3-24-2016

Whitelisting System State In Windows Forensic Memory Visualizations

Joshua A. Lapso

WHITELISTING SYSTEM STATE IN WINDOWS FORENSIC MEMORY VISUALIZATIONS

THESIS

Joshua A. Lapso, Capt, USAF

AFIT-ENG-MS-16-M-029

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-16-M-029

WHITELISTING SYSTEM STATE IN WINDOWS FORENSIC MEMORY VISUALIZATIONS

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

Joshua A. Lapso, B.S.C.E.

Capt, USAF

March 2016

AFIT-ENG-MS-16-M-029

WHITELISTING SYSTEM STATE IN WINDOWS FORENSIC MEMORY VISUALIZATIONS

THESIS

Joshua A. Lapso, B.S.C.E.
Capt, USAF

Committee Membership:

Dr. Gilbert L. Peterson
Chair

Dr. Barry E. Mullins
Member

Dr. Timothy H. Lacey
Member

Abstract

Examiners in the field of digital forensics regularly encounter enormous amounts of data and must identify the few artifacts of evidentiary value. The most pressing challenge these examiners face is manual reconstruction of complex datasets with both hierarchical and associative relationships. The complexity of this data requires significant knowledge, training, and experience to correctly and efficiently examine. Current methods provide primarily text-based representations or low-level visualizations, but levee the task of maintaining global context of system state on the examiner. This research presents a visualization tool that improves analysis methods through simultaneous representation of the hierarchical and associative relationships and local detailed data within a single page application. A novel whitelisting feature further improves analysis by eliminating items of little interest from view, allowing examiners to identify artifacts more quickly and accurately. Results from two pilot studies demonstrates that the visualization tool can assist examiners to more accurately and quickly identify artifacts of interest.

Acknowledgements

This is Sunday, and the question arises, what'll I start tomorrow?

- Kurt Vonnegut, Jr.

I would like to express my deepest gratitude to my wife for her unwavering support through this academic endeavour. To my faculty advisor, Dr. Gilbert Peterson, thank you for the research motivation and consistently sound advice. To Dr. Mullins and Dr. Lacey, thank you for the constant support along the way. Lastly, I would like to thank Jimmy Okolica for being an outstanding research partner, it was great working with you!

Joshua A. Lapso

Table of Contents

Page

List of Figures

List of Tables

WHITELISTING SYSTEM STATE IN WINDOWS FORENSIC MEMORY VISUALIZATIONS

I. Introduction

Modern criminal investigations frequently include evidence obtained from electronic devices such as computers, smart phones, tablets and even refrigerators. Hinshaw[4] estimates data storage is doubling every nine months, twice the rate of Moore's Law. As datasets grow with technology, the time required to analyze the data increases. Adding additional manpower is not a likely solution for reducing the temporal factor associated with data analysis[5], this is especially true in fiscally constrained environments.

Beebe and Clark [6] encourage further research in data mining such as information visualization (InfoVis), for immediate application in digital forensics discipline. This has proven successful in storage media[7], device interaction[8], memory[9] and triage[5] analysis methods. These visualization tools lead to more accurate identification of forensic artifacts by calling on the examiner's intuition and knowledge of the process. However, it still remains to be seen if forensic visualization tools provide faster results.

1.1 Research Objectives

The objectives of this research are to unite three characteristics of analysis into a single memory visualization tool. The characteristics necessary to produce a successful memory visualization tool are:

1. Examiners must maintain local and global context throughout analysis.

2. Examiners must be able to quickly connect data.

3. A forensic visualization tool cannot be divorced from low level details that must be documented in an examiner's report.

1.2 Research Hypothesis

Together, the objectives provide context throughout analysis and shrink the search space for an examiner. The hypothesis is that a visualization tool that meets these objectives will make an examiner more accurate and faster.

1.3 Research Goals

The primary goal of this research is to develop and evaluate a fully functional memory analysis tool based on Baum's[9] memory visualization proof-of-concept. The specific objectives of this research are as follows:

1. The memory visualization tool should simultaneously display hierarchical and associative relationships.

2. A novel, behavioral whitelisting function should filter processes of little interest from view.

1.4 Hypothesis Evaluation

To evaluate this hypothesis, three steps are necessary. First, extend Baum's[9] proof of concept into a fully functional tool. Second, implement a behavioral whitelisting function. Lastly, conduct pilot studies to test the tool's efficacy.

Baum's proof of concept is not scalable. For this reason, the redesigned visualization tool uses an industry standard, single-page application model with database support. A searchable table module makes low-level details available to the examiner

when needed. Additionally, rewritten sorting functions abstract the operating system and allow for future development.

The novel behavioral whitelisting feature, written as a server-side module, integrates with the database solution for speed and scalability. The whitelisting feature removes items of little interest from view, shrinking the examiners search space. This feature allows an examiner to identify anomalies more rapidly.

Pilot studies using human subjects test the research hypothesis. These studies evaluate participants completing forensic exercises using the memory visualization tool. Graded exercises establish a baseline for analysis, while post-study surveys provide qualitative data for content analysis. The themes selected for content analysis should show the memory visualization tool produces more accurate artifact identification and with the whitelisting feature, reduces time spent on a task.

1.5 Results Overview

Both pilot studies showed positive results for participants using the memory visualization tool. The first pilot study showed improved accuracy in artifact identification when participants used the visualization tool. The first pilot study did not evaluate differences in completion times. The second pilot study showed improved accuracy in artifact identification and a reduction in completion time for participants using the visualization tool. The results of these pilot studies support the hypothesis, and suggest the memory visualization tool is ready for large scale testing and evaluation.

1.6 Summary

This chapter framed the research problem by discussing issues facing modern forensic examiners and promising solutions found in information visualization. The focus and goals for this research establish key requirements going forward. Lastly,

the hypothesis and evaluation methods form the basis of this research.

Chapter two provides an overview of related work in digital forensics, information visualization, and single page web application platforms. The chapter highlights the need for visualization methods in digital forensics to better analyze enormous datasets along with four examples of forensic visualization tools. Single page web applications are discussed as platform independent frameworks for visualization tools.

Chapter three describes the key features of the memory visualization tool. It provides a system overview and demonstration of basic usage. Use cases are provided to showcase how the memory visualization tool assists an examiner in identifying user activity and malware on a system.

Chapter four describes the research methodology. It discusses the design of two pilot studies involving human subjects. Lastly, this chapter states the hypotheses for this research.

Chapter five details the results and analysis of data collected during the pilot studies. Quantitative data collected from the forensic exercises are used to validate post-study survey responses. Qualitative data collected in post-study surveys are analyzed using content analysis methods.

The final chapter draws conclusions regarding the results of this research and discusses the overall accomplishments. Future work is proposed in order to refine existing features and implement new features in the memory visualization tool.

II. Literature Review

Forensic visualization tools establish state of the art analysis techniques for digital forensic examiners. Recent successes applying visual analysis techniques to storage media[7], device interaction[8], memory[9] and triage[5] encourage further research in this discipline. This thesis focuses on reducing the temporal factor associated with forensic memory analysis and improving the accuracy of forensic artifact identification by extending a forensic memory visualization proof of concept into an operational tool.

This chapter highlights current work in the digital forensic sciences and describes how web applications, information visualization (InfoVis), and distributed database systems play a key role in digital forensics research. This chapter discusses how single page web applications run code independent of operating systems, because browsers have a built-in JavaScript engine. It presents popular InfoVis tools and details one that is based on JavaScript, which is ideal for operating system independence. Lastly, it reviews new database technologies which provide the data-mining platform required for scalable InfoVis.

2.1 Digital Forensics

Digital forensics is "the discipline that combines elements of law and computer science to collect and analyze data from computer systems, networks, wireless communications, and storage devices in a way that is admissible as evidence in a court of law."[10]. Digital forensics can be divided into four distinct disciplines: computer, network, mobile and database forensics. Kruse[11] characterizes computer forensics as the preservation, identification, extraction, documentation and interpretation of computer data. Network forensics assists with monitoring, intrusion detection and auditing by analyzing traffic from an active network[12]. Mobile forensics deal solely

with digital evidence obtained from devices such as tablets, cellphones and PDAs[13]. Database forensics study databases and the associated metadata[14].

While each discipline specializes in a unique set of device characteristics, they all tend to follow a standard model used by most investigative bodies including the United States Department of Justice (DOJ)[1][15]. Figure 1 depicts the seven step DOJ digital forensic process model described below:

- Obtaining and Imaging Forensic Data
- Forensic Request
- Preparation/Extration
- Identification
- Analysis
- Forensic Reporting
- Case Level Analysis

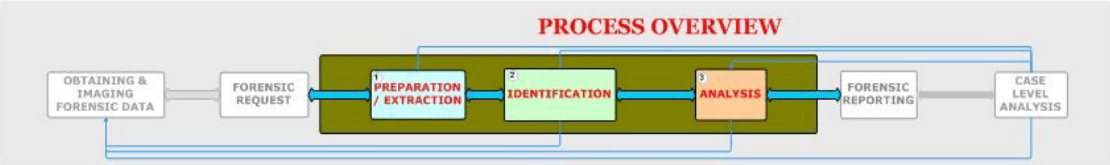


Figure 1. The Digital Forensic Model[1].

2.1.1 Sources of Forensic Evidence.

Digital evidence comes in many forms. However, there are two key characteristics of all digital evidence defined by the Scientific Working Groups on Digital Evidence and Imaging Technology (SWGDEIT)[16]. First, it is in binary form, which means it

is represented by ones and zeros and stored on some sort of electronic media. Second, to be considered evidence it must be information of probative value, meaning useful towards proving something in a court of law[16]. Any device containing any type of electronic storage has the potential to contain digital evidence. Devices include but are not limited to those listed in Table 1[15].

Table 1. Potential Sources of Digital Evidence.

Device Type	Examples
Hard Drives	Internal (SCSI, SATA, IDE) / External (USB, Firewire, Ethernet)
Removable Media	CDs, DVDs, Floppy Disks
USB Thumb Drives	Various types
Memory Cards	Secure Digital (SD) Cards, Compact Flash Cards
Handheld Devices	PDAs, Tablets, Smartphones, Media players, Gaming systems
Embedded Devices	Vehicles, Appliances, Homesecurity Systems, etc.
Peripheral Devices	Memory, Keyboard, Mouse, Hubs, Webcams, Microphones

2.1.2 Memory Forensics.

Volatile memory, most commonly reffered to as Random Access Memory (RAM), contains critical pieces of information about a system's state. This information is only available while the system is 'powered on' and is lost forever after a system is turned off[17]. There have been instances of Malicious Software (Malware) that only reside in RAM and thus would never be found on the other forms of computer storage media[18]. Also, other items of interest such as active processes, open directory and file handles, current network connections, and command history can only be found in RAM[19]. Several methods for memory acquisition, both hardware and software based, have been developed and are employed regularly by first responders.

Hardware-based acquisition methods have proved challenging. One method proposed using a PCI expansion slot to house a device that could directly access the memory bus[17]. This proof of concept was a success, however, it required that

all machines be fitted with the PCI card prior to the incident. Unfortunately, this method is too costly and impractical for large scale use[17]. Another method for hardware-based memory acquisition utilized the Firewire protocol (IEEE 1394)[20]. This approach allowed direct access to RAM, however, a compromised firewire device would also have direct access to memory[21]. Unfortunately, most modern motherboards do not contain a Firewire interface[20].

Software-based memory acquisition is currently the method of choice for incident responders. It requires careful bookkeeping, as each command entered by an examiner modifies a section of memory with the process being executed[18][19]. The most popular software tools are derivatives of the Unix-based drive copying tool, known as `dd`[22][19]. Once a raw memory image has been collected and preserved, there are several avenues for artifact extraction and identification.

The Volatility Framework[23] is a collection of open source memory analysis tools written in Python. Volatility offers "[a] single, cohesive framework" that "[r]uns on Windows, Linux, or Mac" with a focus on "forensics, incident response, and malware"[23]. Detailed information on specific usage can be found in *The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory*. Table 2 contains a list of modules provided by Volatility Framework Version 2.4[23]. A key addition to the framework is the svcscan module[23].

Table 2. Common Volatility Modules.

Module	Description
imageinfo	List what type of system your image came from
pslist	List the processes of a system
dlllist	Display a process's loaded DLLs
handles	Display the open handles in a process
connections	View the active connections
hivelist	List all subkeys in a hive
svcscan	See which services are registered on your memory image

The Compiled Memory Analysis Tool (CMAT)[24] is an open source memory anal-

ysis tool written in C++. Unlike Volatility, which converts symbol files to a format it can interpret, CMAT uses direct access to the Microsoft Symbol Server to obtain the correct symbol files. Currently, CMAT cannot access the service list from within the memory dump and responders must use the Windows "tasklist \svc" command to obtain the service state information during the incident response phase[9]. Table 3 shows CMAT output feature files.

Table 3. CMAT Output Feature Files.

Feature File	Data	Details
1	Process Information	Process IDs, Process Names, User IDs
2	Network Information	Active network connections
3	Process Loaded Modules	Loaded modules by Process ID
4	Process File Handles	Open Files by Process ID
5	Process Registry Keys	Registry Keys by Process ID
6	System Loaded Modules	System Drivers by Name

Rekall[25] is one of the most recent memory forensic suites. The Rekall Framework is forked from the Volatility Framework and transforms each module into libraries, for use with other tools. Specifically, Rekall integrates with Google Rapid Response[26] for a complete memory analysis suite from memory capture acquisition through forensic analysis. Rekall uses the same modules as seen in Table 2. Rekall moved away from Volatilities' signature scanning technique and accesses the Microsoft Symbol Server for global symbols[27].

2.1.3 Overwhelming Datasets.

As capacity increases for both volatile and non-volatile memory, the digital crime scene evidence is expanding in both volume and vivacity. Beebe and Clark[6] cite the rapidly increasing temporal factor for digital forensic analysis as a main reason to pursue improving digital investigative methods. They discussed various data mining techniques as potential methods for enhancement of forensic analysis. One of the

prospective methods for enhancing analysis is data visualization[28]. The practice of data visualization, also known as information visualization, provides analysts with an intuitive representation of data especially when dealing with complex datasets like those seen in digital forensics[29].

2.2 Information Visualization

Information Visualization (InfoVis) is "the study of transforming data, information, and knowledge into interactive visual representations"[29]. This section provides an overview of InfoVis and introduces popular visualization tools and frameworks.

2.2.1 Information Visualization Overview.

InfoVis is a rapidly growing applied science which gained appeal with the rise of big data analytics[29]. Lui, et al.[29] describe the five main modules of the InfoVis pipeline: data transformation and analysis, filtering, mapping, rendering and user input (UI) controls. The primary venues for InfoVis research are Empirical Methodologies, Interactions, Frameworks, and Applications.

Empirical methodologies consist of modeling and evaluation[29]. Modeling can be broken down into several sub-categories, but the most important to this study is visual representation models. Visual representation models address context preservation for element comparison, visual difficulties to help users interpret information being visualized, privacy preservation to protect sensitive data, and uncertainty caused by the visualization process. Evaluation mainly consists of user studies such as surveys, crowd sourcing, and laboratory studies. Lui, et al.[29] address the shortcomings of rigorous laboratory studies, specifically the lack of statistical reliabilities. Crowd sourcing typically supplements rigorous laboratory studies in order to add statistical reliability.

Interaction techniques were traditionally divided into seven categories: "select, explore, reconfigure, encode, abstract, filter, and connect." Liu, et al.[29] update this concept with two categories: windows, icons, mouse, pointer (WIMP) and post-WIMP. WIMP includes basic interactions like selection, highlighting, filter, and brushing, along with more advanced interactions such as visual comparison and faceted navigation. The goal is to allow users to specify which data they want to visualize for analysis. Post-WIMP interactions refer to those using modern devices such as stylus, pen, touch pad or motion capture interfaces.

Systems and frameworks have received much attention in recent years[29]. Frameworks in this context are high-level taxonomies and algorithms which describe data models and are more theory based, leaving the implementation open to the researcher. Effective frameworks are characterized by the visualization process as shown in Figure 2. The relative nature of uncertainty in various frameworks affects visualizations and recognition is key to good visualizations[2][30].

An application's usefulness is directly dependent upon quality data. Furthermore, proper application to a given dataset is just as important, as some visualization applications apply different emphasis and meaning to data. Key areas of research in InfoVis are static and dynamic graph visualizations, text visualization, map visualizations, and finally multivariate data visualization. Much of the new research in InfoVis has focused on visualizing multivariate data, which is data with multiple independent variables. Multivariate data visualization helps analysts identify, locate, distinguish, categorize, cluster, rank, compare, associate, or correlate the data under analysis[29].

Visualizations applicable to digital forensic investigations rely on multivariate data. Much of the multivariate visualization process is directly analogous to the forensic acquisition and analysis process[6]. As is the case with most aspects of data mining, a strong database solution is required.

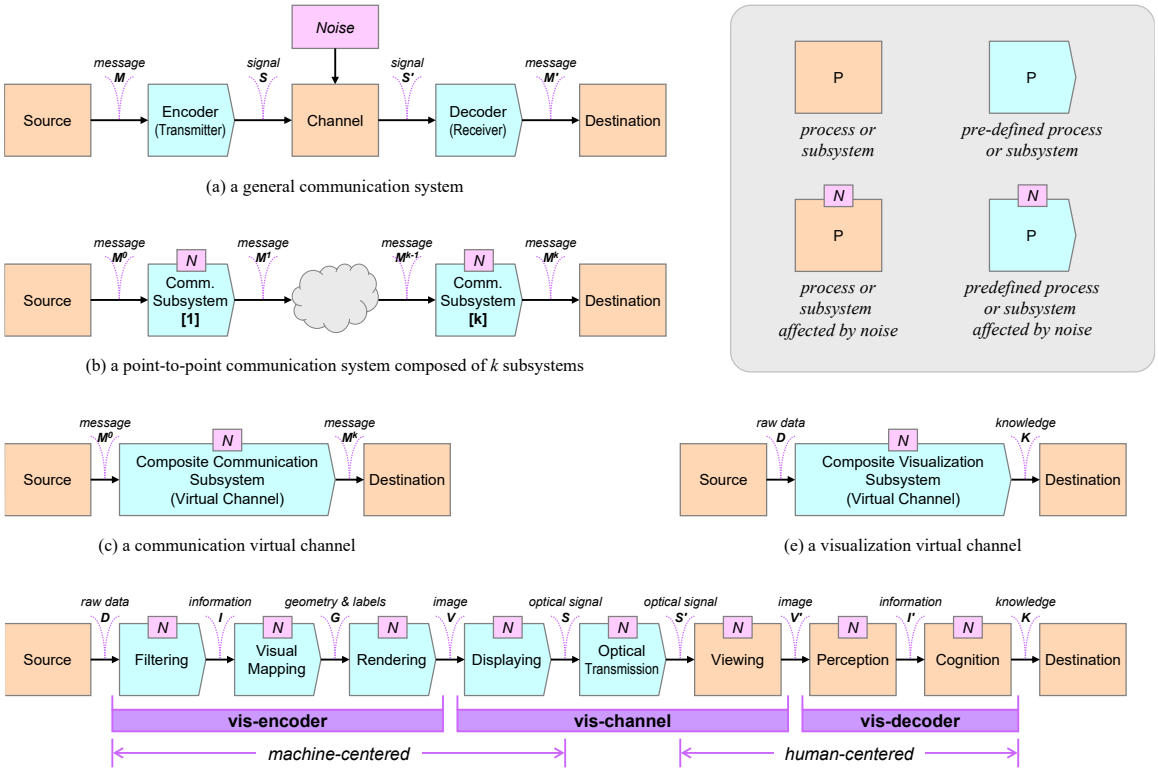


Figure 2. Models of Communication and Visualization[2].

2.2.2 Information Visualization Models.

In InfoVis, hierarchical relationships are typically represented as trees, tree-maps, adjacency diagrams or nested objects[3][31]. Figure 3 shows a traditional hierarchical model while Figure 4 shows examples of modern hierarchical visualizations. An acyclic hierarchy is well represented by all visualizations shown in Figures 3 and 4a through 4f. However, cyclic hierarchies would not be depicted well by nested objects such as Figures 4d and Figure 4f or tree-map visualizations as shown in Figure 4c.

Network visualizations show relationships between linked data. As Baum[9] discusses, representing linked data is a key component of forensic memory visualizations. Several network visualizations are shown in Figures 5a through 5c. In Figures 5a and 5c, the thicker lines identify frequent interaction, while the thin lines show infrequent interaction. Figure 5b only shows interaction exists, but does not show frequency.

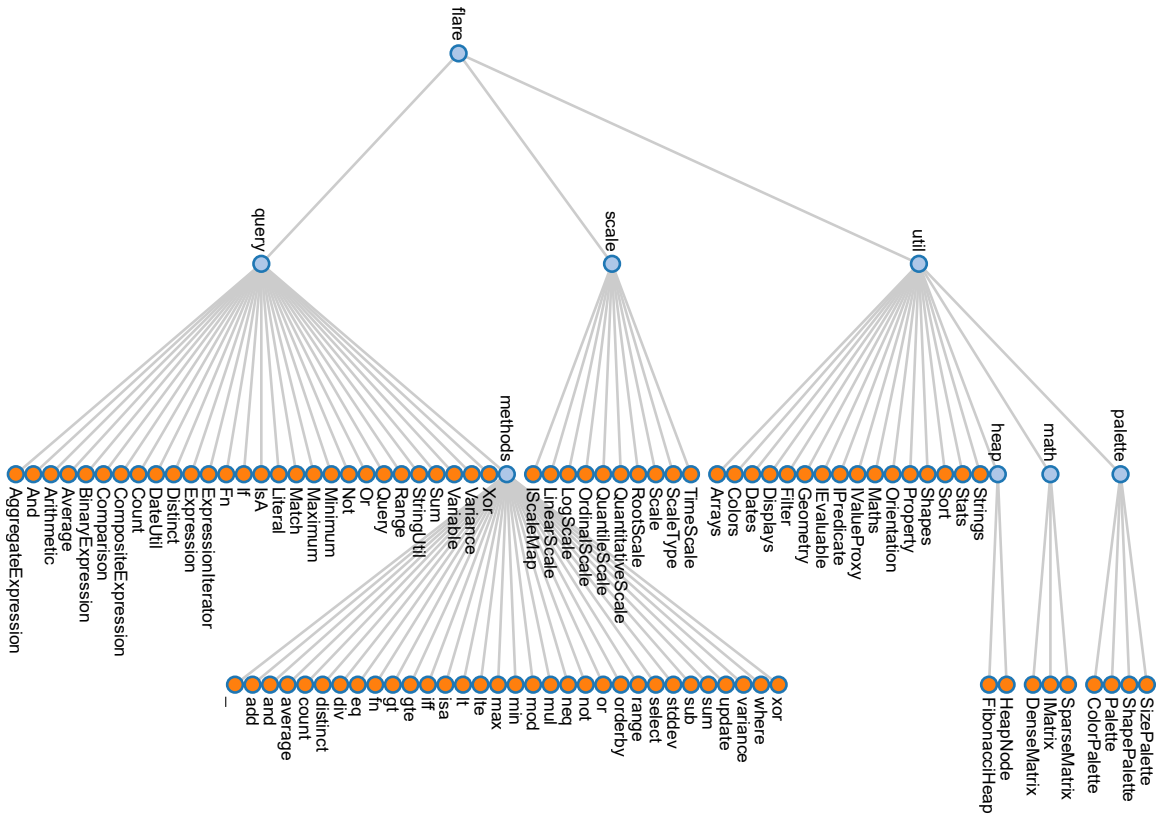
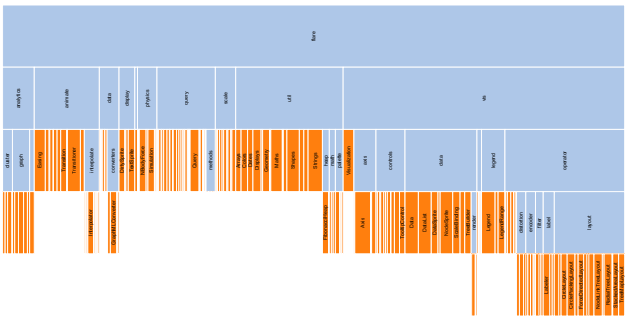
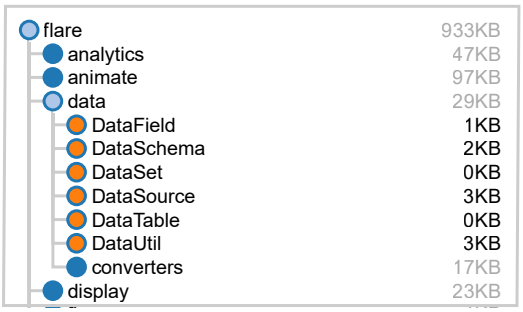


Figure 3. Traditional Hierarchical Visualization[3].

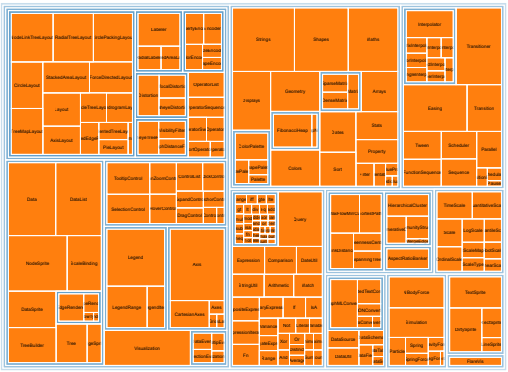
Timeline visualizations represent time-dependent data and assist the viewer in understanding chronological events[32]. Timelines are useful tools for computer forensic examiners due to the preponderance of time-stamped computer operations[7][8]. Figure 6 depicts a common timeline visualization format, using a file system's Modify, Access, Create (MAC) timestamps. Figure 6a shows the complete timeline for the contents of a directory and their associated MAC times, while Figure 6b shows the same data focused around the period of highest activity.



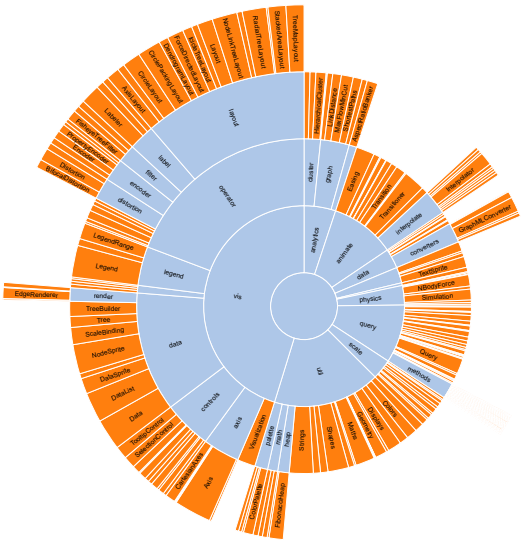
(a) Icicle Tree.



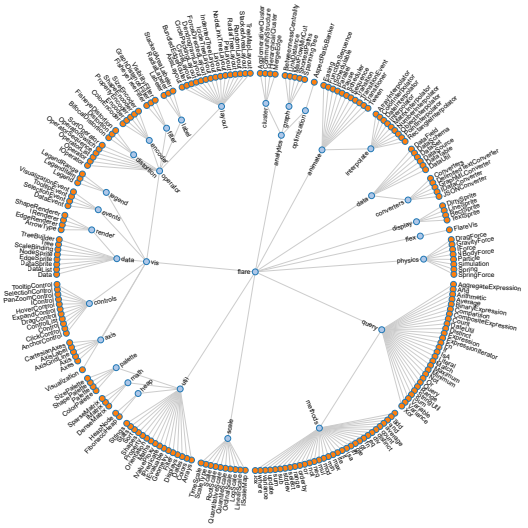
(b) Indented Tree.



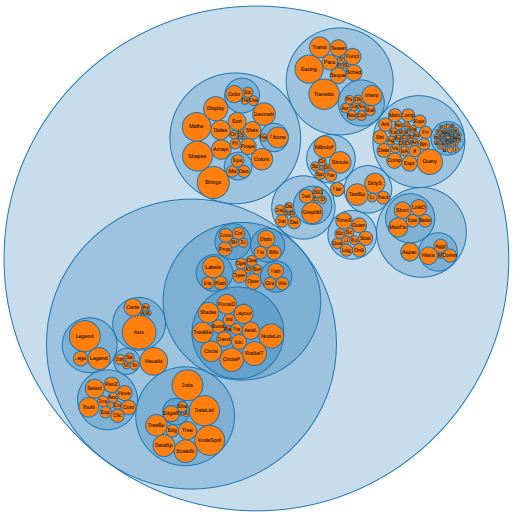
(c) Tree Map.



(d) Adjacency.



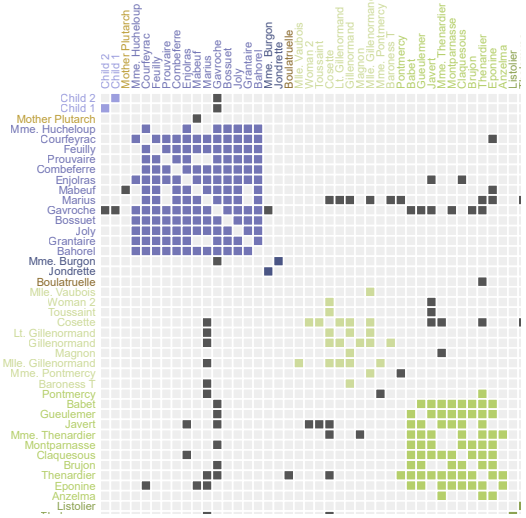
(e) Cartesian Tree.



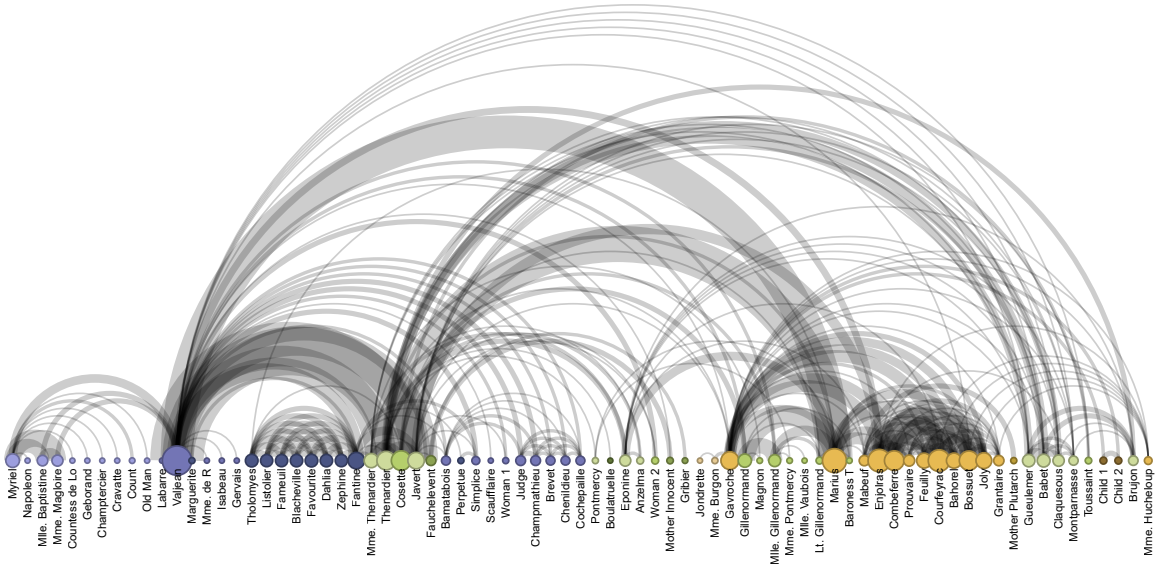
(f) Node Packing.

Figure 4. Modern Hierarchical Visualization Models[3].

(a) Force Directed.

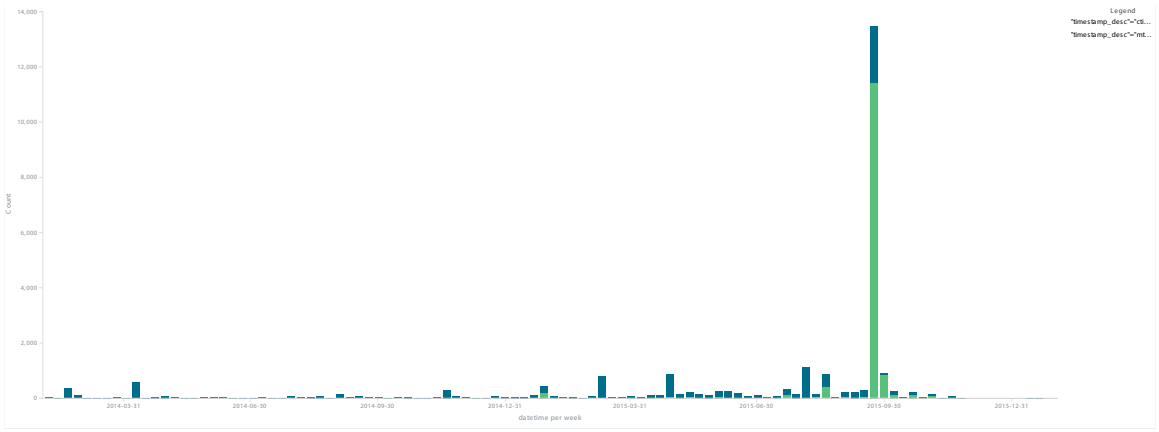


(b) Matrix.

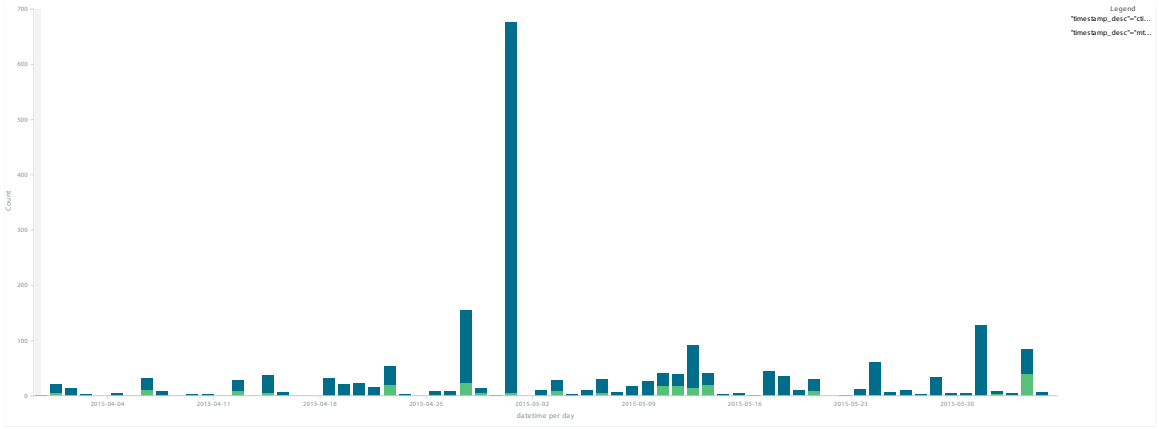


(c) Arc.

