NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

FIELD LEVEL COMPUTER EXPLOITATION PACKAGE

by

Adrian Arvizo
Vincent Janowiak

March 2007

Thesis Advisor:	Chris Eagle
Second Reader:	George Dinolt

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 2007	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE Field Level Computer Exploitation Package	5. FUNDING NUMBERS
6. AUTHOR(S) Adrian Arvizo and Vincent Janowiak	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT (maximum 200 words)

On today's battlefield whether in Afghanistan or Iraq, ground combat forces are dealing with an adversary that has embraced the use of computers and electronic devices. Until now, there was no package of consolidated forensic tools available to the ground combat forces with the capability of conducting a quick interrogation of these devices. After a unit has captured a target that possesses electronic devices that require immediate exploitation, the devices are transferred to higher authority. Valuable time is lost locating and capturing associates of the target as the information is sent away to higher authority for analysis. The product of this thesis, "Interrogator," was designed to prevent or reduce the time lost by allowing anyone to quickly retrieve data that is stored on a computer. This capability will positively aid a small unit commanders' ability to exploit critical vulnerabilities of the enemy in a timely manner and improve the survivability of the unit and the ability to complete their mission.

14. SUBJECT TERMS Computer forensics, Graphical User Interface, Linux	15. NUMBER OF PAGES 127
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

FIELD LEVEL COMPUTER EXPLOITATION PACKAGE

Adrian E. Arvizo
Lieutenant, United States Navy
B.S., University of Arizona, 2001

Vincent J. Janowiak
Lieutenant Commander, United States Navy
B.S., Norfolk State University, 2002

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 2007

Authors: Adrian Arvizo

Vincent Janowiak

Approved by: Chris Eagle
Thesis Advisor

George Dinolt
Second Reader/Co-Advisor

Peter J. Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

On today's battlefield, whether in Afghanistan or Iraq, ground combat forces are dealing with an adversary that has embraced the use of computers and electronic devices. Until now, there was no package of consolidated forensic tools available to the ground combat forces with the capability of conducting a quick interrogation of these devices. After a unit has captured a target that possesses electronic devices that require immediate exploitation, the devices are transferred to higher authority. Valuable time is lost locating and capturing associates of the target as the information is sent away to higher authority for analysis. The product of this thesis, "Interrogator," was designed to prevent or reduce the time lost by allowing anyone to quickly retrieve data that is stored on a computer. This capability will positively aid a small unit commanders' ability to exploit critical vulnerabilities of the enemy in a timely manner and improve the survivability of the unit and the ability to complete their mission.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGEMENTS

The authors would like to acknowledge the financial support of SPAWAR, Code 2875, for allowing the purchase of the equipment used in this thesis. This work was performed under Contract N6600106WR00101. We would also like to thank Brian White from SPAWAR as our mentor. Finally, we would like to thank our thesis advisors Chris Eagle and Dr. George Dinolt for giving us the opportunity to work on this thesis, taking the time for its review and providing the guidance and constructive feedback necessary to make it a worthwhile learning experience.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. THE WAR ON TERRORISM

Photographs and documents about Iraqi training camps that existed during Saddam Hussein's regime were part of a very large collection of items that were captured in postwar Iraq and Afghanistan [Ref. 1]. The items include handwritten notes, typed documents, audiotapes, videotapes, compact discs, floppy discs, and computer hard drives. The data retrieved from digital devices together with photographs and other items gave United States intelligence officials an inside look at the activities of Hussein's regime in the months before the Iraqi War. Intelligence officials who have worked on document exploitation tell us there were roughly two million "exploitable items" captured in postwar Afghanistan and postwar Iraq. Of that number, they say, some 50,000 have been fully exploited [Ref. 1].

These two million exploitable items are no longer in theater; they have been removed and distributed to other organizations where more analysis resources are available. There are very limited methods, tools and personnel available to conduct this kind of exploitation in the field. This is mostly because the current tools that exist for this work are very difficult to use. There are a few experts in the field who are capable of using the tools to conduct the exploitation, but the number of these experts is not nearly large enough to accomplish the amount of useful work that could be done.

As the exploitation takes place within other organizations outside of the battlefield, the data is not

analyzed or consolidated and disseminated in a timely fashion for battlefield use. Why should the warfighter in the field not have the ability to tap into this wealth of knowledge on a real-time basis?

It may be necessary to piece together several documents, images and voice recordings in order to arrive at the conclusion that terrorists are being trained and the location of said training. This does not mean that interrogating an individual piece of electronic equipment in the field is a waste of time; it's certainly possible that doing so would yield information that might immediately be useful. We have developed a tool that we call the "Interrogator" to assist in the field evaluation of captured computer devices for intelligence information.

B. USAGE SCENARIO

The following scenario presents a real world operation that demonstrates how Interrogator could have been utilized to assist the ground combat forces. This scenario has been sanitized for public release.

1. Background

Unit A has received numerous tips from locals about someone who we will call "Bad Man." This individual has been implicated in a number of attacks against local and coalition forces. Multiple source intelligence sources indicate that Bad Man has links with known terrorist cells throughout the region and is constantly on the move. Conventional wisdom states that Bad Man serves as a facilitator for anti-coalition forces by smuggling various items to finance acts of intimidation and murder. However,

Unit A does not know the location of Bad Man and has no physical evidence to support its reports about his actions.

2. Mission Planning

One day, Unit A gets a Human Intelligence (HUMINT) report that details the location of Bad Man within their area of operations (AO). The commander decides that he has the proper resources to detain Bad Man issues an order directing his immediate capture. During mission preparation, the raid force commander, with staff support, starts mission planning that can be generalized with the acronym SMEAC: Situation, Mission, Execution, Administration and Logistics, Command and Signal. The raid force commander includes a post raid sensitive sight exploitation (SSE), which is the evidence collection, handling and detainee processing part of actions on the objective. The results of the SSE will help Unit A determine if the reporting on Bad Man is accurate.

3. Operations

Unit A conducts the raid and captures Bad Man. During the SSE, a couple of computers and communication devices are discovered at his location. Unit A collects all the evidence and ships it off to higher authority where it is registered and eventually undergoes forensic analysis.

4. Post Operations

Weeks later, during the data mining of Bad Man’s hard drives, a document is found that lists several well known terrorists; some who are in custody and others who are still operating throughout the country. Bad Man’s points

of contact show their contact number, email information, previous missions, and current location. The analysis organization quickly realizes the value of this find and puts the information in a report and immediately sends it back to the headquarters for the unit that initially obtained the equipment. On a positive note, the discovered information proves sufficient enough to have a local judge detain Bad Man until he can stand trial. On the other hand, as unit commanders get the report on Bad Man's contact, they direct raids to round up all of Bad Man's associates only to find out that they are too late. Thus, terrorist cells that had any contact with Bad Man or his associates change their tactics, techniques and procedures (TTPs) to continue future operations with little to no interference. Coalition forces must start all over again in trying to understand and neutralize the adversarial force in their AO.

5. Conclusion

The capability for data reduction that Interrogator provides would positively aid small unit commanders in their ability to improve the cycle time for the collection and application of actionable intelligence[1], thus enabling them to exploit critical vulnerabilities of the enemy more rapidly. Force multipliers are needed on today's battlefield where intelligence is time sensitive and lives are on the line daily. In its current form, Interrogator does not have the capability to work with foreign

[1] Actionable Intelligence (AI) is defined as the means of providing a commander and his warriors a level of situational understanding, delivered with speed, accuracy and timeliness, to conduct successful operations.

languages. This is a challenge that could be overcome then Interrogator could be pushed to small unit commanders. The goal of Interrogator is to help commanders improve the survivability of their unit, which means saving lives and accomplishing their mission of neutralizing or destroying the enemy more efficiently.

C. PURPOSE OF THE STUDY

The purpose of the work documented in this thesis was to develop a user-friendly tool capable of extracting data from a computer that might be useful as near real-time intelligence to the warfighter in the field. By accomplishing this, one of our goals was to support ground combat forces operating as described in the scenario above.

Some examples of the types of data that could serve as real time intelligence are files that contain mission information or details about locations of associates, contact information, web browser history, images and videos. Gathering this information from a single enemy computer may not yield all of the enemy's secrets, but it might lead to the next terrorist stronghold or even reveal a phone number that can be used to track down the enemy. With that in mind, it was our goal to provide the ability to retrieve files containing data useful for the warfighter in the field.

There are many software and hardware tools that are capable of extracting this data but none provide an integrated environment from which to collect all types of information with one "tool" at a time. Many of these tools

require a detailed theoretical and working knowledge of operating systems and command line interfaces, commonly referred as CLIs.

A command line interface is nothing more than a tool that an operator uses to communicate with a computer. This sounds like a simple process but the reality is that there are rules that must be followed that apply to the syntax for a CLI. Unfortunately, the syntax and the rules change and are dictated by either the operating system (for example, MS-DOS or UNIX) or by embedded systems (i.e., Juniper Networks or Cisco Systems) [Ref. 2]. In addition to knowing the syntax and the rules, the operator must also know the different options that can be applied to the commands.

Here are some examples of the format typically used for a CLI:

[doSomething] [how] [toFiles]
or
[doSomething] [how] [sourceFile] [destinationFile]
or
[doSomething] [how] < [inputFile] > [outputFile]

Here is an example taken directly from Appendix C of this thesis:

```
find $1 -regextype posix-extended -iregex "$2" -print |
\sed -e 's/.*/"&"/' | xargs egrep -a -i -l "$3"
```

This is the Linux command sequence to search a directory for files with specified keywords. The syntax for each command is described in its respective “Manual Page” [Refs. 3, 4, 5, 6]. Every set of characters is a

different argument that must be applied in order to complete the search. To be able to construct commands like this efficiently, the operator must have a reference available to look up the details of each of the commands and the arguments that they might use or have many years of Linux forensic experience and an excellent memory.

Today the operating environment that most computer users are familiar and comfortable with is called a graphical user interface (GUI). Microsoft has dominated the operating system industry since the introduction of its Windows operating system that implements a GUI.

A GUI is a computer environment that attempts to simplify the user's interaction with the computer. This is done by representing programs, commands, files, and other options as visual elements in the form of graphical elements such as icons, pull-down menus, buttons, scroll bars, windows, and dialog boxes. By selecting one of these elements, either using a pointing device such as a mouse or a selection from a menu, the user can initiate different activities, such as starting a program, printing a document or running the command noted above for conducting a keyword search [Ref. 2].

The average warfighter in the field lacks the skill and training to invoke tools using complex command lines and, in fact, many well trained computer users would struggle with such a task. With the use of Interrogator, nearly any member of a combat ground unit, after very little training, would be equipped to conduct a search of a computer. If the data retrieved during the search yields

information that leads to further successful missions and lives being saved, then Interrogator will have served a useful purpose.

D. THESIS ORGANIZATION

In this thesis we present some of the basic concepts that are involved with computer forensics. Chapter II discusses a few of the techniques that are currently being practiced as well as the difficulties associated with these techniques relative to the project. Chapter II also provides an introduction to the privacy issues and legal aspects of computer forensics. Chapter III discusses the development in-field forensics product we produced as part of our work. This discussion includes operating systems that were investigated for use as the project's platform and the reason KNOPPIX was selected. Chapter III also provides details of the process of putting the project together and an introduction to remastering KNOPPIX.

The most important part of Chapter III is the implementation of the search tools that were selected for use in Interrogator. Chapter IV contains our conclusions along with a summary of the project and potential areas for future research. There are three appendices attached, that make up a significant portion of the thesis. Appendix A is the User Manual for Interrogator. It can be used as a standalone document for direction on using Interrogator. Appendix B contains detailed instructions for remastering KNOPPIX with the many customizations required for Interrogator. Appendix C contains the Java source code for

creating the GUI that links all the tools together. This code is what makes Interrogator the most user friendly option for any basic computer data retrieval.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

A. POINTS OF CONTACT

Major Hezekiah Barge was a member of the Information Operations Staff for First Marine Expeditionary Force and he is also a graduate of the Naval Postgraduate School. Major Barge initially approached NPS staff and proposed the idea of NPS students designing a GUI-based, user-friendly package for conducting quick computer interrogations in the field. This proposal was then brought to our attention by our thesis advisor as a potential thesis topic. Major Barge expressed the need for this because of the difficulty in using the tools that are currently available. He also noted that there was a lack of time, lack of training and lack of experience on the part of the users that would be able to take advantage of the data retrieved from the computer. Major Barge explained that if a tool could be designed that nearly anyone could use, regardless of training or experience, it could potentially provide invaluable intelligence information in the field.

In July 2006, NPS hosted an Information Warfare Workshop, and one of the main focus topics was Cyber Forensics[2]. During the cyber forensics talks, a presentation was given by Andrew Woods, a Special Agent with the Navy Criminal Investigative Service (NCIS). After

[2] The workshop was held at the SCI level at the Naval Postgraduate School 25-27 July 2006. It was sponsored by the department of Energy/Office of Intelligence (DOE/IN-1), Joint Information Operations Project Office (JIOPO) and Space and Naval Warfare Systems Center San Diego (SSC-SD). Participants included Naval Criminal Investigative Service (NCIS), Army Criminal Investigative Service (Army CIS), Air Force Office of Special Investigations (AFOSI) and Department of Defense Cyber Crime Center (DC3).

discussing this thesis research topic with Special Agent Woods, he was very enthusiastic about it and provided a point of contact within NCIS, Timothy Fowler. Special Agent Fowler works in the Cyber Department at NCIS and is directly involved with computer forensics with the armed forces in the field. After several email exchanges and telephone conversations, it became apparent that there is a real need for a user-friendly tool capable of collecting data to be analyzed for intelligence in the field.

Also presenting during the cyber forensics talk was a representative from Department of Defense Cyber Crime Center (DC3)[3]. The DoD Cyber Crime Center is the center of excellence for the investigation of computer crimes against the Department of Defense [Ref. 10]. Within DC3 is the Defense Computer Forensics Lab (DCFL), whose mission is to provide digital evidence processing, analysis, and diagnostics for any DoD investigation that requires computer forensic support. The representative from DC3 also reinforced the need for the type of product that was intended to be developed by this thesis research. DC3 has a Digital Forensics Toolkit available on CD through the Defense Cyber Crime Institute (DCCI). This toolkit will be discussed in the section on current tools.

B. GENERAL NOTES

As the United States fights the war against terrorism, there are thousands of fighting men and women in the field throughout the entire world. During any one particular

[3] DC3 is located in Linthicum, MD. It was founded on 1 Oct 2001 and consists of the Defense Cyber Crime Institute (DCCI), the Defense Computer Forensics Laboratory (DCFL) and the Defense Computer Investigations Training Program (DCITP).

mission in the field they may obtain electronic equipment that was or is currently in the hands of the enemy or suspected enemy. There are no written procedures that instruct the warfighter in the field what to do with a piece of electronic media or electronic equipment while they have it in their possession. Electronic media and equipment, in this case, refers to desktop computers, laptop computers, PDAs, cell phones, floppy disks, CDs, DVDs, flash drives, etc. In most cases, the equipment obtained in the field is passed up the chain of command. Once it has been taken from the battlefield, it is handled and processed by personnel designated and trained to handle electronic media but is not quickly interrogated for information that could prove useful as real time intelligence. In some cases electronic media and equipment are delivered to other organizations where deep forensic analysis takes place. Again, this analysis does not meet the needs of the warfighter for real time intelligence data in the field [Ref. 11].

Computer forensics examinations can be conducted for the purpose of performing a root cause analysis of a computer system that failed or is malfunctioning, or to find out who is responsible for misuse of computer systems, or perhaps who committed a crime using a computer system or against a computer system. Computer forensic techniques and methodologies are commonly used for conducting computing investigations - again, in the interest of determining what happened, when it happened, how it happened, and who was involved. These are not the only ways that computer forensics can be used; it can also be applied to fight the war on terrorism.

C. CURRENT PROCEDURES

The results presented in this thesis concern the development of a data retrieval tool called “Interrogator” that is designed to be used on desktop and laptop computers that have been confiscated. It is also capable of being operated from one computer while interrogating a hard drive removed from another computer or interrogating a USB flash drive. Interrogator is a data retrieval tool that has underlying functionality that is similar to a computer forensics tool. The difference is that the end user and the purpose of the data retrieval are significantly different from those for traditional computer forensic tools.

A discussion of traditional computer forensics is necessary in order to understand the difference between traditional computer forensics and the data retrieval that is conducted using Interrogator.

D. COMPUTER FORENSICS

Computer forensics, like other fields of forensics, comes with a complete and demanding set of requirements for handling and maintaining the electronic data that has been collected in order for that data to be used as legal evidence [Ref. 16]. Some of the privacy and legal aspects of electronic data retrieval will be discussed later in this chapter. These privacy and legal aspects, in a general sense, do not apply to the use of the Interrogator if it is used for its intended purpose.

One common definition of computer forensics is the application of forensic science techniques to computer-based material. One interpretation of this is that

computer forensics is the process of identifying, preserving, analyzing, and presenting digital evidence in a manner that is acceptable in a legal Proceeding [Ref. 9]. Using currently available tools, this collection of computer data requires the collecting agent or user to have an in-depth knowledge of computer hardware and software in order to retain and establish the integrity of the data collected.

In many cases, information is gathered during a computer forensics investigation that is not typically available or viewable by the average computer user, such as deleted files and fragments of data that can be found in the space allocated for existing files - known as slack space. Special skills and tools are needed to obtain this type of information or evidence.

E. TYPES OF FORENSICS

1. Disk Imaging

One of the most common and popular operations in computer forensics is to copy the entire hard drive; this is also referred to as disk imaging[4]. Once this is done it is then possible to search the image along with information that may be found on other peripherals and on back up media [Ref. 7]. Data contained in the disk image includes files, slack area, and unallocated space. This method provides the investigator the opportunity to work with the drive image and the data contained there at a slower and more determined pace and allows the flexibility of being able to use many more computer forensic tools when time is not an

[4] Disk imaging is the process of creating a bit-for-bit copy of a partition or hard drive.

issue. More importantly, this method allows the investigator to maintain the disk in the exact condition it was received. It can be proven in court that the image has not been tampered with through the use of hash algorithms such as MD5 and SHA-1.

As a forensic investigator, it is essential to ensure that a perfect snapshot of the system can be taken. One of the problems that ensue with this method of computer forensics is that nearly anything that is done to a system can change it. For example, unplugging the network cable from a running system will change the system — but leaving the network plugged in will change it too! Even if you decide to do nothing, the system will change because, at a minimum, the time on the system constantly changes. The best an investigator can do is to create a representation of the current state of the system as accurately as possible, documenting what steps he took in preparation for creating the image.

After imaging the device, the investigator can later mount[5] the image as if it were an actual hard drive partition. He can perform the same operations using the same tools as if he had the original disk in his possession. This allows the investigator to perform the analysis at his leisure rather than with the time constraints of an on-site investigation.

[5] Mounting is the process of making a file system ready for use by the operating system, typically by reading certain index data structures from storage into memory ahead of time.

2. Live Incident Response

There might be an occasion when a computer incident has taken place and the investigator cannot afford to shut down the system or remove the computer from a network. There are a few reasons why this might be true but the most likely reasons are that a proper backup has not been made or the only evidence of the incident is in memory. In these cases it is necessary to do a live incident response.

The data collected during a live incident response can be broken down into two categories: volatile and non-volatile. If the computer were to be shut down or unplugged from the power source the information in the volatile memory would be lost. A few examples of information that could be retrieved from this memory are current network connections, running processes and files currently in use by any running programs. If this information is not collected while the computer is turned on, then it will not be retrievable after the computer is shut down. Non-volatile memory is where all of the data is stored that could be retrieved during many other types of computer forensics analysis.

Interrogator was not designed to conduct a forensic data reduction on a computer that cannot be shut down. The warfighter in the field is not so concerned with network connections or running processes that would be lost in the volatile memory. There is a potential desire to see any open files, as stated above, if those files were some form of document. By using Interrogator, finding these files is still possible as they may have been saved in temporary file directories.

3. Network

Another aspect of the computer forensics field is the study of network forensics. This has become a necessity because of the constantly growing eCommerce industry and the stiff competition between companies to obtain the economic fortunes of the world. Enterprise networks can have the same concepts of computer forensics applied that are used on individual computers.

