Customization and automation in the future of digital forensics:

Live OS forensics with FENIX (forensic examiner unix)

by

Sean David Howard

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Co-Majors: Information Assurance and Computer Engineering

Program of Study Committee:
Dr. Doug Jacobson, Major Professor
Dr. Thomas Daniels
Dr. Steffen Schmidt

Iowa State University

Ames, Iowa

2007

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

FENIX (Forensic Examiner uNIX) is a Linux based live OS (Operating System) created to be used in remote environments for incident response and digital forensics. Between a joint effort between the Center for Information Protection (CIP) and the Iowa State University Police Department (ISUPD), FENIX has been tailored to suit the needs and requests of law enforcement forensic specialists. The very basis for FENIX is to allow ISU police officers the ability to carry an easy to operate and customize forensic toolkit with them at the scene of a crime allowing for better acquisitions and a deeper analysis to be conducted in an investigation.

FENIX isn't a standard forensic toolkit as others are. FENIX is a custom build of the Linux kernel with several of its own applications created for specific OS and forensic purposes and designed for a user friendly interface. However, since most users will be more familiar with existing tools those tools have the ability to interface easily with FENIX.

CHAPTER 1. INTRODUCTION

According to the 2005 Computer Security Institute / Federal Bureau of Investigations (CSI/FBI) survey, over 130 million dollars were lost due to computer crime in 2005. This overwhelming trend has led to an increase in demand for law enforcement agents trained in computer security. Crime committed through computer networks can be seen in the form of illegal content, internet fraud, identity theft, unauthorized access, sabotage, and abuse of internet communications. Due to the increase in computer crime in recent years, law enforcement agencies have begun to start computer forensic laboratories, employ computer forensic investigators, and train their current investigators in forensic techniques.

Unfortunately, every aspiration of technology is made available to criminals and users with malicious intent. New areas of research spring up everyday in order to deal with the new problems that technology has brought us. Areas of research like digital forensics that allows us to scientifically analyze evidence in criminal investigations. Digital forensics is a relatively new research area which offers a lot of possibility for growth and improvement.

Digital forensics research is an area that continues to grow. Much like other technologies, digital forensics piggybacks off of any new technology that comes out. New data formats, cell phone databases, new web browser cache entries are all obstacles that digital forensic specialists have to deal with. With the addition of new programs used by any criminal, a new tool must be used or an old one modified to deal with the new influx of data. Every month a new tool comes out that deals with these new data types and is eventually placed into someone's forensic toolkit.

When the tool is new and highly rated, LIVE CD's and OS's (operating systems) alike rush to add them to their distributions. Unfortunately, the addition isn't fast enough for most people. In fact most LIVE OSs are running their applications a few versions behind simply because they cannot keep up with the constant change in this digital millennium.

Most users not only expect that they can update their systems but also expect to be able to do it quickly and easily. Regular OSs can fix this problem with update tools like YAST (SUSE), YUM (Yellow Dog), RPM (Redhat), and APT (Debian). These package managers are not only highly effective but they allow even the most novice of users to quickly and easily update their systems.

LIVE OSs, however, are not as lucky; their memory is cleared when you reboot. This can create problems when installing a new package that requires a reboot. All information downloaded by the package managers (YAST, YUM, RPM, and APT) are lost on LIVE CDs and have to be updated every time you boot. It is even more unfortunate for the forensic community that more often than not, the computers that are being analyzed need to be kept in a protected environment far from any hazards such as access to the Internet. Extreme limitations in forensic environments make it nearly impossible to update operating systems or tools from the internet.

1.1 The FENIX Project

FENIX (Forensic Examiner uNIX) is a forensic LIVE OS like the Penguin Sleuth Kit, Spada, and Helix but FENIX has enhanced features that overcome the downfalls of the aforementioned LIVE OSs. FENIX features the ability to update part of the OS using modules that can be added and removed to change applications that reside on the LIVE OS. To update the OS a user needs 1) a computer to build the LIVE OS, 2) an Internet connection to obtain modules, and 3) media to store the OS. This modularity is a technology that has been used in other LIVE OSs like SLAX, goblinX, and BackTrack but has yet to be used on a forensic LIVE OS therefore making it a brand new technology.

FENIX is built upon a standard Linux Slackware distribution which allows for a comfortable feel to those that are familiar with Slackware. In addition to allowing for an easy

to understand configuration, FENIX provides graphical utilities to help investigators quickly navigate through the task of obtaining the information necessary for the investigation.

FENIX is a LIVE CD (bootable CD operating system) which grew out of the need for enhanced customization and automation in digital forensic research. FENIX has been designed and created from the study and research of presently available remote acquisition and analysis tools and their drawbacks. FENIX seeks to overcome these drawbacks with its enhanced ability to use new hardware configurations, be customized at anytime, automate some of the simpler functions of acquiring a disk image, and be user friendly.

1.1.1 The Approach

FENIX is a forensic LIVE OS as previously mentioned. It is a 1) forensic acquisition and analysis tool, as well as 2) a live OS, and should be evaluated on those two criteria. Because FENIX is a forensic OS it must abide by the laws of digital forensics which are:

1. Identify sources of documentary or other digital evidence.
2. Preserve the evidence.
3. Analyze the evidence.
4. Present the findings.

These principles must be obeyed in order for FENIX to be seen as a forensic OS. More specifically, FENIX must be able to determine the differences in sources of information; to preserve the evidence by opening the evidence in read-only environments; . to analyze the evidence; and able to present its findings. Due to the nature of Linux, some of these options are already in place. For instance, in order to identify sources of digital evidence, one only has to look at the names of the drives, the size, the serial numbers, and what bus the drive is connected to. It is the other three principles that FENIX must follow that are not innately available.

In order for FENIX to be derived as a forensic OS, the second stipulation is that it must be an OS. Slackware is the core to FENIX and therefore fulfills this stipulation with ease. However, I would like to assert that most users today are not very familiar with Linux and, therefore, a forensic OS should be intuitive and easy to use. There are a lot of OSs available today that require a great deal of work and never quite functions as expected. FENIX should run in an expected manner and should have the features that most users come to expect. This is where the customizations come into play. Since users have the options to customize just about everything they should be able to get their OS running as expected, assuming at least an intermediate understanding of Linux.

FENIX has additional features that are rarely seen, but are sometimes in mentioned in other applications. These features are unionFS and squashFS and they are the very core behind the modularity of FENIX. SquashFS is a filesystem type that is used in read-only / memory systems. This is very common in LIVE OSs as the type of media is usually read-only (CDs and memory sticks). SquashFS seeks to effectively function in a dual role. SquashFS shrinks the content of the OS by allowing the contents to be compressed and decompressed upon bootup. In addition, squashFS mounts the decompressed data into a read-only memory area. This read-only boot is extremely powerful in systems where the hard drive should not be booted (like forensic applications). In order for FENIX to maintain a small compressed size the Lempel-Ziv-Markov chain-Algorithm (LZMA) was used.

UnionFS is the other portion of the modularity in FENIX. UnionFS is a Linux filesystem service which implements a union mount for Linux filesystems. It allows the files and directories of separate file systems, branches, to be transparently overlaid to form a

single coherent filesystem. Contents of directories with the same path within the merged branches will be seen together in a single merged directory within the new virtual filesystem. When mounting branches the priority of one branch over the other is specified. So when both branches contain a file with the same name one gets priority over the other. Older LIVE OSs like KNOPPIX implemented UnionFS to merge changes within the filesystem and the contents of the CD, giving higher priority to the changed files. FENIX, however, in addition to giving changes higher priority gives every module added to the OS higher priority. This allows for updates to the OS. For instance, Apache will include a file to /etc/init.d/apache while the path /etc/init.d/ will already be available due to the core of the filesystem. This allows each module to add its own files into the proper locations using the unionFS transparent overlay system. This dual implementation of unionFS and squashFS is becoming common in newer modular LIVE OSs like goblinX and SLAX but has yet to be used in a functional forensic OS.

