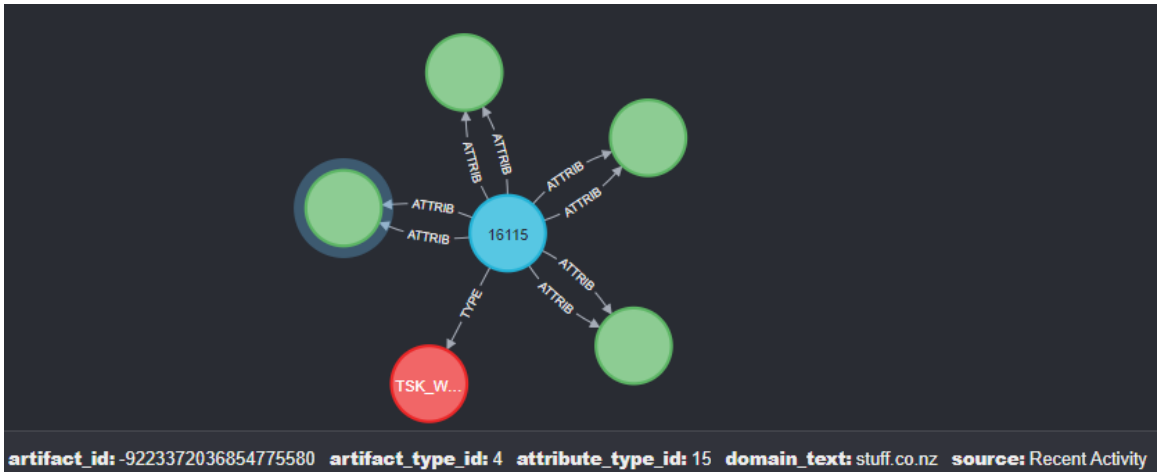

Figure 19: Web History Attributes.

The selected sample attribute node in Figure 19 shows the domain "stuff.co.nz" was visited. The domain information combined with the other attributes creates the final abstraction node. The abstraction provides the username of the account that visited the web page, domain of the web page, timestamp, the specific URL visited, timestamp, and the group ID 5. The following cypher query creates the abstraction node and connects it to the time tree.

```
MATCH (a:blackboard_artifact)
WHERE a.artifact_type_id = "4"
MATCH (a)-[b:ATTRIB]->(c:blackboard_attributes)
WHERE c.attribute_type_id = "15"
WITH *, COALESCE(c.domain_text,"Local File Opened") AS
   Event
MATCH (a)-[d:ATTRIB]->(e:blackboard_attributes)
WHERE e.attribute_type_id = "1"
WITH *, e.url_text AS Trace
OPTIONAL MATCH (a)-[f:ATTRIB]->(g:blackboard_attributes)
```

```
WHERE g.attribute_type_id = "14"
WITH *, COALESCE(g.username_text,"Unknown") AS Description
MATCH (a)-[h:ATTRIB]->(i:blackboard_attributes)
WHERE i.attribute_type_id = "33"
WITH *, i.timestamp AS Timestamp
CREATE (web_trace:web_trace{artifact_id:a.artifact_id,
   Event:Event,Trace:Trace,Description:Description,
   Timestamp:datetime({epochSeconds:toInteger(Timestamp)})
   .epochMillis,Group:"5"});
MATCH (m:web_trace)
WITH m.Event AS Event,m.Timestamp AS Timestamp,m.
   Description AS Description, collect(DISTINCT m.Trace)
   AS Trace, m.Group as Group, collect(m.artifact_id) AS
   artifact_id
MERGE (Abstraction:Abstraction{Event:Event,Trace:("URL: ")
   +apoc.text.join(Trace,'<br>'),Description:("Username:
   ")+Description,Timestamp:Timestamp,Group:Group,
   artifact_id:artifact_id})
WITH Abstraction
CALL ga.timetree.events.attach({node: Abstraction, time:
   Abstraction.Timestamp, relationshipType: "AT_TIME",
   create: true, resolution: "Second"})
YIELD node RETURN node;
MATCH (a:Abstraction)
UNWIND a.artifact_id AS art_id
MATCH (b:blackboard_artifact)
```

```
WHERE art_id = b.artifact_id
CREATE (b)-[:ABSTRACTION_LINK]->(a);
```

The final abstraction node is created and relationships connect it to the original artifact and the time tree. Figure 20 shows the web history abstraction node (orange), the original artifact (blue) and the time tree relationship to the abstraction node. The example node shows that an unknown user visited the domain "stuff.co.nz" at 21:05:55 on 20 January 2019. When Autopsy is unable to determine the user who visited the domain, AIER displays "Unknown". Additionally, the full URL of the visited domain is provided.

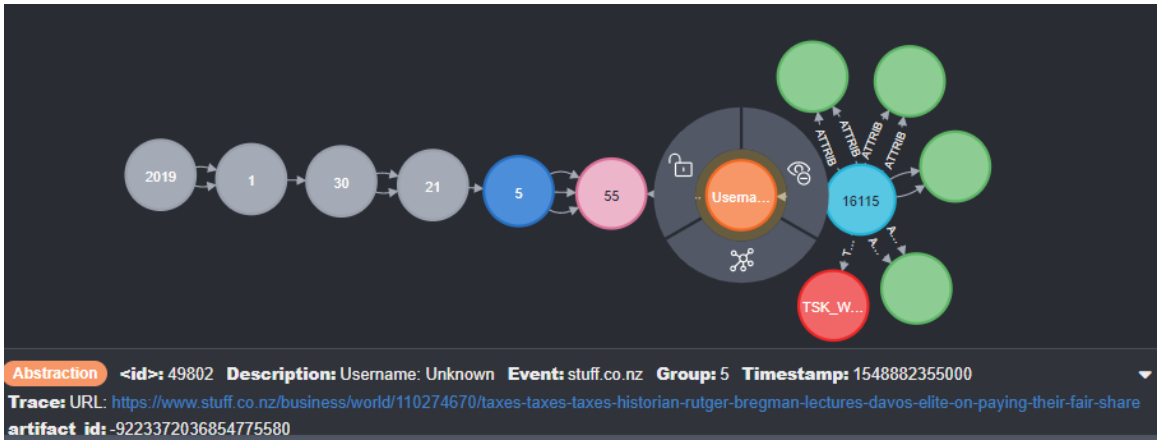

Figure 20: Web History Abstraction.

3.3 GraphQL

GraphQL [31] is the query language that collects the abstraction data from Neo4j and presents it to the timeline UI for display. GraphQL allows AIER to select exactly what data to retrieve and it returns an array of abstractions. GraphQL is made up of schema and resolver functions. The GraphQL schema and resolver functions in this research were primarily developed by Adderley, but modified for Autopsy integration. Figure 21 shows the GraphQL schema used to retrieve the abstraction

data for the timeline UI. The schema uses the start time and end time to return the event string, trace string, description string, timestamp, and group to which the abstraction belongs.

```
type abstraction {
  startTime: Float!
  endTime: Float!
  Event: String
  Trace: String
  Description: String
  Timestamp: Float
  Group: Int
}
```

Figure 21: GraphQL Schema.

Figure 22 shows the resolver responsible for pointing to the abstraction query that translates the schema to a cypher query. The resolver used in AIER is a simplified version of Adderley's resolver function. The only required query for the timeline is the abstraction query because all of the abstraction data was compiled into a single abstraction node for each artifact we want to show in AIER.

```
export const resolvers = {
  Query: {

    abstraction: neo4jgraphql


  }
};
```

Figure 22: GraphQL Resolver.

Figure 23 shows the cypher query that the resolver points to. The cypher query finds the abstraction node that exists between the start time and end time bounds

that the user enters into the timeline interface. The query collects the requested data and returns an array for use in the timeline.

```
type Query {
    abstraction(startTime: Float!, endTime: Float!):
        [abstraction] @cypher(statement:
        "MATCH
        (abs:Abstraction) WHERE abs.Timestamp >= startTime AND abs.Timestamp <= endTime
        WITH abs.Timestamp as timestamp, collect (DISTINCT abs) AS abs UNWIND abs AS x RETURN x")
}
```

Figure 23: GraphQL Cypher Query.

3.4 React

React [33] is a JavaScript library developed by Facebook used to build user interfaces. AIER utilizes the vis.js timeline JavaScript library. The library was originally modified by Adderley to create a visual timeline to display the abstracted artifacts created in earlier steps. AIER allows users to enter a start time and end time then submit the results. The requested start and end time is passed from the timeline UI to GraphQL discussed in Section 3.3. AIER integrates Adderley's Temporal Analysis Integration Management Application (TAIMA) timeline, with some modification, into Autopsy.

3.5 Integrated User Interface

The integrated timeline provides an easily accessible display to view the abstracted artifacts created in the previous sections. After the examiner runs the standard Autopsy ingest modules, and the AIER ingest module, the timeline is ready for use. To view the timeline, the examiner can click the timeline button already available in Autopsy. To retain the original Autopsy timeline functionality, the examiner is presented with a dialog and must choose between the standard Autopsy timeline and the AIER Abstracted timeline as shown in Figure 24.

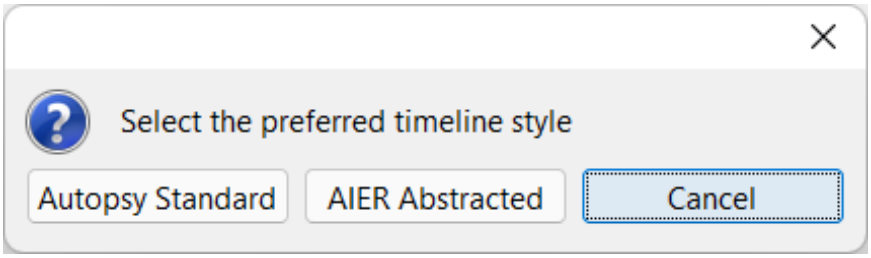

Figure 24: Timeline Selection Dialog.

3.5.0.1 Timeline Data Display

Once the examiner selects the AIER Abstracted timeline, a new window opens to display the timeline interface. The timeline is displayed in a new window using the open source chromium browser to handle the JavaScript interface. The browser window is presented as a new window with no decoration or buttons, but retains Autopsy branding and icons. The window is designed to appear as integrated as possible into the Autopsy UI. Additionally, this secondary window can make use of a second monitor so the examiner can utilize the entire Autopsy interface alongside the timeline. Figure 25 shows the timeline interface presented to the examiner. The examiner can enter their desired start and end time, select submit and the abstracted artifacts are displayed in their respective groups.

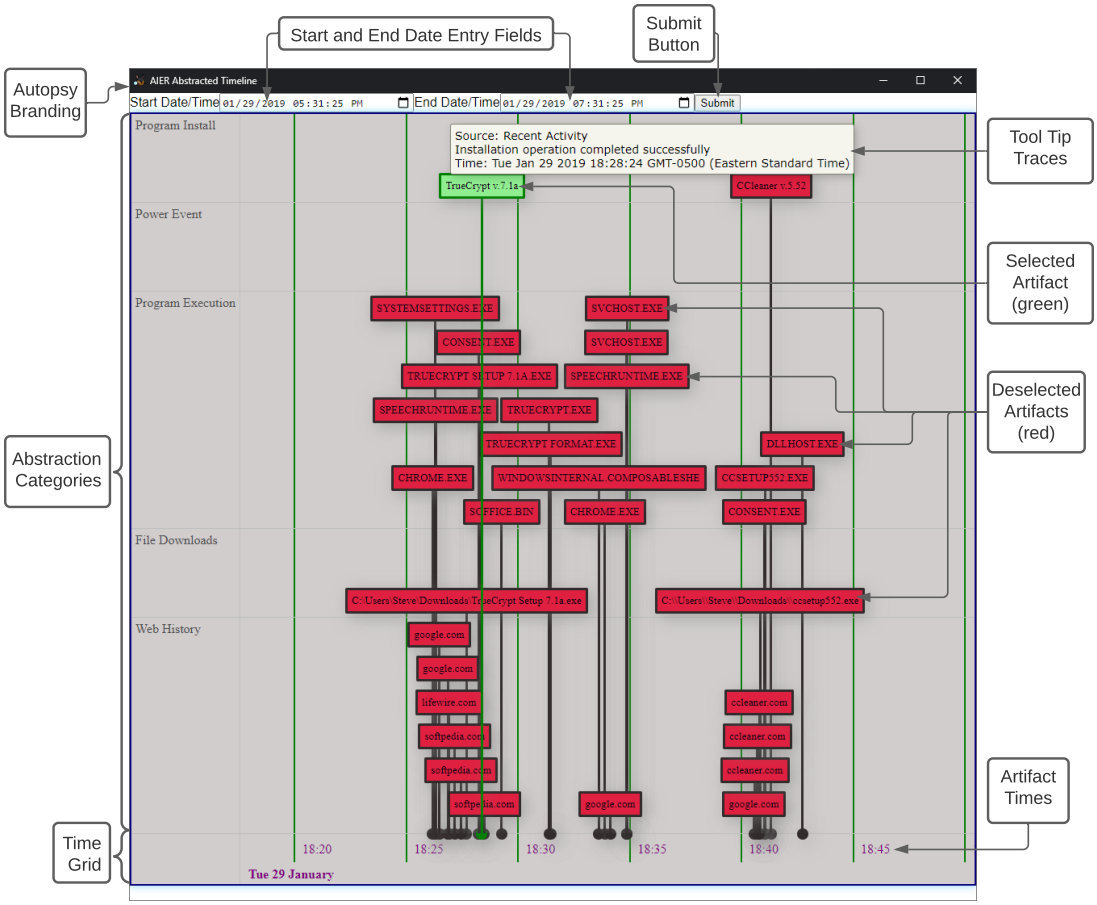

Figure 25: AIER Interface Detail.