> *Network forensic analysis has actually been around as long as there have been networks; tcpdump and to a lesser extent windump have been used for years to analyze network traffic. In the past few years, however, network forensic software has become as sophisticated as applications designed for host-based forensics. [Ref. 9]*

Again, Interrogator was not designed to conduct any form of network forensics. In the battlefield, the need to obtain real time intelligence information does not include the ability to investigate a network intrusion. The tools that would be necessary for network forensics are called Intrusion Detection System (IDS) tools. These come in the form of both hardware and software but are not included in this thesis research. All the tools listed below, with the exception of DC3, are open source and can be downloaded free of charge from their respective websites listed in the References.

F. FORENSIC TOOLS

In this section we give the reader a brief description of some computer forensics tools. It is, by no means, an exhaustive list; just a brief description of a few tools that are currently being utilized in the field of computer forensics. Detailed information on Live CDs, which most of these tools are based on, can be found in Chapter III.

1. DC3

Of the forensic tools investigated for this thesis, the Digital Forensics Toolkit is the only non-bootable CD. It contains several forensic utilities that can be executed from an already-running Windows system. The toolkit is provided by the Defense Cyber Crime Institute [Ref. 10], but is only available to United States Department of Defense, Law Enforcement and Counterintelligence personnel. The forensic tools are applications that must be installed on the local hard drive and then executed from there. Although the tools are useful, they were not designed to retrieve the data that was identified as being useful as real time intelligence.

2. FCCU

Federal Computer crime Unit (Belgian) – FCCU is a CD-ROM based forensic disk that is based on KNOPPIX 5.01[6]. This particular forensics disc is very powerful and useful. The downside to this collection of tools, as with many forensic tools, is that it is difficult to use because all user interaction is through the command line interface.

[6] KNOPPIX is a Linux operating system distributed on a CD-ROM that can be run without installation.

The most recent version of the CD has 163 command line utilities. The user manual is available in PDF format from the FCCU web site [Ref. 11]. Although the user manual is helpful, it still requires the user to have significant experience in the Linux command line. This tool did not meet the requirements of the thesis study.

3. Helix

Helix is another Linux-based bootable CD-ROM. Helix claims to be a forensically sound environment. The procedures used to collect data are designed to retain the data in a format that makes it acceptable to use as legal evidence. This is not something that was considered to be necessary in the development of Interrogator or on the battlefield. Helix also has two operating modes: Linux and a Microsoft Windows executable feature. It can be used to conduct live incident response or to boot into the Linux environment. In the Linux environment it does boot into a graphical user interface but significant training is still required before it can be used by an in-field user. The most recent version of Helix is 1.8 and more information can be found at the Helix website [Ref. 13].

4. Insert

Insert is also a Linux-based bootable CD-ROM. It is made and distributed by Inside Security. One big advantage that Insert has is the fact that it is quite small compared to most bootable CDs. At only 60 megabytes it can be burned on to a credit card size CD-ROM by over-burning[7].

[7] Over-burning (or over-recording) is the common term referring to storing more data on a recordable disc than its reported capacity indicates.

This can also be a disadvantage since the number of tools included on the disc is limited by its size. It can be used from the GUI or from the command line. The command line, for reasons already stated, is not suitable for use in the field. Although it has a GUI, it is only used to launch the included forensic utilities. Each utility then has its own text-based interface, which the user must use to enter various parameters. The most recent version of Insert is 1.3.9a and more information can be found on the Insert website [Ref. 14].

5. F.I.R.E.

Forensic **I**ncident **R**esponse **E**nvironment is a bootable CD that boots into a very primitive but effective menu-driven interface. The most recent version is 0.3.5b and is available for download but has not been updated since May of 2003. The disc contains several tools that would not prove to be useful in the field. As a forensics CD, the tools that might prove useful include image processing tools but the disc really seems to focus on virus scanning and system penetration testing, neither of which is needed for the data retrieval that is desired by the warfighter. More information on FIRE can be found at the website [Ref. 15].

G. PRIVACY AND LEGAL ISSUES

The Interrogator product created as part of this research is intended to be used by military forces to interrogate equipment that has been obtained during military operations. Therefore the privacy and legal implications associated with using the Interrogator do not

fall under the general requirements discussed here. This section is intended to be for background information relative to forensics in general.

There are always privacy and legal implications to consider anytime data is gathered that can potentially be used against another person. It is certainly an issue with regard to collection of evidence for the purpose of establishing guilt and the need to prosecute for a crime committed. These privacy and legal implications are not expected to be to be an issue when dealing with data gathered from enemy electronic devices.

Most computer forensics involves the application of investigative and analytical techniques to acquire and protect potential legal evidence. The Fourth Amendment[8] to the United States Constitution limits the ability to search for evidence without a warrant. With this restriction, any search of a computer or similar electronic device would be considered a violation of privacy much like a file cabinet [Ref 16]. Again, the use of Interrogator in its intended environment would not lead to issues of legality or privacy. For more detailed explanations on this subject, see [Refs. 16, 17].

[8] The Fourth Amendment to the United States Constitution states: “The right of the people to be secure in their persons, houses, papers, and effects against unreasonable searches and seizures, shall not be violated, and no warrants shall issue, but upon probably cause, supported by oath or affirmation, and particularly describing the place to be searched, and the persons to or things to be seized.” [Ref. 16]

III. DEVELOPMENT

A. OPERATING SYSTEM PLATFORMS

In order to effectively analyze computer storage devices, it is necessary to have an operating system (OS) running on the computer with the connected drives. The computer should not be booted with its existing OS for several reasons. First, the contents of the underlying disk will change during analysis. Second, the existing OS may be password protected. Third, there may be traps in place to erase the data or send a signal when the native operating system detects an unauthorized user.

The solution is to boot the computer system from a known good OS. Traditionally, OSs require a lengthy installation and configuration procedure tailored for the specific hardware the system is expected to operate on. This type of OS would not meet the goals of the project because there is not enough time to do this near the battlefield and the fact that the target system would be modified. The recent advent of Live CDs (also called Live Distributions) allows a wide range of computer system configurations to be booted from a single disc or USB drive without having to install an operating system onto the computer's hard drive [Ref 18].

Live CDs exist for several different computer architectures including Intel x86, PowerPC, and AMD64 [Ref. 18]. The vast majority of personal computers sold are based on the x86 platform [Ref. 19], which is also the

target platform for the Interrogator project. Since it is the most common platform, nearly all Live CDs are designed to run on x86 computers.

Several Live distributions were investigated for use as the OS upon which Interrogator would run. Only one disc, BartPE [Ref. 20], was based on Windows while the rest were based on Linux. The final selection was based on hardware compatibility, boot time, and customizability.

BartPE initially looked promising from an ease-of-use perspective because it is based on Windows XP. Unfortunately, it had several problems that prevented its use for this project. Its hardware compatibility is rather limited, requiring third-party drivers to be added individually to support devices such as SATA and SCSI controllers. Customization was also limited since additional functionality had to be added with the use of BartPE plugins[9]. Finally, it did not appear to have the ability to mount disk partitions with read-only access, which would be an essential feature to prevent a user from altering or damaging the original data.

We evaluated Linux-based distributions including Ubuntu [Ref. 21], SuSe [Ref. 22], Fedora [Ref. 23], Kanotix [Ref. 24], and KNOPPIX [Ref. 48]. After experimenting with each of them on several systems, it was determined that KNOPPIX provided the best all-around solution. Outstanding

[9] A BartPE plugin may be a driver or an application that can be loaded into BartPE to add functionality.

features included full NTFS[10] read/write drivers and extensive customization capabilities via remastering. Additionally, KNOPPIX booted properly on all systems that were used during the testing phase. Details of these systems can be found in Chapter III, Section E, Tables 1 and 2.

B. DESIGN OF THE GRAPHICAL USER INTERFACE (GUI)

One of the project's primary goals was ease-of-use for computer novices. The GUI played a pivotal role in meeting the requirements. We wanted to minimize the workload and training required of the user while still being able to obtain useful data from the system.

Once the system has booted, the user is presented with a simple interface containing a series of buttons across the top, two output windows in the center, and three buttons at the bottom as seen in Figure 1.

[10] NTFS stands for New Technology File System. It is the standard file system of Windows NT and its descendants: Windows 2000, Windows XP, Windows Server 2003, and Windows Vista. NTFS replaced Microsoft's previous FAT file system used in MS-DOS and early versions of Windows.

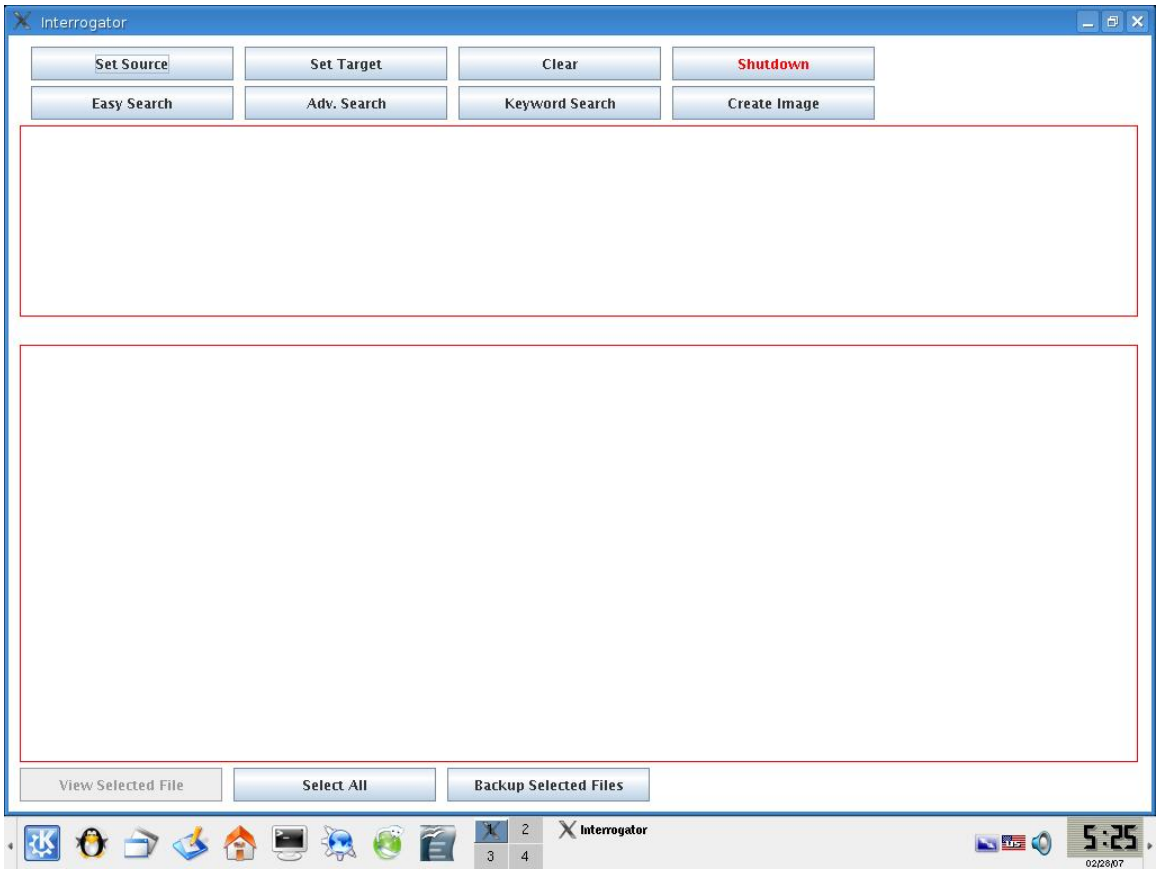

Figure 1. Main Interrogator Screen.

The two upper rows of buttons allow the operator to perform the following tasks:

1. Set Source

The source drive is the drive that is to be searched by the operator. When this option is selected, the operator will be presented with a pull down menu for selecting the source drive as shown in Figure 2.

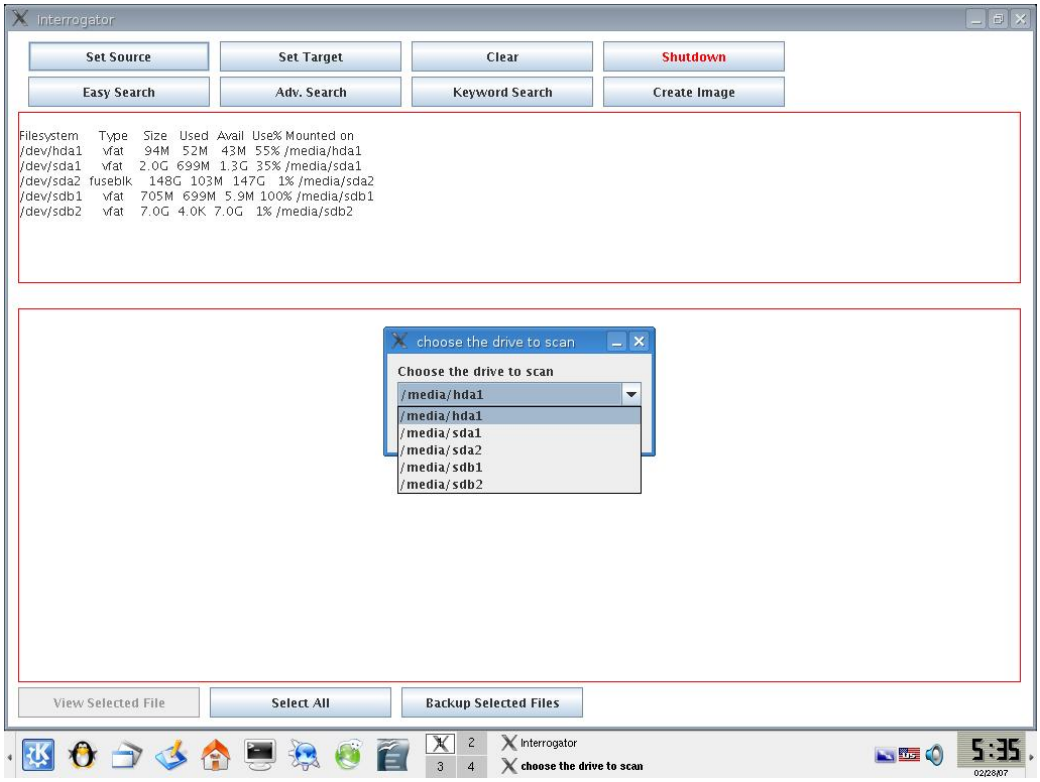

Figure 2. Set Source Window.

2. Set Target

The target drive is the drive to be used for saving data retrieved during the search. When this element is selected, the operator will be presented with a pull down menu for selecting the target drive as shown in Figure 3.

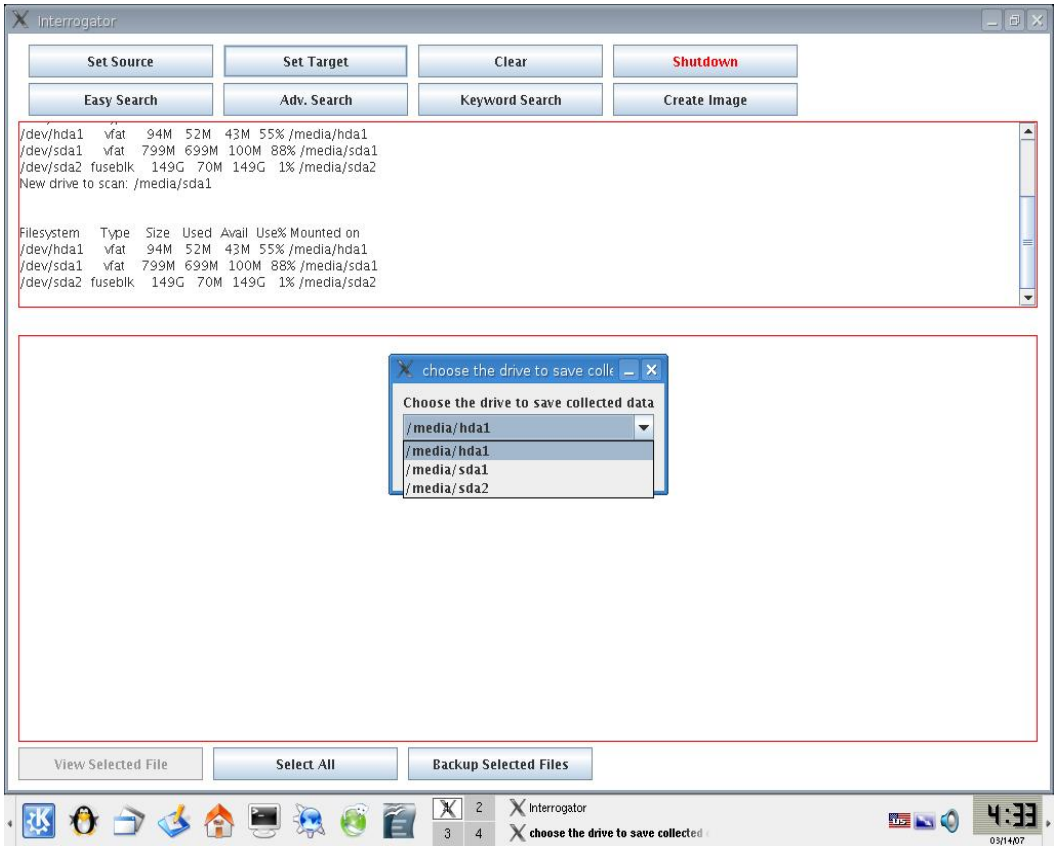

Figure 3. Set Target Window.

3. Easy Search

Easy Search allows the operator to conduct a search of the hard drive by specifying one or more types of files. The most common file extensions are built into the search function. When this element is selected the operator will be presented with a menu to select the file types to search for as shown in Figure 4. For example, if the operator wishes to search for maps and video recordings, he should select the images and videos checkboxes.

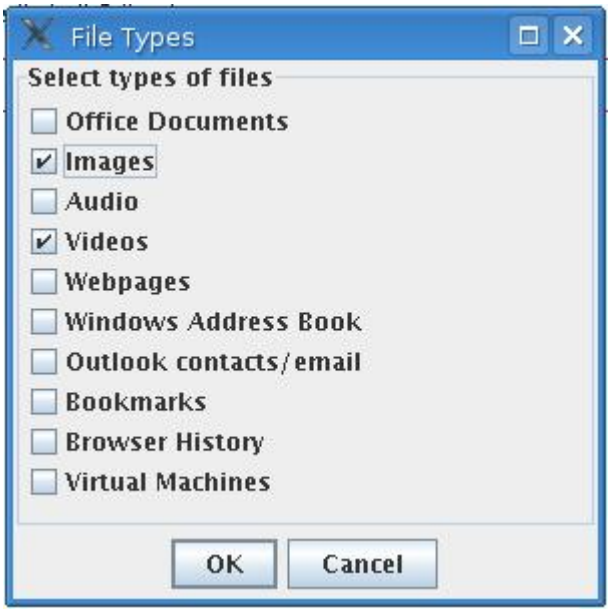

Figure 4. Easy Search Dialog Box.

4. Advanced File Search

This function is for users with an understanding of file types and extensions. Advanced file search allows the operator to search for documents based on the file extension of a specific file type. This is useful when the operator wants to search for specific file types or wishes to search for a file type not listed in the Easy Search. For example, if jpeg type images are the only files that need to be searched for, the operator would enter “.jpg” (without quotes) in the dialog box and only search for those files with a .jpg extension. If text files and avi videos is the requirement for the search, the operator would enter “.txt;.avi” (without quotes) as shown in Figure 5.

Figure 5. Advanced Search Dialog Box.

5. Keyword Search

The Keyword Search function allows the operator to search for keywords in the contents of files. When this option is selected, the operator will be presented with the dialog box as shown in Figure 6. On the left side, the operator must select the file types to be included in the search. On the right side, the operator must enter the keywords to be searched for in the file types that are selected.

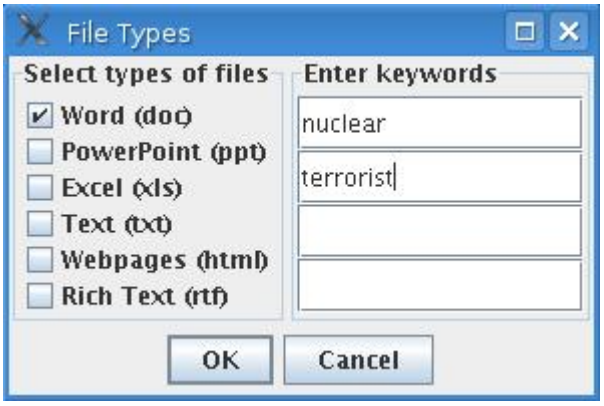

Figure 6. Keyword Search Dialog Box.