What separates FENIX from these newer modular LIVE OSs is FENIX's ability to automate the OS build process by connecting to its server and allowing the user to quickly and easily search its database entries for new content to download and compile into a LIVE OS. In fact, this server design helps to create a cohesive design so as to automate the customized creation of the OS. In addition to automation, FENIX features the ability customize the applications on the LIVE OS, as well as some core elements of the distribution.

1.1.2 The Concerns

There are several concerns that surround an OS capable of so much. Not only is this an operating system capable of reproducing itself but it is a forensic OS that must adhere to certain rules. The confrontation between customizability and functionality is not a new debate. However, it should be noted that the customizability takes precedence in this debate. It is in my findings that forensic LIVE OSs are in need but there is a greater need for customized LIVE OSs. As such FENIX has been created to be either a forensic OS or an OS that can carry out other needs. The user has the ability to make the LIVE OS non-forensic by allowing it write access to media that is connected to it. This feature is available when the user builds the OS.

How does FENIX build itself? It doesn't. A companion application known as Reburn builds FENIX and can even be placed as a module within FENIX so as to be able to rebuild FENIX from its own LIVE CD platform. Additionally Reburn is able to be installed on almost all Linux distributions provided they have the necessary dependencies.

1.1.3 The Solution - Reburn

Reburn is the process of configuring and building FENIX and is in the form of an application called by the user. So as to be thorough, Reburn takes the build and configuration process through 10 steps in order to create FENIX. Each one of these steps is necessary to get a depth of detail in the customization. In fact, most of the customizations are limited only to the media on which the LIVE OS will go. This simply implies that FENIX allows you the ability to not just put the OS on a CD, but a memory stick or a hard drive as well.

Additional concerns now come to the forefront of debate including the users own knowledge of forensic principles. Will the user be able to distinguish which customizations are necessary for forensic OS building? Yes, in fact there are a few areas in which there is a specific labeled forensic option. This option should allow the user to know precisely what is

necessary for their forensic OS. In addition, there is a save and load configuration option which allows a person with deeper knowledge of the subject matter to create a LIVE OS, store the configuration, and send the configuration to someone. This configuration is in the format of an XML file and can be seen in Appendix C. The schema can be seen in Appendix B.

1.1.4 The Design

The FENIX project is a three-fold arrangement of devices and entities. The FENIX OS, the core to this thesis, is the product of a process. The process is Reburn which can be launched from FENIX or a regular Linux desktop or Linux emulated desktop. Reburn talks to the FENIX server which is located on the Internet available to everyone. Upon a build or rebuild of a FENIX, Reburn will make requests from the FENIX server for information regarding modules currently available for download and will download them into temporary place on the user's computer. Upon completion of the configuration, Reburn will build, demonstrate, and write the OS to a media. This can further be explained in a closer look at Reburn as it is creating the FENIX CD.

First, FENIX is downloaded into a simple directory structure. The structure is necessary for the ease of manually updating the LIVE CD and is inherent in the cousins of FENIX: SLAX and goblinX. The directory structure includes several important directories that should be of some note.

/linuxcd/base/ – primary modules are included that are loaded as the OS.

/boot/ – boot loader and initialized ram disk directory.

/linuxcd/modules/ – modules that are loaded at boot but are not part of the core

/linuxcd/optional/ – modules that are not loaded at boot but can be later

/linuxcd/rootcopy/ – files that are placed in here are copied to the root after boot

/linuxcd/tools/ – applications from the Linux-Live toolkit

After the directory structure is laid out, the Reburn program begins to ask questions to the user concerning the configuration of FENIX.

Second, a previously made configuration file can be loaded that contains all of the settings from the last build or a new configuration can be made. The configuration file is an XML file that contains an XML schema designed specifically for the creation of the FENIX build. The XML schema has been placed in Appendix B and a sample XML configuration has been placed in Appendix C. Additionally, Reburn requires the user to name the location for where the FENIX directory structure should be located.

Third, the user automatically connects to the FENIX server located on the Internet and downloads a list of available kernels. Once the kernel is selected the information is stored for later.

Fourth, the user navigates a list of modules that are available from the FENIX server. The list of modules selected is stored in an internal structure within the Reburn program.

Fifth, Reburn takes the modules within the FENIX directory structure and scans for startup applications. The information is then displayed to the user for the user to decide what services should be started at startup by FENIX.

Sixth, KDE customizations are available where the user is able to select local files, backgrounds, and screensavers to use in their FENIX machine. The local files can be anything that they want available on the LIVE CD.

Seventh, the KDE taskbar can be customized to include four chosen applications that are quickly and easily available to the user. The taskbar takes an input of path to the application and path to the icon to be displayed.

Eighth, Reburn allows you to enhance FENIX by choosing your purpose for your LIVE CD. The purposes are available through quick links and allow you to choose your individual choices. Some of the quick links are: Forensics, Workstation, System Recovery, Penetration Testing, Windows Emulation, DVD Viewing, and CD Burning.

Ninth, the image of the LIVE CD, FENIX is created into an .iso format and can be stored for later use if you want to distribute the creation or store it for another time.

Tenth, Reburn allows you to test your image by emulating it onto your desktop through the external application QEMU. This lets you see and test your new LIVE CD without wasting a CD to see if you like it.

Finally, Reburn will burn your CD upon completion of accepting your demonstration. This allows you the ability to quickly burn once you have verified that your settings work as you requested.

1.2 Criteria Review

The success of FENIX should be measured in its ability to be a functioning OS for the majority of people that come in contact with it, as well as uphold the principles of forensic processes while handling evidence. Since HELIX, the Penguin Sleuth Kit, and Spada have all been used in forensic work for the last few years, FENIX should be no different. However, this area should be tested with the most basic of users all the way up to advanced users that are knowledgeable in the operations of most Linux systems.

Additionally it is my personal goal to maintain the benefits of other LIVE OSs while removing any of the shortcomings of the other LIVE OSs. These additional criteria can be more of a subjective conquest but I will discuss this further, in greater detail, later on in the paper.

CHAPTER 2. REVIEW OF LITERATURE

Naturally, newer technology has fewer papers written about it so it is necessary to find the source from which papers are written and test those while obtaining user opinions about the topic of LIVE OSs. As I have stated, FENIX is a forensic tool, which allows for both the examination of FENIX as a tool and FENIX as an OS. With these two areas of research combined a good idea of what should be expected of FENIX can be produced.

2.1 Digital Forensic Tool Research

Mocas , Kornblum, and NIST (EXAMPLE: CHANGE LATER Allen, B. S. (1984), Bruner, J. (1960) and Cox, S. R. (1974)) have done quite a bit of work with core forensic principles that should be included in any digital forensic tool. In Mocas' paper, "Theoretical Underpinnings of Digital Forensics Research," Mocas asserts that all digital evidence has certain properties that should be handled by forensic tools. These properties are discussed below. Kornblum's paper, "Preservation of Fragile Digital Evidence by First Responders," is a paper discussing the issues and process by which a forensic examiner might use to obtain the evidence. Kornblum explains in greater detail tools that are necessary when using a LIVE OS. The NIST document on, "General Test Methodology for Computer Forensic Tools," focuses more on the subject of how to test a forensic toolkit to make sure that it is functioning in a manner that is perceived as forensically sound. These topics are discussed in greater detail in the sections below.

2.1.1 Properties of Digital Evidence

Digital forensic tools are the creation of research which is often done aside from the practice of forensics. Unfortunately, most researchers aren't allowed the opportunity nor have the ability to develop research within the confines of real forensic cases. Mocas states that the separation of research and practice is acceptable but the combination of theory with

real world application would be, “highly useful for researchers to understand the context in which their research may be applied.”[1] If the researcher is able to understand the use and practices that are underlying in the nature of digital forensics it may help further his research efforts. Likewise, if the forensic technician is allotted the privilege of understanding the technology this would be mutually helpful in court when tools come under the scrutiny of the court system. Mocas uses the example of presenting digital evidence in court. If the research or the arguments used by the forensic technician to showcase their evidence is not well founded then producing evidence may not be good enough. When creating digital forensic tools it may be important to address the admissibility and reliability of the forensic tools to find and process the evidence.