The following figures show the steps a user took to download and install the common encryption software TrueCrypt. First, Google Chrome was opened as shown in the program execution category in Figure 26. Next, the user browsed to Google.com shown in the web history category, and searched for True Crypt followed by a visit to a softpedia.com page about TrueCrypt as shown in Figure 27. Next, the user downloaded TrueCrypt from softpedia.com and saved the executable as indicated in the file downloads category shown in Figure 28. Within the program execution category `TRUECRYPT SETUP 7.1A.EXE` was run and the tooltip next to the selected artifact indicates the user executed the downloaded program from the `/USERS/STEVE/DOWNLOADS`

directory at 18:28:17 on 29 January 2019 as shown in Figure 29. Lastly, within the program install category, TrueCrypt version 7.1a was installed a short time after the setup executable was run and is shown in Figure 30.

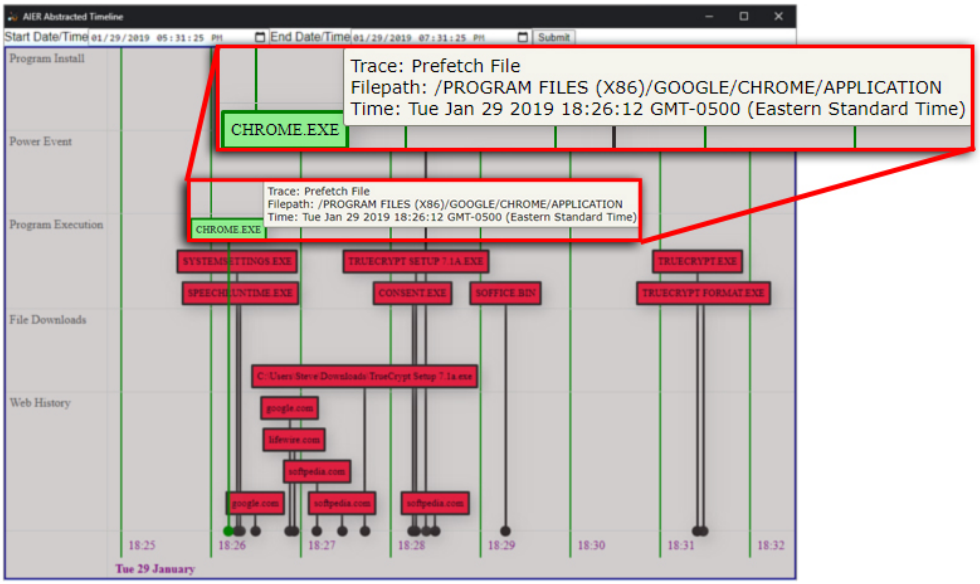

Figure 26: Chrome Execution.

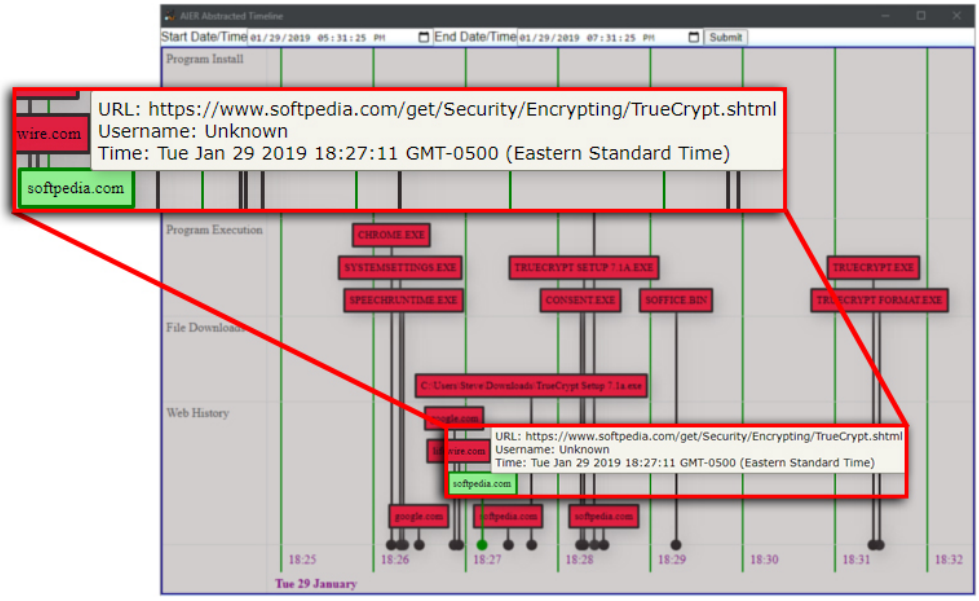

Figure 27: Google Web History.

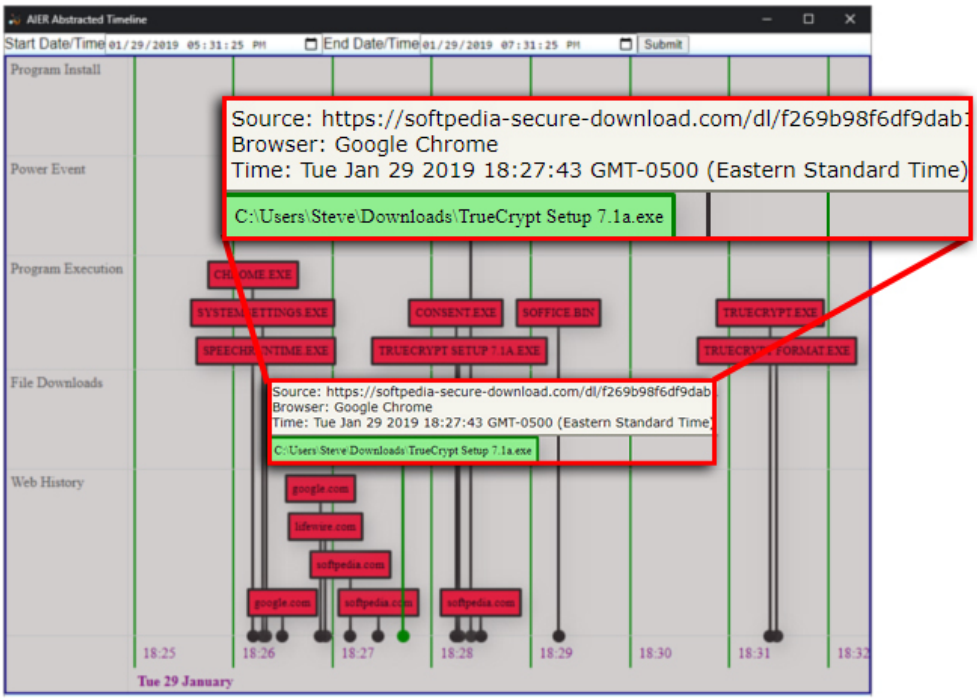

Figure 28: TrueCrypt Downloaded.

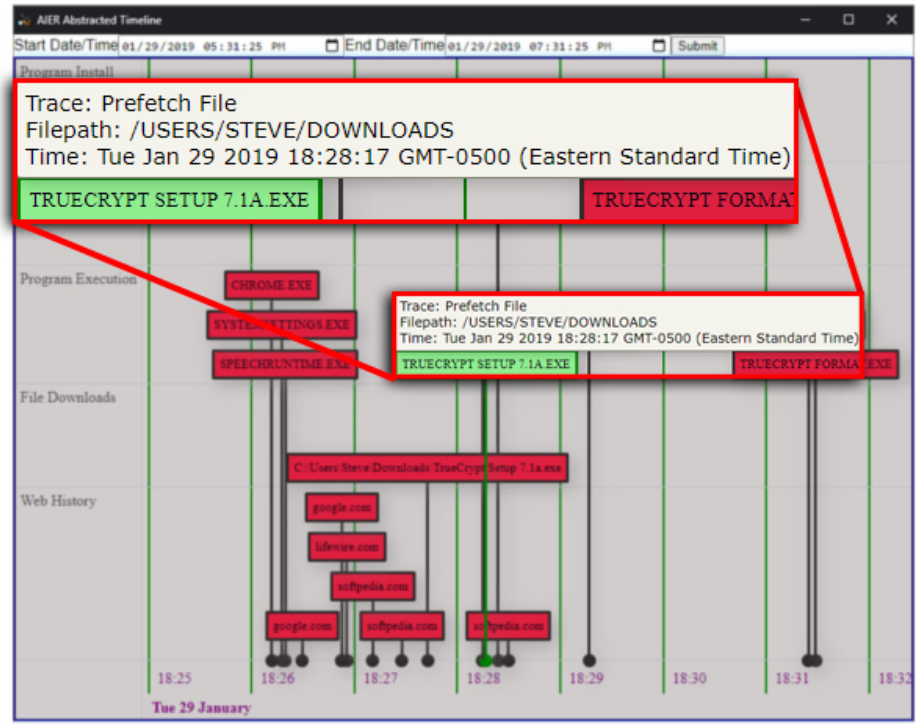

Figure 29: TrueCrypt Setup Executed.

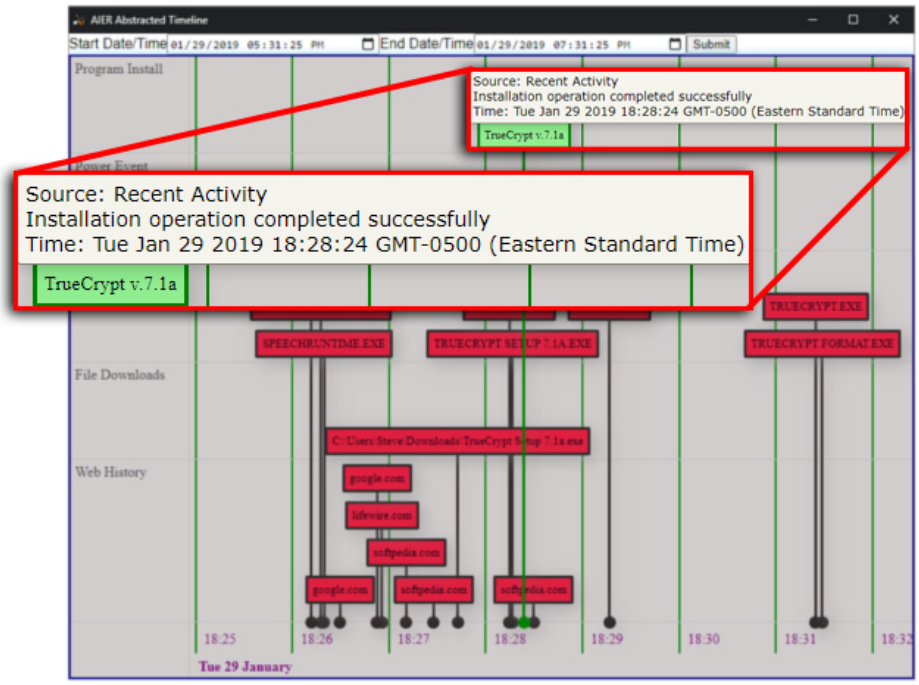

Figure 30: TrueCrypt Installed.

3.5.0.2 Autopsy - AIER UI Integrations

In some cases, an examiner may not have a date and time to start the reconstruction process or may have limited information to work with. Without start and end date parameters the examiner may use Autopsy's keyword search to find related artifacts. The integration of AIER allows the examiner to right-click an artifact and view it in the AIER timeline alongside other temporally related artifacts. The right-click context menu selection is shown in Figure 31.

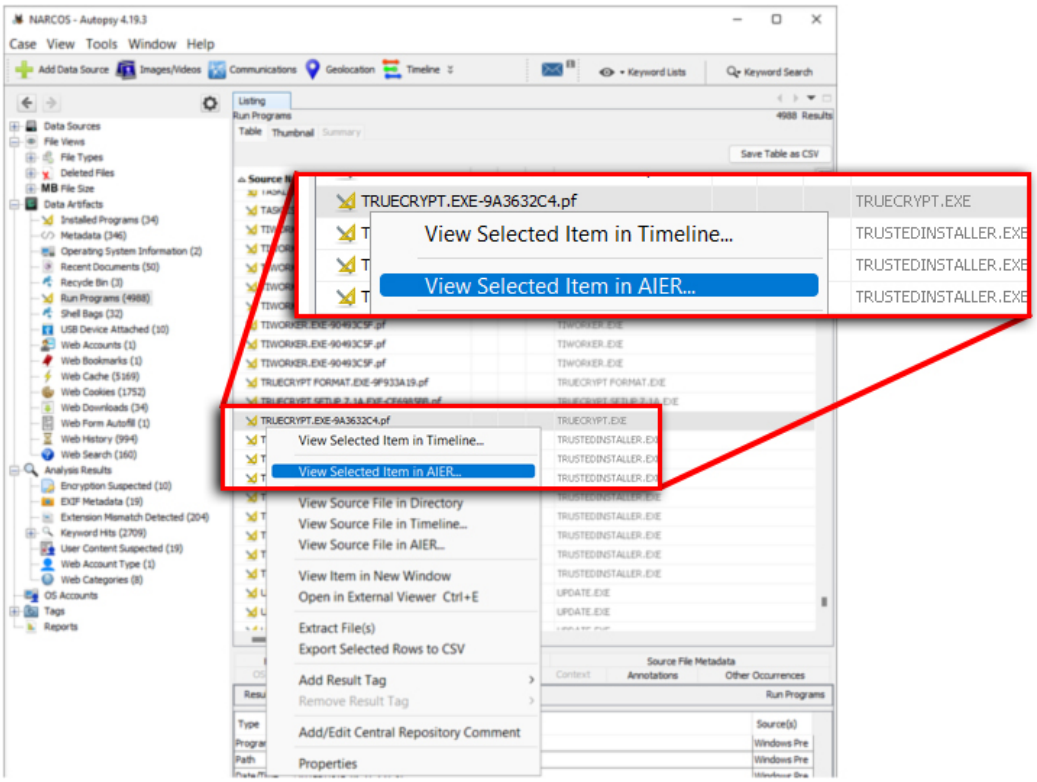

Figure 31: View in AIER Context Menu.