6. Create Image

The Create Image function allows the operator to create an exact copy of the desired partition (bit-by-bit)

and store it in a file on the target drive. The resulting file will be the same size as the original partition. When this element is selected, the operator will be presented with a dialog box as shown in Figure 7. The operator must choose the partition to be imaged.

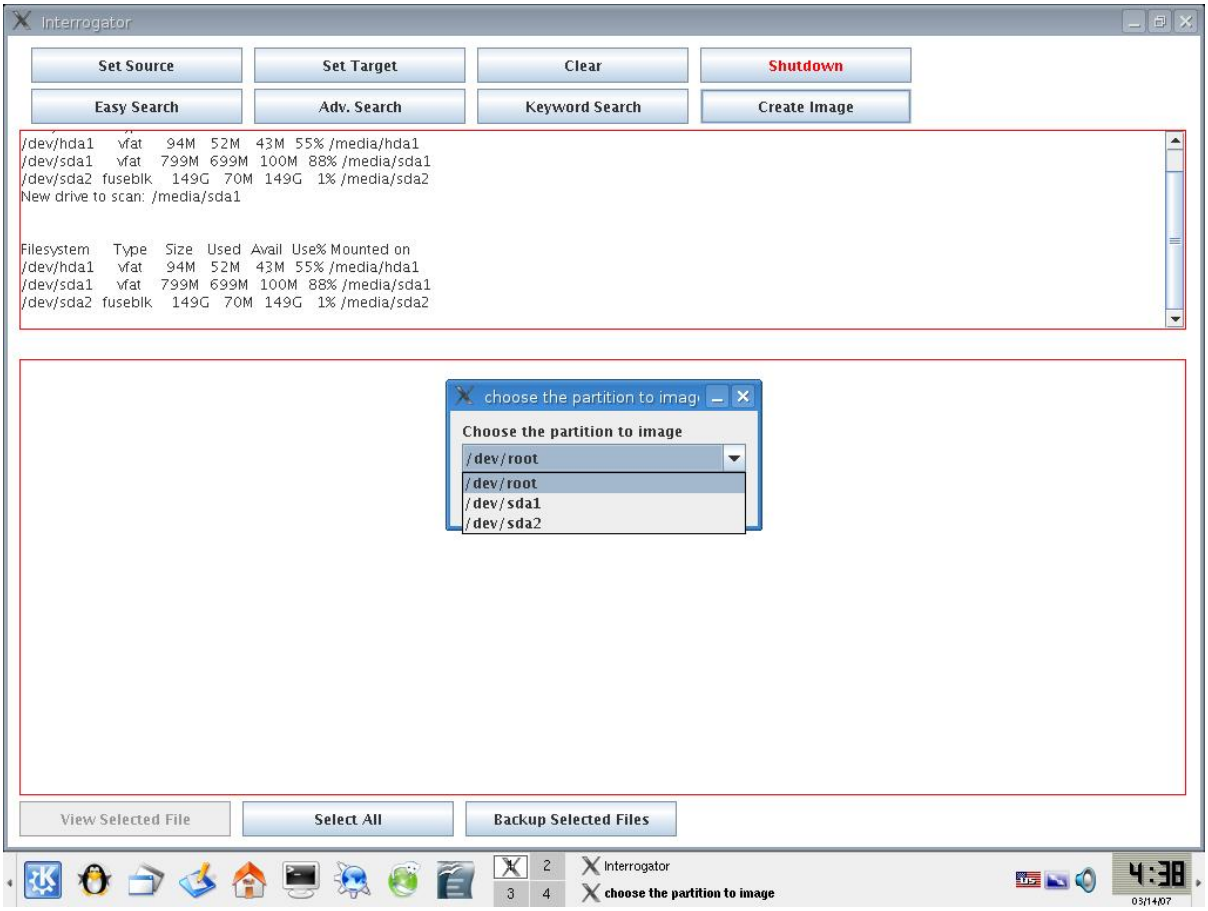

Figure 7. Create Image.

After choosing the partition to be imaged, the operator will then be presented with an input dialog box to enter the name of the image, as shown in Figure 8.

Figure 8. Naming the Image.

When the operator selects “OK” after naming the image, Interrogator will start the process of creating a forensically sound[11] image to the location previously specified and will also create an MD5 hash of the image and store the hash in the same folder.

7. Clear

Clicking this button will simply clear both the upper and lower windows of the screen so the operator may conduct another search without cluttering the screen with previous search data.

8. Shutdown

Selecting the shutdown button will terminate Interrogator and KNOPPIX, and shutdown the system. When the system has completed the shutdown process the operator will be presented with a shutdown screen as shown in Figure 9.

[11] Forensically sound means that the copy is a complete and accurate representation of the source drive. The copy must contain every bit, byte and sector of the source drive, including unallocated space and slack space.

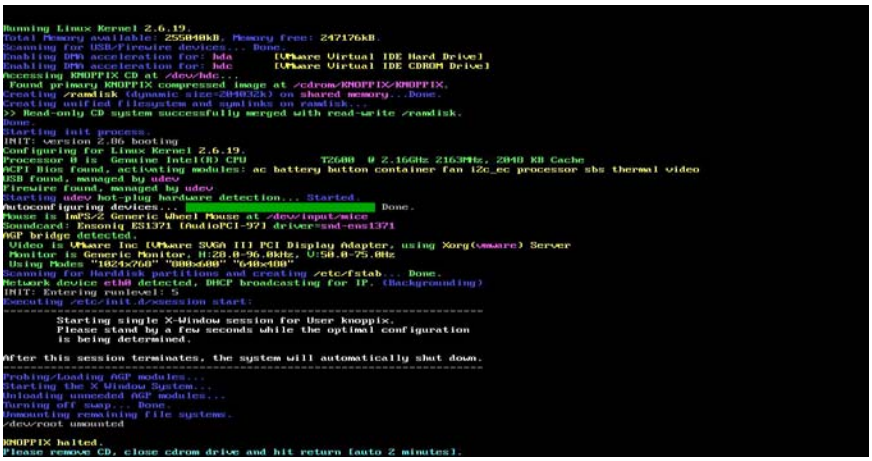

Figure 9. Shutdown Screen.

9. Output Windows

The upper output window provides the user with context-sensitive information such as partition information when setting the source and target. The lower output window displays a list of filenames resulting from the three types of searches.

10. View and Backup Files

The three bottom buttons allow the user to view the selected file, select all files, and backup the selected files to the target partition. The user may also double-click any filename to view it rather than clicking the button.

11. Keyboard Input

There are only a few times the user is required to input characters from the keyboard. The user must type the name of the target folder to store collected files. He also needs to type keywords for a keyword search. The advanced search also requires user input, but the average

user is not expected to use this feature. Finally, creating a partition image requires the user to type the name of the image file.

C. REMASTERING

Taking a Live CD and customizing it to meet specific needs is known as remastering. This project required extensive use of the remastering process.

The KNOPPIX Live CD contains over 1.7 GB of data and programs encompassing the OS and all the included software and utilities. Standard recordable CDs only store 700 MB. The Live CD is able to store more than 700 MB through the use of a compressed loopback device (cloop) [Ref. 25].

Some changes such as creating custom graphics and adding tools to the file system were relatively trivial. Automatically mounting all partitions in a read-only manner and creating custom file associations were more time-consuming. The most complex customization involved integrating Outlook 2003 (a Microsoft Windows application) into KNOPPIX. Detailed steps of the remastering process can be found in Appendix B.

D. INCORPORATED TOOLS

Implementing Interrogator's core features required functionality obtained from several sources. Java offered built-in methods to perform file searching[12] as well as linking the GUI to command line utilities. Several open source utilities were used to convert machine-readable files to a format suitable for humans. Partition imaging

[12] Java's File class provides methods for accessing files and directories.

functionality was provided by dcfldd, which is described below. Finally, a pair of commercial applications was necessary to read Outlook PST files.

1. Standard Java Methods

The Java language and libraries include a wealth of classes and associated methods that helped in the construction of the Interrogator GUI. Swing and the Abstract Window Toolkit (AWT)[13] were used to create the entire interface including all buttons, windows, and lists. Java's File class provided access to directories and files, which were used in conjunction with loops to traverse target partitions. The Process and Runtime classes permitted the direct execution of command line utilities from within the GUI and access to all output generated by those utilities.

2. Standard Linux Commands

Linux has several built-in command line utilities that were useful for this project. Every command in Linux contains a myriad of arguments to control its operation. This functionality is a large part of what makes Linux so powerful and, at the same time, difficult to use. Commands of particular use to Interrogator include ***find*** [Ref. 3], ***egrep*** [Ref. 6], ***mount/umount*** [Ref. 26], ***df*** [Ref. 27], ***cp*** [Ref. 28], ***and shutdown*** [Ref. 29]. Each of these will be briefly described here. Specific arguments and syntax used can be found in Appendix C.

[13] Swing is a Java toolkit for developing graphical user interfaces. It includes elements such as menus, toolbars and dialog boxes. Swing is written in Java and is platform independent, unlike the Java Abstract Window Toolkit (AWT), which provides platform-specific code.

The ***find*** command searches a directory for a given filename or regular expression. The ***egrep*** command searches a file's contents for a particular string or regular expression. Partitions[14] are mounted and unmounted with ***mount*** and ***umount***, respectively. Partition info such as name, size and usage details are provided by the ***df*** command. The ***cp*** command provides a mechanism to copy files from one location to another. Finally, ***shutdown*** will gracefully stop all services, unmount partitions, and prepare the system for removal of power.

3. Wabread[15]

Windows Address Book files were parsed with the help of a utility known as wabread (a.k.a. libwab) [Ref. 30]. Usage is fairly simple, requiring only an input file as a command line argument. The resulting human-readable address information is output to standard output or a file, if desired.

4. Pasco

A key function of Interrogator is the ability to easily read web browser history files. Two web browsers were targeted for this project: Microsoft Internet Explorer (IE) and Mozilla Firefox. These two browsers were used because they are the most widely used browsers. According to OneStat.com, Internet Explorer is utilized by approximately 85 percent of the market and Mozilla Firefox

[14] A partition is an allocated portion of a hard disk that can be formatted with a file system. A hard disk may contain multiple partitions.

[15] Wabread (a.k.a. libwab) is an open-source command line utility that you can use to export your addresses from a Windows Address Book used in Microsoft Outlook and Outlook Express)

is utilized by approximately 12 percent [Ref. 31]. The remainder of the market is taken up by Netscape, Opera and Safari. The most widely used web browser application is Microsoft Internet Explorer (IE). Web browsers, in their default configuration, keep track of recently visited websites and store the data in a history file for the convenience of the user.

Implementing the ability to read IE history files was obviously very important. Pasco was written by Keith J. Jones and parses an IE history file into a human-readable output [Ref. 32]. It was incorporated into Interrogator through the use a Bash[16] shell script that sends the index.dat file to Pasco as an argument and then sends the output to a kwrite[17] window, allowing the user to easily see what websites were recently visited.

5. Mork.pl

Mozilla Firefox stores its history in a file called history.dat, which is located in the user's profile. This file is in a poorly documented format known as Mork [Ref. 33]. Jamie Zawinski wrote a perl script[18] named mork.pl to convert the history file to a format better suited for human interpretation [Ref. 34]. Mozilla and SeaMonkey also store their history in history.dat using the Mork format and are supported by Interrogator. It was integrated into Interrogator in the same fashion as Pasco.

---

[16] Bash is a Unix shell. It is the default shell on most Linux systems and can be run on most Unix-like operating systems.

[17] KWrite is a lightweight text editor for the K Desktop Environment (KDE) and is similar to Microsoft Wordpad.

[18] A script is a program written is a scripting language. It allows the implementation of a series of commands without having to input each command individually.

6. dcfldd

Imaging is performed with the help of dcfldd [Ref. 35]. It is an enhanced version of the well known dd[19] utility with features useful for forensics. The relevant extra feature used in this project is the ability to output a hash of the image. The image is file that contains a copy of a disk partition, copied bit-for-bit from the original. The utility dcfldd was incorporated into Interrogator with the appropriate parameters gathered utilizing the GUI interface and passed to a shell script behind the scenes. The contents of the script can be found in Appendix C.

7. CrossOver Linux and Outlook 2003

Outlook is the most widely used Personal Information Manager (PIM) [Ref. 36]. PIMs manage email, appointments, tasks, and contacts. Because of its extremely large market share, including the ability to view Outlook data was crucial to this project. Unfortunately, implementing this capability was not an easy task.

Countless hours were spent searching for a way to parse Outlook PST files. Outlook uses a “PST file” to store the data mentioned above. The major roadblock was Microsoft’s use of a proprietary and unpublished format for the PST file. No PIMs are capable of importing Outlook 2003 data except for Outlook itself. There is a Linux tool to parse Outlook files, libpst [Ref. 37], but it is limited to use with Outlook 97/2000 files.

[19] dd, short for data definition, is a common Linux program whose primary purpose is the low-level copying and conversion of files.

We came to the conclusion that Outlook 2003 PST viewing capability would require the use of Outlook 2003 itself. This posed several problems since Outlook is very large and only runs on Windows 2000, Windows XP or higher. The solution came via a commercial application from CodeWeavers called CrossOver Linux [Ref. 38]. CrossOver allows certain Windows applications to run directly in Linux without the Windows OS. Fortunately, the latest version of the software, released during the development of Interrogator, supports Outlook 2003. Using Outlook 2003, Interrogator can also open PST files from older versions of Outlook including Outlook 97, 98, 2000, and 2002.

Integration of CrossOver and Outlook into the Interrogator KNOPPIX disc required eliminating a large number of packages from the original KNOPPIX distribution since the two programs together consumed several hundred megabytes of storage. Space was gained by removing redundant applications and other software deemed unnecessary toward achieving the project’s goals.

Another problem was encountered after integrating the two programs into the project. Outlook does not have the capability of opening PST files from a read-only source. The partition being analyzed is mounted in a forensically sound read-only manner to prevent changes to the original contents. The solution was to first copy the PST file to the KNOPPIX home directory, where it is stored temporarily

“in RAM[20]. A script was written to automate the steps necessary to view the PST file, the details of which can be found in Appendix C.

E. TESTING

Testing of Interrogator KNOPPIX was conducted for actual operational capabilities on four laptop computers and one desktop computer. These computers ranged from Celeron processors to Dual Core processors. Preliminary testing consisted of ensuring the core functions performed as intended. Timing of the operations was not taken into consideration during this first part.

After Interrogator was verified to function according to design it was then tested for boot up and search times on twenty different computers. Several brand name computers were selected with varying processor types and speeds. The focus of the test was to compare the amount of time it took to boot the system and then display the Interrogator Main Screen and the amount of time required to conduct an Easy Search for images. This testing was not conducted for the purpose of comparing the operational characteristics of each system but only to demonstrate the capability of Interrogator on a wide range of systems.

The boot time is dependent on the processor type and speed, and the speed of the CD drive, if booting from a CD. The time required for all searches are dependent on the size of the hard drive, the number of files it contains, and the drive’s performance characteristics (seek time and

[20] Very large PST files may cause problems during analysis if the file size exceeds the amount of available system RAM. The largest PST file opened during testing was 722MB on a system with 2GB RAM.

transfer rate). The last test, indicated with a yes or no result, was whether or not the computer successfully booted from a USB device. Boot time via USB was observed to be significantly faster than CD-ROM, however, we decided to report the worst-case boot time since many computer systems are not capable of booting via USB. It should also be noted that overall performance of Interrogator is markedly faster via USB because of the significant seek time advantage of flash memory and hard drives compared to CD-ROM.

The basic information documented for testing included computer make and model, number of processors, amount of RAM installed, type and speed of the processors, boot time, hard drive interface, size of the primary partition, search time, and whether or not the system could boot from a USB flash drive. The results of testing ten laptop computers are shown in Table 1 and the results of testing ten desktop computers are shown in Table 2.

Laptop Computers

Make/Model	# of Proc	Ram Size	Processor Type and Speed	Boot Time mm:ss	Hard Drive Interface/Size/ Used (GB)	Srch Time mm:ss	USB Y/N
Dell Latitude C640	1	384 MB	Mobile P4 2.6GHz	03:25	30	01:05	N
Toshiba Satellite A105	1	2 GB	Celeron M 1.6GHz	03:45	Serial/ 56 / 48	02:25	N
Dell Latitude D620	2	2 GB	Intel Core Duo 2.16 GHz	02:55	Serial/ 73 / 44	01:25	Y
Compaq Evo N1015v	1	512 MB	AMD Athlon 1.4 GHz	04:05	IDE/ 28 / 13	01:50	N
Gateway 400VTX	1	765MB	Celeron 2.0GHz	05:00	IDE / 28 / 9.7	02:25	N
Sony Vaio VGNC260	2	2 GB	Core Duo 1.83GHz	03:25	Serial/150/43	00:52	Y
HP Pavillion 9233	2	2 GB	Core 2 1.66GHz	02:40	Serial/110/16	01:20	N
HP Pavillion 6243	2	1 GB	Intel T2060 1.6GHZ	02:47	Serial/110/14	01:45	N

Table 1. Laptop Computer Testing Data

Desktop Computers

Make/Model	# of Proc	Ram Size	Processor Type and Speed	Boot Time mm:ss	Hard Drive Interface/Size/ Used (GB)	Srch Time mm:ss	USB Y/N
Dell Optiplex GX400	1	512 MB	P4 1.4 GHz	03:05	IDE/58/6	00:32	N
Dell Optiplex GX620	2	3 GB	P4 3.6 GHz	04:05	Serial/ 150 / 42	12+	Y
HP Pavillion 1777	2	2 GB	Core 2 2.13GHz	02:00	Serial/ 400 / 36	00:45	N
Sony VAIO VGCLS20E	2	2 GB	Core 2 1.66GHz	03:00	Serial/ 140 / 43	00:47	N
Dell Optiplex GX270	2	2 GB	P4 3.0 GHz HT	03:45	IDE/ 75 / 13	02:55	N
Compaq Presario SR2180NX	2	1 GB	Pentium D 2.8 GHz	02:45	Serial/ 230 / 25	02:00	N
Acer Aspire AST180-UA350B	1	512 MB	AMD ATHLON 2.2 GHz	03:15	Serial/ 70 / 23	00:30	N
Acer Aspire ASL100-UA380A	1	1 GB	AMD ATHLON 2.4 GHz	03:05	Serial/ 70 / 25	01:00	N
Dell Dimension 4700	2	3 GB	P4 3.0 GHz HT	03:07	Serial/ 140 / 64	01:20	Y

Table 2. Desktop Computer Testing Data

THIS PAGE INTENTIONALLY LEFT BLANK

IV. CONCLUSIONS

A. SUMMARY

The purpose of this thesis was to create a package of consolidated forensic tools that would conduct a search of a computer and attached storage devices. The final product would need to meet the following requirements:

- useful for a quick interrogation
- conducive to use in the battlefield
- usable by personnel without specific training in computer forensics

The need and desire for this field-level computer exploitation package was first brought to the attention of Naval Postgraduate School Staff by Marines in the Information Operations Department at Camp Pendleton, CA. The request was for the development of a package that could be used by anyone in the field to conduct computer searches. This requirement meant the package needed to be very user-friendly.

The development of this package first involved determining what information would be useful as real-time intelligence. Then, existing tools available and in-use had to be examined. The research results were essential in developing a plan to design the package and decide on the best method of compilation.

Programming the Interrogator interface began before a final decision had been made for the host OS. Java was the logical choice due to its cross-platform portability and ease-of-use. After investigating many options for a host operating system, a decision was made to use KNOPPIX. This decision was based on its size, boot time, and extensive hardware compatibility.

Remastering KNOPPIX several times was necessary in order to integrate the changes required during the development of the package. While the process of conducting this procedure is already well documented, it is not a trivial task and proved to be time consuming but essential to the development. Remastering allowed the inclusion of additional tools and complete customization of the OS. The remastered distribution was made to accommodate both bootable CDs and USB drives.

The final package, called Interrogator, was created with these tools and with the aforementioned goals in mind. It was designed to search for many types of files including, but not limited to, office documents, graphics, audio, video, and browser histories. Along with being capable of searching and retrieving many different types of files useful to the war-fighter in the field, Interrogator is also very easy to use. It is completely graphic and menu driven and comes with all of the tools to both conduct the searches and save the data that has been collected. Interrogator is not capable of conducting an analysis of the data; however, without data collection, analysis is not possible.

