2019

Developing an automated malware detection analysis and reporting tool for MS-Windows

David Matingi Mutyethau
Faculty of Information Technology (FIT)
Strathmore University

Developing an Automated Malware Detection, Analysis and Reporting tool for MS Windows

Mutyethau David Matingi

Master of Science in Information Systems Security

2019

Developing an Automated Malware Detection, Analysis and Reporting tool for MS Windows

Mutyethau David Matingi

Submitted in partial fulfillment of the requirements for the Degree of Master of Science in Information Systems Security

Faculty of Information Technology

Strathmore University

Nairobi, Kenya.

JUNE, 2019

Declaration

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the dissertation contains no material previously published or written by another person except where due reference is made in the thesis itself.

Mutyethau David Matingi

12th June 2019

Approval

The dissertation of Mutyethau David Matingi was reviewed and approved by the following:

Dr. Ondrej Rysavy,
Assistant Professor, Faculty of Information and Technology,
Strathmore University

Dr. Joseph Orero,
Dean, Faculty of Information and Technology,
Strathmore University

Professor Ruth Kiraka,
Dean, School of Graduate Studies,
Strathmore University

ABSTRACT

Memory and computer forensics is a field that has witnessed a lot of advancements in the recent past. Memory forensics enables investigators acquire and investigate the content of a computer's RAM while computer forensics enable the investigator to acquire information from the hard drive. While valuable artifacts can be extracted from computers, the use of this technique presents several challenges, such as, data acquisition, searching for artifacts and data analysis of extracted information.

The variants of malware families share typical behavioral patterns reflecting their origin and purpose. The behavioral patterns obtained either statically or dynamically can be exploited to detect and classify unknown malwares into their known families using machine learning techniques. This dissertation aims to create a malware detection, analysis and reporting tool that shall be open source, user friendly, intuitive and automated for MS Windows. The tool shall assist forensic investigators in discovering crucial information in the suspect computer such as malware present. The tool shall analyse content stored in the computer's hard drive and captured memory images. This shall include analysis of single files, folders, hard disk partitions and the entire hard disk. For live memory, the tool shall aim to determine processes and files that were open or present at time of live analysis.

Keywords: Digital Forensics, Information Security, Data visualisation, Information analysis, Random Access Memory, Graphical User Interface, Command Line Interface, Malware analysis.

Contents

List of Figures

List of Tables

Abbreviations/Acronyms

CPU- Central Processing Unit

MEMDUMP-Memory Dump

OS- Operating System

PID- Process ID

RAM- Random Access Memory

UI- User Interface

Acknowledgment

First I thank the Almighty God for the strength and ability to undertake my studies. Secondly, I express my sincere and deep gratitude to my guide Dr. Ondrej Rysavy, an assistant professor in Department of Information Systems at Brno University of Technology in Czech Republic, for the invaluable guidance and support. He provided me with resources and guidance for this dissertation. Thirdly I wish to thank my family, Mr. & Mrs. Richard Matingi, lovely wife Judy, brothers Joe and Dominic and sister Mercy for their prayers and encouragement through my study period. Fourthly my sincere gratitude goes to Strathmore University and @iLabAfrica fraternity for according me the opportunity to undertake my Masters studies in the institution and for the resources availed to us in this period. Lastly, my appreciation goes to all my family and friends who have been very supportive of my journey. God bless you all.

Chapter 1: Introduction

1.1 Background of the Study

Malicious software or malware is defined as any software that deliberately fulfills a harmful intent of the attacker (Bayer, Moser, & Kirda, 2006). This is often done with the aim to gain access to network resources, computer systems, disrupt computer operations and even gather information from the computer without the owner's consent. Malwares present themselves in wide range of variations like Worm, Virus, Rootkit, Trojan-horse, Backdoor, Botnet, Spyware, Adware (Egele, Manuel, Theodoor, Engin, & Kruegel, 2012). These classes of malwares are not mutually exclusive meaning therefore that an individual malware might reveal the features of multiple classes at the same time.

According to a survey carried out by Fire Eye in June 2013, 47% of the organizations experienced malware network breaches/ security incidents in the year 2012 (FireEye, 2013) .The malwares are continuously growing in volume, variety in terms of innovative malicious methods and velocity (Data, 2012). They are evolving, becoming more sophisticated and using innovative methods to target computers and mobile devices. McAfee indexes over 100,000 new malware samples every day (McAfee, 2013). This translates to about 69 new threats per minute or approximately one malware threat per second.

1.2 Problem Statement

While there is some level of automation in forensic tools, some tools require the investigator to have knowledge of the Command Line Interface in order to do proper investigation. Secondly, most of the automated forensic tools are commercial and expensive making it difficult for financially deprived parties to efficiently make use of them to gather information (Vinod, Jaipur , Laxmi, & Gaur, 2009). Thirdly, most forensic malware detection tools either analyse live memory or captured memory images at a time (Gandotra, Bansal, & Sofat, Zero-day malware detection., 2016).

1.3 Research Objectives

1.3.1 General Objective

The objective of this dissertation is to develop an automated malware detection, analysis and reporting tool for MS Windows for forensic purposes.

1.3.2 Specific Objectives

1.To analyse the current capabilities of forensic memory malware analysis tools.

2.To identify the output of forensic memory malware analysis tools and specify the data that is to be analysed in our research.

3.To develop and test a proof-of-concept implementation of a malware detection, analysis and reporting tool.

4.To validate the developed tool using a set of controlled experiments.

1.4 Research Questions

1. What are the capabilities of the current forensic memory malware analysis tools?

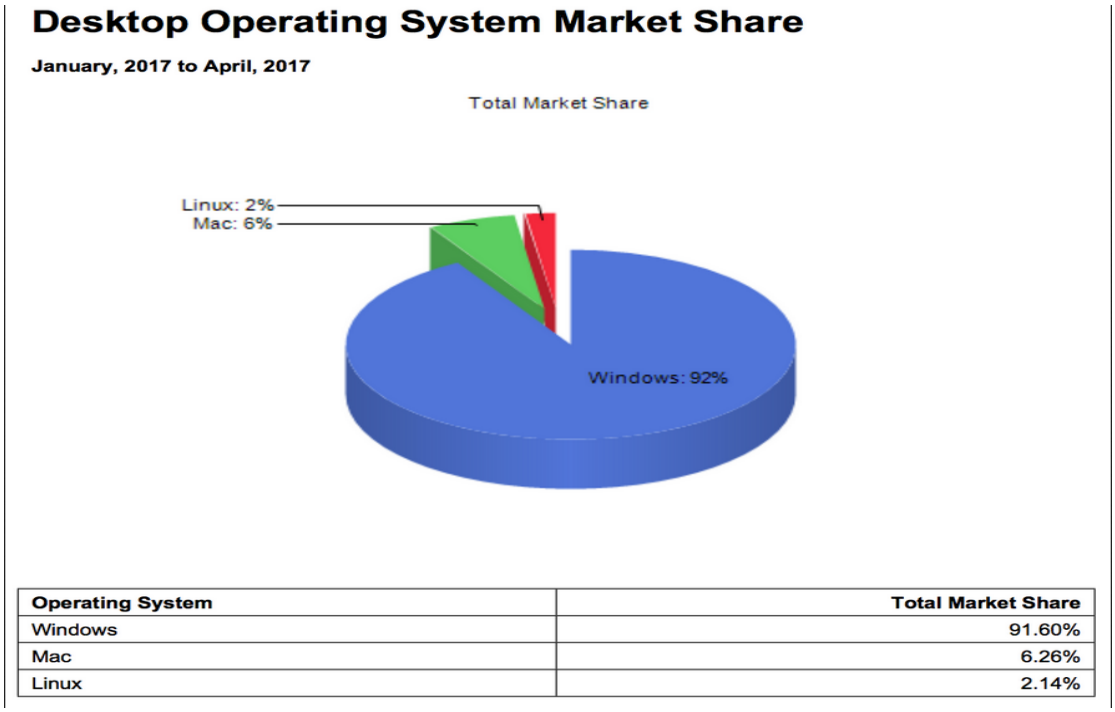
2.What output is given by forensic memory malware analysis tools and what data will be analysed in our research?

3.How can we develop and test a malware detection, analysis and reporting tool?

4. Does our tool meet the set functionalities when validated using a set of controlled experiments?

1.5 Assumptions

This study makes a major assumption that most forensic experts are likely to come across crimes perpetrated through the Microsoft Windows family of operating system. This is largely attributed to its popularity by most organisations and its positioning in the market as illustrated in figure 1.1. It also makes an assumption that most advanced forensic analysis tools are commercial and expensive for the common forensic expert.



Operating System	Total Market Share
Windows	91.60%
Mac	6.26%
Linux	2.14%

Figure 0.1: Worldwide OS Market Share Report, 2017 (Netmarketshare, 2017)

This research also takes into assumption that most forensic tools consume dump memory images in .E01 file format.

1.6 Research Motivation

The motivation behind this research emanates from the increased need to come up with innovative techniques to assist in digital forensic analysis. One proposed way is the use of Graphical User Interface techniques to offer a unique view on collection datasets (Osborne & Turnbull, 2012). According to Osborne and Turnbull (2012), visualisation tools have the ability to "assist in the discovery of new evidence, make decisions based on data, and also provide explanations pertaining to certain phenomena existent in a data set". Also, with the increased storage capacities and increased complexity of data, forensic analysts are finding it hard to properly analyse and make use of data available for investigations (Neufeld, 2010). There is therefore, need to address these issues through the development of a tool which shall be able to deal with the increased volumes of digital evidence. The tool ought to be intuitive and effective for both skilled investigators and forensic analysts, for both live and captured images from suspect computers. With proper analysis and investigations, evidence found in computers can reveal crucial information about the state of both the memory and physical drives at the time of acquisition, and whether it was infected with a malware (Ligh, Case, Levy, & Walters, 2014). Memory images will be analysed in .E01 file formats. The .E01 file format can be generated by Encase or Winen tool (Software, 2017) although Encase is the most widely known and used forensic tool, that has been produced and launched by the Guidance Software Inc. for creating .E01 files (Garfinkel, 2014)

1.7 Scope and Limitations

The problems identified under our research problem need to be addressed through the development of a tool which shall be able to deal with the increased volumes of digital evidence. The tool ought to be intuitive and effective for both skilled investigators and forensic analysts. With proper analysis and investigations, evidence found in computers can reveal crucial information about the state of both the memory and physical drives at the time of acquisition, and whether it was infected with a malware (Ligh, Case, Levy, & Walters, 2014). This research will analyse computer drives, captured computer memory image as well as live memory. Live memory will be analysed to determine running processes and open files at the time of analysis. The computer drive and captured memory image will be analysed for malware present at time of analysis and acquisition. Computer drives will analyse single files, folders, hard disk partitions as well as the whole computer hard drive. Memory images will be analysed in .E01 file formats. The .E01 file format can be generated by Encase or Winen tool (Software, 2017) although Encase is the most widely known and used forensic tool, that has been produced and launched by the Guidance Software Inc. for creating .E01 files (Garfinkel, 2014)

The major limitation is the study will focus on one family of operating system, the Windows operating system. The tool will also focus on one file type for memory image, the .E01 file type.

1.8 Chapter Summary

This chapter has highlighted the problem, the objectives, as well as the scope and motivation behind this study. Chapter 2 highlights literature on existing digital forensic tools and tries to identify gaps and areas of advancement.

Chapter 2: Literature Review

2.1 Introduction

While digital forensics has been around for a while, cyber-criminal activities have continued to rise day by day. The main aim of forensics is to extract, analyse and present the report analysed in an admissible manner for use in a court of law (National Institute of Justice, 2012). The digital forensic investigation is supported by either general purpose or specifically focused software tools (Data, 2012). While commercial tools provide a rich user interface that is suitable for less experienced analysts, they are quite expensive (Vinod, Jaipur , Laxmi, & Gaur, 2009). Open source tools often require more experienced users. Often, to solve a case the analyst needs to combine several different tools, including open source software.

This chapter discusses existing malware detection and analysis tools, forensic visualisation tools, MS Window address formats and tries to identify a gap in the way the tools detect, analyse and present digital evidence.

2.2 Digital Forensics

Digital forensics is a branch of forensic science encompassing the recovery and investigation of material found in digital devices, often in relation to computer crime (Kruse & Heiser, 2002). Digital forensics uses scientific approach to gather and analyse digital data independent of media (Pendergas, 2010). The main goal is usually to determine likely core effects given the hints discovered. This approach utilises a hypothesis-driven method. This is where evidence is either in support or against the set hypothesis. An example can be a certain company is suspected to have committed a financial fraud. Sources important to the investigation will be secured (laptops,

desktops, phones, servers etc.). The evidence gathered could support this hypothesis or dispute it. Digital forensics is carried out in six main stages, as illustrated in Figure 2.1:

1. The process starts when a suspicion has been raised (NIST, 2016). Relevant data is identified.

2. Acquisition process then follows. Acquisition is the process of preserving the data in question, determining the order of volatility of the data identified, preserve the integrity of the evidence and start the chain of custody (NIST, 2016).

3. After acquisition examination of the data acquired is done. This involves uncompressing, decrypting, carving, or decoding the data in question using forensic tools (NIST, 2016).

4. Analysis of the data examined then follows. Here, the investigators modify the hypothesis, and determine a sequence of events based on the evidence acquired from examination (NIST, 2016).

5. After this there is the documentation stage and then the presentation stage (NIST, 2016).

6. Presentation should be done in a way that is acceptable in a court of law (NIST, 2016).

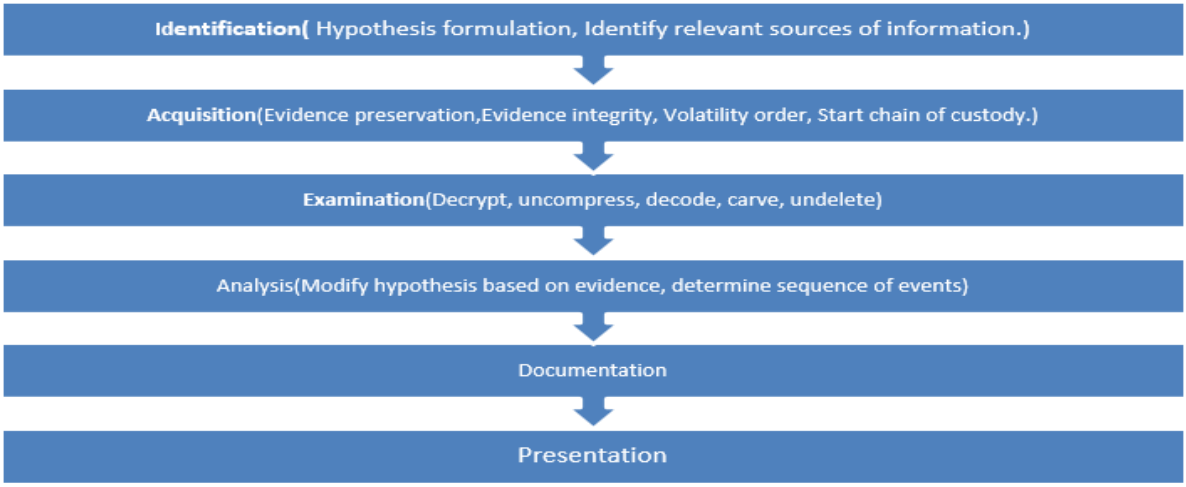


Figure 0.1: Stages of Digital Forensics Process (NIST, 2016).

2.3 Memory Forensics

Memory forensics can be described as a research field that came to light in 2005 and has since gained momentum since its inception through the Digital Forensic Research Workshop (DFRWS)'s "Memory Forensics Challenge" (Digital Forensic Research Workshop , 2008). The challenge sparked research interest in the analysis of contents recovered from the memory dumps. As a result of this, the memory forensics field research has focused on the analysis of memory dumps without relying on hard coded structures. This further introduced the concept of live forensics. This is defined as a method aimed at gathering as much information from a running system as possible (Jones & Etzkorn, 2016). It mainly relies on the underlying system so as to acquire information of running processes. In recent developments, first responders who are usually tasked with the role of performing evidence collection and preservation have started to include live memory forensic tools in their set of tools since these tools have been established as less destructive compared to executing new processes on the system in order to capture data (Brezinski & Killalea, 2002). This literature review gives a synopsis of the commonly used methods of acquiring and analysing a computer's memory. The key focus of this review is on procedures that are used to analyse and visualise

evidence from the Windows family of operating systems, this being based on the notion that forensic investigators are likely to come across Windows-based computers given their popularity and positioning of the operating system in the market. The review will also analyse the gaps in forensic investigation in terms of data visualisation.

2.4 Existing Forensic Memory Analysis Tools

This section takes a look at the existing memory analysis tools and tries to identify gaps in their operation and usability.

The Windows Memory Toolkit now renamed to Comae (http://www.moonsols.com/windows-memory-toolkit/) is a closed, copyrighted tool developed by MoonSols. Windows Memory Toolkit has both pro and free editions, with the main difference being that the pro edition is scriptable and has interactive analysis modes. The tool also supports the analysis of memory images acquired from a wide range of Windows operating systems, including Windows XP through 7. Unfortunately, neither version of the tool supports extensibility through plugins or an API. The analysis techniques are also not disclosed due to the closed source nature of the product.

HB Gary's Responder (https://www.hbgary.com) is a Window physical memory forensics and automated malware analysis tool. The tool offers physical memory analysis and malware detection in one application. It is able to mine information concerning the operating system, open files, running processes, open registry keys, passwords, network activity, web mail and malware. This tool also uses another HB Gary tool, Fast Dump Pro, to freeze the state of the operating system and extract the RAM. However, the certified edition is not free and the community edition is offered as an evaluation only. The source is also closed and there are no techniques of extending the software.

Memoryze from Mandiant (https://www.fireeye.com/services/freeware/memoryze.html) is digital forensic software intended to help investigators discover malware and other malicious activity in

volatile and live memory captures. Memorize is capable of performing acquisition and analysis, with support for full system memory acquisition and extraction of a single process's memory space to disk. The analysis procedures provided simplify details of running processes such as those hidden by rootkits. For every process the tool is able to identify open files, open registry keys, virtual address space, loaded DLLs, network sockets and active connections belonging to the particular process. The tool also supports raw binary memory images captured via any means. The tool is freely available, but is very complex to use and does not work in some operating systems sometimes even after installation (Shreshtha & Chhikara, 2016).

Belkasoft Live RAM Capturer (https://belkasoft.com/ram-capturer) is a free volatile memory forensic tool to capture the live RAM. It is equipped with 32-bit and 64-bit kernel drivers allowing the tool to operate in the most privileged kernel mode. The memory dump will be stored with .mem extension and later it the memory dump can be analyzed using Belkasoft evidence center tool. The main disadvantage of this tool is after capturing memory image, another Belkasoft program need to be downloaded and its reporting is not easily understood.

The Volatility framework (Schatz, 2007) is an open source software framework built to simplify volatile memory forensics. Volatility framework provides analysis support in many ways, including extracting running processes, network connections, the image date and time, network sockets, open files, and currently loaded kernel drivers. However, Volatility framework is operated from the Command Line Interface which makes it difficult for experts and analysts with little knowledge on this to effectively utilise the tool. Another setback of Volatility framework is that it is more suited for UNIX based operating Systems hence analysts intending to use it for Windows analysis face a lot of challenges.

2.5 Gap Analysis

The analysis of memory images is an important part of any digital forensic investigation, in particular by the first responders. Many procedures have been developed that enable crucial state information to be extracted from a memory image. These procedures comprise the ability to list the currently executing processes (including those hidden by malware) and to identify files, registry keys and network connections that a process had opened. One aspect of memory analysis that could prove highly valuable to investigators is the ability to identify and analyse processes and malware from memory dumps. While some of the tools reviewed in this chapter have demonstrated the capability to detect crucial information including malware, they have a drawback in that they are not open sourced, some are complex for the average investigator, while others lack extensibility. The gap identified is the need for the development of a malware detection and analysis tool that is open source, and offers automated analysis and reporting techniques and has extensibility capabilities. This will aim to analyse crucial information in computers and memory images and also present a user friendly, interactive platform for third user parties interested in analysing evidence.

2.6 Chapter Summary

This chapter has discussed several memory forensic tools, malware analysis tools and forensic data visualisation tools. These tools however, are faced with the challenge of forensically analysing and detecting malware. The tools are also commercial in nature and often require subscriptions. This will help investigators extract, find important evidence quickly and easily and present it in a user friendly format.

Chapter 3: Research Methodology

3.1 Introduction

This chapter covers the methodologies used in conducting the study as well as how the data collected was analysed to support the assumptions of the researcher. The design and development methodologies of the proposed system are also highlighted in this chapter.

Rapid Application Development (RAD) will be considered in order to realize the defined research objectives; it is a type of incremental model. Key goal is to realise fast development in a limited time frame, as well as develop a quality system at a considerably low investment cost (CMS, 2005). The RAD model is characterised by four phases: requirements planning, user design, development, and cutover. There is a cycle between the user design and construction phases (Introductions to RAD, 2017).

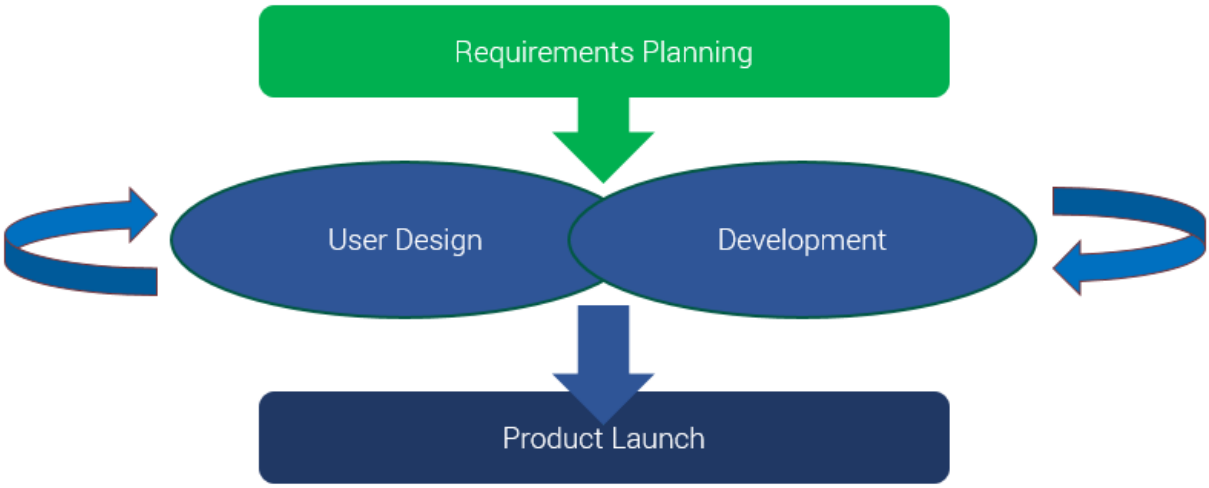


Figure 0.1: RAD Development Model

The main advantage of RAD development is the developed system's ability to adapt to change as requested by end users. Applications are developed in a short span of time. It is also the right

methodology for rapid development as well as prototyping. The sections below highlight the various methodologies that will be used to conduct the research.

3.2 Review of Existing Tools

In this phase, the existing tools shall aim to discuss the current capabilities of memory forensic tools as well as their sample output data. This shall answer our first and second research questions, what are the current capabilities of the use of current memory-based analysis tools as well as the use cases pertaining to malware analysis and what data is to be analysed. This stage shall also include the evaluation of the tools from the perspective of our research. This shall include what input in terms of file types the tools accept, what output they produce, what is the execution platform, the operating system they run on, their capability of analysing both live and captured memory, if they are free versions or Pro version, etc.

3.3 Requirements Planning

This phase shall aim involve identifying business needs, project scope, constraints, and system requirements. A feasibility study will be used to check that the requirements are specific and attainable. After requirement gathering, requirements will be analysed for their validity, and the possibility of incorporating the requirements in the system to be developed will be outweighed. Requirements specifications will be documented for the next phase.

3.4 System Design

System design acts as a road map to indicate how the researcher will answer the research questions presented earlier in chapter one of this dissertation (Wikipedia.org, 2012). The system and software design will be prepared from the requirement specifications which were studied in the first phase. System Design helps in specifying hardware and system requirements

and helps in defining overall system architecture. User requirements identified in requirement analysis will be used in system design specification. Functional specifications will be documented graphically. Deliverables will be context diagrams which define the scope of the system under study. A logical data flow diagram of the proposed system should answer questions such as who will perform the task, how the tasks will be performed and the media type (Sangolly, 1997). Computer-aided software engineering (CASE) tools will be used to develop software that is high-quality, defect-free, and maintainable (Computer Aided Software Engineering, 2016). Case tools to be used in system development shall be Visual paradigm online tool will be used to develop the sequence diagrams, flow charts as well as use case diagrams.

This section aims to answer our third research question, how can a malware detection, analysis and reporting tool be developed and tested as a proof of concept. This section shall also aim to outline the following.

3.4.1 User Interaction Diagrams

These are diagrams meant to highlight how the users shall interact with the system. It shall aim to give a clear understanding of what input is required from the user and what output will be given by the system. These shall include Use case diagrams, sequence diagrams, and data flow diagrams.

3.4.2 System Architecture

System architecture is defined as the conceptual model that defines the behavior, structure and overall views of a system. This section shall aim to give a better understanding on how the front end of the system communicates with the backend processes of the system as well as the system wireframes.

3.5 System Development

The tools used in the development process are Python programming language (https://www.python.org) for both backend and front end and SQLite database (https://sqlite.org/) to store hashed malware signatures locally on the client machine. Python programming language was preferred because it is a high-level, interpreted and general-purpose dynamic programming language that focuses on code readability. The syntax in Python helps the programmers to do coding in fewer steps as compared to other programming languages such as Java or C++. SQLite database was preferred because of its lightweight nature and reliability. SQLite has several advantages the main being it is less prone to bugs rather than custom written file I/O codes. When using SQLite, multiple processes can be attached with same application file and can read and write without interfering each other. It can also be used with all programming languages without any compatibility issue.

3.6 Testing

Testing is vital as it makes sure the system performance conforms to user specifications (Buede, 2009). This aims to answer our third research question, how can we develop and test the developed tool using a set of controlled experiments. Testing shall enable us obtain some preliminary results and improvements shall be proposed.

System testing was carried out using V-Model methodology (Cavalcanti & Borba, 2010). V-Model is an SDLC model where the execution of processes happens in a chronological manner in V-shape. Also known as the Verification and Validation model, the development and QA activities are done simultaneously. There is no isolated phase called testing, rather testing starts right from the requirement phase. System Testing, User Acceptance Testing, Integration Testing and Unit Testing

will all happen simultaneously while verification and validation activities will go hand in hand as illustrated in Figure 3.1.

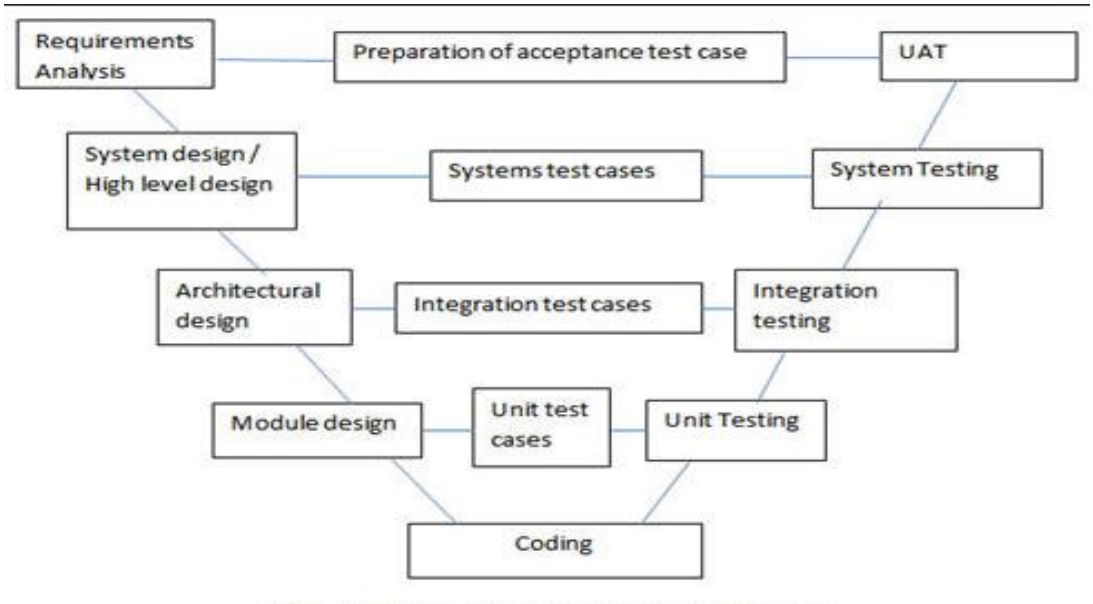


Figure 3.1: V-Model Testing Methodology *(Cavalcanti & Borba, 2010)*.