A dialog appears, as shown in Figure 32, allowing the examiner to select the timestamp they wish to use within the AIER timeline. They may choose the modified, accessed, changed, or created time. The examiner can also specify the number of seconds, minutes, hours, or days they want to include on either side of the selected item allowing the examiner to see the selected artifact and all the other related artifacts on the timeline within the selected time range.

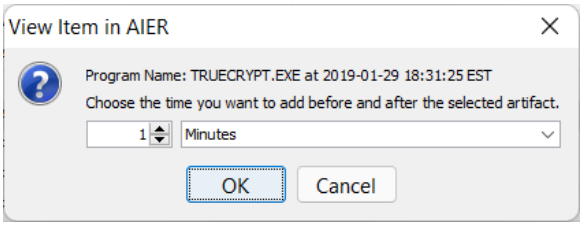

Figure 32: View in AIER Dialog.

An examiner often needs to authenticate an artifact or needs more information about an artifact they find in AIER. If the examiner double-clicks an artifact on the AIER timeline, the data about that artifact is transmitted back to the Autopsy interface and a new content view pane is shown with additional data about the artifact. Figure 33 shows the content viewer pane when TrueCrypt was double-clicked on the AIER timeline. The double-click function completes the integration of the AIER timeline into Autopsy to automate event reconstruction and enable examiners to gain an understanding of the events that occurred on the target image.

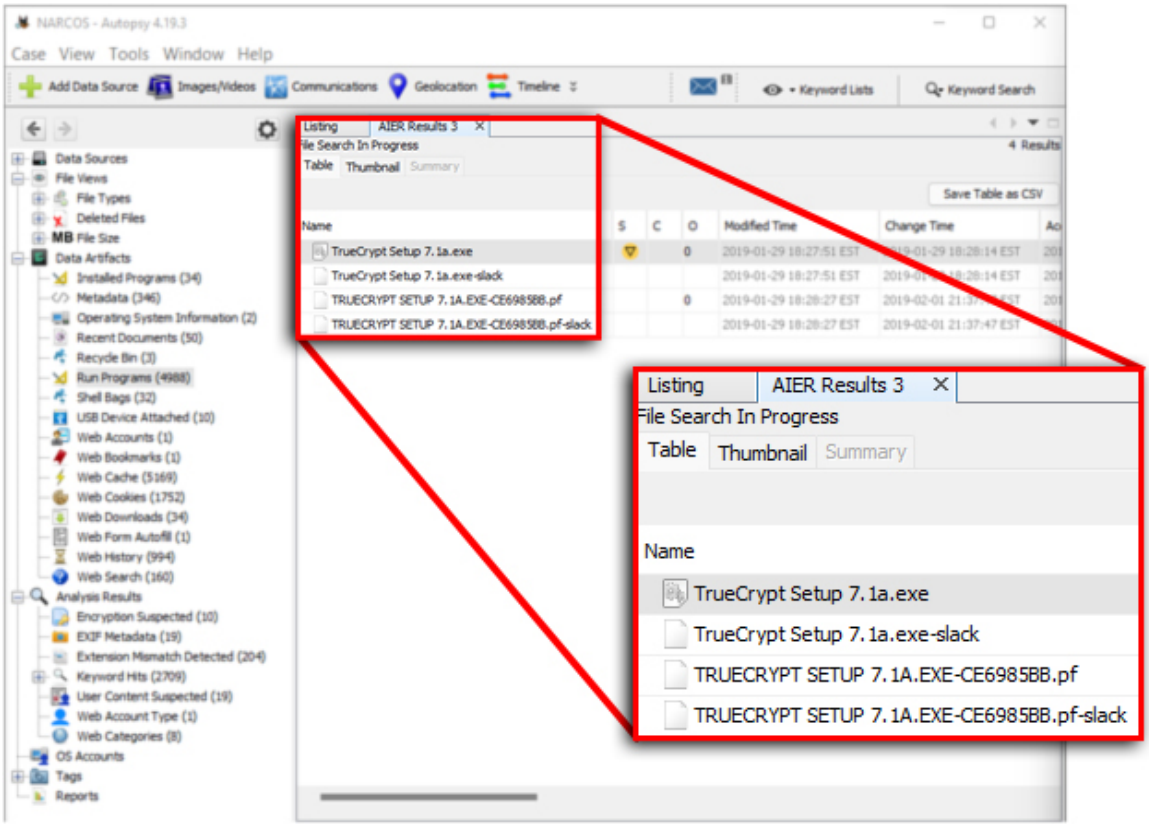

Figure 33: AIER Double-Click Results.

3.6 Summary

The integrated AIER timeline and ingest module help to automate the reconstruction process for the digital forensic examiner. Prior to this research, to make use of the ideas and processes researched by Schelkoph and Adderley, the forensic examiner had to manually process an image, transform the data, create the graph database, and initiate the timeline. The integration and automation of these processes through AIER improves the examiner's workflow, decreases information overload, and provides the examiner with the information needed to reconstruct cyber events more rapidly. Now, even a novice examiner can use new or old autopsy cases with the ingest module. The ingest module examines the Autopsy database, uses Plaso to extract Windows event logs, compiles the data, abstracts the data into high-level events, organizes the data for rapid querying within the time tree, and presents the data on the integrated AIER timeline. In addition to entering manual start and end times on the timeline, the examiner can right click items in Autopsy for further inspection in AIER. Lastly, the examiner can double-click items in AIER for further inspection in Autopsy. These varying uses for the timeline give the examiner great flexibility to further enhance their workflow.

IV. Integration Assessment Process

This chapter discusses the assessment of the Autopsy Integrated Event Reconstruction (AIER) timeline integration and data ingest process. Included in the chapter is an explanation of how the typical digital forensics examiner would use AIER. Additionally, three user stories were developed to test the integration and ingest process. Lastly, this chapter explains how the integration of the AIER timeline and ingest process fit within the workflow of the digital forensics examiner.

4.1 Assessment

Chapter III explained the process to ingest an image in Autopsy through the AIER ingest module and display an abstracted timeline. AIER allows examiners to improve their workflow through abstraction, automation, and visualization. The following sections discuss the typical digital forensic examiner workflow, and illustrate three use cases that show the viability of the automated event reconstruction process. Viability is confirmed by locating key artifacts within the five AIER timeline categories. In each scenario I act as the examiner and had limited information prior to the development of the user scenario to prevent bias. Each system image has a set of specific questions or goals for each image. Accurately answering these questions or accomplishing these goals proves the viability of the AIER process.

4.2 Digital Forensic Examiner Workflow Integration

Similar to physical investigation processes, digital forensics relies on three main phases. The extraction and preservation of data, identification of evidentiary artifacts, and reconstruction and analysis of events. AIER is concerned with the last two phases, identification and reconstruction of events. The preservation of data only requires

the examiner to start the forensic duplication of the target image. The last two phases are the most time consuming within the examiner's workflow. Identification and reconstruction of the events found on the target image require the examiner to manually search and document artifacts. Event reconstruction is a skill and requires the judgement of the examiner to determine if an artifact is substantial. While the goal is not to replace the examiner's judgement, AIER assists the examiner by eliminating irrelevant data through abstraction. The following sections illustrate the integration of the AIER timeline in the examiner's workflow in more detail.

4.3 User Stories

The following user stories demonstrate the examiner's ability to find key artifacts within the five abstraction categories using the AIER timeline. Each scenario starts with a brief background story followed by a walkthrough of the ingest process and event discovery in the integrated timeline.

The evaluation of AIER uses three Windows system images provided by 1) the National Institute of Standards and Technology (NIST) Computer Forensic Reference Data Set [35], 2) Digital Corpora [36], and 3) Air Force Institute of Technology (AFIT). These image operating systems include Windows XP and Windows 10. Currently, the AIER timeline is limited to Windows operating systems, however Section 5.3 discusses the inclusion of other operating systems in the future. NIST provides questions and the answers to help guide the forensic process for the first image and scenario one will use a portion of these questions that specifically fall within the abstraction categories. The other two scenarios set a goal for the examiner and tests AIER to see if it can meet the goal.

4.4 Scenario 1 - Mr. Evil

The following scenario involves a suspect by the name of Greg Schardt [37]. Schardt's computer, a Dell CPi notebook, was found abandoned on 20 September 2004 with a Personal Computer Memory Card International Association (PCMCIA) wireless networking card and homemade antennae. Schardt is suspected to be a hacker and is known by his online persona, Mr. Evil. His associates say he would occasionally park his vehicle near wireless access points and attempt to intercept traffic in an effort to acquire credit card numbers, usernames, and passwords. The goal of this scenario is to locate evidence of hacking software and any data that identifies Mr. Evil as the user of the computer. The scenario was chosen to illustrate the ability of the timeline to work with older operating systems. Scenario 1 uses Windows XP, and due to the age of this operating system some artifacts may not be detectable. This is a limitation of the Sleuth Kit and Autopsy. The AIER ingest module relies on the accuracy of the Autopsy database to create abstracted artifacts.

Using the integrated timeline and ingest module we will answer the following questions, and where applicable, confirm their accuracy with answers provided by NIST:

1. Find at least six installed programs that may be used for hacking [35].
2. When were these programs run?
3. What user likely ran these programs?
4. Find any indication the installed programs were downloaded from the internet.
5. Determine the internet source of the downloaded programs.
6. Note any power on and power off events indicated by the system [35].
7. Find additional information to determine the identity of the primary user.

4.4.1 Scenario 1 Walkthrough

To begin processing the image, a new case is created in Autopsy and the standard Autopsy modules are run. The standard Autopsy modules populate the Autopsy database for use by the AIER ingest module. Next, the AIER ingest module is run to build the abstracted timeline for examiner review as shown in Figure 34. After completion of the ingest module, the examiner searches for artifacts within the image.

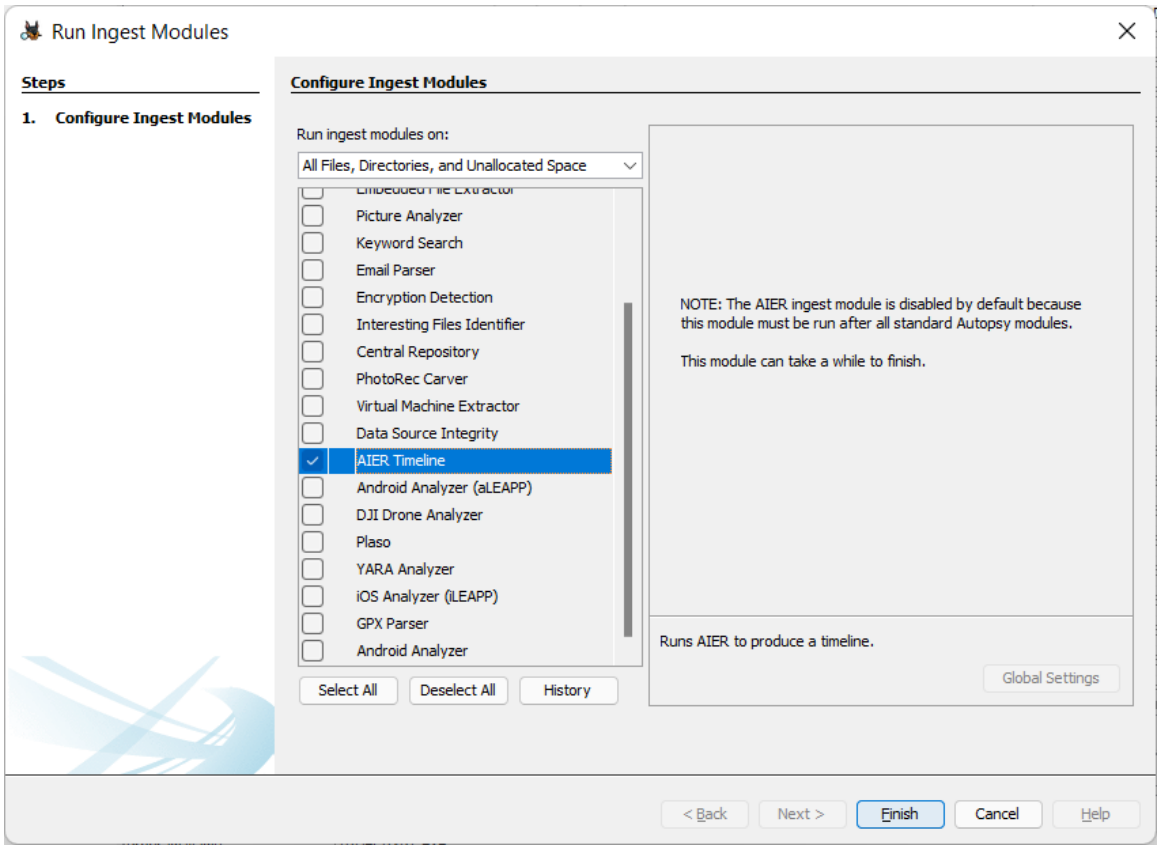

Figure 34: AIER Timeline Ingest.