B. FUTURE RESEARCH

1. Open Source Production

One of the goals of this product was to maintain it as a completely open source product. Unfortunately, we were not able to read Outlook PST files with open source software. Interrogator is capable of reading these PST files but there are two non-open source programs involved. Crossover Linux was purchased to run Windows software in a Linux environment and Outlook 2003 is being used to read the PST files. Crossover Linux is a commercialized version of Wine [Ref. 39] with

additional features necessary for integration with Interrogator [Ref. 40]. Wine does not currently support Outlook 2003 nor does it seamlessly integrate[21] with KDE. Perhaps a future version of Wine will include these features. If reading these PST files can be done with completely open source software then the Interrogator package becomes a bit more flexible and affordable.

2. Foreign Language Support

Interrogator has not been tested with drives containing foreign language documents. Although file extensions and format should still be compatible, the language used in the documents may very well be something other than English. Interrogator may have problems opening files with names containing non-English characters as well as searching for keywords with non-English characters.

3. One-Click Searching

The user friendliness of Interrogator has been one of the objectives of this study from the beginning. Making it even easier to use could benefit the user in the field and would make it usable by a larger population. One way of doing this would be to develop a series of canned queries that would implement specific data retrieval, copy the data to a target drive and exit the program without the need to do each step individually. Implementation of these queries could then be accomplished through the use of the main menu or hot-keys. This would also allow for much quicker data retrieval in the event that time is critical and does not permit a full system scan.

[21] Seamless integration includes the ability to double-click on an email attachment and have it open in the appropriate application.

4. Instant Messenger Data Collection

Instant Messengers store contact information, passwords, and in some cases, chat lots. The common instant messengers are AOL Instant Messenger, Yahoo Instant Messenger, MSN Web Messenger, Google Talk and ICQ. Interrogator could be enhanced to search for instant messenger files and collect this information.

5. Anti-Reverse Engineering

In the event someone obtains a copy of Interrogator that should not have it, there should be a means of protecting all of the files contained in the program to prevent an analysis of the methods used for data collection. If an adversary determines the types of files targeted or the searching methods used by Interrogator, it would be easy to hinder Interrogator's usefulness.

6. Eliminate Unnecessary Features

There are many default features that are started during the KNOPPIX boot process. The main startup script for KNOPPIX is located at /etc/X11/Xsession.d/45xsession. Some of these features are unnecessary and could be eliminated to speed up the boot process. Additionally, hundreds of unnecessary packages are included with the KNOPPIX distribution. Many were already eliminated for this project but further research of the remaining packages could result in more eliminations.

7. Remote Server Data Transfer

KNOPPIX comes preloaded with a full networking suite. A future function of Interrogator could include a script for easily transmitting retrieved data to a remote server. Secure data transmission could be ensured through the use of the SCP

command line utility. SCP would be useful if high priority files are found and need to be immediately sent to a remote location. Another possibility would be using cryptcat [Ref. 41] to securely transfer data that is not already stored in a file (e.g., a directory listing). Naturally, these utilities should be utilized in the background and accessible to the user through buttons in the GUI.

8. Additional PIM Support

Although Outlook is the most widely used personal information manager, it would be beneficial to include support for other common PIMs such as Outlook Express and Mozilla Thunderbird.

9. Additional Address Book Parsing

Currently, the only address books that are capable of being read by Interrogator are Windows Address Book (WAB) and PST files from Microsoft Outlook. Another future feature to be designed into Interrogator would be the ability to parse additional address books such as the one for Mozilla Thunderbird or address books saved on local hard drives by other third party email programs.

10. File Search by Format

File extensions are the primary means of associating files with applications in Windows environments. In Linux environments, file extensions are often used but not required, which means Interrogator may be less effective against Linux computers. Linux file systems were not a priority for this project since NCIS indicated Linux is rarely encountered in the field [Ref. 42].

There is occasionally an attempt to hide a particular file simply by changing the extension of the filename. This type of data obfuscation would hinder the search capabilities of Interrogator. Interrogator only searches for files based on the extension for each file type. A method of searching for files based on other common file format structures would provide additional assurance that all useful intelligence is retrieved.

11. Searching Within Compressed Archives

Many computer operators use the popular ZIP file format [Ref. 43] to store files or exchange files with other users via the Internet. A ZIP file is an archive that may contain one or more files in a compressed or uncompressed format. A useful addition to Interrogator would be the ability to search within ZIP and other archive files.

12. Virtual Machine File Search

Interrogator is capable of looking for and identifying virtual machines based on the file extensions for those machines (vmdk, vmsd, vmx, vmem, and vmss). It is not capable of looking at the file system inside the virtual machine. To be able to complete the same level of search inside a virtual machine that is currently capable on the host operating system would be very useful. If the host operating system is only used as a method of running VMWare or Virtual PC with all of the real data stored in the virtual machine, it is not possible to analyze the data in a timely manner with Interrogator. VMWare had released a Perl script for mounting virtual disks in Linux, but the file is currently inaccessible from their website [Ref. 44]. Alternative solutions may be available and should be implemented in future research.

13. Searching Logical Volumes

Certain operating systems use Logical Volume Management (LVM) partitions by default. This type of partition is capable of being mounted by KNOPPIX, but requires several steps and is not currently incorporated into Interrogator. Further development should include the ability to analyze LVM partitions.

14. Malicious Software Search

There are many known Trojans, Worms and Viruses in the form of executable files. Many of these files have hash values associated with them that are available in easily accessible databases. The ability to search for malicious executables and compare the hash values against known malicious executables would be useful in determining how badly the computer is infected before continuing a search and potentially downloading a file that could corrupt the target drive and/or be spread to other computers. Additionally, it would be useful to incorporate an anti-virus utility such as ClamAV.[22]

15. Formatting Output Files

Interrogator is capable of searching for browser histories from Mozilla Firefox and Microsoft Internet Explorer. The output files resulting from such a search contain all of the data that is required to determine internet web sites the user has visited. The output files are difficult to read, however, because of the formatting. An improved output format, such as a spreadsheet or comma separated values capable of being imported into a spreadsheet, containing only the most critical fields would expedite the ability of the operator to determine the existence of the type of data being searched for.

[22] ClamAV is a free, open-source anti-virus toolkit for UNIX.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. USER'S MANUAL

This appendix is intended for personnel who will be performing a forensic data search of a laptop computer, desktop computer, removed hard disk, external hard disk or USB flash/thumb drive.

A. INTRODUCTION

Interrogator KNOPPIX is an easy-to-use, field-deployable, bootable CD-ROM disc. It provides a user-friendly, graphical user interface (GUI) to perform computer forensics or basic data retrieval. The program is designed to be used by someone with limited computer skills, but the disc also includes a complete Linux distribution with command line interface for users requiring advanced functionality.

B. SYSTEM REQUIREMENTS

Interrogator KNOPPIX is based on KNOPPIX 5.1.1, which supports a wide variety of computer hardware. The minimum requirements for the system used to boot Interrogator are listed below.

- Intel Compatible CPU (i486 or later)
- 256 MB of RAM (at least 512 MB recommended)
- Bootable CD-ROM drive
- SVGA-compatible (800 x 600) graphics card
- Serial, PS/2 or USB mouse

C. INSIDE THE KIT

If you received this product as part of the complete kit, this is what should be included:

- This user manual
- One bootable CD with the complete Interrogator program
- One 8 GB USB flash drive. This includes the complete Interrogator program and can also be used as a target drive to save retrieved data.
- One 160 GB, USB powered, external hard drive. This will be used for saving disk images but may also be used as a target drive to save retrieved data.
- One USB to SATA/IDE hard drive adapter. This can be used to attach a hard drive that has been removed from a computer, to a computer that is being utilized to run the Interrogator program.
- One AC power supply for the USB to SATA/IDE hard drive adapter.

D. BEFORE YOU BEGIN

Before you begin, you must know a couple of things. You need to decide if you will use the CD or the USB flash drive to boot the system. Nearly all computers will boot up from a CD. This will be the primary means of booting most systems. Very few computers will boot from a USB flash drive, but it is the faster method. This will normally work only on very new computers.

Next, you need to decide where you will save the data that you will retrieve. The most convenient target drive will be a USB flash drive and you can use the flash drive included in the kit. The only limitation to using a flash drive is available storage. If you are only going to look for and retrieve documents and images, a flash drive will probably suffice.

If you intend to image a hard drive, your target drive needs to have at least the same capacity as the hard drive being imaged. For this you will need the portable hard drive included or some other hard drive of your own choosing.

You can use your own computer to retrieve data from a hard drive that has been removed from another computer. To do this you will need to use the USB to SATA/IDE hard drive adapter.

E. BOOTING INTERROGATOR KNOPPIX

Ensure all hard drives and thumb drives you wish to analyze are attached to the computer before turning on the power. If you plan on backing up files that you retrieve during the search, ensure the backup device is also connected. Insert the Interrogator KNOPPIX disc into the CD-ROM drive immediately after turning on the power.

Different computers require different methods to boot from a CD-ROM disc. Pay careful attention to messages that appear on the screen as the computer is starting. There should be a message stating to press a certain key for the boot menu. This key may be “esc” or “F12.” Press the indicated key and a menu will be presented to select the drive you wish to boot the computer from. If the computer is capable of being booted from a USB flash drive, then there will be a USB option on this menu. Select the option that you want to use to boot the computer.

If the computer has successfully recognized the Interrogator KNOPPIX disc a screen will appear with the Interrogator KNOPPIX logo as shown in Figure 1. Press the “Enter” key to start the utility. It may take up to five minutes to start if the computer has a slow CD-ROM drive. You will see the main Interrogator screen shown in Figure 2 when the computer has completed the boot process.

Figure 10. Interrogator Splash Screen.

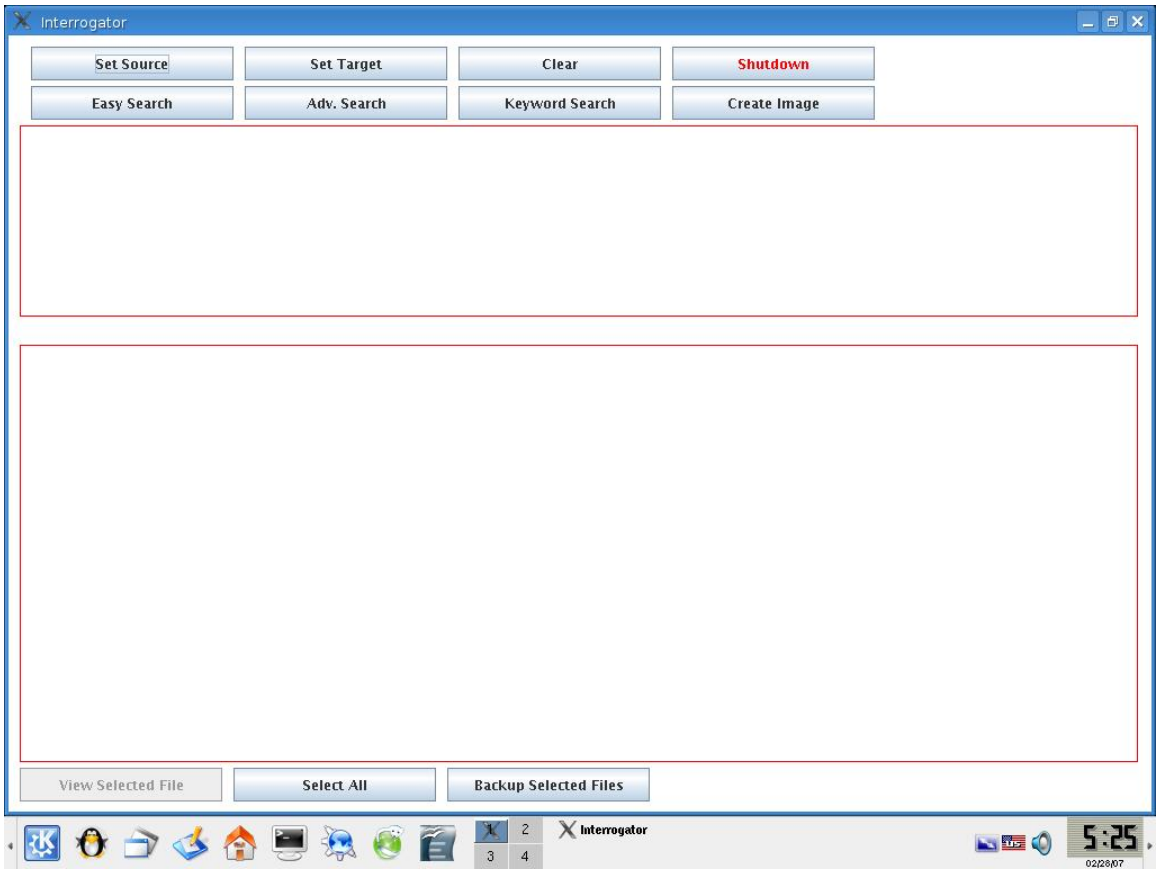

Figure 11. Main Interrogator Screen.

F. SELECTING THE SOURCE AND TARGET DRIVES

This section will describe the process of selecting the drive to be interrogated and selecting a drive for saving data.

1. Set the source drive to analyze

Once you have booted the computer and have reached the main menu for the Interrogator program the first thing to do is to tell the computer which drive to interrogate:

Click the “Set Source” button.

A listing of all available drives will appear in the status window as shown in Figure 3. There will also be other information about the drives that you can choose from. Most important is the size because this is generally the first

indicator of which drive to choose. It will also tell you how much of the drive is in use and how much is available. This will help you decide which drive to choose in the pull down menu.

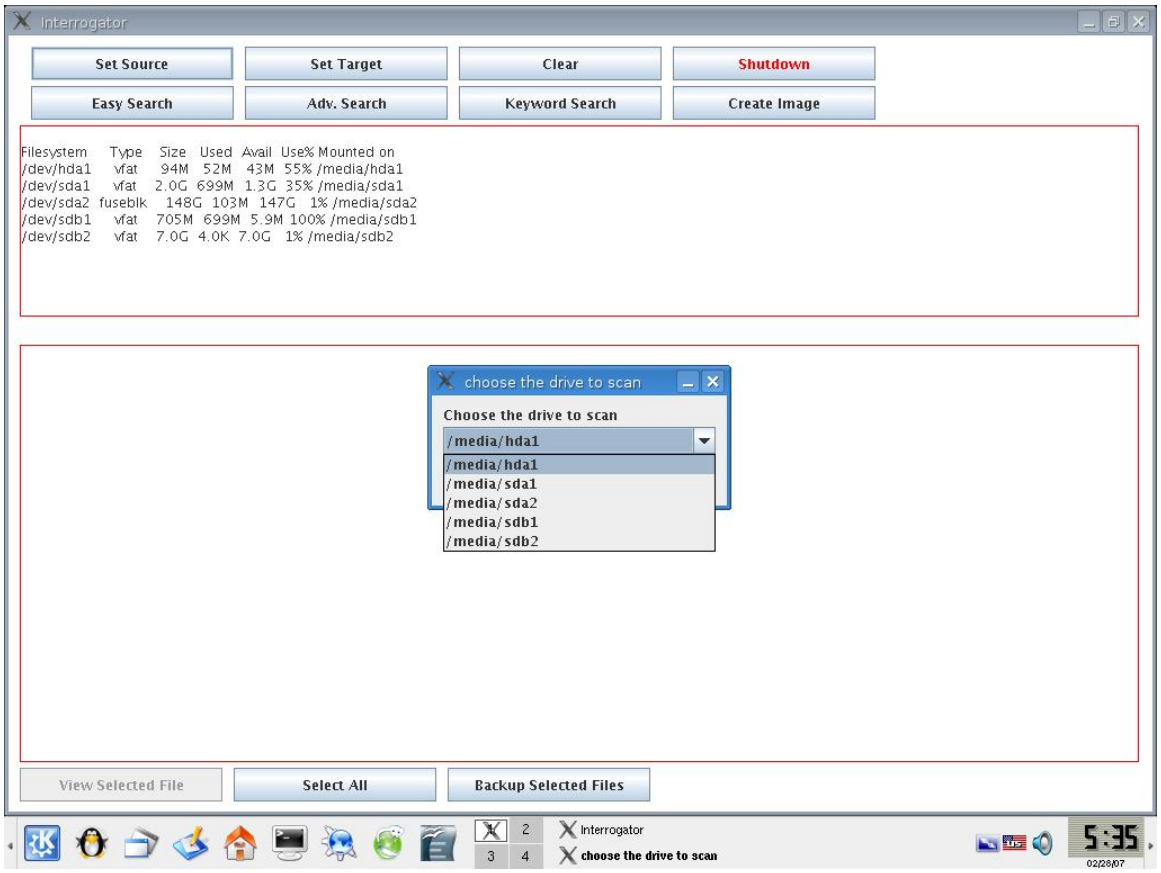

Figure 12. Set Source Window.

Choose the drive you wish to analyze in the drop-down menu and click ok.

2. Set the target drive

This step is optional and only necessary if you plan to make a copy of files found during the search step.

Click the “Set Target” button.

A listing of all available drives will appear in the status window including their size and other information just like the list that appears when you select the source drive in Figure 3.

In the drop-down menu, choose the drive where you would like to store backed up files and click ok.

If you do not have write access to the selected drive you will be asked if you would like to enable write access.

Finally, you will be prompted to enter a name for the folder where all backup files will be stored in the target drive.

G. SEARCH OPTIONS

Now that you have booted the system and selected the proper source and target drives, it’s time to begin your search. This section will detail what you need to know to conduct searches of the source drive. You have a couple options when it comes to conducting searches. You may conduct an easy search or an advanced search. Both methods will be described here.

1. Click the “Easy Search” button.

The dialog box in Figure 4 will appear where you can select the type of files to search for. You may select any combination of file types and then click ok.

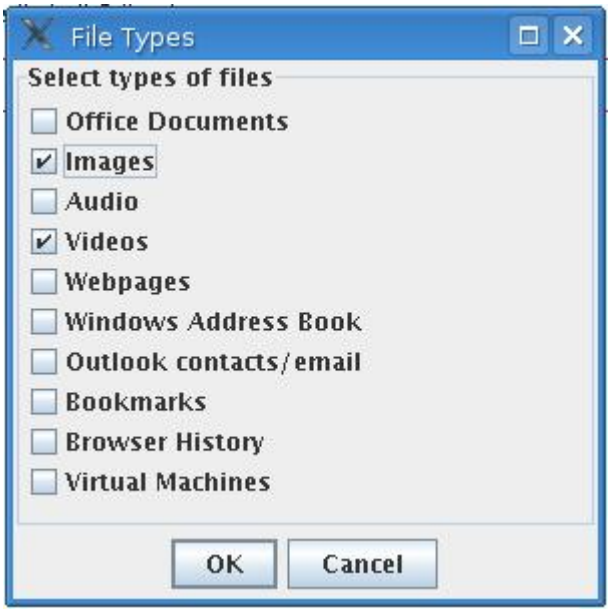

Figure 13. Easy Search Dialog Box.

Interrogator will scan the source drive for all files of the selected types. This is done by searching for the files based on the file extension (i.e., .doc, .pdf, .jpg, etc.). The duration of the search process depends upon the size of the source drive and the number of file types selected. It is best to only select one file type to expedite searches and to avoid a large list of resulting filenames. If you know that you only want to see documents, then you should only select this option. Likewise, if you know that you only want to see images, then you should only select this option.

All files meeting the chosen criteria will appear in the results window, which is the lower half of Figure 5.