Integration testing was done to ensure the interaction between different modules of the system were seamless in their functioning. User acceptance testing was done through a peer review of the developed system in order to get views from the prospect users of the tool. System testing involved checking whether the developed tool achieved its desired objectives. This included;

i. Analysing computer hard drives. This involved scanning files, folders, hard disk partitions and the whole hard disk. This was aimed at scanning for malware present in files analysed.

ii. Analysing live memory. This aimed at analysing processes running on suspect computer, files open, processes consuming most CPU time and memory and network activities at time of analysis.

iii. Analysing captured memory image. This analyses processes and files in the captured memory and scans for malware.

Unit testing is a level of software testing where individual units and components of a software are tested. Testing was done on every individual functionality of the tool to ascertain it was working as expected. This tested functionalities on uploading of individual files, scanning of folders, disk partitions and whole disk partitions. It included uploading of memory images and scanning them for malware. It also included analysis of live memory to view processes that are running at time of analysis.

3.7 System Validation

System Validation is a set of actions used to check the compliance of the developed system with its purpose and functions (ISO/IEC/IEEE, 2015). System validation aimed to certify that our system conformed to our research objectives. This was achieved through testing the developed tool using a set of controlled experiments. Live memory from local computer was analysed to determine running processes, open files, processes consuming most CPU time and resources and network activities. Hard drives from local computer was analysed to scan for malware in file, folder, disk partitions as well as the whole disk drive. We used captured memory images downloaded from independent forensic sites https://digitalcorpora.org and https://www.cfreds.nist.gov/data leakage case/data-leakage-case.html. The two memory images contained data captured from suspect computer and our aim was to validate the data therein and if it was infected with malware and the tool was able to detect malware. Validation was also done by comparing results from our tool with other independent malware identification tools.

3.8 Chapter Summary

This chapter has defined the techniques and processes that are to be involved in the development of the system. It has covered sections on analysis, design, system development, testing and validation.

Chapter 4: Review of Existing Forensic Tools

This chapter will answer the first and second research question, what are the capabilities of the existing malware forensic tools and what use cases pertain to malware analysis and what data will be analysed. Malware Forensic tools shall be analysed with an aim of gaining an understanding on how the malware analysis and reporting is carried out.

4.1HB Gary Responder

HB Gary's Responder (https://www.hbgary.com) is a Window physical memory forensics and automated malware analysis tool. It is an application that is known for its ease of use, streamlined workflow, and rapid results. The Professional platform is designed for Incident Responders, Malware Analysts, and Computer Forensic Investigators who require actionable intelligence quickly. Responder Professional provides powerful memory forensics, malware detection, and software behavioral identification with Digital DNA. Figure 4.1 shows a screenshot of a sample result from HB Gary Responder.

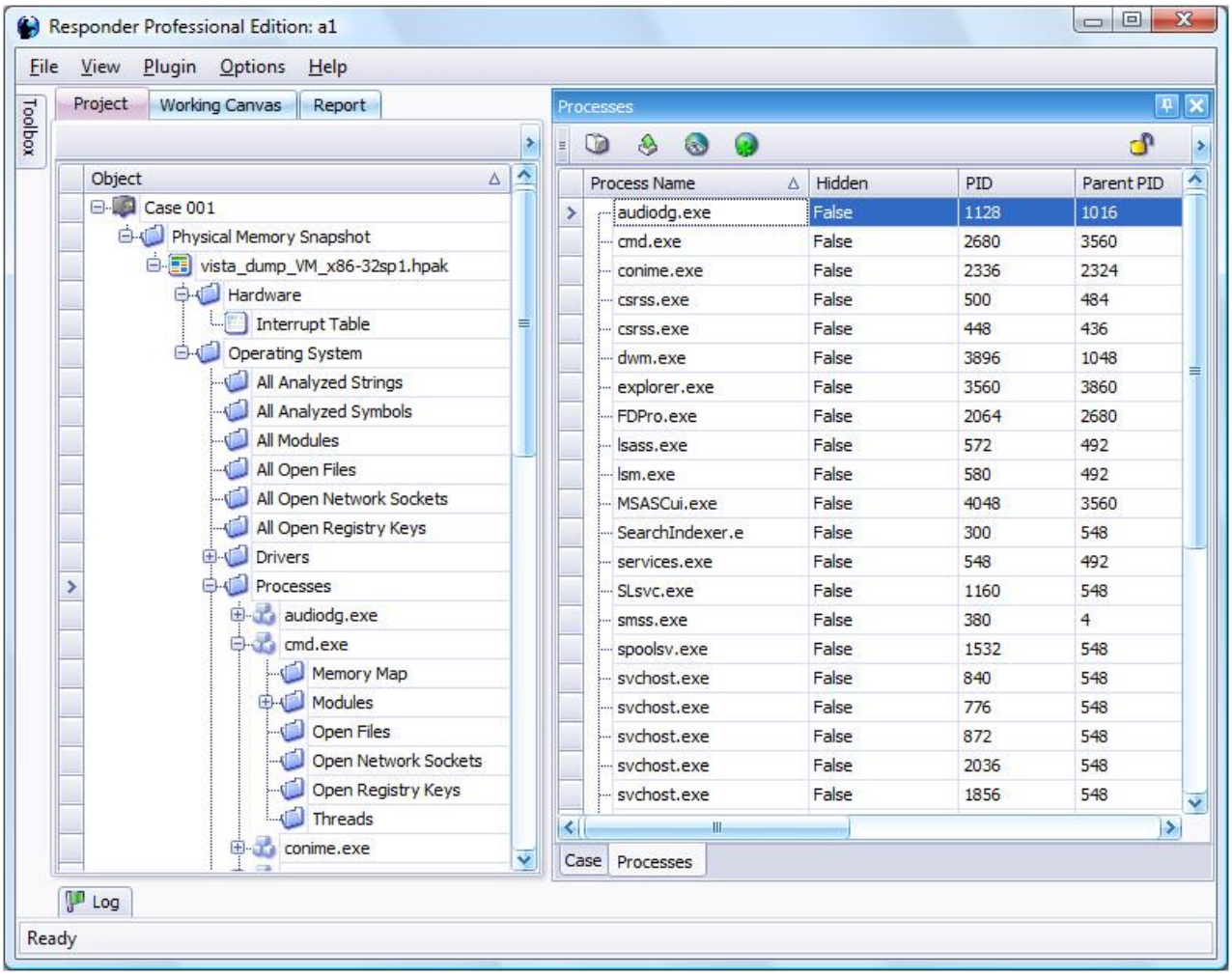


Figure 0.1: Sample Result Screen from HB Gary Tool

HB Gary gives a comprehensive analysis and report on all processes running at the time of acquisition. The tool offers physical memory analysis and malware detection in one application. It is able to mine information concerning the operating system, open files, running processes, open registry keys, passwords, network activity, web mail and malware. This tool also uses another HB Gary tool, Fast Dump Pro, to freeze the state of the operating system and extract the RAM.

4.2 Memoryze

Mandiant's Memoryze (https://www.fireeye.com/services/freeware/memoryze.html)is free memory forensic software that helps incident responders find evil in live memory. Memoryze can acquire and/or analyze memory images and on live systems can include the paging file in its analysis. Mandiant's capabilities include:

i. Image the full range of system memory (no reliance on API calls).

ii. Image a process' entire address space to disk, including a process' loaded DLLs, EXEs, heaps and stacks.

iii. Image a specified driver or all drivers loaded in memory to disk.

iv. Enumerate all running processes (including those hidden by rootkits), including:

v. Identify all drivers loaded in memory, including those hidden by rootkits. For each driver, Memoryze can:

   a) Specify the functions the driver imports and exports.

   b) Hash the driver (MD5, SHA1, and SHA256. disk-based).

   c) Verify the digital signature of the driver (disk-based).

   d) Output all strings in memory on a per driver basis.

vi. Report device and driver layering, which can be used to intercept network packets, keystrokes and file activity.

vii. Identify all loaded kernel modules by walking a linked list. Identify hooks (often used by rootkits) in system call table, the interrupt descriptor tables (IDTs) and driver function tables.

Figure 4.2 shows sample results by Memoryze.

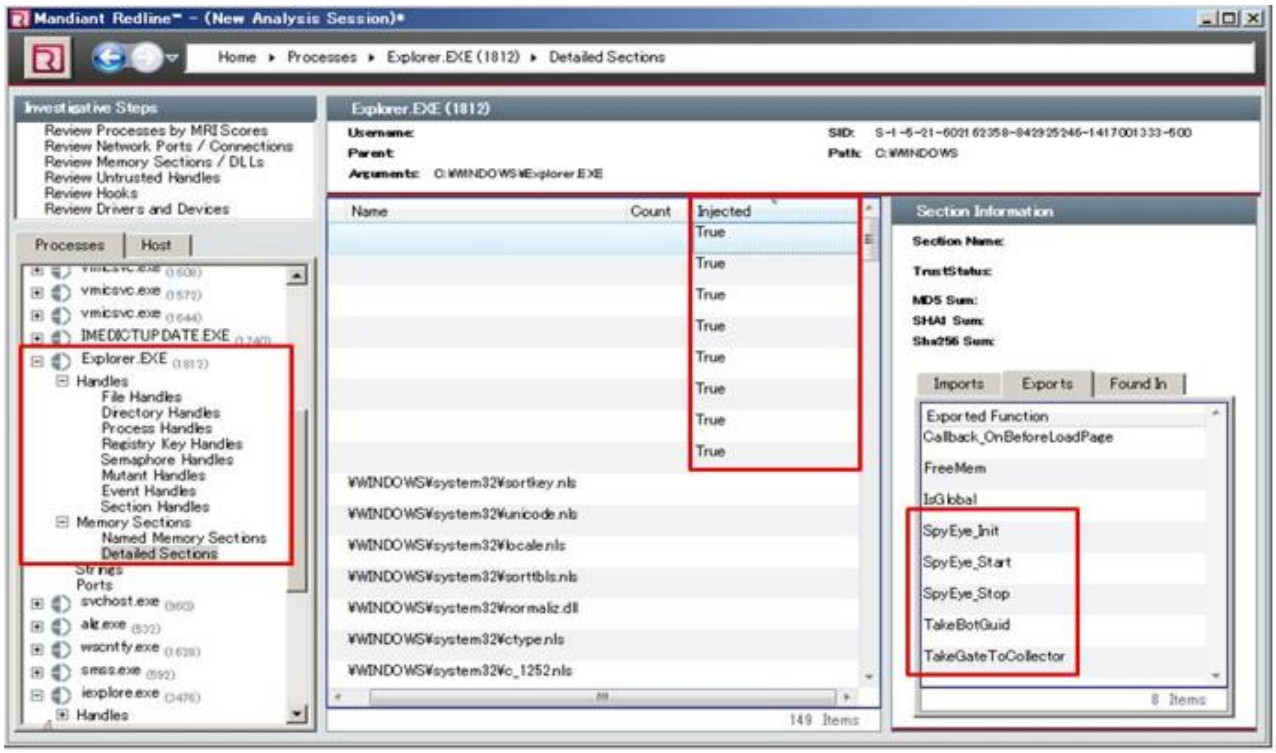


Figure 0.2: Sample Output from Memoryze Tool

4.3 Volatility Tool

Volatility (Schatz, 2007) is an open source memory forensics framework for incident response and malware analysis. It is written in Python and supports Microsoft Windows, Mac OS X, and Linux (as of version 2.5. Volatility supports a variety of sample file formats and the ability to convert between these formats: Raw/Padded Physical Memory, Firewire (IEEE 1394), Expert Witness (EWF),32- and 64-bit Windows Crash Dump,32- and 64-bit Windows Hibernation (from Windows 7 or earlier), 32- and 64-bit Mach-O files, Virtualbox Core Dumps, VMware Saved State (.vmss) and Snapshot (.vmsn), HPAK Format (FastDump), QEMU memory dumps, LiME format. Figure 4.3 shows sample results from Volatility tool during analysis.

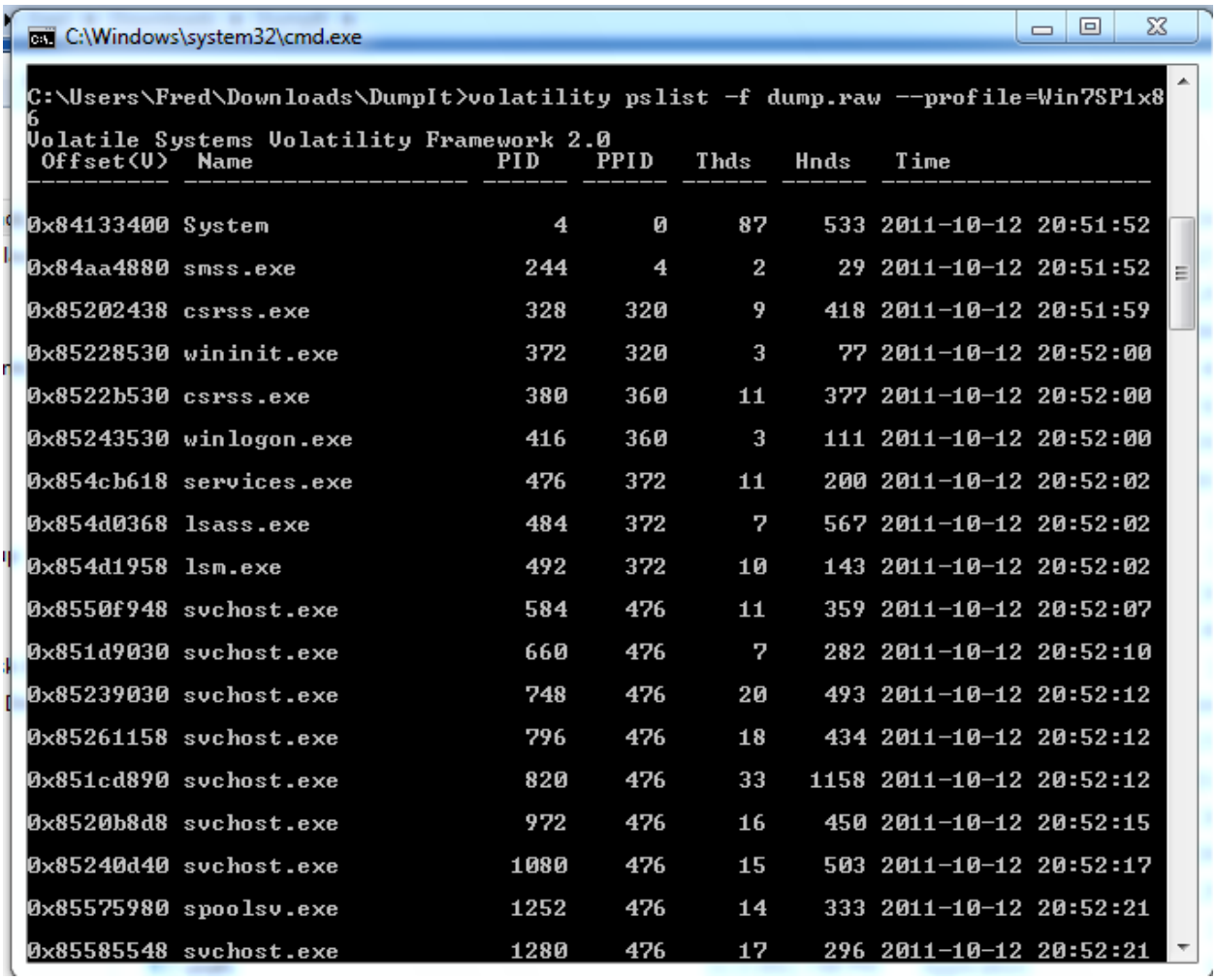


Figure 0.3: Sample Result Screen from Volatility Tool

4.4 Windows Memory Toolkit (Comae)

Comae (http://www.moonsols.com/windows-memory-toolkit/) provides you with memory acquisition and analysis capabilities. It diagnoses your machines and assist you in hunting threats or adversaries that are hiding from your traditional security solutions. Sample result screen is shown in figure 4.4.

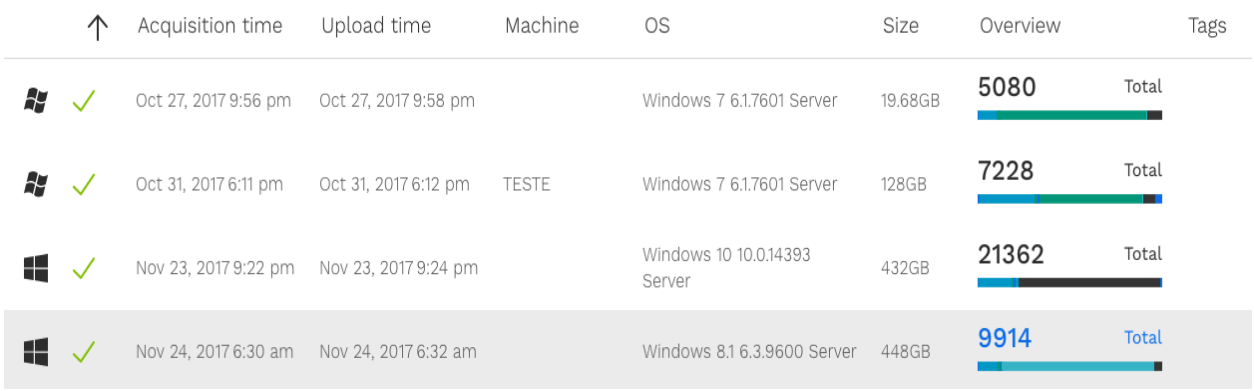


Figure 0.4: Sample Result Screen from Comae

4.5 Chapter Summary

This chapter answers our first and second research questions, to analyse the current capabilities of forensic memory-based malware analysis tools and identify the use cases of malware analysis and specify the data provided by analysis tools that is to be analysed. Table 4.1 gives an overview of the tools discussed, the input accepted, output given, supported OS and capabilities. From the table, it is clear that there is a need to offer a solution that will offer more options. Our tool, in addition to analysing running processes, open files, network activities, PIDs will analyse both live and captured memory and it shall be open-source and free for every forensic investigator. This we believe shall assist most investigators who are not in positions of getting pro versions of the analysed forensic malware tools above to conduct investigations through the tool.

Tool	HB Garry Responder	Comae	Memoryze	Belkasoft	Volatility
Feature					
File formats Accepted	Raw memory dump	Raw memory dump, .json	Raw memdump, .dd, .E01	.mem	Raw memdump, .vmsn, .vmss
Output	OS information Open files Running Processes Registry files Network Activity Process IDs Malware	Machine Name OS Information File Size Upload time	Open files Malware Running Processes Loaded DLLs Network Activity	Files open Malware files present	Running Processes Network connections Network sockets Open files
Free/Pro Edition	PRO	Both Free and Pro. Free is limited in functionalities.	Free	PRO	Free
Ability to perform both Live analysis and imaged memory analysis	Yes	No. Does analysis on Captured memory images	No. Does analysis on Live memory.	Yes	No. Does analysis on Live memory only.
OS Supported	Win7,8,10	Win XP, 7,8,10	Win7,8,10	Win XP, 7	Unix, Win7

Table 0.1: Comparison of Analysed Tools and Their Capabilities

Chapter 5: Requirement Planning

5.1 Introduction

This phase is very crucial in the software development lifecycle. All requirements are defined in this phase. Any future changes made must be integrated in this document. Requirement specification will be used in the testing phase to confirm if the objectives are met. After using case studies and other secondary materials; the following requirements were identified. The requirements are both functional and non-functional

5.2 Functional Requirements

The major requirement is the development of a malware detection and analysis tool. The tool should be capable of detecting and recognising memory dumps extracted from the Windows memory, perform automated analysis of the dump, as well as generate a report based on the analysis.

Functionalities included in the tool are:

i. Scan E01 captured memory image
ii. Scan file in hard drive
iii. Scan folder in hard drive
iv. Scan partition in hard drive
v. Scan whole hard drive
vi. View processes running on live memory
vii. View process files present in live memory
viii. View registry files present in live memory
ix. View network activity in live memory
x. Generate reports for analysis performed.

5.3 Non-Functional Requirements

5.3.1 Hardware Requirements

1. Machine: Intel Core i3 or higher
2. Clock Speed: 2.5 GHz or higher
3. System Memory: 4 GB or higher

5.3.2 Software Requirements

1. Operating Systems: Windows 10/8/7
2. Database System running on SQLite
3. Front – end: Python
4. Back-end: Python

Chapter 6: System Design

6.1 Proposed System Modules

Design of a system is the first step in the development process. The main aim is usually to design a model of an object that will be built. The system's quality is adopted in this phase. A strong design translates into a stable system that can be tested.

The proposed system is a malware detection and analysis tool. The goal of the system is to identify the type of malware present in a memory dump at time of acquisition. The proposed system architecture has three main modules: Upload, Analyse, Reports.

Upload: This section will allow the investigator upload files and memdumps acquired from computers under investigation.

Analyse: After the upload of the memory image has been done, this module will initiate the process of analysing the file and try to detect presence of a malware, and if any, try to identify based on their characteristic behaviors and patterns from an already existing database of existing malware signatures. Analysis on live memory will also be done to determine processes, process files, registry files, resources consuming most CPU memory and time at the time of analysis. Analysis will also be conducted on computer hard drives and determine files present and whether they are infected with malware or not.

Report: Once analysis is complete, user will be able to generate a report on the analysis performed.

6.2 Logical System Design

This section aims to describe the whole system and the process by which is developed. It refers to the technical specifications that will be applied in implementing the proposed system. It includes process workflow, and the use case diagrams.

6.2.1 Use Case Diagram

A use-case diagram models the behavior of the system, a subsystem or class. It is more important for visualising the system, specifying and making systems and subsystems approachable. Our tool has three functionalities, one for analysing hard drives for malware, second for analysing captured memory images for malware and lastly for analyzing live memory for processes running. This is illustrated in figures 6.1, 6.2 and 6.3.

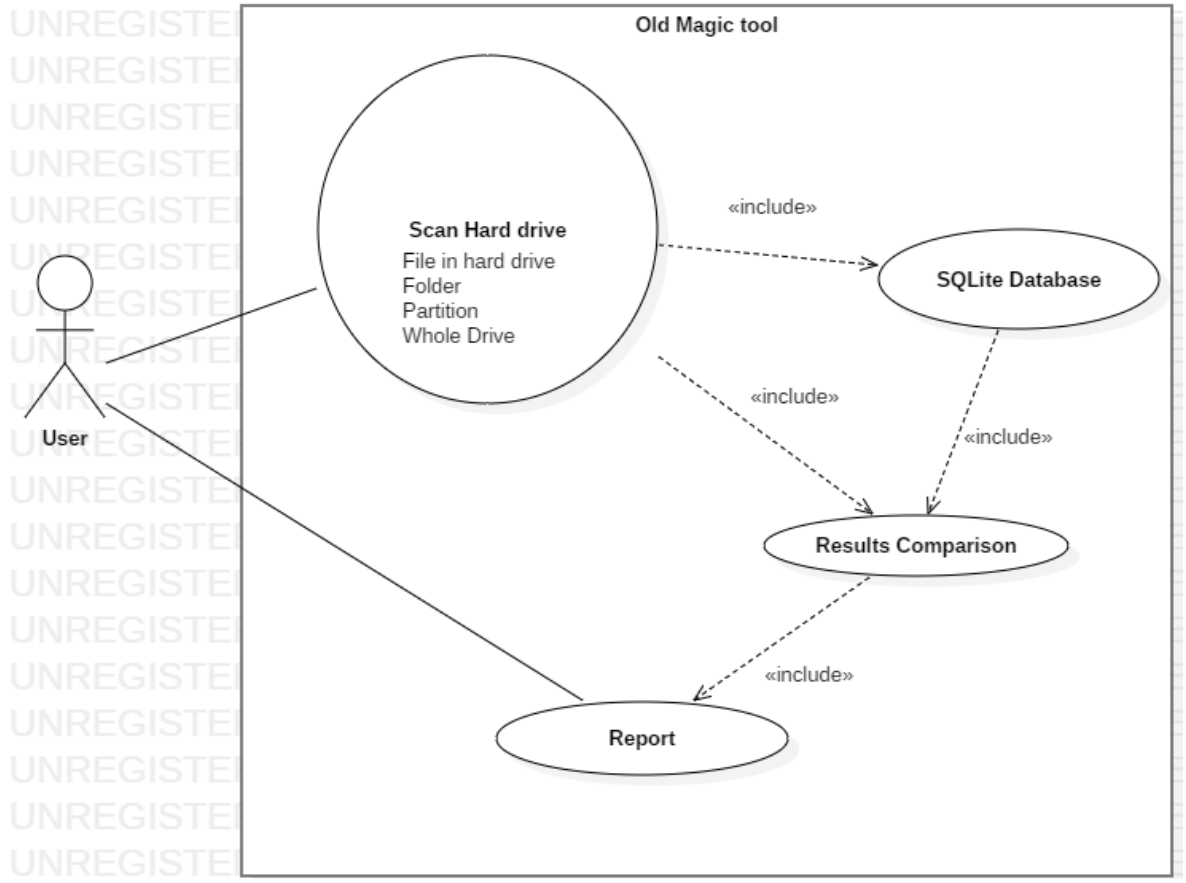


Figure 0.1: Use Case Diagram for Analysing Hard Drive

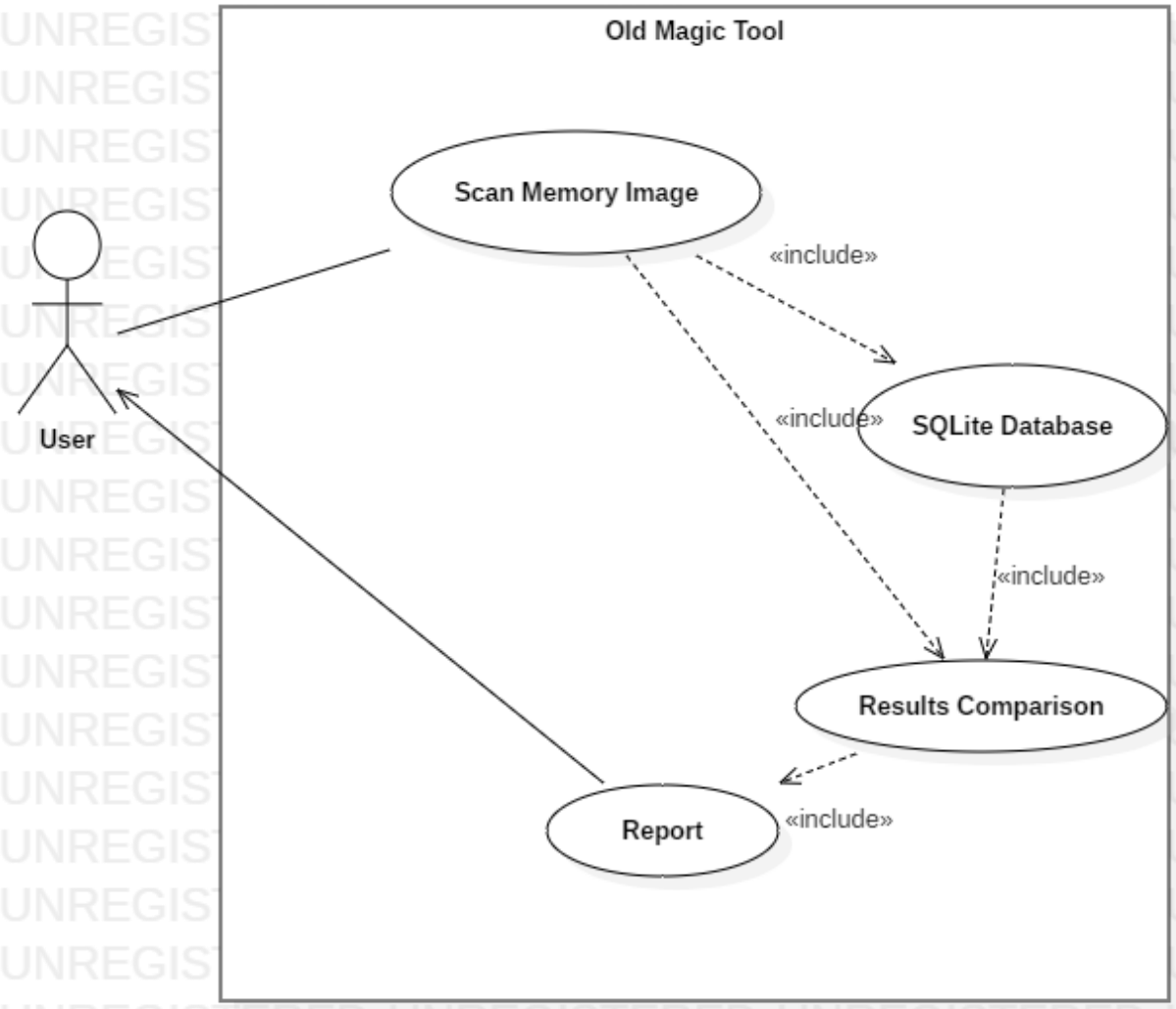


Figure 0.2: Use Case Diagram for Analysisng Captured Memory Images

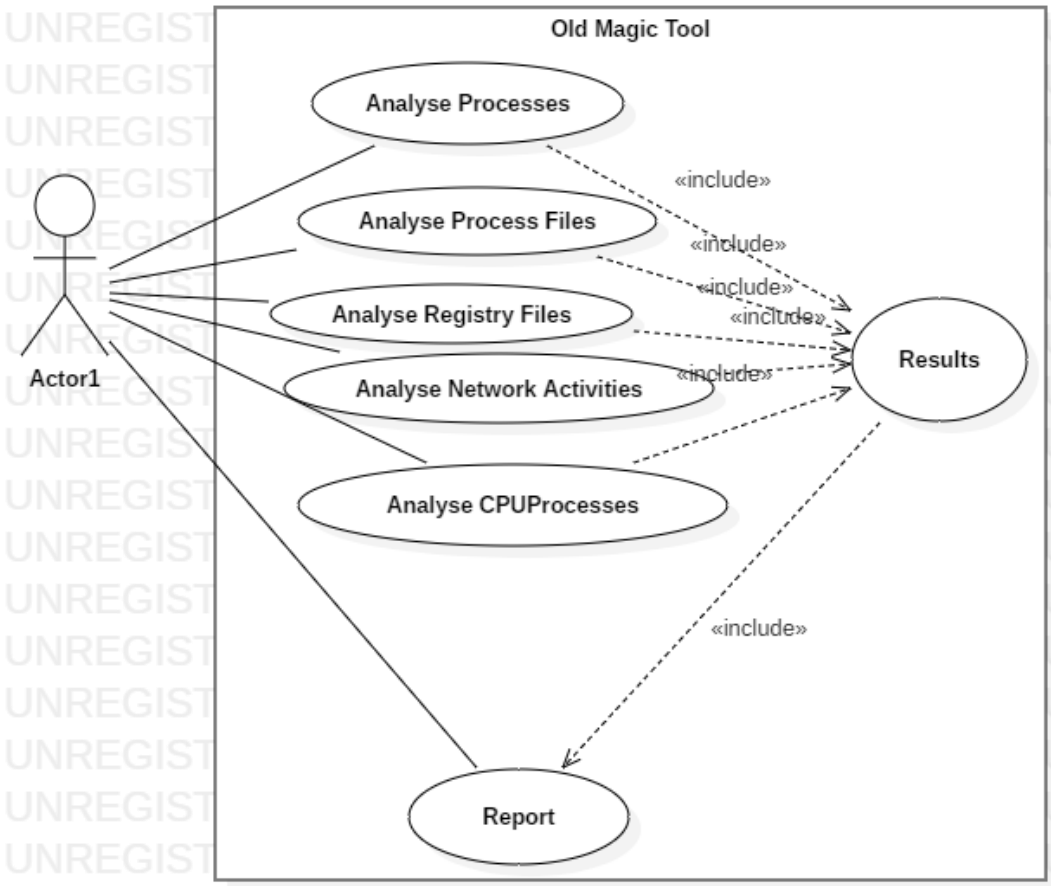


Figure 0.3: Use Case Diagram for Analysing Live Memory

6.2.2 Flow Chart Diagram

A flowchart is a diagram that represents an algorithm, workflow or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows. Our tool has three functionalities, analyzing live memory, analyzing computer hard drive and analyzing captured memory images. This is illustrated in figures 6.4, 6.5 and 6.6.