After completion of the ingest module, the examiner begins searching for artifacts within the forensic image. The target system was found abandoned on 20 September 2004, therefore a search between 1 September and 20 September 2004 is a good place to start. September yields no results so the examiner conducts a search for the month

of August. The search yields many results and creates a large scroll-able timeline as indicated in Figure 35. These results are concentrated around the 19, 20, 25, 26, and 27th of August 2004.

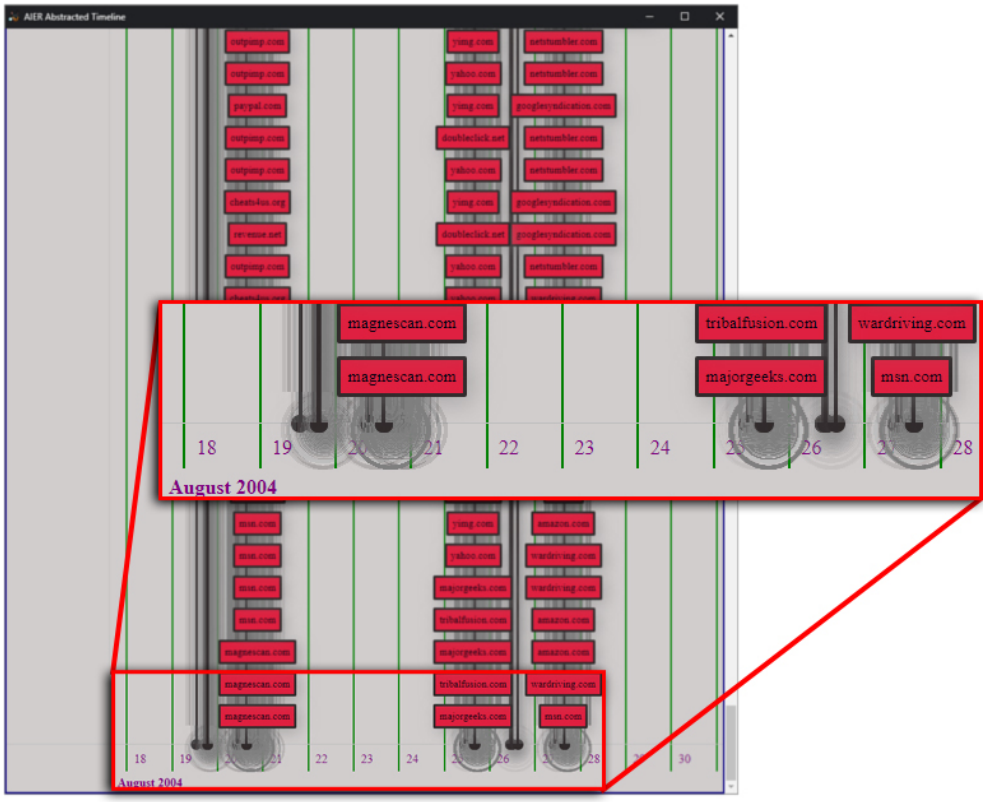

Figure 35: August Search Results.

Starting with 19 August 2004 we can conduct a search for the entire day. The search yields many results but nothing that indicates suspicious behavior. Next, a search including the entire day of 20 August 2004 provides many results with many suspicious artifacts. These artifacts are concentrated around 11:00:00 to 12:00:00, 15:00:00 to 16:00:00, 19:00:00 to 20:00:00, and 20:00:00 to 21:00:00. Starting with a search for the 11:00:00 to 12:00:00 time block we find the system was started at approximately 11:01:22 as shown in Figure 36.

Figure 36: System Start.

Shortly after system startup, a series software installs occurred to include the following hacking tools as shown in Figure 37:

- Anonymizer Bar 2.0 - Installed at 11:05:09
- Cain and Abel v2.5 beta45 - Installed at 11:05:58
- Cute FTP - Installed at 11:09:02
- Cute HTML - Installed at 11:09:03
- 123 Write All Stored Passwords - Installed at 11:13:08

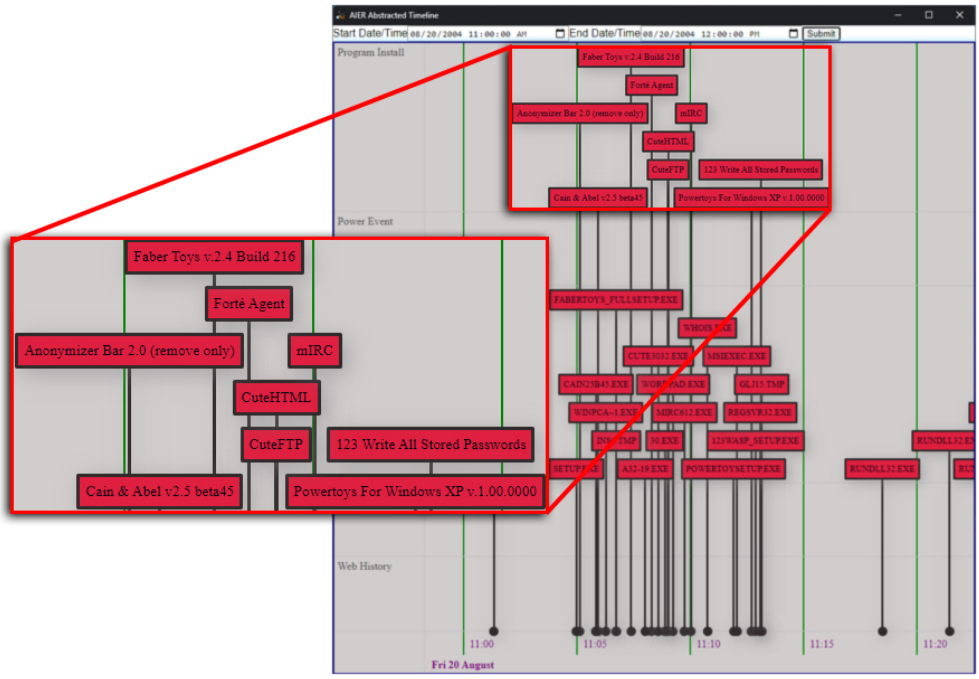

Figure 37: Hacking Tool Installations.

In addition to the program installation artifacts we can see the execution of the setup files that initiated the installation of the programs in Figure 38. A review of the executed programs reveals the execution of the setup file for Power Toys. Power Toys was executed from one of Mr. Evil's directories and could indicate that the user Mr. Evil executed these installations as shown in Figure 38. No additional artifacts are shown in the other categories to indicate the source of these executables.

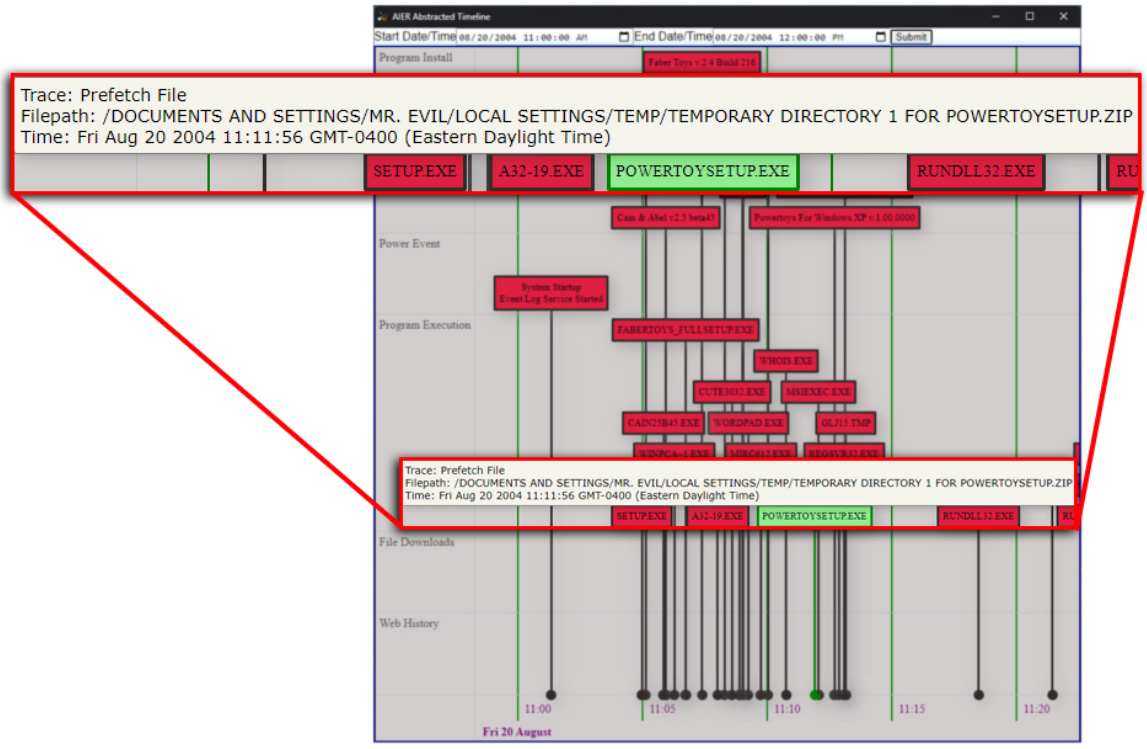

Figure 38: Setup File Execution.

Next, a search for the entire day on 25 August 2004 reveals more suspicious activity. The system was booted and shortly after, the suspect executed `LALSETUP250.EXE`. Figure 39 shows the suspect installed Look@LAN at approximately 11:56:11 and the installation is another indication of the suspect using hacking software. Additionally, Figure 40 shows user Mr. Evil doing web research about hacking and computer threats on elitehackers.com later that day.

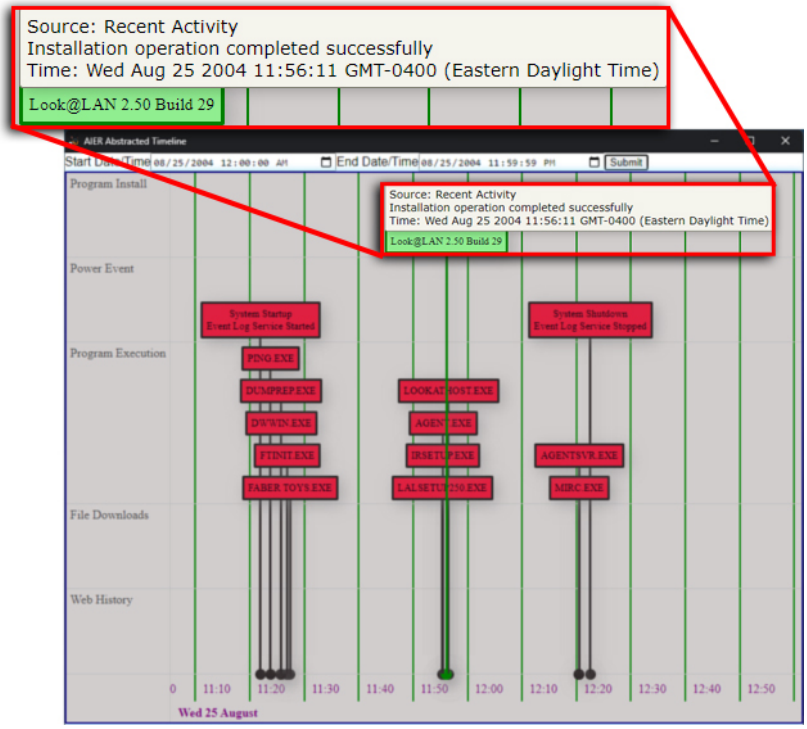

Figure 39: Look@LAN Installed.

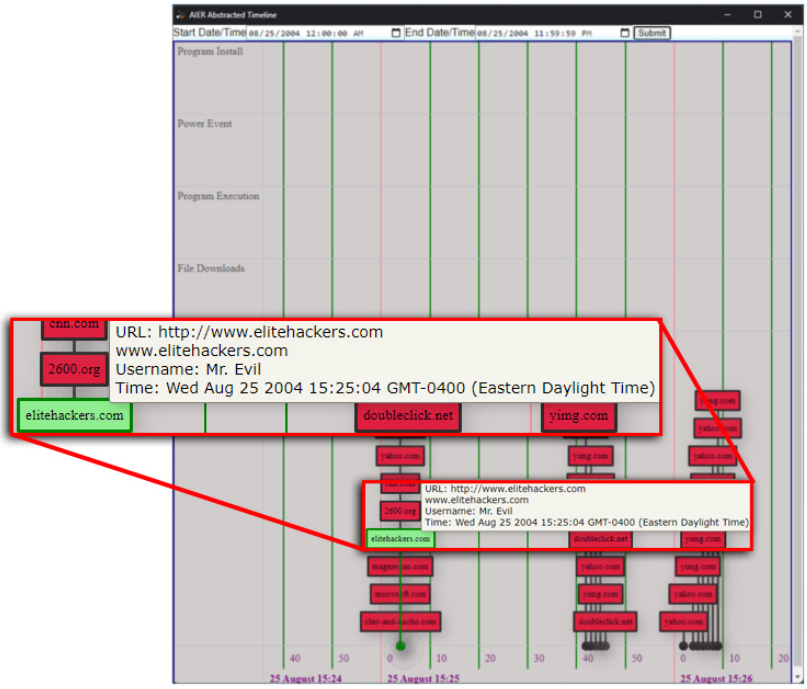

Figure 40: Hacking Research Web History.