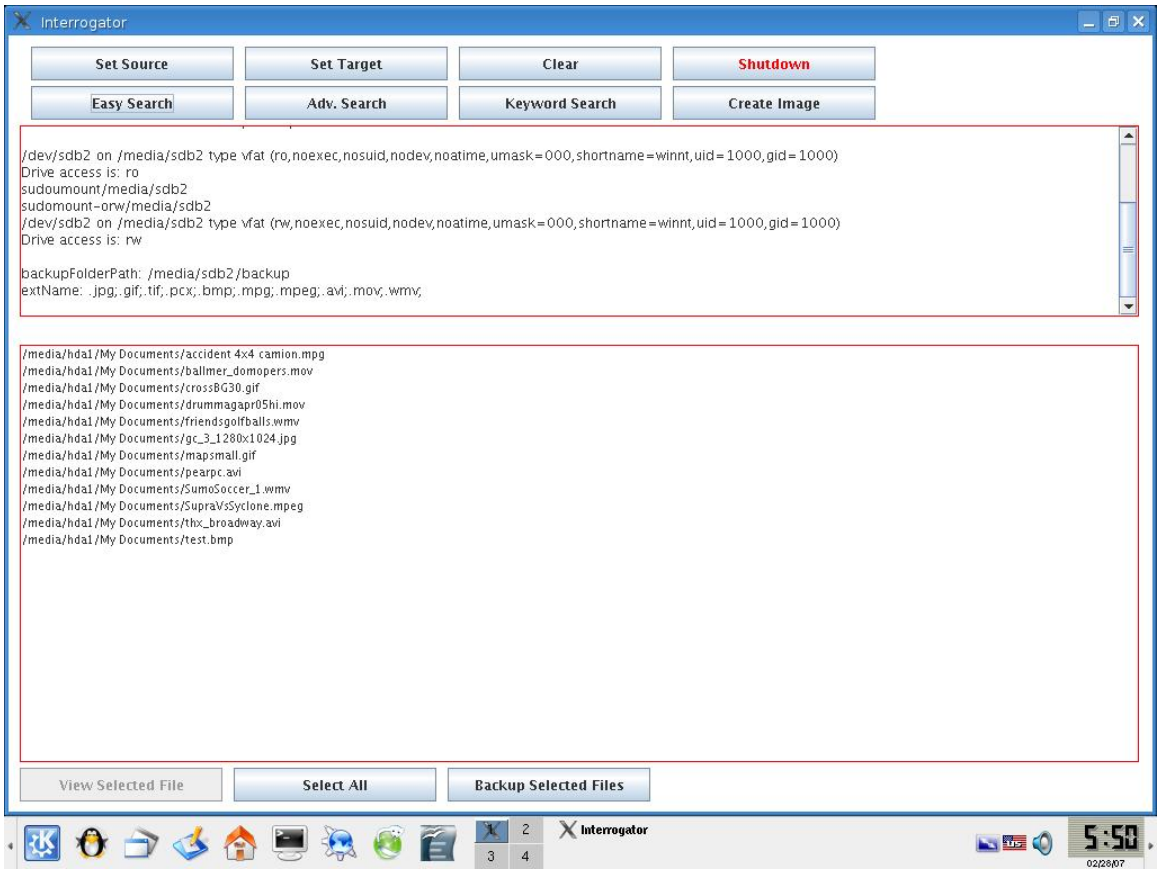

Figure 14. Easy Search Results.

Double-click on any filename to view its contents. If the filename ends with .pst it is an Outlook personal folder file. There are a few additional steps required to view the contents. Outlook 2003 will start after double-clicking the file. Click on the File menu in Outlook, then click Open, then click Outlook Data File. Next, click the My Documents icon. Now you will see the file you want to view. Double click the filename. A new "Personal Folders" item will appear in the mail folders pane. Click the "+" to the left of Personal Folders. Now you can see all the email folders from the source file. Click on a subfolder such as Inbox and then a list of emails will appear in the center window pane. Clicking on an email will show its contents in the right window pane. To view contact information you will need to click the Contacts button on the lower-left

side of the window. In the upper-left you will see Contacts followed by Contacts in Personal Folders. Click Contacts in Personal Folders. Now all contacts in the file will appear on the right side of the screen.

1. Advanced Search

Click the “Adv. Search” button.

The dialog box in Figure 6 will appear allowing you to enter the file extension of a specific file type. This is useful if you know that you only want to look for one file type or if you need to search for a file type not listed in the Easy Search. For example, if you only wanted to see jpeg type images, you would enter “.jpg” (without quotes) in the dialog box and only search for those files with a .jpg extension. If you want to search for text files and avi videos you would enter “.txt;.avi” (without quotes). It is important to separate multiple extensions with a semicolon.

Figure 15. Advanced Search Dialog Box.

All files of the specified extension will appear in the results window.

Double-click on any file to view its contents.

2. Keyword Search

The Keyword Search function will search for keywords in the contents of files.

Click the “Keyword Search” button.

A dialog box will appear as shown in Figure 7.

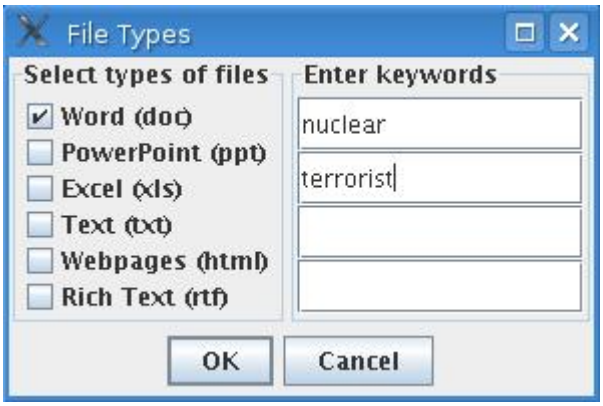

Figure 16. Keyword Search options dialog box.

On the left side, select the file types to be included in the search.

On the right side, enter the keywords you want to find.

Click ok.

All files of the selected types containing any of the entered keywords will appear in the results window.

Double-click on any file to view its contents.

H. CREATING IMAGES

The Create Image function will create an exact copy of the desired drive (bit-by-bit) and store it in a file on the target drive. The resulting file will be the same size as the imaged drive.

Click the “Create Image” button.

Select the drive you would like to copy.

Enter a filename for the image as in Figure 8.

Figure 17. Naming the Image.

The file will be stored in the folder entered during the "Set Target" function. Additionally, an MD5 hash of the image will be stored in the same folder with the extension .md5.

Creating an image is the most time consuming process of all of Interrogator's features. The duration is dependent on the speed of the source and target drives as well as the size of the drive that is being imaged.

Note: Most modern hard disks are many gigabytes (GB) in size. If your target drive is formatted with FAT32 you will not be able to image a drive larger than 4GB. NTFS or EXT3 are the recommended file systems for imaging hard disks.

I. SHUTTING DOWN

When you are done analyzing the drive you should properly shutdown the system prior to disconnecting any drives. Click the "Shutdown" button. Click "Yes" and wait until you are notified that the computer is ready to turn off. You will be prompted to remove the CD and hit the enter key to complete the shut down process as shown in Figure 9.

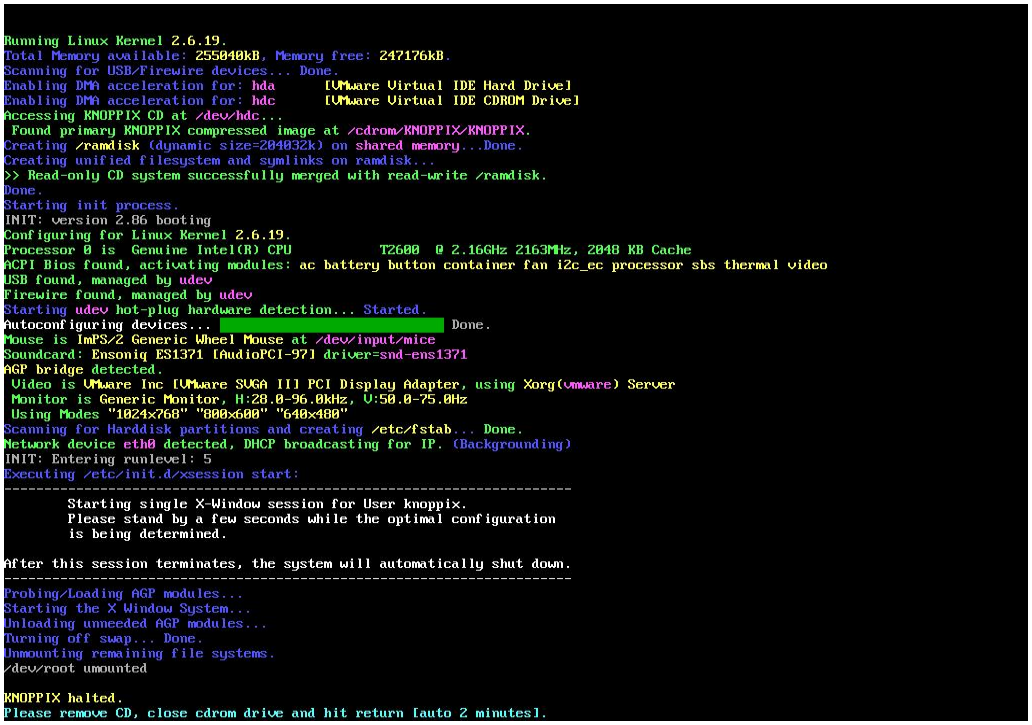

Figure 18. Shutdown Screen.

J. TROUBLESHOOTING

This section will help solve common issues that may arise during the operation of Interrogator KNOPPIX.

1. Computer fails to start Interrogator KNOPPIX.

Different computers require different methods to boot from a CD-ROM disc. Pay careful attention to messages that appear on the screen as the computer is starting. There should be a message stating to press a certain key for the boot menu. This key may be “esc” or “F12.” Press the indicated key and select CD-ROM.

Some older computers may not provide an option to select a boot device on startup. In this situation you must enter the BIOS and change the boot order. When the computer is powered on there will be a message indicating which key to press to enter

the BIOS or SETUP. Press the indicated key and look for an option to specify the boot order. Set the CD-ROM drive as the highest priority.

2. A password is requested when attempting to enter the computer BIOS or SETUP.

Unfortunately, bypassing the BIOS password generally requires opening the computer case and is beyond the scope of this manual. In this situation it is best to use another computer for running Interrogator. If the drive to be analyzed is inside the password protected computer you must physically remove the hard disk and attach it to another computer via the included USB to IDE/SATA adapter.

3. A “busy, please wait” message has been on the screen for a long time.

Analyzing a drive can be very time consuming especially on large drives. Most computers have a hard drive indicator light. The light will appear solid or blink when the drive is being accessed. If the light is on please allow more time for the process to complete. If the hard drive light has not lit for a couple minutes it is possible that the computer has locked up. In this case, restart the computer and retry the operation.

4. The drive you want to analyze is not in the list.

There are a few possible causes. It is possible the drive is formatted with an unsupported file system or none at all. All common file systems are supported by Interrogator KNOPPIX including FAT16, FAT32, NTFS, EXT2, EXT3, and ReiserFS.

Another possibility is that the hard drive is damaged and not recognized by the computer. If this drive is the main system drive you can verify it is good by attempting to start

the computer without Interrogator KNOPPIX. If the computer boots into anything other than Interrogator KNOPPIX it is most definitely not damaged.

5. You see an error that you do not have write access when attempting to backup files or create an image.

After clicking ok you will be asked if you want to enable write access. If that does not work, the drive may be physically locked. Some USB thumb drives or flash memory have a switch to lock the drive. Please verify the drive is unlocked.

6. The computer fails to boot from the USB drive.

There is a workaround if you would like to boot from USB for enhanced performance but the computer is only capable of booting from CD. First, you need to insert both the bootable Interrogator CD and the bootable USB drive. Enter the boot menu after the computer is powered on. Select CD-ROM. Once the initial Interrogator KNOPPIX screen appears you can enter the following command to redirect the boot to the USB drive:

knoppix bootfrom=/dev/sda1

where sda1 is the USB drive. The USB drive may not be sda1 on all systems. All USB and SATA drives will have the sd prefix. If the computer has a single SATA hard drive, the USB drive will be sdb1. If there are two SATA hard drives, the USB drive will be sdc1.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. REMASTERING INTERROGATOR KNOPPIX

This appendix is intended for personnel who would like to customize Interrogator KNOPPIX to meet specific needs.

A. INTRODUCTION

Remastering a Linux live CD is not a trivial task. There are a few guides to remastering KNOPPIX available on the Internet. Unfortunately, they are outdated and contain some incorrect information. This guide was developed with information from “Building Your Own Live CD” [Ref. 45] and “Knoppix Remastering Howto” [Ref. 46].

A moderate level of Linux experience is required to follow the instructions below. This guide will use VMWare Server [Ref. 47] and KNOPPIX 5.1.1 Live CD [Ref. 48]. Both products are available free of charge online.

B. SYSTEM REQUIREMENTS

Remastering requires a large amount of RAM and disk space. The minimum requirements are as follows:

1 GB RAM

4 GB available disk space on a Linux file system

C. STEP-BY-STEP REMASTERING

1. Preparation

Download VMWare Server from http://vmware.com

Install VMWare Server

Download KNOPPIX 5.1.1 Live CD from http://knoppix.com

Create a new virtual machine with 4GB SCSI hard disk

Configure the CD-ROM to use the KNOPPIX disc

Start the VM

Open the command line

Enter super user mode

su

Create a partition using the following commands:

fdisk /dev/sda1

n

p

1

w

Format the partition

mkfs.ext3 /dev/sda1

Mount the partition with read/write access

mount –t ext3 /dev/sda1 /media/sda1 –o rw

Create working directories

mkdir /media/sda1/source

mkdir /media/sda1/master

Copy files necessary for remastering

cp –Rp /cdrom/* /media/sda1/master

cp –Rp /KNOPPIX/* /media/sda1/source

rm /media/sda1/master/KNOPPIX/KNOPPIX

2. Changing Startup Graphic and Background

The startup graphic is a low resolution image that appears when the computer begins to boot from the KNOPPIX disc. Create

a 640 x 400 pixel image in the image editor of your choice. You must save it as a 16-color GIF image. The next commands will assume you named the file logo.gif and will convert the image to lss16 format.

giftopnm < logo.gif > logo.ppm

ppmtolss16 < logo.ppm > logo.16

Next, you need to save the file in the proper directory and overwrite the existing startup graphic.

cp logo.16 /media/sda1/master/boot/isolinux/

The KDE desktop background can be any JPEG image. 1024 x 768 pixels is the ideal resolution. You need to save the file as background.jpg. Simply copy it to the appropriate folder and overwrite the original background.

cp background.jpg /media/sda1/master/KNOPPIX/

3. Removing Packages

If you need to add more than a couple megabytes of files to the remastered disc you will need to remove some packages. Removing packages requires chrooting into the source folder.

chroot /media/sda1/source

Now you can list the installed packages by size:

***dpkg-query -W --showformat='${Installed-Size} ***
${Package}\n' | sort –n

Remove packages with the command:

apt-get remove --purge name-of-package-to-remove

4. Adding Utilities/Scripts

Any utilities or scripts you want added to the remastered disc should be placed in /media/sda1/usr/bin/

You also need to chmod 755 the executables you add.

chmod 755 readwab.sh

5. Setting File Associations

This will show how to associate files with the extension *.wab to the script readwab.sh.

Create the mime link:

pico \ etc/skel/.kde/share/mimelnk/all/addressBook.desktop

Type the following:

[Desktop Entry]

Comment=

Hidden=false

Icon=

MimeType=all/addressBook

Patterns=*.wab

Type=MimeType

Save and exit.

Create the application link:

pico \ etc/skel/.kde/share/applnk/.hidden/readwab.sh.desktop

Type the following:

[Desktop Entry]

Exec=/usr/bin/readwab.sh

InitialPreference=2

MimeType=all/addressBook

Name=readwab.sh

Terminal=false

Type=Application

Save and exit.

Edit the user profile:

pico etc/skel/.kde/share/config/profilerc

Add the following at the top:

[all/addressBook - 1]

AllowAsDefault=true

Application=kde-readwab.sh.desktop

GenericServiceType=Application

Preference=1

ServiceType=all/addressBook

Save and exit.

6. Final Remastering Steps

Change to the master folder

cd /media/sda1/master

Create compressed filesystem

mkisofs –L –R –l –V “KNOPPIX ISO9660” –v ****

–allow-multidot /media/sda1/source **| **

***create-compressed_fs – 65536 > ***

/media/sda1/master/KNOPPIX/KNOPPIX

Create ISO image

***mkisofs –pad –l –r –J –v –V “Interrogator” ***

***-no-emul-boot –boot-load-size 4 ***

***–boot-info-table –b boot/isolinux/isolinux.bin ***

***-c boot/isolinux/boot.cat –hide-rr-moved ***

-o Interrogator.iso /media/sda1/master/

Your remastered KNOPPIX is located at

/media/sda1/master/Interrogator.iso

The last step is to burn the ISO image to a disc using your favorite CD burning software. Nero [Ref. 49] and Roxio [Ref. 50] are good choices. Optionally, Interrogator can be installed on a USB flash drive or hard drive in a few steps. The USB device must have at least 700 MB of storage and needs to be formatted with FAT16 or FAT32. First, boot a computer with the Interrogator CD and ensure the USB drive is connected. Once the system is fully loaded, open a terminal window. Enter super user mode:

su

Then start the USB install utility:

mkbootdev

Select the USB drive from the list. If the drive already contains a FAT16 or FAT32 partition select “don’t change partitions on USB storage.” If there is no FAT partition or you want to erase the existing data, choose “Create system on USB storage using a FAT32 partition.”

APPENDIX C. SOURCE CODE

All source code written for Interrogator KNOPPIX is included below. This includes Java files for the GUI and Bash shell scripts.