Figure 5. Network Visualization Models[3].



(a) Full Timeline For Directory MAC Times.



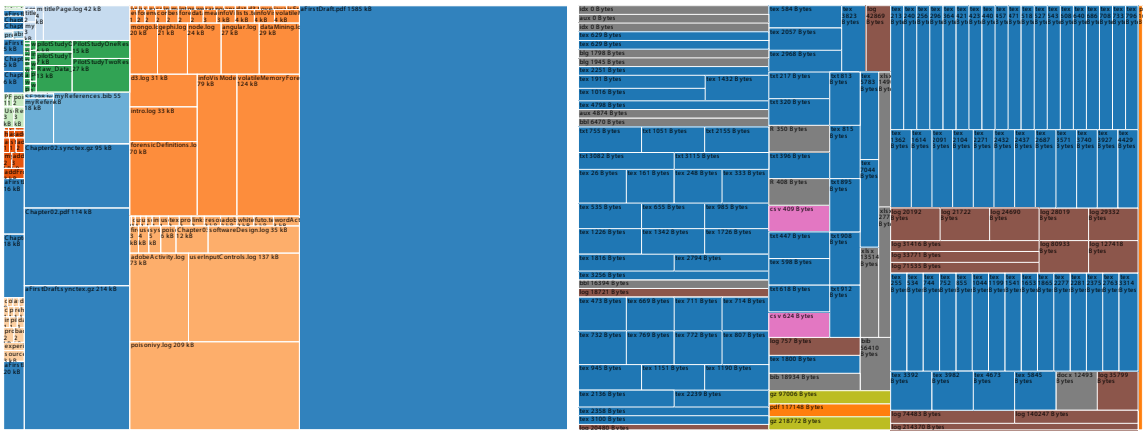
(b) Period Of Highest Activity Timeline For Directory MAC Times.

Figure 6. Timeline Visualizations.

2.2.3 Forensic Visualization Tools.

There are numerous publications on the extraction of forensic evidence through hard drive imaging[7][33], registry keys[19], and memory captures[24]. However, there are fewer publications in InfoVis, for forensic analysis.

In 2006, Teerlink and Erbacher[7] prototyped a file search tool that visualizes file size, date, and type using unique shapes and colors similar to the visualization seen in Figure 7. The result of their experiment showed that users found more files of interest with a visualization tool when compared to using traditional search strings. More compellingly, a single file, not found with search strings had a 100 percent find rate using their visualization tool. Had their experiment been a real investigation, key evidence would have been missed using traditional search strings.



(a) Treemap Directories And Files By Size. (b) Treemap Directories And Files By Type.

Figure 7. Files By Size And Extension Using Treemaps.

Osborne[8] introduced the Explore, Investigate and Correlate (EPIC) model which visualized events and interaction on and between devices. The EPIC tool provided two visualizations, one focused on inter-entity relationships and the other intra-device events such as "email, Short Message Service (SMS), Multimedia Messaging Service (MMS), phone call and website visit".

Henderson[5] explored a triage method for non-volatile storage media through a

timeline visualization of modify, access and create disk activities similar to those in Figure 6. This tool would automatically generate a second timeline visualization focused around the period of greatest activity similar to Figure 6b. The resulting visualization limited the search space to a finite timeframe for investigators.

Baum[9] applied the idea of visualizing volatile memory and showed that malicious software (malware) could be detected more accurately through data visualization. These visualization tools offer a proof of concept to the forensic community, but come with restrictions normally associated with prototypes such as limited flexibility or scalability.

These visualization tools highlight the benefits of intuitive representations of large, interrelated datasets. However, there is still a need for data filtering to focus the scope of forensic analysts. The most obvious method for filtering data during static analysis is whitelisting. Whitelisting effectively skips over "known good" files and applications in order to shrink the search space. Traditional whitelisting methods, such as hashing and signature comparison, work well for file storage media such as hard drives, flash memory, and magnetic tape, but do not directly translate to system state information stored in volatile memory[34].

2.2.4 Information Visualization Tools.

InfoVis tools vary drastically. They range from stand-alone tools to web-based development libraries. Each vary in the level of knowledge required, difficulty to build, and flexibility of visual outputs. Some are intended for one-and-done static representation of data, while others are dynamic and interactive [35]. Three of the most popular, interactive visualization tools were selected for further examination.

2.2.4.1 Data Driven Documents JavaScript Library (D3).

The Data Driven Documents (D3) JavaScript library is a powerful Infovis tool that employs direct inspection and manipulation of the World Wide Web Consortium (W3C) standard document object model (DOM). The DOM is the underlying representation utilized by HTML, XML, JavaScript and others. It exposes the hierarchical content (i.e. elements) of a web page for reference or manipulation. This direct exposure allows for both partial scene graph modifications and transparent debugging which are typically made unavailable by toolkit specific scene graph abstractions[36]. D3 utilizes scalable vector graphics (SVG) to create a wide array of dynamic visualizations [35].

2.2.4.2 Raphael.

Raphael brands itself as a "small JavaScript library that should simplify your work with vector graphics on the web" [37]. Like D3, Raphael employs SVG elements for its web-based visualizations. Users of Raphael can manipulate basic graphics primitives as well as implement event listeners for interactive visualizations [35]. However, due to a lack of abstractions and referenceable scene graph, complex visualizations become extremely tedious to build and even more difficult to debug [36].

2.2.4.3 Gephi.

Gephi is an open source toolkit for graph and network visualization and analysis [38]. Unlike the web-based visualization tools, Gephi is powered by an ad-hoc OpenGL engine making it extremely fast and capable of handling daunting datasets [39]. Gephi can process dynamic and interactive visualizations which are exportable to SVG objects [38], however, this feature is not accomplished on-the-fly and thus is not feasible in a web-based tool.

2.3 Database Frameworks

This section discusses current research in database technology, explores the benefits and drawbacks of Structured Query Language (SQL), Not Only SQL (NoSQL) and NewSQL, and presents an optimal solution for specialized data-mining platforms.

2.3.1 Comparing SQL To NoSQL.

Data mining and big data analytics are driving an evolving need for storing and querying large datasets. Much of the time data is unformatted or only semi-formatted. These models do not fit the standard relational databases such as SQL. NoSQL database systems such as MongoDB or Cassandra can solve the problem. MongoDB and its native MapReduce implementation allows parallel processing for large repositories[40].

Relational databases such as SQL follow the classic Atomic, Consistent, Isolation, Durable (ACID) model. This model is difficult to implement in distributed environments because of the strict consistency requirement. SQL has been in existence for much longer than NoSQL. As a result, many large organizations are heavily dependent upon SQL and could not migrate to NoSQL without a complete system overhaul[40]. Lastly, SQL is supported by practically every programming language.

NoSQL grew up alongside the internet giants, operating distributed systems with unstructured data. There are three NoSQL models: key-value, column oriented, and document oriented. These models employ the Basically Available, Soft state, Eventual consistancy (BASE) paradigm along with the Consistency, Availability, and Partition tolerance (CAP) theorem. The two main strengths of NoSQL databases are scalability and a schema-less design. For scalability, NoSQL has better cross-node operation than SQL. It also adapts better to large volumes of data in the distributed environment[40]. In terms of a schema-less design, the user is not required

to think about database evolution and thus makes updating easy. Currently, relational databases contain obstacles to adhoc upgrades or schema changes and as data evolves, it must be modeled to the database schema. In schema-less design, the user defined model evolves with the data.

MongoDB's [40] strength is read-intensive operations such as data mining. It has a strong open-source community and is considered an alternative to Hadoop. MongoDB uses the concept of database sharding, which is a horizontal partitioning of the data stores many times across multiple systems. More precisely, MongoDB implements auto-sharding which load balances systems automatically when an imbalance is detected. Each mongod is a Mongo server instance and shards are collections of replicas. However, MongoDB's scalability comes from the native MapReduce implementation.

2.3.2 Sacrificing Consistency or Availability.

Abadi[41] argues the CAP Theorem has a more limited impact on modern distributed database systems (DDBS) than often assumed by researchers. The trade-off between consistency and latency is more dominant and should be factored in more heavily. He claims that CAP was misunderstood and misapplied which led to unnecessary restrictions on DDBS design. He notes that CAP is intended for dealing with failures and not meant to limit capabilities during a system's normal operations. Several modern systems, in his opinion, have been incorrectly influenced by these trade-offs. Abadi claims unifying the two sets of latencies under a theorem he calls PACELC (If P, then A OR C; Else C OR L) would better serve DDBS designers[41].

Latency and availability are closely tied as a system that is too latent becomes essentially unavailable[41]. For this reason replication is required both to provide availability and low latency. Replication immediately incurs a trade-off. Alternatives

for replication are: send to all, send to master, or send to one. Sending to all incurs latency due to preprocessing required to avoid replica divergence. Send to master realizes latency in three forms: synchronous - slowest entity, asynchronous - propagation or overloaded master, or hybrid - quorum agreement delay. Send to one must overcome latency induced by preprocessing and quorum agreement.

Under Abadi's [41] PACELC each NoSQL DDBS can be classified. Dynamo, Cassandra, and Riak as PA/EL systems, meaning they choose availability over consistency and latency over consistency. VoltDB/H-Store and Mega -store are PC, they always choose consistency. MongoDB is said to be PA/EC, it chooses availability in failure, but consistency during normal operations. Lastly, PNUTS is PC/EL, consistency during failure or latency otherwise. The implementation of PACELC will not solve all the problems for DDBS. There will always be trade-offs due to the inherent nature of distributed systems. However, incorporating the latency/consistency trade-offs early in the design phase is beneficial to all.

2.3.3 NoSQL Database Assessment.

The emergence of new NoSQL paradigms has solved problems plaguing relational database systems such as the ineffective big data storage and processing or inefficient transactions and join operations. Further evaluation and baselining is needed for existing NoSQL systems. Abramova and Bernardino [42] perform a side-by-side comparison of MongoDB and Cassandra, two of the leading open-source NoSQL DDBS.

MongoDB is a document-oriented NoSQL database system (briefly introduced in section 2.3.1), where documents are the smallest unit. Documents are stored in collections and collections in databases. MongoDB uses binary JavaScript object notation (BSON). The key characteristics of MongoDB are its durability and consistency. MongoDB uses a master/slave replication implementation to provide durability of

data. Only masters can write, therefore slaves serve as read-only backups. If the master goes down, the most updated slave takes over. Since MongoDB version 2.2, the system uses locks to ensure consistent data. The main similarity that MongoDB shares with relational database systems are: create, read, update, delete (CRUD) operations[42].

Cassandra [42] is a member of the column-oriented family of NoSQL databases. Cassandra uses columns and rows similar to those of relational database systems. However, Cassandra can handle both structured and unstructured data. The key characteristics of Cassandra are durability and availability. Cassandra implements a peer-to-peer model for replication to provide durability. Cassandra can operate in both synchronous and asynchronous replication modes. Abramova and Bernadino[42] highlight Cassandra's indexing ability as a key feature. This feature contributes to the overall speed and availability of Cassandra, but comes with considerable overhead as the number of nodes increase.

In digital forensic analysis, consistency is more critical than speed. While Cassandra outperforms MongoDB in scalability, MongoDB is still highly scalable and much more consistent. MongoDB's auto-sharing and native MapReduce implementation makes for a strong data mining platform[42][40]. This will be key to implementing a sharded whitelisting database inside a visualization tool.

2.4 Web Application Platforms

Web applications are a popular avenue for developing system agnostic tools, due in large part the the Representational State Transfer (REST) architectual style. Fielding [43] introduces REST as a "coordinated set of architectural constraints that attempts to minimize latency and network communication, while at the same time maximizing the independence and scalability of component implementations". JavaScript and its

associated “Full Stack” implementations provide quality fast prototyping solutions.

2.4.1 An Introduction To JavaScript.

Javascript is found in practically every mainstream web browser. It is a functional language often refered to as “Lisp in C’s clothing”[44]. The European Computer Manufacturers Association (ECMA) specified ECMAScript (now JavaScript) was the first lambda language to achieve wide-spread acceptance and eventually beat out Java as the language of the web [44]. JavaScript provides a superb platform for system agnostic web applications. Furthermore, JavaScript is asynchronous allowing for concurrent operations through its eventing architecture rather than multi-threading. It relies on callback functions for data integrity and reduces the resource overhead associated with multi-threading [45].

2.4.2 JavaScript Frameworks.

In the early days of web development, applications were heavily coupled to a back-end server. There was little worry about styling or DOM manipulation. Those days have quickly passed and given way to new frameworks targeting the client-side browser. A few of the numerous frameworks heavily relied on by present day developers include jQuery[46], AngularJS[47] and Ember[46]. They each provide different methods for achieving high-performing, mobile and scalable user interfaces across multiple web browsers[46]. These Model-View Controllers (MVC) have quickly carved out a niche in various corners of the web application market. However, for the all purpose single page web application, the MEAN Stack provides a powerful front-to-back experience [48].

2.4.3 The MEAN Stack.

The MEAN Stack is just one of the combinations of JavaScript technologies dubbed "Full Stack" web applications. MEAN is an acronym stemming from the names of tools comprising it, MongoDB, ExpressJS, AngularJS and Node. Each provide a specific set of functionality to a web application. AngularJs provides a sleek front-end browser experience to the client. MongoDB provides fast NoSQL data storage. On the back-end, Node and ExpressJS provide an asynchronous web server with dynamic routing for database queries [48].

In the context of the MEAN Stack, Mongo [48] provides the back-end database for a web application. It stores data in web friendly BSON format, or binary JSON. Furthermore, since MongoDB is a schemaless system, it adapts to each application and when coupled with ExpressJS and Node.js there is almost seamless integration with the web environment.

ExpressJS[48] is a web framework package for Node. ExpressJS organizes the server side into the MVC architecture and provides routing features. The idea behind ExpressJS is that it should be a "fast, unopinionated, minimalist web framework." Essentially providing a robust set of features without obscuring Node.js [49]. The main attraction of ExpressJS is its built in network middleware, for lack of a better term [50].

AngularJS[47] is a popular front-end library for web applications. AngularJS provides a declarative programming environment for developing user interfaces over HTML. Google [47] labels this project the "superheroic JavaScript MVW framework", where MVW stands for Model-View-Whatever. This is because unlike jQuery, Angular defines the DOM prior to run-time. AngularJS extends HTML directives with its own declarations. It is written much like HTML with additional features and controlled through a JavaScript controller function. Lastly, AngularJS provides au-

tomatic data binding through its $scope variable which makes DOM manipulation as simple as variable assignment and mutation [48].

Node.js or Node[48] is a server-side implementation of JavaScript targeted at supporting long running server processes. Node is simply a platform that allows Javascript to run outside of a browser. In contrast to other high-level languages such as Java, C# and numerous other scripting languages, Node (and JavaScript as a whole) does not utilize multi-threading to implement concurrent operations [48]. Instead, it uses an asynchronous I/O event-driven model eliminating costly overhead and bookkeeping. Essentially, Node is a JavaScript engine embedded within a single-threaded daemon [51].

2.5 Summary

This chapter discusses the current state of the computer forensics discipline. A discussion of new database, data-mining and InfoVis technologies set a platform for exploration of new analysis methods. Lastly, a survey of current web application libraries establishes initial design considerations for new tools.

Several visualization methods found practical application in various forensic disciplines. Some of the methods focus on visually differentiating files by types or size. Other methods aim to narrow the search space or reduce the load of monotonous tasks for investigators. Each of the forensic visualization tools discussed shows the potential for InfoVis techniques in the fields of digital forensics.

III. Memory Forensics Visualization

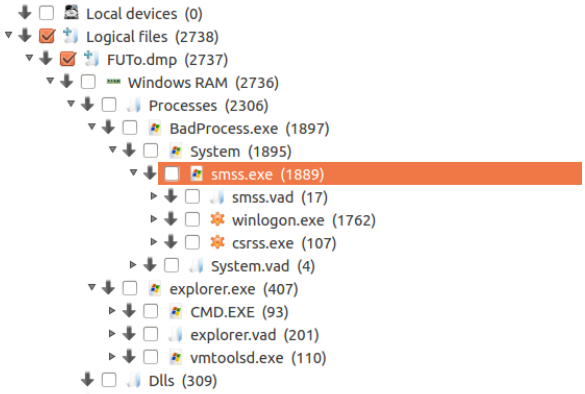
Modern incident response tools leave forensic examiners with an enormous collection of data and the daunting task of locating useful artifacts. This research seeks to improve speed and accuracy of artifact identification during the analysis process. Using a combination of InfoVis methods, the memory visualization tool provides global and local views of the memory capture data to the examiner in a single visualization structure. The examiner is then free to interact with, rather than reconstructing the data and apply intuition to the analysis processes. In addition, through a novel white-listing process, the memory visualization tool filters items of little interest from view, effectively shrinking the search space.

This section details the system overview, user interface client, display characteristics, and the white-listing process employed by the Memory Visualization Tool. This chapter concludes with two use cases detailing identification of user activity and malware. The use cases walk through three memory images, one known to be free of malware, showing user activity on a system, one containing an instance of the rootkit `FUTo` and the last exploited with the remote administration tool `Poison Ivy`.

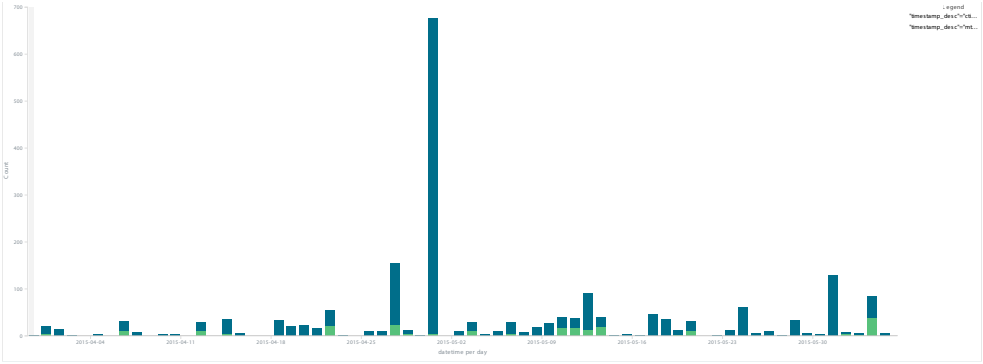
3.1 System Overview

Computer system forensic examiners and incident response teams work under various prescribed timelines derived from federal, state and private regulations[52]. Evolving technologies, most recently the arrival of vertical negative-AND (NAND) structures[53], drastically increase available storage and continue to put examiners behind the curve as existing timelines do not consider rapid leaps in future technology. The existing tools and methods provided to these examiners often generate ever-growing sets of data visualized as text or simple visualizations such as the trees and

histograms shown in Figure 8. Figure 8a shows a graphical process tree reconstruction from the Digital Forensics Framework (DFF)[54]. Figure 8b presents a histogram of MAC times created using Plaso (log2timeline and psort), Elasticsearch, Logstash, and Kibana (Plaso-ELK)[55]. These products provide a local view of data, but lack a global context beneficial to the examiner. This levies the task upon the examiner to manually or internally connect the data in order to make it useful.



(a) DFF Graphical Process Tree Reconstruction.



(b) Plaso-ELK Timeline For Directory MAC Times.

Figure 8. Simple Visualizations.

Data sources in forensic examinations can have both hierarchical and associative relationships. Hierarchical relationships are seen in system structures such as the process tree, logical file system directory structure and registry (or equivalent configuration files) structure. Associative relationships are found between processes and their open network connections, file/registry handles, system modules, and/or

services. In general, tree visualizations represent hierarchical data, while network visualizations depict connectedness. The problem lies in that neither visualization method can simultaneously represent both types of data.

The memory visualization tool presents both types of data simultaneously. This hybrid visualization method uses three types of visualizations, as shown in Figure 9, on a single canvas. In the center of the visualization, nested circles represent the system's process tree, a hierarchical structure. Around the perimeter, a donut chart (i.e., a modified pie chart) represents system resources (e.g., network connections, file/registry handles, system modules, and services). System resources are hierarchical in nature, but at an arbitrary level are resources nodes with an associative relationship to nodes in the process tree. Associative relationships between process nodes and systems resource nodes are shown using lines (i.e., edges) as in standard network diagrams.

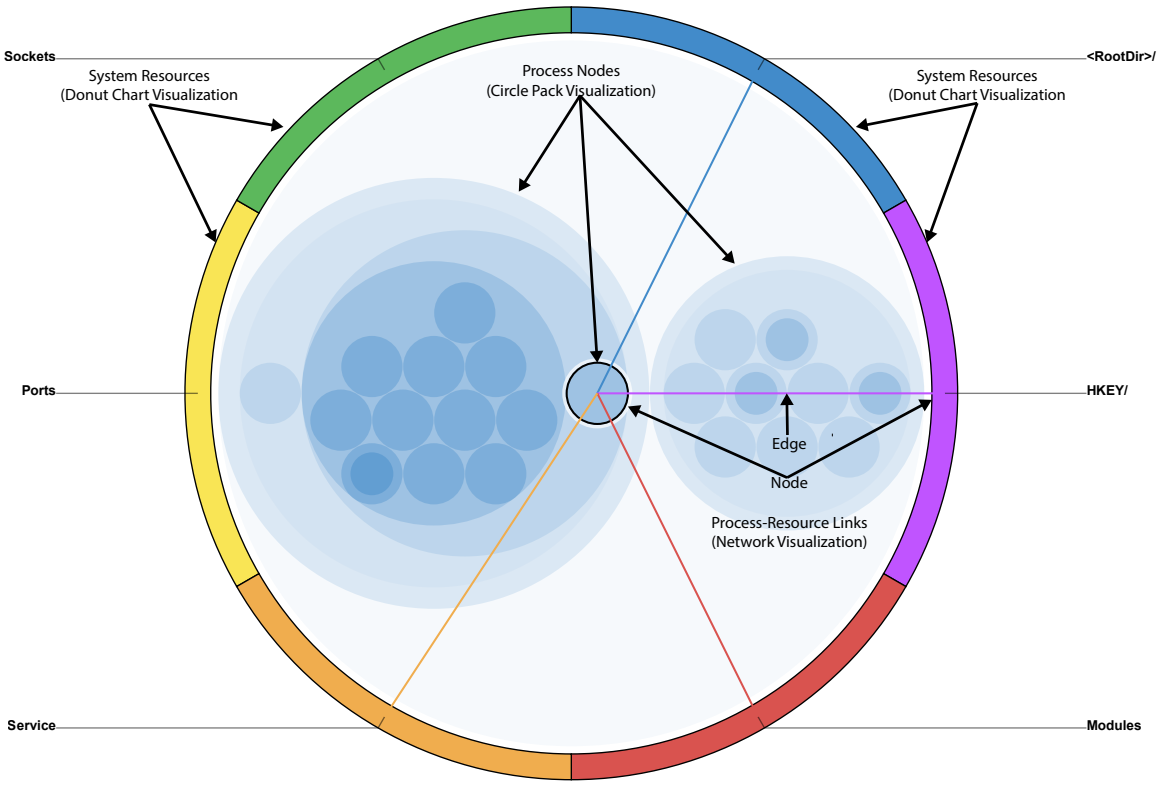


Figure 9. Overview Of Hybrid Visualization Components.

The goal of the memory visualization tool is to provide a tool that helps examiners perform a more accurate analysis in less time than current methods and tools. By providing both global and local views of the data, the examiner is freed from the task of determining connectedness of data, which may not be of interest, and can focus on applying knowledge and intuition to the analysis process. Furthermore, the global view provides the examiner with the proverbial "Big Picture" as they step further into the local views to get the descriptive raw data. Additionally, the visual representation of data allows novice examiners to better understand and explore the system under analysis, freeing up expert analysts for tasks more suited to their expertise.

In addition to these visualization techniques, the memory visualization tool applies a novel whitelisting process with the goal of shrinking the search space by eliminating items of little interest from view. The whitelisting process works off the premise that much of the data found on a live computer system is consistent and repetitive between instances of the same operating system. The tool filters consistent data from view based on a confidence percentage asserted by the examiner. The interactive nature of the tool allows the examiner to identify items of interest within a few clicks of the mouse rather than riffling through pages of text files.

Baum's[9] proof of concept provided many key inspirations, but ultimately could not refactor on a dynamic level. The tool required a ground up build supporting modularity and scalability. Additionally, new data structures written in JavaScript Object Notation (JSON) enabled use of built-in JavaScript functions. These considerations yielded an all new, fully functional memory visualization tool.

Baum's[9] proof of concept did not use dynamic functions for determining hierarchical relationships. For example, if a user changed the logical directory file structure from the default settings, the file handles would not appear in the tool. Also, if a process ID (PID) was lower than its parent process ID (PPID), which can happen

when a PID is recycled, the tool would not correctly visualization the process tree. Developing new dynamic functions independent of the operating system, solved these issues.

The new memory visualization tool, completely rewritten in the MEAN Stack framework, is comprised of server-side Javascript using the NodeJS Engine and client-side Javascript using AngularJS running in a web browser as depicted in Figure 10. ExpressJS provides the routing interface between AngularJS and NodeJS. Mongoose connects AngularJS and NodeJS to the MongoDB database for find, update, remove, and insert operations. The whitelist module is a C++ node module that receives commands from the NodeJS Server and has direct access to both the memory image and the whitelist databases. Memory image feature files are uploaded to the NodeJS server from the client workstation and then imported to the memory image database.

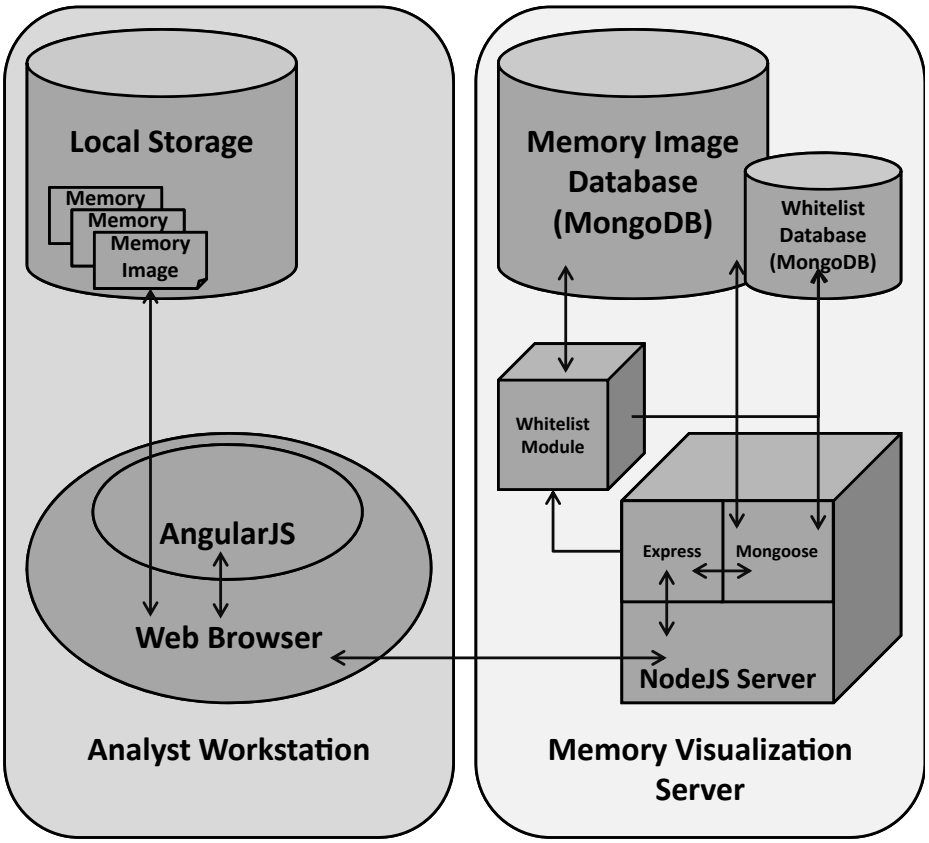


Figure 10. Memory Visualization Tool Diagram.

3.2 User Interface Client

Figure 11 highlights the three main components of the memory visualization tool's user interface. The 'Image Select' and 'Functional Buttons' components control initialization of the 'Interactive Visualization' and toggle features within the 'Interactive Visualization' and 'Raw Text View' components. The 'Raw Text View' component displays raw text data from the source database in a sortable as well as searchable table. The majority of this section focuses on the 'Interactive Visualization' component, which is the primary InfoVis development in this tool.

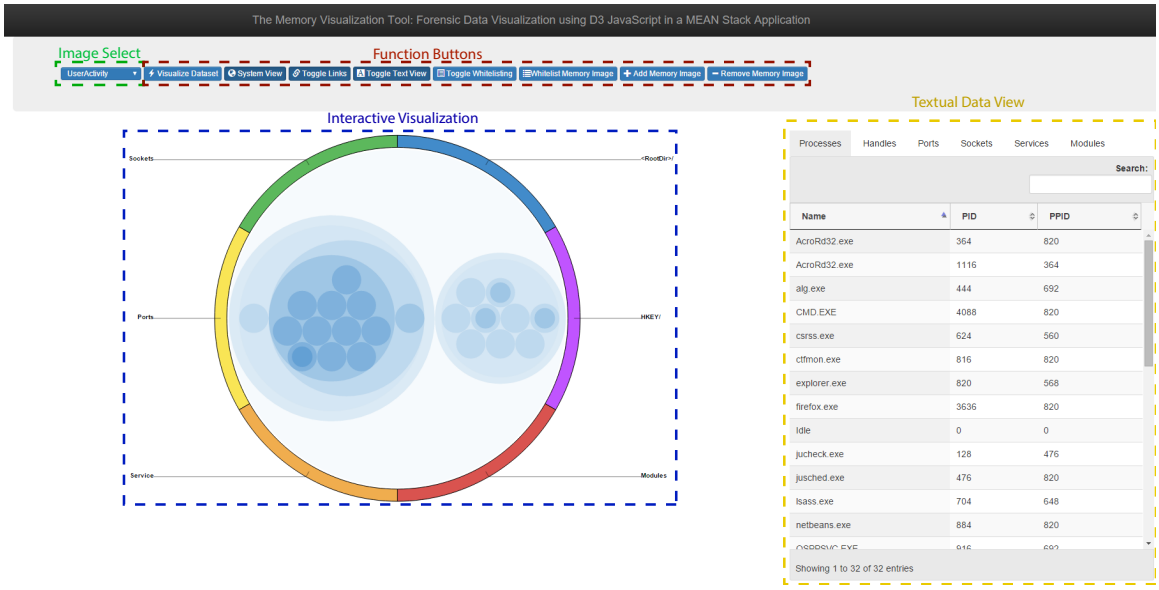


Figure 11. User Interface Orientation.

3.2.1 Image Select and Function Buttons.

Figure 12 shows the nine basic function buttons. These buttons are used to initialize the visualization tool, add or remove datasets from the database, and toggle features on or off during analysis. Additional user input controls are attached to components of the visualization. These components are discussed later in this section.

Table 4 describes the function of each button. Some buttons open an additional

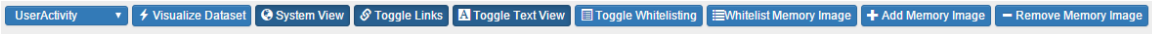


Figure 12. Image Select and Function Buttons.

control window via modals. Modal windows allow the visualization to retain state while adding additional hypertext mark-up language (HTML) views and function scope.

Table 4. Function Button Descriptions.

Button	Action
Image Select Menu	A list of memory images currently loaded in the image database.
Visualize Dataset	Initializes the visualization tool with selected image.
System View	Resets the visualization tool to the initial global view.
Toggle Links	Enables or disables process to resource link display.
Toggle Text View	Enables or disables raw data display using datatables plug-in.
Toggle Whitelisting	Opens Whitelisting precision select modal.
Whitelist Memory Image	Opens Whitelisting image select modal.
Add Memory Image	Opens Memory image upload and import select modal.
Remove Memory Image	Opens Memory image remove select modal.

The examiner inputs a desired percentage for whitelisting between 0 and 100 through the control modal shown in Figure 13. The percentage represents the lowest risk threshold the examiner is willing to accept. The whitelisting algorithm is discussed later in this chapter. An entry of 100% will mark process nodes as "likely safe" only if it appears in every system in the whitelist database. Likewise, an entry of 80% will mark processes as "likely safe" if it appears the same way in at least 80% of the systems.

To add a new memory image to the database, the examiner uploads a directory containing the memory feature files. The directory is selected for upload using the modal in Figure 14. Prior to database import, the examiner can name the memory image or leave the default time-stamped image name. This image is only added to the working database and is not automatically added to the whitelisting database.

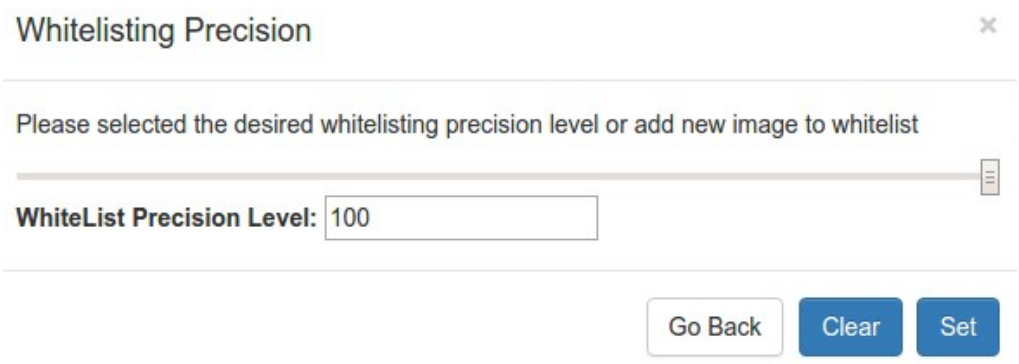


Figure 13. Set Whitelist Percentage Modal.