Next, a search for the entire day of 26 August 2004 reveals an attempt by the user to use Look@LAN to potentially hack other systems. Figure 41 shows the user's hacking session lasted approximately one hour from 11:00:00 to 12:00:00.

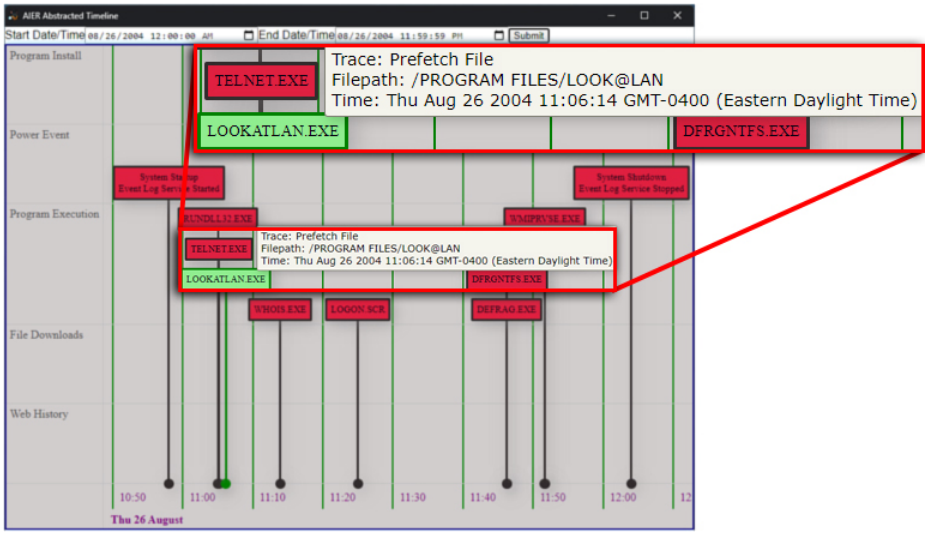

Figure 41: Hacking Session.

Lastly, a search for the entire day of 27 August 2004 reveals a substantial amount of activity indicating the user was trying to hack other systems. Figure 42 shows the system was started followed by the installation of `NETSTUMBLER.EXE`, a common wireless network utility. Shortly after the execution of `NETSTUMBLER.EXE` the user ran `WINPCAP`, a tool that captures network packets. After the packet capture was complete, the user installed `ETHEREAL`, the original name of Wire Shark, a common packet inspection tool. Lastly, the user ran `CAIN.EXE`, a common password cracking tool. Shortly after running `CAIN.EXE` the user Mr. Evil browsed to msn.com and signed in. The execution of `CAIN.EXE` followed by the log in at msn.com indicates Mr. Evil likely accessed someone's account.

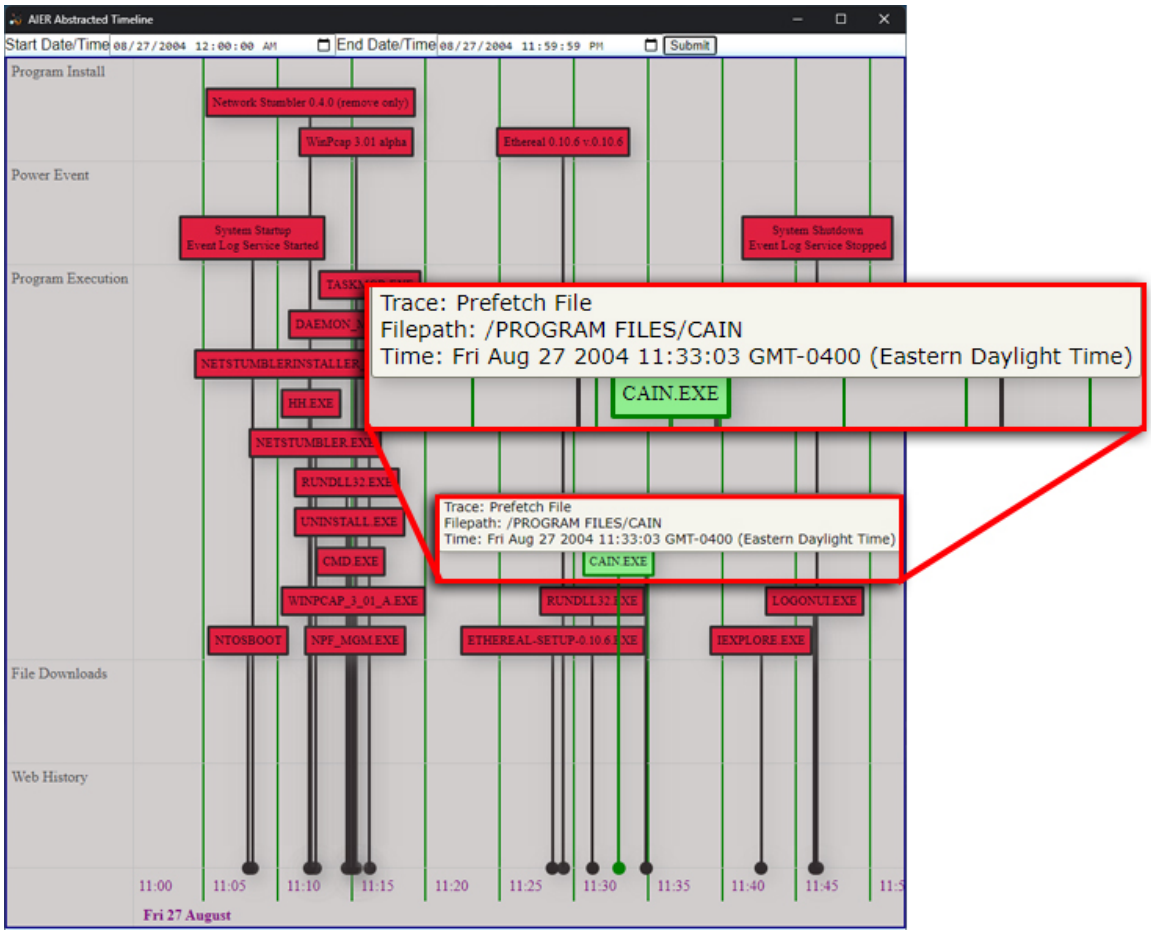

Figure 42: Last Hacking Session.

4.4.2 Scenario 1 Analysis and Results

The integration of the AIER timeline and ingest module into Autopsy allowed the examiner to find the necessary information to determine the steps the hacker took in the scenario. The scenario provided a general starting time frame and the examiner was able to determine concentrated dates of activity. Working through each date the examiner could see a pattern of hacking research, hacking tool execution and installations, and the potential capture of a user's data. Additionally, the examiner was able to answer most of the questions described at the beginning of this section. The examiner was able to find all six hacking tools, when they were run, the user of

these programs, and the examiner was able to note system power on and power off events. Unfortunately, Autopsy was unable to provide any web download history, and subsequently the AIER timeline was unable to display download results. The lack of information was due to the age of the operating system and limitations of Autopsy. Where applicable, the questions provided by NIST were verified to be accurate when compared to the results the AIER timeline provided.

4.5 Scenario 2 - NARCOS

The following scenario is provided by Digital Corpora and involves three suspects [38]. The scenario will only focus on one of the suspects named Steve Kowhai and the scenario will only use Kowhai's computer image. Digital Corpora provides the following background information about Kowhai:

> Steve is a big player drug distributor/dealer in the lower north island of New Zealand and is wanting to find some quality product to expand his growing empire even more. Steve has contacted a source (John) in the US to smuggle in a taster of the product he plans to buy in larger quantities later. Steve has provided John with information about New Zealand and points on how best to smuggle the product into Wellington without raising any alarms at customs. Steve knows a thing or two about digital forensics and decided to use steganography to hide the document within a picture [36].

The image used in this scenario utilizes Windows 10 Pro and represents a more modern operating system as compared to scenario one. The more modern operating system results in more artifacts detected by Sleuth Kit. Better artifact detection subsequently increases the number of abstracted artifacts displayed in the AIER timeline. In this scenario there are numerous artifacts and potential evidentiary items. For the purposes of this research the focus will only be on discovering steganography software and the potential source.

4.5.1 Scenario 2 Walkthrough

The scenario background did not provide any starting date to begin working from, therefore to begin working with this image we will start in the Autopsy user interface to establish a start date for the timeline. The background states Steve is familiar with steganography, so we can conduct a keyword search in Autopsy to look for programs that could assist Steve in hiding data. Figure 43 shows the results of the keyword search for the term "steganography".

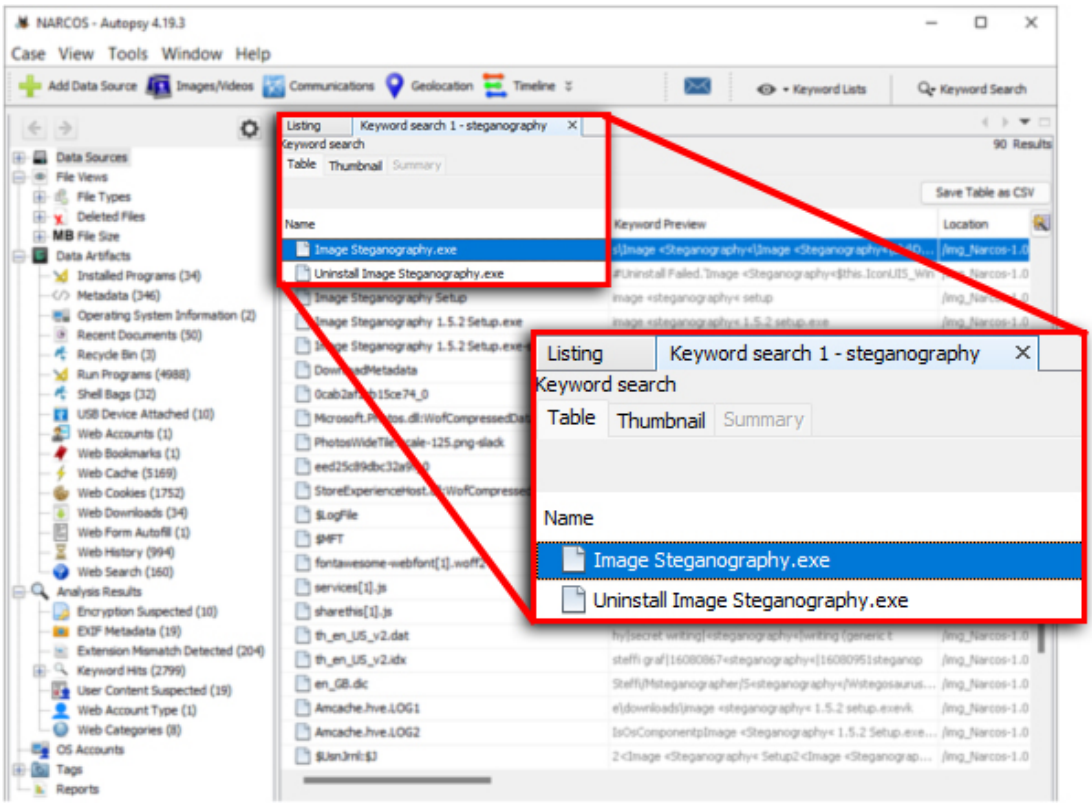

Figure 43: Steganography Results.

Next, right-clicking on `Image Steganography.exe` and selecting "View Source File in AIER..." displays a dialog box as shown in Figure 44. Selecting the Modified Time and a reasonable scope of 10 minutes either side of the timestamp for the selected artifact we can view the execution of the file in the AIER timeline.

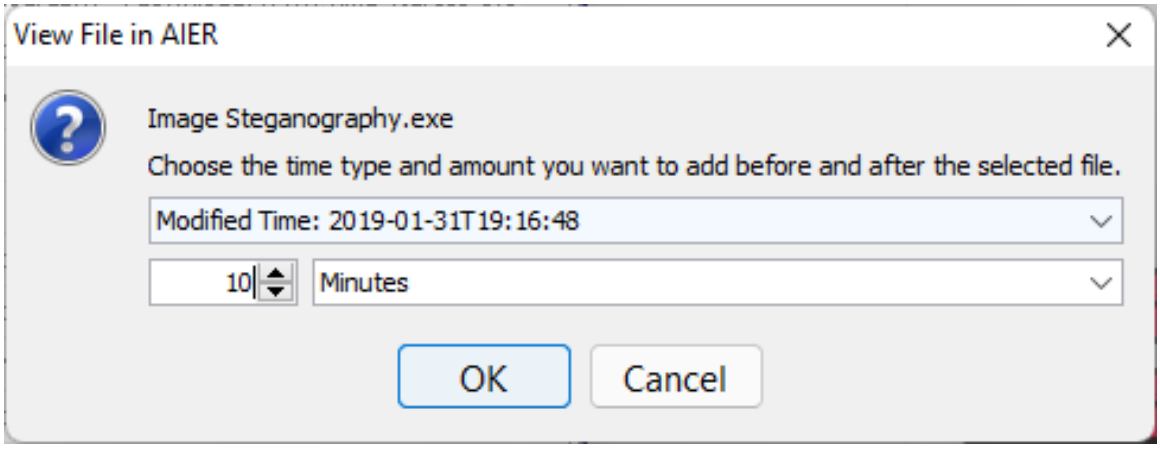

Figure 44: AIER Dialog.