A. INTERROGATOR.JAVA

```
/usr/bin/Interrogator/Interrogator.java

/*
 Interrogator version 0.4.8 2/27/07
 Adrian Arvizo
 This is a GUI for forensic utilities.
 File: Interrogator.java
 */

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.*;
import javax.swing.event.*;

/**
 * Interrogator class
 *
 * <p>
 * This is the top-level class for the Interrogator project.
 */
class Interrogator extends JFrame implements ActionListener,
ListSelectionListener {

        // ---------------------------------
        // Data Members
        // ---------------------------------

        /**
         * X coordinate of the frame default origin point
         */
        private static final int FRAME_X_ORIGIN = 0;

        /**
         * Y coordinate of the frame default origin point
         */
        private static final int FRAME_Y_ORIGIN = 0;

        /**
         * Default width for buttons
         */
        private static final int BUTTON_WIDTH = 180;

        /**
         * Default height for buttons
         */
        private static final int BUTTON_HEIGHT = 30;

        /**
         * Constant for platform specific newline
         */
        private static final String NEWLINE = System.getProperty("line.separator");

        /**
```

```
 * Constant for empty string
 */
private static final String EMPTY_STRING = "";

/**
 * Interface Buttons
 */
private JButton setSourceButton, setTargetButton,
shutdownButton, clearButton, backupButton, advSearchButton,
easySearchButton, createImageButton, keywordSearchButton,
viewButton, selectAllButton;

/**
 * Upper output window
 */
private JTextArea textArea;

/**
 * List to show resulting filenames from searches
 */
private LinkedList docList;

/**
 * Extensions to search for
 */
private String extName;

/**
 * Source and target locations
 */
private File sourcePath, targetPath;

/**
 * Location of backup folder
 */
private String backupFolderPath = "";

/**
 * List to store partitions on system
 */
private LinkedList drives;

/**
 * Window that displays to indicate the program is busy
 */
private ProgressIndicator progressIndicator;

/**
 * List to display resulting filenames in GUI
 */
private JList list;

/**
 * Model to specify JList list
 */
private DefaultListModel listModel;

// ---------------------------------
// Main method
// ---------------------------------
public static void main(String[] args) {
        Interrogator frame = new Interrogator();
        frame.setVisible(true);

}

// ---------------------------------
// Constructors
// ---------------------------------

/**
 * Default constructor
 */
public Interrogator() {
        final Container contentPane;

        //set the frame properties
        setTitle("Interrogator");
        setLocation(FRAME_X_ORIGIN, FRAME_Y_ORIGIN);
        setSize(java.awt.Toolkit.getDefaultToolkit().getScreenSize());
        contentPane = getContentPane();
```

```
contentPane.setBackground(Color.white);
contentPane.setLayout(null);

//create the progress indicator window
progressIndicator = new ProgressIndicator(null, true);

//create and place buttons on the content pane
setSourceButton = new JButton("Set Source");
setSourceButton.setBounds(20, 10, BUTTON_WIDTH, BUTTON_HEIGHT);
setSourceButton
.setToolTipText("Click to choose the drive you wish to analyze");

setTargetButton = new JButton("Set Target");
setTargetButton.setBounds(210, 10, BUTTON_WIDTH, BUTTON_HEIGHT);
setTargetButton
.setToolTipText("Click to choose the drive to store collected data");

shutdownButton = new JButton("Shutdown");
shutdownButton.setBounds(590, 10, BUTTON_WIDTH, BUTTON_HEIGHT);
shutdownButton.setForeground(Color.red);
shutdownButton.setToolTipText("Click to shutdown the system");

clearButton = new JButton("Clear");
clearButton.setBounds(400, 10, BUTTON_WIDTH, BUTTON_HEIGHT);
clearButton.setToolTipText("Click to clear the output windows");

easySearchButton = new JButton("Easy Search");
easySearchButton.setBounds(20, 45, BUTTON_WIDTH, BUTTON_HEIGHT);
easySearchButton.setToolTipText("Click to search for files");

advSearchButton = new JButton("Adv. Search");
advSearchButton.setBounds(210, 45, BUTTON_WIDTH, BUTTON_HEIGHT);
advSearchButton
.setToolTipText("Click to specify the search extensions");

createImageButton = new JButton("Create Image");
createImageButton.setBounds(590, 45, BUTTON_WIDTH, BUTTON_HEIGHT);
createImageButton.setToolTipText("Click to copy the entire drive");

keywordSearchButton = new JButton("Keyword Search");
keywordSearchButton.setBounds(400, 45, BUTTON_WIDTH, BUTTON_HEIGHT);
keywordSearchButton
.setToolTipText("Click to search for keywords within files");

viewButton = new JButton("View Selected File");
viewButton.setToolTipText("Click to view the selected file");
viewButton.setActionCommand("View Selected File");
viewButton.addActionListener(new ViewListener());
viewButton.setEnabled(false);

selectAllButton = new JButton("Select All");
selectAllButton.setToolTipText("Click to select all files in list");
selectAllButton.setActionCommand("Select All");
selectAllButton.addActionListener(new SelectAllListener());

backupButton = new JButton("Backup Selected Files");
backupButton
.setToolTipText("Click to backup all selected files in the lower window");

//add the upper buttons
contentPane.add(setSourceButton);
contentPane.add(setTargetButton);
contentPane.add(shutdownButton);
contentPane.add(clearButton);
contentPane.add(advSearchButton);
contentPane.add(easySearchButton);
contentPane.add(createImageButton);
contentPane.add(keywordSearchButton);

//register this frame as an action listener of the buttons
setSourceButton.addActionListener(this);
setTargetButton.addActionListener(this);
shutdownButton.addActionListener(this);
clearButton.addActionListener(this);
backupButton.addActionListener(this);
advSearchButton.addActionListener(this);
easySearchButton.addActionListener(this);
createImageButton.addActionListener(this);
keywordSearchButton.addActionListener(this);

//Create the upper status window and place it on the screen
```

```
                textArea = new JTextArea();
                textArea.setToolTipText("This is the status window");
                textArea.setEditable(false);
                final JScrollPane scrollText = new JScrollPane(textArea);
                scrollText.setBorder(BorderFactory.createLineBorder(Color.red));
                contentPane.add(scrollText);

                //Create the list and put it in a scroll pane.
                listModel = new DefaultListModel();
                list = new JList(listModel);
                list.setFont(new Font("Dialog", Font.PLAIN, 10));
                list.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
                list.setSelectedIndex(0);
                list.addListSelectionListener(this);
                list.setVisibleRowCount(5);
                list.setToolTipText("Double-click any file to open");
                final JScrollPane listScrollPane = new JScrollPane(list);
                listScrollPane.setBorder(BorderFactory.createLineBorder(Color.red));

                //Opens the file when the file is double-clicked.
                list.addMouseListener(new MouseAdapter() {
                        public void mouseClicked(MouseEvent evt) {
                                if (evt.getClickCount() == 2) {
                                        openFile();
                                }
                        }
                });

                //Add the lower output window and buttons to the screen
                contentPane.add(listScrollPane);
                contentPane.add(viewButton);
                contentPane.add(selectAllButton);
                contentPane.add(backupButton);

                //Check if the window is resized and arrange components appropriately
                this.addComponentListener(new ComponentAdapter() {
                        public void componentResized(ComponentEvent e) {
                                scrollText.setBounds(10, 80, getWidth() - 30, 170);
                                listScrollPane.setBounds(10, 275, getWidth() - 30,
                                                getHeight() - 350);
                                listScrollPane.repaint();
                                textArea.updateUI();
                                list.updateUI();
                                viewButton.setBounds(10, getHeight() - 70, BUTTON_WIDTH,
                                                BUTTON_HEIGHT);
                                selectAllButton.setBounds(200, getHeight() - 70, BUTTON_WIDTH,
                                                BUTTON_HEIGHT);
                                backupButton.setBounds(390, getHeight() - 70, BUTTON_WIDTH,
                                                BUTTON_HEIGHT);
                        }
                });

                //register 'Exit upon closing' as a default close operation
                setDefaultCloseOperation(EXIT_ON_CLOSE);
        }

//      ------------------------------------------------
//      Public Methods:
//
//      void    actionPerformed  ( ActionEvent )
//
//  void valueChanged     ( ListSelectionEvent)
//
//      ------------------------------------------------

        /**
         * Standard method to respond the action event.
         *
         * @param event
         *            the ActionEvent object
         *
         */
        public void actionPerformed(ActionEvent event) {

                if (event.getSource() instanceof JButton) {
                        JButton clickedButton = (JButton) event.getSource();

                        if (clickedButton == setSourceButton) {
                                setSource();
                        } else if (clickedButton == setTargetButton) {
                                setTarget();
```

```
                        } else if (clickedButton == shutdownButton) {
                                shutdown();
                        } else if (clickedButton == clearButton) {
                                clearText();
                        } else if (clickedButton == backupButton) {
                                copyFiles();
                        } else if (clickedButton == advSearchButton) {
                                advancedSearch();
                        } else if (clickedButton == easySearchButton) {
                                easySearch();
                        } else if (clickedButton == createImageButton) {
                                createImage();
                        } else if (clickedButton == keywordSearchButton) {
                                keywordSearch();
                        } else {
                                clearText();
                        }

                        //scroll the upper output window to the bottom
                        textArea.setCaretPosition(textArea.getDocument().getLength());

                }
        }

        /**
         * This method is required by ListSelectionListener.
         *
         * @param ListSelectionListener.
         *
         */
        public void valueChanged(ListSelectionEvent e) {
                if (e.getValueIsAdjusting() == false) {
                        if (list.getSelectedIndex() == -1) {
                                //No selection, disable view button.
                                viewButton.setEnabled(false);
                        } else {
                                //Selection, enable the view button.
                                viewButton.setEnabled(true);
                        }
                }
        }

//      ------------------------------------------------
//      Private Methods:
//
//      void    addToList                       ( String )
//      void    advancedSearch          ( )
//      void    clearText                       ( )
//      void    copyFiles                       ( )
//      void    createImage                     ( )
//      void    easySearch                      ( )
//      void    keywordSearch           ( )
//      void    mountPartition          ( String )
//      void    openFile                ( )
//      void    resetList                       ( )
//      void    setSource                       ( )
//      void    setTarget                       ( )
//      void    setWriteAccess          ( String )
//      void    shutdown                ( )
//
//      int []  selectedIndices         ( )
//
//      File    setDrive                ( String )
//
//      String  checkDriveAccess ( String )
//      String  valueOf                         ( int )
//
//      ------------------------------------------------

        /**
         * Adds an item to end of the list.
         *
         * @param newItem item to be added to list.
         *
         */
        private void addToList(String newItem) {
                listModel.addElement(newItem);
        }

        /**
         * Performs an advanced file search by extensions
```

```
 * using native java File code
 *
 */
private void advancedSearch() {
        if (sourcePath == null) {
                //source not yet selected, display warning
                JOptionPane.showMessageDialog(null, "Error: no source selected!",
                                "Error: no source selected!", JOptionPane.ERROR_MESSAGE);
        } else {
                //source is selected
                //initialize the list of found files
                docList = new LinkedList();
                //ask the user which extensions to search for
                extName = JOptionPane.showInputDialog("Extension name");
                if (extName != null) {
                        //user has entered extensions, perform search
                        progressIndicator.setVisible(true);
                        visitAllFiles(sourcePath);
                        File docTemp;
                        //add all resulting files to the lower output window
                        for (int i = 0; i < docList.size(); i++) {
                                docTemp = (File) docList.get(i);
                                addToList(docTemp.getAbsolutePath());
                        }
                }
                try {
                        Thread.sleep(100);
                } catch (InterruptedException e) {
                }
                progressIndicator.setVisible(false);
        }
}

/**
 * Clears both output windows
 *
 */
private void clearText() {
        textArea.setText(EMPTY_STRING);
        resetList();
}

/**
 * Copies files to the user selected target
 *
 */
private void copyFiles() {
        if (backupFolderPath.equals(""))
                //target has not been selected, display warning
                JOptionPane.showMessageDialog(null,
                "Error: You have not selected a target");
        else {
                //target is selected, copy files to target
                try {
                        String s = null;
                        Process p = null;
                        //determine which files are selected in the file list
                        int[] selectedIndices = selectedIndices();
                        for (int i = 0; i < selectedIndices.length; i++) {
                                String[] copyCommand = { "cp", "--parents",
                                                valueOf(selectedIndices[i]), backupFolderPath };
                                textArea.append(copyCommand[0] +
                                                                        copyCommand[1] +
                                                                        copyCommand[2] +
                                                                        copyCommand[3] + NEWLINE);
                                progressIndicator.setVisible(true);
                                p = Runtime.getRuntime().exec(copyCommand);
                                p.waitFor();
                                try {
                                        Thread.sleep(100);
                                } catch (InterruptedException e) {
                                }
                                progressIndicator.setVisible(false);
                        }
                        BufferedReader stdInput = new BufferedReader(
                                        new InputStreamReader(p.getInputStream()));

                        BufferedReader stdError = new BufferedReader(
                                        new InputStreamReader(p.getErrorStream()));
                        while ((s = stdInput.readLine()) != null) {
                                textArea.append(s + NEWLINE);
```

```
                        }
                        while ((s = stdError.readLine()) != null) {
                                textArea.append(s + NEWLINE);
                        }
                } catch (Exception ex) {
                        ex.printStackTrace();
                        return;
                }
        }
}

/**
 * Creates a forensically sound image of a partition
 *
 */
private void createImage() {
        File sourcePartition = setDrive("partition to image");
        if (sourcePartition != null) {
                //source partition is selected, continue creating image
                if (!backupFolderPath.equals("")) {
                        //backup folder has been entered, continue creating image
                        String sourcePartitionPath = sourcePartition.getAbsolutePath();
                        textArea
                        .append("source path: " + sourcePartitionPath + NEWLINE);
                        String imageName = JOptionPane
                        .showInputDialog("Name the image to be stored on: "
                                        + backupFolderPath
                                        + "\nThe .dd extension will be added automatically");
                        String imageFullPath = backupFolderPath.concat("/" + imageName
                                        + ".dd");
                        String imageHashFullPath = backupFolderPath.concat("/"
                                        + imageName + ".md5");
                        textArea.append(imageFullPath + NEWLINE);
                        textArea.append(imageHashFullPath + NEWLINE);

                        if (imageName != null) {
                                //image name has been entered, continue creating image
                                try {
                                        String s = null;
                                        Process p = null;
                                        String[] imageCommand = { "dcfldd.sh",
                                                        sourcePartitionPath, imageFullPath,
                                                        imageHashFullPath };
                                        textArea.append(imageCommand[0] +
                                                                                imageCommand[1] +
                                                                                imageCommand[2] +
                                                                                imageCommand[3] + NEWLINE);

                                        progressIndicator.setVisible(true);

                                        p = Runtime.getRuntime().exec(imageCommand);
                                        p.waitFor();
                                        try {
                                                Thread.sleep(100);
                                        } catch (InterruptedException e) {
                                        }
                                        progressIndicator.setVisible(false);
                                        BufferedReader stdInput = new BufferedReader(
                                                new InputStreamReader(p.getInputStream()));
                                        BufferedReader stdError = new BufferedReader(
                                                new InputStreamReader(p.getErrorStream()));
                                        while ((s = stdInput.readLine()) != null) {
                                                textArea.append(s + NEWLINE);
                                        }
                                        while ((s = stdError.readLine()) != null) {
                                                textArea.append(s + NEWLINE);
                                        }
                                } catch (Exception ex) {
                                        ex.printStackTrace();
                                        return;
                                }
                        } else {
                                //image name not provided, display error
                                JOptionPane.showMessageDialog(null,
                                                "Error: no name provided!\nImage not created.",
                                                "Error: no name provided!",
                                                JOptionPane.ERROR_MESSAGE);
                        }
                } else {
                        //backup folder has not been entered, display error
                        JOptionPane.showMessageDialog(
```

```
                                        null,
                                        "Error: no target selected!\nImage not created.",
                                        "Error: no target selected!",
                                        JOptionPane.ERROR_MESSAGE);
                }
        }
}


/**
 * Performs an easy file search by extensions
 * using native java File code
 *
 */
private void easySearch() {
        if (sourcePath == null) {
                //source not selected, display error
                JOptionPane.showMessageDialog(null, "Error: no source selected!",
                                "Error: no source selected!", JOptionPane.ERROR_MESSAGE);
        } else {
                //source selected, continue easySearch
                InterrogatorSearchOptions searchOptions =
                                new InterrogatorSearchOptions(this, true);
                searchOptions.setVisible(true);
                docList = new LinkedList();

                extName = searchOptions.getSelectedExtensions();
                if (extName != null && !extName.equals("")) {
                        //user has entered extensions and it's not blank
                        progressIndicator.setVisible(true);
                        textArea.append("extName: " + extName + NEWLINE);
                        visitAllFiles(sourcePath);
                        File docTemp;
                        for (int i = 0; i < docList.size(); i++) {
                                docTemp = (File) docList.get(i);
                                addToList(docTemp.getAbsolutePath());
                        }
                }
                try {
                        Thread.sleep(100);
                } catch (InterruptedException e) {
                }
                progressIndicator.setVisible(false);
        }
}

/**
 * Allows the user to search for keywords in files
 *
 */
// Search for keywords in files
private void keywordSearch() {
        if (sourcePath == null) {
                //source not set, display error
                JOptionPane.showMessageDialog(null, "Error: no source selected!",
                                "Error: no source selected!", JOptionPane.ERROR_MESSAGE);
        } else {
                //source is set, continue keywordSearch
                KeywordSearchOptions keywordSearchOptions =
                                                new KeywordSearchOptions(this, true);
                keywordSearchOptions.setVisible(true);
                extName = "";
                extName = keywordSearchOptions.getSelectedExtensions();
                String keywords = "";
                keywords = keywordSearchOptions.getKeywords();

                if (!keywords.equals("") && !extName.equals("")) {
                        //user has entered keywords and selected extensions
                        progressIndicator.setVisible(true);
                        textArea.append("extName: " + extName + NEWLINE);
                        textArea.append("keywords: " + keywords + NEWLINE);

                        try {
                                String s = null;
                                Process p = null;
                                String[] keywordSearchCommand = { "keywordSearch.sh",
                                        sourcePath.getAbsolutePath(), extName, keywords };
                                for (int i = 0; i < keywordSearchCommand.length; i++) {
                                        textArea.append(keywordSearchCommand[i] + " ");
                                }
                                textArea.append(NEWLINE);
```

```
                                progressIndicator.setVisible(true);
                                p = Runtime.getRuntime().exec(keywordSearchCommand);
                                p.waitFor();
                                try { Thread.sleep(100);
                                } catch (InterruptedException e) {
                                }
                                progressIndicator.setVisible(false);
                                BufferedReader stdInput = new BufferedReader(
                                                new InputStreamReader(p.getInputStream()));
                                BufferedReader stdError = new BufferedReader(
                                                new InputStreamReader(p.getErrorStream()));
                                while ((s = stdInput.readLine()) != null) {
                                        textArea.append(s + NEWLINE);
                                        addToList(s);
                                }
                                while ((s = stdError.readLine()) != null) {
                                        textArea.append(s + NEWLINE);
                                }
                        } catch (Exception ex) {
                                ex.printStackTrace();
                                return;
                        }
                }
                try { Thread.sleep(100);
                } catch (InterruptedException e) {
                }
                progressIndicator.setVisible(false);
        }
}

/**
 * Changes a mount from read-only to write access
 *
 * @param targetPartition the partition to mount
 *
 */
private void mountPartition(String targetPartition) {
        try {
                String s = null;
                Process p = null;
                //create the mount command
                String[] mountCommand = { "sudo", "mount", targetPartition, "-o",
                "ro" };
                textArea.append(mountCommand[0] +
                                                mountCommand[1] +
                                                mountCommand[2] +
                                                mountCommand[3] +
                                                mountCommand[4] + NEWLINE);
                progressIndicator.setVisible(true);
                p = Runtime.getRuntime().exec(mountCommand);
                p.waitFor();
                try {
                        Thread.sleep(100);
                } catch (InterruptedException e) {
                }
                progressIndicator.setVisible(false);
                BufferedReader stdInput = new BufferedReader(new InputStreamReader(
                                p.getInputStream()));
                BufferedReader stdError = new BufferedReader(new InputStreamReader(
                                p.getErrorStream()));
                while ((s = stdInput.readLine()) != null) {
                        textArea.append(s + NEWLINE);
                }
                while ((s = stdError.readLine()) != null) {
                        textArea.append(s + NEWLINE);
                }
        } catch (Exception ex) {
                ex.printStackTrace();
                return;
        }
}

/**
 * Opens the file in KDE's registered viewer application.
 *
 */
private void openFile() {
        String pathTemp = (String) list.getSelectedValue();
        try {
                Process p = null;
                String[] buttonFourCommand = { "kfmclient", "exec", pathTemp };
```

```
                p = Runtime.getRuntime().exec(buttonFourCommand);
        } catch (IOException e2) {
                System.out.println("exception happened - here's what I know: ");
                e2.printStackTrace();
                System.exit(-1);
        }
}

/**
 * Clears the list.
 *
 */
private void resetList() {
        listModel.removeAllElements();
}

/**
 * Allows the user to set the source drive to analyze
 *
 */
private void setSource() {
        File sourcePathTemp = sourcePath;
        sourcePath = setDrive("drive to scan");
        if (sourcePath == null)
                //source not selected, set source to previous source
                sourcePath = sourcePathTemp;
}

/**
 * Allows the user to set the target drive
 *
 */
private void setTarget() {
        if (sourcePath == null) {
                //source not selected, display error
                JOptionPane.showMessageDialog(null, "Error: Set the source first!",
                                "Error: Set the source first!", JOptionPane.ERROR_MESSAGE);
        } else {
                //source selected, continue setTarget
                targetPath = setDrive("drive to save collected data");
                if (targetPath != null    && sourcePath.getAbsolutePath().equals(
                                                targetPath.getAbsolutePath())) {
                        //target is selected, but is same as the source, display error
                        JOptionPane.showMessageDialog(null,
                        "Error: You selected the source drive.\nPlease try again.");
                        setTarget();
                } else if (targetPath != null) {
                        //target is selected and is not the source
                        String access = checkDriveAccess(targetPath.getAbsolutePath());
                        if (access.equals("ro")
                                //target is mounted read-only, prompt for write access
                                        && JOptionPane.showConfirmDialog(null,
                                                        "Allow write on drive\n"
                                                        + targetPath.getAbsolutePath(),
                                                        "The target drive is read-only",
                                                        JOptionPane.YES_NO_OPTION) == 0) {
                                setWriteAccess(targetPath.getAbsolutePath());
                        }
                        //check if write access was enabled
                        access = checkDriveAccess(targetPath.getAbsolutePath());
                        if (access.equals("rw")) {
                                //write access is enabled
                                String backupFolder = JOptionPane
                                .showInputDialog("Backup name");
                                if (backupFolder == null) {
                                        //backup folder was not entered
                                        backupFolderPath = "";
                                } else {
                                        //backup folder was entered, continue setTarget
                                        backupFolderPath = targetPath.getAbsolutePath() + "/"
                                        + backupFolder;
                                        textArea.append("\nbackupFolderPath: "
                                                        + backupFolderPath + "\n");
                                        File dir = new File(backupFolderPath);
                                        if (!dir.exists()) {
                                                //user entered folder name
                                                //check if folder was created
                                                boolean success = dir.mkdir();
                                                if (!success) {
                                                        //folder not created, display error
                                                textArea.append("Directory Creation Failed");
```

```
                                                }
                                        }
                                }
                        } else {
                                //write access was not enabled, display error
                                JOptionPane.showMessageDialog(
                                        null,
                                "Error: no write access!\nBackup folder not created.",
                                        "Error: no write access!",
                                        JOptionPane.ERROR_MESSAGE);
                                mountPartition(targetPath.getAbsolutePath());
                        }
                }
        }
}

/**
 * Changes a mount from read-only to write access
 *
 * @param targetPartition the partition to set write access
 *
 */
private void setWriteAccess(String targetPartition) {
        try {
                String s = null;
                Process p = null;
                //unmount drive that needs write access
                String[] umountCommand = { "sudo", "umount", targetPartition, };
                textArea.append(umountCommand[0] +
                                                umountCommand[1] +
                                                umountCommand[2] + NEWLINE);
                //mount drive with write access
                String[] mountCommand = { "sudo", "mount", "-o", "rw",
                                targetPartition, };
                textArea.append(mountCommand[0] +
                                                mountCommand[1] +
                                                mountCommand[2] +
                                                mountCommand[3] +
                                                mountCommand[4] + NEWLINE);
                progressIndicator.setVisible(true);
                p = Runtime.getRuntime().exec(umountCommand);
                p.waitFor();
                p = Runtime.getRuntime().exec(mountCommand);
                p.waitFor();
                try {
                        Thread.sleep(100);
                } catch (InterruptedException e) {
                }
                progressIndicator.setVisible(false);
                BufferedReader stdInput = new BufferedReader(new InputStreamReader(
                                p.getInputStream()));

                BufferedReader stdError = new BufferedReader(new InputStreamReader(
                                p.getErrorStream()));

                while ((s = stdInput.readLine()) != null) {
                        textArea.append(s + NEWLINE);
                }
                while ((s = stdError.readLine()) != null) {
                        textArea.append(s + NEWLINE);
                }
        } catch (Exception ex) {
                ex.printStackTrace();
                return;
        }
}

/**
 * Shuts down the system using the command "shutdown -h now"
 *
 */
private void shutdown() {
        int shutdownPrompt = JOptionPane.showConfirmDialog(null,
                        "Are you sure\n you want to shutdown?", "Shutdown?",
                        JOptionPane.YES_NO_OPTION);
        if (shutdownPrompt == 0) {
                //user selected yes to shutdown, continue shutting down
                try {
                        String s = null;
                        Process p = null;
                        String[] shutdownCommand = { "sudo", "shutdown", "-h", "now" };
```

```
                        textArea.append(shutdownCommand[0] +
                                                shutdownCommand[1] +
                                                shutdownCommand[2] +
                                                shutdownCommand[3] + NEWLINE);
                        p = Runtime.getRuntime().exec(shutdownCommand);
                        BufferedReader stdInput = new BufferedReader(
                                        new InputStreamReader(p.getInputStream()));
                        BufferedReader stdError = new BufferedReader(
                                        new InputStreamReader(p.getErrorStream()));
                        while ((s = stdInput.readLine()) != null) {
                                textArea.append(s + NEWLINE);
                        }
                        while ((s = stdError.readLine()) != null) {
                                textArea.append(s + NEWLINE);
                        }
                } catch (Exception ex) {
                        ex.printStackTrace();
                        return;
                }
        }
}

/**
 * Traverse all files in the given directory
 *
 * @param dir the directory to traverse.
 *
 */
private void visitAllFiles(File dir) {
        if (dir.isDirectory() && dir.canRead()) {
                //is a directory and has read access
                //creates an array of files matching the provided extension
                File[] docList2 = dir.listFiles(new FileFilter(extName));
                if (docList2 != null) {
                        //required to handle NTFS junctions
                        //add all matching files to the master linked list (docList)
                        for (int j = 0; j < docList2.length; j++) {
                                docList.add(docList2[j]);
                        }
                        //create array of files/folders contained in dir
                        String[] children = dir.list();
                        // iterate through all subfolders
                        for (int i = 0; i < children.length; i++) {
                                visitAllFiles(new File(dir, children[i]));
                        }
                }
        }
}

/**
 * Returns an array containing the indices of all selected items.
 *
 */
private int[] selectedIndices() {
        return list.getSelectedIndices();
}

/**
 * Scans the system for mounted partitions,
 * prompts the user to select a partition,
 * and returns the chosen partition
 *
 * @param setDriveText title of choose partition dialog.
 *
 */
private File setDrive(String setDriveText) {
        String s = null;
        File newPath = null;
        drives = new LinkedList();
        try {
                Process p = null;
                p = Runtime.getRuntime().exec(
                                "df -x squashfs -x iso9660 -x tmpfs -x aufs -T -h");

                BufferedReader stdInput = new BufferedReader(new InputStreamReader(
                                p.getInputStream()));

                BufferedReader stdError = new BufferedReader(new InputStreamReader(
                                p.getErrorStream()));
                int driveCounter = 0;
                //list to store partition info
```

```
                LinkedList partitionInfo = new LinkedList();
                String cdSource = "null";
                textArea
                .append(NEWLINE
                        + "Filesystem     Type    Size   Used  Avail  Use% Mounted on"
                        + NEWLINE);
                //display mounted partitions
                while ((s = stdInput.readLine()) != null) {

                        Comparator partitionSort = new PartitionSort();
                        String sTemp = s;
                        if (s.matches(".*cdrom.*")) {
                                //mount is cdrom or KNOPPIX boot partition
                                //parse its mount point
                                cdSource = s.substring(0, 9) + ".*";
                        }
                        if (s.matches(".*Drive.*") || s.matches(".*hd[a-z][1-9].*")
                                        || s.matches(".*sd[a-z][1-9].*")
                                        || s.matches(".*ext.*")) {
                                //mount is a partition useful for Interrogator
                                if (setDriveText.equals("partition to image"))
                                        //user called setDrive from the createImage button
                                        s = s.substring(0, 9);
                                else {
                                        //user called setDrive from setSource or setTarget
                                        // start from the 20th character and grab everything
                                        // from the forward slash to the EOL
                                        s = s.substring(s.indexOf("/", 20), s.length());
                                }
                                if (s.matches(".*cdrom.*") || sTemp.matches(cdSource)) {
                                //mount point contains "cdrom" and is KNOPPIX boot part
                                } else {
                                        //mount is not KNOPPIX boot partition
                                        if (driveCounter > 0) {
                                        //more than one partition is mounted, display this
                                                //textArea.append(sTemp + NEWLINE);
                                                partitionInfo.add(sTemp);
                                        }
                                        //add the partition to the linked list
                                        drives.add(s);
                                        //increment the number of drives
                                        driveCounter++;
                                        //sort the linked list of partition info
                                        //alphabetically
                                        Collections.sort(partitionInfo, partitionSort);
                                }
                        }
                }
                //display info of all partitions on the upper output window
                for(int i = 0; i < partitionInfo.size(); i++)
                        textArea.append((String)partitionInfo.get(i) + NEWLINE);
                //sort the partitions alphabetically
                Collections.sort(drives);
                //scroll upper output window to end of text
                textArea.setCaretPosition(textArea.getDocument().getLength());
                if (driveCounter < 2) {
                        //only KNOPPIX boot partition is mounted
                        JOptionPane.showMessageDialog(null, "Error: no drives found!",
                                        "Error: no drives!", JOptionPane.ERROR_MESSAGE);
                } else {
                        //at least one additional part. is mounted, fill drive array
                        String newDriveArray[] = new String[driveCounter - 1];
                        for (int i = 1; i < drives.size(); i++) {
                                newDriveArray[i - 1] = (String) drives.get(i);
                        }
                        while ((s = stdError.readLine()) != null) {
                                textArea.append(s + NEWLINE);
                        }
                        String newPathTemp = (String) JOptionPane.showInputDialog(null,
                                        "Choose the " + setDriveText, "choose the "
                                        + setDriveText, JOptionPane.PLAIN_MESSAGE,
                                        null, newDriveArray, newDriveArray[0]);
                        if (newPathTemp != null) {
                                //a partition has been selected, set the path
                                newPath = new File(newPathTemp);
                                textArea.append("New " + setDriveText + ": "
                                        + newPath.getAbsolutePath() + NEWLINE + NEWLINE);
                        }
                }
        } catch (IOException e) {
                System.out.println("exception happened - here's what I know: ");
```

```
                e.printStackTrace();
                System.exit(-1);
        }
        return newPath;
}

/**
 * Checks if the partition is mounted read-only or with write access.
 *
 * @param partition partition to check access.
 *
 */
private String checkDriveAccess(String partition) {
        String s = null;
        String driveAccess = "ro";
        try {
                Process p = null;
                p = Runtime.getRuntime().exec("mount");
                BufferedReader stdInput = new BufferedReader(new InputStreamReader(
                                p.getInputStream()));
                BufferedReader stdError = new BufferedReader(new InputStreamReader(
                                p.getErrorStream()));
                while ((s = stdInput.readLine()) != null) {
                        if (s.matches(".*" + partition + ".*")) {
                                textArea.append(s + NEWLINE);
                                int indexTemp = s.indexOf("r");
                                driveAccess = s.substring(indexTemp, indexTemp + 2);
                                textArea.append("Drive access is: " + driveAccess
                                                + NEWLINE);
                        }
                }
                while ((s = stdError.readLine()) != null) {
                        textArea.append(s + NEWLINE);
                }
        } catch (IOException e) {
                System.out.println("exception happened - here's what I know: ");
                e.printStackTrace();
                System.exit(-1);
        }
        return driveAccess;
}

/**
 * Returns an item in the list.
 *
 * @param i the ith item in the list.
 *
 */
private String valueOf(int i) {
        return (String) listModel.get(i);
}

// -------------------------------------------------
// Inner Class: PartitionSort
//
// Sorts the partitions alphabetically.
// -------------------------------------------------

// Inner Class: PartitionSort
class PartitionSort implements Comparator {
        public int compare(Object obj1, Object obj2) {
                //Type-cast objects to Strings.
                String str1 = (String) obj1;
                String str2 = (String) obj2;
                str1 = str1.substring(0,9);
                str2 = str2.substring(0,9);
                System.out.println(str1);
                //compare the strings
                return str1.compareTo(str2);
        }
}

// -------------------------------------------------
// Inner Class: SelectAllListener
//
// Selects all items in the list.
// -------------------------------------------------

// Inner Class: SelectAllListener
class SelectAllListener implements ActionListener {
```

```
                public void actionPerformed(ActionEvent e) {
                        int start = 0;
                        int end = listModel.getSize() - 1;
                        if (end >= 0)
                                list.setSelectionInterval(start, end);
                }
        }

        // ------------------------------------------------
        // Inner Class: ViewListener
        //
        // Opens the file when the button is clicked.
        // ------------------------------------------------

        // Inner Class: ViewListener
        class ViewListener implements ActionListener {

                public void actionPerformed(ActionEvent e) {
                        // This method can be called only if
                        // there's a valid selection
                        // so go ahead and remove whatever's selected.

                        int size = listModel.getSize();
                        if (size == 0) { // List is empty, disable firing.
                                viewButton.setEnabled(false);
                        } else { // Select an index.
                                openFile();
                        }
                }
        }
}
```