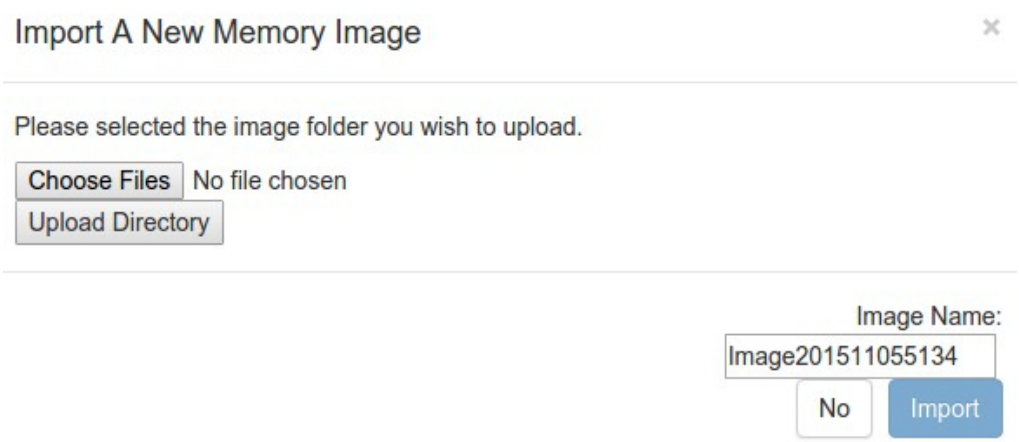


Figure 14. Add Memory Image Modal.

The examiner can remove an image from the working database using the control window shown in Figure 15. The image is not removed from the whitelisting database if it had been used for whitelisting.

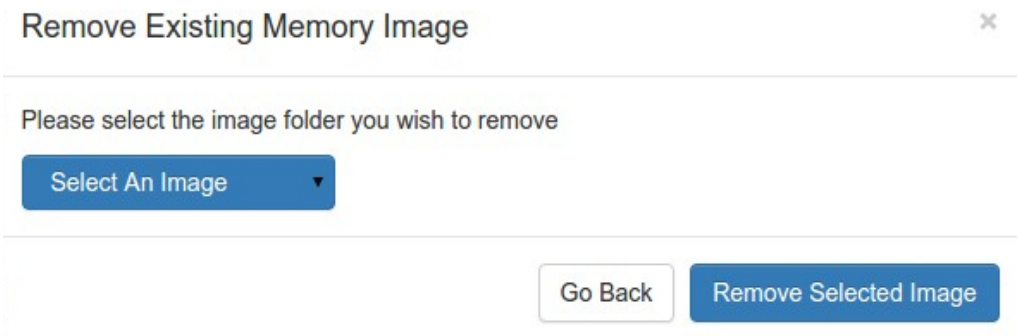


Figure 15. Remove Memory Image Modal.

If an image is determined free of malicious code, an examiner processes it through the whitelist module via the modal in Figure 16.

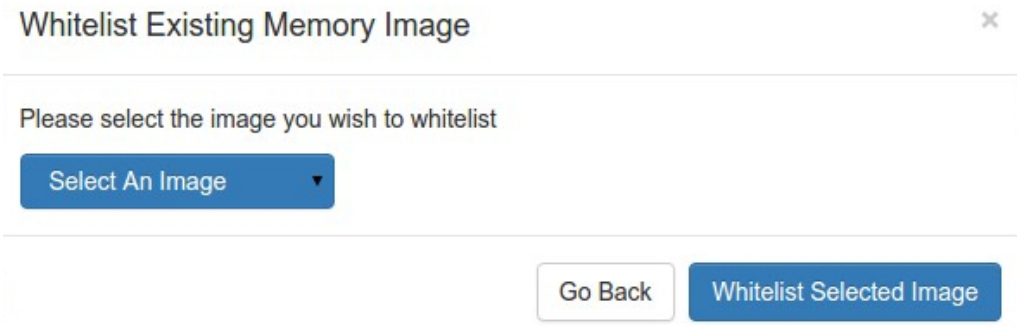


Figure 16. Whitelist Memory Image Modal.

3.2.2 Process Nodes.

Circle packing provides an intuitive representation of the process tree in a given memory image. This differs from a standard tree diagram in that relationships are implied by spacial position rather than lines, displaying greater amounts of data in a smaller space. Processes are nested inside of their parent process with the root node being Microsoft Window's `System Idle Process`. Figure 17 shows the process state of a "clean" Microsoft Windows XP SP3 system. The labels identify system processes (those initiated by the operating system) and user processes (those initiated by a user).

Process nodes offer two additional user controls. Holding a mouse curser over a node circle initiates tooltip window that shows the process name and process ID (PID) as shown in Figure 18. Process nodes also accept a mouse click event which highlights the node clicked as seen in Figure 19 and set that node into scope for the process to resource links which is discussed later in this section.

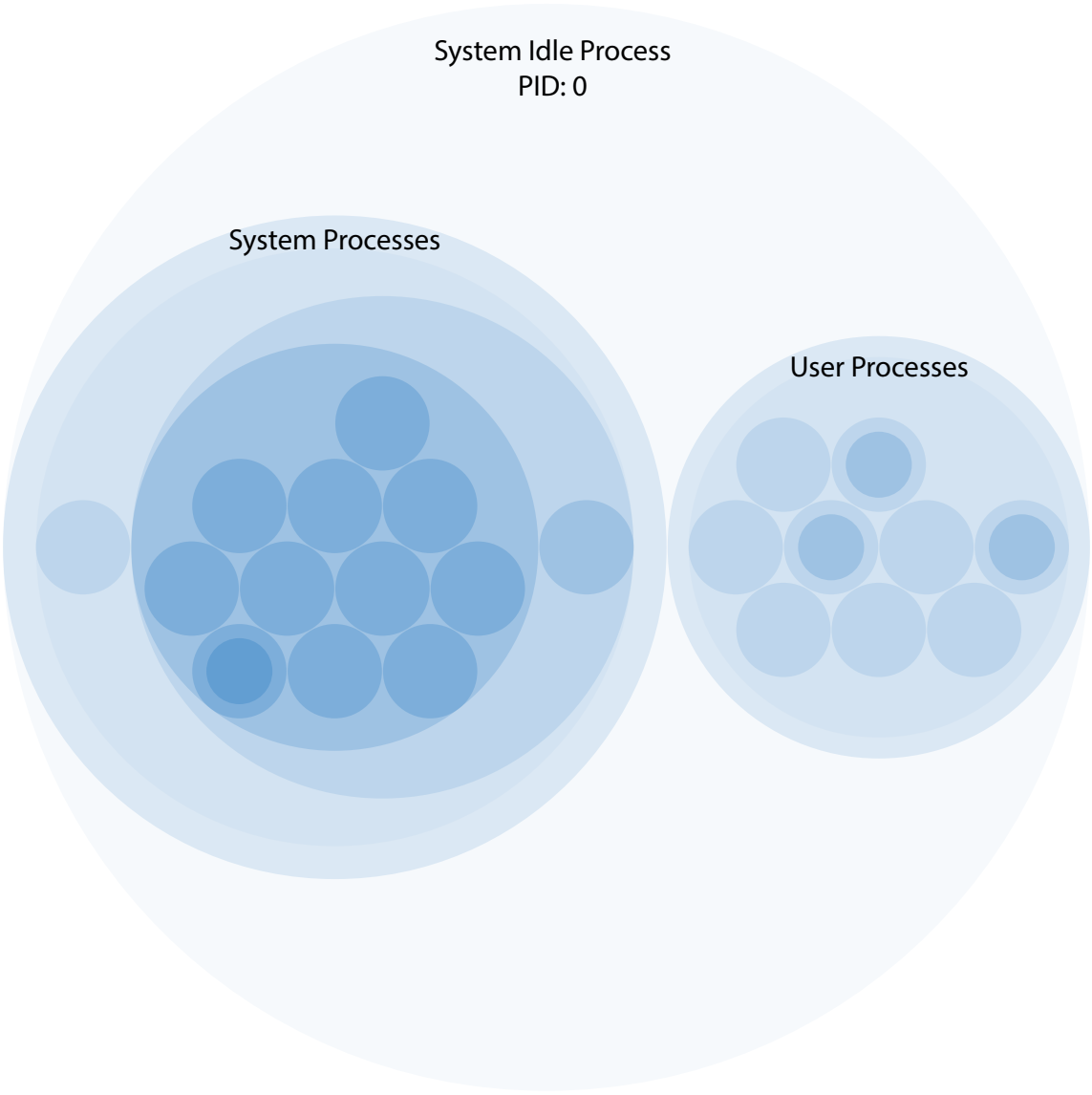


Figure 17. Process Node Hierarchy.

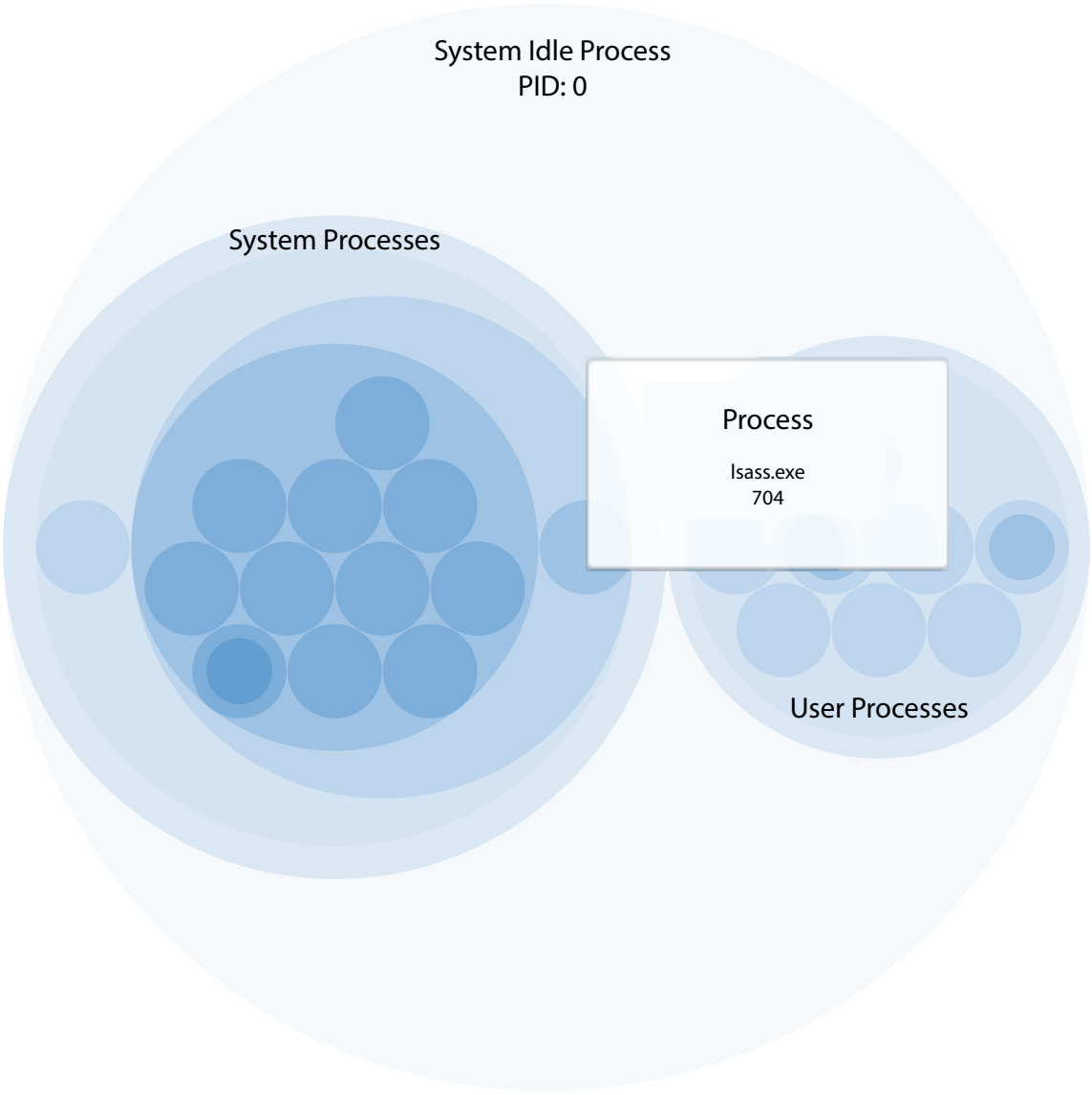


Figure 18. Mouse Over Control.

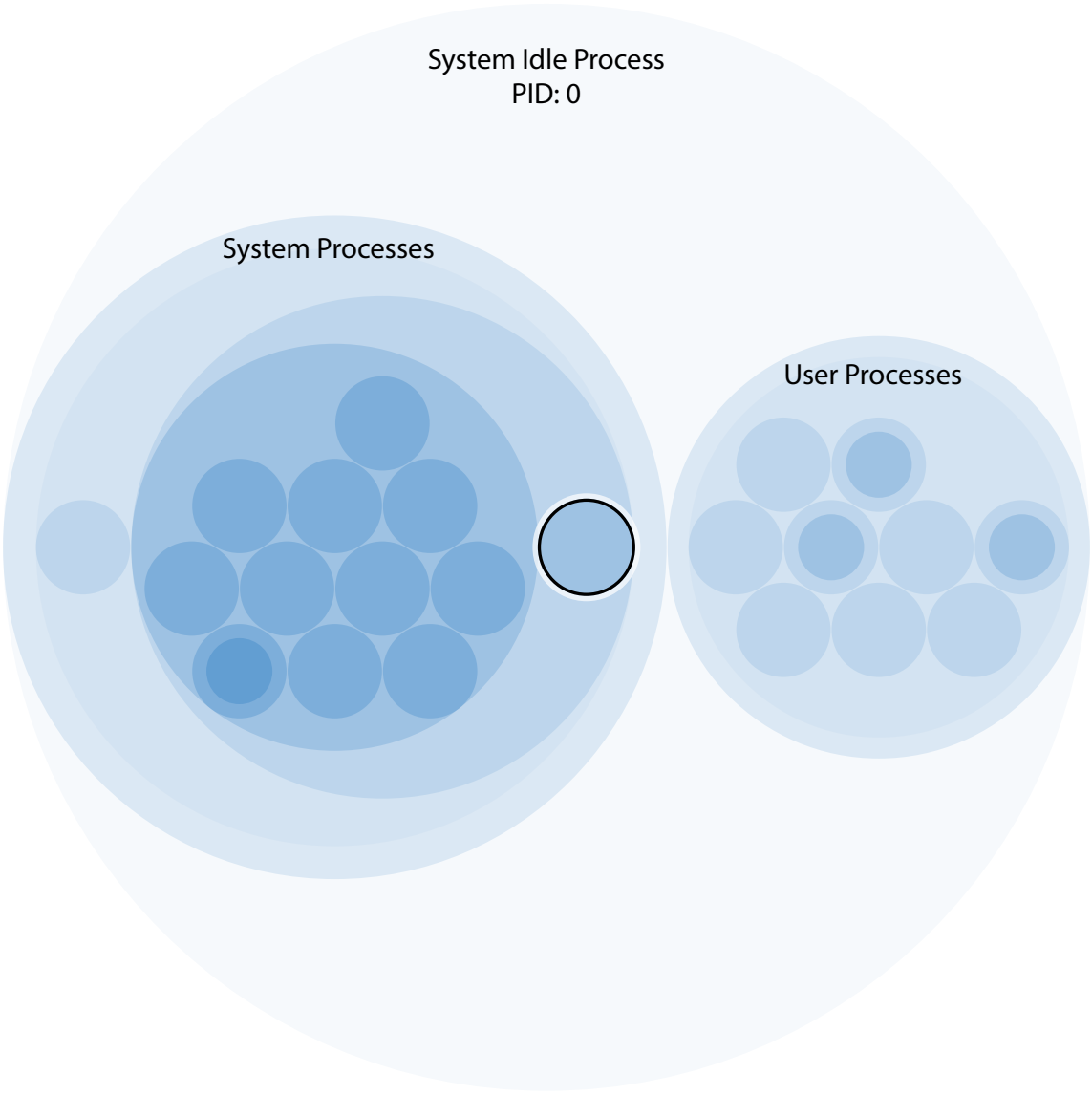


Figure 19. Mouse Click Control.

3.2.3 Resource Arcs.

A donut chart, much like a pie chart shows parts of a whole. When observing the initial system view shown in Figure 20, six system resources (File Handles, Registry Keys, Modules, Services, Ports, and Sockets), which are defined in Table 5 make up the outer donut.

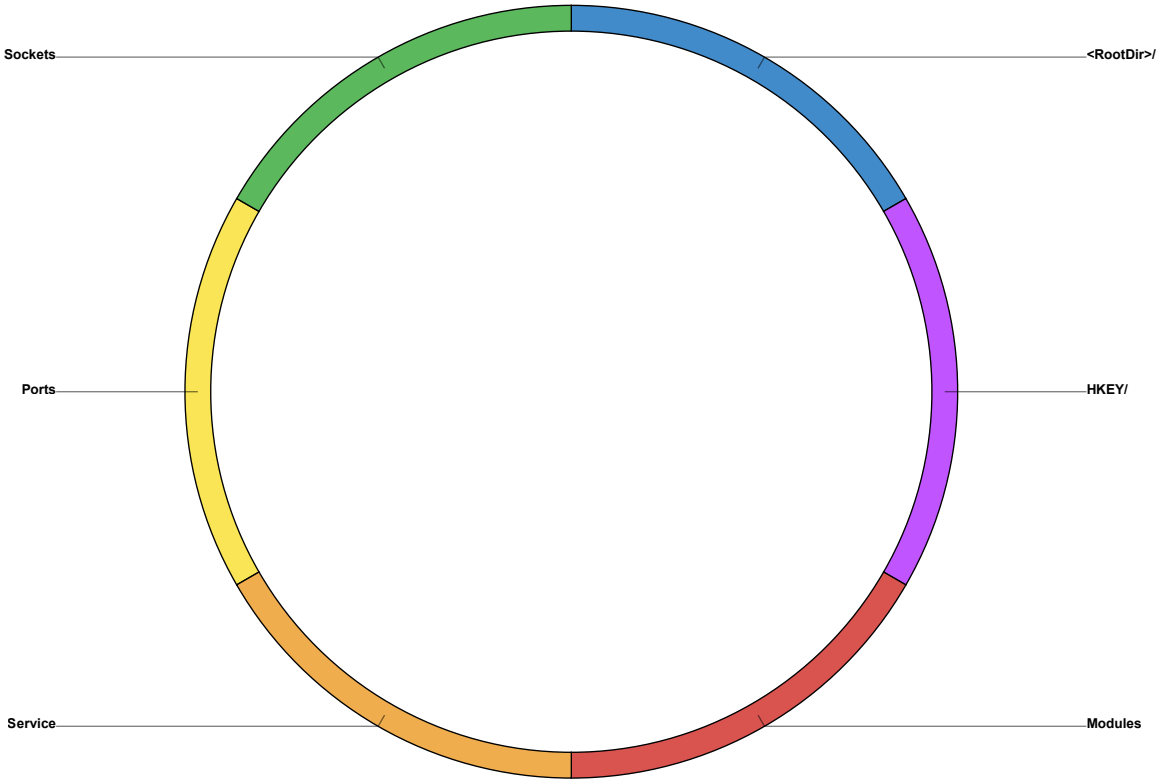


Figure 20. System Resource Arcs.

Table 5. System Resource Definitions.

Resource	Definition
File Handle	A unique identifier* linking an open file to owning PID.
Registry Key Handle	A unique identifier* linking a registry key to owning PID.
Module	Core executable programs and shared system libraries.
Service	A background program providing a specific function.
Port	The operating system end-point of an network connection.
Socket	The process end-point of a network connection.

** Except when a file handle held by a process is duplicated, or process inherits the file handles of the parent*[56].

Three additional user controls are attached to the resource arcs. A mouse-over as seen in Figure 21 displays the name of the resource over which the mouse is currently hovering.

Figure 21. Mouse Over Tool Tip.

A mouse click steps through a hierarchical resource tree and display all branches and leaf nodes at each new level. Leaf nodes are opaque and clicking on them highlights all nodes associated with that specific resource. Figure 22 shows the service specific arc for all running services on the system.

Rolling the mouse wheel up while hovering over an arc steps back one level in the resource tree until the initial system view is reached.

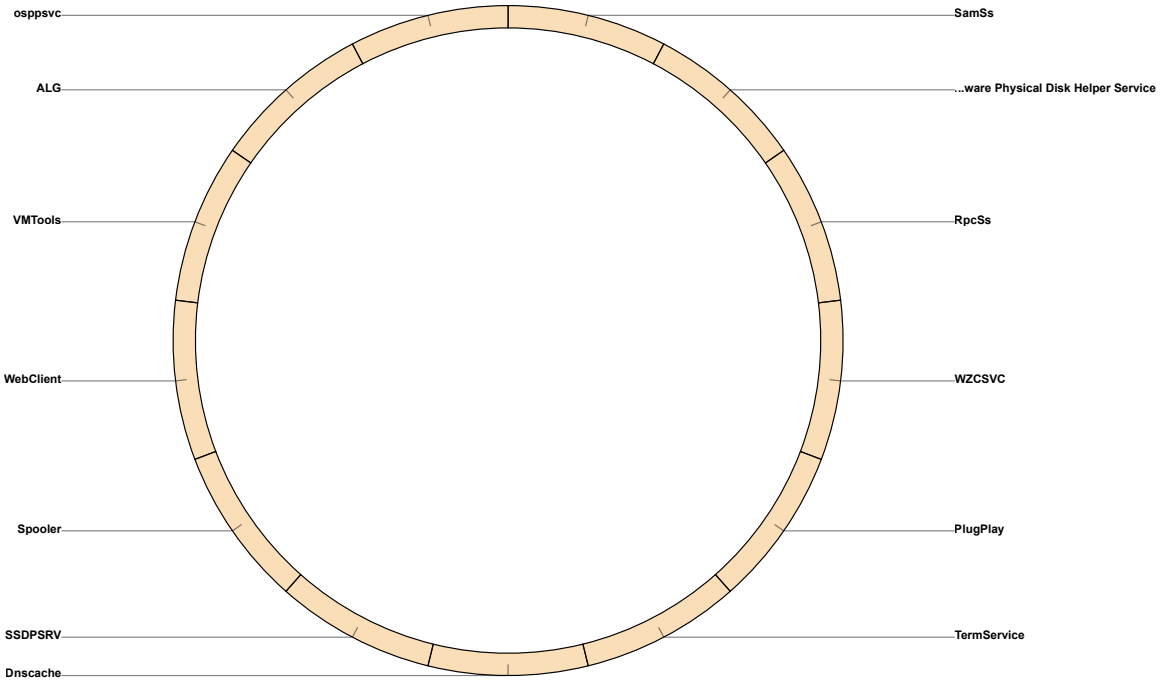


Figure 22. Service Specific Arcs.

3.2.4 Process and Resource Links.

Links show a relationship between connected data. Much like edges in a network diagram, links show a one-to-one, one-to-many, many-to-one, or many-to-many relationship between process nodes and resource arcs. Figure 23 shows the global system view with links enabled for the selected process `lsass.exe`.

Selecting a resource arc steps into that particular resource and draw links from the selected process node to the resources in the current display. Figure 24 depicts the service links for selected node `lsass.exe`.

As noted in the resource arc discussion above, clicking on an opaque resource highlights all associated process nodes, and with links enabled, draws links from the select nodes to all associated resources. Figure 25 illustrates this functionality.

Using the mouse-wheel up control or selecting the 'System View' control button steps backwards in the resource tree while leaving the multiple nodes selected as seen in Figure 26.

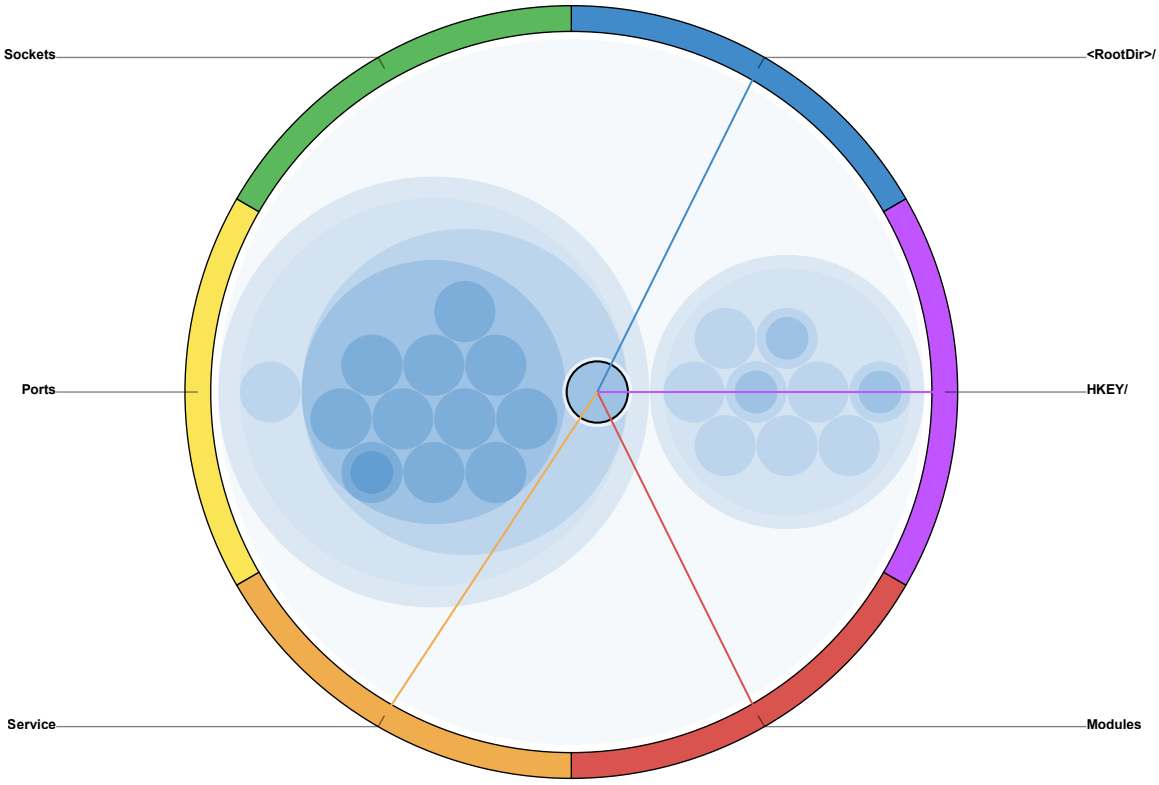


Figure 23. System View Links For lsass.exe.

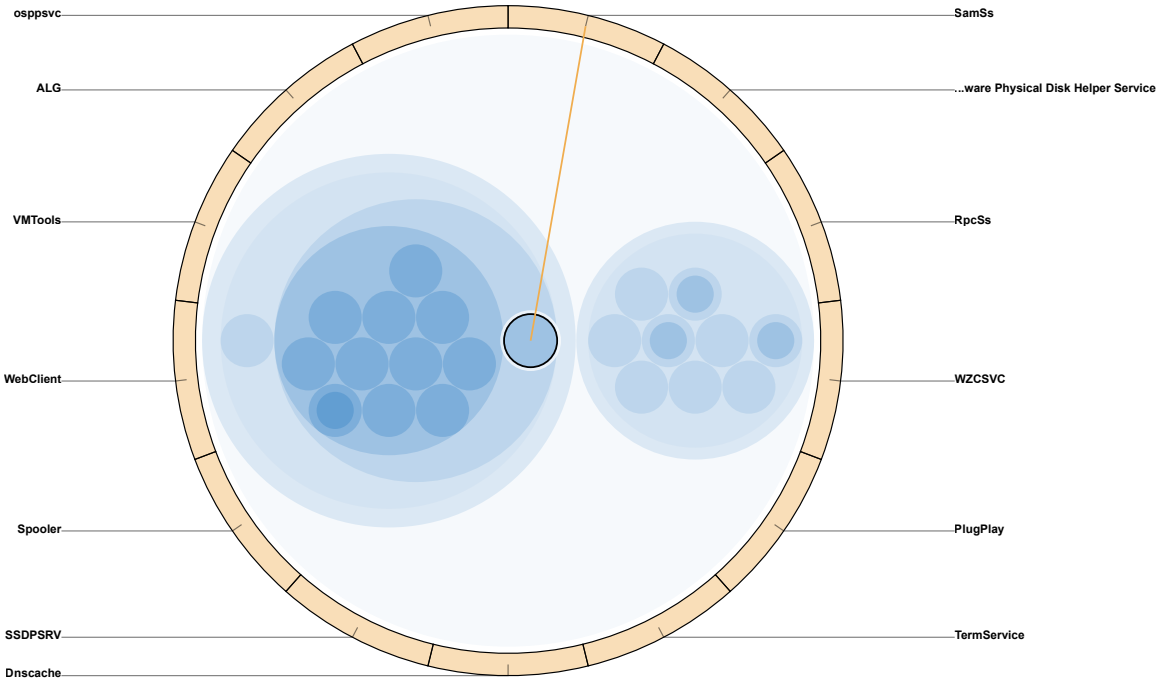


Figure 24. Service Links For lsass.exe.

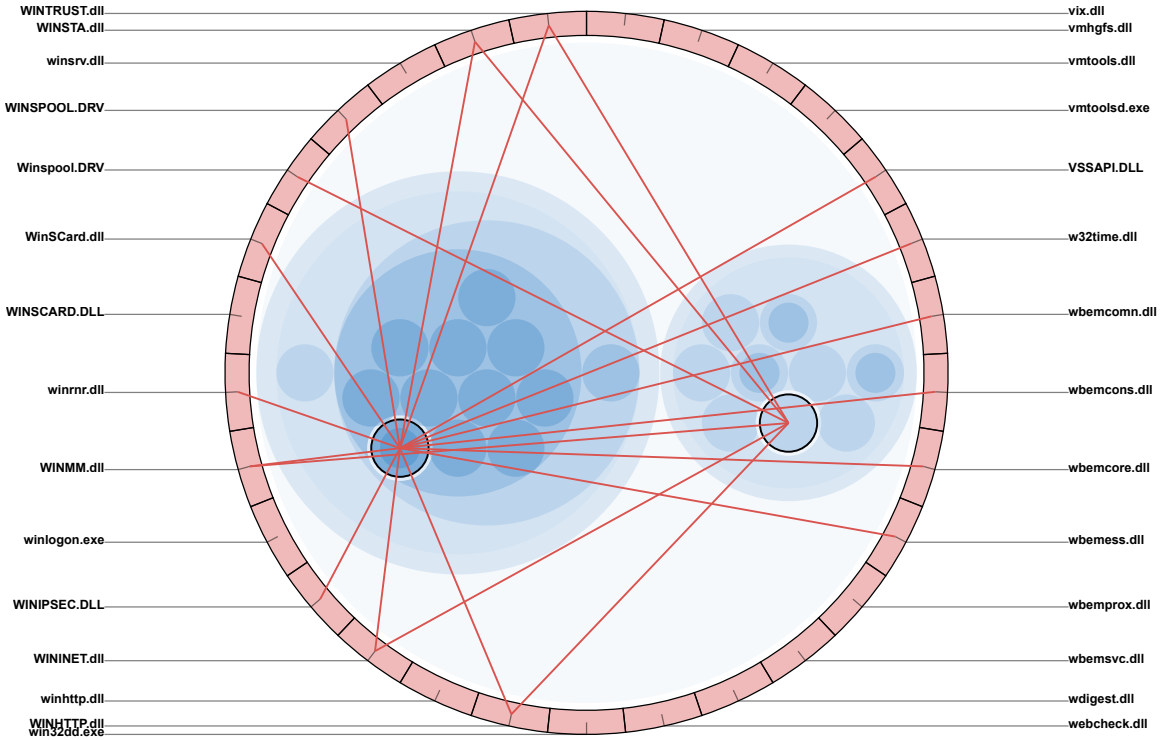


Figure 25. Module Links for Multiple Nodes.

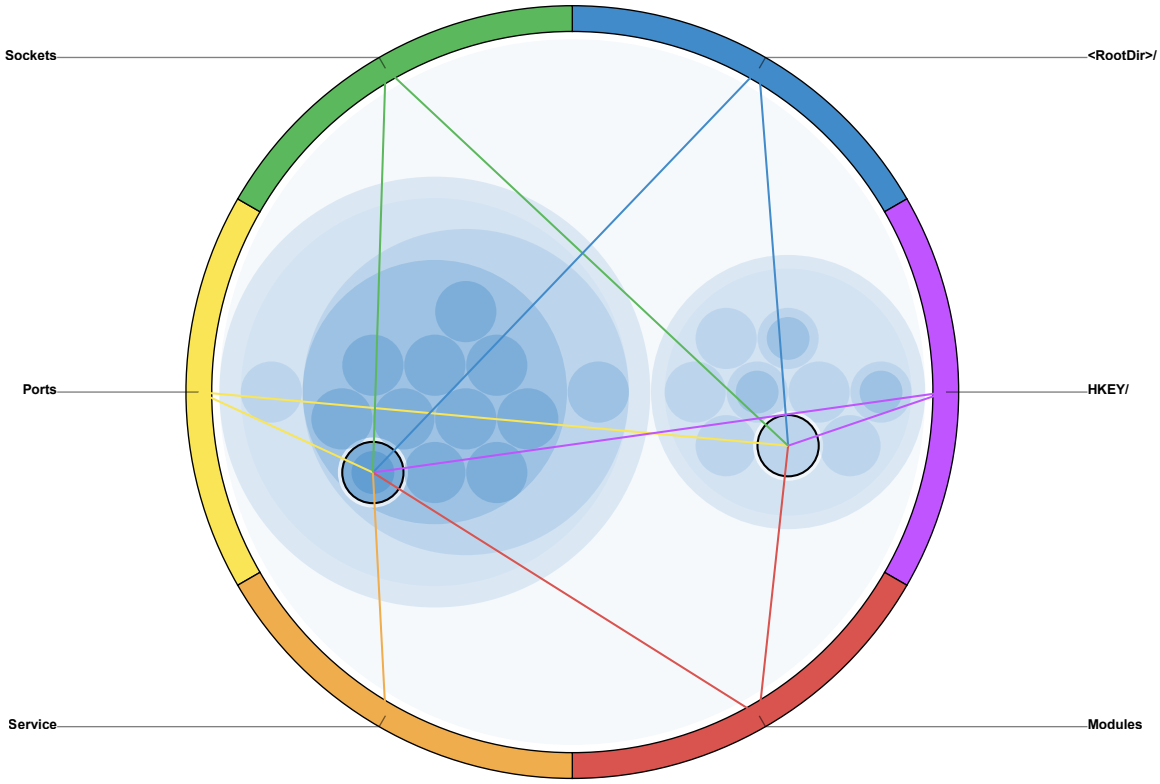


Figure 26. System View Links for Multiple Nodes.

3.2.5 Textual Data View.

The textual data view using the DataTables Javascript plug-in provides raw text data to the analyst with a few additional controls and features. Using HTML tabs, the examiner can switch the table view between the system resources. Figure 27 shows the process list for a clean Microsoft Windows XP Image. DataTables provides a text search function that limits the table with each letter typed. Furthermore, each header is alphabetically or numerically sortable.