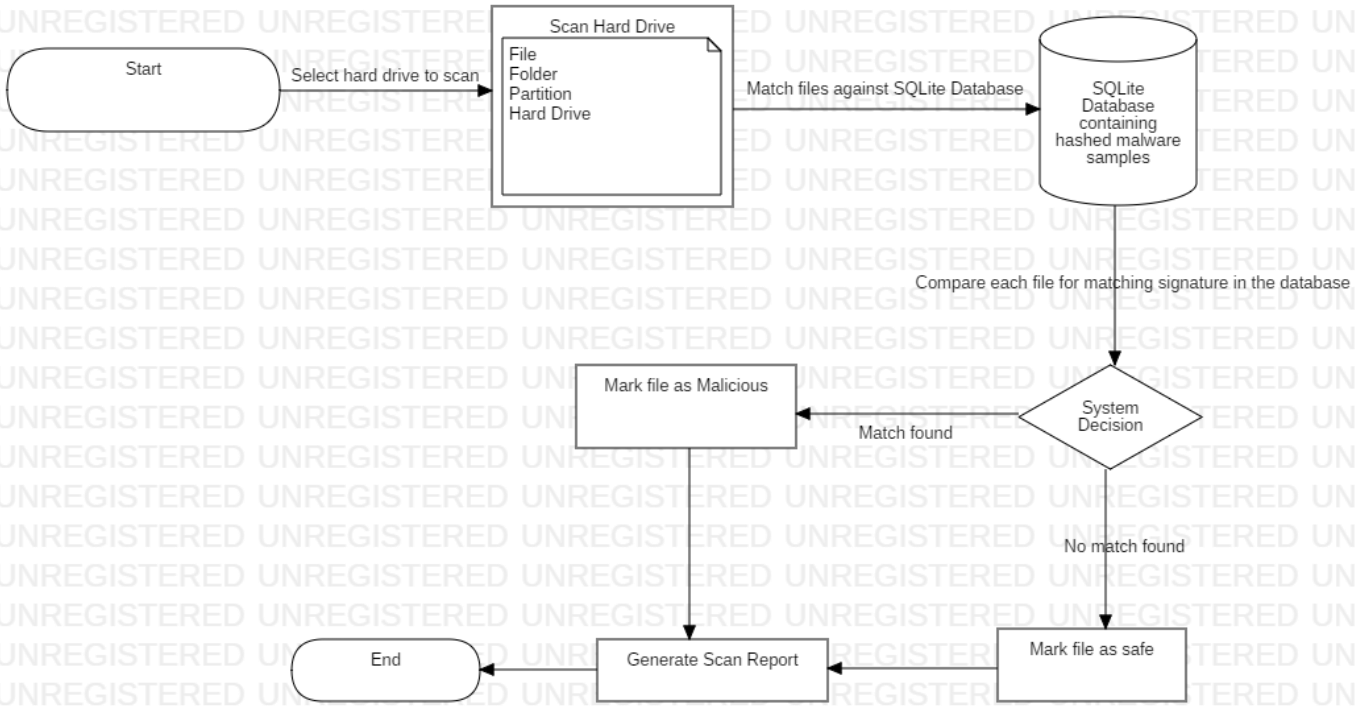


Figure 0.4: Flow Chart Diagram for Analysis of Hard Drive

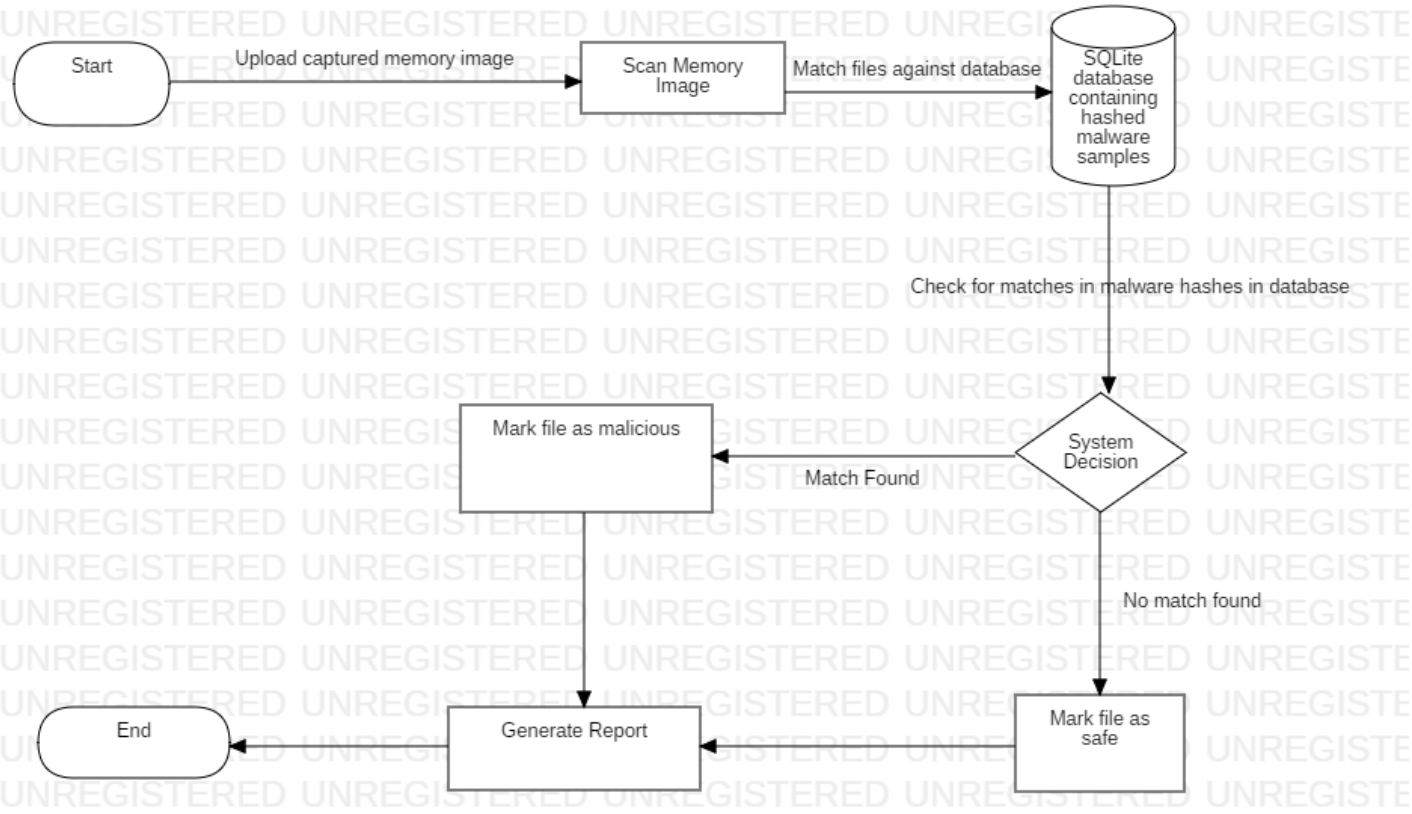


Figure 0.5: Flow Chart Diagram for Analysis of Captured Memory Image

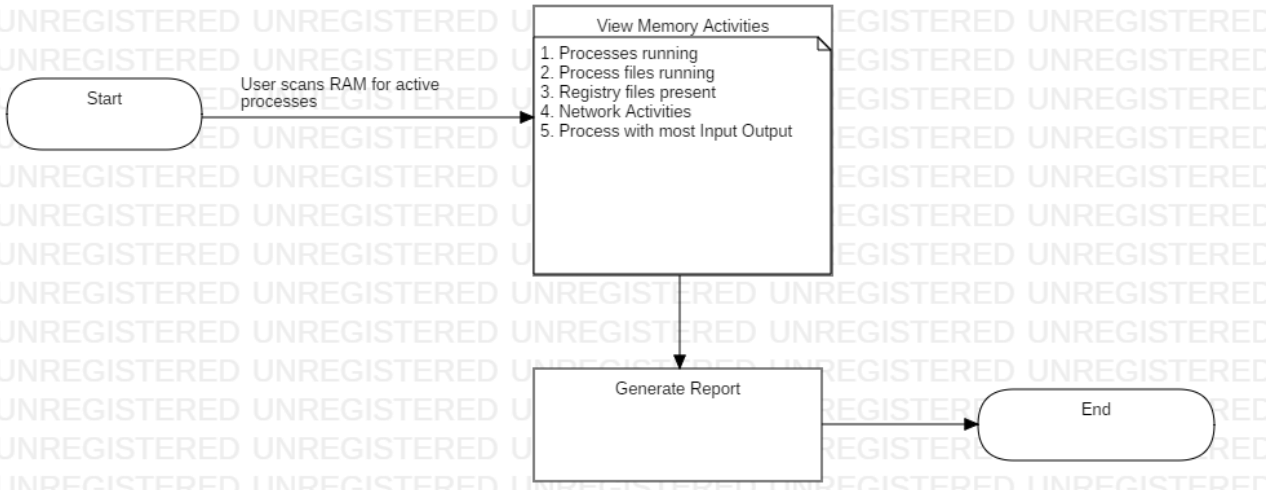


Figure 0.6: Flow Chart Diagram for Analysis of Live Memory

6.2.3 Sequence Diagram

Sequence Diagram defines the manner in which the user feeds data into the system and the manner in which information is displayed back to the user. Figures 6.7, 6.8 and 6.9 illustrate the sequence of activities for the three functionalities of our tool.

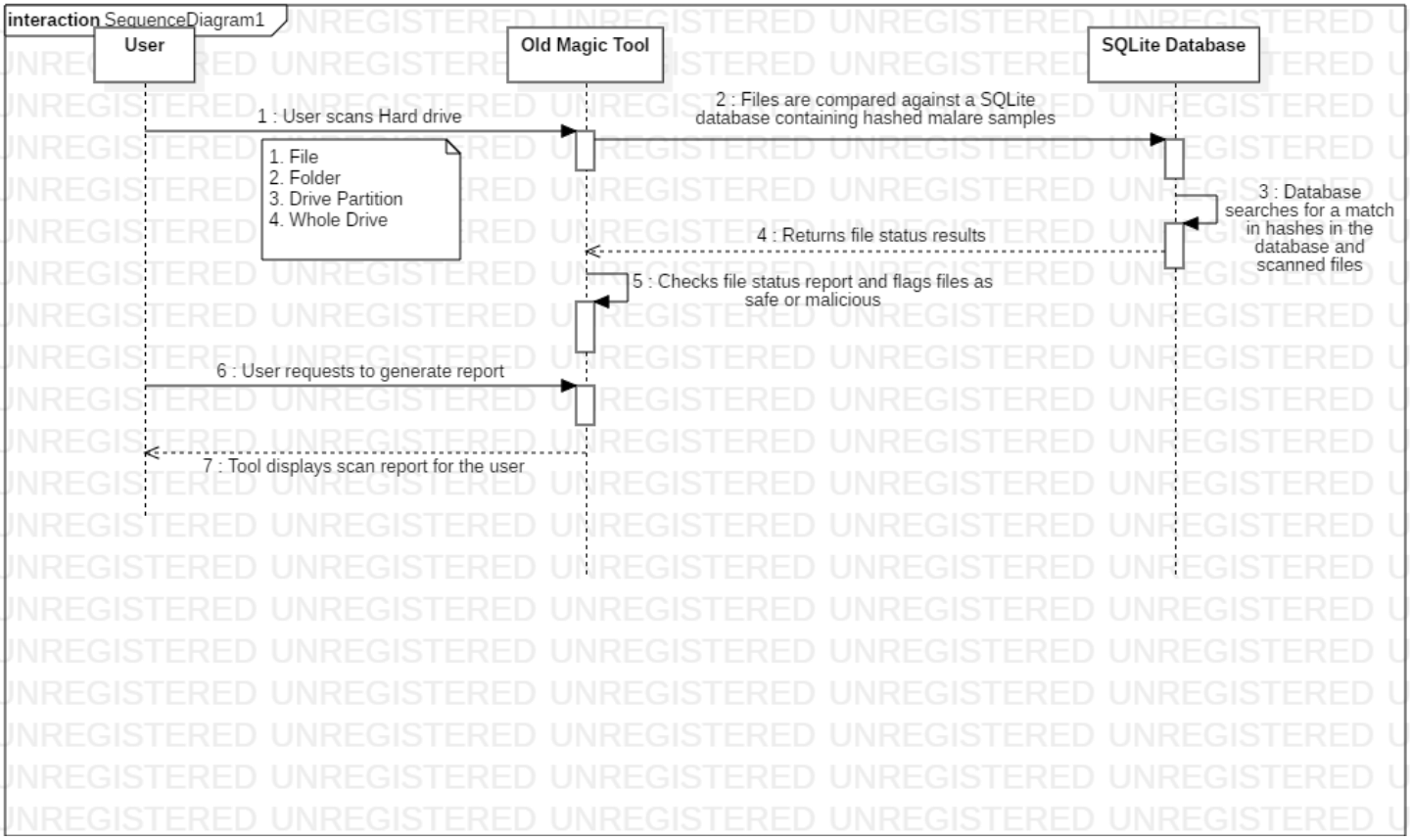


Figure 0.7: Sequence Diagram for Analysis of Computer Hard Drive

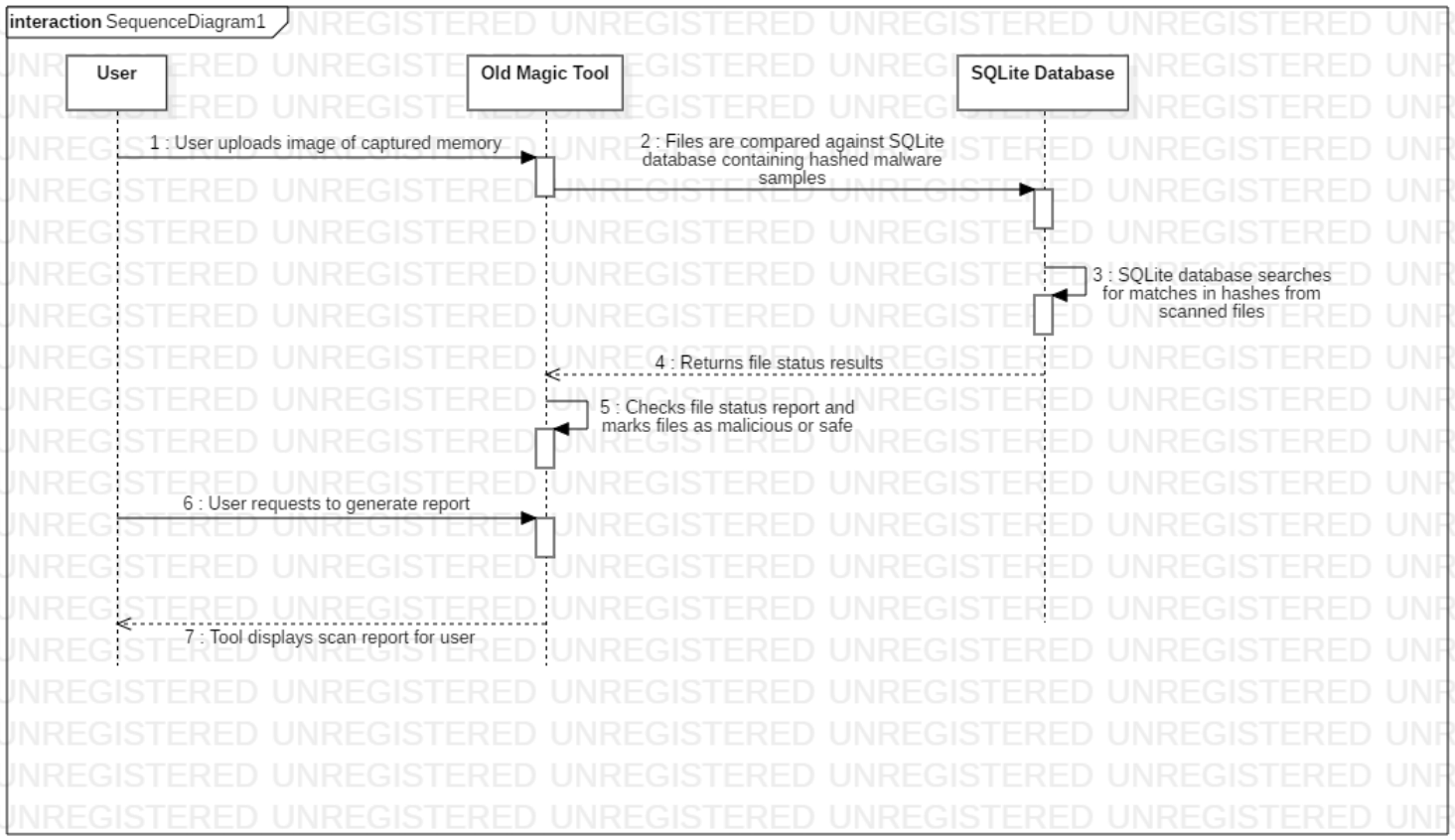


Figure 0.8: Sequence Diagram for Analysis of Captured Memory Image

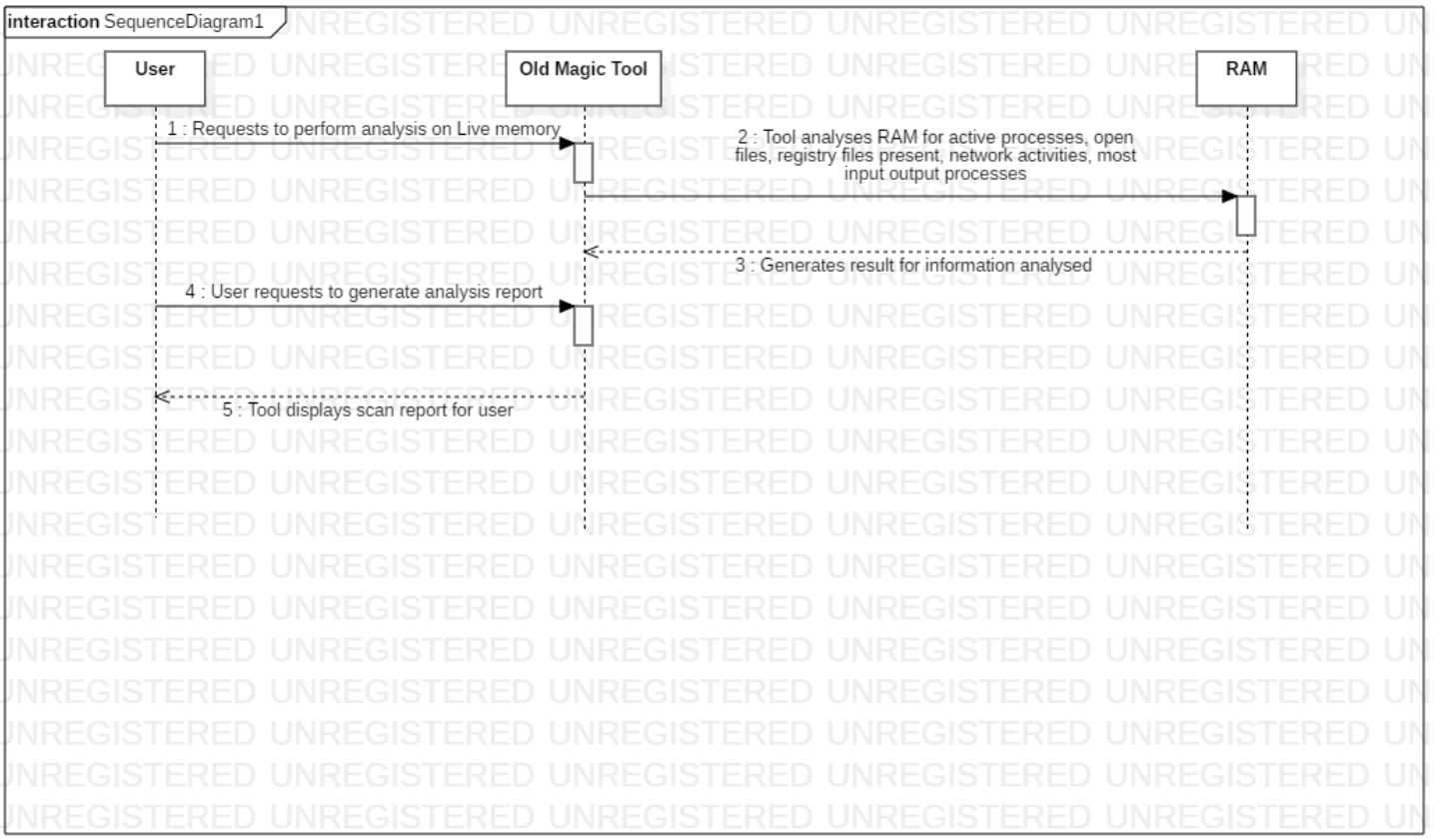


Figure 0.9: Sequence Diagram for Analysis of Live Memory

6.2.4 Wireframes

A wireframe is a visual guide that represents the skeletal framework of a system. Wireframes are created for the purpose of arranging elements as they will appear on the actual product (Garret & James, 2010). The wireframe depicts the page layout or arrangement of the system's content, including interface elements and navigational systems, and how they function. Wireframes usually lack typographic style, color, or graphics, since the main focus lies in functionality, behavior, and priority of content. Wireframes can be pencil drawings or sketches on a whiteboard, or they can be produced by means of a broad array of free or commercial software applications. Our wireframes were created using online visual paradigm software. Figures 6.10, 6.11, 6.12, 6.13 and 6.14 show application screens and sample output data.