2.1.1.1 Integrity

All forensic tools should afford a method for integrity of their evidence. This is very useful in court to prove that any data found within the evidence hasn’t been modified in any manner and that the evidence found is available in the original copy. Mocas goes so far as to make mention of two types of integrity to be used for evidence. There is the basic data integrity, assuring that digital information is not modified (either intentionally or accidentally) without proper authorization [1]. Agreeably, Mocas found this definition to be vague and open to interpretation. Mocas defines a digital form of integrity as duplication integrity: assuring that, given a data set, the process of creating a duplicate of the data does not modify the data (either intentionally or accidentally) and the duplicate is an exact bit copy of the original data set [1].

The difference between the two is that of macro and micro duplication. Both definitions require the evidence remain in a read-only state and remain unmodified during its duplication but the latter of the two definitions require that the duplicate be an exact bit copy. To a developer of forensic tools, this is a large significance to the research and development

that goes into the tool. What integrity provides is the ability to prove that two digital images are alike in perfect synchronization so one may be analyzed by officers while keeping the original in pristine condition, safe and away from the probing searches of forensic investigators. If in the event that the duplicate would be challenged a simple checksum of information can be taken on both images and if they match then the duplication is perfect. Integrity is important in court cases to provide admissible evidence and as such all forensic tools should have some method of providing integrity checks.

2.1.1.2 Authentication

Authentication provides forensic examiners with knowledge of information that surrounds the evidence. Provided a given illegal file, authentication allows us the knowledge of knowing who owned the illegal file content and who may have looked at it. Mocas provides us with two definitions of authentication. The first from computer security says authentication is knowing that the apparent author of text is in fact the true author [1]. This definition appears to apply more to cryptographic or secure communication and applies little to forensic evidence. The second definition, stemming from the legal disciplines state authentication is knowing that the electronic evidence is what its proponent claims. This latter definition is closer to what I need to use and is taken from the federal guidelines for searching and seizing electronic evidence. The passage speaks specifically about the author or creator of the evidence. This information can be specifically applied to the example of attempting to find out who owned the file and who accessed the file.

2.1.1.3 Reproducibility

This function seeks to be the process by which a forensic tool can build an accurate and reliable representation of events. In the Daubert ruling [Daubert v. Merrell Dow Pharmaceuticals, 509 U.S. 579 (1993)], used by federal courts to determine the admissibility

of expert opinion testimony, is in part based on being able to reliably test the merits of scientific evidence [1]. Reproducibility in any experiment helps provide a strengthened belief that a hypothesis is credible. Similarly, if a forensic tool is able to perform reproducible results, then the procedures are seen as credible.

Mocas provides us with yet another definition that helps in fully understanding the meaning of reproducibility. This definition states that given a data set or set of devices, the processes used to gather and / or examine evidence from the data set or devices are reproducible. Although all scientific evidence used in court doesn't meet this rule, it is still important to consider when researching and designing a digital forensic tool.

2.1.1.4 Non-interference

The key to any successful first response investigation is the ability to minimize the disturbances to the crime scene. This helps to preserve the crime scene from any distracting contaminations that could occur. When dealing with physical crimes the first officer on the location is to lock down the crime scene and not let any external forces into the area. Digital forensics is not so different. The most important aspect of securing a crime scene is to preserve the scene with minimal contamination and disturbance of physical evidence. In digital forensic terms non-interference is defined as assuring that the method (or tool) used to gather and / or analyze digital evidence does not change the original data set [1]. As stated, this is an important part of a forensic tool. If this principle cannot be attained a lesser appreciated property is still available – the property of identifiable interference. Mocas defines identifiable interference as, "assuring that when the method (or tool) used to gather and / or analyze digital evidence does change the original data set that the changes are identifiable." [1].

2.1.1.5 Minimalization

This property deals with informational/physical items that should be protected and/or not seized when conducting an investigation. Occasionally minimalization is governed by Federal law and is generally not necessary but should still be understood when making a forensic tool. Minimalization occurs when an item does not have evidentiary value. For instance, if a development server gets hacked by an outside source a forensic investigator might require the information on the server, such as logs and a listing of patches and applications. However, the investigator does not need the server itself. Minimalization allows the investigator to carry out his job without disrupting the jobs of others. Minimalization: assuring that the minimum amount of data was processed (seized and / or examined). This allows for the impacted service to have minimal damage due to the attack and ensuing investigation.

2.1.2 First Response Issues

Since FENIX is designed to be a first response forensic toolkit it is essential that it provide all of the expected functionality of a first response toolkit. Kornblum[2] discusses the fragile nature of digital evidence in first response situations. Often it is possible to modify or remove digital data without traces available to follow.

2.1.2.1 Fragile Evidence

As discussed briefly earlier, digital evidence follows rules that are both like and unlike regular evidence. In one sense once a crime scene is secured, the evidence of a crime such as fingerprints tend to be always available and one would have to go out of their way to destroy it. Even if the integrity of the evidence is threatened preserving the evidence can be easily accomplished with a minimum of expertise on the investigator’s behalf [2].

However, when a computer or digital evidence is within a crime scene, the situation is not as easily secured. In digital evidence the existence and tracing of information is not as straightforward. There is no quickly available information upon initial seizure of a computer. In order to find the details of an investigation, one must perform a full forensic analysis of the computer.

Unfortunately, technology has a way of both speeding things up and slowing things down. In this case a full system analysis is slow and difficult to process. This process is begun by securing the crime scene, which in this case, requires seizing the computer or the hard drive. Upon seizure, most computers tend to be on. The first step in a forensic seizure is to unplug the computer without shutting down the system. All the data on the computer that is not saved to the hard drive is then lost forever. Unfortunately newer computers have been known to have several gigabytes of information which can contribute to a lot of lost information when a computer is unplugged without saving any data.

An outsider to forensics might then consider not turning off the computer and acquiring information while the computer is still running. The problem with this arises in the situation of allowing an external user network access to the computer. Assuming the computer has been previously compromised, an intruder could remove evidence remotely without physically having access to the crime scene.

Another concern for investigators and toolkits is tainting the evidence. A user may instruct a computer to execute a previously defined taint method to corrupt evidence before investigators are able to take a look at it. As such proper precautions should be taken to ensure that these issues are not allowed to occur in the crime scene. Forensic toolkits and investigators alike should do their best to plan ahead and properly secure digital crime scenes.

2.1.2.2 Methods of Preservations

Kornblum explains that there are three types of evidence [2]:

1. **Transient data** – Information that will be lost at shutdown, such as open network connections, memory resident programs, etc.
2. **Fragile data** – Data that is stored on the hard disk, but can easily be altered, such as last accessed time stamps.
3. **Temporarily accessible data** – Data that is stored on the disk, but that can only be accessed at certain times.

Kornblum lists dangers to each of these types of data. Transient data is at risk of being lost if the machine turns off or is unplugged. Not only is transient data lost when a computer loses power but also programs and files can be moved from fragile data to transient data by copying from the hard drive into memory, then while the program is running the program is erased from disk. All trace of the evidence is lost upon being unplugged.

Fragile data is stored on hard disk. This information can be easily altered. Kornblum notes that this information is especially a constant worry for first responders trying to determine if an incident has occurred. Fragile data that can be modified by accident through access dates on files or temporary files. Once this information has been changed, there is no way of recovering the original data.

Finally, I must think about encrypted file systems. In these filesystems data isn't accessible all the time. These filesystems store information on the hard drive encrypted and only after an access device (password, finger print, or smart card) is presented will this information become available. Kornblum states that it is usually necessary for investigators to ascertain whether the system is an encrypted filesystem or not. If it is, the information should be captured before shutting down the system.

2.1.2.3 A Simple Model for a First Response Tool

The properties mentioned in Moca's paper should be remembered here in this section. These properties are: integrity, authentication, reproducibility, non-interference, and minimalization. As such a few items should be noted. The forensic integrity of the system must be maintained. The tool should do all evidence handling without any intervention from the user. The tool should gather all of the pertinent information that will be lost either during examination or transportation of the evidence. The tool should provide the first responder with enough information to determine if an incident has occurred. These are the simple rules that must be followed in order to provide a forensic first responder tool.