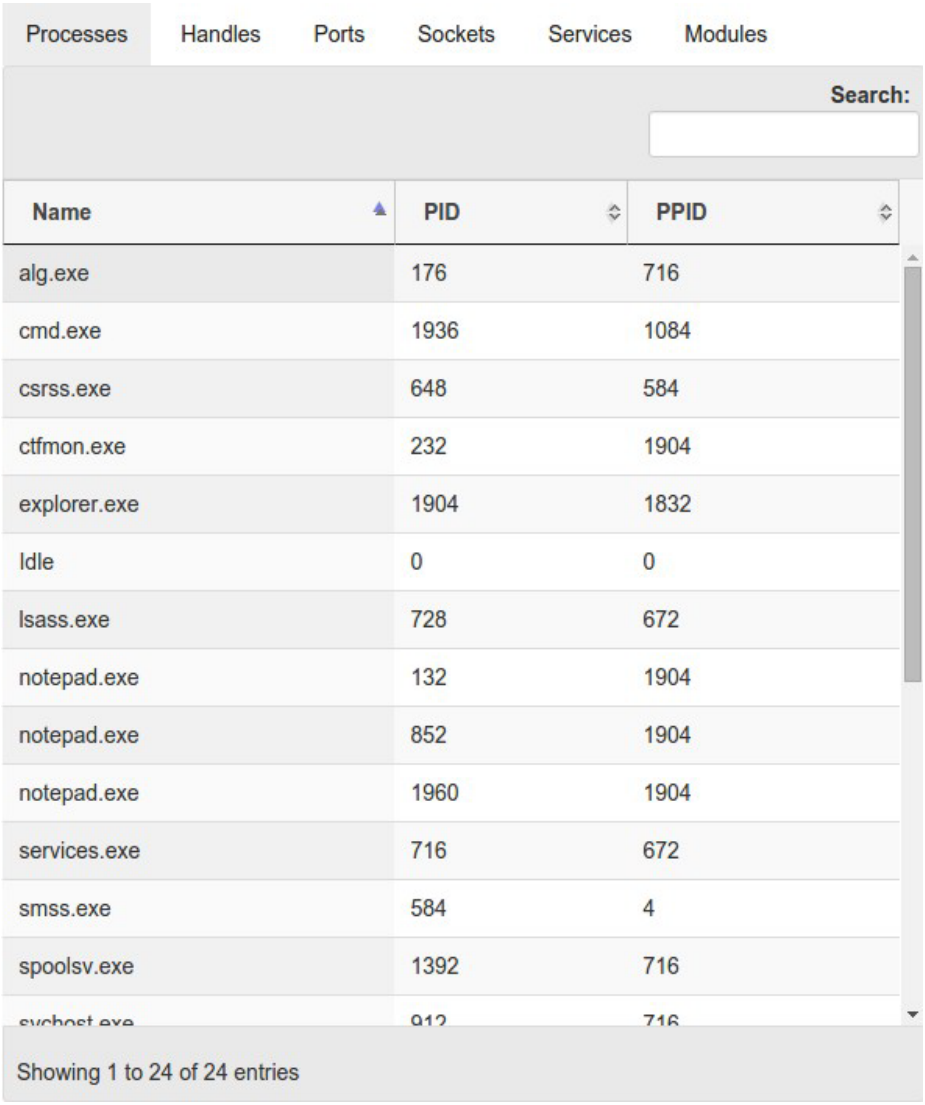


Figure 27. Process List In DataTables.

As the examiner types in the search field, the table is limited to entries with matching strings. Typing "lsa" in the search field limited the table to a single entry for `lsass.exe` as shown in Figure 28.

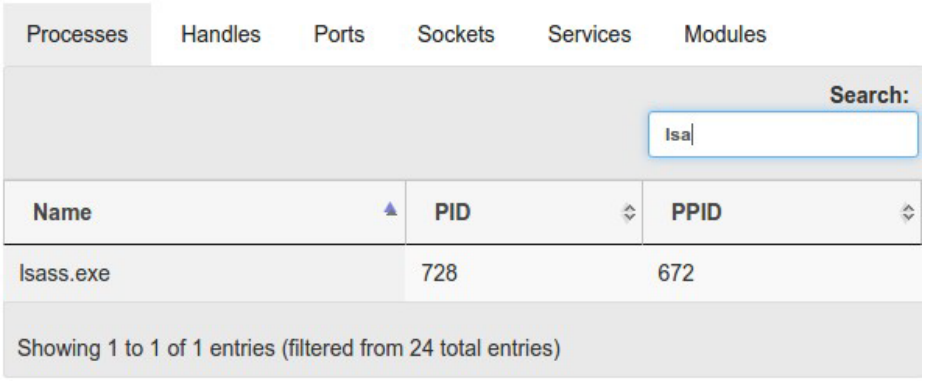


Figure 28. lsass.exe Search In DataTables.

Mouse click event listeners are appended to each table row and highlight an associated process node when clicked. This feature is illustrated in Figure 29.

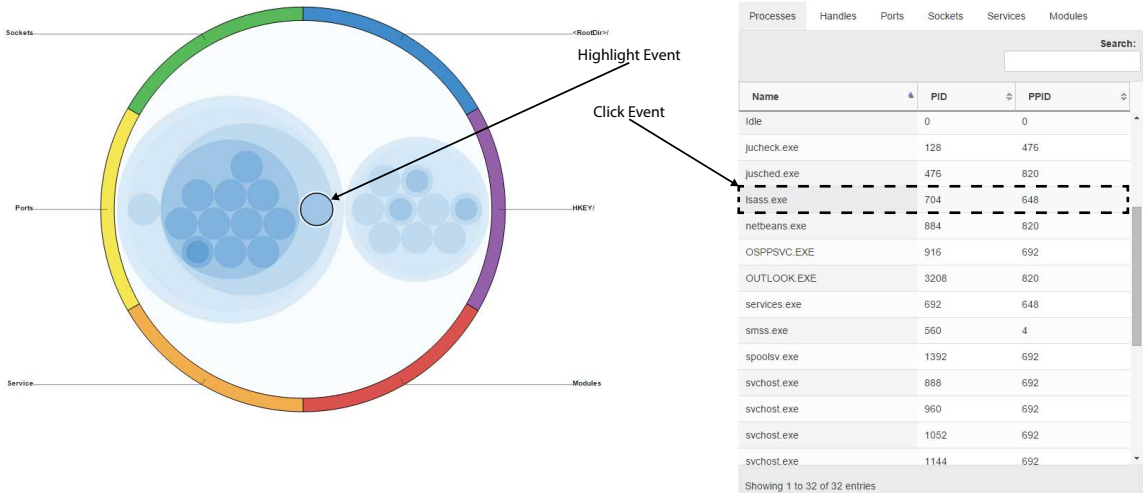


Figure 29. lsass.exe Highlighted Using DataTables Click Event.

3.3 WhiteListing

The memory visualization tool uses a novel, behavioral whitelisting algorithm. This approach to whitelisting looks at a process by application name and its associ-

ated resources to determine if an application behaves in the same manner as other applications of the same name. When a given application behaves the same as other applications bearing the same name, it is likely that application is genuine. For instance, a compromised version of `svchost.exe` would appear differently than genuine versions from a Microsoft release.

The whitelisting process has two main functions:load new images and update resulting percentages. Product versioning is accounted for in the whitelisting process. During both the load and results functions, each memory image is only compared to those of the same major and minor version (i.e., Processes in Windows 5.1 (aka Windows XP) are not compared against other versions of Windows such as 7, 8, 8.1 or 10). The whitelisting process is depicted by Figure 30. This process works best with a very large database of clean memory images, ideally in the tens of thousands.

The load function iterates through each process in the image being loaded and compares it to processes already in the whitelist database. If a given process behaves the same way as a similarly named processes in the database, the whitelisting module increments the count for the number of times that application appears in the whitelist database. However, if the application behaves differently (i.e., uses different dlls, open files, or registry keys, etc...) or it is the first time an application of that name is added to the whitelist, a new application entry is created and the application is assigned a unique application identifier (AppID).

The results function looks at each image in the working database. The function attempts to match each process in a selected image to an AppID and assigns the associated percentage denoting how often that AppID appears in the whitelist database. If no matching AppID exists, the process is assigned zero percent. The percentages are updated in the working database for use in the visualization as described earlier

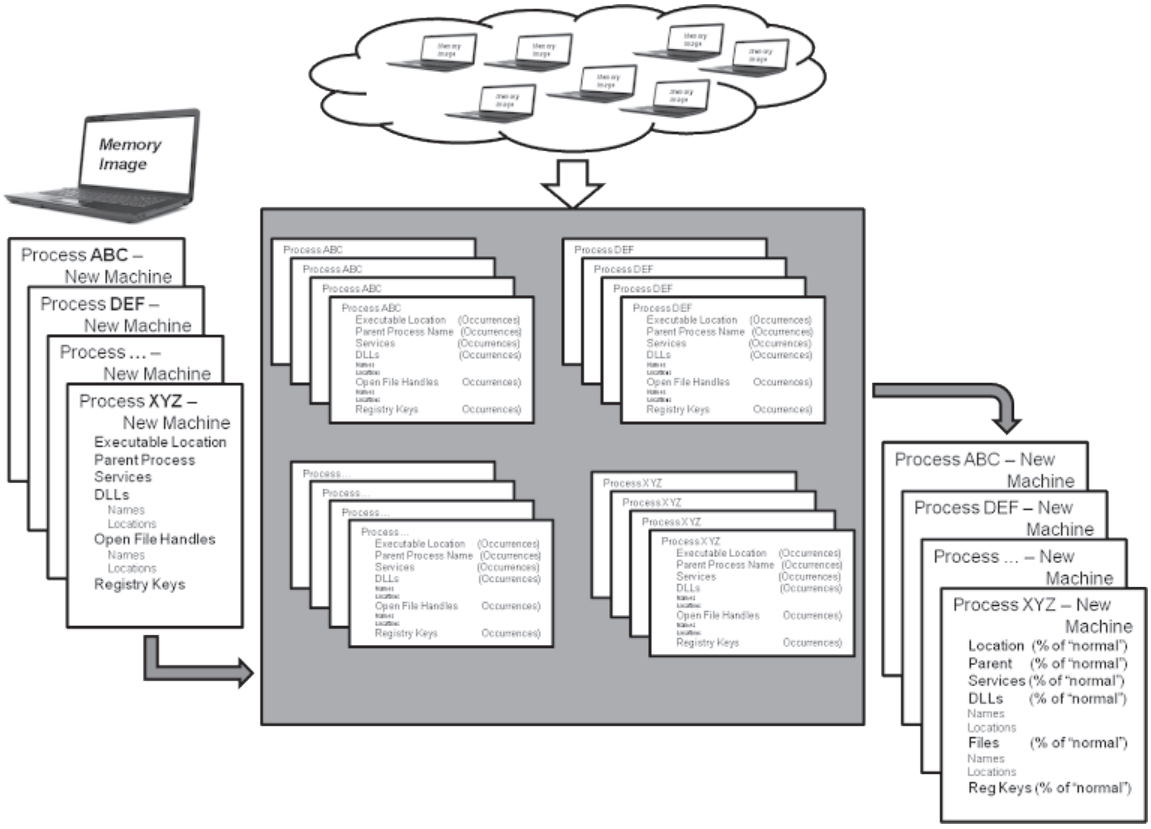


Figure 30. Whitelist Method Diagram.

in this chapter.

3.4 Use Case Examples

To demonstrate use of the memory visualization tool, this section presents two use cases. The first use case describes how to identify artifacts pertaining to user activity on a system. In this case, the examiner focuses on user processes and their associated handles. The second use case describes how to use the tool to detect the presence of malware. In the second case, the examiner utilizes the whitelisting feature and looks for odd behavior in loaded modules and network connections.

3.4.1 User Activity.

The researcher creates a scenario to demonstrate how to analyze user activity with the memory visualization tool. In this scenario the user, Administrator, is editing `TheSecretPlan.docx` in Microsoft Word, reading `HowToWriteAVirus.pdf` in Adobe Reader, and is browsing to `http://www.mpgh.net/forum/showthread-.php?t=578711` (a java language virus writing tutorial page) with Mozilla Firefox. The following sections show how to locate forensic artifacts using the memory visualization tool.

To begin analysis, the researcher selects the Image 'UserActivity' from the drop-down menu and click 'Visualize Dataset' as shown in Figure 31. When the visualization has finished loading, the researcher clicks the 'Begin Memory Analysis' button shown in Figure 32.

Figure 31. Visualize User Activity Image.

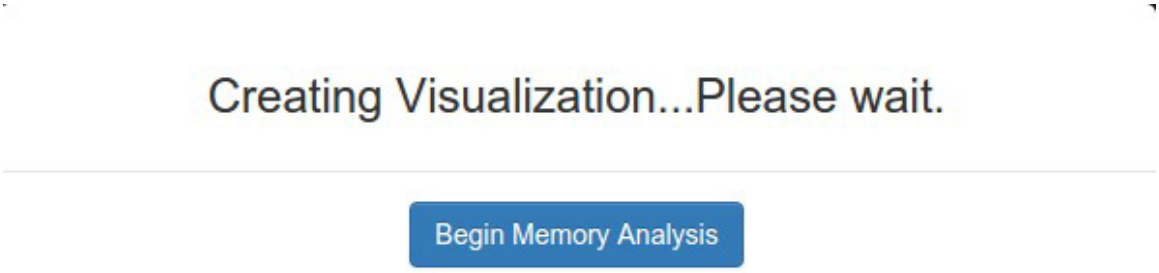


Figure 32. Begin Memory Analysis.

3.4.1.1 User Processes.

Looking at the user processes (i.e., the child processes of `Explorer.exe`) shown in Figure 33, the examiner identifies five user processes likely initiated by the user:

netbeans, WINWORD, OUTLOOK, AcroRd32, and Firefox. The processes ctfmon, jucheck, and vmtoolsd are likely started automatically at login. Lastly, the process win32dd is the memory capture tool initiated by the incident response team. The following sections explore the Microsoft Word, Adobe Reader, and Mozilla Firefox Processes.

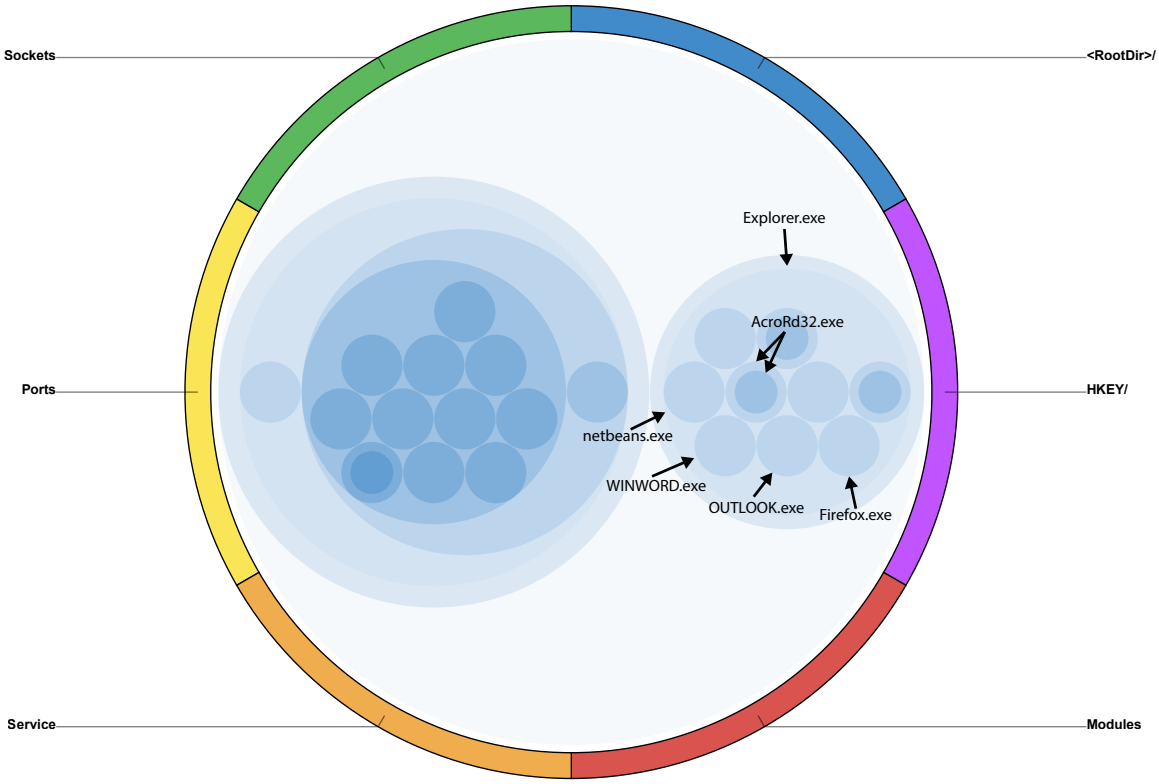


Figure 33. User Processes in User Activity Image.

3.4.2 Microsoft Word Handles.

The researcher highlights the process WINWORD.exe by hovering the mouse curser over the desired process node and performs a right mouse click as in Figure 34. He then turns on the resource links by clicking the 'Toggle Links' button seen in Figure 35 to view process to resource links for WINWORD.exe as seen in Figure 36.

The blue link indicates that WINWORD.exe has open file handles, while the pur-

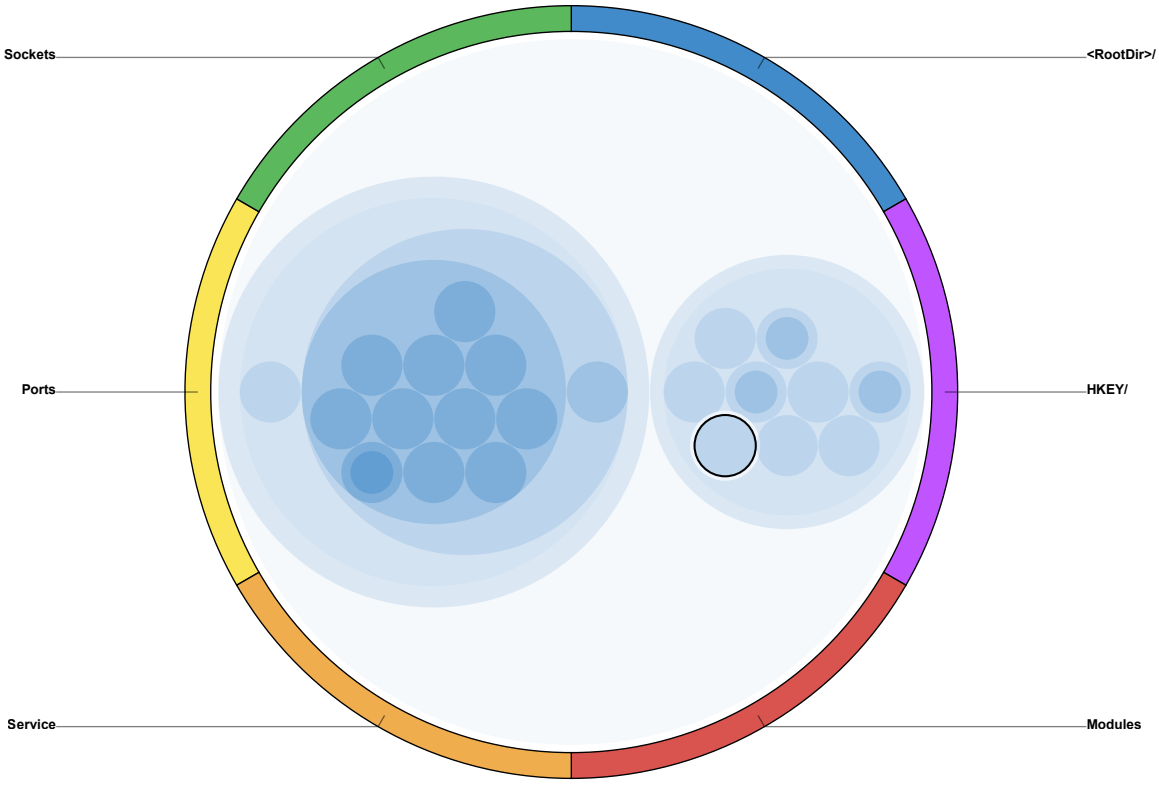


Figure 34. Highlight WINWORD.exe.

Figure 35. Turn-On Node-Resource Links.

ple and the red links indicate open registry keys and loaded modules respectively. Performing a right mouse click on the `<RootDir>/` resource arc steps into the root of the open file handles and the resulting display shown in Figure 37 shows all subdirectories and file handles in the root directory. The links identify the subdirectories and file handles with which `WINWORD.exe` is associated.

Following the link to the `Documents and Settings` subdirectory by right-clicking its resource arc enters into that directory and displays its subdirectories and file handles as seen in Figure 38.

Following the link to the `Administrator` subdirectory steps into that directory and provide the results view in Figure 39.

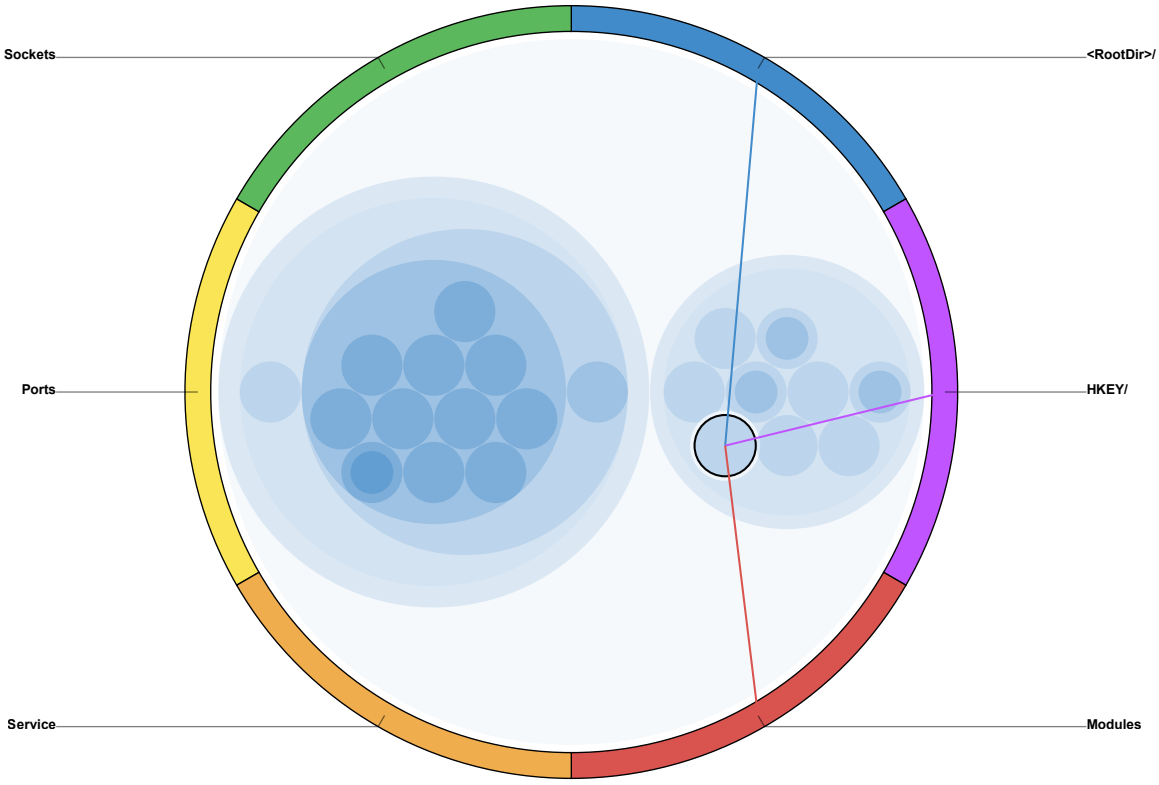


Figure 36. WINWORD.exe Resource Links.

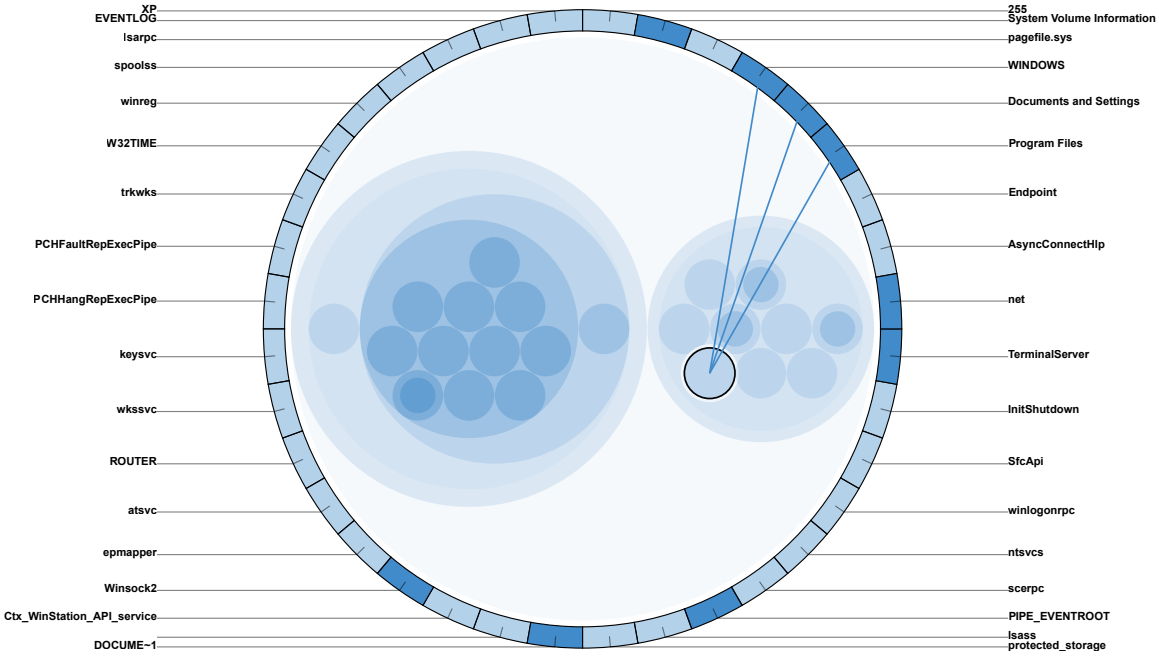


Figure 37. WINWORD.exe Resource Links Root Directory.
Note: Opaque arcs denote file handles.

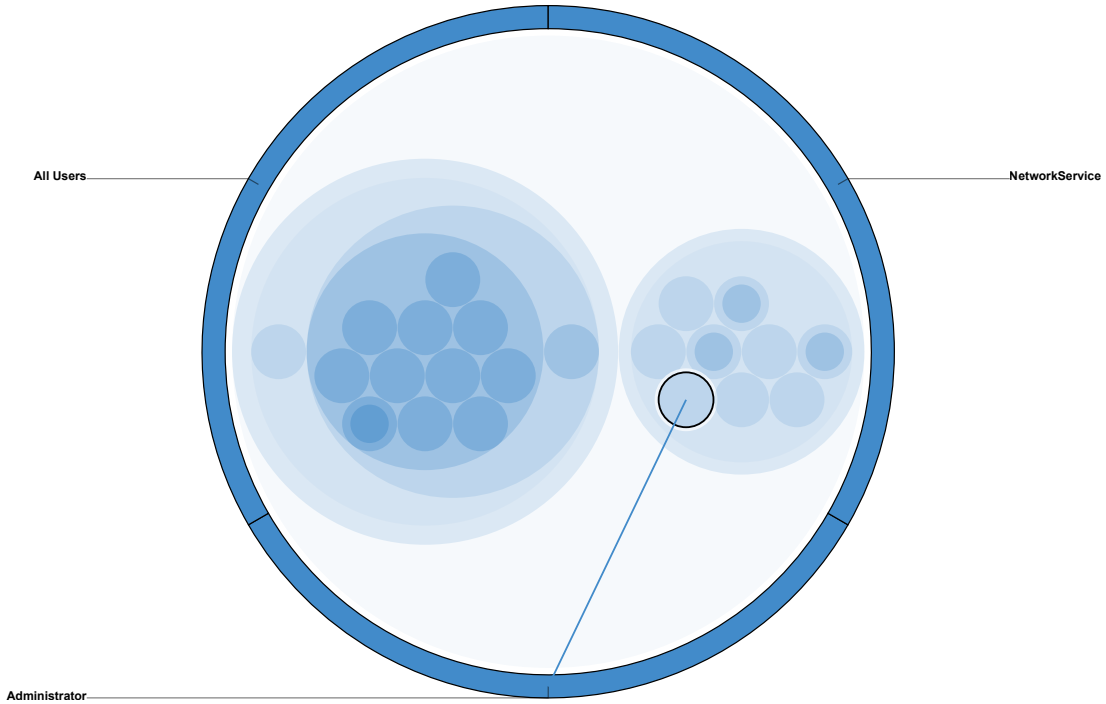


Figure 38. WINWORD.exe Resource Links Documents and Settings.

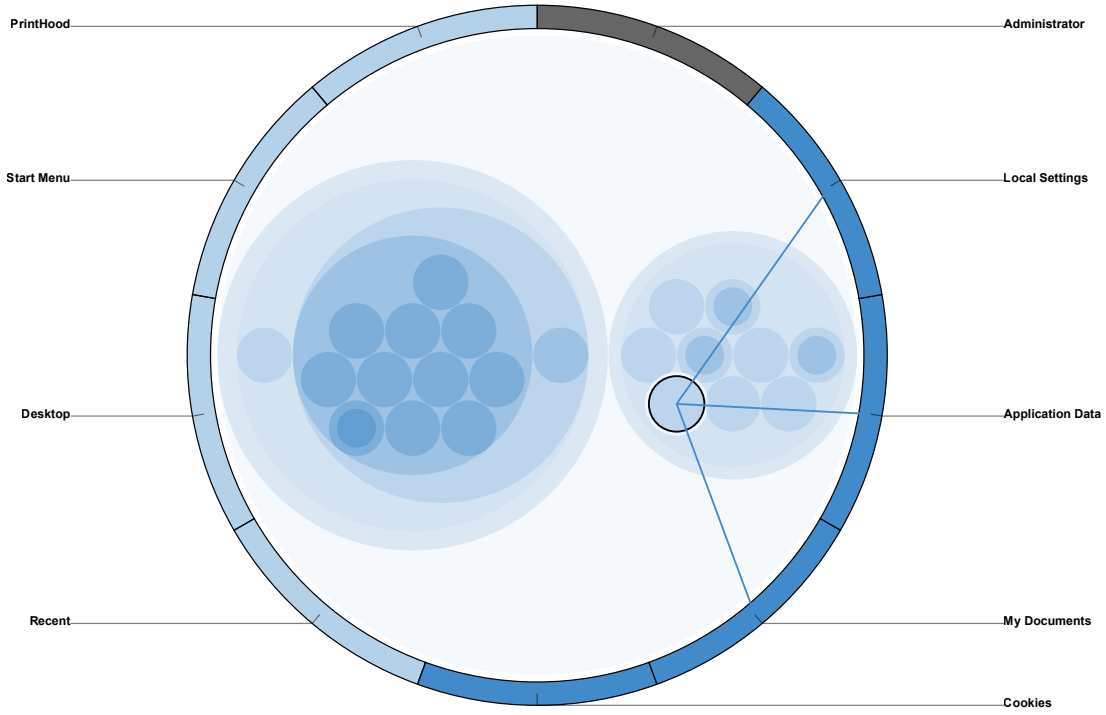


Figure 39. WINWORD.exe Resource Links Administrator
Note: The gray arcs denote directory handles.

Stepping into the `My Documents` subdirectory brings up the final view shown in Figure 40 which has a single link to the file handle `C:\Documents and Settings\Administrator\My Documents\TheSecretPlan.docx`.

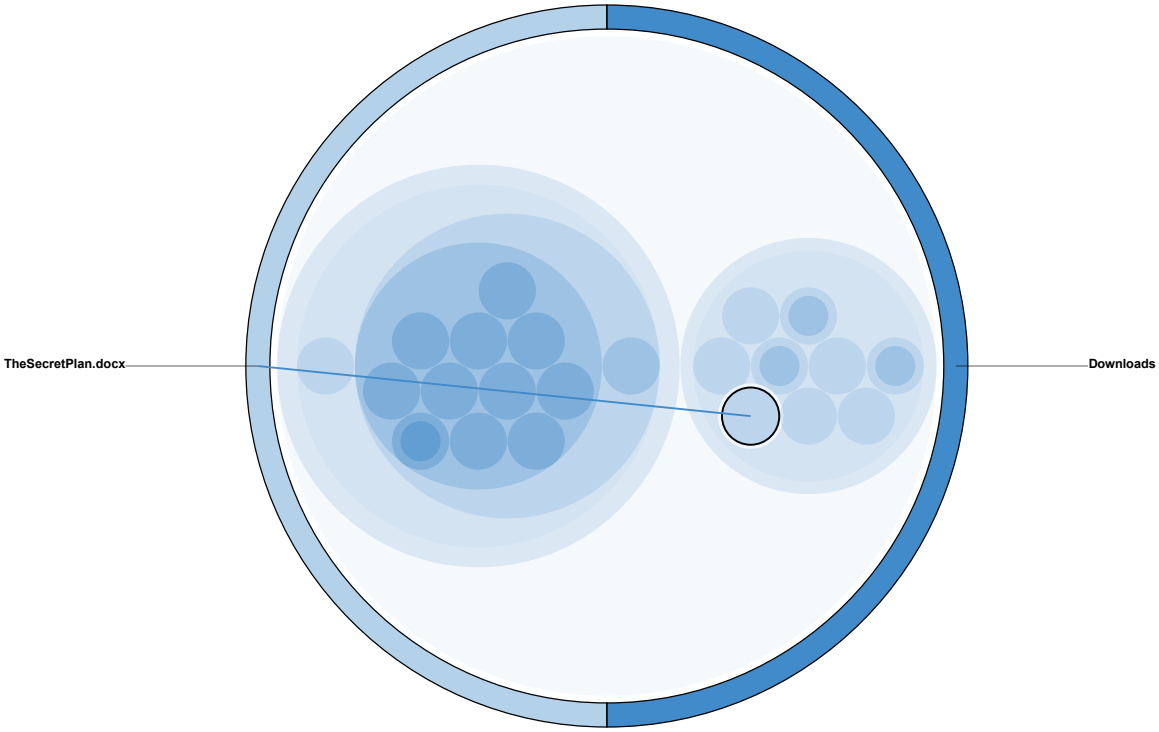


Figure 40. WINWORD.exe Open File Handle TheSecretPlan.docx.

Click the 'System View' button illustrated in Figure 41 to return to the global system view.

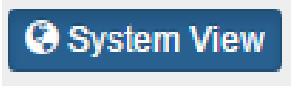


Figure 41. Return to System View.