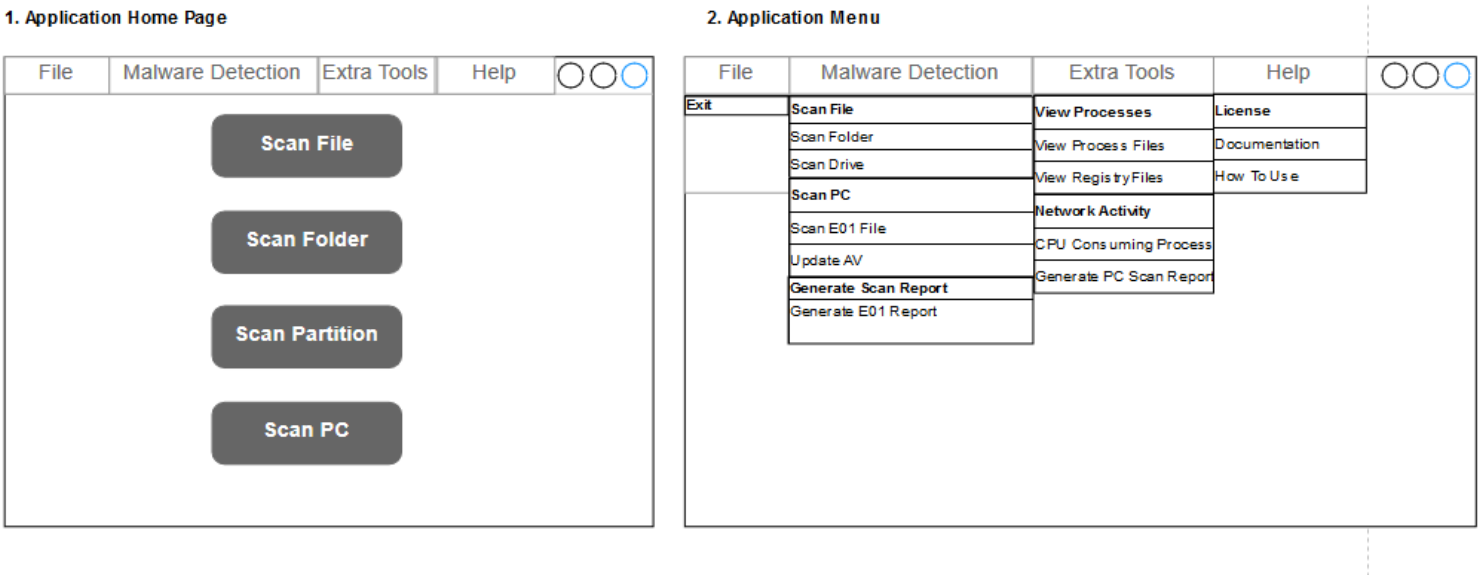


Figure 0.10: Application Wireframe 1

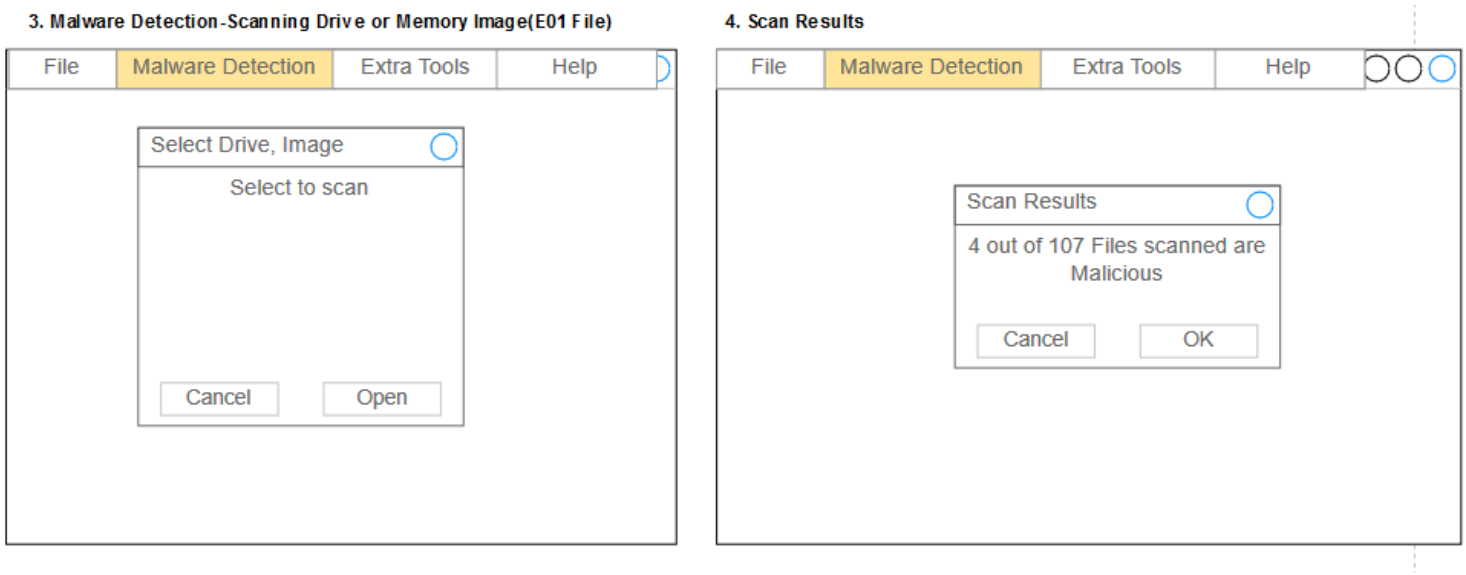


Figure 0.11: Application Wireframe 2

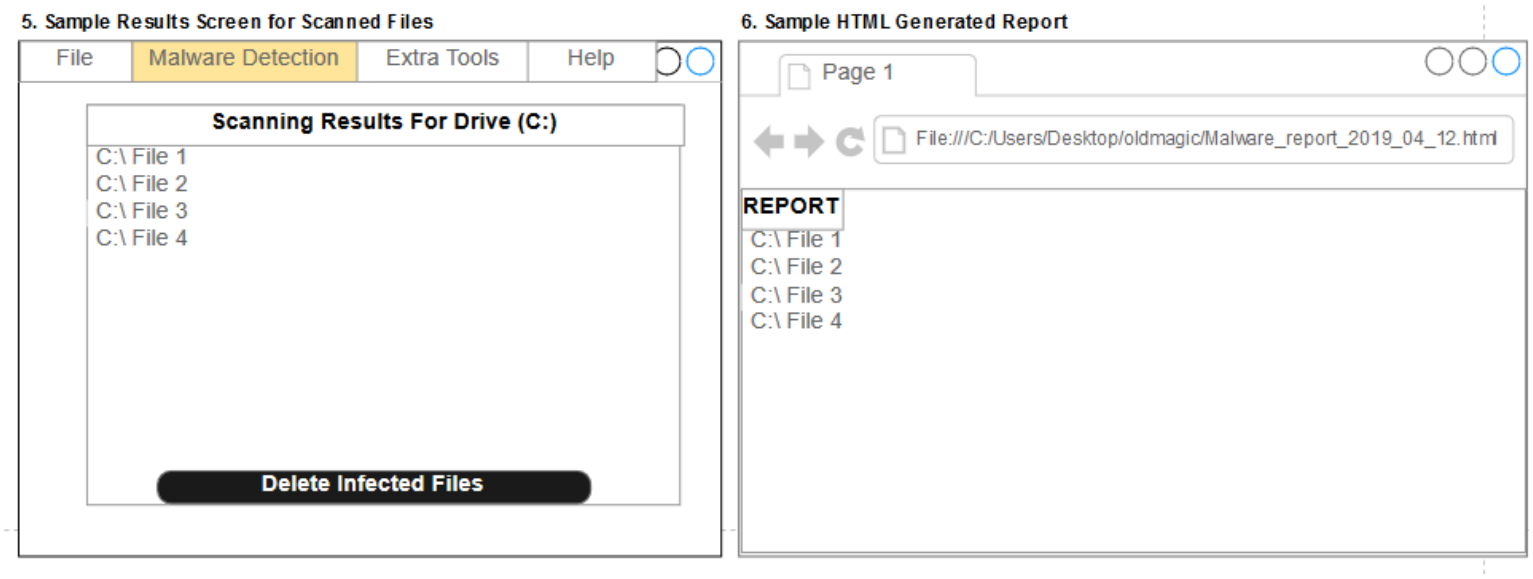


Figure 0.12: Application Wireframe 3

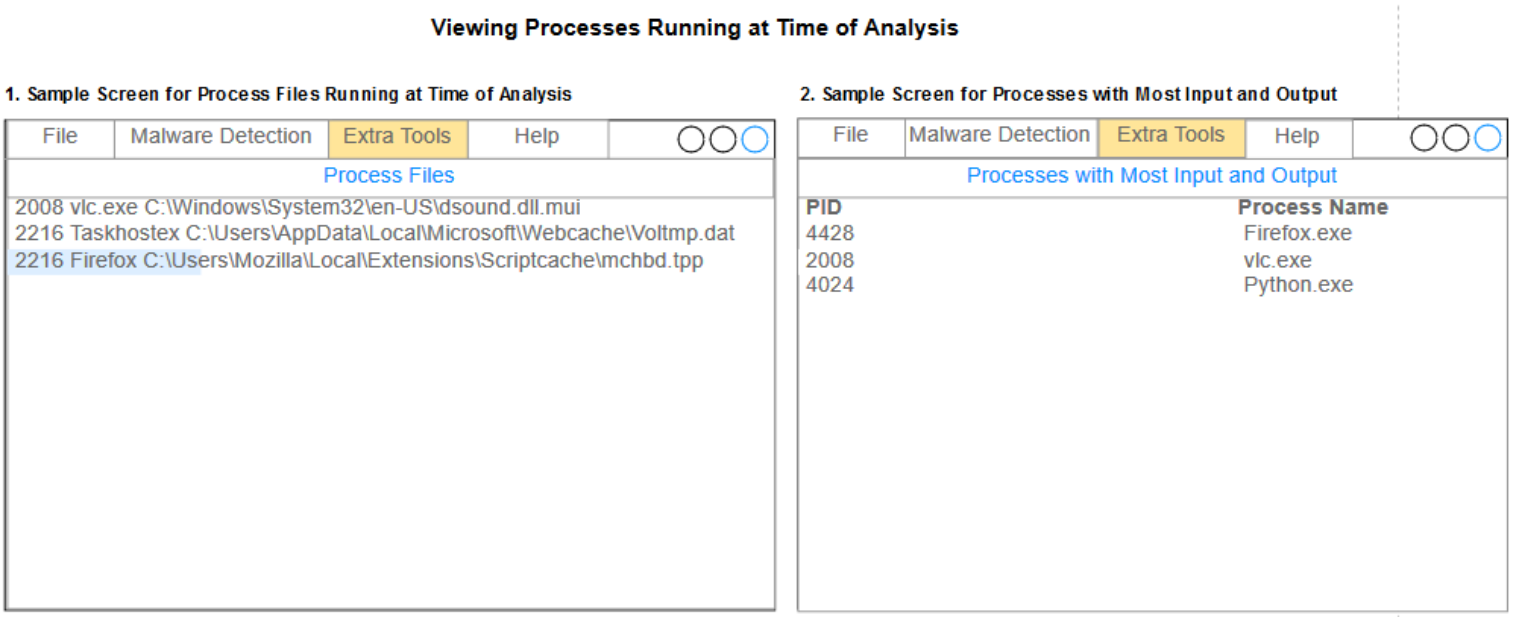


Figure 0.13: Application Wireframe 4

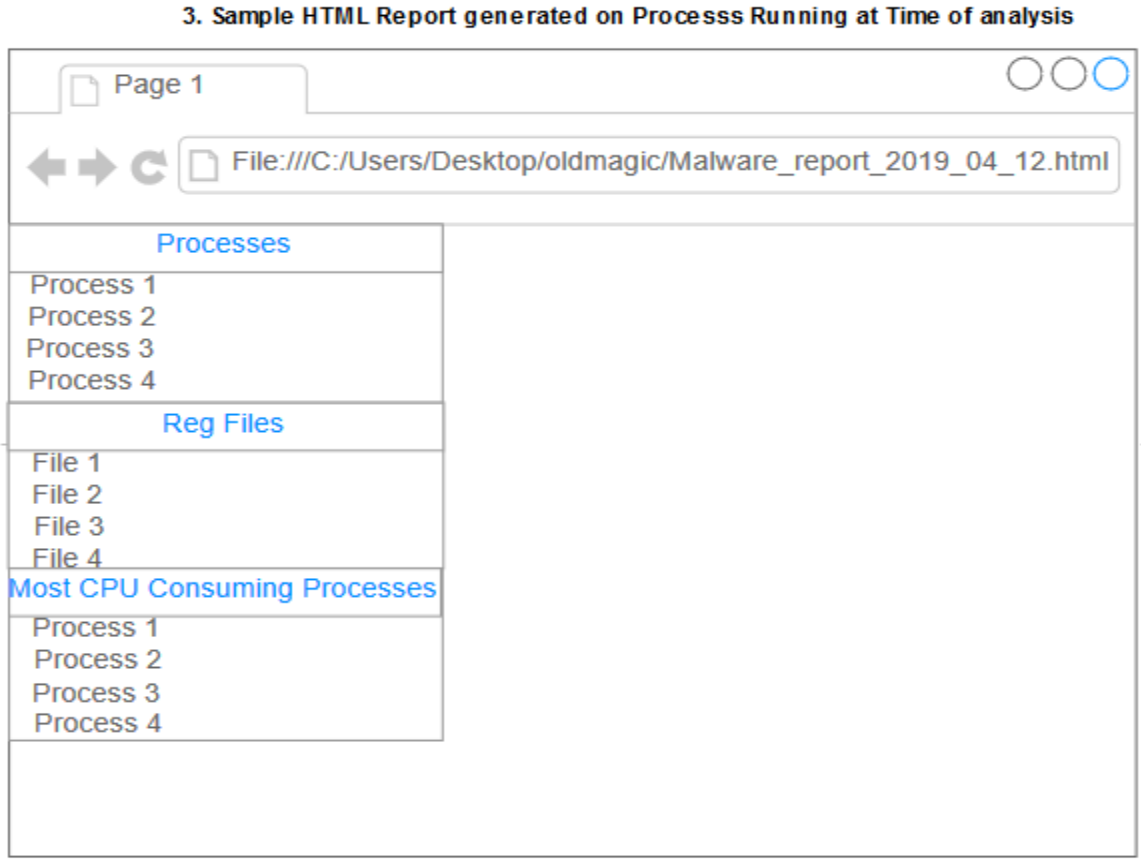


Figure 0.14: Application Wireframe 5

6.3 System Architecture

The system is designed to have a two-tier architecture with frontend client interface and a backend, which is hosted remotely. The client interface consists of a python based interface that allows the user to upload files, scan folders, hard drives or files and generate reports for scanned files through a supported web browser that displays information in a presentable manner.

The backend consists of a SQLite database that contains millions of MD5 hashed malware signatures. During analysis of files, every file is crosschecked against this database to match them against worldwide known malware signatures. Once a match is found, the file is flagged as malicious or potentially harmful and a report is generated for further analysis. The client interface and backend communicate with each other through the internet. Figure 6.15 shows the system architecture.

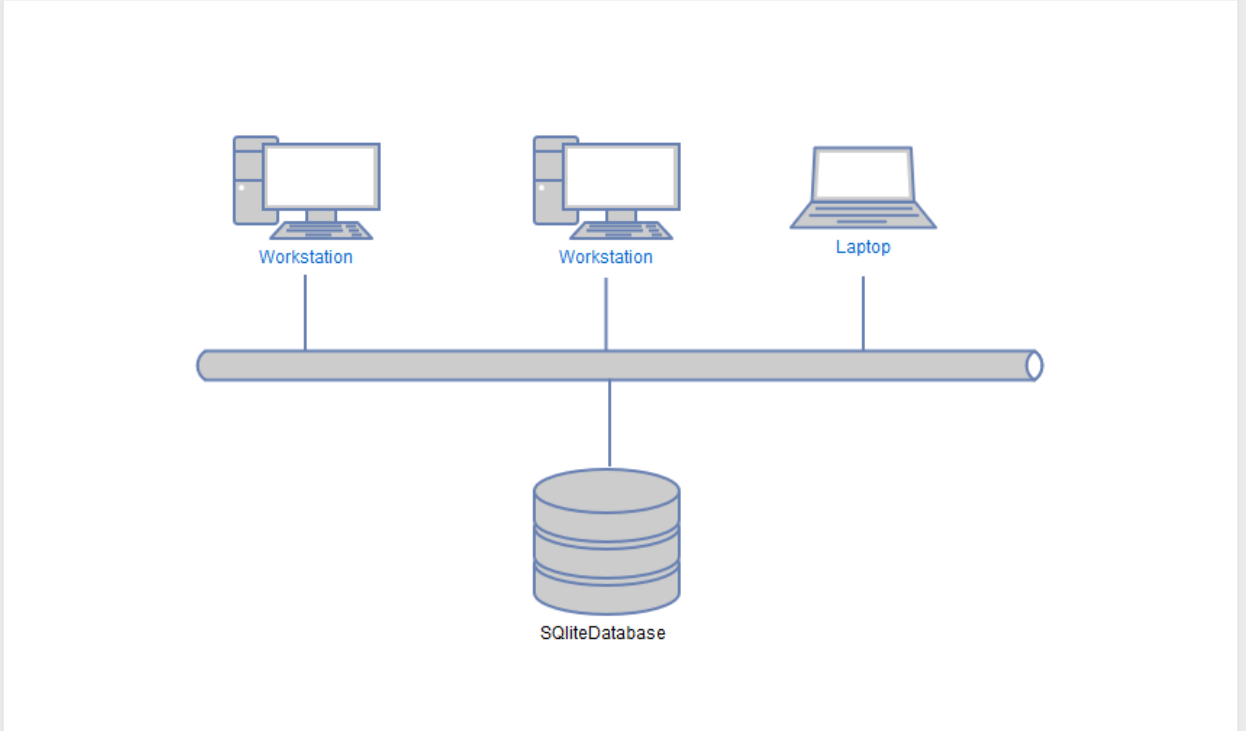


Figure 0.15: System Architecture

Chapter 7: System Implementation, Testing and Validation

This chapter describes in detail how the implementation, testing and validation of the tool is carried out and the results obtained. Implementation is carried out in accordance to the methodology outlined in Chapter 3 of this research. Testing and validation were carried out on locally deployed version of the tool.

7.1 Implementation

The implemented tool is dubbed Old Magic. The implementation of the tool incorporates requirements and specifications identified in the system analysis so as to implement the system design and architecture. The tool has three modules; the malware detection, extra tools and help modules as illustrated in figure 7.1.

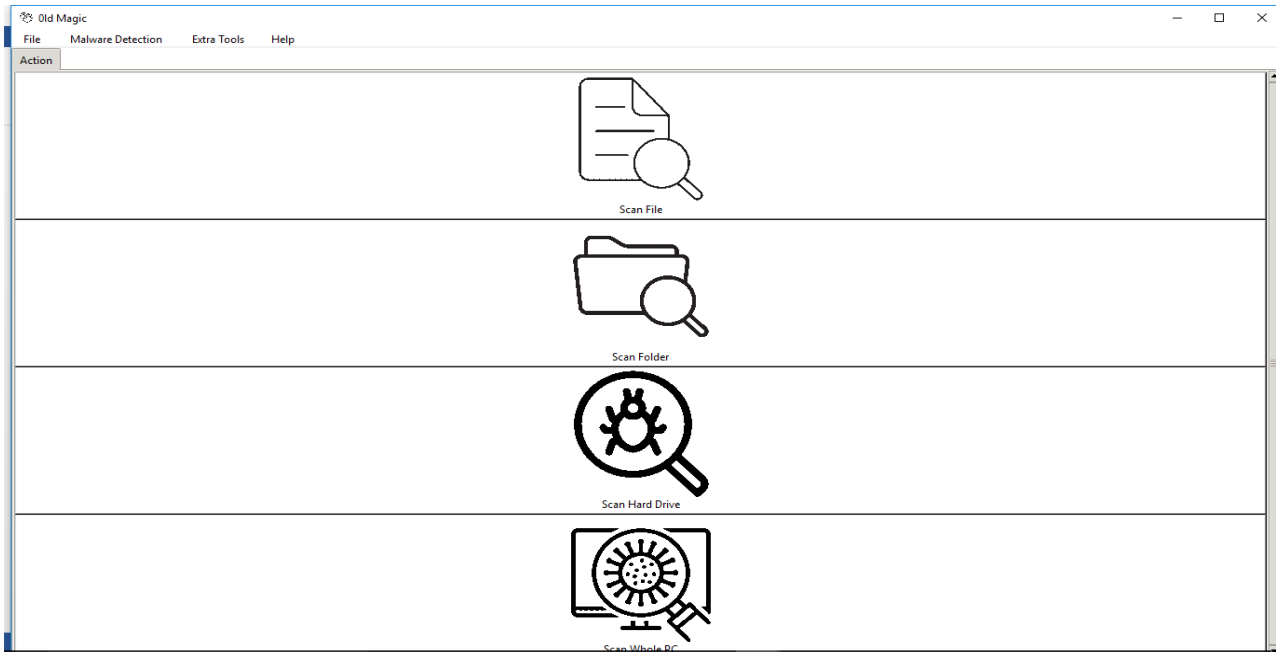


Figure 0.1: Application Home Page

The malware detection module enables us to scan suspect computer hard drives. To scan hard drive, the investigator is able to scan a single file, a folder in the suspect computer, a partition within the hard drive or the whole hard drive for malware. The investigator is able to upload captured memory image through the scan E01 file menu present under malware detection. This module also enables user to generate scan reports for analysed and scanned files in .html format. The malware detection menu is illustrated in figure 7.2.

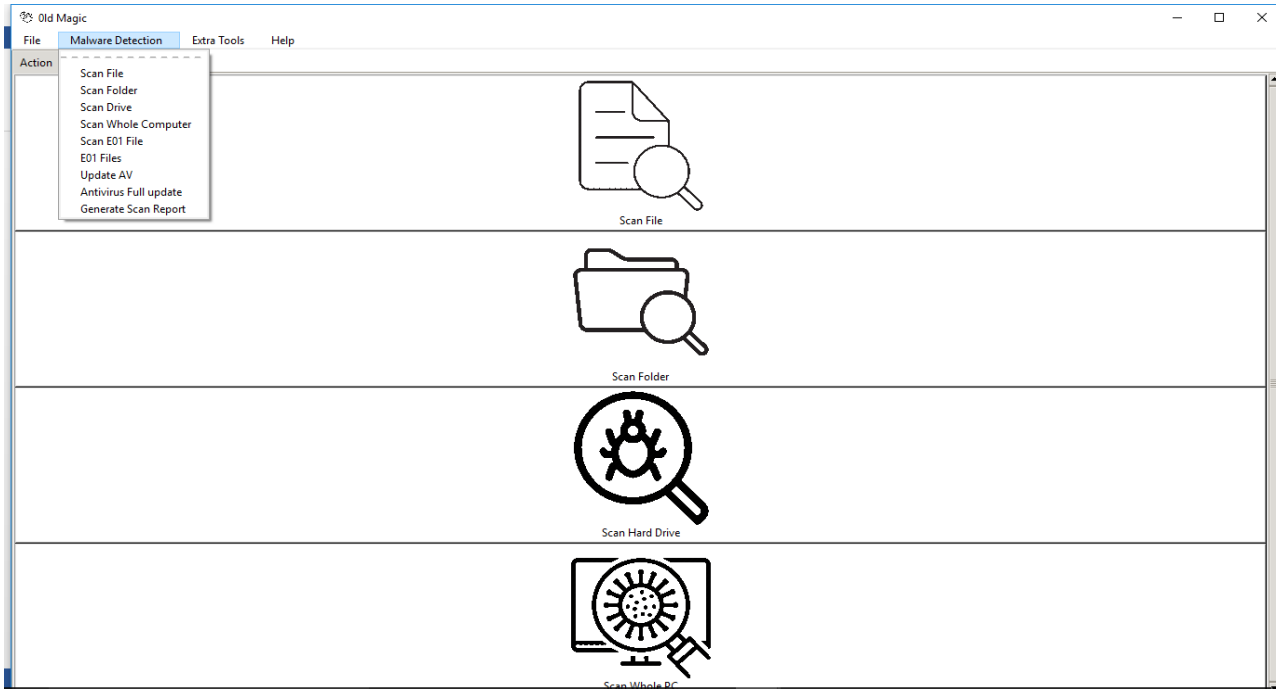


Figure 0.2: Malware Detection Menu

The extra tools module enables user to analyse live memory for open files and processes running as shown in figure 7.3.

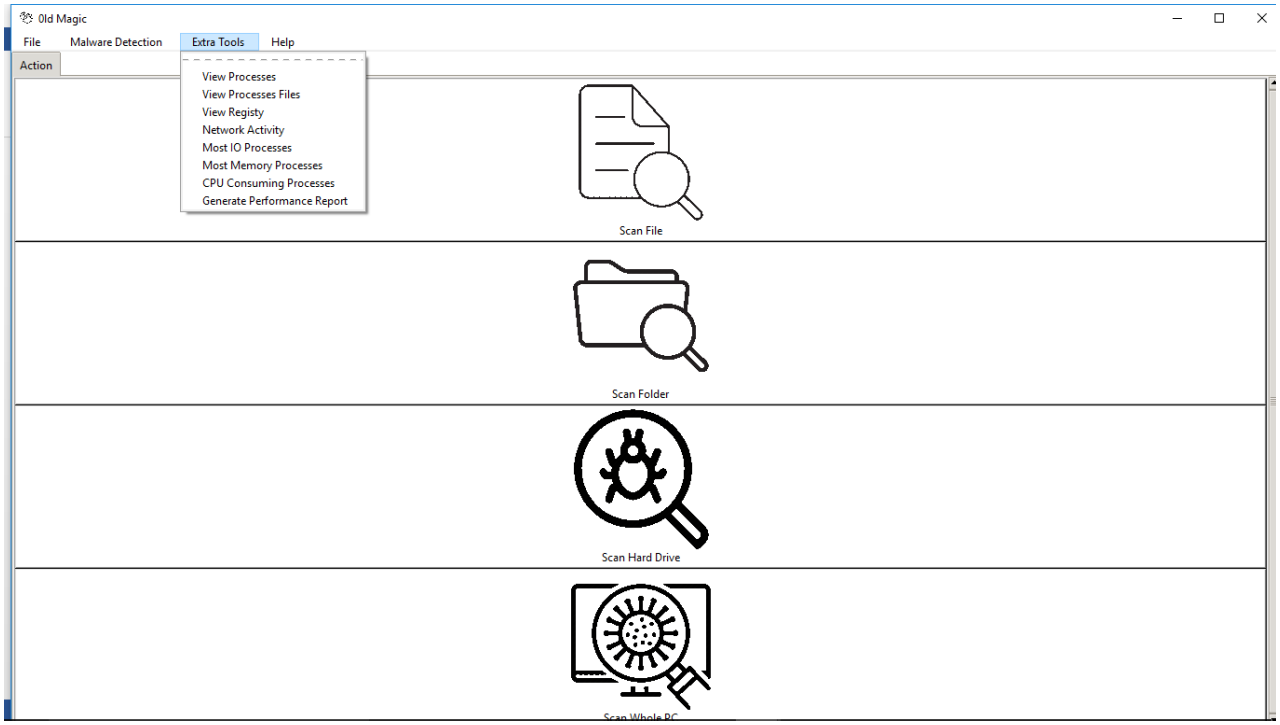


Figure 0.3: Extra Tools Menu

User is able to view CPU memory in terms of which processes and their process files are running at time of analysis, registry files present at time of analysis, network activities and ports open, most input and output processes, processes utilising most CPU memory as well as processes consuming most CPU time. This module allows the examiner to generate a report on all processes running at time of analysis.

The help module gives users a guide on how to use the software, product documentation as well as the software license information. This is illustrated in figure 7.4.

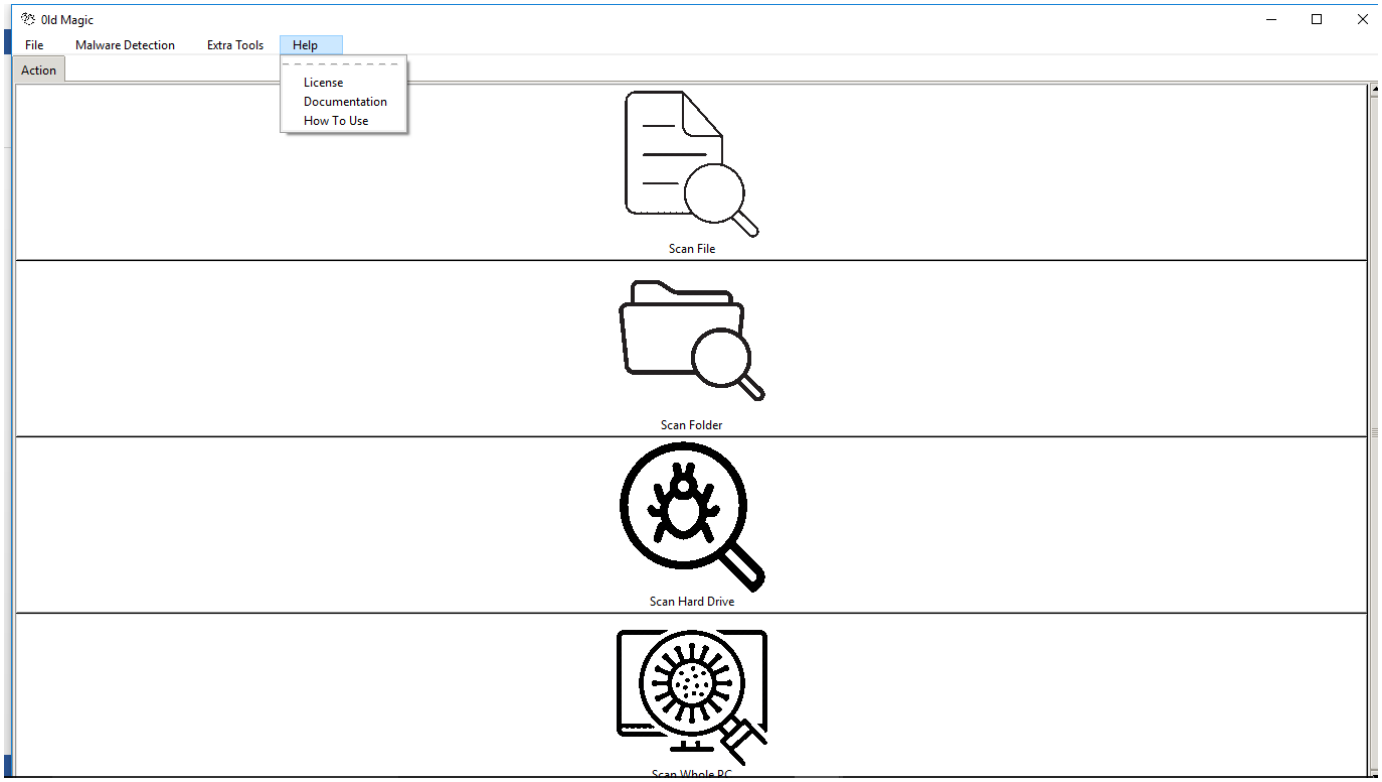


Figure 0.4: Help Menu

The tool can upload a memory image captured from a suspect computer. The tool is also able to scan computer drive file, folder, hard drive partition or the whole drive. The tool also performs analysis on live memory. A database of worldwide known malware signatures hashed in MD5 was used in order to compare against analysed files. Our database fetched this data from virusshare.com, one of the largest online malware repositories that is updated daily according to Kumara and Jaidhar (2017). Based on this, files are flagged when the file signature is matched against a known malware signature. Once analysis is complete, crucial information on malware identified is displayed through a user interface developed using Python programming language. This is discussed under the testing phase.

7.2 Testing

The tool was run to test for functionality and any bugs. Testing was done in a systematic way in that every functionality of the tool was tested to ensure it was working as expected. Testing was done on the functionalities highlighted below.

i. Analysis of hard drive. This included scanning files, folders, disk partitions as well as whole hard drive.
ii. Analysis of captured memory images.
iii. Analysis of live memory.
iv. Generation of reports for files and processes scanned.

The above will be highlighted under the functional testing section.

7.2.1 **Functional Testing**

Functional testing checks if core requirements and functionalities are met.

7.2.1.1 Scanning File

The investigator is able to scan a single file and determine whether it is malicious or not. The investigator selects scan file functionality, a selection box will appear where they can select file to scan as shown in figure 7.5.