Selecting a scope of 10 minutes surrounding the selected artifact gives additional context around the artifact that can help determine its origin. The timeline displays the execution of the steganography program after a Google search for "image steganography download" and a subsequent visit to sourceforge.net. Sourceforge.net is a common file hosting website. Shortly after a visit to sourceforge, a file is downloaded to user Steve's downloads folder as shown in Figure 45. Soon after the download, the Image Steganogrpahy executable is executed.

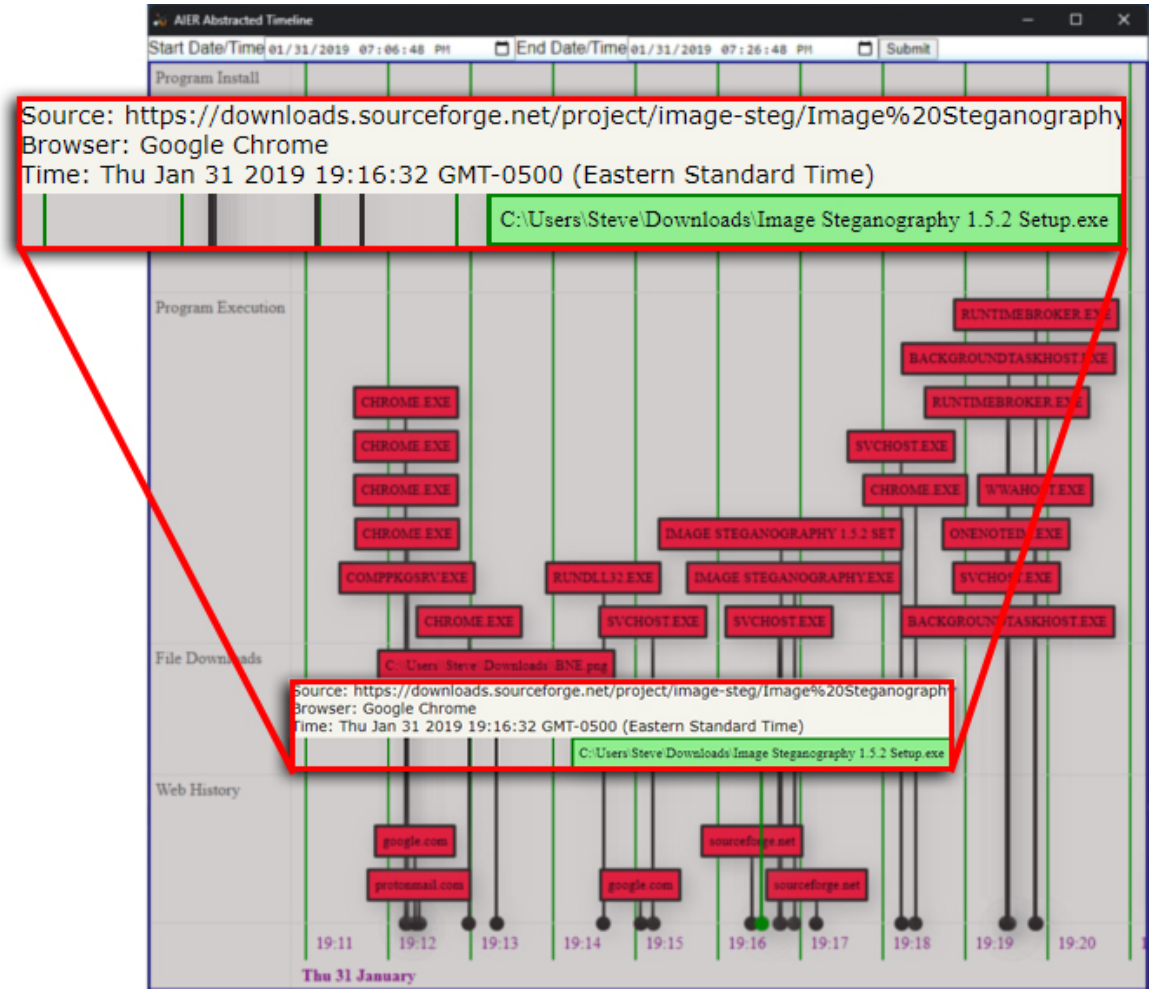

Figure 45: Steganography Download.

Figure 45 indicates Steve has hidden data using the steganography tool and likely sent the packaged image via protonmail, Steve's chosen webmail service. Using the Autopsy user interface and the AIER timeline together enables the examiner to find artifacts, determine their source, and reconstruct the significant artifacts along a timeline. Without the AIER timeline the examiner needs to manually locate the source of the executable and trace how it got onto the system. AIER eliminated the need for the examiner to manually gather all the artifacts with timestamps surrounding the steganography executable. The examiner would need to write down the times-

tamp for these artifacts and conduct searches based on those timestamps to manually find correlated artifacts. Additionally, the examiner would need to manually order the artifacts to establish their order. AIER automates this process to improve the examiner's workflow.

4.5.2 Scenario 2 Analysis and Results

Scenario two provided a different opportunity to use the AIER timeline integration. The scenario did not provide a starting time frame for the examiner to work from. The examiner in this scenario started the process in the Autopsy user interface with a string search. The string search resulted in the discovery of steganography software. The search did not provide any additional information about the source of the software. However, by using the right-click function and viewing in the AIER timeline the examiner was able to determine the web based source of the software, when and where the software was downloaded, who downloaded the software, and when the software was used. The examiner was also able to gather more information about how the embedded image was transmitted after it was created with the steganography tool. The full trace of suspicious activity coupled with the integrated Autopsy features allows the examiner to make quick and accurate reconstructions.

4.6 Scenario 3 - Boddy Inc.

The final scenario uses Windows 10 and was taken from the Computer Science and Computer Engineering (CSCE) 527 Cyber Forensics course offered at the AFIT. This image is part of an ongoing police investigation due to an altercation between Mr. Golden Mustard and Miss Ruby Scarlet arguing over a computer. During the response police seized the hard drive image used in this scenario and some questionable Universal Serial Bus (USB) thumb drives. One of the thumb drives was a keylogger

device. The keylogger did not provide any clue as to who is responsible for its use on the computer. There are seven individuals that used the shared computer; Mr. Boddy, Col Mustard, Mr. Green, Professor Plum, Miss Scarlet, Mrs. Peacock, and Mrs. White. In this scenario the examiner attempts to figure out who is responsible for the keylogger found connected to the system.

4.6.1 Scenario 3 Walkthrough

This scenario provides very little information for the examiner to work from. The examiner starts their search by using the Autopsy timeline histogram to determine when most of the activity on this system occurred. Viewing the histogram indicates that most activity occurred in 2017 and the only web activity on the system occurred in 2017. Additionally, in 2017, nearly all activity occurred in the month of June as shown in Figure 46.

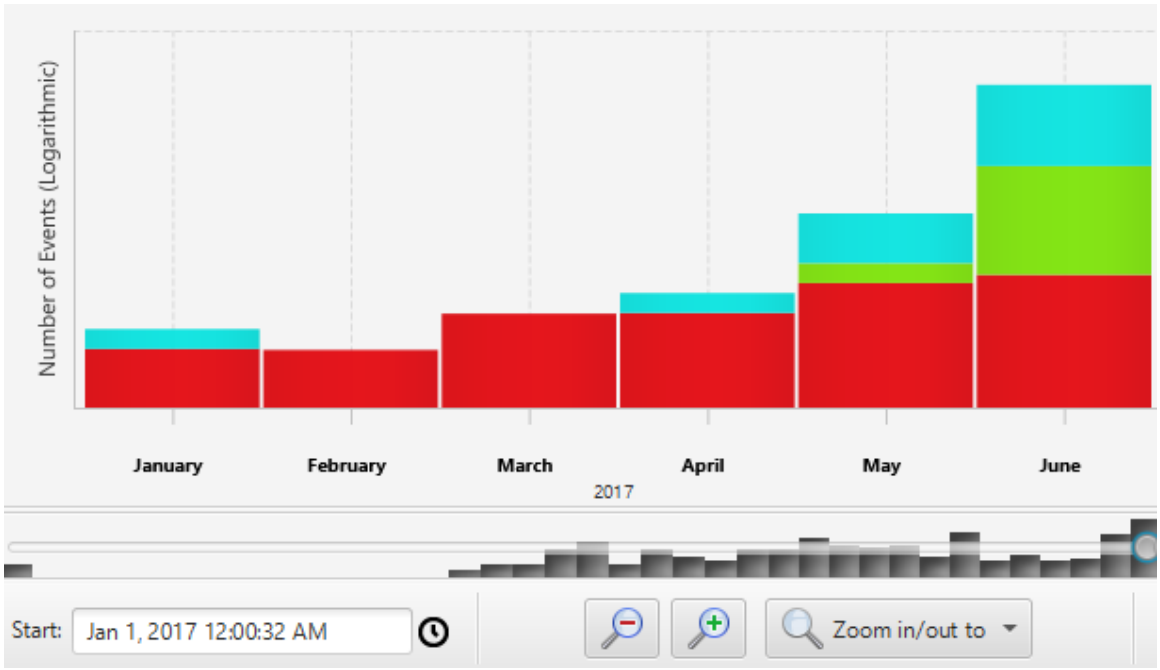

Figure 46: Boddy Inc. Activity.

Knowing most of the events occurred in the month of June, now the examiner can do a simple keyword search for "keylogger" in the Autopsy user interface. The search resulted in web cache artifacts including the word keylogger and most of the events occured in June. One of Autopsy's first results is a photo of a PS/2 style keyboard keylogger as shown in Figure 47.

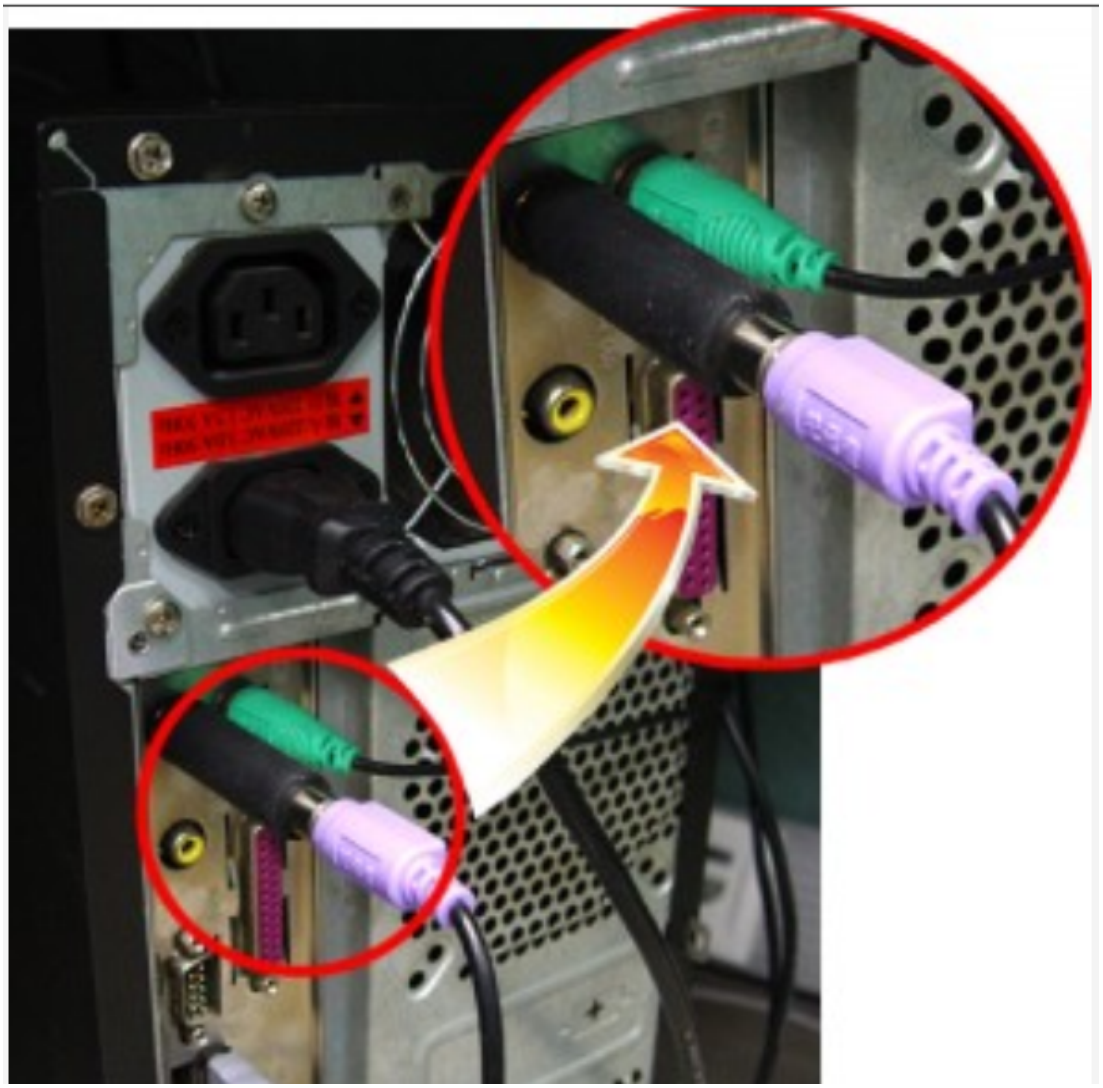

Figure 47: PS/2 Keylogger.