3.4.2.1 Adobe Reader Handles.

Select `AcroRd32.exe` (PID 1116), which is a child process (or more correctly the plug-in process) of `AcroRd32.exe` (PID 364) as shown in Figure 42. As with the `WINWORD.exe` example, following the blue link to the root directory will show all sub-

directories and file handles in the root directory as seen in Figure 43. Quickly stepping through the directory path `<RootDir>\Documents and Settings\Administrator\My Documents\Downloads\` will bring up the views shown in Figures 44, 45, 45, 46, and 47 respectively. Figure 47 shows a single link between the process `AcroRd32.exe` and the file handle for `C:\Documents and Settings\Administrator\My Documents\Downloads\HowToWriteAVirus.pdf`. Click the 'System View' button to return to the global system view.

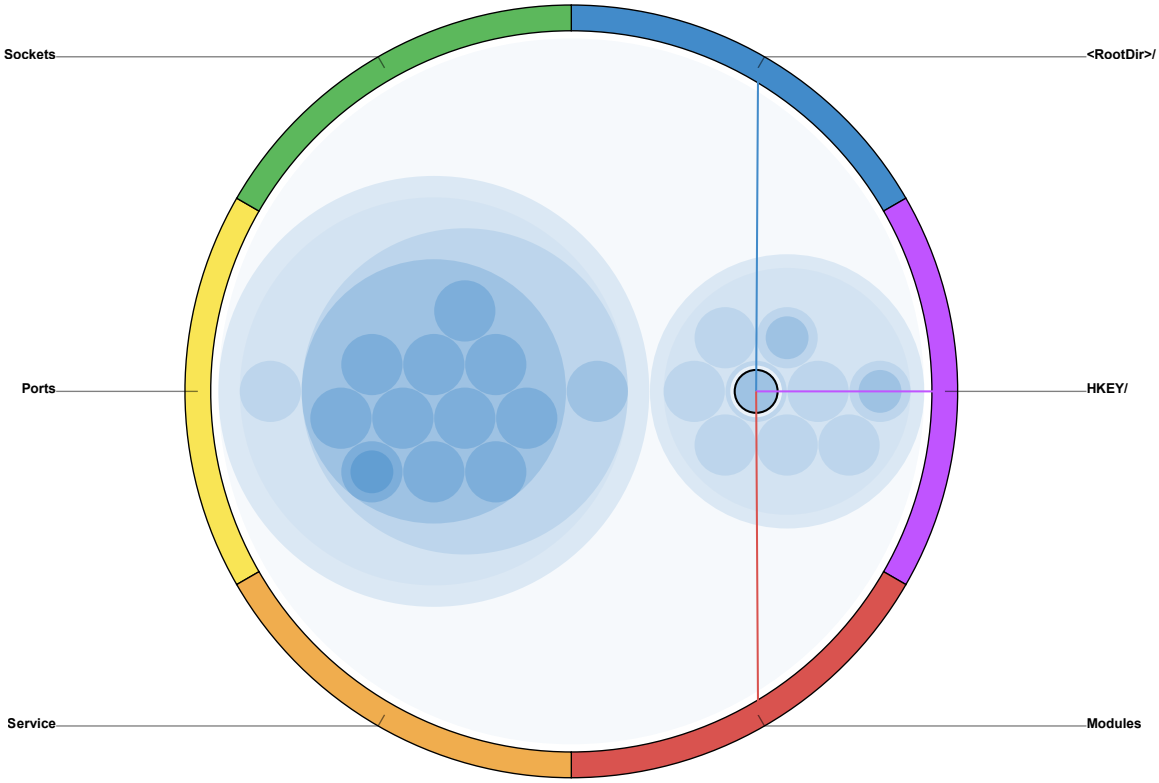


Figure 42. AcroRd32.exe Resource Links.

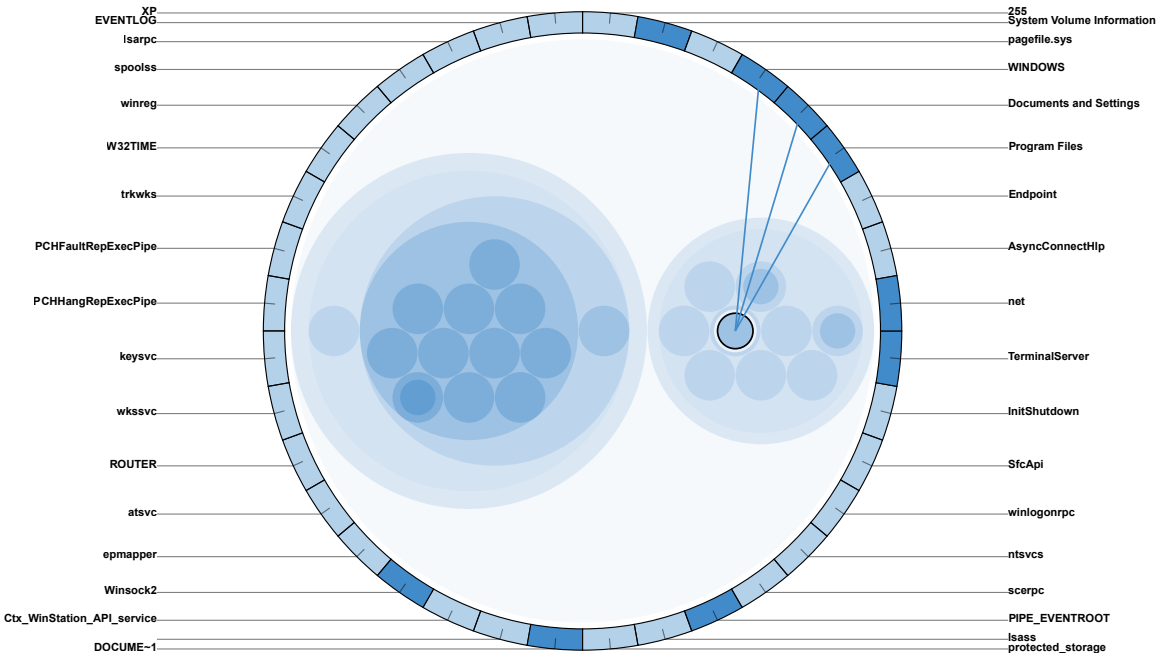


Figure 43. AcroRd32.exe Resource Links Root.

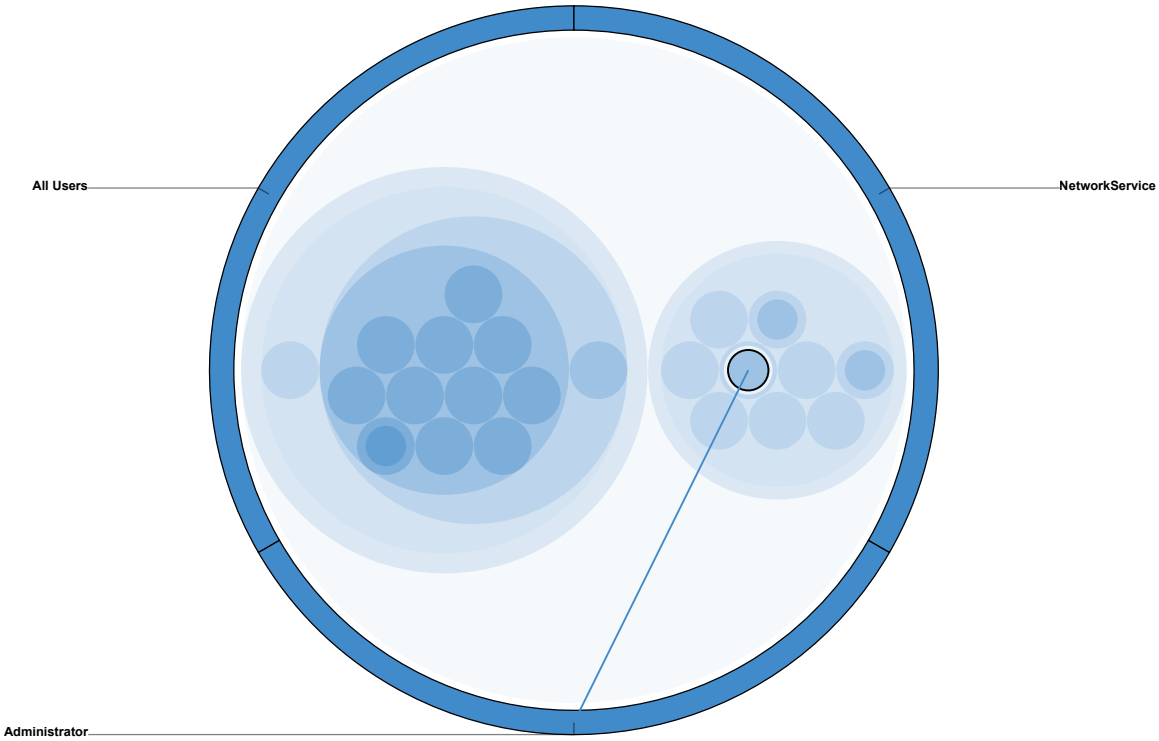


Figure 44. AcroRd32.exe Resource Links Documents and Settings.

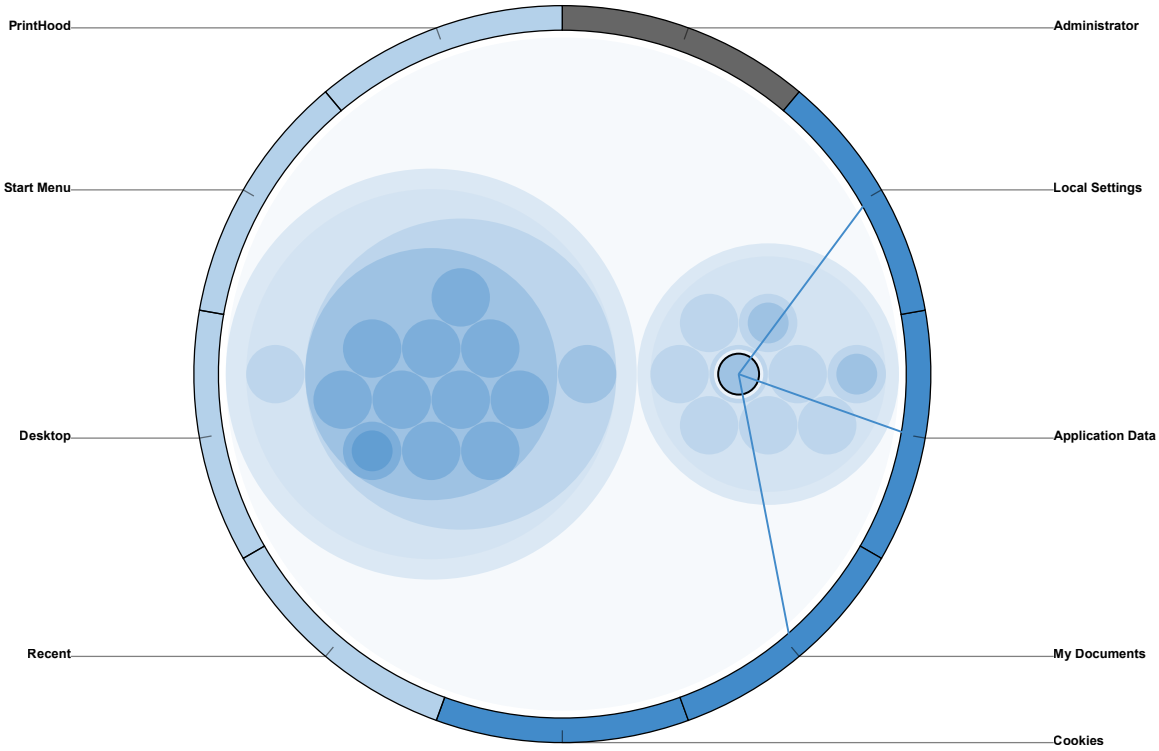


Figure 45. AcroRd32.exe Resource Links Administrator.

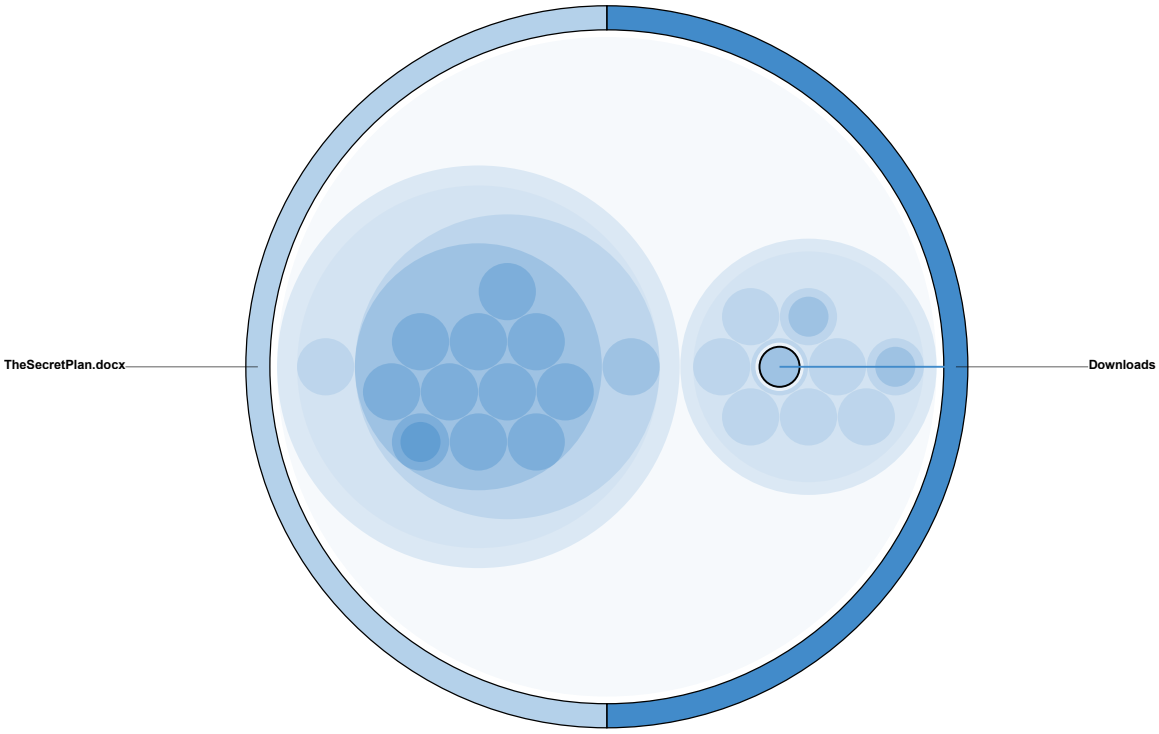


Figure 46. AcroRd32.exe Resource Links My Documents.

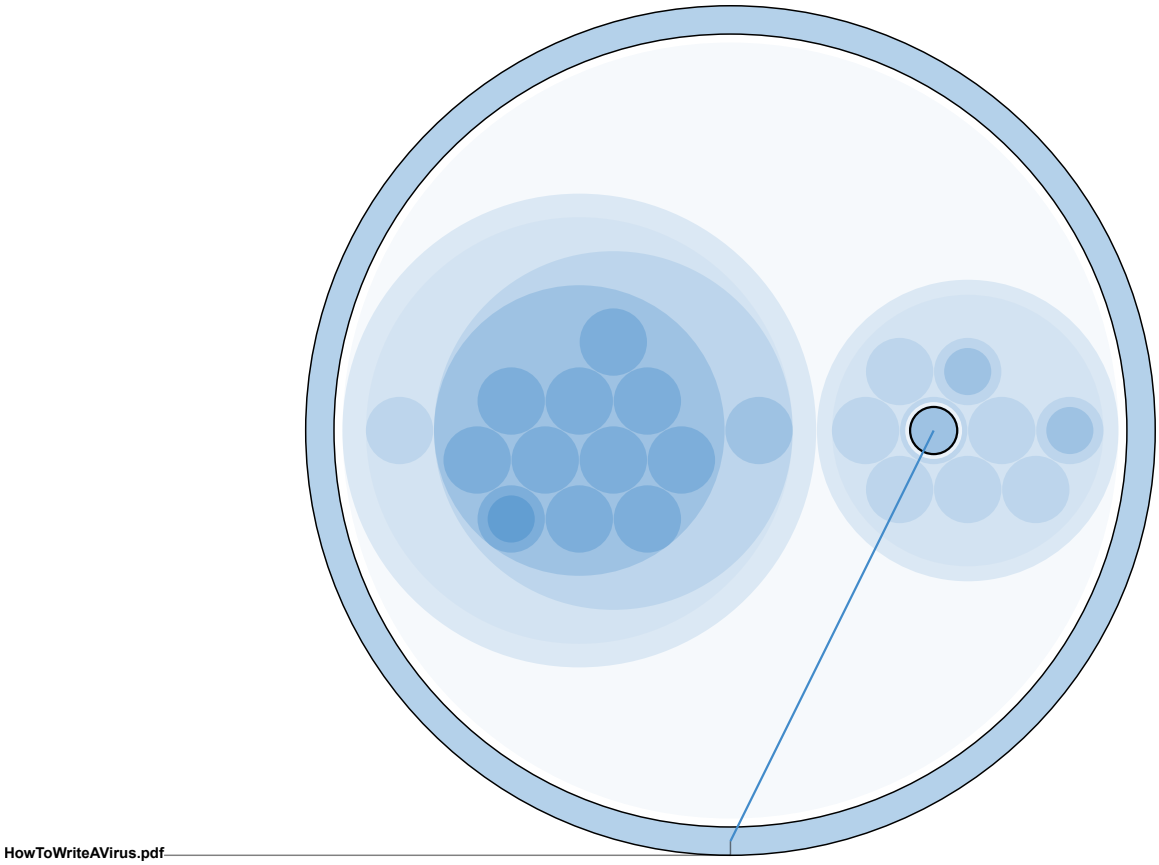


Figure 47. AcroRd32.exe Open File Handle.

3.4.2.2 Firefox Handles and Connections.

Select the `Firefox.exe` process node as shown in Figure 48. Following the green link shows all open sockets. However, as seen in Figure 49, the only sockets `Firefox.exe` has open are loopback addresses. Click the 'System View' button to return to the global system view. Following the blue link to the root directory brings up the file view seen in Figure 50. Quickly stepping through the directory path `<RootDir>\Documents and Settings\Administrator\Application Data\Mozilla\Firefox\Profiles\4a4novg1.default` generates the views seen in Figures 51, 52, 53, 54, 55, 56, and 57 respectively. In Figure 57, `Firefox.exe` has several links to file handles, but Mozilla Firefox stores browsing history in `places.sqlite` and that is the file handle of interest. Examining the contents of `places`

`.sqlite` would show that the User visited the website `Tutorial: Java Beginning Virus Programming, MultiPlayer Game Hacking & Cheats (http://www.mpgh.net/forum/showthread.php?t=578711)`.

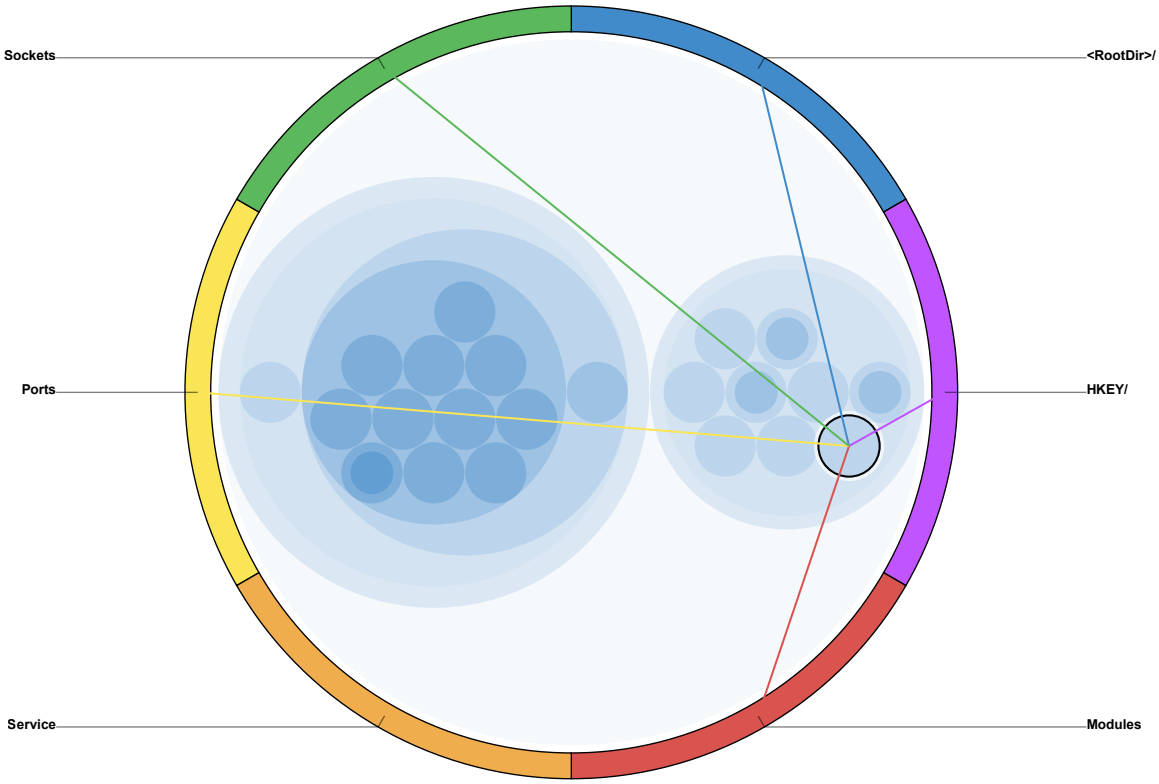


Figure 48. Firefox.exe Resource Links.

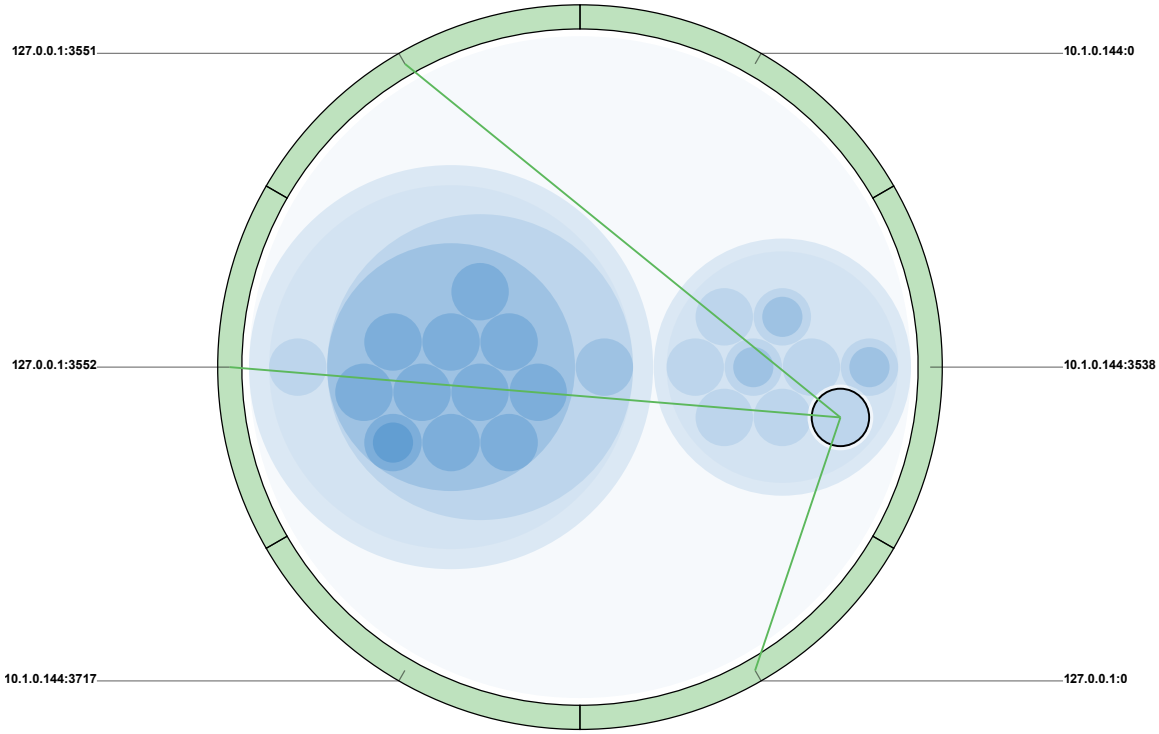


Figure 49. Firefox.exe Sockets Links.

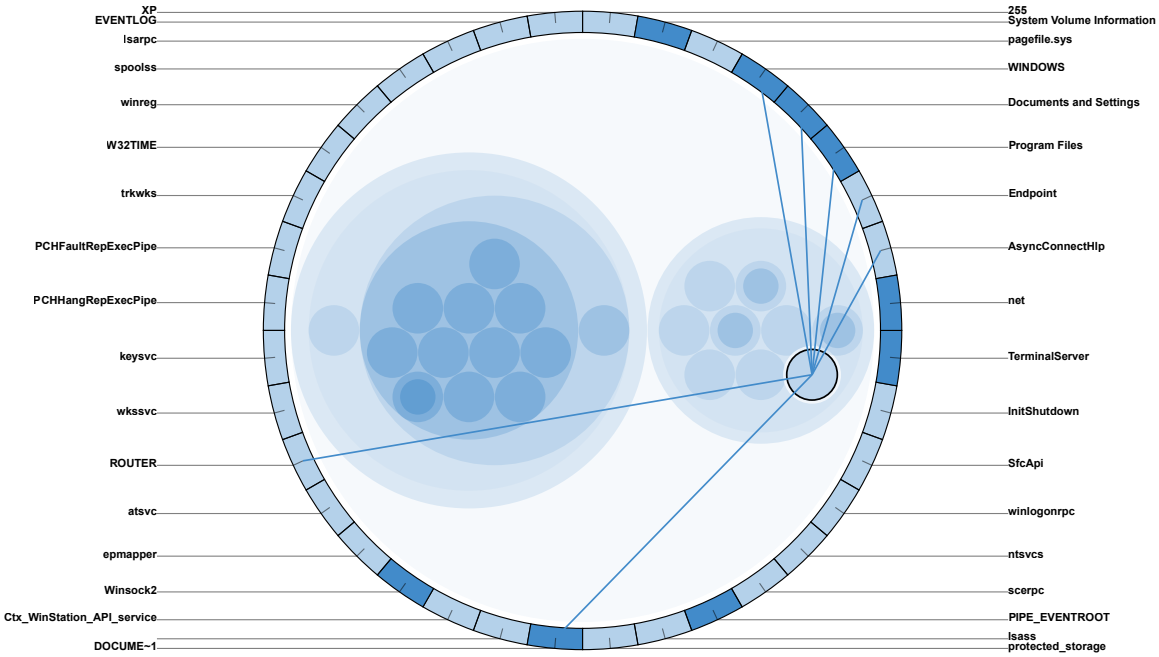


Figure 50. Firefox.exe Resource Links Root.

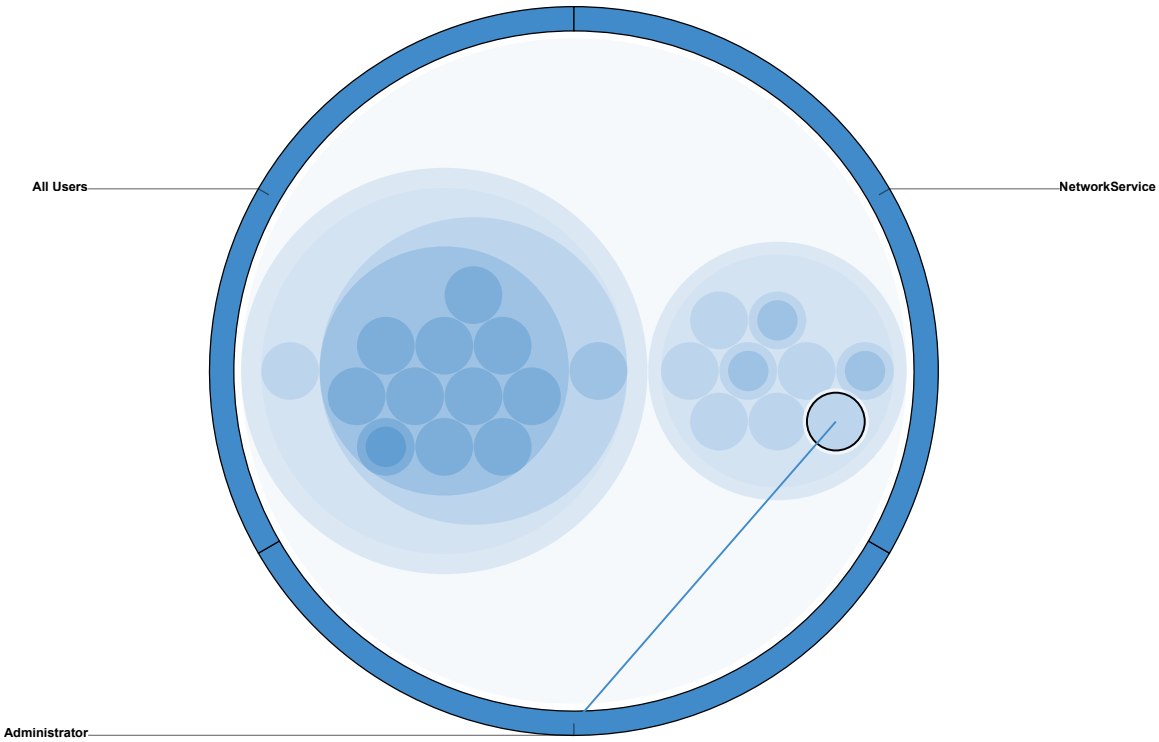


Figure 51. Firefox.exe Resource Links Documents.

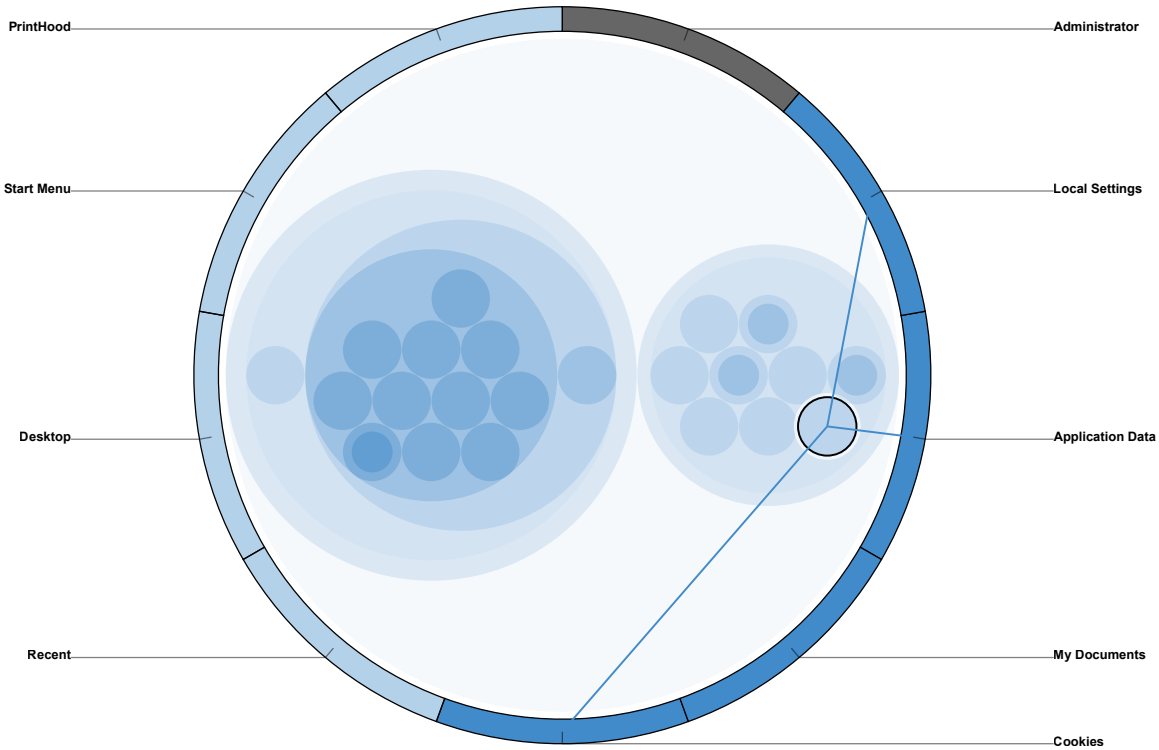


Figure 52. Firefox.exe Resource Links Administrator.

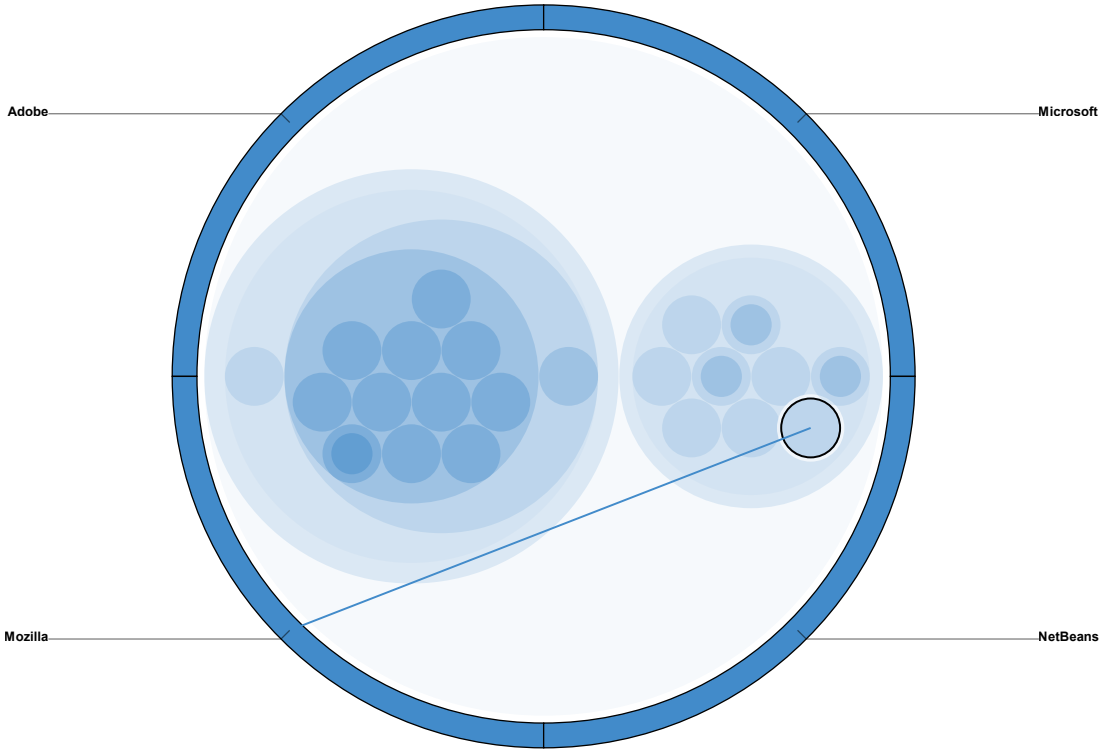


Figure 53. Firefox.exe Resource Links Application Data.