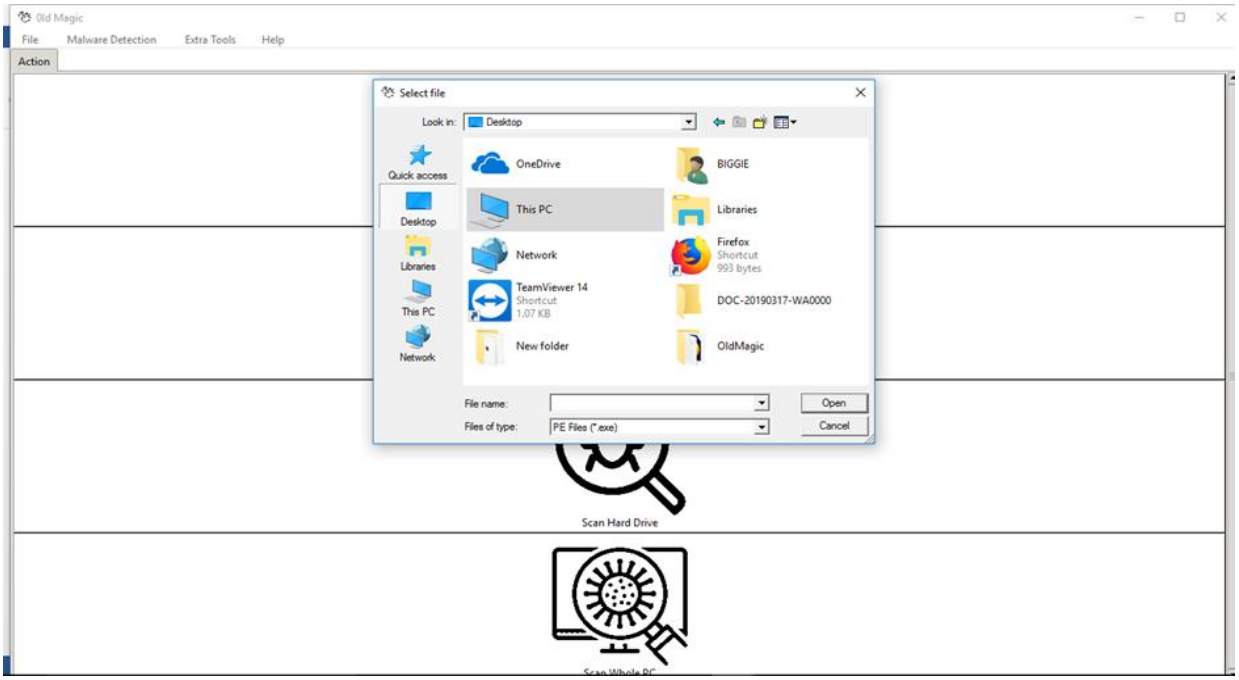


Figure 0.5: File Scanning Page

Once a file is selected it is scanned and the results given as highlighted in figure 7.6.

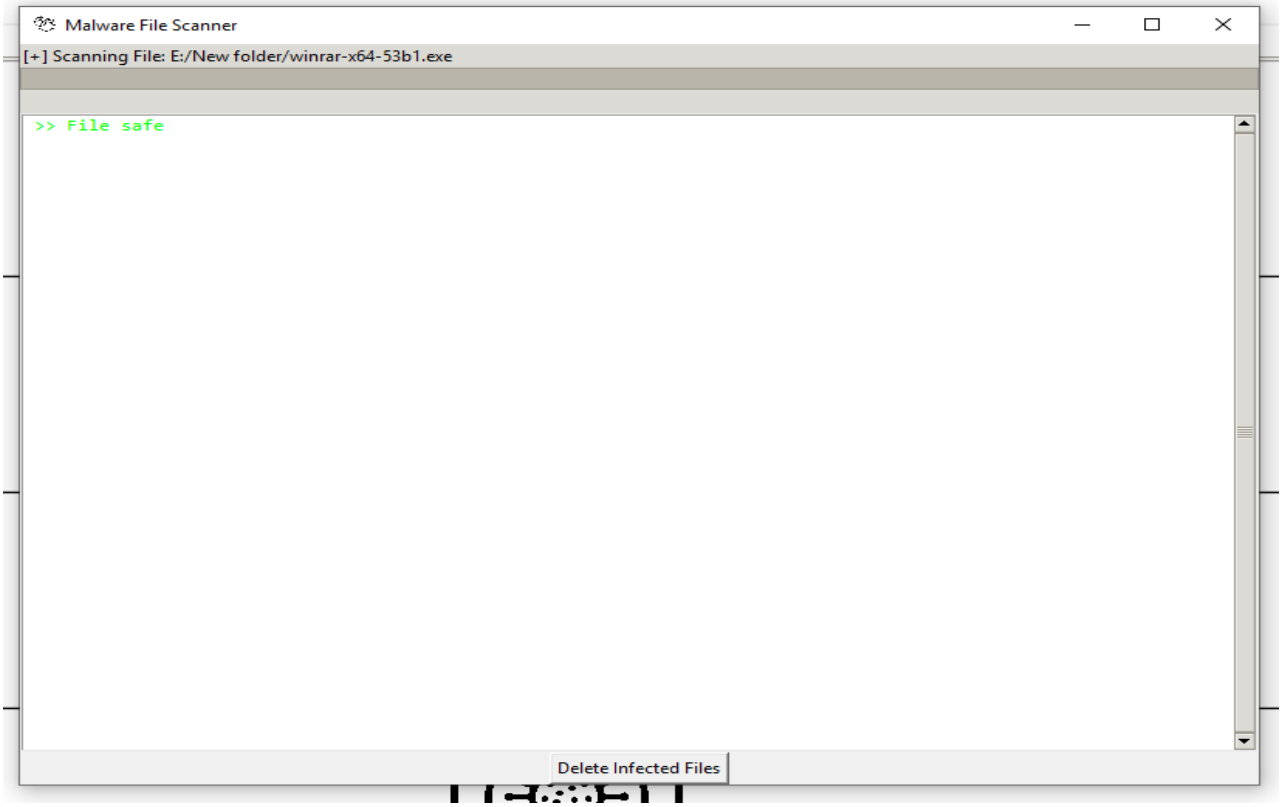


Figure 0.6: File Scan Results Page

7.2.1.2 Scanning Folder

The scanning folder functionality is showcased through the application homepage that allows investigator to select a target folder as illustrated in figure 7.7.

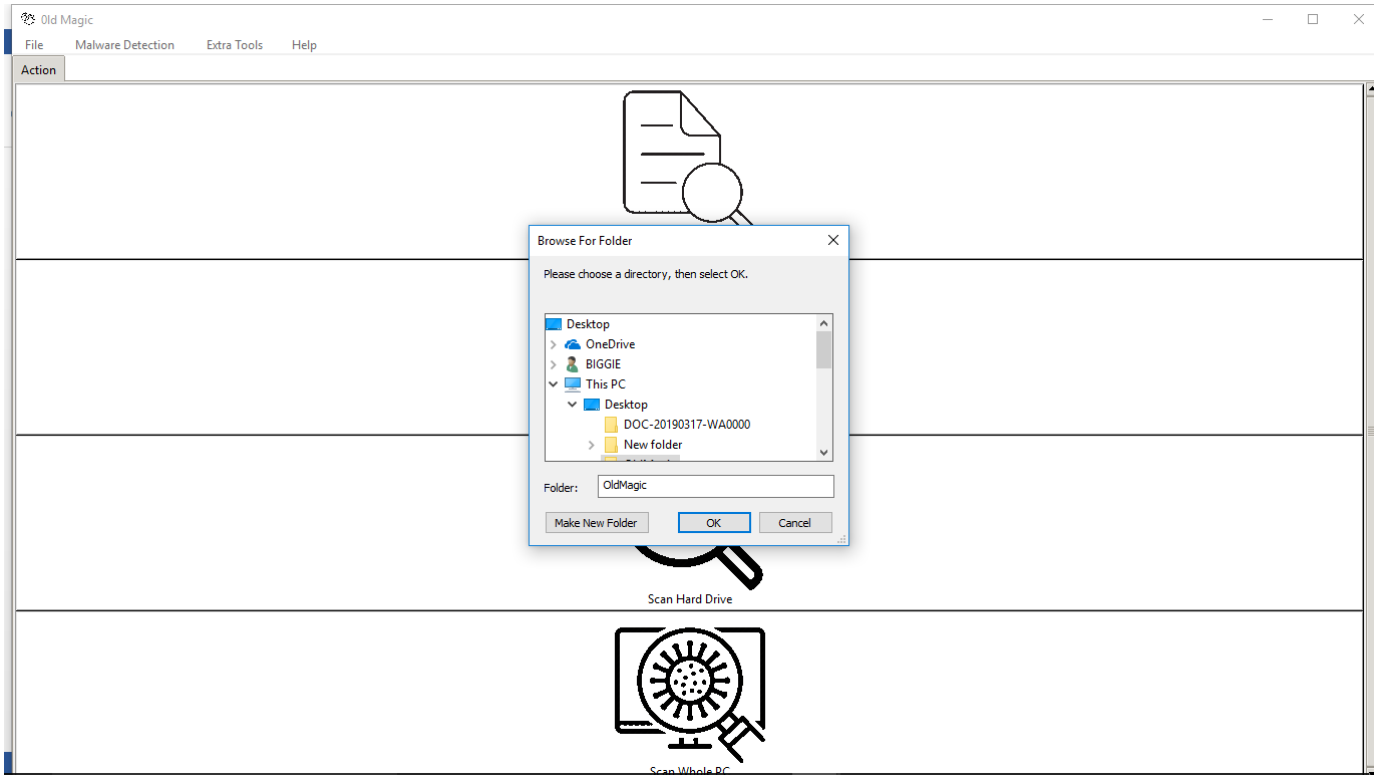


Figure 0.7: Application Folder Scanning Page

After analysis is done the tool gives the total number of files scanned and how many files were flagged to be potential malwares. This is illustrated in figure 7.8 and 7.9.

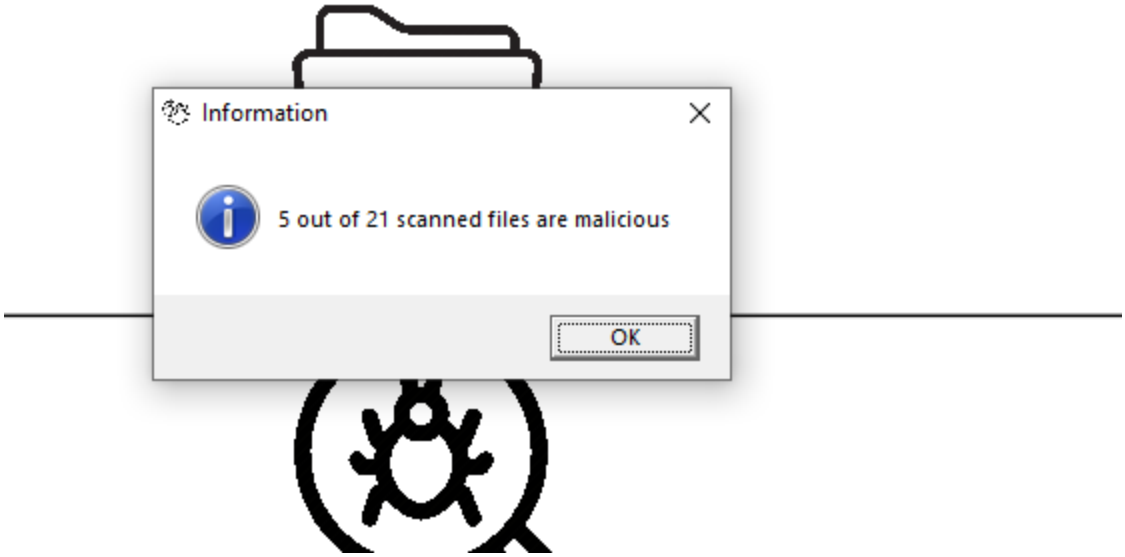


Figure 0.8: Folder Scanning Output

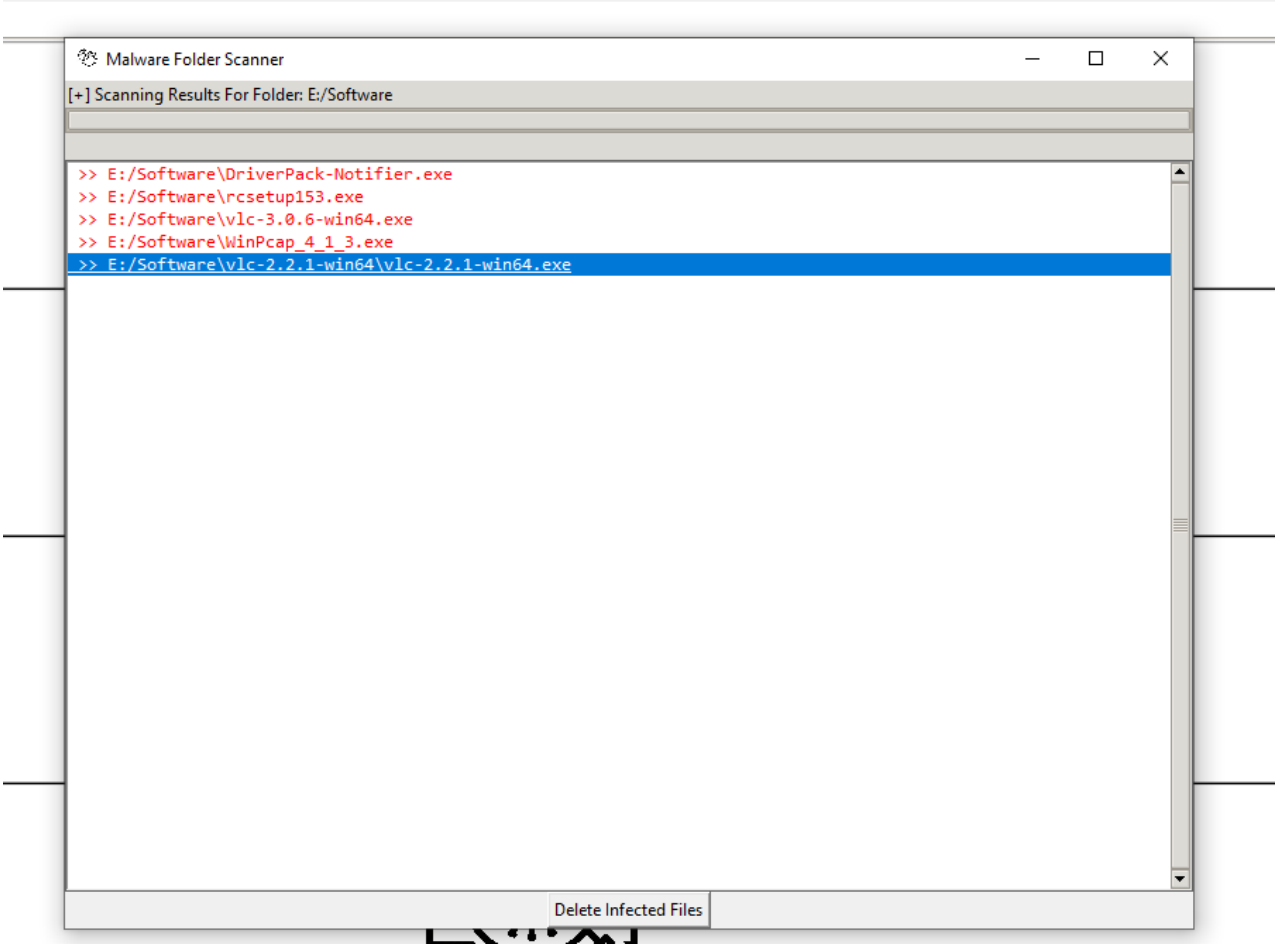


Figure 0.9: Folder Malware Scanning Results

7.2.1.3 Scanning Hard Drive Partition

The application page that allows investigator to scan a particular partition of the hard disk is illustrated in figure 7.10. From this page the investigator is able to select which partition they want to analyse.

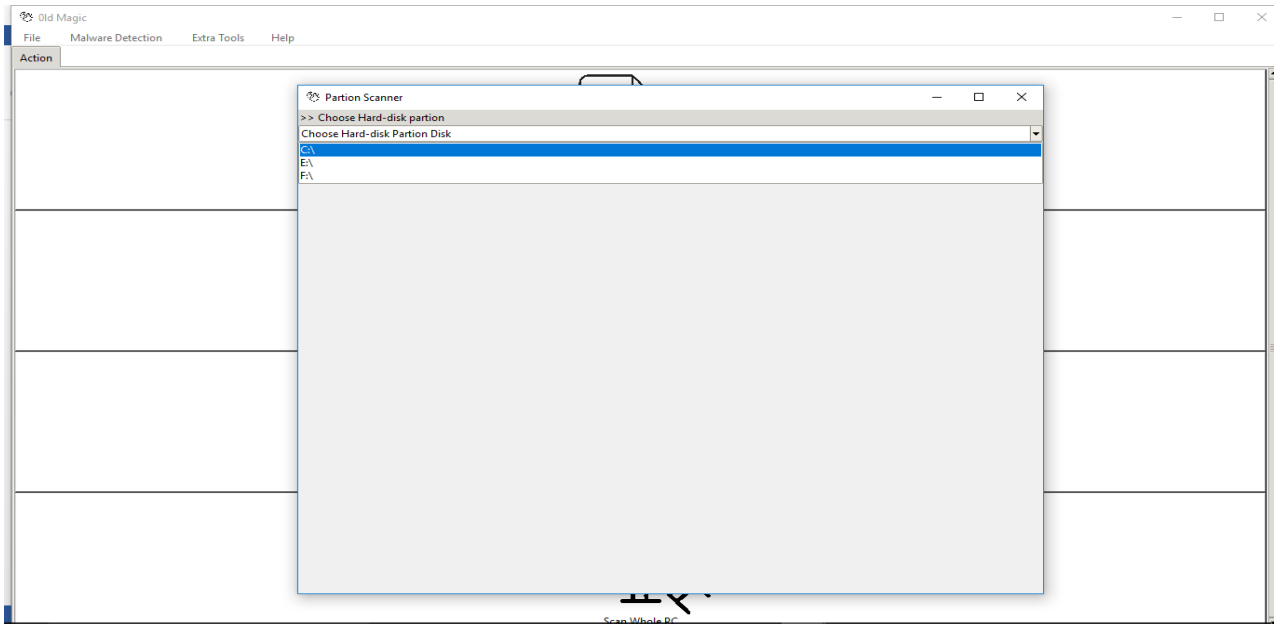


Figure 0.10: Application Hard Disk Partition Page

After scanning, the tool gives an analysis of how many suspected malicious files have been found and also gives the total count of the number of files scanned. It also lists down the individual suspected files as shown in figures 7.11 and 7.12.

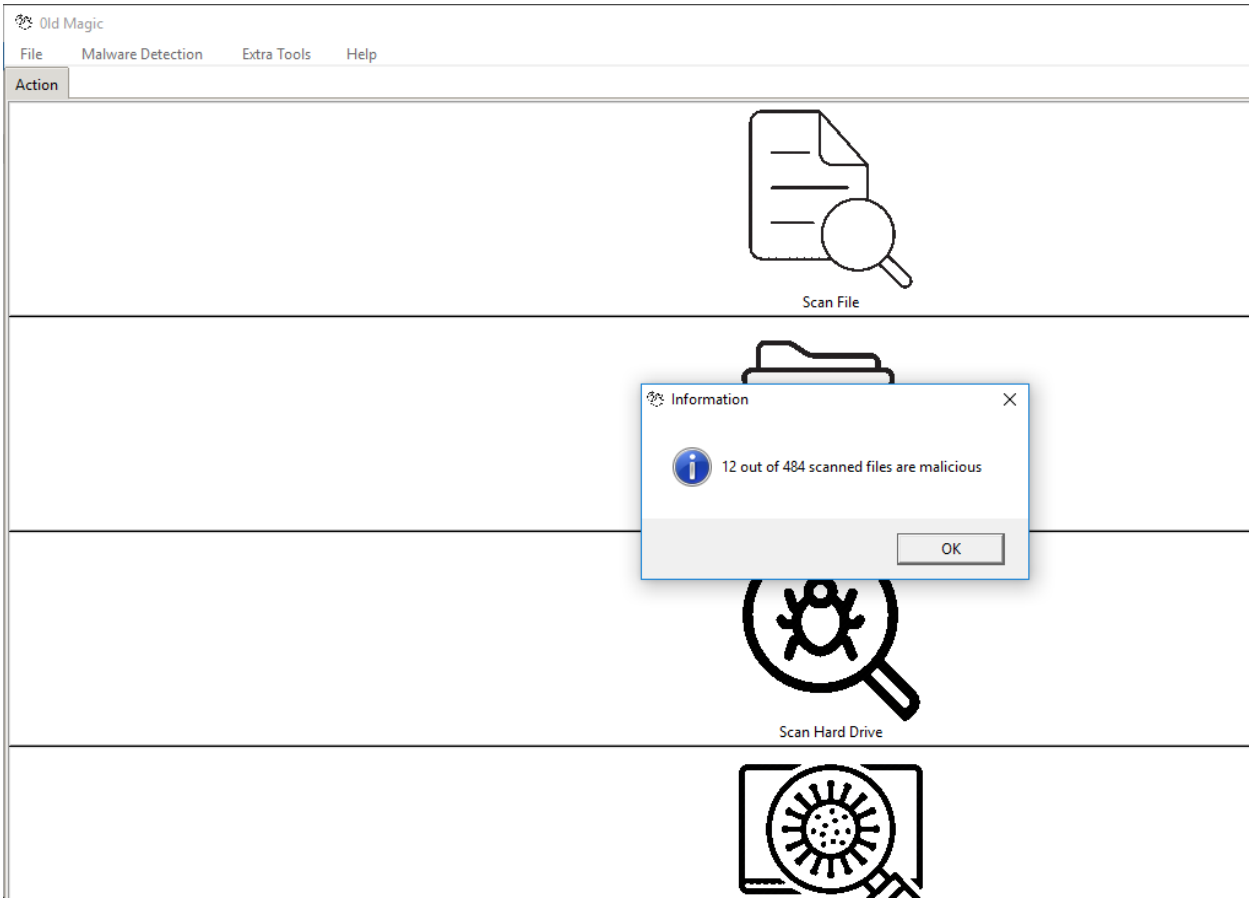


Figure 0.11: Scanning Output Page

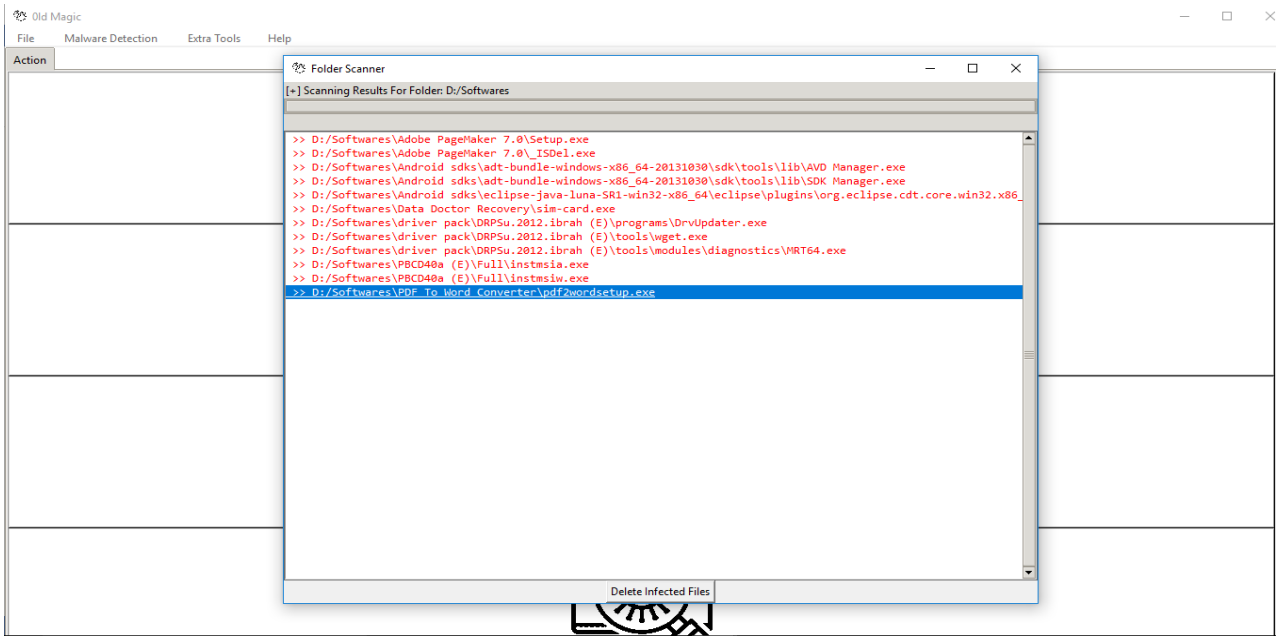


Figure 0.12: Hard Drive Partition Scan Results

7.2.1.4 Scanning Whole PC

This involves scanning the whole computer hard drive. The application page that allows investigator to scan the whole PC is illustrated in figure 7.13.

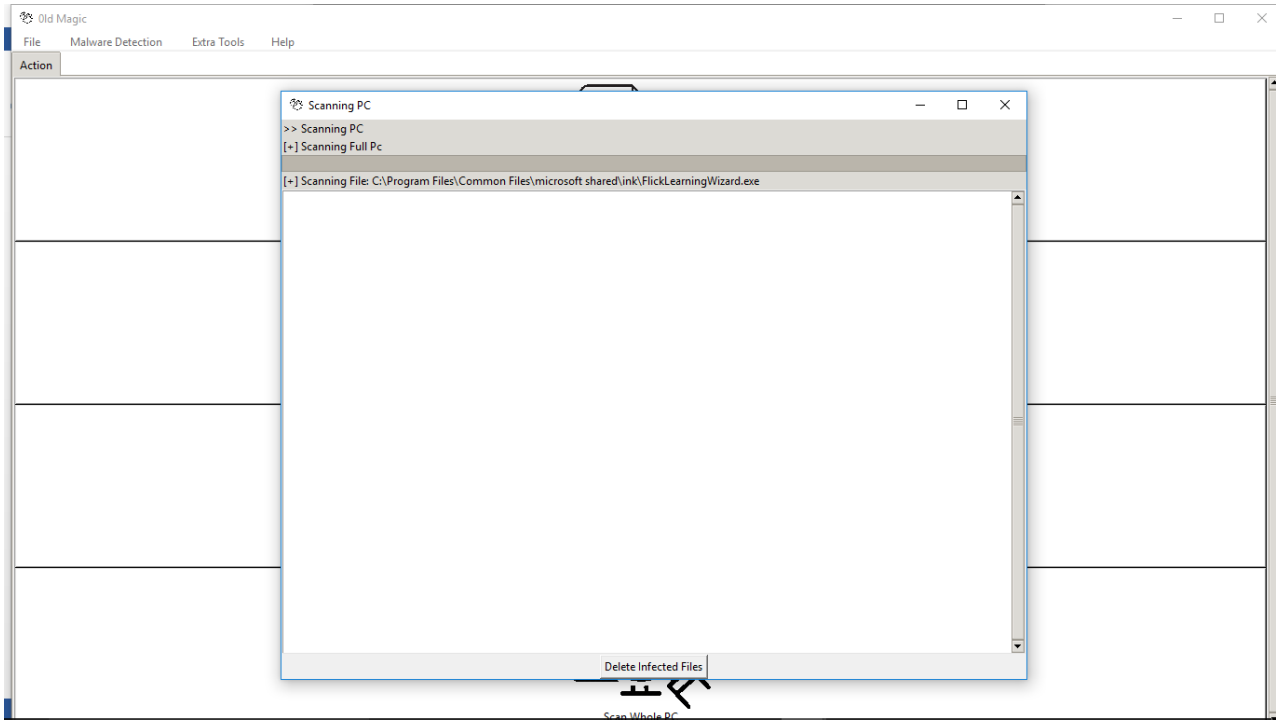


Figure 0.13: Application PC Scanning Page

After analysis, the detected malicious files are displayed in a result screen as shown in figure 7.14 and 7.15.