2.1.3 Test Methodologies for Forensic Tools

This section provides a description of the general approach taken to develop the test methodology for computer forensic tools and the rationale behind this approach in NIST development practices. The FENIX forensic LIVE OS uses NIST's testing methodology to determine FENIX's effectiveness as a forensic tool. This methodology is based on NIST's extensive experience in testing, proven methodologies for conformance testing, and international standards and guidelines for testing.

2.1.3.1 Approach

The general approach for testing computer forensic tools can be summarized as follows [3]:

1. establish categories of forensic requirements,
2. identify requirements for a specific category,
3. develop test assertions based on requirements,
4. develop test code for assertions,
5. identify relevant test cases,
6. develop testing procedures and method,
7. report test results.

As suggested by ISO 17025 for cases where there is no standard test method, the results of each step will be made available for public review. This guarantees the process is an open, public discourse which incorporates and reflects the needs of a wide variety of law enforcement practitioners and suppliers of computer forensic tools.

2.1.3.2 Forensic Requirements Categories

Forensic requirement categories are groupings of forensic functions that are determined by expert users. Grouping these forensic functions provides a smaller set of requirements that can be systematically approached by specialized forensic experts for testing. Narrowing the scope facilitates identifying the requirements for each functional grouping.

2.1.3.3 Forensic Requirements Specification

A forensic requirements specification prescribes the technical or functional requirements to be fulfilled by a product. Specifically, it identifies the requirements and features applicable to a category of forensic tools. A group of experts from federal, state, and local law enforcement organizations initially identify a list of requirements or specifications for the category of forensic functions. However, the final requirements are based on consensus review from the communities that use the tools. These requirements will be used as the basis for developing test assertions and test cases that will be used to test the forensic tools.

2.1.3.4 Test Assertions

A test assertion is a statement of behavior, action, or condition that can be tested or measured. The test assertions bridge the gap between the narrative of the specification and the test cases. Each test assertion is an independent, complete, testable statement for a

requirement in the specification. Each test assertion results in the realization of one or more test cases.

2.1.3.5 Test Cases

A test case specifies what is to be tested or one instance of what is to be tested. Test cases apply the assertions to the environments that are to be tested. Economic considerations limit the number of test cases that can be selected. Public review and the opinions of the group of experts from both computer science and forensic practitioners are used to narrow the number of test cases. In addition, the NIST Statistical Engineering Division is developing an independent experimental design to assist in the selection of test cases.

2.1.3.6 Test Method

The test method is a combination of the software used for testing and the procedures for completing the testing. Since testing methods for the forensic LIVE OS is a non-standard test, it is necessary to create our own that generally follow that of NIST's testing methods. These methods involve and closely follow that of forensic processes. A test should be created for each of the forensic properties mentioned before and are as follows: integrity, authentication, reproducibility, non-interference, and minimalization.

2.1.3.7 Test Results Reporting

This international standard gives specific information on what should be included in any test report. According to NIST, the forensic tool test results should include [3]:

1. a title stating what product was tested
2. identification of the testing environment, i.e., where the tests were run;
3. unique identifier for the test report; [identifier will be repeated on each page in order to ensure that the page is recognized as a part of the test report]
4. the name and address of the vendor;

5. identification of the testing software used;
6. unambiguous identification of the product tested including version, patches, etc.;
7. the test with the criteria for measurement;
8. the name(s), function(s) and signature(s) or equivalent identification of person(s) authorizing the test report;
9. where appropriate and needed, opinions and interpretations;
10. additional information which may be required by specific methods, clients, or groups of clients.

2.1.3.8 Repeatability and Reproducibility

Test results must be *repeatable* and *reproducible*. The procedures of the test method in addition to the testing software ensure this. The basic concepts of these two conditions are defined in ISO 5725, “Accuracy (trueness and precision) of measurement methods and results.” [8, 9]:

repeatability: Precision under repeatability conditions.

repeatability conditions: Conditions where independent test results are obtained with the same method on identical test items in the same laboratory by the same operator using the same equipment within short intervals of time.

reproducibility: Precision under reproducibility conditions.

reproducibility conditions: Conditions where test results are obtained with the same method on identical test items in different laboratories with different operators using different equipment.

As applied to computer forensic testing, repeatability is defined as the ability to get the same test results on the same testing environment (same computer, disk, mode of operation, etc.) Reproducibility is defined as the ability to get the same test results on a different testing environment (different PC, hard disk, operator, etc.).

2.2 Operating System Research

As the previous authors have provided research on forensic tools, Gleason & Fahey[16], Grundy[18], and the Knoppix design team[17] have done work on open source LIVE CD research. This section will primarily look at what is expected of LIVE OSs and how to obtain those goals. Unfortunately there hasn't been a great deal of information written on this topic and as such it is necessary to look at currently available LIVE OSs, their manuals, and user guides.

2.2.1 Helix

Helix was developed as a tool to provide the ability to acquire forensically sound images of many types of hard drives and partitions on systems running unique setups such as RAID arrays. Over time Helix has expanded to include newer open source tools and in addition to Linux tools, Windows tools were added allowing functionality for both Windows and Linux acquisitions and analysis.

2.2.1.1 What is Helix?

Helix is a LIVE OS that contains tools for both Windows and Linux. Helix itself was initially meant to be used as a bootable CD in first responder environments to acquire and analyze computer systems. The OS on the CD is a variant of Linux, known as Knoppix. Helix is listed as a work in progress and is stated that Helix is to be used by individuals with proper incident response and forensic training.

Although Helix is strongly based upon Knoppix, Helix has been tweaked to follow with its own series of modifications. Some of these major changes include [15] :

1. Helix will never use swap space found on a system – even if forced
2. The Helix automounter will set up drives it finds but will force a mount to be read only, noatime, noexec, nodev, noauto, user
3. Helix incorporates as many open source forensics/incident response tools that could be found.
4. Helix allows a capability for "Knock and Talk." This allows a parole officer to preview a system for graphic images that may violate a parole.
5. Helix has a Windows-side executable environment.
6. Helix has added an overlay system to allow writes to the CD.
7. Helix is updated every 3 months to keep current.

2.2.1.2 Forensic Topics with Helix

Helix will automount drives connected to the system and has the option to remount drives into read/write mode. This can come in handy but Helix strongly urges its users to have a hardware based read-only mechanism as well. This will surely disallow the tainting of evidence.

In addition to a revamped read-only hardware scheme, Helix has a long list of open source applications that come with the disk. There is a list for Windows and a list for Linux but for the sake of brevity I will list a few. On the Linux side of things, Helix includes Foundstone's open source tools, SysInternal's open source tools, NTSecurity open source tools, and a small list of PERL tools by Harlen Carvey. In addition to these tools, Helix comes with a host of its own tools which include, but are not limited to, a custom Cygwin

(Linux Emulation tool), FAU, GNU Tools, wft, getinfo, debugging tools, GNU-Win32 Static-Binaries, Linux Static-Binaries, and Solaris Static-Binaries.

These features allow Helix to be deployed a greater number of machines with almost any tool available to the forensic examiner.

2.2.2 Knoppix

Knoppix is the father of most current LIVE OSs. As stated above, Helix was almost purely derived from Knoppix and is a mere modification of it to a forensically sound status. Knoppix has several positives going for it. First, Knoppix allows officers the ability to examine an offender's computer without altering evidence just like Helix. Second, Knoppix allows officers to view multiple operating systems without difficulty. Third, Knoppix is free. All of these positive remarks are a great reason to try out Knoppix. Unfortunately, as I saw earlier, Helix has all of aforementioned accolades but has a whole slew of programs to go along with it.

2.2.2.1 First Responder Expectations

A first responder utilizing Knoppix is required to look at the manual and run a series of commands that will eventually boot into the system. The problem with this task is that it is a long process and most forensic examiners aren't as knowledgeable in digital forensics as one would like. The fact is that most people know Windows but not much Linux. When a user is prompted to determine what partition they want to read, it is enough to make the forensic examiner take the drive out of the computer and load up Encase to begin a forensic investigation easily. The task of attempting to do forensic acquisitions in Linux can be rather daunting and is a consideration when designing FENIX and other LIVE OS forensic toolkits.