B. INTERROGATORSEARCHOPTIONS.JAVA

/usr/bin/Interrogator/InterrogatorSearchOptions.java

```
/*
 InterrogatorSearchOptions
 Adrian Arvizo
 This class creates the easy search dialog box.
 File: InterrogatorSearchOptions.java
 */

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

/**
 *  Ch14JCheckBoxSample1 class
 *
 * <p>
 * A sample frame to illustrate the use of checkbox buttons.
 */
class InterrogatorSearchOptions extends JDialog implements ActionListener {

//----------------------------------
//    Data Members
//----------------------------------

    /**
     * Default frame width
     */
    private static final int FRAME_WIDTH    = 300;

    /**
     * Default frame height
```

```
     */
    private static final int FRAME_HEIGHT   = 300;

    /**
     * X coordinate of the frame default origin point
     */
    private static final int FRAME_X_ORIGIN = 150;

    /**
     * Y coordinate of the frame default origin point
     */
    private static final int FRAME_Y_ORIGIN = 250;

    /**
     * An array of JCheckBox objects
     */
    private JCheckBox[] checkBox;

    /**
        * StringBuffers for easy manipulation of the extensions
        */
    private StringBuffer selectedExtensions;

//----------------------------------
//    Constructors
//----------------------------------

    /**
     * Default constructor
     */
    public InterrogatorSearchOptions(JFrame frame, boolean modal) {
       super(frame, modal);
        Container   contentPane;
        JPanel      checkPanel, okPanel;

        JButton     okButton, cancelButton;

        //create and array of buttons with the following titles
        String[] btnText = { "Office Documents", "Images", "Audio", "Videos",
                             "Webpages",     "Windows     Address     Book",     "Outlook
contacts/email",
                             "Bookmarks", "Browser History", "Virtual Machines" };

        // set the frame properties
        setSize      (FRAME_WIDTH, FRAME_HEIGHT);
        setTitle     ("File Types");
        setLocation  (FRAME_X_ORIGIN, FRAME_Y_ORIGIN);

        contentPane = getContentPane( );
        contentPane.setBackground(Color.white);
        contentPane.setLayout(new BorderLayout());

        //create and place the checkboxes
        checkPanel = new JPanel(new GridLayout(0,1));
        checkPanel.setBorder(BorderFactory
                             .createTitledBorder("Select types of files"));
        checkBox = new JCheckBox[btnText.length];
        for (int i = 0; i < checkBox.length; i++) {
            checkBox[i] = new JCheckBox(btnText[i]);
            checkPanel.add(checkBox[i]);
        }

        //create the tooltips for all checkboxes
```

```
        checkBox[0].setToolTipText("doc, xls, ppt, pdf");
        checkBox[1].setToolTipText("jpg, gif, tif, pcx, bmp");
        checkBox[2].setToolTipText("mp3, wav, wma, mid, aac");
        checkBox[3].setToolTipText("mpg, mpeg, avi, mov, wmv");
        checkBox[4].setToolTipText("htm, html");
        checkBox[5].setToolTipText("wab");
        checkBox[6].setToolTipText("pst");
        checkBox[7].setToolTipText("bookmarks.html");
        checkBox[8].setToolTipText("index.dat, history.dat");
        checkBox[9].setToolTipText("vmdk, vmsd, vmx, vmem, vmss");

        //create and place the OK button and enable ok by enter key
        okPanel = new JPanel(new FlowLayout());
        okButton = new JButton("OK");
        okButton.addActionListener(this);
        okPanel.add(okButton);
        getRootPane().setDefaultButton(okButton);

        //create and place the cancel button and enable cancel by escape key
        cancelButton = new JButton("Cancel");
        cancelButton.addActionListener(new ActionListener() {
             public void actionPerformed (ActionEvent event) {
                    closeDialog();
             }
        });
        okPanel.add(cancelButton);
        addCancelByEscapeKey();

        contentPane.add(checkPanel, BorderLayout.CENTER);
        contentPane.add(okPanel, BorderLayout.SOUTH);

        //register 'Exit upon closing' as a default close operation
        setDefaultCloseOperation( EXIT_ON_CLOSE );
    }

//    ------------------------------------------------
//    Public Methods:
//
//    void      actionPerformed        ( ActionEvent )
//
//  String   getSelectedExtensions     ( )
//
//    ------------------------------------------------

       /**
        * Determines which file types are selected when the user clicks ok.
        *
        * @param event
        *
        */
    public void actionPerformed(ActionEvent event) {

        //create an empty StringBuffer
       selectedExtensions = new StringBuffer("");

       //for each checkBox that is selected, append extensions to the buffer
        if (checkBox[0].isSelected())
             selectedExtensions.append(".doc;.xls;.ppt;.pdf;");
        if (checkBox[1].isSelected())
             selectedExtensions.append(".jpg;.gif;.tif;.pcx;.bmp;");
        if (checkBox[2].isSelected())
             selectedExtensions.append(".mp3;.wav;.wma;.mid;.aac;");
        if (checkBox[3].isSelected())
```

```
            selectedExtensions.append(".mpg;.mpeg;.avi;.mov;.wmv;");
        if (checkBox[4].isSelected())
            selectedExtensions.append(".html;.htm;");
        if (checkBox[5].isSelected())
            selectedExtensions.append(".wab;");
        if (checkBox[6].isSelected())
            selectedExtensions.append(".pst;");
        if (checkBox[7].isSelected())
            selectedExtensions.append("bookmarks.html;");
        if (checkBox[8].isSelected())
            selectedExtensions.append("index.dat;history.dat;");
        if (checkBox[9].isSelected())
            selectedExtensions.append(".vmdk;.vmsd;.vmx;.vmem;.vmss;");

        //close the dialog box when done
        this.setVisible(false);
    }

    /**
        * Returns a String of the entered keywords.
        *
        */
    public String getSelectedExtensions () {
       if (selectedExtensions != null)
            return selectedExtensions.toString();
       else return null;
    }

//    ------------------------------------------------
//    Private Methods:
//
//    void addCancelByEscapeKey (  )
//
//    void   closeDialog  ( )
//
//    ------------------------------------------------

       /**
        * Enables the escape key to perform a cancel operation.
        *
        */
    private void addCancelByEscapeKey(){
        String CANCEL_ACTION_KEY = "CANCEL_ACTION_KEY";
        int noModifiers = 0;
        KeyStroke escapeKey = KeyStroke.getKeyStroke(KeyEvent.VK_ESCAPE,
                          noModifiers, false);
             InputMap inputMap = this.getRootPane().getInputMap(
                          JComponent.WHEN_ANCESTOR_OF_FOCUSED_COMPONENT);
        inputMap.put(escapeKey, CANCEL_ACTION_KEY);
        AbstractAction cancelAction = new AbstractAction(){
            public void actionPerformed(ActionEvent e){
                closeDialog();
            }
        };
        this.getRootPane().getActionMap().put(CANCEL_ACTION_KEY, cancelAction);
      }

    /**
        * Hides the dialog.
        *
        */
    private void closeDialog () {
       this.setVisible(false);
```

```
   }
}
```

C. PROGRESSINDICATOR.JAVA

/usr/bin/Interrogator/ProgressIndicator.java

```
/*
    ProgressIndicator
    Adrian Arvizo
    This class displays a dialog box to indicate the program is busy.
    File: ProgressIndicator.java
 */

import javax.swing.*;
import java.awt.*;

class ProgressIndicator extends JDialog {

//----------------------------------
//    Data Members
//----------------------------------

    /**
     * Default frame width
     */
    private static final int FRAME_WIDTH    = 300;

    /**
     * Default frame height
     */
    private static final int FRAME_HEIGHT   = 200;

    /**
     * X coordinate of the frame default origin point
     */
    private static final int FRAME_X_ORIGIN = 150;

    /**
     * Y coordinate of the frame default origin point
     */
    private static final int FRAME_Y_ORIGIN = 250;

//----------------------------------
//    Constructors
//----------------------------------

    /**
     * Default constructor
     */
    public ProgressIndicator(JFrame frame, boolean modal) {
      super(frame);
        Container   contentPane;
        final JProgressBar progressBar = new JProgressBar();
        JLabel waitText = new JLabel("Busy, Please Wait");
```

```
        //set the frame properties
        setSize      (FRAME_WIDTH, FRAME_HEIGHT);
        setTitle     ("Busy, Please Wait...");
        setLocation  (FRAME_X_ORIGIN, FRAME_Y_ORIGIN);

        contentPane = getContentPane( );
        contentPane.setBackground(Color.white);
        contentPane.setLayout(new BorderLayout());
        contentPane.add(progressBar, BorderLayout.CENTER);
        contentPane.add(waitText, BorderLayout.CENTER);
        progressBar.setIndeterminate(true);

        try {this.setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));}
      finally {   }
        setDefaultCloseOperation( EXIT_ON_CLOSE );
    }
}
```