Next, the examiner discovers more information about this internet session by right-clicking and selecting the artifact from Figure 47 then choosing "View Source

File in AIER...". The examiner chooses a reasonable time scope of one hour. Internet activity is displayed including searches for penetration testing and software keyloggers. Hovering over an artifact and reading the tooltip box provides additional information. Figure 48 shows an unknown user viewed the web site symantec.com and specifically viewed an article about spyware keyloggers.

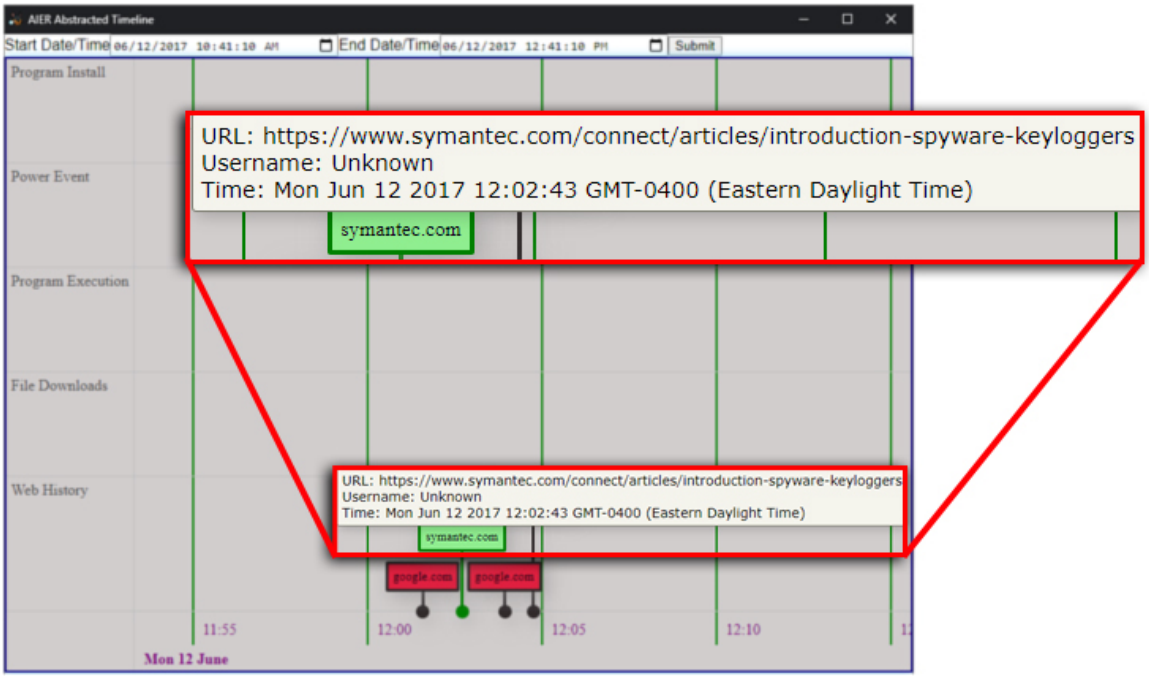

Figure 48: Keylogger Searches.

The timeline indicates the user is unknown because the Autopsy database did not provide a specific user attributed to this session. However, there are other ways to determine the user who visited this web page. Double clicking the highlighted web history artifact shows the examiner more information about the artifact in the Autopsy user interface.

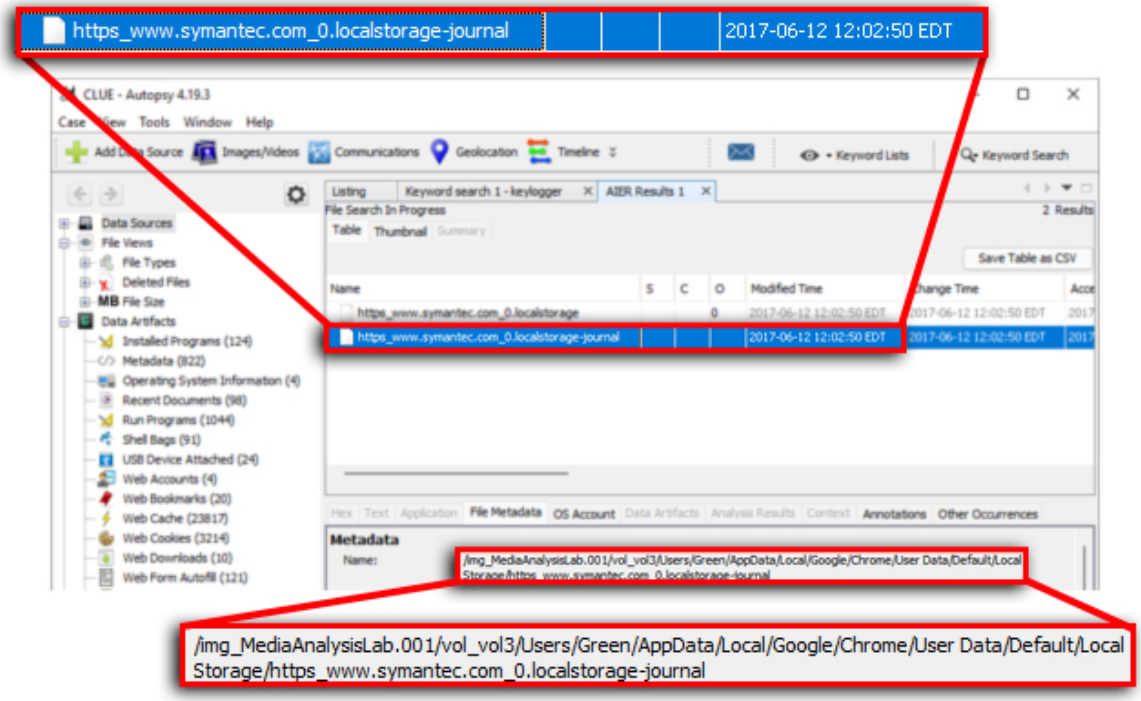

Figure 49: Keylogger Trace.

After double clicking an artifact in AIER, an Autopsy content viewer pane opens to display any potential trace that could help figure out who viewed the web page. At the bottom of Figure 49 you can see the file path for the Google Chrome cache information related to this web page visit. The file path would indicate that Mr. Green was doing research about keyloggers. Figure 50 shows a visit to hakshop.com shortly after the symantec.com visit. Double clicking on the hakshop.com artifact opens another Autopsy content viewer pane to display the metadata about this particular web page visit. The metadata reveals that Mr. Green also visited this web page as shown in Figure 51

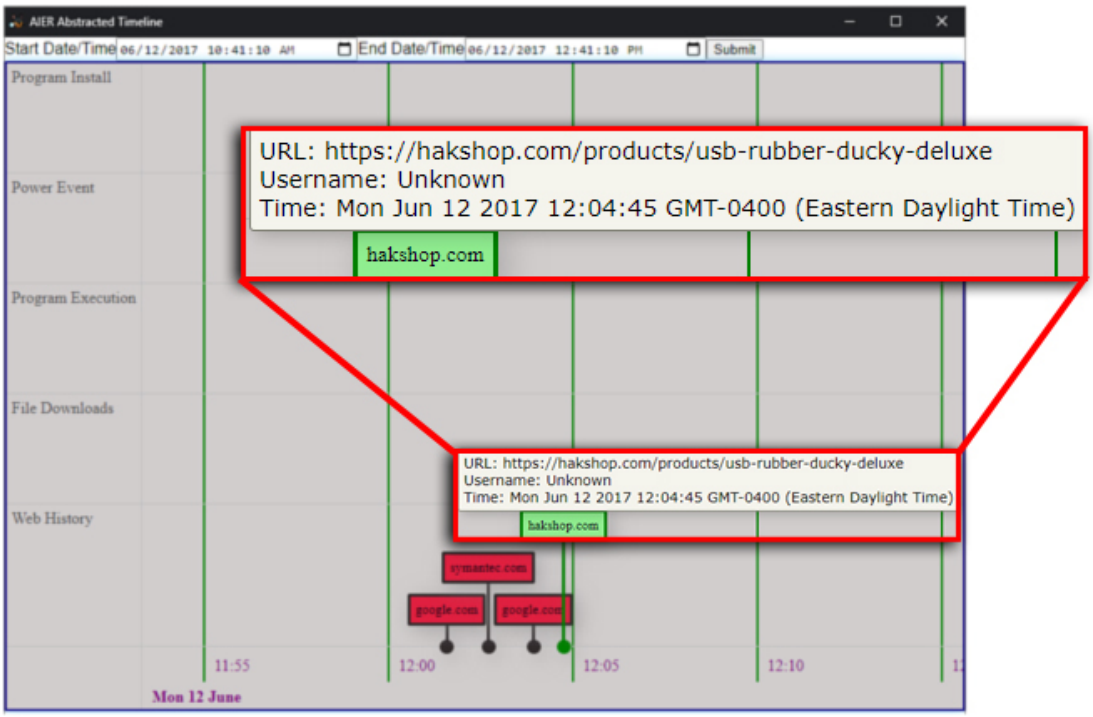

Figure 50: USB Rubber Ducky Trace.

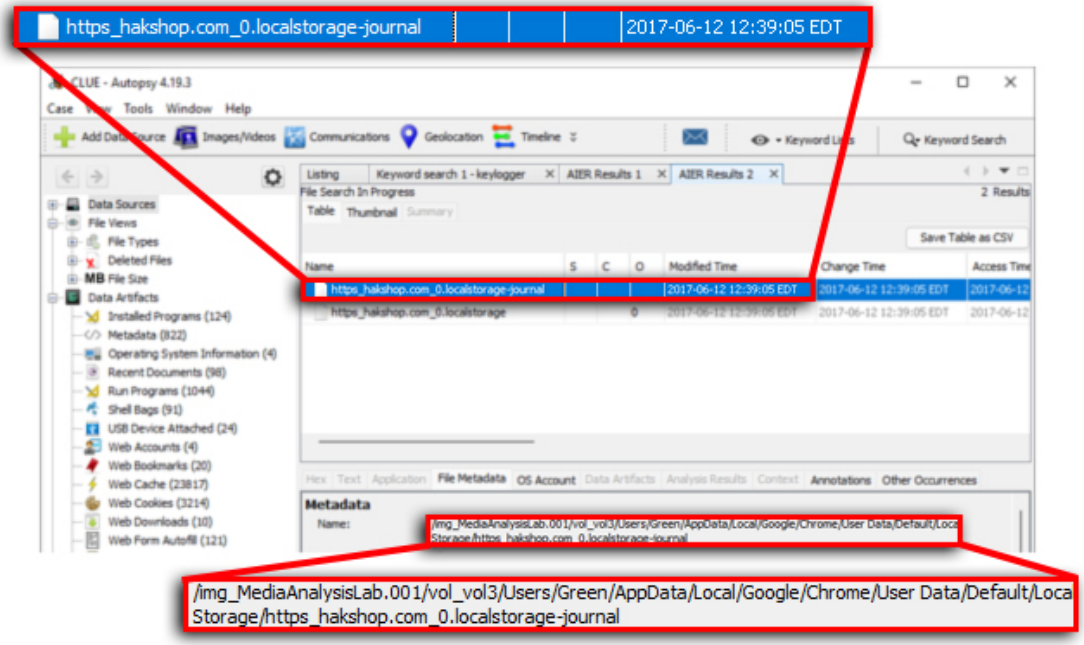

Figure 51: USB Rubber Ducky Metadata.

Visiting the web page shown in Figure 50 takes the examiner to a web based store selling hacking devices. Available for purchase at this web page is the same USB device seized at the scene. Figure 52 shows the USB device for sale on the web page visited by Mr. Green. The examiner can deduce that Mr. Green is likely to have planted the keylogger device.

Figure 52: USB Rubber Ducky [39].

Viewing the device at hakshop.com, the examiner can see the main integrated circuit on the USB thumb drive. It appears to be an Atmel chip and is visible within the USB devices section of the Autopsy user interface. The device is listed as an attached USB device. Further research on the Atmel chip indicates that it is an integrated circuit for a keyboard explaining how the USB device captures typed data. Figure 53 shows when the keylogger was connected to the system.

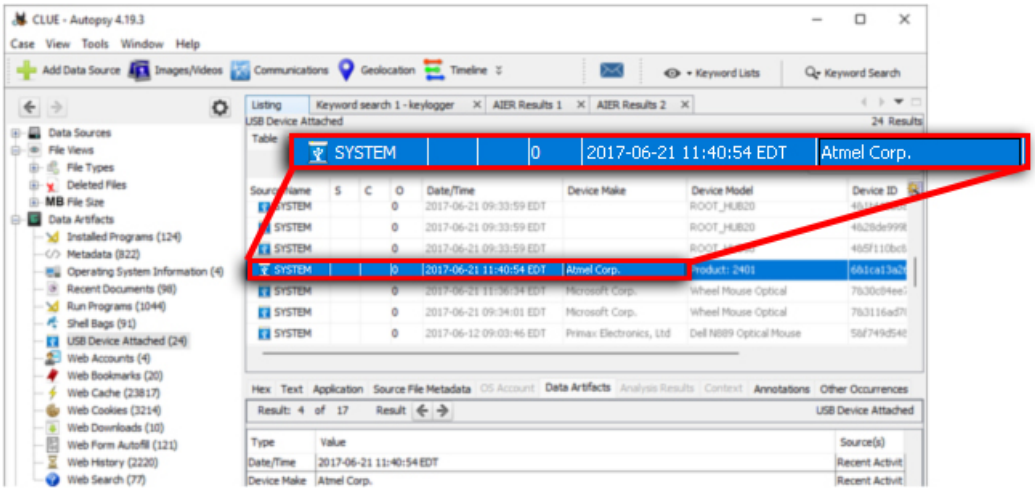

Figure 53: Keylogger USB Device Attached.