Under Knoppix a thumb drive is supposed to be available for saving files of interest. Although this seems pleasant it can be a nightmare if the system automounts everything in

read-only. If the first responder, usually not a forensic investigator, has to be able to remount the thumb drive in Linux to be read-write capable. This seems a small task for those of us who are knowledgeable about Linux, but to the first responder this can be an arduous task.

2.2.3 Law Enforcement Forensic Examiner

This section deals with using basic Linux to help in forensic examinations. Most of the trouble in setting up a forensic environment is that of getting comfortable with the environment and getting applications to run as one would expect. Unlike LIVE OSs, regular Linux distributions have the option of being able to tweak programs until they work just right so settings can be saved.

However LIVE OSs have the advantage that someone has already done this work for you and the application is ready to run once activated.

2.2.3.1 Advanced Forensic Tools

There are more forensic examiner tools available to examiners in the Linux realm than those that are widely available for LIVE OS implementations. This is, of course, because the applications were made on Linux and have only been ported to one of the LIVE OSs. One of the most well-known tools is the Sleuthkit. This is a tool written by Brian Carrier and is both an acquisition and analysis tool. This tool is all inclusive and could easily be seen as the “Microsoft Office” of open source forensic tools. Autopsy is used as the GUI front end to the utility and works using a webpage based input to conduct the exam of the digital evidence.

SMART, by ASR Data, is another great tool but is not free like Autopsy/Sleuthkit. It is a GUI based forensic tool for Linux that allows a full set of forensic analysis capabilities to be wielded easily. SMART is full of options and is right click driven program. Most

functions are driven through mouse clicks which make it easier for those users who are more familiar with GUI applications rather than command line driven programs.

2.3 Research Area Review

As explained, FENIX is both a forensic tool and LIVE OS. I have discussed the necessity for FENIX to function like a forensic tool which requires the maintenance of integrity, authentication, reproducibility of results, non-interference from handling the evidence, and minimalization of the need to acquire information. The need for FENIX to handle as much data as possible is also inherently evident. Transient data is constantly being overwritten in a system and a need for capturing memory while it is available is necessary. In addition to transient data, FENIX needs to be able to handle fragile data and temporarily accessible data. This requires FENIX to be able to acquire images when the information is available and use up as little transient data as possible while performing its operations.

In this area I took a look at some currently available LIVE OSs and I saw what makes them different. Some of these options, like Helix, feature the ability to place as much information as possible onto one CD and allow the user to pick and choose what to use in his ongoing investigation. While others, like Knoppix, make the user do most of the investigations without highly developed tools of the trade. Instead the user relies on simplistic tools and schemes which make the process easier to understand for the investigator. While in the non-Live OS forensic Linux example, a basic install of Linux was our model and reflected how difficult a regular OS can be to setup a forensic environment. However, there was a tradeoff, that once the environment was deemed perfect, it was finalized and didn’t need to be remade every time the computer rebooted.

Finally, a resource for testing forensic tools was discussed. This discussion allowed me to get a better understanding of how to go about designing and testing FENIX as a forensic tool and an operating system.

CHAPTER 3. FENIX METHODS AND PROCEDURES

The FENIX project is a complex series of servers, clients, and applications working together in order to create a forensic LIVE OS. The goal of this project is to establish the ability for a user to easily upgrade and maintain their forensic OS while adhering to the standards of forensics. Current forensic LIVE OSs like Helix and the Penguin Sleuth Kit offer a slew of forensic tools and features that make them great. FENIX however, wishes to maintain those great features but allow users the ability to upgrade their systems and put their own applications onto the OS without having to worry about corrupting the performance.

There are many underlying technologies that make FENIX able to succeed at its goals. However, before I delve too far into how FENIX works, it would serve me well to take a look at the project in its entirety and then focus on the details.

3.1 System Overview

Reburn isn't necessarily the foundation of FENIX but it creates the foundation. Reburn is a program designed to be installed on any Linux distribution and requires a few dependencies. Currently, Reburn requires the use of c++, qt 3.3 >= 4.0, qt libraries of mySQL interactions, QEMU, and an Internet connection. C++ and qt are necessary in order to build Reburn into an application on the system it is being installed on. QEMU however, is an emulation application and is necessary for a demonstration of the built application before it is burned onto a CD or placed onto a memory stick. Reburn is launched by the user and will request the user to point to a directory structure that should have already been extracted onto the hard drive. This directory structure seeks to function as the skeleton for our product.

As the user proceeds through Reburn, information is sent across the Internet using a username and password scheme to obtain information about FENIX kernels and modules. The modules and kernels are sent back to Reburn which places them into their respective locations in the OS skeleton. This procedure can be seen below.

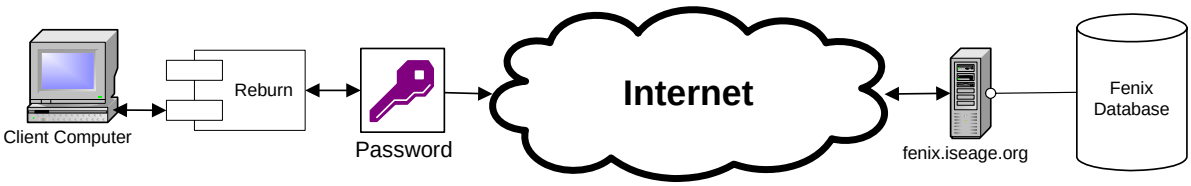


Figure 1. The FENIX Project Object Diagram

Upon completion the user is asked to customize the OS which includes almost everything from startup applications to backgrounds, screensavers, and taskbar management. Upon completion of the customizations, FENIX is built by Reburn and then demonstrated through QEMU. The user then decides if more tweaking is necessary. If so, the user will back up and modify the options until the demonstration is exactly what is required. If the demonstration is what is required, the user proceeds onto the next stage which is burning the image to CD or copying the data to a memory stick.

From this point forward the user has successfully created FENIX, a forensic LIVE OS. While discussing the process of how to make FENIX, some details were glossed over and should be covered here. In order to maintain a certain sense of order, configuration files are available. These configuration files save all of the options requested by the user and when selected will load all of the options as they were last chosen. This includes all modules and kernels that were downloaded and placed in the skeleton. Configuration files allows the user to recreate a copy of FENIX that is identical to the last version he created.

The FENIX Server maintains all kernels and modules. The kernels and modules are updated and obtained by users who login to the site, discuss the OS and port their favorite applications to the .lzm file format. This file format is exactly the same as the SLAX file

format and the modules between the two OSs are exchangeable. As such, the two servers are often merged in order to facilitate the needs of the users and allow the users a broader choice of applications.

Now that I have covered some of the details that were glossed over, I should delve deeper into the underlying processes of the FENIX Project.

3.1.1 Build Process

The FENIX forensic OS is built by Reburn and requires a pre-existing directory structure of FENIX to be available. This directory structure is available online at the website in a simple tar.gz format. The user may go online and download both Reburn and the FENIX directory structure from the same location. Upon downloading the directory structure, one is able to extract the directory structure using the command:

```
tar –xzvf fenix-skeleton-1.0.tar.gz
```

Figure 2. FENIX Skeleton Extraction Using tar

Issuing this command extracts the directory structure onto the hard drive in the directory that the command was issued. Now, a directory named “fenix-skeleton-1.0” should be seen within the current directory.

3.1.1.2 FENIX Directory Structure

It should be noted that it is not necessary to actually navigate into the FENIX skeleton directory but should be done so if knowledge of the system you are using is necessary in your line of work, such as expert witness in a trial. If you navigate into the directory, you might notice that there are several directories and several files. The directory structure includes six important directories that should be of some note.

/linuxcd/base/ – primary modules are included that are loaded as the OS.

/boot/ – boot loader and initialized ram disk directory.