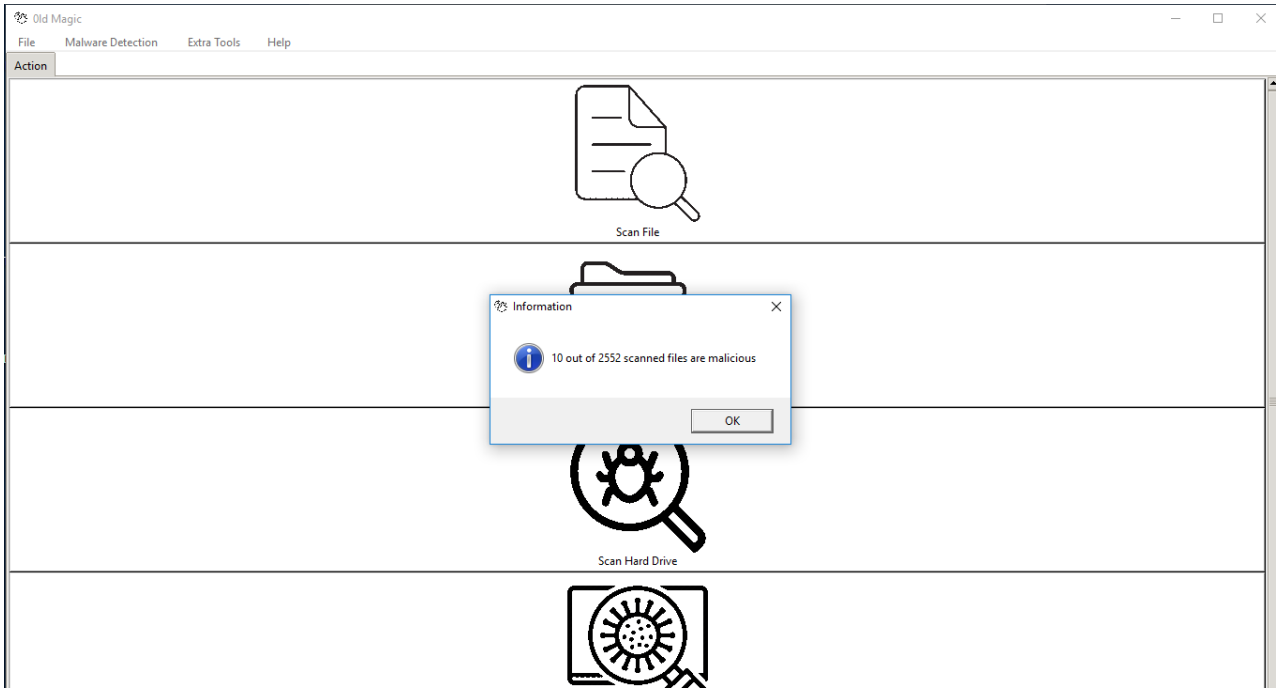


Figure 0.14: Whole PC Scan Output

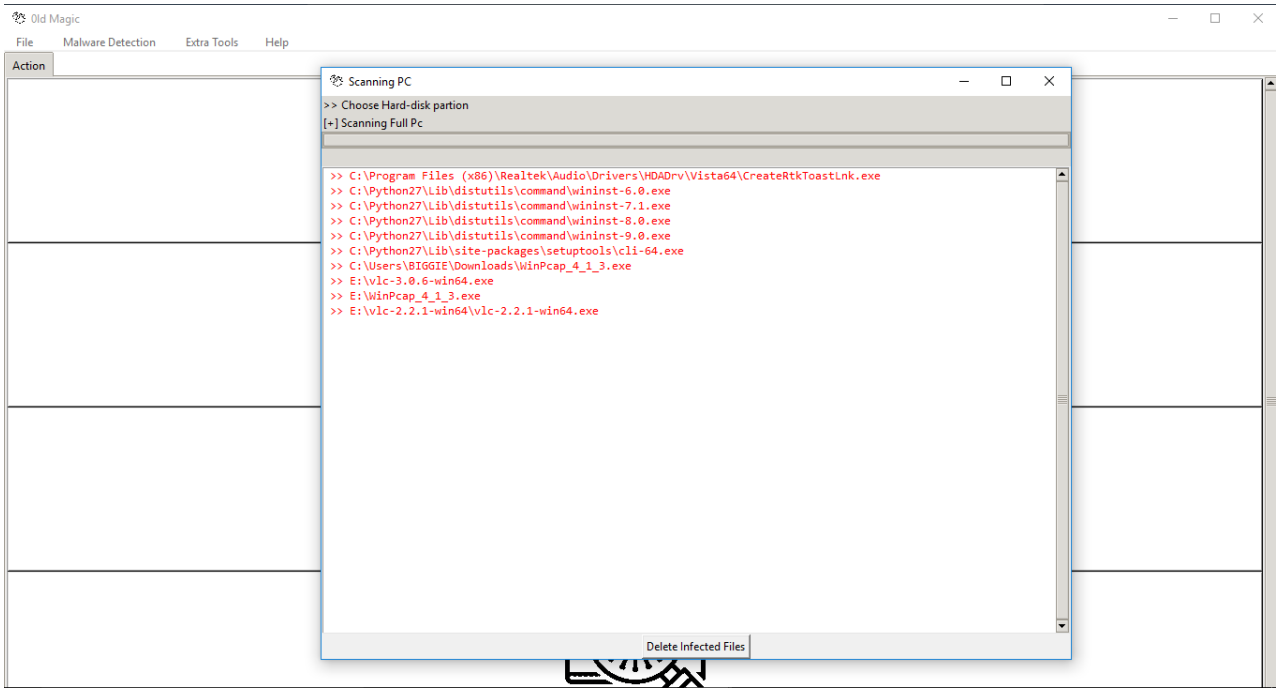


Figure 0.15: Whole PC Scan Results

7.2.1.5 Analysis on E01 Captured Memory Images

The tool is able to analyse .E01 captured memory image, and give a list of the files contained therein and also check for malware present. This is illustrated in figures 7.16, 7.17, 7.18 and 7.19.

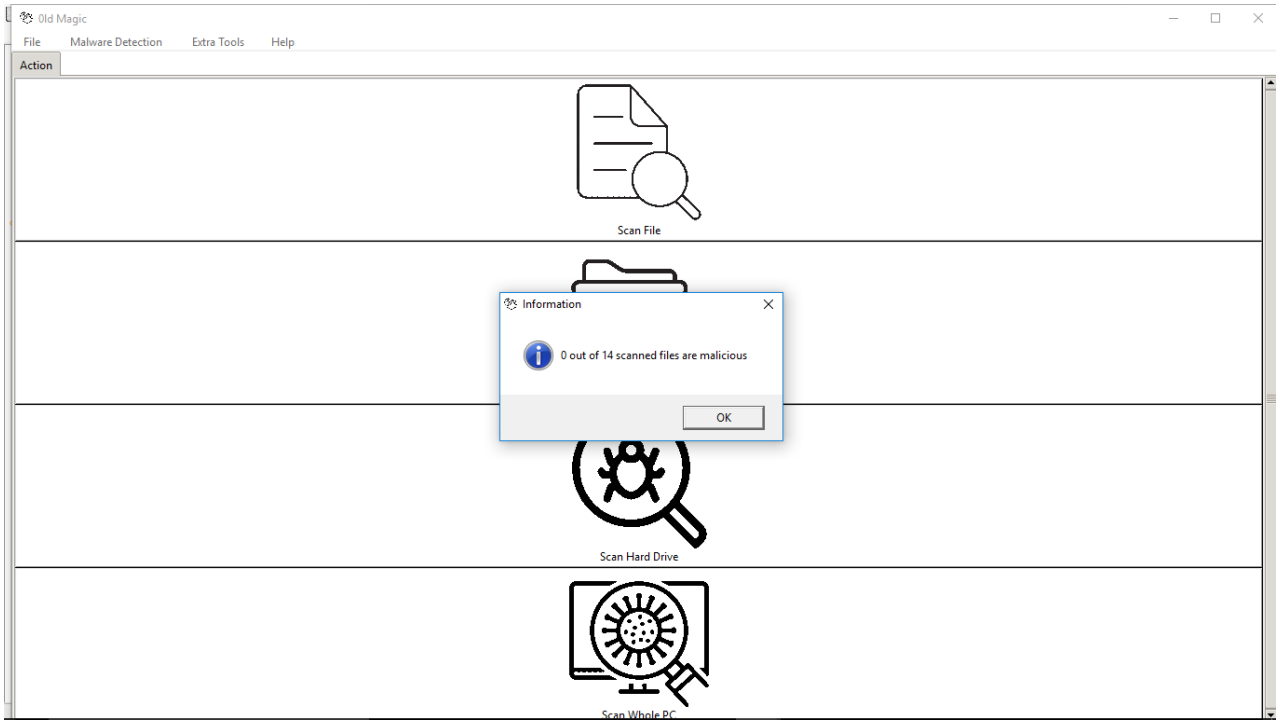


Figure 0.16: Analysis Performed on E01 File

Figure 7.18 shows analysis of files contained in the E01 file. These results can also be saved in .html report format.

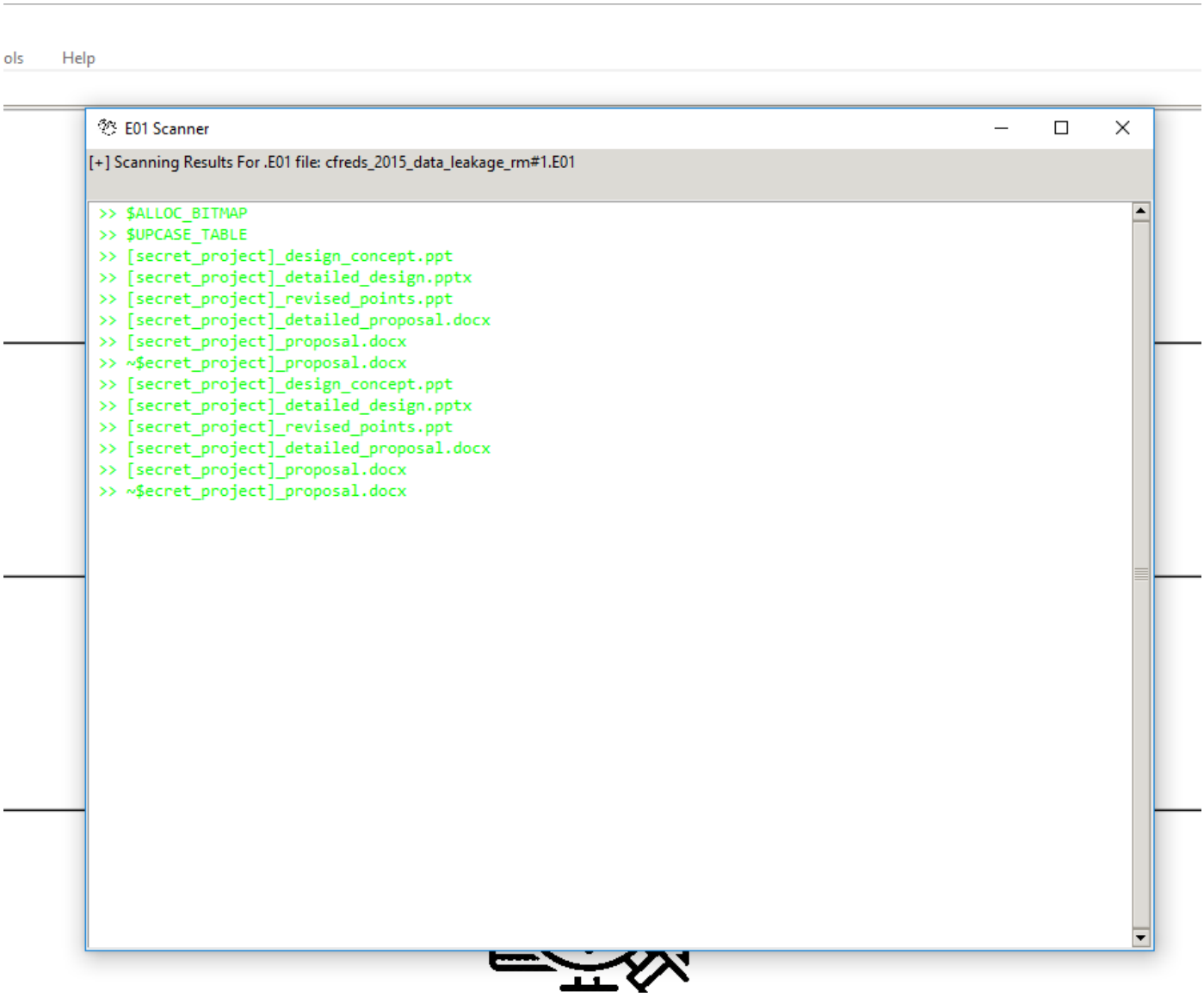


Figure 0.17: Results of the Scanned E01 File

Figures 7.19 and 7.20 shows analysis of individual process files contained in another E01 sample file.

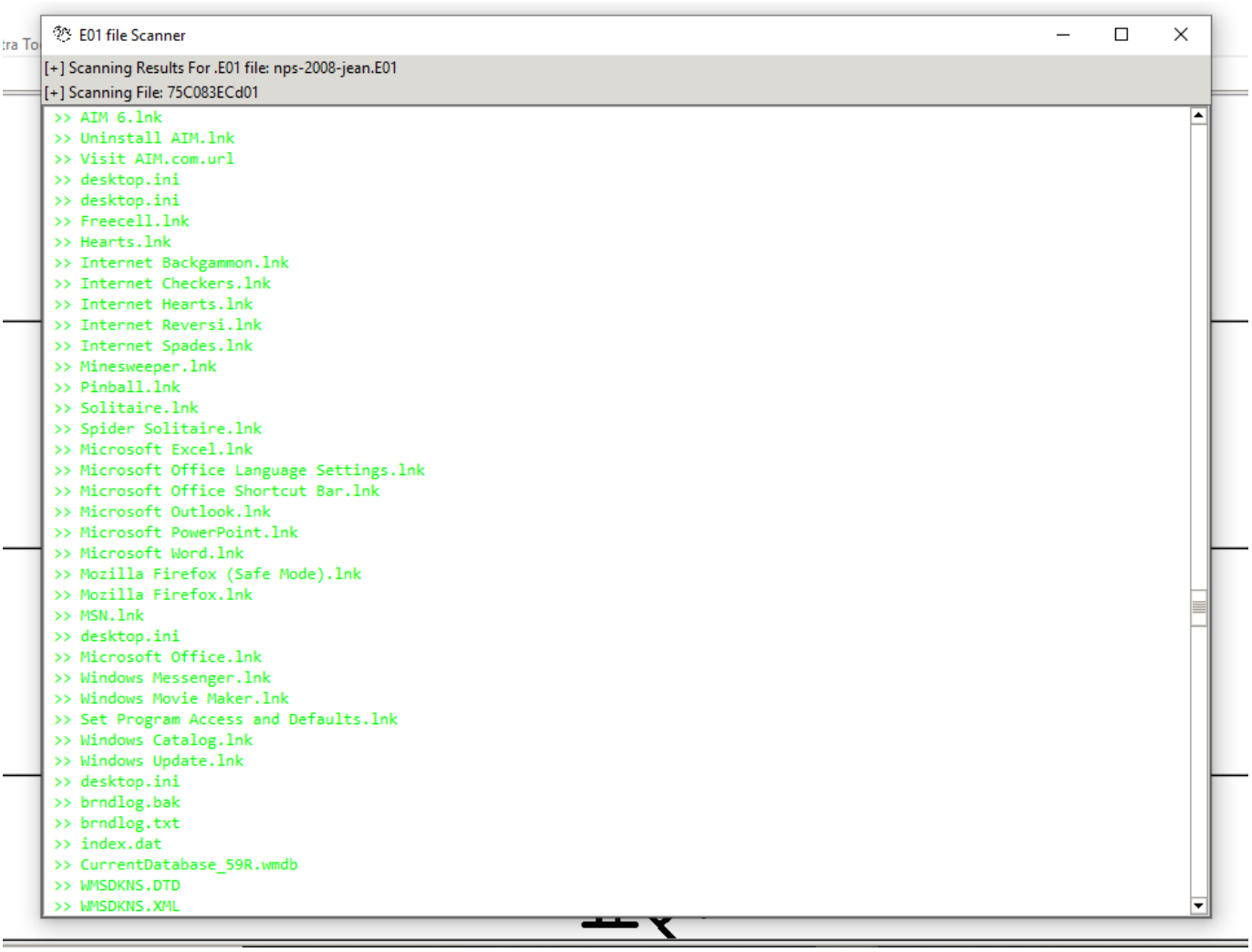


Figure 0.18: Result Analysis of Process Files in E01 Sample File

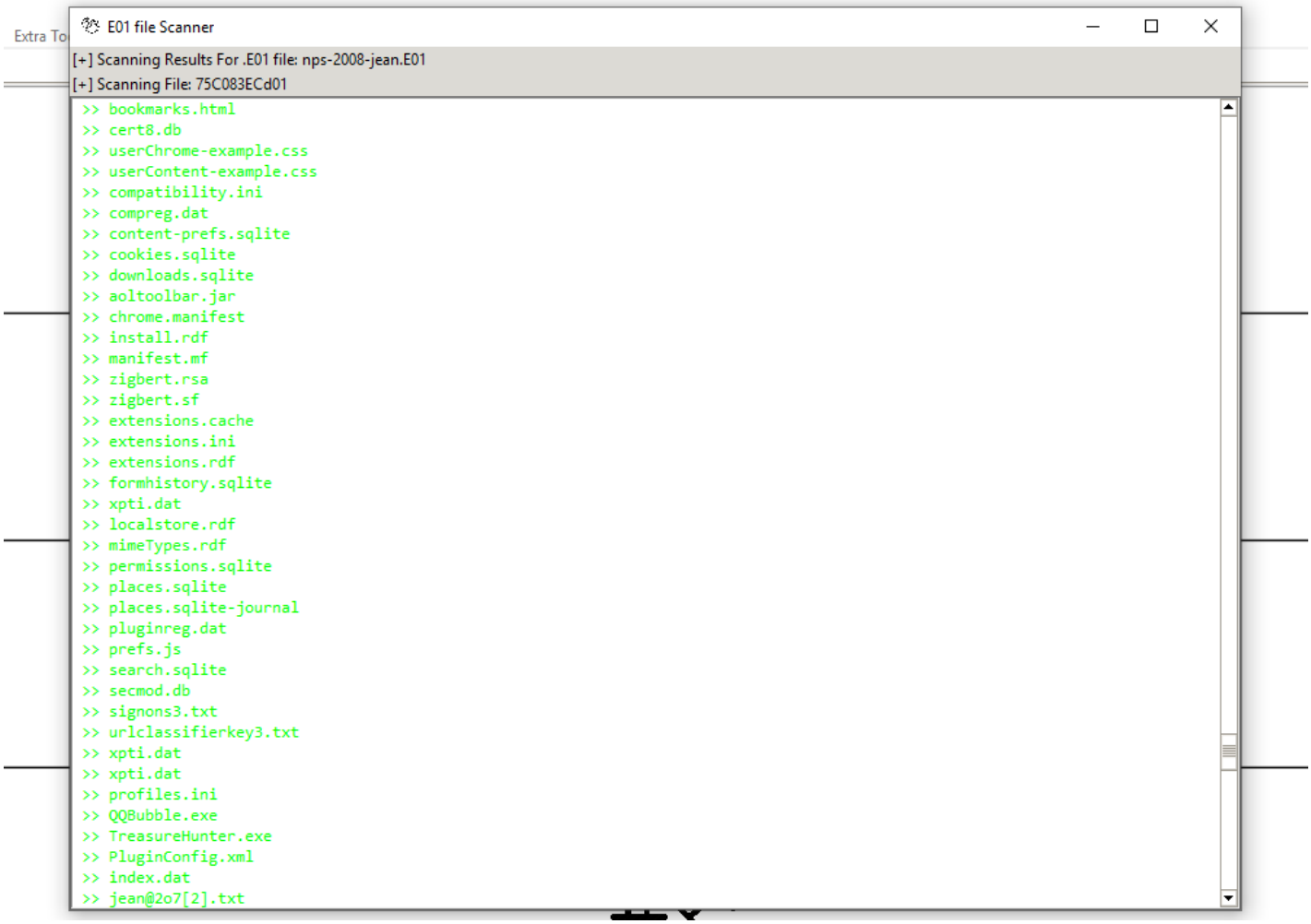


Figure 0.19: Result Analysis of Process Files in E01 Sample File

7.2.1.6 Analysis of Computer Processes

The tool is able to perform analysis on live memory. This enables the investigator to determine processes and their process files are running at time of analysis, registry files present at time of analysis, network activities and ports open, most input and output processes, processes utilising most CPU memory as well as processes consuming most CPU time. Figure 7.20 shows result screen of processes running at time of analysis.

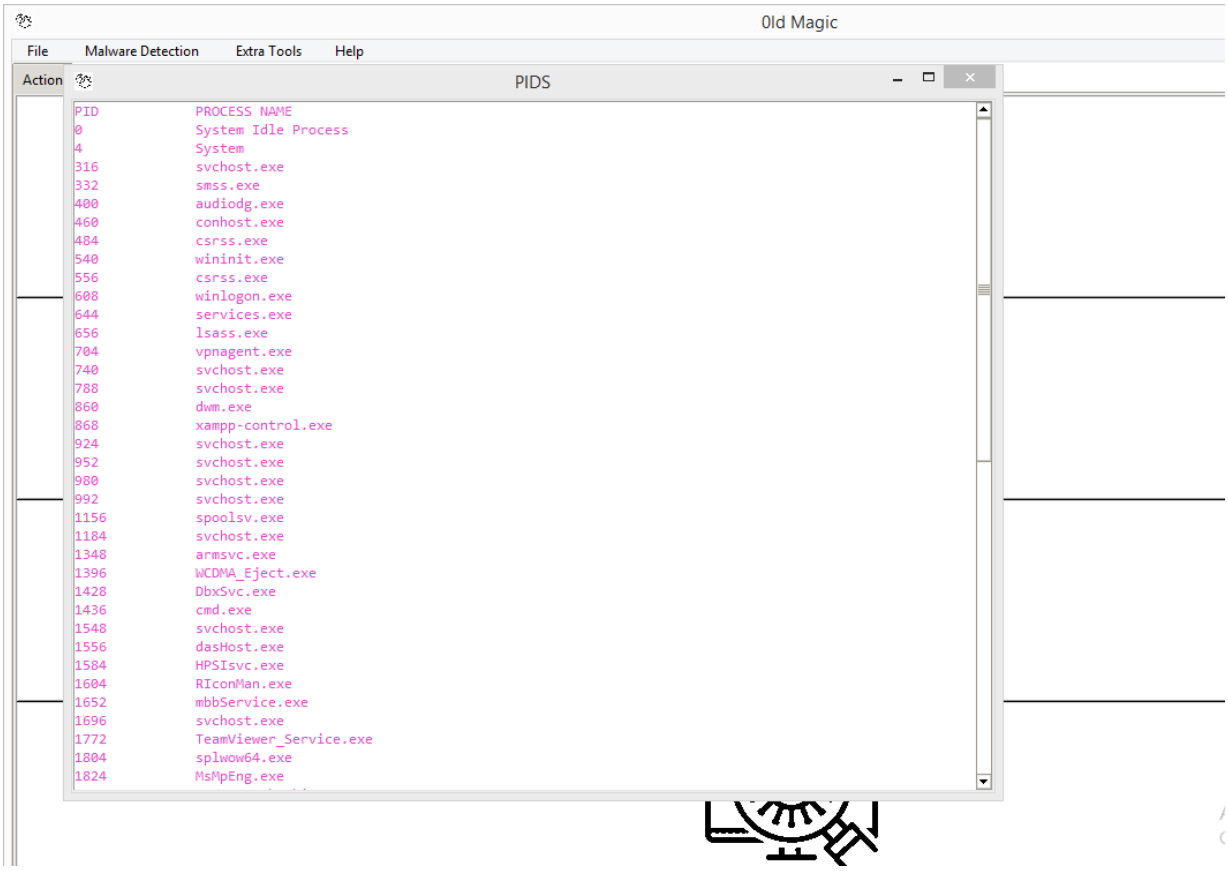


Figure 0.20: Processes and Their Process IDs Running at Time of Analysis

Figure 7.21 shows process files in use by processes running.

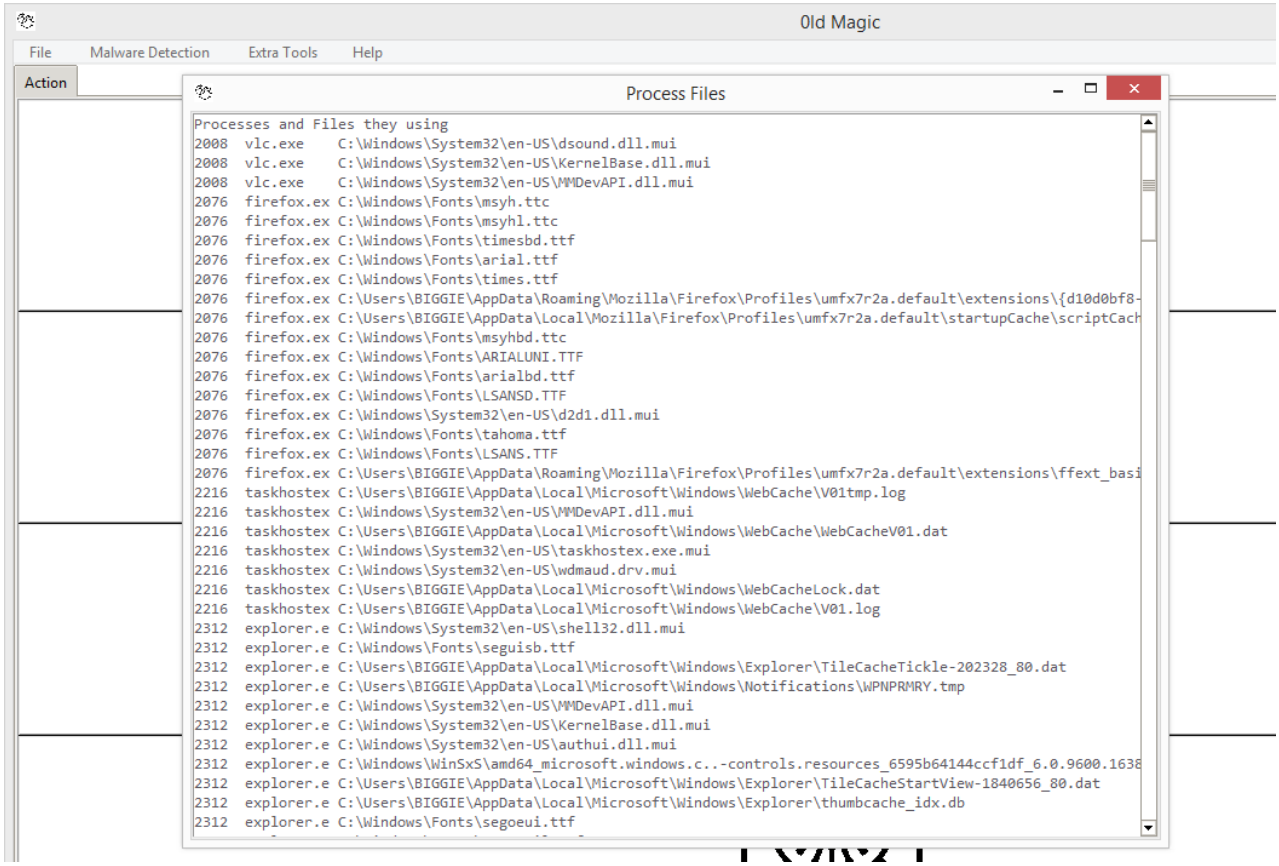


Figure 0.21: Process Files Running at Time of Analysis

Figure 7.22 shows registry files present at time of analysis.

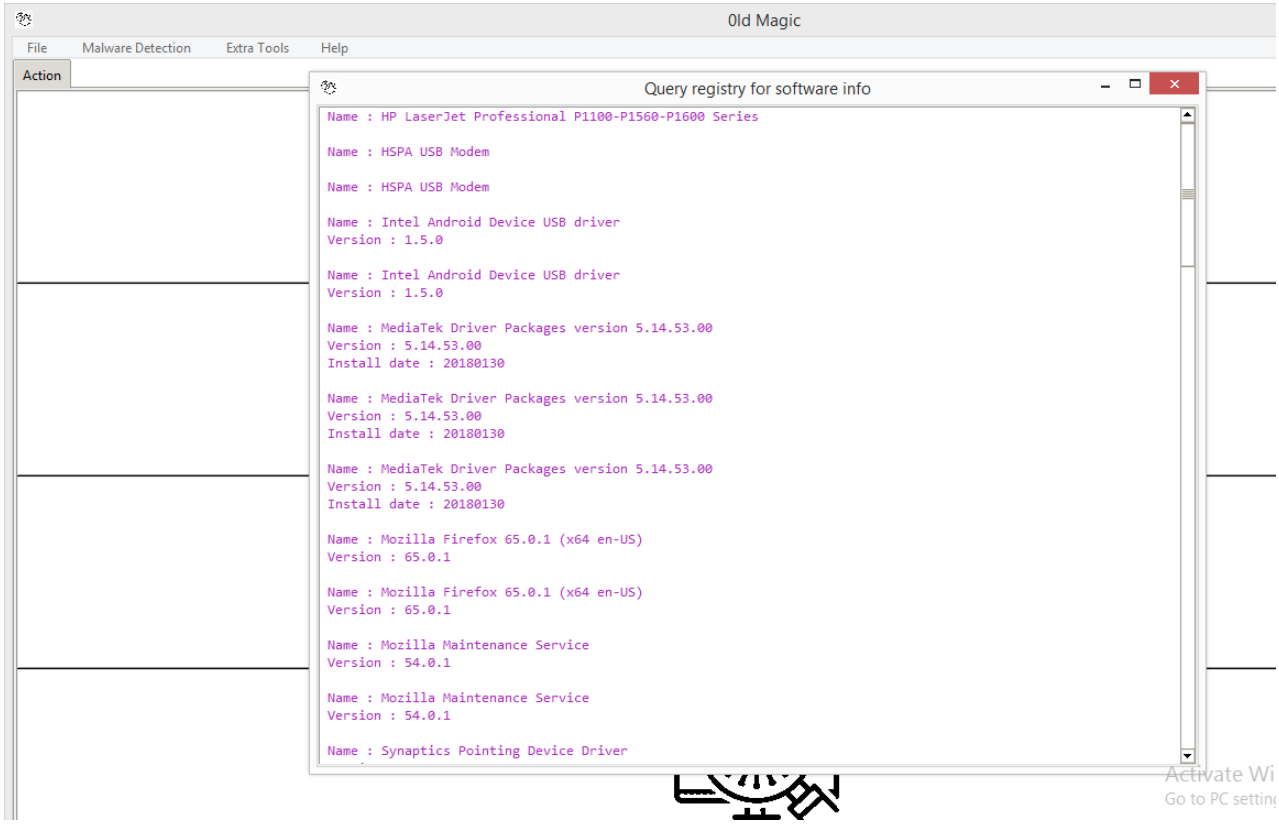


Figure 0.22: Registry Files Present at Time of Analysis

Figure 7.23 shows the network IP address, port number used by the network and the PID.