D. KEYWORDSEARCHOPTIONS.JAVA

```
/usr/bin/Interrogator/KeywordSearchOptions.java


/*
 KeywordSearchOptions
 Adrian Arvizo
 This class creates the keyword search dialog box.
 File: KeywordSearchOptions.java
 */

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class KeywordSearchOptions extends JDialog implements ActionListener {

//      ---------------------------------
//      Data Members
//      ---------------------------------

        /**
         * Default frame width
         */
        private static final int FRAME_WIDTH    = 300;

        /**
         * Default frame height
         */
        private static final int FRAME_HEIGHT   = 200;

        /**
         * X coordinate of the frame default origin point
         */
        private static final int FRAME_X_ORIGIN = 150;

        /**
         * Y coordinate of the frame default origin point
         */
        private static final int FRAME_Y_ORIGIN = 250;

        /**
         * An array of JCheckBox objects
         */
```

```
        private JCheckBox[] checkBox;

        /**
         * StringBuffers for easy manipulation of the extensions and keywords
         */
        private StringBuffer selectedExtensions, keywords;

        /**
         * Fields where the user may enter keywords
         */
        private JTextField keyword1, keyword2, keyword3, keyword4;

//      ---------------------------------
//      Constructors
//      ---------------------------------

        /**
         * Default constructor
         */
        public KeywordSearchOptions(JFrame frame, boolean modal) {
                super(frame, modal);
                Container   contentPane;
                JPanel      checkPanel, okPanel, keywordPanel;

                JButton     okButton, cancelButton;
                String[] btnText = { "Word (doc)", "PowerPoint (ppt)", "Excel (xls)",
                                "Text (txt)", "Webpages (html)", "Rich Text (rtf)" };

                //initialize the fields for keywords
                keyword1 = new JTextField();
                keyword1.setText("");
                keyword2 = new JTextField();
                keyword2.setText("");
                keyword3 = new JTextField();
                keyword3.setText("");
                keyword4 = new JTextField();
                keyword4.setText("");

                //set the frame properties
                setSize      (FRAME_WIDTH, FRAME_HEIGHT);
                setTitle     ("File Types");
                setLocation  (FRAME_X_ORIGIN, FRAME_Y_ORIGIN);

                contentPane = getContentPane( );
                contentPane.setBackground(Color.white);
                contentPane.setLayout(new BorderLayout());

                //create and place the checkboxes
                checkPanel = new JPanel(new GridLayout(0,1));
                checkPanel.setBorder(BorderFactory
                                .createTitledBorder("Select types of files"));
                checkBox = new JCheckBox[btnText.length];

                //add the checkboxes to the check panel
                for (int i = 0; i < checkBox.length; i++) {
                        checkBox[i] = new JCheckBox(btnText[i]);
                        checkPanel.add(checkBox[i]);
                }

                //create the area for keywords and add the fields
                keywordPanel = new JPanel(new GridLayout(0,1));
                keywordPanel.setBorder(BorderFactory
                                .createTitledBorder("Enter keywords"));
                keywordPanel.add(keyword1);
                keywordPanel.add(keyword2);
                keywordPanel.add(keyword3);
                keywordPanel.add(keyword4);

                //create and place the OK button and enable ok by enter key
                okPanel = new JPanel(new FlowLayout());
                okButton = new JButton("OK");
```

```
                okButton.addActionListener(this);
                okPanel.add(okButton);
                getRootPane().setDefaultButton(okButton);

                //create the cancel button and enable cancel by escape key
                cancelButton = new JButton("Cancel");
                cancelButton.addActionListener(new ActionListener() {
                        public void actionPerformed (ActionEvent event) {
                                closeDialog();
                        }
                });
                okPanel.add(cancelButton);
                addCancelByEscapeKey();

                contentPane.add(checkPanel, BorderLayout.WEST);
                contentPane.add(keywordPanel, BorderLayout.CENTER);
                contentPane.add(okPanel, BorderLayout.SOUTH);

                //register 'Exit upon closing' as a default close operation
                setDefaultCloseOperation( EXIT_ON_CLOSE );
        }

//      ------------------------------------------------
//      Public Methods:
//
//      void      actionPerformed        ( ActionEvent )
//
//  String     getKeywords                          ( )
//
//  String     getSelectedExtensions  ( )
//
//      ------------------------------------------------

        /**
         * Determines which file types are selected when the user clicks ok.
         *
         * @param event
         *
         */
        public void actionPerformed(ActionEvent event) {

                //flag that specifies if multiple checkboxes are checked
                boolean multipleSelections = false;

                //create an empty StringBuffer
                selectedExtensions = new StringBuffer("");

                //for each checkBox that is selected, append those extensions to the buffer
                if (checkBox[0].isSelected()) {
                        multipleSelections = true;
                        selectedExtensions.append("(.*doc)");
                }
                if (checkBox[1].isSelected()) {
                        if (multipleSelections == true)
                                selectedExtensions.append("|(.*xls)");
                        else {
                                multipleSelections = true;
                                selectedExtensions.append("(.*xls)");
                        }
                }
                if (checkBox[2].isSelected()) {
                        if (multipleSelections == true)
                                selectedExtensions.append("|(.*ppt)");
                        else {
                                multipleSelections = true;
                                selectedExtensions.append("(.*ppt)");
                        }
                }
                if (checkBox[3].isSelected()) {
                        if (multipleSelections == true)
                                selectedExtensions.append("|(.*txt)");
```

```
                else {
                        multipleSelections = true;
                        selectedExtensions.append("(.*txt)");
                }
        }
        if (checkBox[4].isSelected()) {
                if (multipleSelections == true)
                        selectedExtensions.append("|(.*html)");
                else {
                        multipleSelections = true;
                        selectedExtensions.append("(.*html)");
                }
        }
        if (checkBox[5].isSelected()) {
                if (multipleSelections == true)
                        selectedExtensions.append("|(.*rtf)");
                else {
                        multipleSelections = true;
                        selectedExtensions.append("(.*rtf)");
                }
        }

        //close the dialog box when done
        this.setVisible(false);
}

/**
 * Returns a String of the entered keywords.
 *
 */
public String getKeywords () {

        //create and empty StringBuffer
        keywords = new StringBuffer("");

        //flag that specifies if multiple keywords are entered
        boolean multipleKeywords = false;

        //for each keyword that is entered, append it to the keywords buffer
        if (!keyword1.getText().equals("")) {
                keywords.append("(" + keyword1.getText() + ")");
                multipleKeywords = true;
        }
        if (!keyword2.getText().equals("")) {
                if (multipleKeywords = true) {
                        keywords.append("|(" + keyword2.getText() + ")");
                }
                else {
                        keywords.append("(" + keyword2.getText() + ")");
                        multipleKeywords = true;
                }
        }
        if (!keyword3.getText().equals("")) {
                if (multipleKeywords = true) {
                        keywords.append("|(" + keyword3.getText() + ")");
                }
                else {
                        keywords.append("(" + keyword3.getText() + ")");
                        multipleKeywords = true;
                }
        }
        if (!keyword4.getText().equals("")) {
                if (multipleKeywords = true) {
                        keywords.append("|(" + keyword4.getText() + ")");
                }
                else {
                        keywords.append("(" + keyword4.getText() + ")");
                        multipleKeywords = true;
                }
        }
        if (!keywords.equals(""))
```

```
                        return keywords.toString();
                else return null;
        }

        /**
         * Returns a String of the selected extensions.
         *
         */
        public String getSelectedExtensions () {
                if (selectedExtensions != null)
                        return selectedExtensions.toString();
                else return null;
        }

//      ------------------------------------------------
//      Private Methods:
//
//      void addCancelByEscapeKey (  )
//      void closeDialog ( )
//
//      ------------------------------------------------

        /**
         * Enables the escape key to perform a cancel operation.
         *
         */
        private void addCancelByEscapeKey(){
                String CANCEL_ACTION_KEY = "CANCEL_ACTION_KEY";
                int noModifiers = 0;
                KeyStroke escapeKey = KeyStroke.getKeyStroke(KeyEvent.VK_ESCAPE,
                                noModifiers, false);
                InputMap inputMap = this.getRootPane().
                getInputMap(JComponent.WHEN_ANCESTOR_OF_FOCUSED_COMPONENT);
                inputMap.put(escapeKey, CANCEL_ACTION_KEY);
                AbstractAction cancelAction = new AbstractAction(){
                        public void actionPerformed(ActionEvent e){
                                closeDialog();
                        }
                };
                this.getRootPane().getActionMap().put(CANCEL_ACTION_KEY, cancelAction);
        }
        /**
         * Hides the dialog.
         *
         */
        private void closeDialog () {
                this.setVisible(false);
        }
}
```

E. FILEFILTER.JAVA

/usr/bin/Interrogator/FileFilter.java

```
/*
    FileFilter 0.2
    Adrian Arvizo
    This class filters files based on extension.  It takes a string of
extensions
    separated by semi-colons.
    File: FileFilter.java
 */

import java.io.*;

public class FileFilter implements FilenameFilter {
```

```
//    ---------------------------------
//    Data Members
//    ---------------------------------

      /**
       * The array storing each extension for which to search
       */
      private String[] extName;

      /**
       * Flag indicating if the file ends with the searched extension
       */
      private boolean containsExtension;

//    ---------------------------------
//    Constructors
//    ---------------------------------

      /**
       * Default constructor
       */
      public FileFilter () {
      }

      /**
       * Constructor taking string of extensions as an argument
       * Example argument: "txt;doc;xls;ppt;"
       */
      public FileFilter (String extNameTemp) {
            //fill the array with each extension separated by a semi-colon
            extName = extNameTemp.split(";");
      }

//    ------------------------------------------------
//    Public Methods:
//
//    boolean      accept        ( File,  String )
//
//    ------------------------------------------------

      /**
       * Checks if the file ends with the specified extension.
       *
       * @param dir folder where file is found
       * @param name is name of file
       *
       */
      public boolean accept(File dir, String name) {
            name=name.toLowerCase(); //for case insensitivity
            //ensure this is reset to false before each test
            containsExtension=false;
            for (int i=0; i<extName.length; i++) {
                  if (name.endsWith(extName[i].toLowerCase()))
                        //set to true if the file has this extension
                        containsExtension=true;
            }
```

```
            //returns true if the file has any of passed extensions
            return containsExtension;
      }
}
```

F. STARTINTERROGATOR.SH

```
/usr/bin/startInterrogator.sh

#!/bin/bash

# mount all drives except type noproc, sysfs, tmpfs, devpts
sudo mount -a -t noproc, sysfs, tmpfs, devpts

# change to the Interrogator directory
cd /usr/bin/Interrogator

# start Interrogator as root
sudo java Interrogator &
```

G. WABREAD.SH

```
/usr/bin/wabread.sh

# This script will parse a Windows Address Book file
# and pass the output to kwrite
# $1 is the input file, e.g. addressBook.wab

wabread "$1" | kwrite --stdin --geometry 1024x768+0+0
```

H. FIREFOXHISTORY.SH

```
/usr/bin/firefoxHistory.sh

# This script will parse a Firefox history file
# and pass the output to kwrite.
# $1 is the input file, e.g. history.dat

mork.pl "$1" | kwrite --stdin --geometry 1024x768+0+0
```

I. IEHISTORY.SH

```
/usr/bin/IEHistory.sh

# This script will parse an Internet Explorer
# history, cache, or cookie file
# and pass the output to kwrite.
# $1 is the input file, e.g. index.dat
pasco "$1" | kwrite --stdin --geometry 1024x768+0+0
```

J. DCFLDD.SH

/usr/bin/dcfldd.sh

```
# This script will create a forensically sound image
# $1 is the input file, e.g. /dev/sda1
# $2 is the output file, e.g. /media/sdb1/image1.dd
# $3 is the md5 hash output file, e.g. /media/sdb1/image1.md5

dcfldd if="$1" conv=noerror,sync hashconv=after md5log="$3" of="$2"
```

K. KEYWORDSEARCH.SH

/usr/bin/keywordSearch.sh

```
# This script will search a directory for files with
# the given keywords.
# $1 is the directory to search, e.g. /home/knoppix
# $2 is the file extensions to search, e.g. '(.*doc)|(.*txt)'
# $3 is the keywords to look for, e.g. "(nuclear)|(terrorist)
# -regextype posix-extended changes the type of regular expression syntax
# -iregex means ignore case
# -print prints the full filename to standard output
# sed -e 's/.*/"&"/' allows the handling of filenames with spaces
# xargs handles passing output from one command to the input of another
# egrep searches for words in a file using extended regular expressions
# -a processes a binary file as text
# -i means ignore case
# -l outputs only the name of the file containing the keywords

find $1 -regextype posix-extended -iregex "$2" -print | \
sed -e 's/.*/"&"/' | xargs egrep -a -i -l "$3"
```

L. STARTOUTLOOK.SH

/usr/bin/startOutlook.sh

```
#!/bin/bash

# This script copies an Outlook PST file to the
# knoppix directory and launches Outlook 2003.
# $1 is the input file, e.g. myData.pst

cp "$1" /home/knoppix
"/home/knoppix/.cxoffice/win2000/desktopdata/cxmenu/StartMenu.c^5E3A^5Fwindow
s^5Fprofiles^5Fcrossover^5FStart^2BMenu/Programs/Microsoft+Office/Microsoft+O
ffice+Outlook+2003"
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1. Hayes, Stephen F. *Saddam's terror training camps: What the documents captured from the former Iraqi regime reveal – and why they should all be made public*. The Weekly Standard. January 2006.

2. Answers.com. "Command Line Interface," http://www.answers.com/topic/command-line-interface. Last visited on 27 February 2007.

3. Hurricane Electric. "find," http://man.he.net/?topic=find&section=all, 1998. Last Visited 16 March 2007.

4. Hurricane Electric. "sed," http://man.he.net/?topic=sed&section=all, 1998. Last Visited 16 March 2007.

5. Hurricane Electric. "xargs," http://man.he.net/?topic=xargs&section=all, 1998. Last Visited 16 March 2007.

6. Hurricane Electric. "egrep," http://man.he.net/?topic=egrep&section=all, 1998. Last Visited 16 March 2007.

7. Solomon, Michael and Barrett, Diane and Broom, Neil. *Computer Forensics JumpStart*. SYBEX Inc., 2005

8. Hosmer, C. and Gordon, G. *FORENSIC INFORMATION WARFARE REQUIREMENTS STUDY*. WetStone Technologies, Incorporated.

9. Volonino, Linda and Anzaldua, Reynaldo and Godwin, Jana. *Computer Forensics Principles and Practices*. Pearson Prentice Hall, 2007.

10. "Defense Cyber Crime Institute," http://www.dc3.mil/dcci/dcci.htm, Last Visited 1 March 2007.

11. Interview between H. Barge, Major, USMC, Information Operations Staff, First Marine Expeditionary Force, Camp Pendleton, CA, and the authors, 28 October 2005.

12. Linux4n6.be. “The Belgian Computer Forensic Website,” http://www.lnx4n6.be, 19 October 2006. Last Visited 1 March 2007.

13. E-fense. “The Helix Live CD Page,” http://www.e-fense.com/helix/index.php, 6 October 2006. Last Visited 1 March 2007.

14. Inside Security. “INSERT,” http://www.inside-security.de/insert_en.html, Last Visited 1 March 2007.

15. DMZ Services. “F.I.R.E. Forensic and Incident Response Environment Bootable CD,” http://fire.dmzs.com/, 1 April 2004. Last Visited 1 March 2007.

16. Johnson, Thomas. *Forensic Computer Crime Investigation*. CRC Press, 2006.

17. Steel, Chad. *Windows Forensics, The Field Guide for Conducting Computer Investigations*. Wiley Publishing, Inc, 2006.

18. Pillay, Harish. “The Magic of Live CDs,” http://www.freesoftwaremagazine.com/articles/live_cds, 4 February 2005. Last Visited 1 March 2007.

19. Wikipedia. “X86 Architecture,” http://en.wikipedia.org/wiki/X86, 1 March 2007. Last Visited 1 March 2007.

20. “Bart’s Preinstalled Environment (BarPE) Bootable Live Windows CD/DVD,” http://www.nu2.nu/pebuilder, 17 February 2006. Last Visited 1 March 2006.

21. Canonical Ltd. “Ubuntu Home Page,” http://www.ubuntu.com, 2007. Last Visited 17 March 2007.

22. Novell. “Welcome to openSUSE.org,” http://en.opensuse.org/Welcome_to_openSUSE.org, 15 March 2007. Last Visited 17 March 2007.

23. Red Hat. “Fedora Project Wiki,” http://fedoraproject.org/wiki, 17 March 2007. Last Visited 17 March 2007.

24. Schirottke, Jorg (Kano). “Welcome to Kanotix,” http://kanotix.com, 4 December 2006. Last Visited 17 March 2007.

25. Wikipedia. “Cloop,” http://en.wikipedia.org/wiki/Cloop, 8 February 2007. Last Visited 1 March 2007.

26. Hurricane Electric. “mount,” http://man.he.net/?topic=mount&section=all, 1998. Last Visited 17 March 2007.

27. Hurricane Electric. “df,” http://man.he.net/?topic=df&section=all, 1998. Last Visited 17 March 2007.

28. Hurricane Electric. “cp,” http://man.he.net/?topic=cp&section=all, 1998. Last Visited 17 March 2007.

29. Hurricane Electric. “shutdown,” http://man.he.net/?topic=shutdown&section=all, 1998. Last Visited 17 March 2007.

30. Lilith.tec-man.com. “Libwab Home Page,” http://lilith.tec-man.com/libwab/index.html, 31 August 2006. Last Visited 1 March 2007.

31. OneStat.com. “Microsoft’s Internet Explorer global useage share is 85.81 percent according to OneStat.com,” http://www.onestat.com/html/aboutus_pressbox50-microsoft-internet-explorer-7-usage.html, 22 January 2007. Last Visited 14 March 2007.

32. Foundstone. “Free Tools,” http://www.foundstone.com/index.htm?subnav=resources/navigation.htm&subcontent=/resources/proddesc/pasco.htm, 2006. Last Visited 1 March 2007.

33. Wilson, Phil. “How to Export Firefox’s History to a Text File,” http://philwilson.org/blog/2005/01/how-to-export-firefoxs-history-to-text.html, 4 January 2005. Last Visited 1 March 2007.

34. Zawinski, Jamie. “When the Database Worms Eat Into Your Brain,” http://jwz.livejournal.com/312657.html, 3 March 2004. Last Visited 1 March 2007.

35. Sourceforge.net. “dcfldd,” http://dcfldd.sourceforge.net, 19 December 2006. Last Visited 14 March 2007.

36. Riefman, Jeff. “Microsoft’s Sacred Cash Cow,” http://www.seattleweekly.com/2004-06-02/news/microsoft-s-sacred-cash-cow.php, 2 June 2004. Last Visited 1 March 2007.

37. Alioth. “MS Outlook Perosnal Folders Converter,” http://alioth.debian.org/projects/libpst/, 16 January 2006. Last Visited 1 March 2007.

38. CodeWeavers. “CrossOver Linux,” http://www.codeweavers.com/products/cxoffice/, 2007. Last Visited 1 March 2007.

39. CodeWeavers. “Wine HQ,” http://www.winehq.com, 16 March 2007. Last Visited 17 March 2007.

40. CodeWeavers. “The CrossOver Difference: Superior Integration,” http://www.codeweavers.com/products/differences, 2007. Last Visited 14 March 2007.

41. SourceForge.net. “cryptcat – encrypting netcat,” http://sourceforge.net/projects/cryptcat, 18 October 2005. Last Visited 14 March 2007.

42. Fowler, Timothy. “RE: Forensics Thesis Project.” E-mail to Adrian Arvizo and Vincent Janowiak. 14 August 2006.

43. Wikipedia. “ZIP (file format),” http://en.wikipedia.org/wiki/ZIP_(file_format), 11 March 2007. Last Visited 19 March 2007.

44. VMware. “Download Add-ons,” http://www.vmware.com/download/downloadaddons.html, 2007. Last Visited 14 March 2007.

45. Barlow, Daniel. “Building Your Own Live CD,” http://www.linuxjournal.com/article/7246, 1 March 2005. Last Visited 27 February 2007.

46. KNOPPIX.net. “Knoppix Remastering Howto,” http://www.knoppix.net/wiki/Knoppix_Remastering_Howto, 2007. Last Visited 27 February 2007.

47. VMware. “VMware Server,” http://www.vmware.com/products/server, 2007. Last Visited 27 February 2007.

48. Knopper, Klaus. “KNOPPIX,” http://knoppix.com, 2007. Last Visited 27 February 2007.

49. Nero AG. “Nero 7 – Product Features,” http://www.nero.com/nero7/enu/index.html, 2007. Last Visited 1 March 2007.

50. Sonic Solutions. “Easy Media Creator 9 Suite,” http://www.roxio.com/enu/products/creator/suite/overview.html, 2007. Last Visited 1 March 2007.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Chris Eagle
   Naval Postgraduate School
   Monterey, California

4. George Dinolt
   Naval Postgraduate School
   Monterey, California

5. Adrian Arvizo
   Naval Postgraduate School
   Monterey, California

6. Vincent Janowiak
   Naval Postgraduate School
   Monterey, California

7. Brian Whyte
   SPAWAR Systems Center San Diego
   San Diego, California