/linuxcd/modules/ – modules that are loaded at boot but are not part of the core

/linuxcd/optional/ – modules that are not loaded at boot but can be later

/linuxcd/rootcopy/ – files that are placed in here are copied to the root after boot

/linuxcd/tools/ – applications from the Linux-Live toolkit

The base is where core modules will go that are required to load the Linux operating system. These will be rarely updated unless it is necessary to modify the operating system at its core level and should be done by experts only. The base directory includes the Kernel, Kernel Fixes, the X server, X application libraries, Common libraries, KDE, KDE Applications, KDE Office, SLAX enhancements, and FENIX enhancements. Each of these modules is necessary. As the Linux union filesystem loads the modules one by one into its memory space the modules help to build the OS.

The modules directory is similar except modules aren't necessary in the building of the OS. The modules in that directory are mostly applications and other non-vital unionFS modules that are there to support the user in their goals.

The boot directory is read before the system loads the modules into memory. Within this directory are the initialized ramdisk and the bootloader, GRUB. The information in the boot directory is rewritten by Reburn and information should not be placed into this directory manually, unless modifying the GRUB menu graphic located in the file /boot/GRUB/splash.xpm.gz.

The optional directory is just that, optional. These are modules that the user might want to have available but won't want them to load on startup. The significance of this directory is minimal unless attempting to preserve memory in which a minimal load of modules would be optimal and then a slow load of optional modules would commence.

The rootcopy directory stores files that the user wishes to place in the root of the filesystem. At the end of the unionFS process at bootup, the rootcopy directory is unioned as root (/). This serves as an easy way to sneak files into the filesystem easily. Naturally paths are copied over the current paths. This allows for a user to place with the rootcopy folder the following link /rootcopy/etc/init.d/apache. This will direct a link to /etc/init.d/apache and be able to call for a web server startup at boot without having to create a module or rebuild FENIX.

The tools directory is used to house some of the tools used in the development and reconstruction of FENIX. The tools available within are also available for download from Linux-Live. These tools help to build modules and modify modules. These are very useful in the effort to perpetuate the existence of current modules.

3.1.1.3 Reburn Application Process

The directory structure made available by the user is key to Reburn. Reburn downloads all information into the directory structure with the exception of the configuration files. All modules and kernels downloaded from the website are stored within the directory structure. The user chooses options within the application that are either stored in temporary memory as a variable within the program or placed within the directory structure is input. Upon completion of the Reburn application the variables are fed to the Linux-Live boot disk creation scripts to build the OS. When requested the variables are dumped into an XML file for later use. This would be in the event that the user wants to duplicate the CD without the actual CD.

3.1.1.4 FENIX Client Server

The FENIX client server is accessed during the Reburn process early on. During the second stage of execution Reburn calls the SQL server running on the FENIX server. The

Reburn client requests information about all kernels available. The list is sent over to the Reburn client. The Reburn client then displays the information in a table to users. Once the user selects one of the kernels the id is stored within a temporary variable.

Within the server each module and package is issued a unique id. There is no difference between kernels and modules for databases or ids. A single user record will have a different category name under it. For kernels the category is kernel and applications can have many varying categories. Some of these categories are artwork, graphics, multimedia, games, office, education, network, security, system, develop, drivers, multilang, console, libraries, and other. These categories were taken from the SLAX website so as to conform with SLAX's standard.

Additionally, scripts are run weekly to synchronize the information included within the SLAX modules and those within the FENIX modules. In an attempt to combine the efforts, FENIX is able to take on the guise of SLAX when necessary.

When Reburn requests information from the FENIX server, the FENIX server requires a basic username and password authentication. Although this information is being sent over the standard port and is readable, the purpose of the username and password is in order to keep bots from attempting to obtain access and use system resources. All information within this account that the remote client can access is read-only which prohibits the ability to upload any information to the database.

Write access is available to the database using a different account and a different password. The write access is restricted to localhost only which disallows any intruder the ability to modify information remotely. The write access is used for secure accounts on the web server to upload and download their favorite modules. This allows users the ability the fully integrate their applications and digital lifestyles into their LIVE OSs.

3.2 FENIX Functionality – From Boot to User Input

FENIX is a product of the Linux-Live technology and as such boots just like SLAX, goblinX, or any of the other Linux-Live OSs. FENIX's boot process can be divided into three parts. At first, the linux kernel (vmlinuz) is loaded, initrd.gz is unpacked into a 4.4mb ramdisk in the memory and it's mounted as a root filesystem.

The second phase is done by starting /linuxrc script. Linuxrc is part of the Linux-Live scripts which are copied into initrd.gz during the LIVE OS creation. Temporary filesystem (tmpfs) is mounted by linuxrc to /mnt and all files from /base and /modules directories on the CD are inserted into the live filesystem (eg. /base/kde.lzm is mounted to /mnt/kde.lzm, /base/xwindow.lzm is mounted to /mnt/xwindow.lzm, etc.) Next, images specified by load= kernel boot parameter are inserted (from /optional directory on the CD) and finally /mnt is chrooted. (/mnt will become /, so for example /mnt/bin/bash will become /bin/bash).

Finally, in the third phase, /linuxrc starts /sbin/init. It's /mnt/sbin/init in fact, but now /mnt is chrooted so /sbin/init is called (by using exec bash built-in command). From this point linux takes care of the rest loading into the OS depending on the options chosen by the user. This process can be seen in the diagram below.

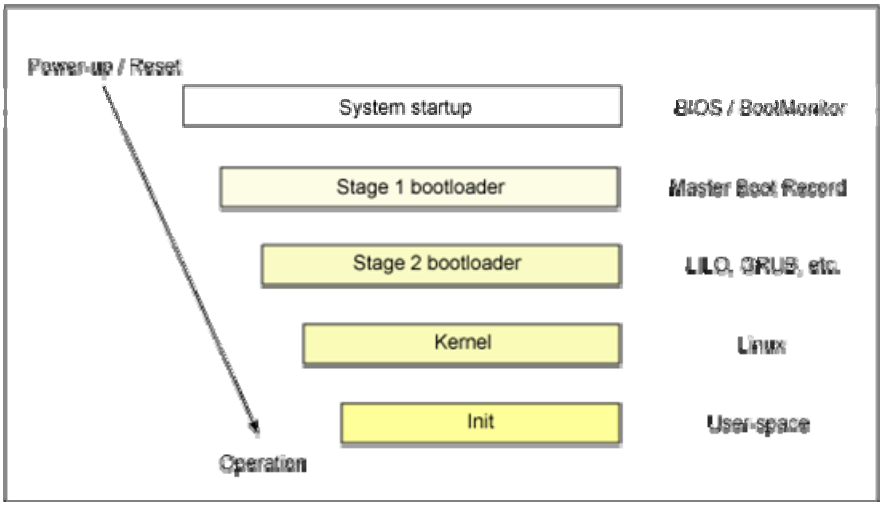


Figure 3. Linux Boot Process, diagram taken from IBM Research [15]

3.2.1 El Torito Boot Process

At the initial startup of the CD or memory stick boot process, the BIOS is read which tells the computer where to find the first master boot record to read. If the BIOS is setup to read the hard drive last with the CD-ROM or a memory stick first, the El Torito boot process has begun. The El Torito Bootable CD Specification is an extension to the ISO 9660 CD-ROM specification. It is designed to allow a computer to boot from a CD-ROM. It was first issued in January 1995 as a joint proposal by IBM and BIOS manufacturer Phoenix Technologies.

A modern PC BIOS will search for boot code on an ISO 9660 CD according to the El Torito specification. If the CD contains bootable code the BIOS will assign a BIOS drive number to the CD drive. The drive number assigned is either 80 (hard disk emulation), 00 (floppy disk emulation) or an arbitrary number if the BIOS should not provide emulation [13].

Emulation allows older operating systems to be booted off a CD, by making it appear to them as if they were booted off a hard or floppy disk. Newer operating systems do not require emulation to boot; all that is needed is an appropriate boot loader such as GRUB.

3.2.2 GRUB (Grand Unified Bootloader)

GNU GRUB is a boot loader package as mentioned in the previous section. GRUB allows a user to have several different operating systems on their computer at once and to choose which one to run when the computer starts. In the case of FENIX GRUB allows multiple kernels or multiple settings to be previously defined and chosen by the user. FENIX automatically has a memory test available to check the memory of the system that the LIVE OS is to be run on. As mentioned GRUB can be used to select from different kernel images

available on a particular operating system's partitions, as well as to pass boot-time parameters to such kernels [14].