4.6.2 Scenario 3 Analysis and Results

This scenario showed another way the AIER timeline and ingest module is used to discover and reconstruct system events. The scenario provided no starting point for the examiner, however, after viewing the timeline and a keyword search, the examiner was able to discover substantial information. The Autopsy and AIER integrations enabled the examiner to reconstruct the events leading to who likely planted the keylogger. While this timeline integration did not answer all the potential questions in this scenario, it enabled the examiner to generate other leads to look into further.

4.7 Summary

This section demonstrated how the AIER timeline and ingest module helps to automate event reconstruction for the forensic examiner. Automating this part of the examiner's workflow helps alleviate some of the problems causing forensic case backlog. In the three scenarios presented in this section, the AIER ingest module pulled data from the Autopsy database and, using Plaso, pulled data from the source image. After the ingest of this data the AIER ingest module abstracted the data and stored the abstractions in a graph database. The data was retrieved and presented in the AIER timeline for use by the examiner. The examiner in the three scenarios was able to use the integrated timeline with the Autopsy user interface in various ways to reconstruct system events.

V. Conclusions

5.1 Summary

This research proposed a solution to help digital forensic examiners overcome expanding data sets and a backlog of digital forensic cases. Using automation, abstraction, and visualization techniques integrated into the forensics tool Autopsy, the workflow of the digital forensic examiner is improved. The integration allows the examiner to make use of the Autopsy Integrated Event Reconstruction (AIER) timeline to reconstruct events during a digital forensic examination.

AIER was able to automate the event reconstruction process for the forensic examiner using a novel Autopsy ingest module and abstraction and visualization techniques. Testing this process using the scenarios in Chapter IV showed that this integration improved the forensic examiner's workflow.

5.2 Limitations

The AIER process is limited by the ability of Autopsy to recognize and record artifacts accurately. The AIER ingest module relies partially on the database used by Autopsy to abstract data and display artifacts on the AIER timeline. If an artifact is not detected and recorded in the Autopsy database it will not be displayed by the AIER timeline.

Another limitation of this process is the reliability on Plaso. This timeline creation tool only works on Windows operating systems. Within this research, the Plaso timeline tool is used to parse Windows event logs. If Windows events are not required it is possible to remove the Plaso process from the AIER ingest module. This is discussed further in Section 5.3.

The Neo4j graph database presents another limitation of this process. Currently,

the AIER ingest module creates Comma Separated Value (CSV) files as an intermediate storage medium before adding that data to Neo4j. This is the most effective way to add data and run the abstractions in Neo4j. This can create a storage and processing bottleneck. Eliminating Neo4j and using a different database medium could improve these problems.

The AIER timeline and ingest module does not support multi-user cases. Each user that wants to work with the timeline will have to run the AIER ingest module individually. Section 5.3 proposes that modifications be made to allow the timeline to be accessible by multiple users.

Lastly, the integrated timeline is currently limited to five categories of abstracted data. These categories can be modified, however, additional abstraction rules will need to be created to accomplish this. The addition of more categories will increase the usefulness of the timeline integration across many more cases.

5.3 Future Work

5.3.1 User Study

A full user evaluation of this work will provide more feedback about the user experience and the impact on the examiner's workflow. AIER was evaluated based on the perspective and experience of the developer and not the end-users. A formal user study would test AIER's usefulness and the impact it has on a variety of forensic examiners. The formal user study would aid AIER's further development and allow for improvements based on the study feedback.

5.3.2 Technical Improvements

Adding more categories to the timeline allows the examiner to display more artifact types. This would increase the number of cases that would find the timeline

useful. Currently, the timeline fits a relatively small number of case profiles due to the limiting abstraction categories.

Another future area of study for this process could include adding options to the AIER ingest module. One of these options within the module could enable and disable the additional abstraction categories. This would allow the timeline to be more tailored to the case. Additionally, with Plaso disabled, the ingest module will not parse Windows event logs. This will allow the timeline to make use of any operating system that Autopsy can ingest.

Autopsy allows multiple users to work on a case at the same time. Adding functionality to allow the AIER timeline to be utilized by multiple users would be very helpful to multi-user cases. One way to solve the problems with multi-user cases is to move Autopsy to JavaServer Faces (JSF) technology. Adopting JSF moves ingest processing work done by Autopsy to the server-side instead of the client-side.

Bibliography

1. Siti Hawa Mokhtar, Gopinath Muruti, Zul Azri Ibrahim, Fiza Abdul Rahim, and Hairoladenan Kasim. A review of evidence extraction techniques in big data environment. *2018 International Conference on Smart Computing and Electronic Enterprise, ICSCEE 2018*, 2018.

2. Nicole Beebe and Jan Clark. Dealing with terabyte data sets in digital investigations. *IFIP International Federation for Information Processing*, 194:3–16, 2006.

3. Darren Quick and Kim Kwang Raymond Choo. Iot device forensics and data reduction. *IEEE Access*, 6:47566–47574, 2018.

4. Linn F Freedman and Cole Llp. C-suites : Cybercrime damages expected to reach 6 trillion by 2021 article by. pages 1–2, 2021.

5. Shams Zawoad and Ragib Hasan. Digital forensics in the age of big data: Challenges, approaches, and opportunities. *Proceedings - 2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security and 2015 IEEE 12th International Conference on Embedded Software and Systems, H*, pages 1320–1325, 2015.

6. TeelinkSheldon and ErbacherRobert F. Improving the computer forensic analysis process through visualization. *Communications of the ACM*, 49:71–75, 2 2006.

7. Simson L. Garfinkel. Digital forensics research: The next 10 years. *Digital Investigation*, 7:S64–S73, 8 2010.

8. Ovie L Carroll, Stephen K Brannon, and Thomas Song. Computer forensics: Digital forensic analysis methodology. *UNITED STATES ATTORNEYS' BULLETIN*, 56, 1 2008.

9. Daniel Schelkoph. Digital forensics event graph reconstruction. *Theses and Dissertations*, 3 2018.

10. Nikolai Adderley. Graph-based temporal analysis in digital forensics. *Theses and Dissertations*, 3 2019.

11. Yunus Yusoff, Roslan Ismail, and Zainuddin Hassan. Common phases of computer forensics investigation models. *International Journal of Computer Science and Information Technology*, 3:17–31, 6 2011.

12. Meagan B. Gallagher and John I. Thornton. Trace evidence in crime reconstruction. *Crime Reconstruction*, pages 247–297, 2011.

13. Mark Pollitt, Eoghan Casey, David-Olivier Jaquet-Chiffelle, and Pavel Gladyshev. A framework for harmonizing forensic science practices and digital/multimedia evidence. *OSAC Technical Series*, 2, 2 2019.

14. Cellebrite. Criminal investigations digital intelligence solutions. `https://cellebrite.com/en/criminal-investigations/`, 2021.

15. OpenText. Opentext encase forensic software. `https://security.opentext.com/encase-forensic`, 2021.

16. Brian Carrier. Autopsy. `https://www.sleuthkit.org/autopsy/`, 2020.

17. Brian Carrier. The sleuth kit framework: Basic framework concepts. `https://www.sleuthkit.org/sleuthkit/docs/framework-docs/basics_page.html`, 2013.

18. Brian Carrier. The sleuth kit. https://www.sleuthkit.org/sleuthkit/, 2020.

19. Brian Carrier. Autopsy user documentation: Autopsy user's guide. http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//, 2021.

20. Brian Carrier. Autopsy: Features. https://www.sleuthkit.org/autopsy/features.php, 2020.

21. Brian Carrier. Autopsy: Timeline analysis. https://www.sleuthkit.org/autopsy/timeline.php, 2020.

22. Brian Carrier. Autopsy forensic browser developer's guide and api reference. http://sleuthkit.org/autopsy/docs/api-docs/4.19.2//, 2021.

23. Stuart Card, Jock Mackinlay, and Ben Shneiderman. *Readings in Information Visualization: Using Vision To Think*. 1 1999.

24. C. Tassone, B. Martini, and Kim Kwang Raymond Choo. Forensic visualization: Survey and future research directions. *Contemporary Digital Forensic Investigations of Cloud and Mobile Applications*, pages 163–184, 2017.

25. Brian Carrier and Brian Carrier. Defining digital forensic examination and analysis tools using abstraction layers. *INTERNATIONAL JOURNAL OF DIGITAL EVIDENCE*, 1:2003, 2002.

26. Yoan Chabot, Aurélie Bertaux, Christophe Nicolle, and M-Tahar Kechadi. A complete formalized knowledge representation model for advanced digital forensics timeline analysis. *Digital Investigation*, 11:S95–S105, 2014.

27. James Okolica and Wright-Patterson Air Force Base. Temporal event abstraction and reconstruction. 12 2017.

28. Log2timeline/plaso: Super timeline all the things. https://github.com/log2timeline/plaso, 2021.

29. Elasticsearch. Elastic stack: Elasticsearch, kibana, beats, logstash. https://www.elastic.co/elastic-stack/, 2021.

30. GRANDstack. Getting started with grandstack. https://grandstack.io/, 2021.

31. Graphql — a query language for your api. https://graphql.org/, 2021.

32. Cypher query language - developer guides. https://neo4j.com/developer/cypher/, 2022.

33. React – a javascript library for building user interfaces. https://reactjs.org/, 2021.

34. Neo4j developer guides - neo4j graph database platform. https://neo4j.com/developer/, 2021.

35. Cfreds portal. https://cfreds.nist.gov/.

36. Digital corpora - scenarios. https://digitalcorpora.org/corpora/scenarios.

37. Hacking case. https://cfreds-archive.nist.gov/Hacking_Case.html.

38. Digital corpora - 2019 narcos. https://digitalcorpora.org/corpora/scenarios/2019-narcos.

39. Usb rubber ducky - hak5. https://shop.hak5.org/products/usb-rubber-ducky-deluxe.

Acronyms

AFIT Air Force Institute of Technology. 52, 66

AIER Autopsy Integrated Event Reconstruction. iv, viii, ix, 3, 4, 5, 11, 12, 13, 14, 16, 20, 21, 22, 23, 24, 25, 26, 37, 40, 41, 42, 43, 47, 48, 49, 50, 51, 52, 53, 54, 61, 62, 63, 64, 65, 66, 69, 70, 73, 74, 75, 76, 77, 1

API Application Program Interface. 4, 17, 18, 20, 21

CSCE Computer Science and Computer Engineering. 66

CSV Comma Separated Value. viii, 4, 16, 17, 24, 25, 26, 27, 30, 76

DEFT Digital Evidence and Forensic Toolkit. 11

DoD Department of Defense. 1

DOJ Department of Justice. viii, 8

FTK Forensic Tool Kit. 10

GPS Global Positioning System. 2

GUI Graphic User Interface. 12, 17

IBM International Business Machines. 10

ID Identification. 23, 27, 29, 30, 31, 32, 33, 34, 35, 38

IoT Internet of Things. 2

JSF JavaServer Faces. 77

JSON JavaScript Object Notation. 17

NIST National Institute of Standards and Technology. 52, 53, 62

OSAC Organization of Scientific Area Committees. 8

PCMCIA Personal Computer Memory Card International Association. 53

PGER Property Graph Event Reconstruction. 3, 5, 16, 17

SIFT SANS Investigative Forensic Toolkit. 11

SQL Structured Query Language. viii, 17, 25

TAIMA Temporal Analysis Integration Management Application. 3, 5, 16, 17, 18, 42

TEAR Temporal Event Abstraction and Reconstruction. 17

TSK The Sleuth Kit. 11, 12

UFED Universal Forensic Extraction Device. 10, 11

UI User Interface. 4, 5, 17, 18, 21, 40, 41, 42, 43

URL Uniform Resource Locator. 35

USAF United States Air Force. 1

USB Universal Serial Bus. x, 66, 71, 72, 73

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
24–03–2022	Master's Thesis	Sept 2020 — Mar 2022

4. TITLE AND SUBTITLE

Automated Reconstructions for
The Digital Forensic Examiner Workflow

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Montgomery, Ryan P, Capt

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-22-M-048

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Intentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

One product of a digital forensics examination is a reconstruction of events recorded in the media. A reconstruction places all of the case relevant trace into temporal, identity and associative relationships. Creating this reconstruction is a manual and time consuming process for the examiner. This thesis presents AIER. AIER integrates automation, abstraction and visualization into the Autopsy forensic software to improve the reconstruction process. The integration utilizes a custom Autopsy ingest module to extract and abstract artifact data and an interactive graph-based timeline visualization module. These improvements to the forensic examiner workflow are evaluated through a series of use cases.

15. SUBJECT TERMS

digital forensics, autopsy, timeline visualization, forensic examination

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Gilbert Peterson, AFIT/ENG
U	U	U	UU	95	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636 x4281 gilbert.peterson@afit.edu