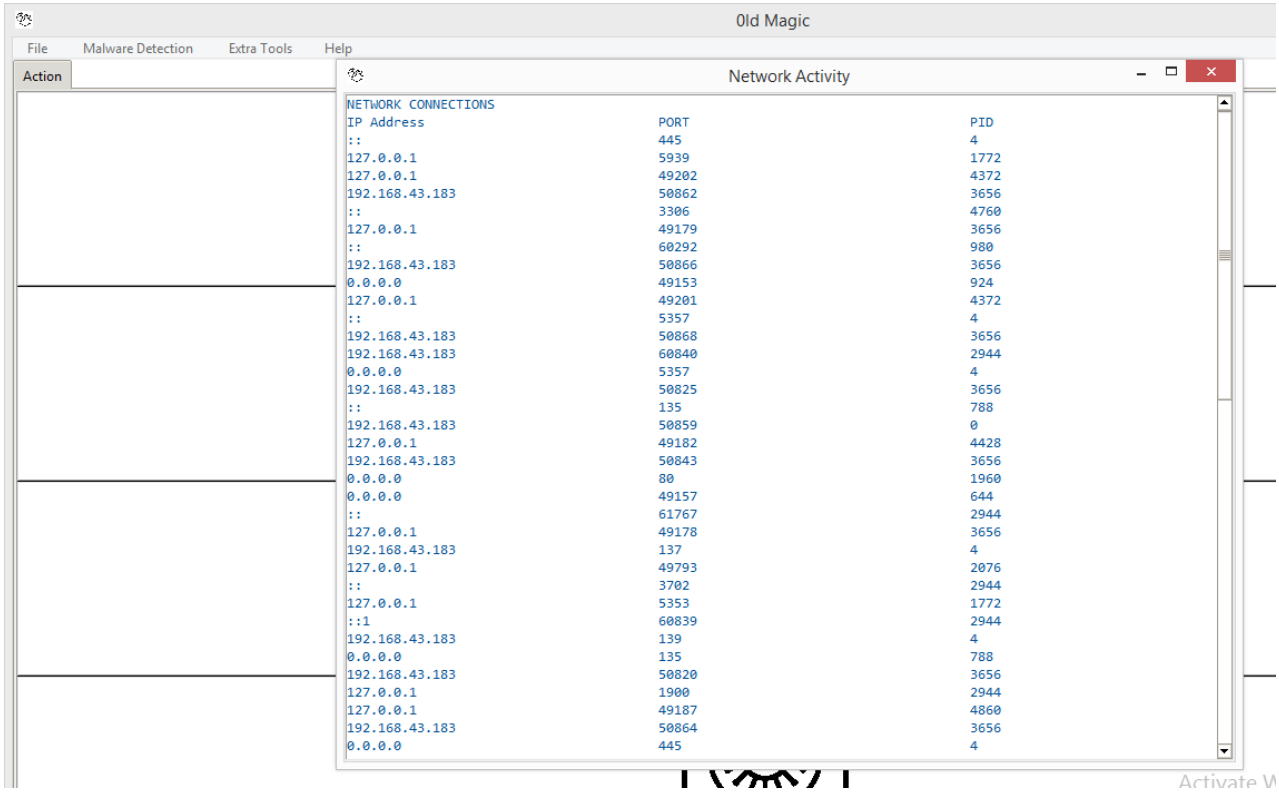


Figure 0.23: Network Activities at Time of Analysis

Figure 7.24 gives a list of processes with most Input Output at time of analysis.

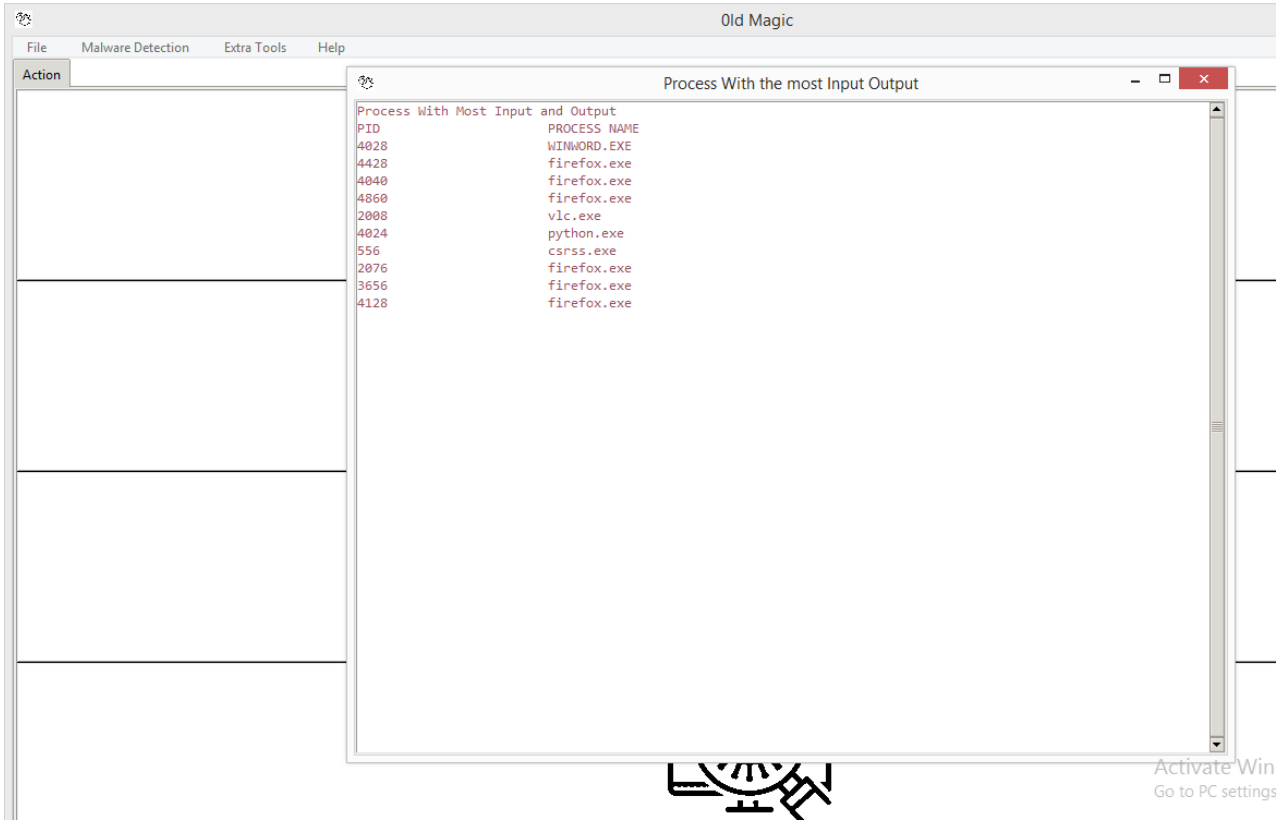


Figure 0.24: Processes with Most Input Output at Time of Analysis

Figure 7.25 shows processes using most CPU Memory.

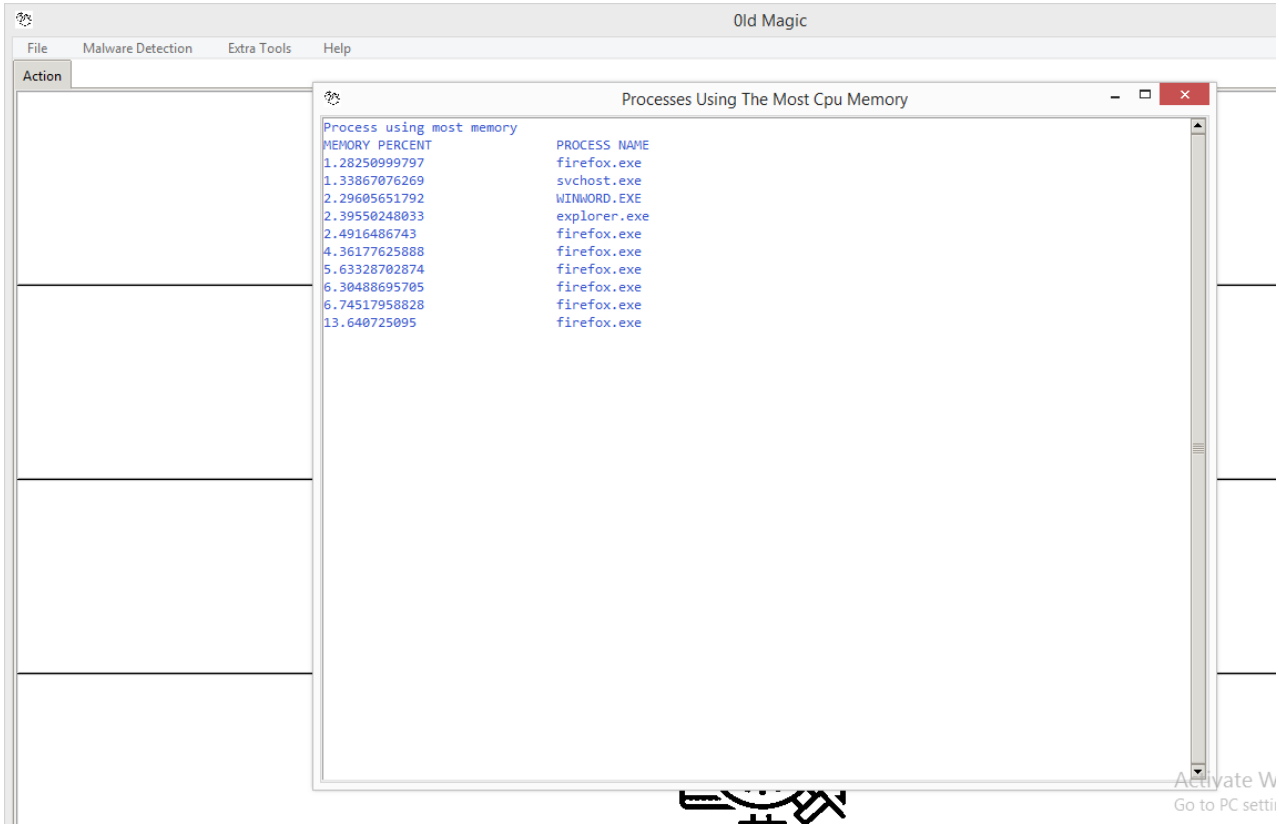


Figure 0.25: Processes Using Most Memory

Figure 7.26 shows analysis on processes with most CPU time.

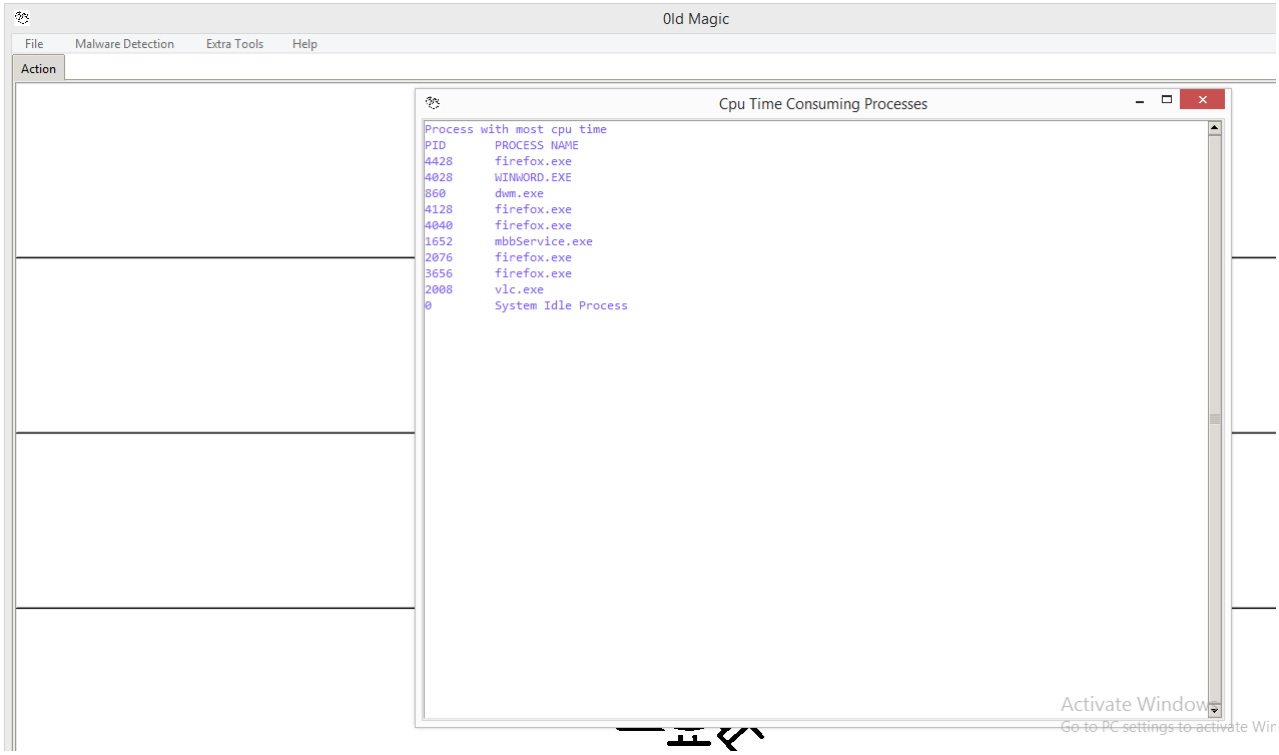


Figure 0.26: Processes with Most CPU Time

7.2.1.7 Reports

The tool generates reports in .html format. Even though other reports formats such as pdf and XML and excel can be used, most forensic tools generate reports in .html format for ease of uploading evidence to other forensic tools. This format was also used because of its compatibility with most forensic tools. Reports stored in .html can be retrieved in future for reference purposes. Figures 7.27, 7.28, 7.29, 7.30 and 7.31 show different types of reports generated by the tool based on various scans and analysis conducted.

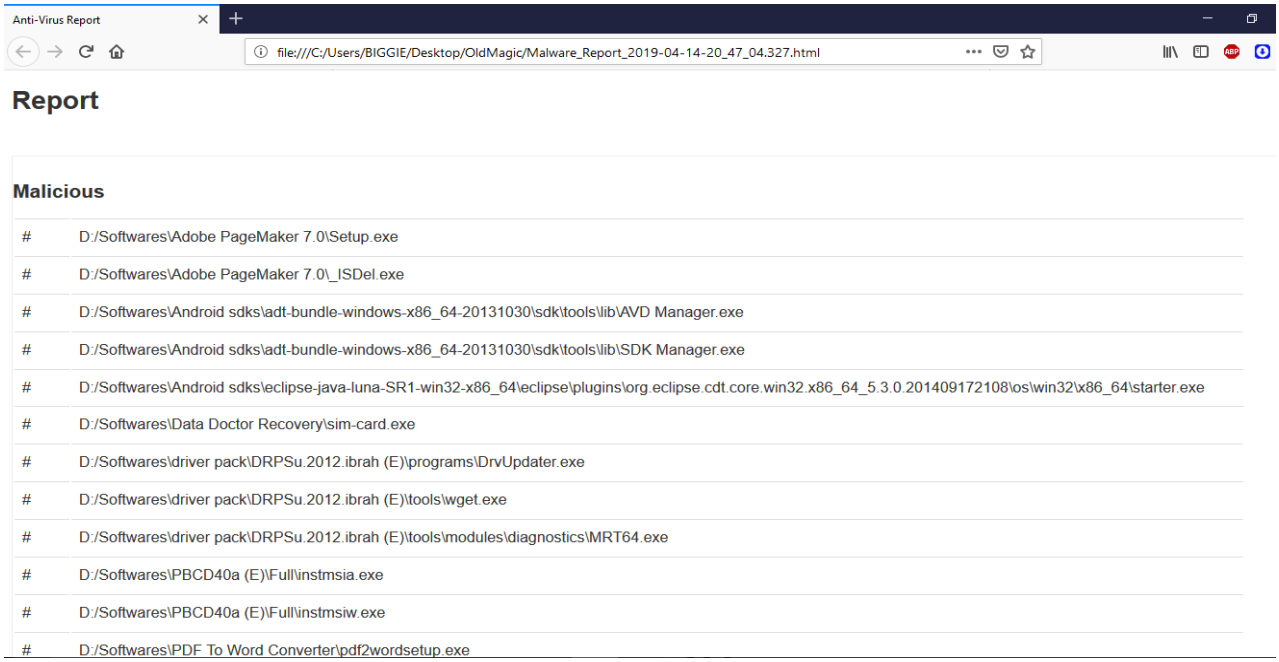


Figure 0.27: Malware Generated Report

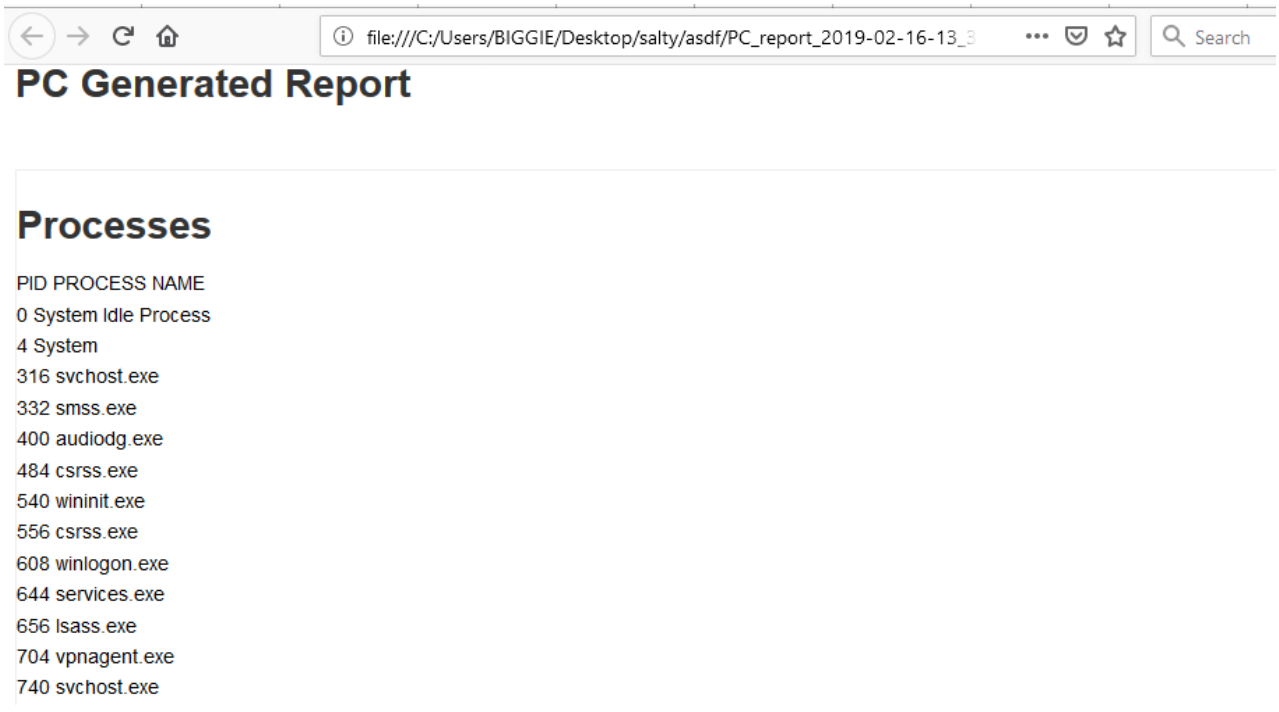


Figure 0.28: Processes Running Generated Report

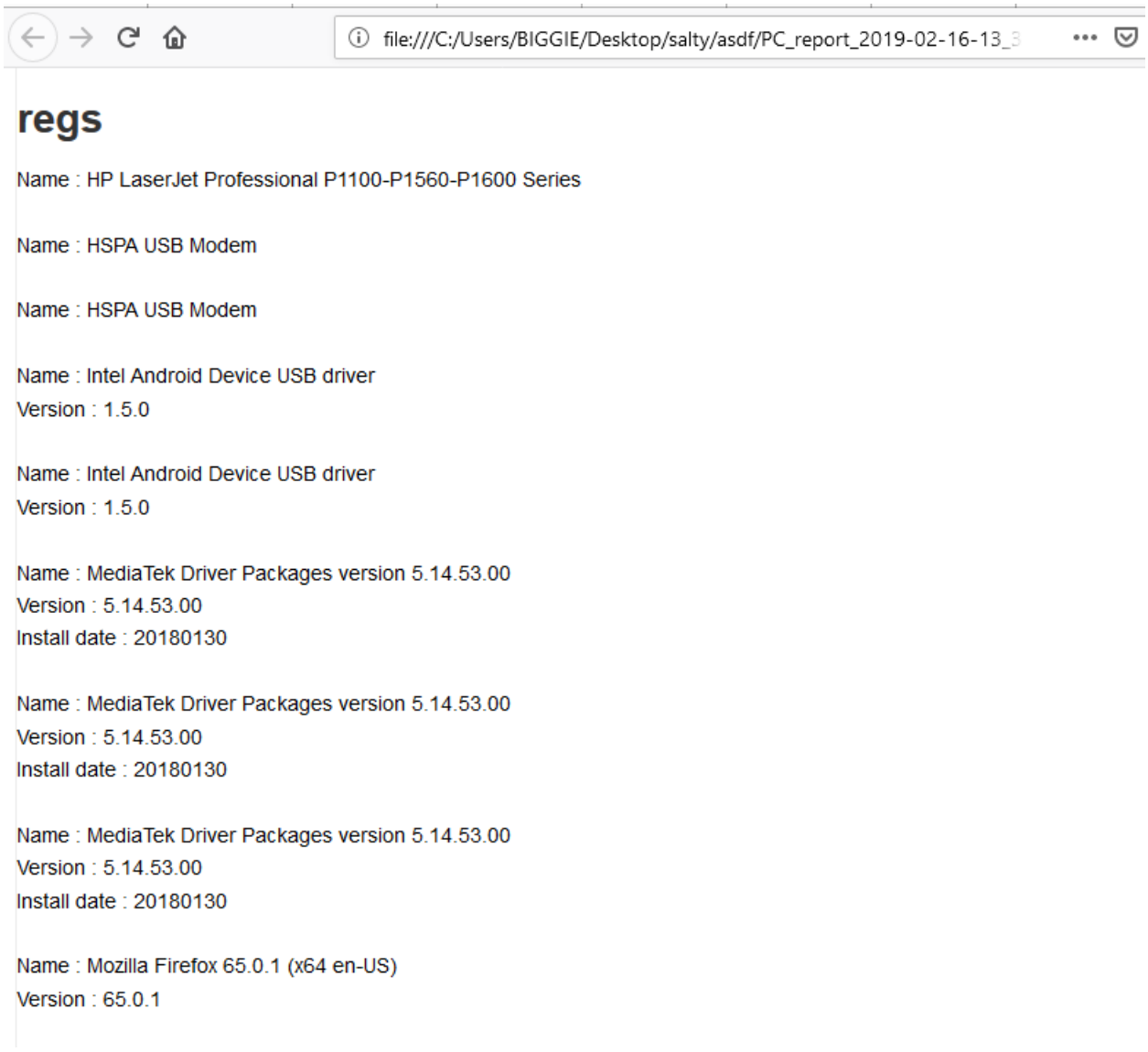


Figure 0.29: Registry Files Generated Report

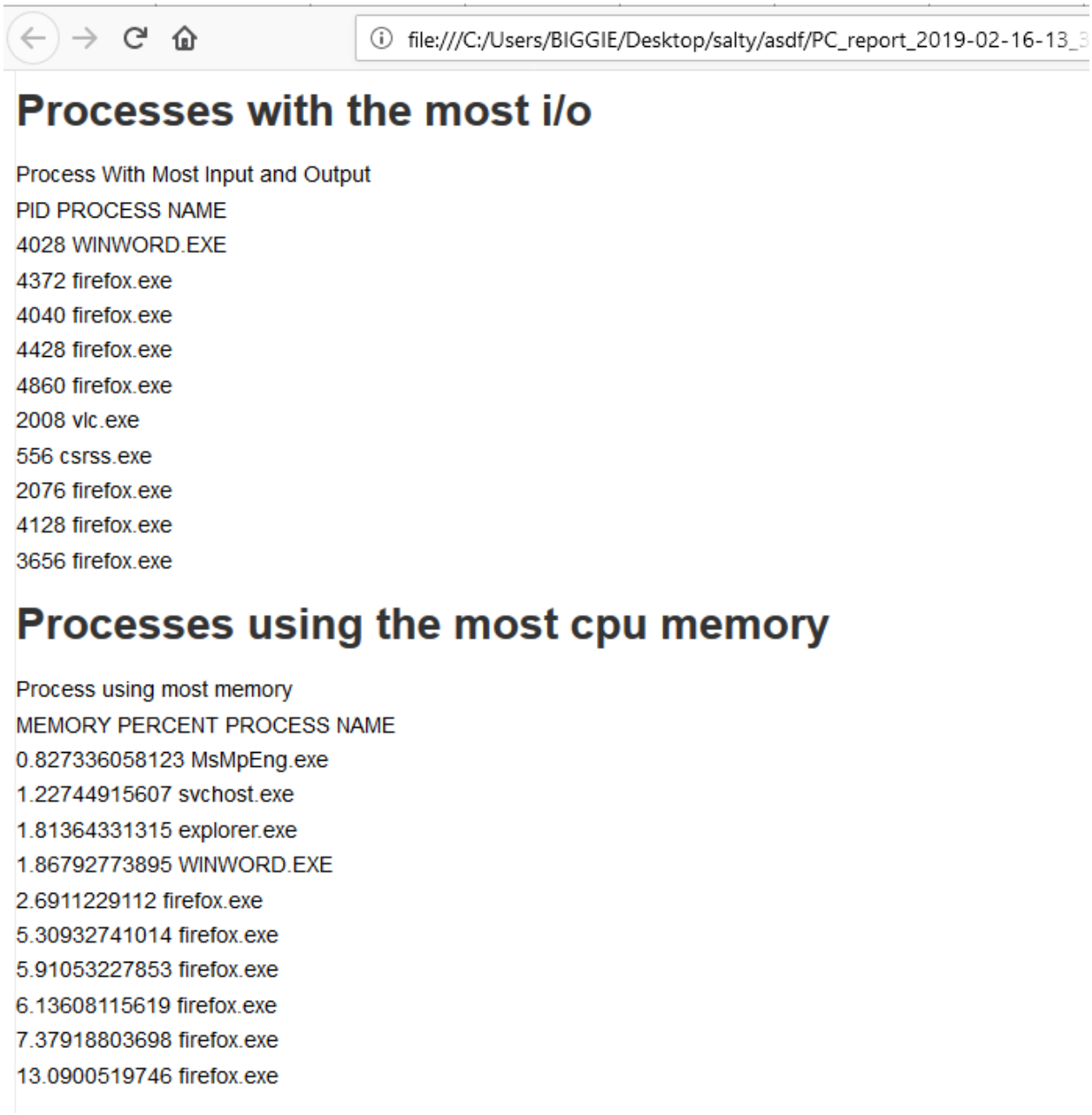


Figure 0.30: Processes with Most Input Output and CPU Time Generated Report

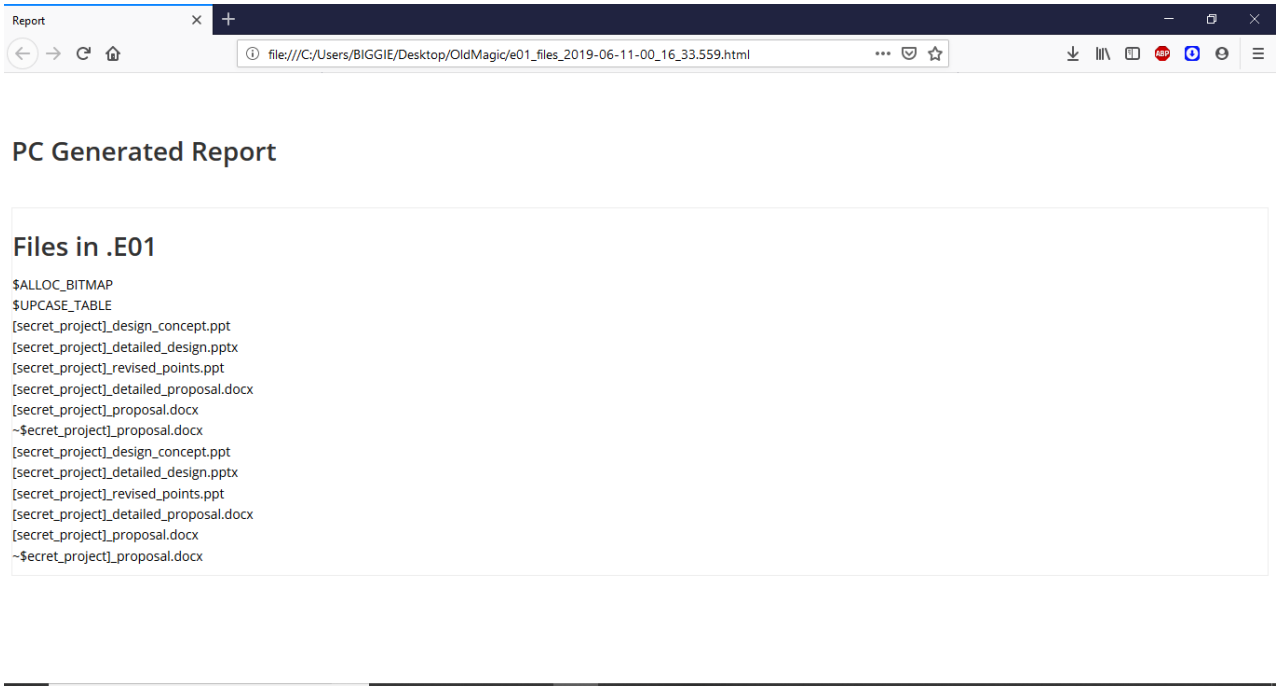


Figure 0.31: E01 Captured Memory Report

7.3 Model Validation

This is defined as the set of processes and activities intended to verify that models are performing as expected, in line with their design objectives, and business uses. Our main objective was to develop an automated malware detection, analysis and reporting tool. Our tool was able to achieve the above in that it has been able to analyse computer hard drive, captured memory and scan for malware present. Our tool was able to analyse live memory and determine crucial memory components such as processes running at time of analysis, files open, processes consuming most CPU among others. Lastly our tool was able to generate reports for analysis done. This was generated in .html format.