3.2.3 Initialized Ramdisk and Kernel

A ramdisk is used as a temporary file system allocated in memory. This filesystem is used by the Linux kernel during boot. While this occurs, a space in memory is allotted as a harddrive for FENIX to load files into. The initrd is typically used for making preparations before the real root file system can be mounted. Fortunately, ramdisk support is compiled into all kernels that come with FENIX and all LIVE OSs. When specified, the boot loader loads the ramdisk into memory and passes it on to the kernel which temporarily mounts it as root and executes the /linuxrc executable. /linuxrc is the kernel executable and begins the Linux boot process.

3.2.4 UnionFS and SquashFS

The initialized ramdisk isn’t the only kernel configuration that had to be compiled in. In fact unionFS and squashFS along with the initialized ramdisk weren’t the only options compiled in. But for now I will only take a look at these three main entities. I already discussed the initialized ramdisk. I lightly touched on unionFS and squashFS as well. To recap, unionFS is used to mount several images or disks into one filesystem. This can be useful when using multiple layers of filesystems. In the case of modularized OSs, unionFS is necessary as it binds all of the modules together within the same root filesystem.

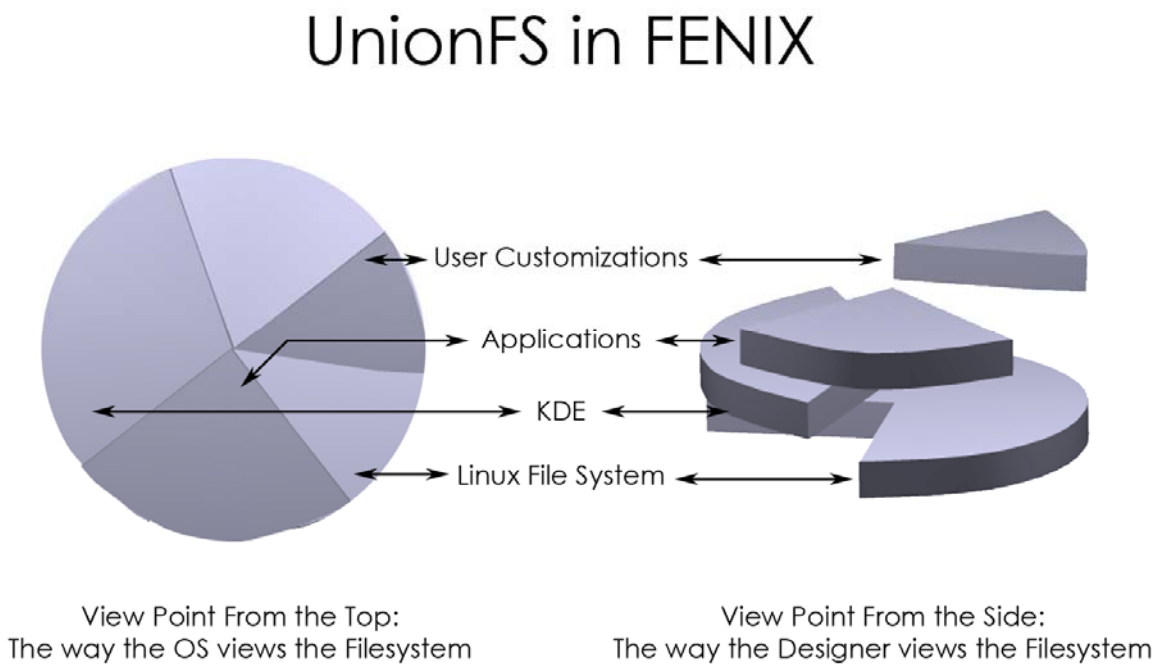


Figure 4. UnionFS Transparent Layers in FENIX

To further explain how unionFS works, I will use the concept of transparent layers. UnionFS allows Linux the ability to overlay transparent filesystems in order to merge several together. Think of transparences of circles which cut into pie wedges. The higher layer covers maybe 1/8 of the circle, while the lower layers cover even less. The bottom layer covers ¾ of the circle. From a top down view, the circle is completely covered. UnionFS seeks to function like this in order to allow for modular expansion of data. The diagram above demonstrates the metaphor used.

In addition to UnionFS, FENIX exhibits yet another useful filesystem known as SquashFS. Squashfs (.sfs) is a free (GPL) compressed read-only file system for the GNU/Linux operating system. Squashfs compresses files, inodes and directories, as well as supports block sizes up to 64K for greater compression. It is implemented as a kernel module just like UnionFS.

Squashfs is intended for general read-only file system use, for archival use (i.e. in cases where a .tar.gz file may be used), and in constrained block device/memory systems (e.g. embedded systems) where low overhead is needed. SquashFS or a variant (cramFS, zisofs, cloop, e2compr, and cromFS) is used in almost every LIVE OS bootsystem. The benefits of SquashFS over other compression filesystems are that it maintains some of the highest compression rates while maintaining efficiency

3.2.5 Linuxrc

The application of linuxrc into the boot process is the beginning of the second phase of booting. This is mentioned earlier but is to be discussed more elaborately here. To further the understanding of linuxrc let us first define it. Linuxrc is a program that is started in the start-up stage of the kernel prior to the actual boot process. This allows FENIX to boot a small modularized kernel and to load the few drivers that are really needed as modules into a small temporary filesystem (tmpfs). Linuxrc assists in loading relevant drivers manually.

3.2.6 Module Loading

Linuxrc then copies all files from the FENIX directory structure under /base and /modules into the live filesystem. Specifically, in the basic FENIX disc, /base/01_kernel.lzm, /base/02_coreFix.lzm, /base/03_xwindow.lzm, /base/04_xap_libs.lzm, /base/05_common.lzm, /base/06_kde_base.lzm, 07_kde_apps.lzm, 08_kde_office.lzm, 09_slax.lzm, 10_fenix_wallpaper.lzm, and 11_Fenix_KDE_Enhancements.lzm are mounted to /mnt/$FILENAME.

Finally the parameters specified by GRUB are passed off to the kernel boot sequence. With the Linux-Live utility, this allows for optional choices of modules to be loaded in addition to the modules that were in /base and /modules. These optional modules are found under the /optional directory within the FENIX directory structure. Once the parameters are

passed, /mnt is chrooted making /mnt become /. This allows for all files within /mnt to become merged using unionFS to become the only / directory.

At this point, linuxrc continues to drive the boot process and passes control to /sbin/init. This new process controls the exec bash built-in command. The boot process is now handled by Linux's boot process.

3.2.7 Linux Boot Process

Once /sbin/init has been run, the main process reads /etc/inittab. The boot scripts are then run. As FENIX's core is that of Slackware, the boot scripts are found in /etc/init.d/. The boot process calls /etc/init.d/*. The boot process then switches to runlevel processes. The runlevels are named and described below.

- 0: Halt
- 1: Single User Mode
- 2-4: User Defined
- 5: X11 only
- 6: Reboot
- Default: Defined in /etc/inittab

FENIX starts up on runlevel 3 then a system initialization script is called under /etc/rc.d/rc.S. The xserver is then called up under the xinit script.

3.2.8 Xserver and KDE

Run level programs are found in /etc/init.d/. The startup script for the xserver is found in /etc/rc.d/rc.4. This script is called when the boot process increases the run level to level 4. The kdm session manager is launched from rc.4. The location of the kdm manager is in /opt/kde/bin/kdm. The following statement is executed within the rc.4 script to launch the KDE GUI.

```
exec /opt/kde/bin/kdm -nodaemon
```

Figure 5. KDE Launch Command From rc.4

At this point KDE takes over the bootup sequence. One of the strengths of FENIX is it usability and the GUI is a key component to any applications usability. As such, FENIX has been enabled to allow for no-login options in secure environments where usernames and passwords are unnecessary. Once the KDM, KDE manager is loaded, if there is no password, the OS will launch you directly into your environment after loading all the necessary core components of the KDE desktop. If there is a password, the user is prompted with a password authentication to login to the system.