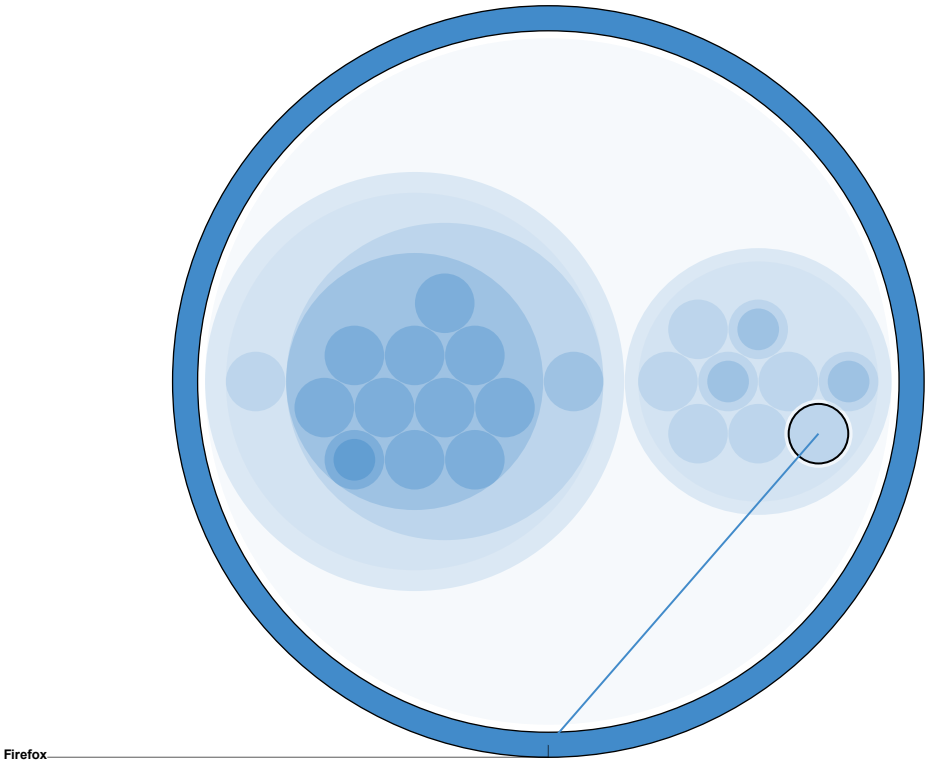


Figure 54. Firefox.exe Resource Links Mozilla.

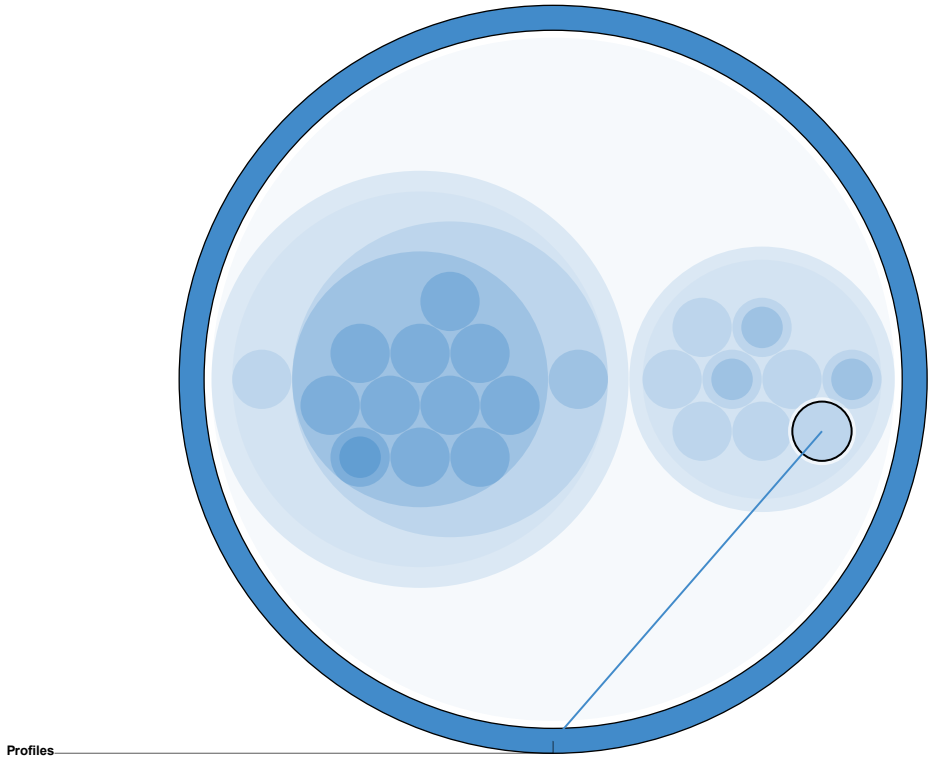


Figure 55. Firefox.exe Resource Links Firefox.

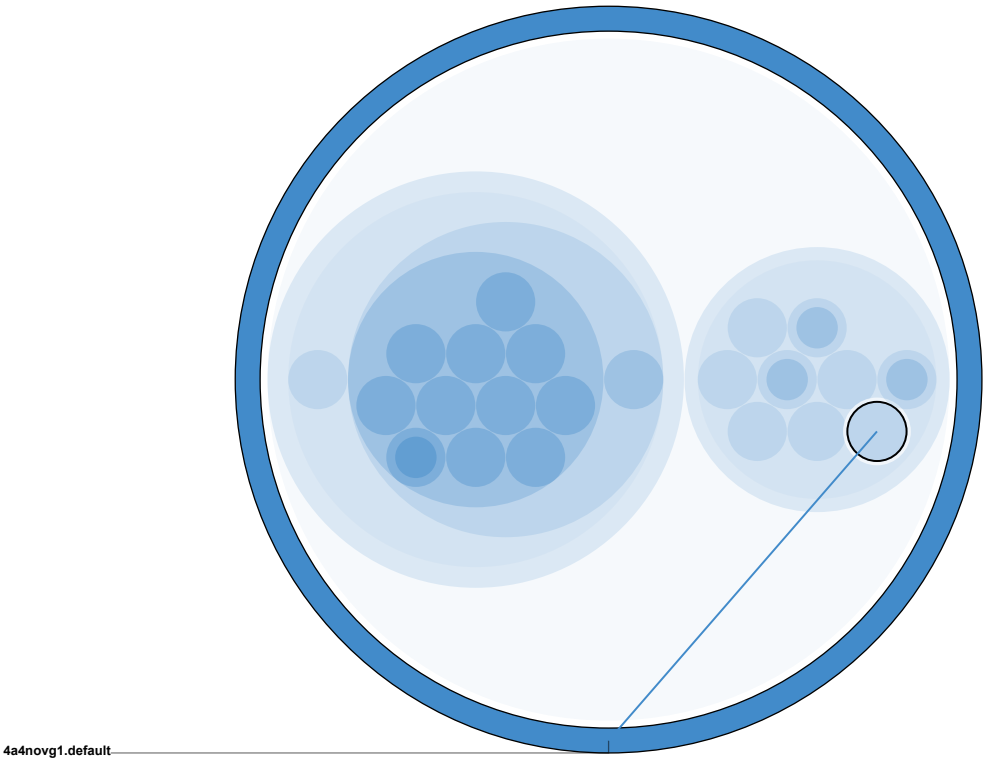


Figure 56. Firefox.exe Resource Links Profiles.

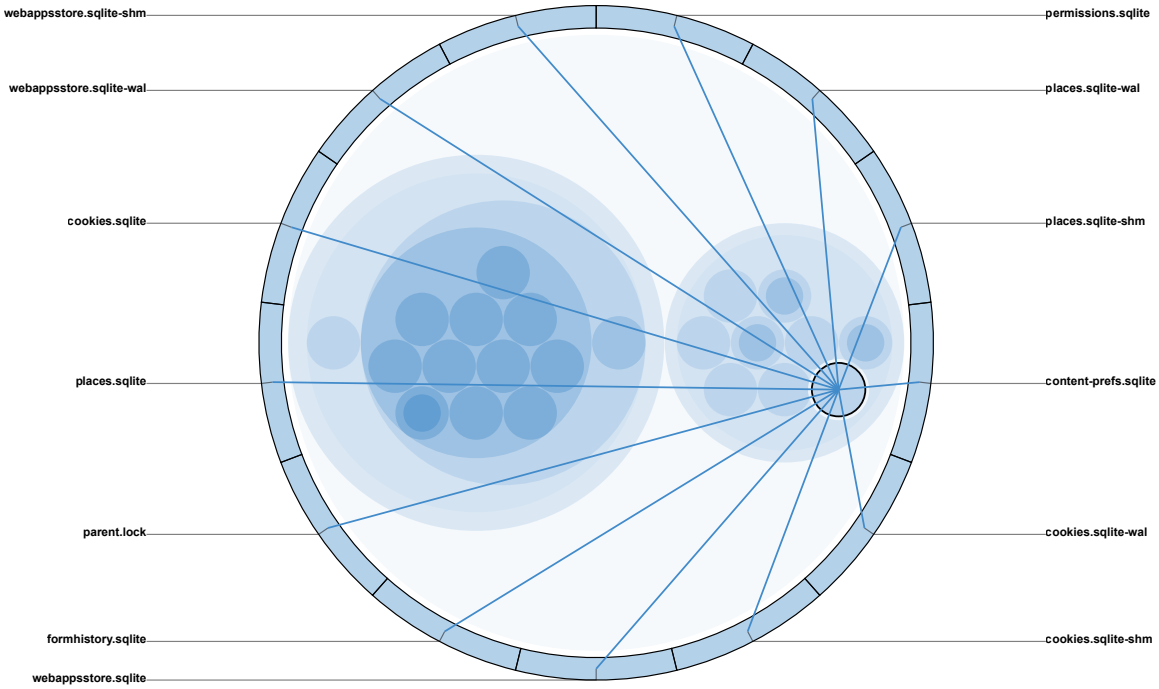


Figure 57. Firefox.exe Resource Link to 4a4novg1.default Profile.

3.4.3 Malware Detection.

Detecting live malware on a system is one of the listed research goals. The following sections describe how to quickly detect malicious code running on a system. The first example is a rootkit (FUTo) and the second is a remote access toolkit (Poison Ivy).

3.4.3.1 FUTo.

FUTo uses Direct Kernel Object Manipulation (DKOM) methods to obscure specified objects within the PspCidTable. Russinovich and Solomon [57] refer to the PspCidTable as a "handle table for process and thread client IDs". As such, Silberman [58] points out, "Every process PID corresponds to its location in the PspCidTable". FUTo removes references to not only itself, but to each process it wants to hide. Furthermore, it sets up a process notify routine which will add the process references

back to the PspCidTable before closing a hidden process. This stops the system from deferencing a null value and ultimately a "Blue Screen of Death" [58].

To begin analysis, the researcher selects the Image 'FUTo' from the drop-down menu and clicks 'Visualize Dataset'. When the visualization has finished loading, the researcher clicks 'Begin Memory Analysis'. From the beginning, this dataset differs from standard Microsoft Windows XP images. The 'System Idle Process' has three child processes as seen in Figure 58 rather than the normal two (`System` and `Explorer`) as shown in Figure 62. To further confirm the odd behavior, the researcher clicks 'Toggle Whitelisting' and selects an acceptable confidence percentage. The resulting view in Figure 59 shows five processes that do not meet the confidence percentage, one of them being the third child process of `System Idle Process`. Performing a mouse-over on the suspicious process node as seen in Figure 60 provides the name `BadProcess.exe` and PID '0' which is reserved for `System Idle Process` as seen in Figure 61. A process hidden by the FUTo rootkit can have any name, but it will always be associated with PID '0'.

This visualization tool is capable of showing all loaded modules, dynamic-link libraries and custom registry keys unique to each implementation of FUTo. Since the process hidden by FUTo is associated with `System Idle Process` (i.e., appears as a loaded module). All artifacts of interest are also associated with `System Idle Process`. It is important to note the textual representation of these artifacts was not detailed in this document due to their length. It is possible to locate all known artifacts associated with the FUTo rootkit using the same techniques shown in this use case.

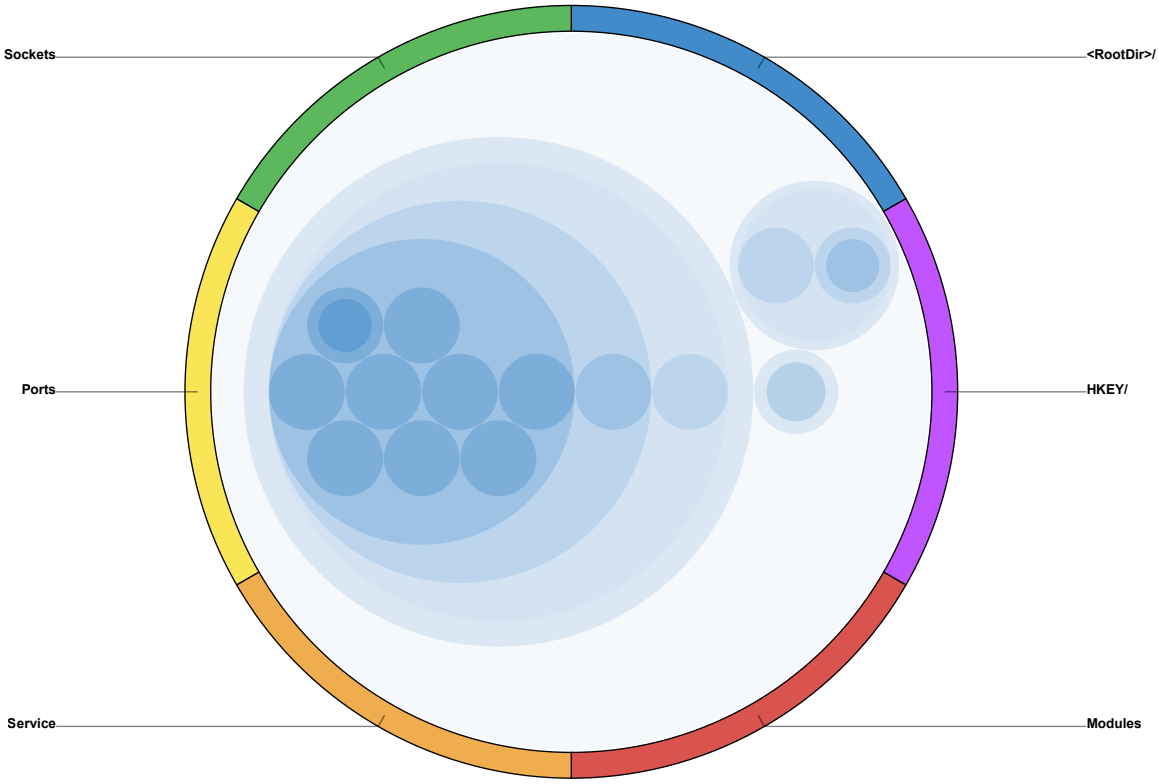


Figure 58. FUTo Image Visualized.

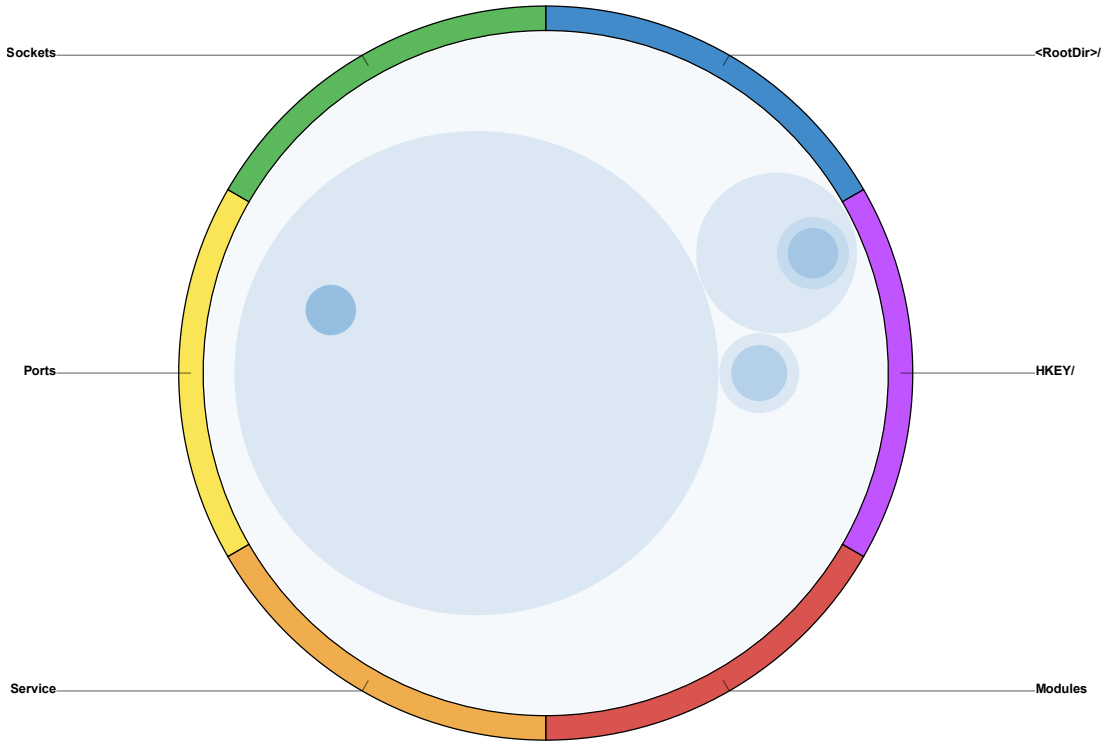


Figure 59. FUTo Image Visualized with Whitelisting.

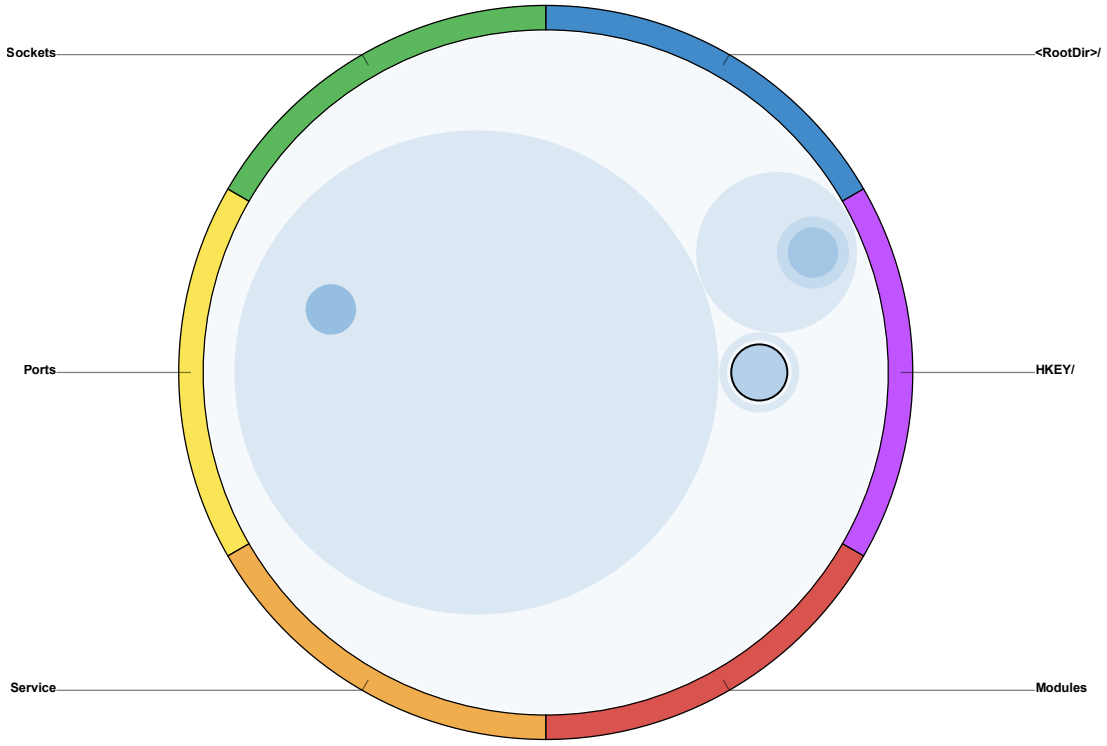


Figure 60. FUTo Image Visualized with Whitelisting BadProcess.exe Selected.

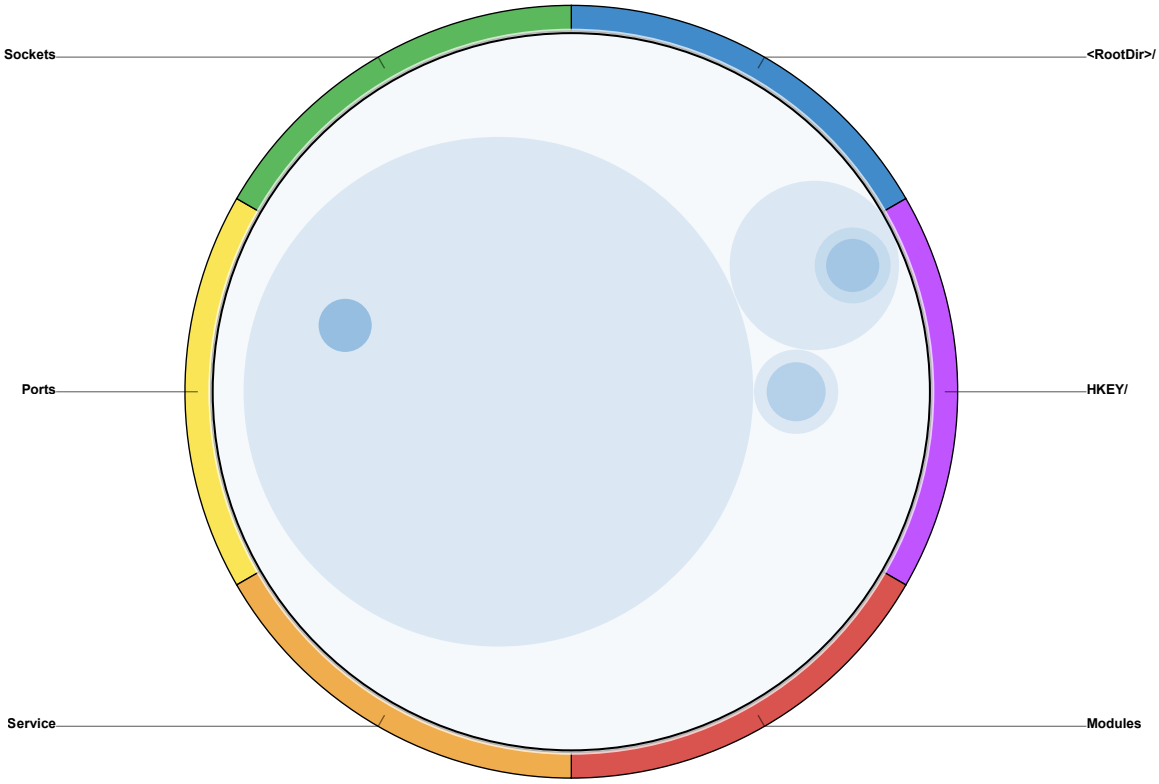


Figure 61. FUTo Image Visualized with Whitelisting System Idle Process Selected.

3.4.3.2 Poison Ivy.

`Poison` Ivy[59] is a remote access toolkit. It is typically delivered through social engineering (i.e., email attachment, web download, or file service.). Once initiated, the code executes in stages. The first stage, named 'initialization and maintenance' by FireEye Inc. researchers [59], injects its code into the `Explorer.exe` process (if persistence is enabled, a watchdog thread is included). The second phase, dubbed 'Network Code' by FireEye, can come in various configurations. By default, it starts a hidden instance of the system's default browser process and injects itself into that hidden process. The code then downloads all additional code and data required for functionality from the attacker's remote client[59].

To begin analysis, the researcher selects the Image 'RAT' from the drop-down menu and clicks 'Visualize Dataset'. When the visualization has finished loading, the researcher clicks the 'Begin Memory Analysis'. At the initial view of this dataset shown in Figure 62, nothing stands out as suspicious.

To limit the scope of the analysis, the researcher clicks 'Toggle Whitelisting' and selects confidence percentage, in this case 65%. The resulting view in Figure 63 shows five processes that do not meet the confidence percentage. Explore each process and its links by toggling links on and selecting the process nodes one at a time.

Under the system processes there is an instance of `svchost.exe` (which is the service host process) with three children: two instances of `wuauclt.exe` and `wscntfy.exe` seen in Figure 64. The process `wuauclt.exe` is the Windows Update Automatic Update Client and `wscntfy.exe` is the Windows Security Center Notify Application. Both should run from `C:\Windows\System32\`.

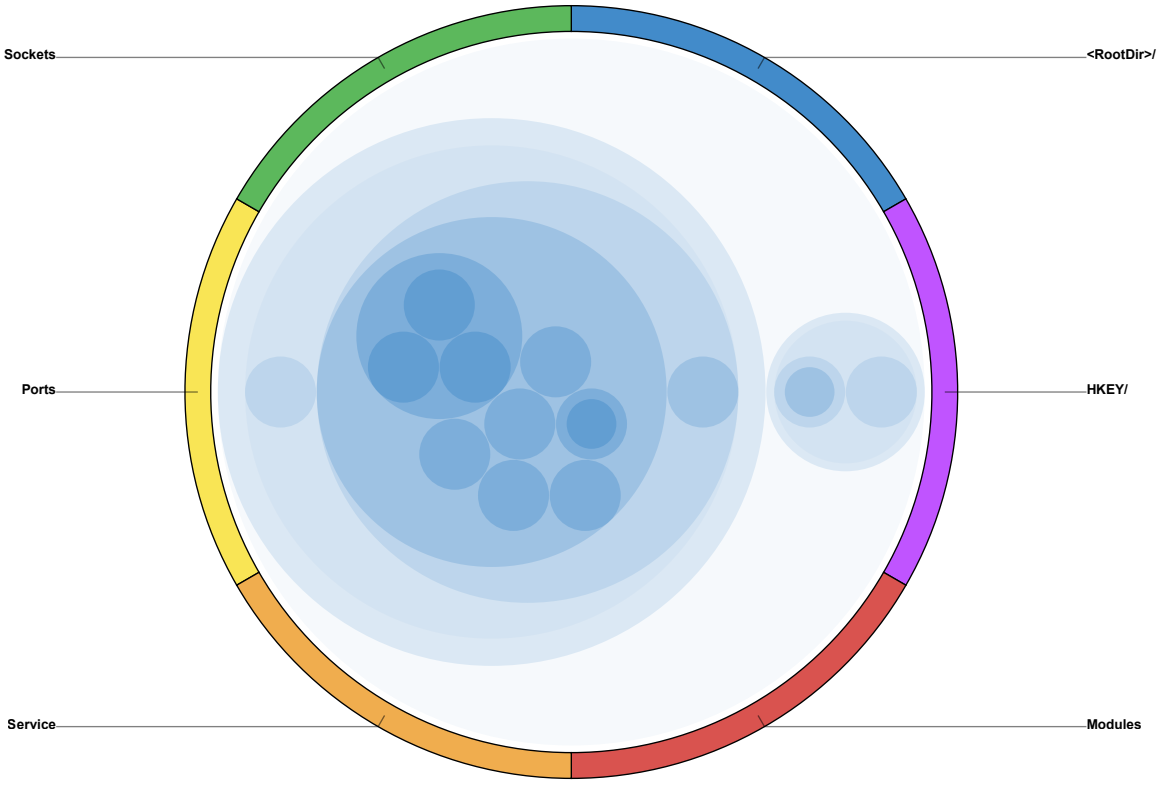


Figure 62. Poison Ivy Image Visualized.

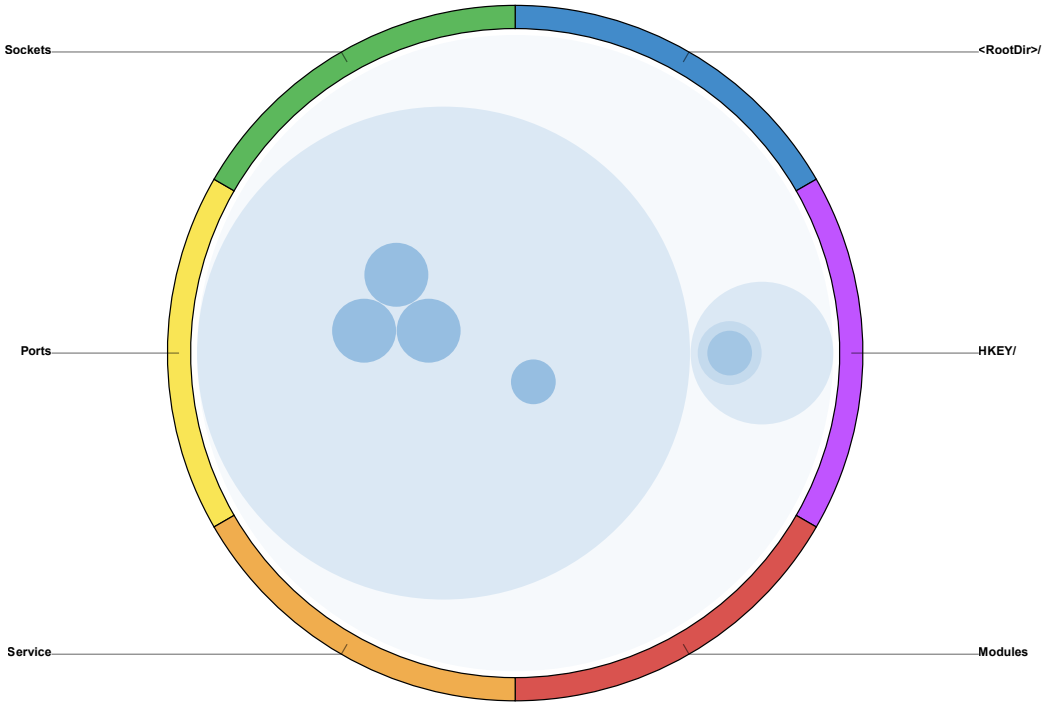


Figure 63. Poison Ivy Image Visualized with Whitelisting.

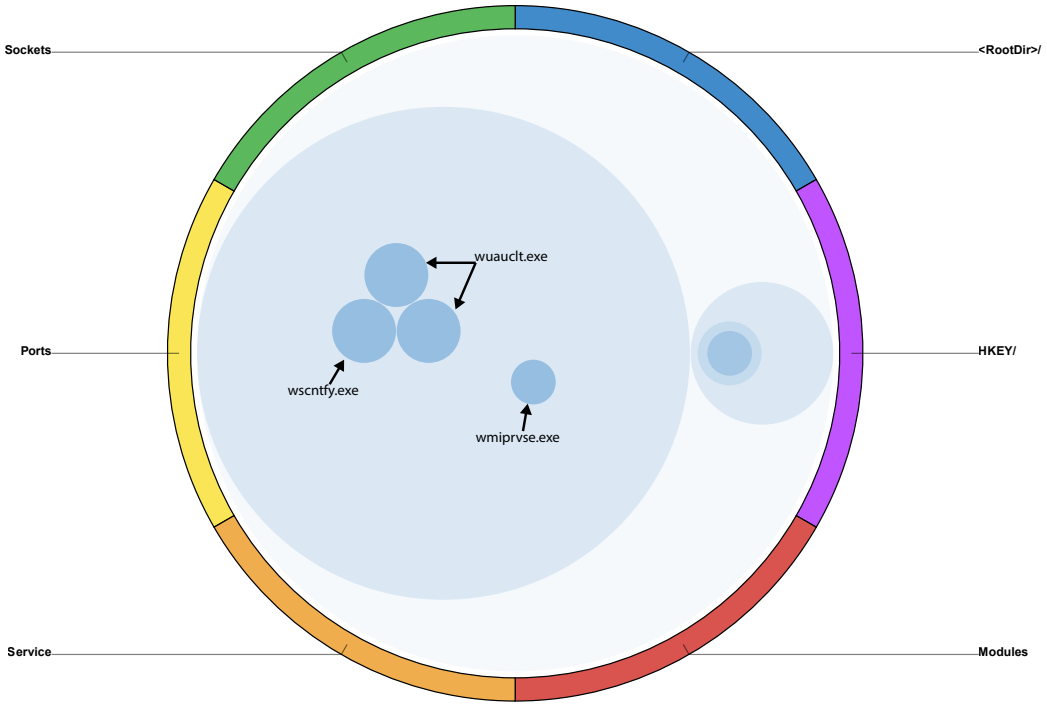


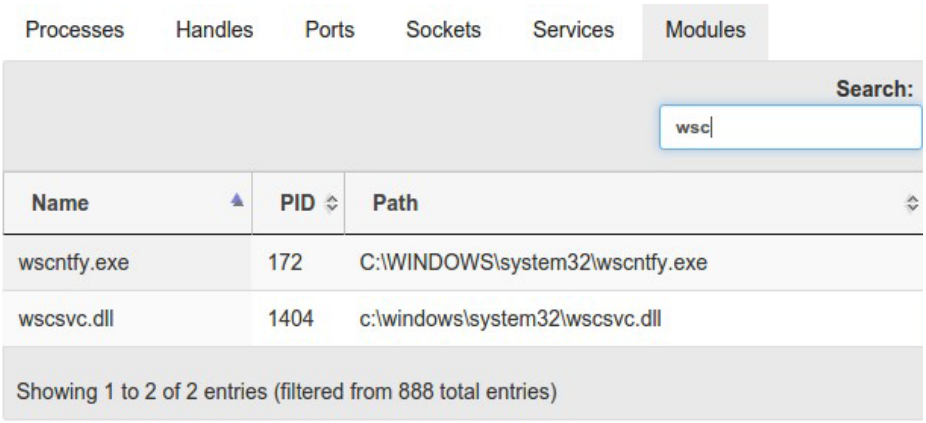
Figure 64. Unmatched System Processes.

Using the textual data view as shown in Figure 65, the researcher confirms both are in fact running from the expected path. Another instance of the service host is running `wmiprvse.exe`, which is the Windows Management Instrumentation process. The process should run from `C:\Windows\System32\Wbem\`. Confirm execution path using the text view as with the others.