Key to the forensic process is results that are repeatable and quality evidence as part of validation. NIST's guidelines state that the test results of forensic software or tools should be repeatable and reproducible (Brunty, 2017). Testing was also done to ensure the developed tool was bug free. A bug is defined as an error or flaw in a computer software that causes the software to give an incorrect or unexpected result (SteelKiwi, 2015). Testing for bugs was done through conducting several tests repeatedly to determine the result given was consistent.

i. *Repeatability* refers to obtaining the same results when using the same method on identical test items in the same laboratory by the same operator using the same equipment within short intervals of time. The test was performed four times on the same folder and produced the same results.
ii. *Reproducibility* refers to obtaining the same results being obtained when using the same method on identical test items in different laboratories with different operators utilizing different equipment. The application was installed on another machine and peer reviewed using the same test case and it resulted in the same results. A whole PC scan was performed on

a different environment and was able to scan for malware as shown in figures 7.32, 7.33 and 7.34.

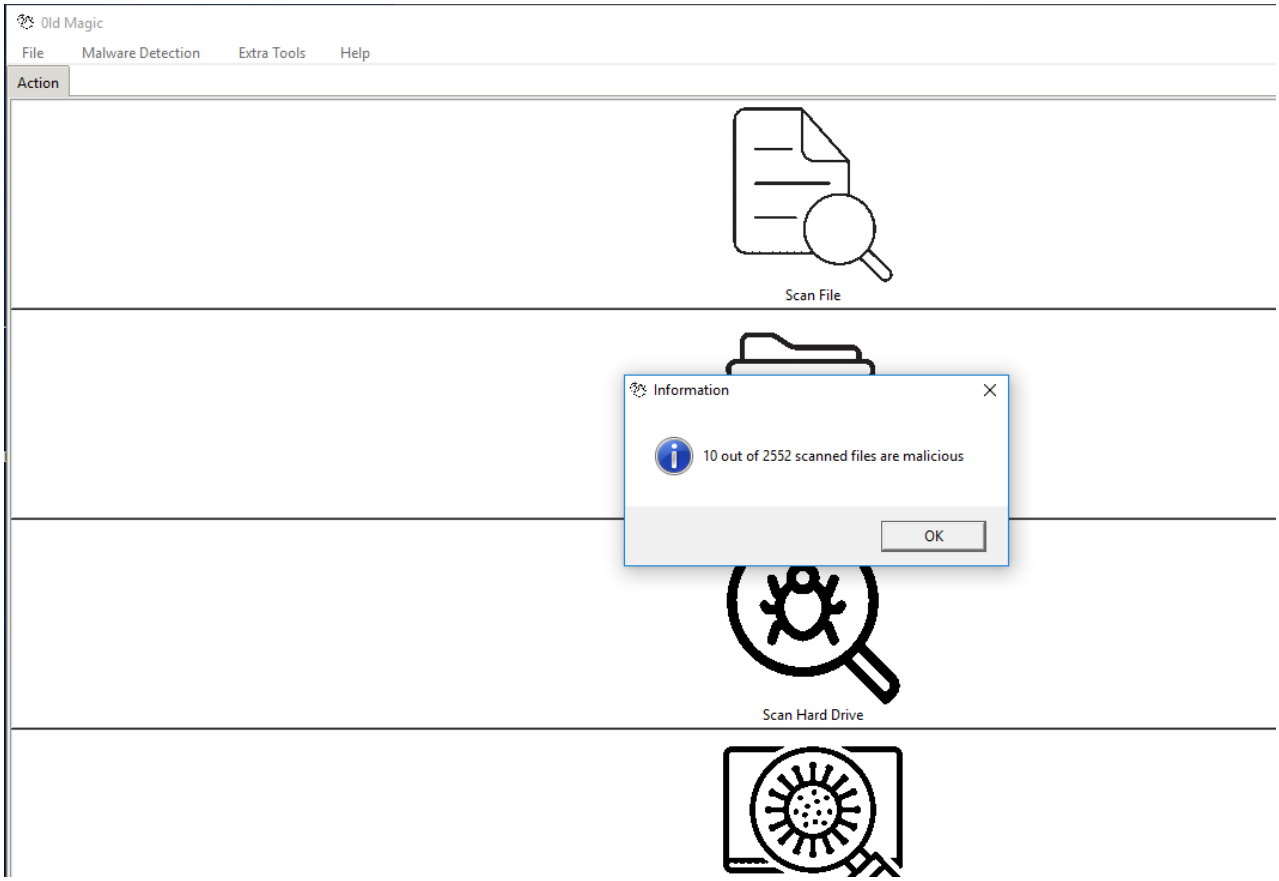


Figure 0.32: PC Scan on Different Environment

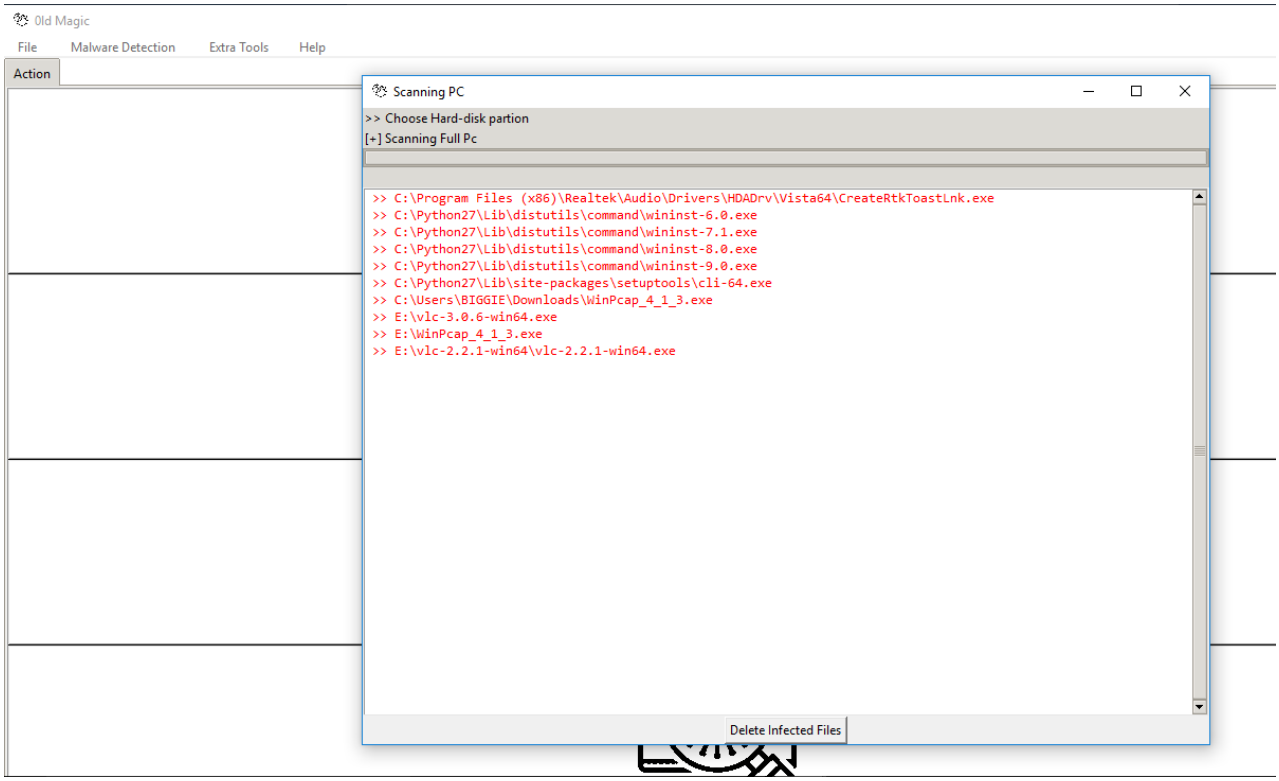


Figure 0.33: Result Scan on Different Environment

Report

Malicious

#	C:\Program Files (x86)\Realtek\Audio\Drivers\HDADrv\Vista64\CreateRtkToastLnk.exe
#	C:\Python27\Lib\distutils\command\wininst-6.0.exe
#	C:\Python27\Lib\distutils\command\wininst-7.1.exe
#	C:\Python27\Lib\distutils\command\wininst-8.0.exe
#	C:\Python27\Lib\distutils\command\wininst-9.0.exe
#	C:\Python27\Lib\site-packages\setuptools\cli-64.exe
#	C:\Users\BIGGIE\Downloads\WinPcap_4_1_3.exe
#	E:\vlc-3.0.6-win64.exe
#	E:\WinPcap_4_1_3.exe
#	E:\vlc-2.2.1-win64\vlc-2.2.1-win64.exe

Figure 0.34: HTML Generated Report on Scanned PC

The folder scanned using our tool was also scanned using Avast anti-virus and similarities were observed with regards to the malware identified as shown in figure 7.35 and 7.36. This was carried out to validate that our tool is correctly identifying malware recognised by other malware identification tools.

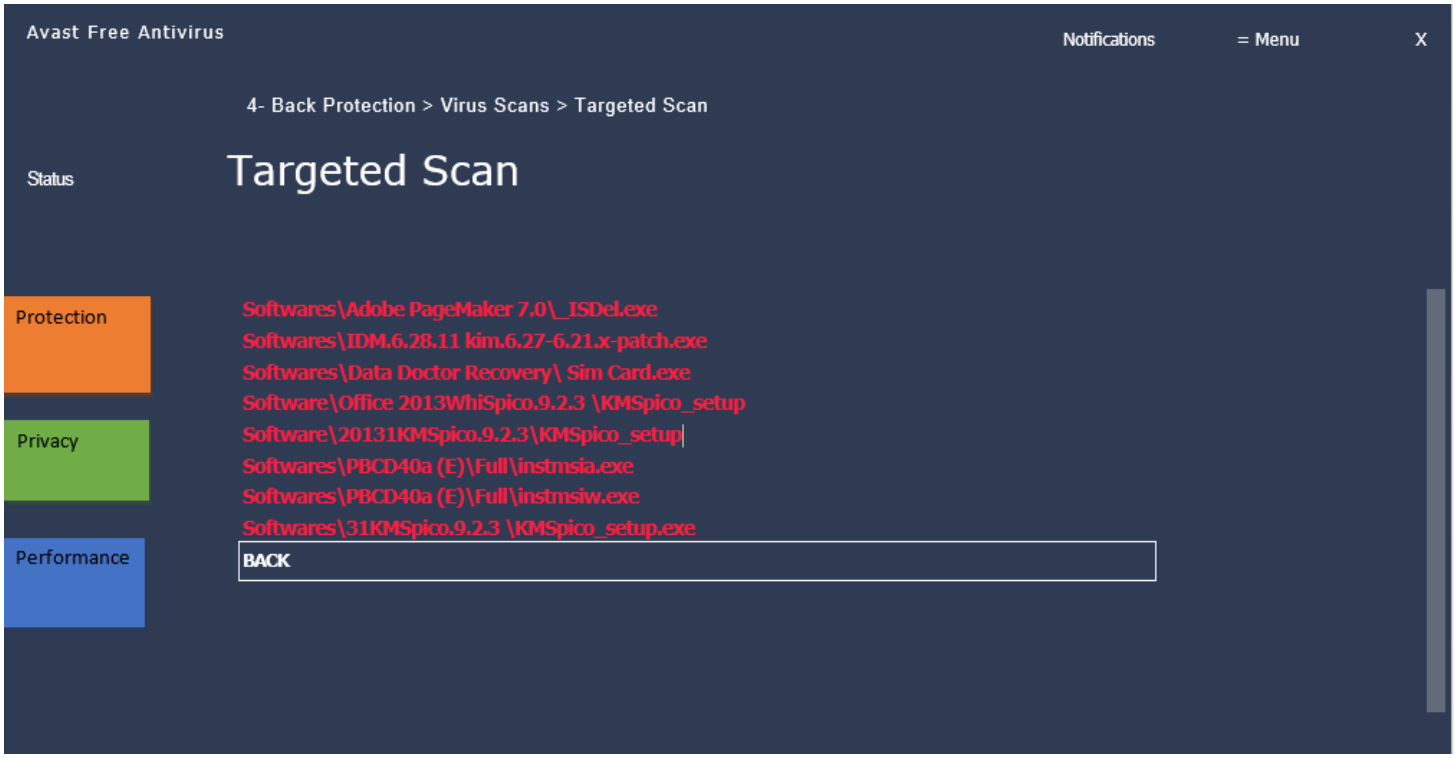


Figure 0.35: Sample Folder Scanned Using Avast Anti-virus

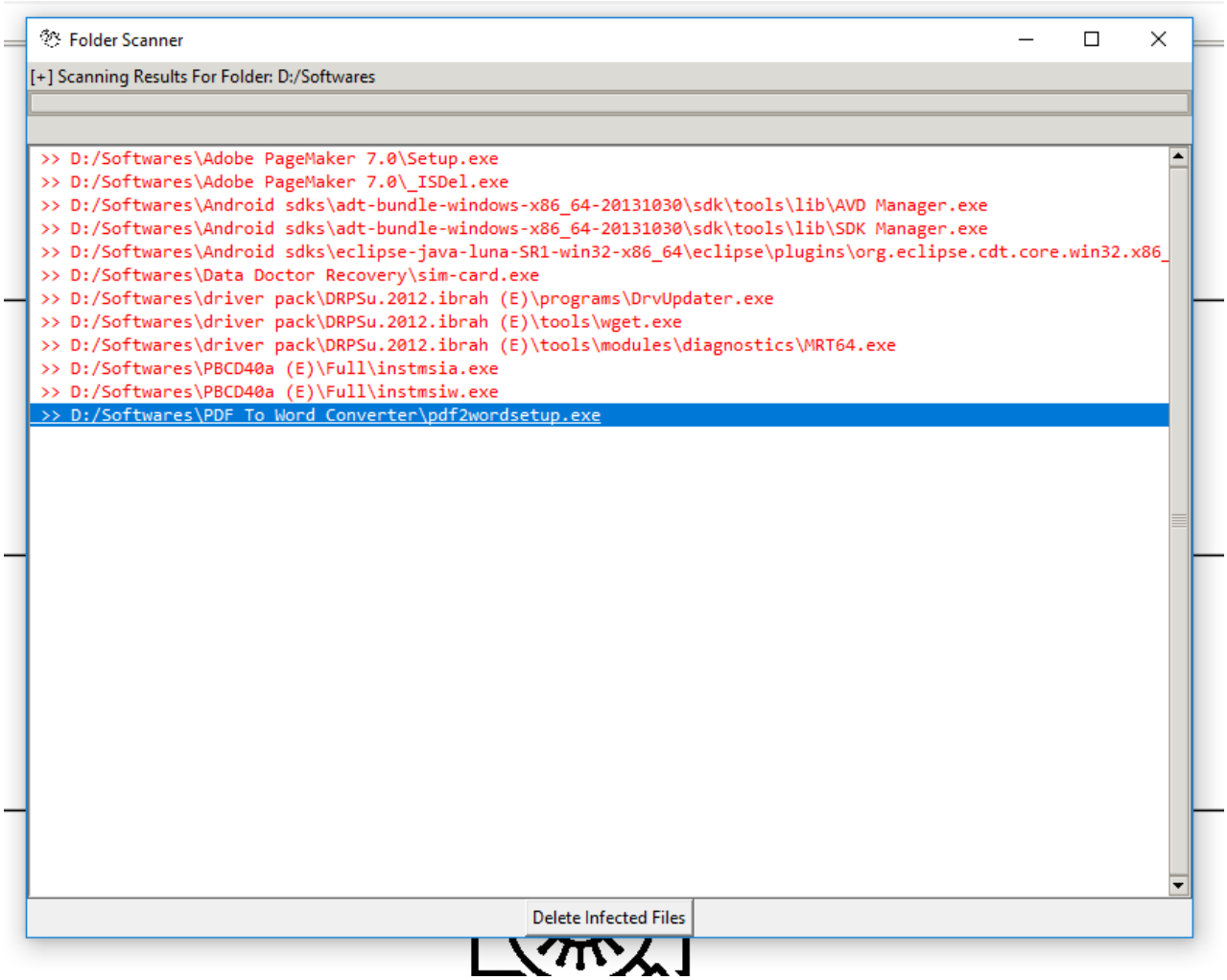


Figure 0.36: Sample Folder Scanned Using Developed Tool

7.4 User Feedback

User feedback is important in determining if the tool met the user requirements and if the tool solved user needs. This was conducted through a peer review and the feedback from users was:

1. The tool was able to have achieve its core functionalities.

2. Not a 100% match was achieved between our tool and Avast Anti-virus tool. This owed to the fact that Avast might not be utilizing the online repository (virusshare.com) as our tool.

Avast is a large scale commercial tool that utilises advanced and complex repositories for their hash databases compared to our database that is locally hosted and limited in storage.

3. The tool contained no bugs at the time of testing as the results achieved were the expected results. There is however, need to create a bug reporting mechanism for future cases of bugs in the tool.

4. The tool is user friendly in that the operations are straight forward and self-explanatory. The help section of the tool offers clear information on the tool features and what each feature does. Improvement needs to be done on the user interface of the tool to make it more appealing to users.

7.5 Model Verification

The problems identified and that informed our research were:

i. While there is some level of automation in forensic tools, some tools require the investigator to have knowledge of the Command Line Interface in order to do proper investigation.
ii. Most of the automated forensic tools are commercial and expensive making it difficult for financially deprived parties to efficiently make use of them to gather information.
iii. Most forensic malware detection tools either analyse live memory or captured memory images at a time.

In order to verify the tool solved the problems mentioned, the developed tool provided a user interface that is fit for both skilled and unskilled investigators. Through the user interface, forensic experts do not need expertise on the Command Line Interface. Secondly, our tool is freely available to all forensic investigators. Lastly, our tool was able to analyse live physical memory and captured memory.

Chapter 8: Discussion of Results

After analysis of the data collected from memory dump, the research achieved its desired objectives as listed in the first chapter of the dissertation.

i. We were able to analyse capabilities of the current memory-based malware analysis tools as discussed in chapter 0 of our research.

ii. Our tool was able to identify use cases pertaining to malware analysis as well as identify data to be analysed by our tool. Use cases included analysis on live memory from personal computer, as well as captured memory images in E01 format

iii. We were able to develop and test our tool through a set of controlled experiments. The tool was able to analyse files and detect malware as well as generate a report for the scanned files. The tool was also able to capture computer information at time of analysis. This included the processes running, their Process IDs, any network activity, CPU consuming processes as well as registry files running at time of analysis. The tool was also able to consume E01 files and analyse files contained therein and detect for malware. These tests and results are discussed in chapter 0.

iv. We were able to validate our tool using a set of controlled experiments. This was done through running the tool in a different environment and getting the same results. This was also done by comparing results from other malware detection tools and getting the same results as our tool gave. This has been highlighted in chapter 0.

The literature review was able to expound on the existing forensic tools, their working and limitations in detail. The review showed other tools that have the capability to analyse data same as our tools but with various limitations such as lack of free versions of the analysed tools, lack of ability to analyse both live and captured memories etc. This research through the tool has shown the ability

to retrieve crucial information from the memory dumps, both live and captured. The tool has also demonstrated the capability to convert dump variables such as .E01 dump files into readable format. The tool is also free for all forensic investigators.

Chapter 9: Conclusions, Recommendations and Future Work

9.1 Conclusions

Information Visualisation provides unique solutions for gaining better understanding of complex and large datasets (Heidi & Enrico, 2012). Information Visualisation techniques have resulted to innovative problem solving in several fields including business, computer science and medicine (Zhang, 2007). These techniques could offer a renewed approach for Digital forensic analysts dealing with the complexity of understanding enormous datasets.

Forensic investigations focused on the data contained in the disk storage can provide a unique overview of the device's state at the time of acquisition (Carvey, 2007). It was therefore paramount to develop a tool that would offer automated, detection and reporting for the memory. Our forensic investigations focused on the data contained in the disk storage as well as readily available captured memory images obtained from https://www.cfreds.nist.gov/data_leakage_case/data-leakage-case.html and https://digitalcorpora.org/corpora/scenarios/m57-jean. The tool was able to analyse both types of memories and was also able to give an overview of the processes running at time of acquisition. Our tool is also open source and free for use by forensic investigators as compared to the other forensic tools analysed in chapter 0 of this research. This gives a forensic investigator a complete tool that will analyse malware and running processes without having to worry about the pro version of the tool. In summary, our tool has been able to achieve the following:

i. Open source and free for all investigators.

ii. Upload captured memory images and scan for malware present.

iii. Scan hard drive for malware. This included scanning files, folders, hard disk partitions and whole drive.

iv.Analyse live memory for processes running at time of analysis.

v.Generate reports on data analysed

9.2 Recommendations

Recommendations for this project include the use of the developed tool for large scale analysis of memory. It is important for school curriculums to include courses on forensics investigations. A legal framework governing the utilisation of forensic tools for analysis can be worked on using joint efforts from law enforcement agencies, governments and financial institutions. Another critical aspect is the validation of forensic tools developed. Malware forensic tools are relatively new as this is a field that has emerged recently and more still needs to be done. Speedy validation by the various bodies would help in the investigation process. Evidence from the validated tools would be considered admissible in a court of law.

9.3 Future Work

In their literature, Beebe and Clark call for thorough improvements in the way forensic investigations are conducted (Beebe & Clark, 2008). They cite the increasing temporal aspect associated with evidence analysis. Data mining practices comprising content retrieval and predictive modeling are highlighted as possible developments to the digital forensic analysis process. Data mining uses techniques from several fields including machine learning, pattern recognition, artificial intelligence, and data visualisation (Freiling & Vomel, 2011) . More research ought to be conducted on effective ways of conducting and automating forensic analysis. This tool has offered an automated way of analysing and finding crucial information from memory dumps and E01 files. Extensibility of the tool should be explored to integrate or plug it to other tools for easier and faster analysis, as well as consume or analyse other images including .dd, and raw dumps of memory. More report formats

ought to be explored to accommodate report generation in pdf and excel formats. As part of future work, the tool should also allow for large scale analysis of E01 files so that it will be able to analyse more complex files and mine information such as processes that were running at time of acquisition of the dump, a feature we were not able to achieve in the stated time frame. The tool should also have plugin ability from other forensic tools, a feature we were also unable to achieve due to time factor limitations. Lastly, the tool user interface will be improved in subsequent versions in an effort to make the tool appealing to users.

Chapter 10 : References

Aljedi, A., Lindskog, D., Zavarsky, P., Ruhl, R., & Almari, F. (2011). Comparative analysis of Volatile memory forensics: Live response vs. memory imaging, in Privacy, Security, Risk and Trust. *IEEE Third International Conference on Social Computing.*

Bayer, U., Moser, A., & Kirda, E. (2006). Dynamic Analysis of Malicious Code. *Journal in Computer Virology*, 67-77.

Beebe, N., & Clark, J. (2008). A Hierarchical, Objectives-based Framework for the Digital Investigations Process.

Borba, P., & Cavalcanti, A. (2007). *Testing Techniques in Software Engineering.* Brazil: PSSE.

Brezinski, D., & Killalea, T. (2002). *Guidelines for Evidence Collection and Archiving (Best Current Practice).*

Brunty. (2017). *Validation of Forensic Tools and Software: A Quick Guide for the Digital Forensic Examiner.*

Buede, D. M. (2009). *The engineering design of systems: Models and methods.* NJ: John Wiley & Sons Inc. Retrieved January 2, 2018

Carvey, H. (2007). *Windows Forensics Analysis.* Syngress.

Cavalcanti, K., & Borba, J. (2010). *Summer School on Software Engineering.*

Computer Aided Software Engineering. (2016)

Creswell, J. (2013). *Research Design: Qualitative, Quantitative and Mixed Methods Approaches.* Chicago: SAGE Publications.

Data, B. (2012). *Addresing Big Data Security |Challenges: The Right Tools for Smart Protection* .

Davies, R. (2012). Introducing Agile Techniques to Teams Outside Software Development.

Digital Forensic Research Workshop . (2008). Memory forensics Challenge.

Egele, Manuel, Theodoor, Engin, & Kruegel. (2012). *A Survey on Automated Dynamic Malware-analysis Techniques and Tools.*

FireEye. (2013). *The need for Speed: Incident Response Survey.*

Freiling, F., & Vomel, S. (2011). A survey of main memory acquisition and analysis techniques for the windows operating system. In *Digital Investigations 8* (pp. 3-22).

Gandotra , E., Bansal, D., & Sofat , S. (2016). Zero-day malware detection. *Sixth International Symposium* (pp. 171-175). Embedded Computing and System Design (ISED).

Gandotra, E., Bansal, D., & Sofat, S. (2016). Zero-day malware detection. *Embedded Computing and System Design (ISED).*

Garret, & James, J. (2010). *The Elements of User Experience: User-Centered Design for the Web and Beyond.* New Riders Press.

Garfinkel, S. (2014). Digital Forensics Research: The Next 10 Years. *Digital Forensics Research Conference.*

Heidi, L., & Enrico, B. (2012). Empirical Studies in Information Visualization: Seven Scenarios.

ISO/IEC/IEEE. (2015). Systems and Software Engineering - System Life Cycle Processes. Geneva, Switzerland: International Organization for Standardization (ISO)/International

Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288.

Jones, J., & Etzkorn, L. (2016). Analysis of digital forensics live system acquisition methods to achieve optimal evidence preservation.

Kruse, W., & Heiser, J. (2002). Computer forensics incident response essentials. Indianapolis: Addison-Wesley, Pearson Education.

Kumara, A., & Jaidhar, C. D. (2017). *Automated multi-Level malwaredetection system based on reconstructed semanticview of executables using machine learning techniques at VMM, .* Future Computer Systems.

Ligh, M. H., Case, A., Levy, J., & Walters, A. (2014). *The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory.* Wiley Publishing.

McAfee. (2013). Infographic: The State of Malware.

National Institute of Justice. (2012). Forensic examination of digital evidence: A guide for law enforcement.

Netmarketshare. (2017). *Bioinformatics Web Development*. Retrieved from cellbiol.com: http://www.cellbiol.com/bioinformatics_web_development/chapter-1-internet-networks-and-tcp-ip/the-tcpip-family-of-internet-protocols/

Neufeld, D. (2010). Understanding Cybercrime. *EEE Hawaii International Conference on System Sciences*, (pp. 1-10). Koloa, Kauai.

NIST. (2016). Forensics Conference.

Osborne, G., & Turnbull, B. (2012). Development of InfoVis software for Digital Forensics. *Computer Software and Applications Conference Workshop*, (pp. 213-217). Izmir, Turkey.

Pendergas, G. (2010). Getting started in digital forensics. Do you have what it takes? Retrieved 2017, from http://computer-forensics.sans.org/blog/2010/08/20/getting-started-digital-forensics-what-takes/

Schatz, B. (2007). *Bodysnatcher*. Retrieved February 18, 2017, from Towards reliable volatile memory acquisition by software,: http://www.sciencedirect.com/science/article/pii/S1742287607000497

Shreshtha, G., & Chhikara, R. (2016). *Memory Forensics: Tools and Techniques.* Haryana, India.

Software, G. (2017). *Guidance Software-Encase Forensics*. Retrieved from Guidance Software: https://www.guidancesoftware.com/encase-forensic

Vinod, P., Jaipur , R., Laxmi, V., & Gaur, M. (2009). Survey on Malware Detection Methods. *Workshop on Computer and Internet Security*, (pp. 74-79).

Wikipedia.org. (2012). *System Design*. Retrieved December 12, 2017, from https://en.wikipedia.org/wiki/Systems_design

Zhang, J. (2007). *Visualization for Information Retrieval.* Wisconsin: Springer Science & Business Media.

APPENDICES

APPPENDIX A: Turnitin Report