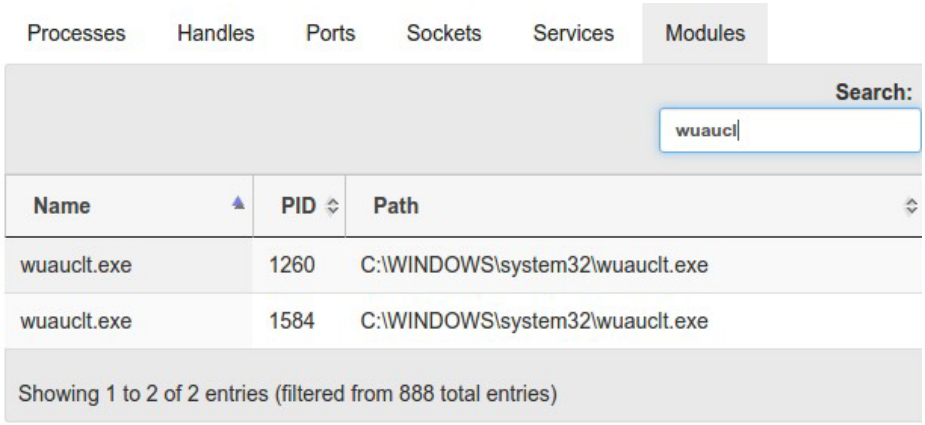
Under the Explorer process in Figure 66 there is one child process, `win32dd.exe`. This process can be ignored as it is the process used to capture live memory by the incident response team. Selecting the `Explorer.exe` process as in Figure 67 identifies links to File, Keys, Modules, Ports, and Sockets. The network connections are abnormal and warrant further investigation. Following the link to network ports shows a single network connection over port 3460.

Returning to the system view and following the link to file handles will step into the root directory. Quickly stepping through the path, `<RootDir>\Windows\System32`, brings up the views seen in Figures 68, 69, 70. The resulting view shows a link to a

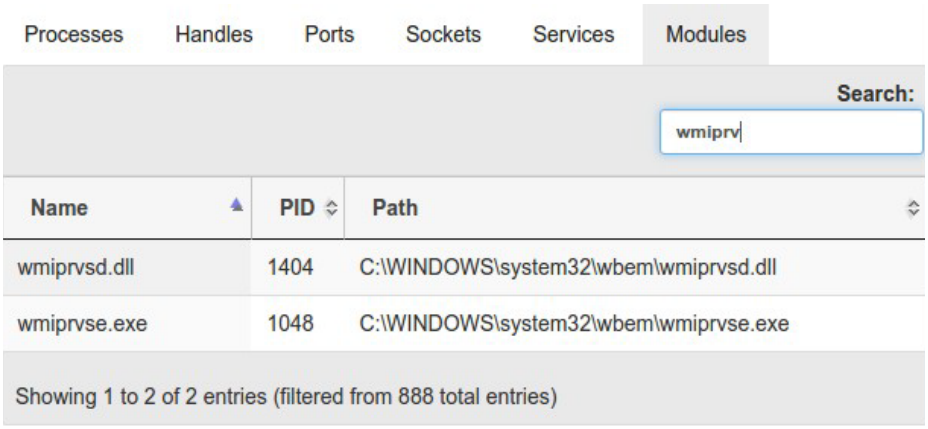
single file handle for `test.exe`. The application `test.exe` is not a Microsoft Windows executable file and should not be in the `System32` directory. These activities are indicative of a Remote Administration Toolkit – in this case, `Poison Ivy`.



(a) Module Path For wuauclt.exe.



(b) Module Path For wscntfy.exe.



(c) Module Path For wmiprvse.exe.

Figure 65. Module Path Identification Using Datatables.

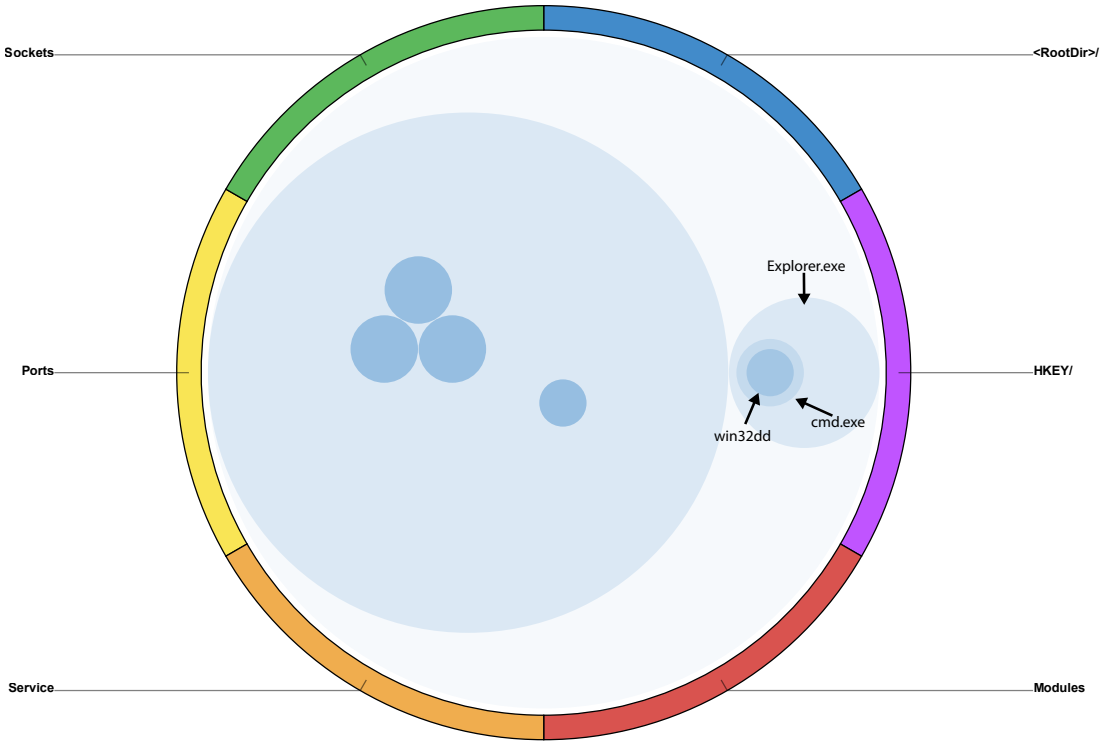


Figure 66. Unmatched User Processes.

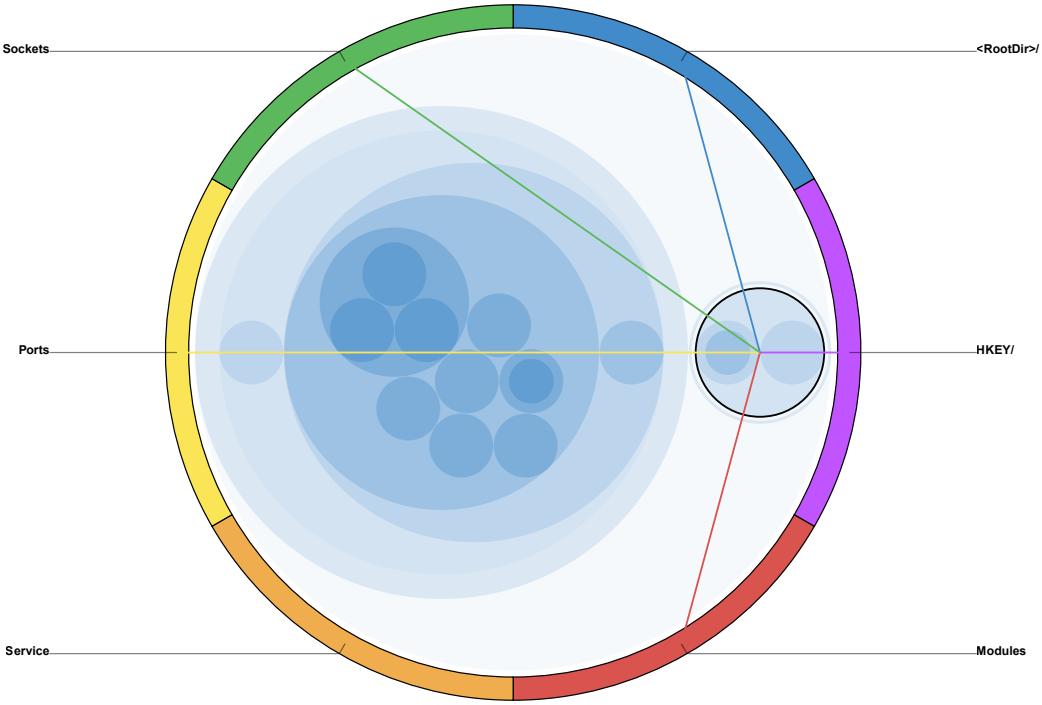


Figure 67. Abnormal Explorer Process.

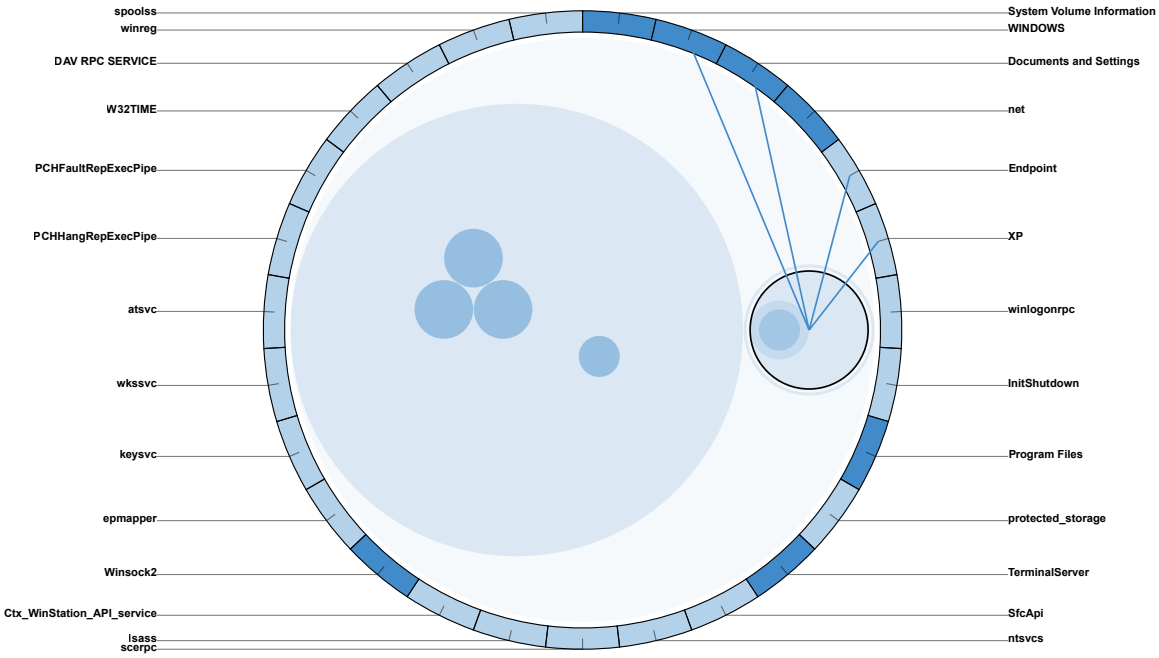


Figure 68. Explorer.exe Resource Links Root.

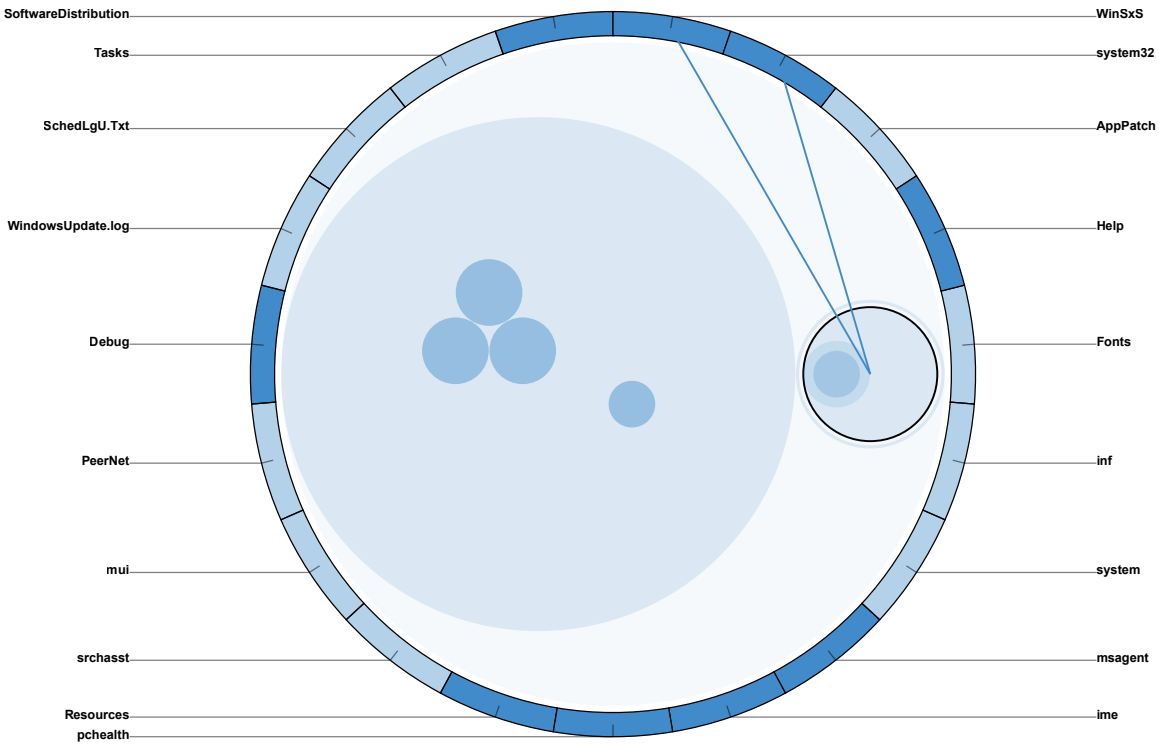


Figure 69. Explorer.exe Resource Links Windows.

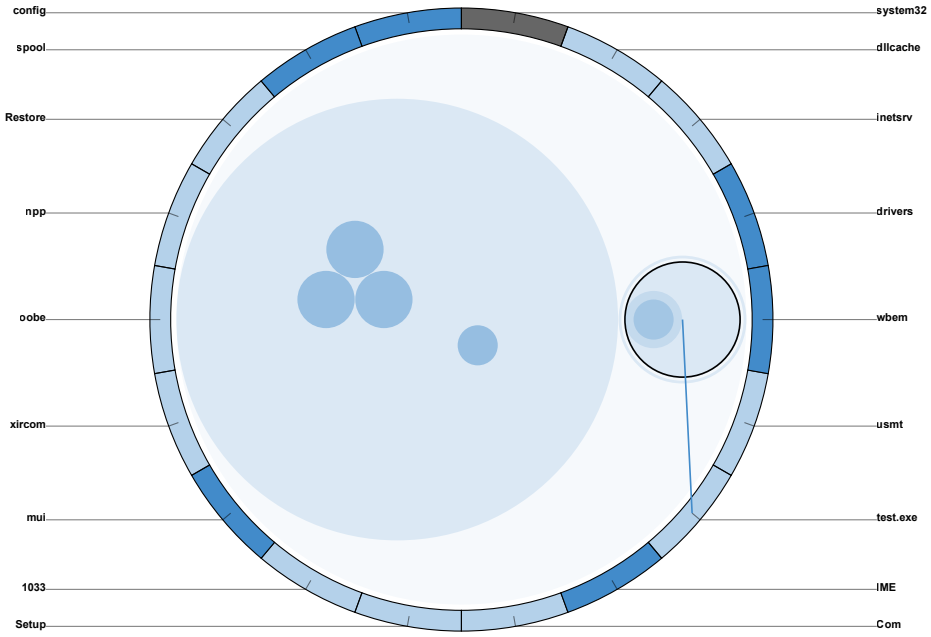


Figure 70. Explorer.exe Resource Links System32.

The memory visualization tool also shows all network connections, loaded modules, threads and modified registry keys, unique to each implementation of Poison Ivy. Note that the textual representation of these artifacts was not detailed in this document due to their length. However, using the same techniques as those presented, each of the artifacts can be located and the textual data view can provide the full details.

3.5 Summary

This chapter introduced the memory visualization tool user interface and provided a user orientation. A discussion of two use cases set an initial basis for examiners to understand how to use the tool when attempting to locate artifacts of interest. The prime focus of this section was artifacts related to user activity and malware. Lastly, this chapter discussed implementation specific details for two images containing malware: `FUTo` and `Poison Ivy`.

IV. Methodology

Digital forensic tools share a common analysis medium, humans. The binary nature of forensic artifact detection presents two distinct outcomes, *Found* and *Not Found*. The merits of a tool cannot simply be measured by how well it detects and presents anomalies in a collection to the human examiner, because each tool also relies on the human examiner, of varying expertise, to correctly determine whether or not an anomaly is an artifact of interest. As such, quantitative methods do not adequately reflect the efficacy of a given tool. This research describes two pilot studies in digital forensics involving human subjects, where memory capture data are analyzed using a qualitative methodology known as content analysis.

This chapter details two separate pilot studies seeking to document the efficacy of the memory visualization tool. Data collection and analysis methods are discussed and outlined. Additionally, sample exercise material is provided and discussed.

4.1 Pilot Studies

To test the efficacy of the memory visualization tool, two pilot studies, involving human participants, were designed to examine the core principles of the tool. The first study sought to determine if the memory visualization tool improved the analysis and understanding of hierarchical and linked data in memory captures. Along the same line of inquiry, the second pilot study extended the first and introduced the whitelisting functionality. The second study sought to examine if the whitelisting process improved the accuracy of artifact identification and reduced analysis time.

In both studies, the memory visualization tool was compared with a traditional textual data approach. While the data was limited to a text view, users were free to employ any search, sort and filter functions available in commercial or open-source

text tools (e.g., Microsoft Office, Libre Office, grep, sort). Both methods used the same source data.

The pilot studies also examined a single user experience objective. As the primary objective, researchers evaluate whether or not the memory visualization tool increased the participant's understanding of the data through simultaneous visualization of global and local views. As a secondary objective, researchers examine whether or not intuition played a role in a participant's understanding of the data.

To evaluate the research goals, researchers collected data from the written submission of each exercise along with participant feedback. The written submission evaluates a participant's successful completion of a forensic exercise and is graded for accuracy. In the second pilot study, individual time metrics are also collected for each exercise.

One of two post-study surveys are administered for all participants. In the first pilot study, participants only respond to the visualization method survey. For the second pilot study, participants respond on either the text-based method survey or the visualization method survey, based on which group the participant was assigned. Each survey question focused on the participant's observations of the method, tool or perception of their own performance. Additionally, a single user experience question focused on the participant's perceived understanding of the data. The final question in the survey, sought to draw out future work and/or recommended modifications to the user interface.

4.2 Experimental Procedures

During the exercise portion of the pilot studies, each participant is presented with data from several fictitious scenarios. Some of the data involves scripted criminal activity, while other data represents normal user activity on a computer. Each par-

ticipant is required to answer questions pertaining to the data. The pilot studies share some data, but the data acquisition, scenario and questions differ. Scenarios are described in Table 6.

Table 6. Fictional Scenario Descriptions.

Name	Description
Scenario One	Consists of three machines running Microsoft Windows XP. Only two of the three machines are accessible to the incident response team. One of the machines is running an instance of Poison Ivy[59]. The other machine only contains normal user activity.
Scenario Two	Consists of five machines running Microsoft Windows XP and three servers running Windows 2008 Server. All machines are accessible to the incident response team. Each machine contains normal user activity along with some questionable activity. One machine is exploited using a Meterpreter[60] Reverse TCP Shell.
Scenario Three	Consists of a single machine running Microsoft Windows XP. The user is browsing virus writing tutorial websites with the Firefox browser. The use has the Netbeans Java IDE open and is currently editing a file called "NewVirus.java". Adobe Acrobat is open reading a document named "How_To_Write_A_Virus.pdf". Lastly, Microsoft Word is open and currently editing a file named "The_Secret_Plan.docx".
Scenario Four	Consists of a single machine running Microsoft Windows XP. The machine is running an instance of BadProcess.exe hidden by FUTo[58] rootkit. This scenario is used exclusively with data from scenario one in section two of the exercise during the second pilot study.

4.2.1 Pilot Study One.

The first pilot study uses five members of a graduate level, Introduction to Cyber Forensics course. The participants, split into a team of two and a team of three, are asked to perform two separate incident responses. Using the data extracted from each fictional crime scene, the teams complete data analysis, artifact identification, and reconstruct the timeline and events through detailed narrative.

In scenario one, participants are limited to text-based analysis tools. In scenario

two, participants are required to use the memory visualization tool. After each scenario, team members submit a collaborative report which is graded for completeness and accuracy of artifact identification. All participants are surveyed using the memory visualization tool survey shown in Figure 71.

Memory Visualization Tool Survey

The purpose of this survey is to accurately assess the Memory Visualization Tool and its effect on analysis time and accuracy of artifact identification. Please provide complete and honest feedback.

What items in the visualization made completing the task easier?

What items in the visualization made completing the task more difficult?

How do you perceive your accuracy in the task was impacted by the visualization? And, why?

How do you perceive your speed to complete the task was impacted by the visualization? And, why?

Did using the tool increase your understanding of the process and data being analyzed?

What would you like to see changed with the visualization?

Figure 71. Visualization Tool Post-Study Survey.

4.2.2 Pilot Study Two.

The second pilot study utilizes eleven master's degree students from the graduate cyber operations program. These participants self-identified as knowledgeable in the subjects of computer operating systems, computer networking, and malware. Participants are split into two groups. Participants in the first group must use text-based tools. Participants in the second group must use the memory visualization tool with the whitelisting function. All participants are provided data from scenarios one, two and three and asked various open-ended questions pertaining to the data. Each question is timed.

In section one of the exercise, participants are presented with data from scenario three and asked to provide specific details about the state of a single system. In section two, each participant examines data from scenario one and four and is required to answer questions about the presence of malware. In section three, participants are presented data from scenario two and asked to report anything suspicious. Each exercise is graded for completeness and accuracy. Time and accuracy results are compared between groups. Each participant completes the survey associated with the group to which they were assigned. Members of the visualization group respond on Figure 71 and members of the text-based methods groups respond on Figure 72.

Memory Visualization Tool Survey

The purpose of this survey is to accurately assess the Memory Visualization Tool and its effect on analysis time and accuracy of artifact identification. Please provide complete and honest feedback.

What text-based tools/functions made completing the task easier?

What about text-based tools made completing the task more difficult?

How do you perceive your accuracy in the task? And, why?

How do you perceive your speed to complete the task? And, why?

Did using text-based tools increase your understanding of the process and data being analyzed?

What would have helped you perform better?

Figure 72. Textual Methods Post-Study Survey.

4.3 Scenario-Based Memory Captures

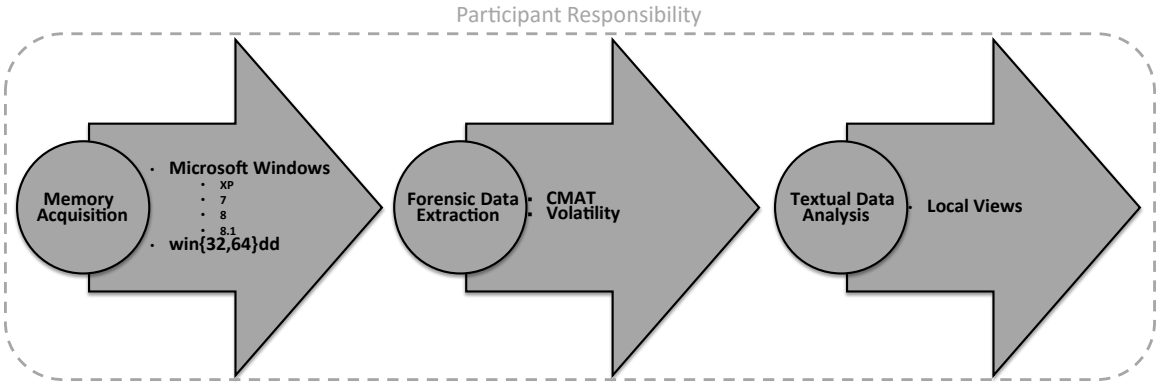
All raw memory capture data are collected from the scenarios described in Table 6. Each scenario is considered a live environment and as such, all machines are in

the state specified by researchers. Live memory captures are obtained using win32dd. Individual memory captures are processed with CMAT [24] and the output memory feature files, which are described in Table 7 are placed in a uniquely named image directory. Each directory containing memory feature files are uploaded individually as a single composite image into the memory visualization image database. Figure 73 depicts the memory image acquisition process used by incident response teams during the pilot studies. During the first pilot study, participants follow Figure 73a for exercise one and Figure 73b for exercise two. For the second pilot study, participants in the textual methods group follow Figure 73c, while participants in the visualization tool group follow Figure 73d.

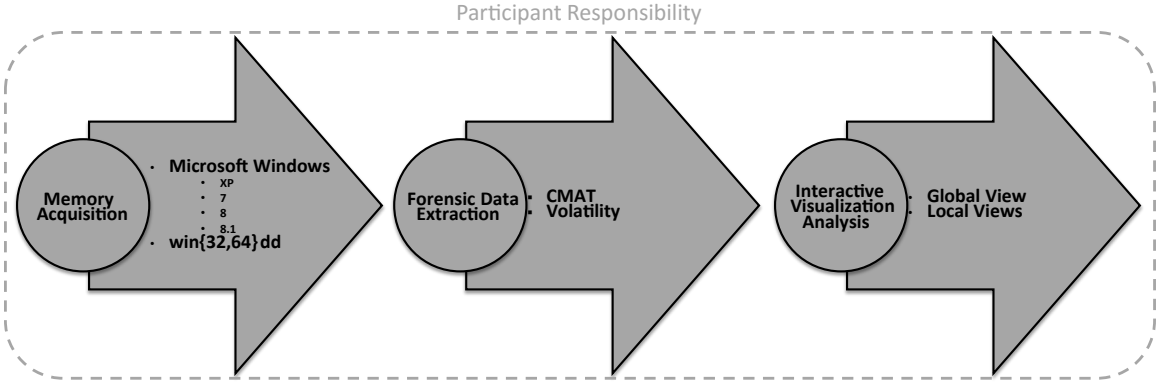
Table 7. Memory Image Feature Files.

Name	Description
Processes	Lists all running processes including name, PID, and PPID
Handles	Lists all file, registry key, directory and thread handles by associated PID.
Modules	Lists all modules including name, path and PID.
Connections	Lists all network connections by PID including local and remote port, protocol, and socket information.
MemDump	Lists system information including processor, operating system, major version, minor version, build, PAE, kernel base, page directory base, machine name and image date.
Services*	Lists all running services including service name, path, and PID.

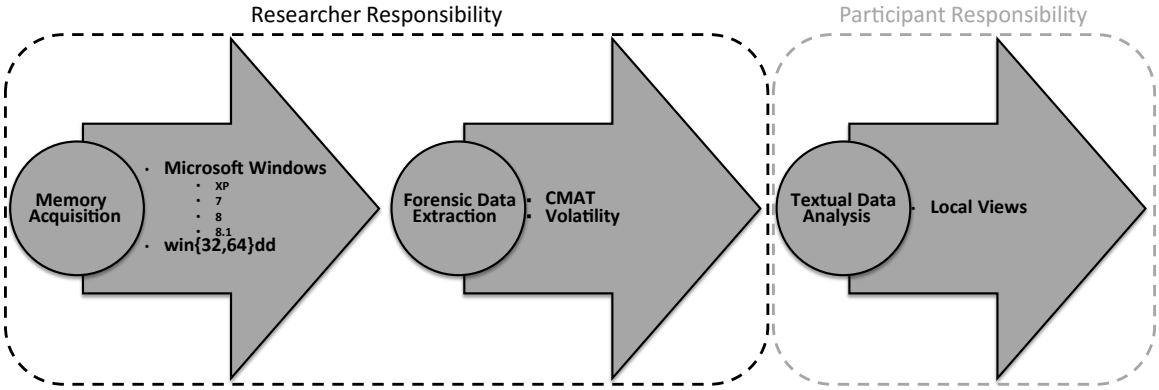
** Services acquired using command 'tasklist /svc /fo csv' not with CMAT*



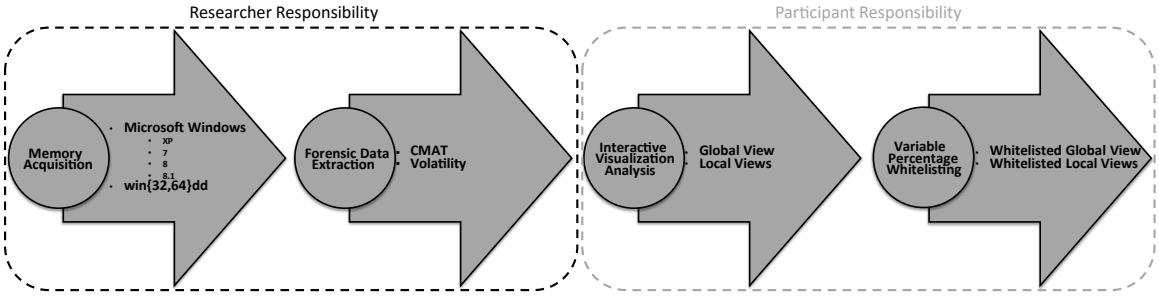
(a) Memory Acquisition Textual Analysis.



(b) Memory Acquisition Visual Analysis.



(c) Memory Textual Analysis Only.



(d) Memory Visual Analysis Only.

Figure 73. Pilot Study Memory Acquisition and Analysis.

4.4 Data Collection Methods

The independent variable in these studies was which tool a participant used. In the first pilot study, participants used both tools (repeated measures). During the second pilot study, half of the participants used the text-based tools and half of the participants used the visualization tool (between groups). The controlled variable was the graded exercise. The dependent variables were survey responses, accuracy of artifact identification and measured time to complete the exercise (only used in the second pilot study).

Time was self reported by participants. Each participant was provided a stop watch and asked to record time in the format 'hh:mm:ss' (h = hour, m = minute, s = second). The accuracy of exercise completion was graded based on researcher knowledge of the scenario. Incompleteness was graded as incorrect.

The post-exercise surveys introduce the qualitative aspect of the pilot studies. The survey responses are the primary interest of the researchers. Each response to the open ended questions are analyzed for keywords and themes. Accuracy and time information are used to bolster survey content by assigning more weight to responses from higher scoring participants.

4.5 Assumptions

1. Digital forensic methods exist for obtaining an accurate physical memory image.

2. The extracted datasets under consideration for each system are limited to process lists, network connections, system services, open file handles, system registry keys, and loaded modules.

3. The operating systems under consideration are limited to Microsoft Windows Operating Systems.

4. The primary examiner(s) have knowledge of the laboratory exercises with which to evaluate accuracy of artifact identification.

5. Examiner(s) have a scholastic background (i.e., knowledge of operating systems, malware, networking and incident response) and a refresher lesson in analysis techniques in order to reduce learning effects during the experiment.

4.6 Hypotheses

The primary hypothesis of these pilot studies is that the memory visualization tool would produce more accurate artifact identification than traditional text-based methods. It was hypothesized that any improved accuracy over text-based methods was due to the visualization tool's ability to simultaneously display hierarchical and associative relationships (i.e., simultaneous global and local view). It was also hypothesized that simultaneous global and local views led to better understanding of the data.

The secondary hypothesis, tested only in the second pilot study, is that filtering items of little interest from view, using the visualization tool's whitelisting function, would reduce time taken to complete a task. It was hypothesized that a reduced workload would in turn reduce time spent on a task.

V. Results

Evaluation of the visualization uses two qualitative pilot studies involving human subjects. During the course of our pilot studies, we collected two forms of data from participants. Each participant (or team) submitted a written solution to a forensic exercise. Secondly, each participant completed an open response survey. The written solutions received a percentage grade for completeness and correctness. The surveys were analyzed for content and themes. We drew conclusions about the efficacy of the memory visualization tool from both sources of data.

This chapter focuses on the results of two pilot studies conducted using human subjects. The results from each study are first analyzed within their respective studies. Additionally, data from participants using the memory visualization tool are analyzed between studies. Findings are compared to the hypotheses established in the previous chapter

5.1 Data Analysis

The source data in the first pilot study consists of two graded exercises and the post-study survey. The source data from the second pilot study consists of a single graded exercise and one of two post-study surveys. The quantitative scores from the exercises play a supporting role to the qualitative survey responses.

We present the quantitative results using basic visualization techniques. Qualitative results are presented through themes extracted from the survey responses. Themes are extracted using word and phrase count techniques, which take into account the use of synonyms.

5.1.1 Pilot Study One.

In this study, there were four pieces of quantitative data, which are listed in Table 8. This data suggests that the visualization tool produces more accurate results.

Table 8. Pilot Study One Quantitative Scores.

Submission	Analysis Type	Score
Team 1 Exercise 1	Text-based Methods	88%
Team 2 Exercise 1	Text-based Methods	89%
Team 1 Exercise 2	Visualization Method	98%
Team 2 Exercise 2	Visualization Method	98%

To explore these initial findings, we examine the survey responses. A basic starting point for content analysis is word usage. Figure 74 was generated with Wordle[61] using only the responses from the post-study survey. This form of word analysis is a helpful starting point, but we need to be more selective in order to find themes. I limited the words to verbs, adjectives and nouns related to the visualization tool, producing the list found in Table 9.

Figure 74. Pilot Study One Word Cloud.

Table 9. Most Frequently Used Words.

NO. Occurences	Word	NO. Occurences	Word
6	helpful	1	ownership
6	easier	1	owner
5	able	1	nesting
4	identify	1	looking
3	speed	1	look
3	reference	1	levels
3	helped	1	level
3	faster	1	knew
3	easy	1	intuitive
2	suspicious	1	improving
2	represent	1	improvements
2	links	1	implies
2	increased	1	identifying
2	improved	1	hierarchy
2	bubbles	1	easily
1	visually	1	ease
1	visible	1	distinguish
1	viewing	1	detect
1	useful	1	context
1	unique	1	containers
1	understanding	1	connections
1	understand	1	connection
1	toggle	1	connecting
1	switching	1	connectedness
1	switch	1	connected
1	programs	1	comparison
1	presented	1	compared
1	positively	1	bubble
1	positive	1	believe

The themes in the first pilot study align with the hypothesis. Improved accuracy with the visualization tool is attributed to its ability to represent hierarchical and linked data simultaneously. Additionally, this representation of data helps the user to better understand the data and apply intuition. Referring back to Table 9, it becomes clear that these themes are present.

When discussing which components of the visualization tool made the tasks easier, one participant responded, "links between the process and system resources." While another participant added, "nesting of the process bubbles." Both of these inputs identify visual representation of hierarchical and linked data within the visualization tool. However, a third participant had affirmed our hypothesis by writing, "[The visualization tool] helped greatly to see ownership (hierarchical) and connectedness of processes and system resources." Survey responses indicate that the participants recognize our method, but it remains to be seen whether or not the methods lead to a better understanding of the data.

We asked our participants how they perceived their accuracy of artifact identification, keep in mind that they have not yet seen their scores. The responses were overwhelmingly positive. One participant claimed to be, "confident", that they "knew potentially infected machines within a few minutes." Along a similar line, participants claimed the visualization tool made it easy to "identify items that should not be on the system." or "believe if a machine was clean because nothing appears unusual." Word usage by the participants (e.g., knew, believe, appears) implies understanding and the quantitative data shows an improvement in scores, but we need to go a step further.

Focused on a specific point, we asked our participants if the visualization tool increased their understanding of the process and the data. Two of the participants thought the visualization tool had no impact on their understanding, but that it made the process "faster" and "easier". The remaining three participants agreed that the visualization tool improved their understanding. They thought the visualization tool "was intuitive", it allowed the user to "visually distinguish" objects that "appear suspicious."

The results of the initial pilot study were positive. Participants showed improve-

ment quantitatively in the exercise scores, but more importantly, qualitative analysis of the post-study survey supported key themes of our hypothesis. These results led to minor changes in the visualization tool's user interface and initiated a second pilot study to confirm the initial findings and test an additional hypothesis.

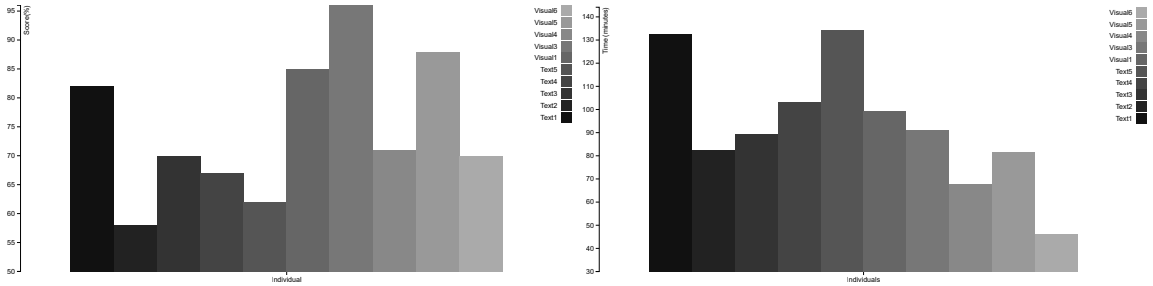
5.1.2 Pilot Study Two.

Our second pilot study contained twelve pieces of quantitative data shown in Table 10: six graded questions and six associated time recordings. After initial review of the quantitative data, we removed the second visualization participant from the study. It was clear to us that this participant did not possess the required knowledge to successfully complete the forensic exercise. We formed this conclusion based on their low score, slow time, and comments from their post-study survey. No other outliers were seen in the quantitative data.

Figures 75a and 75b show individual scores and times respectively. The quantitative data suggests that participants using the memory visualization tool scored higher and had faster completion times than their counterparts using text-based methods.

Table 10. Pilot Study Two Scores and Time By Participant.

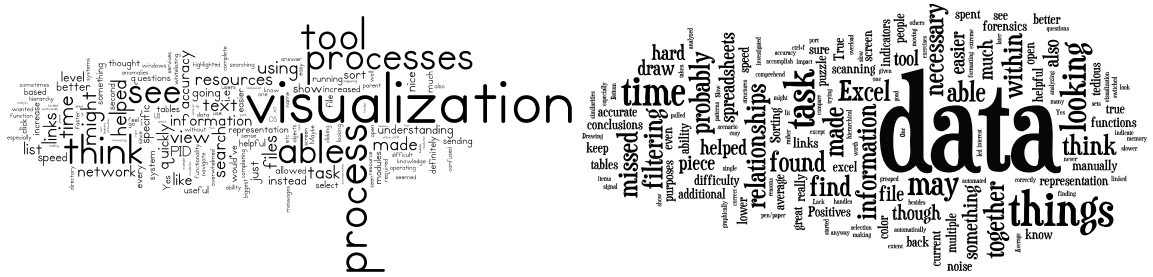
	Question 1		**Question 2**		**Question 3**		**Question 4**		**Question 5**		**Question 6**		Average Total	
Participant	Score	Time	Score	Time	Score	Time	Score	Time	Score	Time	Score	Time	Score	Time
Text1	1.00	0:15:15	1.00	0:12:52	1.00	0:06:04	1.00	0:06:12	0.44	0:42:29	0.50	0:49:43	0.82	2:12:35
Text2	0.85	0:07:12	0.50	0:12:50	0.50	0:01:08	0.88	0:05:30	0.78	0:27:46	0.00	0:28:01	0.58	1:22:27
Text3	0.69	0:03:14	0.50	0:20:26	1.00	0:01:55	0.94	0:04:24	0.56	0:20:13	0.50	0:39:18	0.70	1:29:30
Text4	0.92	0:07:20	0.50	0:07:20	1.00	0:07:20	0.63	0:07:20	0.22	0:44:30	0.75	0:45:00	0.67	1:43:01
Text5	0.92	0:11:15	0.50	0:11:15	1.00	0:11:15	0.94	0:11:15	0.33	0:29:10	0.00	1:00:00	0.62	2:14:10
Visual1	0.92	0:01:30	1.00	0:06:45	1.00	0:04:13	1.00	0:09:35	0.67	0:38:28	0.50	0:38:41	0.85	1:39:12
Visual3	1.00	0:08:11	1.00	0:09:27	1.00	0:02:03	1.00	0:05:43	1.00	0:31:27	0.75	0:34:19	0.96	1:31:10
Visual4	0.77	0:02:33	0.25	0:16:30	1.00	0:01:28	0.81	0:04:59	0.44	0:15:28	1.00	0:26:56	0.71	1:07:54
Visual5	1.00	0:09:30	1.00	0:10:30	1.00	0:03:00	1.00	0:10:00	0.56	0:21:30	0.75	0:27:00	0.88	1:21:30
Visual6	0.69	0:00:30	0.50	0:10:00	1.00	0:01:15	0.94	0:11:50	0.33	0:03:00	0.75	0:19:30	0.70	0:46:05
Text Average	0.862	0:06:06	0.6	0:11:43	0.9	0:05:10	0.875	0:07:37	0.511	0:32:01	0.35	0:42:12	0.683	1:41:40
Visual Average	0.877	0:04:27	0.75	0:10:38	1	0:02:24	0.95	0:08:25	0.6	0:21:59	0.75	0:29:17	0.821	1:17:10



(a) Scores By Individual (b) Times By Individual

Figure 75. Pilot Study Two Score and Time Charts By Individual.

As with the first pilot study, we begin the qualitative analysis by examining the survey responses at the most basic level. Figures 76a and 76b show the most commonly used words while Tables 11 and 12 provide the frequency of usage. It is interesting to note, the most dominant word in the text-based methods surveys is *data*. Using this information, we confirmed the presence of our themes in the survey data and continued the analysis.



(a) Visualization Method Survey Word Cloud (b) Textual Methods Survey Word Cloud

Figure 76. Pilot Study Two Word Clouds.

The themes in the second pilot study were identical to those of the first study and include a third theme: time. We attribute reduced time between artifact identification to the efficacy of the whitelisting feature. As in the first study, we analyze the content of the open response surveys.

Table 11. Most Frequently Used Words Visualization Survey.

NO. Occurences	Word	NO. Occurences	Word
6	able	1	notice
4	understanding	1	located
3	links	1	leveled
3	helped	1	level
2	view	1	interacting
2	quickly	1	interacted
2	hierarchy	1	integrated
2	helpful	1	information
1	whitelisting	1	increase
1	whitelist	1	help
1	visually	1	filtering
1	understand	1	filter
1	tree	1	efficient
1	showing	1	easier
1	show	1	connected
1	shading	1	bubbles
1	perceive	1	aided

Table 12. Most Frequently Used Words Text-Based Methods Surveys.

NO. Occurences	Word	NO. Occurences	Word
19	data	2	tables
6	time	2	sorting
6	task	2	scanning
6	excel	2	representation
5	filtering	2	puzzle
4	within	2	pen
4	together	2	paper
4	think	2	organization
4	relationships	2	manually
4	information	2	links
4	found	2	indicators
4	find	1	scattered
4	between	1	overload
3	spreadsheets	1	filters
2	text	1	filter
2	tedious	1	distracting

We asked the five participants using the memory visualization tool to identify which features made the tasks easier. One participant thought the process node visualization (i.e., circle packing) was the most helpful stating, "it allowed me to very easily see what a particular process was using" while two others stated the "automatic links to resources" or "link view" was the most helpful. Another participant wrote the "leveled local views", using the resource circle, simplified the tasks. We interpret "leveled local views" to mean global view, because a single local view does not have levels. The most affirming response stated, " [T]he whitelisting tool was very useful, but it made me nervous to turn it up too far since I was worried that I might miss something."

Looking at the perceived accuracy in artifact detection of our participants, the findings were very similar to the first study. All five participants stated that the visualization tool increased their accuracy. One of the participants stated, "Without the visualization, I probably would not have been able to complete any of the tasks...', while another wrote "I can't imagine using spreadsheets" to complete the tasks. There were also concerns that the visualization tool made artifact discovery too easy and as one participant wrote, "[T]he tool made me more confident than I should have been...I might have rushed through the data too fast and missed something." Two participants noted the visualization tool improved their accuracy by making "anomalies more obvious" and helping spot "odd behavior" by a process.

We then shifted our focus to how they perceived their time on task. One participant thought the visualization tool "definitely" increased completion speed noting, "I was able to, in a few seconds, locate suspicious processes". Three other participants attributed their speed to the simultaneous global and local views. One participant wrote, "Being able to see the links throughout the UI and navigate through the hierarchy allowed for a better understanding." A second participant stated, "The [resource]

links and process bubbles made some of the questions no brainers", while a third felt "the visualization helped me literally see connections and dig into specific resources with more clarity than I think I would have otherwise."

When asked if the visualization tool increased their understanding of the process and data being analyzed, the participants gave a resounding "yes". The participants agreed the visualization tool helped them "see" or "understand" the data better. The visualizations allowed them to understand "how objects were connected" as opposed to making the connections manually. One participant stated, "the visualization tool did seem to direct me" and "it increased my knowledge, by allowing me to see at a glance" the internal workings of an active system. The results of the second pilot study were mostly positive. Participants using the memory visualization tool showed higher scores and lower completion times than their counterparts using text-based methods. More importantly, key themes in the survey data supported our primary hypotheses. While these responses did not support our hypothesis that the faster analysis times were attributable to the whitelisting feature, they do support the hypothesis that the simultaneous global and local views improve understand of the data and allow the users to apply intuition. Furthermore, these results provide the level of confidence required to move forward from pilot studies into real world testing.

5.2 Summary

This chapter details the results of two pilot studies testing the efficacy of the memory visualization tool. Quantitative scores compared within the studies provided a basis for evaluation of qualitative survey data. Qualitative themes extracted from the survey data test the primary and secondary hypotheses. The primary objective was to show the memory visualization tool produced more accurate artifact identification than traditional text-based methods. The secondary objectives were to associate im-

proved accuracy and reduced completion time with the tool's simultaneously global and local views and whitelisting feature respectively.

Quantitative data from both pilot studies showed higher accuracy in artifact identification when using the memory visualization tool. This data was supported by themes extracted from the post-study response surveys. These confirmed the primary hypotheses that the memory visualization tool would produce more accurate artifact identification than traditional text-based methods. Furthermore, the themes supported the secondary hypothesis that any improved accuracy over text-based methods was due to the visualization tool's ability to simultaneously display hierarchical and associative relationships.

In the second pilot study quantitative data suggested higher scores and faster completion times for participants using the memory visualization tool. The themes from the post-study surveys show that participants perceived their completion times as quick, but the themes did not support the hypothesis that filtering items of little interest from view, using the visualization tool's whitelisting function, would reduce time taken to complete a task. This hypothesis deserves further examination.

VI. Conclusion and Recommendations

This research developed and evaluated a fully functional memory analysis tool based on Baum's[9] memory visualization proof-of-concept. Two pilot studies confirmed the efficacy of the visualization tool through qualitative analysis of key themes contained in post-study survey data. Most importantly, this research confirmed the hypothesis a visualization tool that provides context throughout analysis and shrinks an examiner's search space will make an examiner more accurate and faster.

This chapter highlights the accomplishments of this research and concludes with a presentation of future research opportunities.

6.1 Accomplishments

This research built upon a proof of concept and produced a fully functional analysis tool that can run in practically any operating system environment. Using the MEAN Stack single page application model together with several JavaScript libraries, the memory visualization tool is both maintainable and scalable. The memory visualization tool developed through this research provides forensic examiners a full scale analysis client that aids in artifact identification and anomaly detection. The specific objectives met by this research are as follows:

1. The memory visualization tool simultaneously displays hierarchical and associative relationships.

2. A novel, behavioral whitelisting function filters processes of little interest from view.

Two pilot studies concluded the memory visualization tool produced more accurate artifact identification than traditional text-based methods – supporting our

primary hypothesis. A single pilot study showed the memory visualization tool realized faster completion times than traditional text-based methods.

6.2 Future Work

The limiting factor of the memory visualization tool is its reliance on feature files from CMAT. The MongoDB instance is capable of handling the output of all popular memory analysis tools, but a custom schema is required for each. The best solution may be to design a single document format, taking input from all memory analysis tools and mapping them to the predetermined format.

The whitelisting algorithm has a couple shortcomings to address. The current version cannot differentiate between similarly named applications with varying numbers of open network connections. For example, if processes named `Firefox.exe` has three open network connections and all other instances of a process named `Firefox.exe` have two, the whitelisting function would create a new AppID for the instance `FireFox.exe` with three network connections. In the interim, the whitelisting function was modified to check whether or not connections exist. Therefore, if a process named `Firefox.exe` has one hundred open network connections and all other instances of a process named `Firefox.exe` have two, the instance `FireFox.exe` with one hundred network connections is matched in the whitelist and hidden from view.

The visualization methods used in this research were only tested on Microsoft Windows operating systems. However, the abstraction used in development should allow easy porting to Linux, OS X, and Android. The limiting factor in analyzing other operating systems is CMAT. Rekall[25] and Volatility[23] both support Linux, OS X and Android, CMAT currently does not.

The next version of the memory visualization tool should implement a code ex-

traction feature. This would allow forensic examiners to extract unpacked executable code segments from a memory image using the memory visualization tool's user interface. This would most likely require an additional database schema.

VII. Appendix A – IRB Exemption Letter

DEPARTMENT OF THE AIR FORCE
AIR FORCE INSTITUTE OF TECHNOLOGY
WRIGHT-PATTERSON AIR FORCE BASE OHIO

23 June 2015

MEMORANDUM FOR DR Gilbert peterson

FROM: William A. Cunningham, Ph.D.
AFIT IRB Research Reviewer
2950 Hobson Way
Wright-Patterson AFB, OH 45433-7765

SUBJECT: Approval for exemption request from human experimentation requirements (32 CFR 219, DoDD 3216.2 and AFI 40-402) for Memory Visualization Educational Impacts.

1. Your request was based on the Code of Federal Regulations, title 32, part 219, section 101, paragraph (b) (2) Research activities that involve the use of educational tests (cognitive, diagnostic, aptitude, achievement), survey procedures, interview procedures, or observation of public behavior unless: (i) Information obtained is recorded in such a manner that human subjects can be identified, directly or through identifiers linked to the subjects; and (ii) Any disclosure of the human subjects' responses outside the research could reasonably place the subjects at risk of criminal or civil liability or be damaging to the subjects' financial standing, employability, or reputation.

2. Your study qualifies for this exemption because you are not collecting sensitive data, which could reasonably damage the subjects' financial standing, employability, or reputation. Further, the demographic data you are utilizing and the way that you plan to report it cannot realistically be expected to map a given response to a specific subject.

3. This determination pertains only to the Federal, Department of Defense, and Air Force regulations that govern the use of human subjects in research. Further, if a subject's future response reasonably places them at risk of criminal or civil liability or is damaging to their financial standing, employability, or reputation, you are required to file an adverse event report with this office immediately.

WILLIAM A CUNNINGHAM, PH.D.
AFIT Exempt Determination Official

VIII. Appendix B – Example Exercises

Study One Exercise Example.

Assignment #4
Cyber Forensics - CSCE 527
Due: 8:00 AM, Wednesday August 20, 2014
Incident Response

Purpose:

Gain an understanding of the collectable information from an active machine in a logged in state, and the performance of an incident response.

Scenario:

ABC Corporation is a small board game distributor started by Col. David Mustard (ret) in 2008. They purchase games from the publishers and sell them to game stores. David focuses on sales (getting and keeping customers). In addition to David, there are two full-time employees, John Green who is a combination IT-guy and warehouse manager. And Nancy Scarlett, who handles the actual day-to-day receiving, shipping, and inventory control.

Today, John received a panicked text message/email from David when David discovered that their "Top Customer List" saved on David's computer appears to have been changed. You have been contacted by John to help with the investigation. Has the file been modified and if so, by who and for what purpose?

Assignment:

Before searching the machine, generate the data collection process (process list, current connections, etc.) and a general policy that will you use for approaching a system and analyze the data.

Several suites are available in the course folder. Feel free to use and extend these or make your own scripts.

During the analysis focus on Inman & Rudin (identification, individualization/classification, association, and reconstruction) and the narrative (who, what, when, how, where, why). In the final report, clearly state the forensic question you are answering with a particular piece of evidence.

As an example, if you found and identified a file as potential evidence (identification, what and where) you need to determine how the file arrive on the machine (association, how). Remember that how it did not arrive is as useful as how it did. In doing this, do not just randomly look for things; think about all the ways it could arrive, and write that out (include in the report, and then follow your process).

Prepare a written report detailing the steps and tools you used for this lab, the evidence found, and any other recommended documentation. As always, maintain documentation of your investigation, recording everything that you do and when you do it. Also, in this report present at least two methods you would

use to triage the machine. In the triage discussion, be sure to discuss tradeoffs between the certainty of being clean and the time the triage takes.

Scope:

Your investigation will use a set of Virtual Box machines.

ABC Corporation has a very small digital footprint. David and John each have a computer (Box1 and Box2 respectively) and they are networked together. They have email accounts on Yahoo (GreenABCCorp@yahoo. com and MustardABCCorp@yahoo.com. Though neither know it, they share the same password (Password!123) which is the same password for logging into their machines. Just to help with administration, John has also set up an account for himself on David's machine in case David needs something while he is gone.

Network Summary:
192.168.56.103 Box1 John Green (pwd: Password!123)
GreenABCCorp@yahoo.com
192.168.56.102 Box2 David Mustard (pwd: Password!123)
MustardABCCorp@yahoo.com
192.168.56.104 Box3

Power the boxes on in reverse order: Box3, Box2, and Box1. The presence of Box3 can be ignored for the scenario. All boxes are reachable from the host machine (ipconfig on host to get IP).

Take care to reduce alteration to the state of the system. This way, if you need to come back and re-perform your investigation, you can do so.

Email a soft copy of written report to Gilbert.Peterson@afit.edu.

Study Two Exercise Example.

User Activity Image:

1) List processes the user had running at the time of the memory capture.

2) Did any of the processes listed above have interesting file handles? If so, please list.

3) Which processes had network connections? What where the remote IP addresses?

4) How many processes have Shell32.dll loaded? List each process and indicate whether it is a system (kernel) process or user process?

Malware Images:
For the three memory images provided, please identify which type of malware is present and the suspected process Name and PID.

1) Box1: Process Name _____ PID___ [ ] RootKit [ ] Remote Access Tool Kit [ ] Both [ ] Neither
2) Box2: Process Name _____ PID___ [ ] RootKit [ ] Remote Access Tool Kit [ ] Both [ ] Neither
3) Box3: Process Name _____ PID___ [ ] RootKit [ ] Remote Access Tool Kit [ ] Both [ ] Neither

5 Box Exercise:

Analyze the five images provided and report anything out of the ordinary. This is an open ended question, report what you find...there are many correct answers.

Bibliography

1. O. Carroll, S. Brannon, and T. Song. (2008, January) Computer forensics: Digital forensic analysis methodology. [Online]. Available: http://www.justice.gov/sites/default/files/usao/legacy/2008/02/04/usab5601.pdf

2. M. Chen and H. Jänicke, "An information-theoretic framework for visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1206–1215, 2010.

3. J. Heer, M. Bostock, and V. Ogievetsky, "VIISUALIZATION A Tour through the Visualization Zoo A survey of powerful visualization techniques , from the obvious to the obscure," *Communications of the ACM*, vol. 53, no. 5, pp. 59–67, 2010. [Online]. Available: http://cat.inist.fr/?aModele=afficheN&cpsidt=22906879

4. F. Hinshaw, "Data warehouse appliances Driving the Business Intelligence Revolution," *DM Review*, vol. 14, no. 9, p. 30, sep 2004.

5. G. Henderson, "Triage visualization for digital media exploitation," Master's thesis, Naval Postgraduate School, 2013. [Online]. Available: http://calhoun.nps.edu/bitstream/handle/10945/37636/13Sep_Henderson_Glenn.pdf?sequence=1

6. N. Beebe and J. Clark, "Dealing with Terabyte Data Sets in Digital Investigations," in *Advances in Digital Forensics*, S. Pollitt, Mark and Shenoi, Ed. Springer US, 2005, pp. 3–16. [Online]. Available: http://dx.doi.org/10.1007/0-387-31163-7_1

7. S. Teerlink and R. F. Erbacher, "Improving the computer forensic analysis process through visualization," *Communications of the ACM*, vol. 49, no. 2, p. 71, 2006. [Online]. Available: http://dx.doi.org/10.1145/1113034.1113073

8. G. Osborne, H. Thinyane, and J. Slay, "Visualizing information in digital forensics," in *Advances in Digital Forensics VIII*, ser. IFIP Advances in Information and Communication Technology, G. Peterson and S. Shenoi, Eds. Springer Berlin Heidelberg, 2012, vol. 383, pp. 35–47. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-33962-2_3

9. J. B. Baum, "Windows Memory Forensic Data Visualization," Master's thesis, Air Force Institute of Technology, 2014. [Online]. Available: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA602889

10. United States Computer Emergency Readiness Team. (2008) Computer forensics. [Online]. Available: https://www.us-cert.gov/sites/default/files/publications/forensics.pdf

11. W. Kruse and J. Heiser, *Computer Forensics: Incident Response Essentials*. Addison-Wesley, 2002.

12. G. Palmer, "A Road Map for Digital Forensic Research," *Proceedings of the 2001 Digital Forensics Research Workshop (DFRWS 2004)*, pp. 1–42, 2001. [Online]. Available: http://www.dfrws.org/2001/dfrws-rm-final.pdf

13. R. Ayers, W. Jansen, and S. Brothers, "Guidelines on Mobile Device Forensics (Draft)," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep., 2013. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-101r1.pdf

14. S. Yadav, K. Ahmad, and J. Shekhar, "Analysis of digital forensic tools and investigation process," in *High Performance Architecture and Grid Computing*, ser. Communications in Computer and Information Science, A. Mantri, S. Nandi, G. Kumar, and S. Kumar, Eds. Springer Berlin Heidelberg, 2011, vol. 169, pp. 435–441. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-22577-2_59

15. National Institute of Justice. (2008) Electronic crime scene investigation: A guide for first responders, second edition. [Online]. Available: https://www.ncjrs.gov/pdffiles1/nij/219941.pdf

16. Scientific Working Groups on Digital Evidence and Imaging Technology. (2015) Swgde/swgit digital and multimedia evidence glossary. [Online]. Available: https://www.swgde.org/documents/Current%20Documents/2015-05-27%20SWGDE-SWGIT%20Glossary%20v2.8

17. B. D. Carrier and J. Grand, "A hardware-based memory acquisition procedure for digital investigations," *Digital Investigation*, vol. 1, no. 1, pp. 50–60, 2004. [Online]. Available: http://dx.doi.org/10.1016/j.diin.2003.12.001

18. N. Davis, "Live Memory Acquisition for Windows Operating Systems," *Citeseer*, 2008. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.173.6197&rep=rep1&type=pdf

19. H. Carvey, *Windows Forensic Analysis*. Elsevier, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9781597491563500083

20. M. Burdach, "Physical memory forensics." Black Hat, 2006. [Online]. Available: https://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Burdach.pdf

21. M. Becher, M. Dornseif, and C. Klein, "FireWire: all your memory are belong to us," in *Proceedings of CanSecWest*, 2005, p. 40. [Online]. Available: https://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:FireWire:+all+your+memory+are+belong+to+us#0

22. L. Zeltser. (2015) One-click windows memory acquisition with dumpit. Blog Post. [Online]. Available: https://zeltser.com/memory-acquisition-with-dumpit-for-dfir-2/

23. M. H. Ligh, A. Case, J. Levy, and A. Walters, *The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory*, 1st ed., C. Long, Ed. Indianapolis: Wiley, 2014. [Online]. Available: http://www.amazon.com/Art-Memory-Forensics-Detecting-Malware/dp/1118825098/ref=sr_1_1?s=books&ie=UTF8&qid=1412631153&sr=1-1&keywords=the+art+of+memory+forensicsevernote:///view/5232484/s49/1c7c4311-35ce-473a-9049-9e2b3b2308a0/1c7c4311-35ce-473a-9049-9e2b

24. J. Okolica and G. L. Peterson, "Windows operating systems agnostic memory analysis," *Digital Investigation*, vol. 7, no. SUPPL., pp. S48–S56, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.diin.2010.05.007

25. M. Cohen, "Rekall memory forensic framework," 2013. [Online]. Available: http://www.rekall-forensic.com/docs/Tools/pmem.html

26. J. Stüttgen and M. Cohen, "Anti-forensic resilient memory acquisition," *Digital Investigation*, vol. 10, pp. 105–115, 2013.

27. M. I. Cohen, D. Bilby, and G. Caronni, "Distributed forensics and incident response in the enterprise," *Digital Investigation*, vol. 8, pp. S101–S110, 2011.

28. T. J. Jankun-Kelly, J. Franck, D. Wilson, J. Carver, D. Dampier, and J. E. Swan, Ii, "Show me how you see: Lessons from studying computer forensics experts for visualization," in *Proceedings of the 5th International Workshop on Visualization for Computer Security*, ser. VizSec '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 80–86.

29. Y. Liu, Y. Wang, and Y. Jin, "Research on the improvement of MongoDB Auto-Sharding in cloud environment," *ICCSE 2012 - Proceedings of 2012 7th International Conference on Computer Science and Education*, pp. 851–854, 2012.

30. M. Tory and T. Moller, "Rethinking Visualization: A High-Level Taxonomy," *IEEE Symposium on Information Visualization*, pp. 151–158, 2004. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1382903

31. B. Shneiderman, "Tree Visualization with Tree-Maps: 2-d Space-Filling Approach," *ACM Transactions on Graphics*, vol. 11, no. 1, pp. 92–99, 1992. [Online]. Available: http://dx.doi.org/10.1007/s13398-014-0173-7.2

32. C. Stab, K. Nazemi, and D. W. Fellner, "SemaTime - Timeline visualization of time-dependent relations and semantics," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6455 LNCS, pp. 514–523, 2010.

33. S. L. Garfinkel, "Digital forensics research: The next 10 years," *Digital Investigation*, vol. 7, pp. S64–S73, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1742287610000368

34. S. Chawathe, "Effective whitelisting for filesystem forensics," in *Intelligence and Security Informatics, 2009. ISI '09. IEEE International Conference on*, June 2009, pp. 131–136.

35. A. Graves, "Creation of visualizations based on linked data," *Proceedings of the 3rd International Conference on Web Intelligence, Mining and Semantics - WIMS '13*, p. 1, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?id=2479787.2479828

36. M. Bostock, V. Ogievetsky, and J. Heer, "D3 data-driven documents," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2301–2309, 2011. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2011.185

37. D. Baranovskiy. (2015) Raphal - javascript library. [Online]. Available: http://raphaeljs.com/

38. M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An Open Source Software for Exploring and Manipulating Networks," *Third International AAAI Conference on Weblogs and Social Media*, pp. 361–362, 2009. [Online]. Available: http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154$\backslash$npapers2://publication/uuid/CCEBC82E-0D18-4FFC-91EC-6E4A7F1A1972

39. Gephi.org. (2015) Gephi - the open graph viz platform. [Online]. Available: http://gephi.github.io/

40. L. Bonnet, A. Laurent, M. Sala, B. Laurent, and N. Sicard, "Reduce, You Say: What NoSQL Can Do for Data Aggregation and BI in Large Repositories," *2011 22nd International Workshop on Database and Expert Systems Applications*, pp. 483–488, 2011. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6059864

41. D. Abadi, "Consistency tradeoffs in modern distributed database system design: Cap is only part of the story," *Computer*, vol. 45, no. 2, pp. 37–42, Feb 2012. [Online]. Available: http://dx.doi.org/10.1109/MC.2012.33

42. V. Abramova and J. Bernardino, "NoSQL databases," *Proceedings of the International C* Conference on Computer Science and Software Engineering - C3S2E '13*, pp. 14–22, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?id=2494444.2494447

43. R. Fielding and R. Taylor, "Principled design of the modern Web architecture," *Proceedings of the 2000 International Conference on Software Engineering. ICSE 2000 the New Millennium*, vol. 2, no. 2, pp. 115–150, 2000.

44. D. Crockford, *JavaScript: The Good Parts*. O'Reilly Media, 2008. [Online]. Available: http://www.amazon.com/JavaScript-Good-Parts-Douglas-Crockford/dp/0596517742%3FSubscriptionId%3D0JYN1NVW651KCA56C102%26tag%3Dtechkie-20%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0596517742

45. D. Flanagan, *JavaScript: The Definitive Guide*. O'Reilly Media, 2001. [Online]. Available: http://www.amazon.com/JavaScript-Definitive-Guide-David-Flanagan/dp/0596000480%3FSubscriptionId%3D0JYN1NVW651KCA56C102%26tag%3Dtechkie-20%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0596000480

46. W. Hales, *HTML5 and JavaScript Web Apps*. O'Reilly Media, 2012. [Online]. Available: http://www.amazon.com/HTML5-JavaScript-Apps-Wesley-Hales/dp/1449320511%3FSubscriptionId%3D0JYN1NVW651KCA56C102%26tag%3Dtechkie-20%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D1449320511

47. Angularjs.org. (2015) Angularjs - superheroic javascript mvw framework. [Online]. Available: https://angularjs.org/

48. J. Dickey, *Write Modern Web Apps with the MEAN Stack (Mongo Express AngularJS and Node.js)*. PeachpitPress, 9 2014. [Online]. Available: http://amazon.com/o/ASIN/B00QPZ9LS6/

49. Expressjs.com. (2015) Express - node.js web application framework. [Online]. Available: http://expressjs.com/

50. A. Mardanov, *Express.js Guide: The Comprehensive Book on Express.js: The Comprehensive Book on Express.js*. CreateSpace Independent Publishing Platform, 2013. [Online]. Available: http://www.amazon.com/Express-js-Guide-The-Comprehensive-Book/dp/1494269279%3FSubscriptionId%3D0JYN1NVW651KCA56C102%26tag%3Dtechkie-20%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D1494269279

51. S. Tilkov and S. Vinoski, "Node.js: Using JavaScript to Build High-Performance Network Programs," *Internet Computing, IEEE*, vol. 14, no. 6, pp. 80–83, nov 2010.

52. W. Gortney, *CHAIRMAN OF THE JOINT CHIEFS OF STAFF MANUAL 6510.01B CYBER INCIDENT HANDLING PROGRAM*, Department Of Defense, 1400 Defense Pentagon Washington, DC 20301-1400, dec 2014. [Online]. Available: http://www.dtic.mil/cjcs_directives/cdata/unlimit/m651001.pdf

53. A. Goda and K. Parat, "Scaling directions for 2d and 3d nand cells," in *Proceedings of the 2012 IEEE International Electron Devices Meeting (IEDM)*, Dec 2012, pp. 2.1.1–2.1.4. [Online]. Available: http://dx.doi.org/10.1109/IEDM.2012.6478961

54. Arxsys. (2014) Features - arxsys. [Online]. Available: http://www.arxsys.fr/features/

55. C. Vandeplas, "Finding the needle in the haystack with ELK," in *Digital Forensics and Incident Response (DFIR)*. Prague: SANS Institute, 2014.

56. Microsoft Developer Network. (2015) File handles (Windows). [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/aa364225(v=vs.85).aspx

57. D. S. Mark Russinovich, *Microsoft Windows Internals*, 4th ed., L. A. Robin Van Steenburgh, Ben Ryan, Valerie Woolley, Sally Stickney, Roger LeBlanc, Ed. Redmond, Washington: Microsoft Press, 2005. [Online]. Available: http://csit.udc.edu/~byu/UDC3529315/WindowsInternals-4e.pdf

58. P. C. Silberman, "FUTo," *Uninformed*, vol. 3, p. 14, 2006. [Online]. Available: http://www.uninformed.org/?v=3&a=7

59. FireEye Inc., "Poison Ivy: Assessing Damage and Extracting Intelligence," Milpitas, CA, Tech. Rep., 2014. [Online]. Available: https://www.fireeye.com/content/dam/fireeye-www/global/en/current-threats/pdfs/rpt-poison-ivy.pdf

60. T. M. Project. (2004) Metasploit's meterpreter. [Online]. Available: https://dev.metasploit.com/documents/meterpreter.pdf

61. J. Feinberg. (2014) Wordle - beautiful word clouds. [Online]. Available: http://www.wordle.net/

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
24–03–2016	Master's Thesis	May 2014 — Mar 2016

4. TITLE AND SUBTITLE

Whitelisting System State In Windows Forensic Memory Visualizations

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Lapso, Joshua A., Captain, USAF

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-16-M-029

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

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

Examiners in the field of digital forensics regularly encounter enormous amounts of data and must identify the few artifacts of evidentiary value. The most pressing challenge these examiners face is manual reconstruction of complex datasets with both hierarchical and associative relationships. The complexity of this data requires significant knowledge, training, and experience to correctly and efficiently examine. Current methods provide primarily text-based representations or low-level visualizations, but levee the task of maintaining global context of system state on the examiner. This research presents a visualization tool that improves analysis methods through simultaneous representation of the hierarchical and associative relationships and local detailed data within a single page application. A novel whitelisting feature further improves analysis by eliminating items of little interest from view, allowing examiners to identify artifacts more quickly and accurately. Results from two pilot studies demonstrates that the visualization tool can assist examiners to more accurately and quickly identify artifacts of interest.

15. SUBJECT TERMS

Memory forensics, Incident response, Information visualization, Single page web application, D3.js

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 L. Peterson, AFIT/ENG
U	U	U	U	120	19b. TELEPHONE NUMBER *(include area code)* (937)255-6565 x4281; gilbert.peterson@afit.edu