3.2.9 User Interactions with the GUI

Once a user is in the interface a user should easily be able to navigate the operating system as one would normally. The challenge at this point is getting Windows users comfortable with Linux. Unfortunately most of this area is out of the scope of this project but FENIX has allotted for some similar features between Windows and the KDE desktop. An effort was made to use similar icons and similar placement of navigation components to enable the user to feel more inclined to browse the system.

3.3 Reburn – The Path to a Customized LIVE OS

One of the primary elements of the FENIX Project is Reburn, the application that allows the customization, configuration, and reproduction of the FENIX forensic OS. Without Reburn, FENIX would be just another forensic OS. Reburn allows extreme customizations to be made to the OS from kernel level to file system customizations and

even appearances. FENIX can easily be customized to be not just a forensic OS but a portable webserver or even a temporary workstation. The possibilities are endless.

A great deal of work went into Reburn to provide the end user with as many customization choices as possible while providing the user with easy to access functions. Unfortunately draw backs were inevitable as they always are when attempting to create user friendly environments with a great deal of control.

The design of Reburn is made to be linear. The goal is to not only help organize the customizable options within the data structures of the application but also to help the user organize their thoughts when completing the configuration. The linear cycle starts out with linking the application to the forensic CD structure. The linear path then leads to kernel and module options and finally down to purpose of the CD and the graphical customizations of the OS. Finally the utility allows a user to demo and produce the OS onto a memory stick or CD. The additional option of saving and loading configuration makes it easy for a novice user to create a specific OS that has been designed by a more experienced user. The applications of this feature are highly useful when creating a standard OS for a larger entity, such as a corporation or a university.

3.3.1 Configuration Files

The design for the configuration file standard can be seen in Appendix B and C. In Appendix B the standard is shown as an XML schema with five elements underlying a single element. The element <config> is used only as a container for the actual core elements of the configuration. Beneath the <config> element are additional elements which contain: kernel, modules, startOnBoot, guiTaskbar, special. Each of these elements have specific restrictions to each.

3.3.1.1 Kernel

The kernel element is limited to only one declaration, as FENIX is designed to only support one kernel for now. The kernel listing in the configuration file is also set to be a string. This string contains the id number that is registered to the data that is expected by the user and is linked together through the client server model that will be discussed later.

```
<kernel>24</kernel>
```

Figure 6. The <kernel> Element from ”config” XML Code Snippet

3.3.1.2 Modules

The modules element is not limited to a certain number as only the media can define how many modules are available to fit within FENIX. The restriction of what can go within the modules element is yet another element <module>. Within the <module> element is a restriction of only a string. Again, the string restriction is to be used for the id that is related to the data within the client server model.

```
<modules>
        <module>24</module>
                …
        <module>35</module>
</modules>
```

Figure 7. The <modules> Element from ”config” XML Code Snippet

3.3.1.3 startOnBoot

The startOnBoot element, under <config> is not limited. The restriction to the information contained within the element is that of a string. However, unlike the module and kernel elements, startOnBoot stores paths of files to load on startup.

```
<startOnBoot>/etc/rc.d/apache</startOnBoot>
```

Figure 8. The <startOnBoot> Element from ”config” XML Code Snippet

3.3.1.4 guiTaskbar

The guiTaskbar is used within the FENIX build for its customization of the KDE environment. Within the XML body, the guiTaskbar is restricted to a maximum of 4 entries and a minimum of 1 entry. Within the guiTaskbar tags an entry for the icon location on the disc should be listed. Within the guiTaskbar tag a parameter under the variable “application” should be directed toward the application that should be launched on the forensic OS. An example is seen below:

```
<guiTaskbar application=’/bin/dd’>/root/.kde/icons/64x64/test.ico</guiTaskbar>
<guiTaskbar application=’/bin/oocalc’>/root/.kde/icons/64x64/calc.ico</guiTaskbar>
<guiTaskbar application=’/bin/kword’>/root/.kde/icons/64x64/kword.ico</guiTaskbar>
```

Figure 9. The <guiTaskbar> Element from ”config” XML Code Snippet

3.3.1.5 special

The special element is reserved for special bootup options and other miscellaneous features that determine how FENIX runs. These options include the following:

3.3.1.5.1 nohotplug

This option disables almost all hardware autodetection in the case of hang-ups. Your hardware won't be detected at all. You'll have to use "pcimodules" command after logging into FENIX and you'll have to try to modprobe all needed modules from the list manually.

3.3.1.5.2 nopcmcia

This option allows a user to skip the auto detection of pcmcia hardware devices. This is very useful in the event of hardware detection hang-ups.

3.3.1.5.3 noagp

This option allows a user to skip the auto detection of agp hardware devices. This is very useful in the event of hardware detection hang-ups.

3.3.1.5.4 acpioff

This option allows a user to skip the auto detection of acpi hardware devices. This is very useful in the event of hardware detection hang-ups.

3.3.1.5.5 setroot

This option allows a user to set root's password.

3.3.1.5.6 askpass

This option allows a user to be asked for the password to login to the system.

3.3.1.5.7 disableGuest

This option allows the removal of the guest account.

3.3.1.5.8 copy2mem

This option allows the OS to be copied to memory entirely before the OS is run. This can be memory intensive, but will allow the OS to run at greater speeds. In addition, this option is required in order to read or write to the same CD-ROM that was used to boot from.

3.3.1.5.9 testMem

This option allows the user to test the memory of the machine before the system is run.

3.3.1.5.10 memSize

This option allows a user to restrict the space that FENIX is allowed to use, in memory.

3.3.1.5.11 floppy

This option allows for the mounting of a floppy driver.

3.3.1.5.12 webconfig

This option allows for the enabling of SLAX's webconfig feature. This option allows you to save your modifications to the SLAX website. More information can be read at: http://www.slax.org/webconfig.php

3.3.1.5.13 saveChanges

This option allows the mounting of readWrite to another device in order to write modifications of the FENIX OS and be able to load them later. This almost makes the OS like a regular non-LIVE OS.

3.3.1.5.14 memTestAvail

This makes a memory test available for a user of FENIX, but doesn't require the memory test.

3.3.1.5.15 disableNet

This option disables the network while the computer boots. This is necessary where one would want a safe environment in which to load the OS without allowing any external connections. This can be helpful in a worm infected machine or a machine that is connected to an external connection like the Internet.

3.3.1.5.16 kiosk

This option allows for the computer to be locked down and only minimal features are enabled. This is very helpful in certain cases where one would want the user to have minimal abilities but enough to do basic functions like check email, browse the Internet, and even install temporary applications.

3.3.1.5.17 additional

This option has been made available to future possibilities. It allows the user to make direct commands to the kernel at boot time.

3.3.1.5.18 memStick

This option is available to users who wish to use a LIVE OS on a memory stick instead of a LIVE CD. This option is becoming more and more prevalent and to leave this option out would cause the OS to lose a lot of its convenience and appeal.

3.3.1.5.19 startGUI

This option allows the OS to go right to the GUI on boot. It allows a user to bypass unnecessary steps in their boot process.

3.3.1.5.20 writeProtect

This forensic option write protects the mounting of all data that is plugged into the system. This allows for a forensically sound process. The investigator can remount drives as read-write if requested but only upon request.

3.3.1.6 special

The special element is shown in the block below

```
<special>
        <nohotplug/>
        <nopcmcia/>
        <noagp/>
        <acpioff/>
        <setroot>ThePassword</setroot>
        <askpass/>
        <disableguest/>
        <copy2mem>
                <ejectcd application='true'/>
        </copy2mem>
        <testMem/>
        <memSize>80</memSize>
        <floppy/>
        <webconfig/>
        <saveChanges/>true</saveChanges>
        <memTestAvail/>
        <disableNet/>
        <kiosk/>
        <additional>Additional Boot Options</additional>
        <memStick/>
        <startGUI/>
        <widescreen/>
        <wireless/>
        <writeProtect/>
</special>
```

Figure 10. The <special> Element from "config" XML Code Snippet

3.3.1.6 The XML Element Relationships

The diagram for the relationships are seen below. Each line breaks down an element even further. For instance the special element in the top hierarchy has sub-elements in the far left bot.

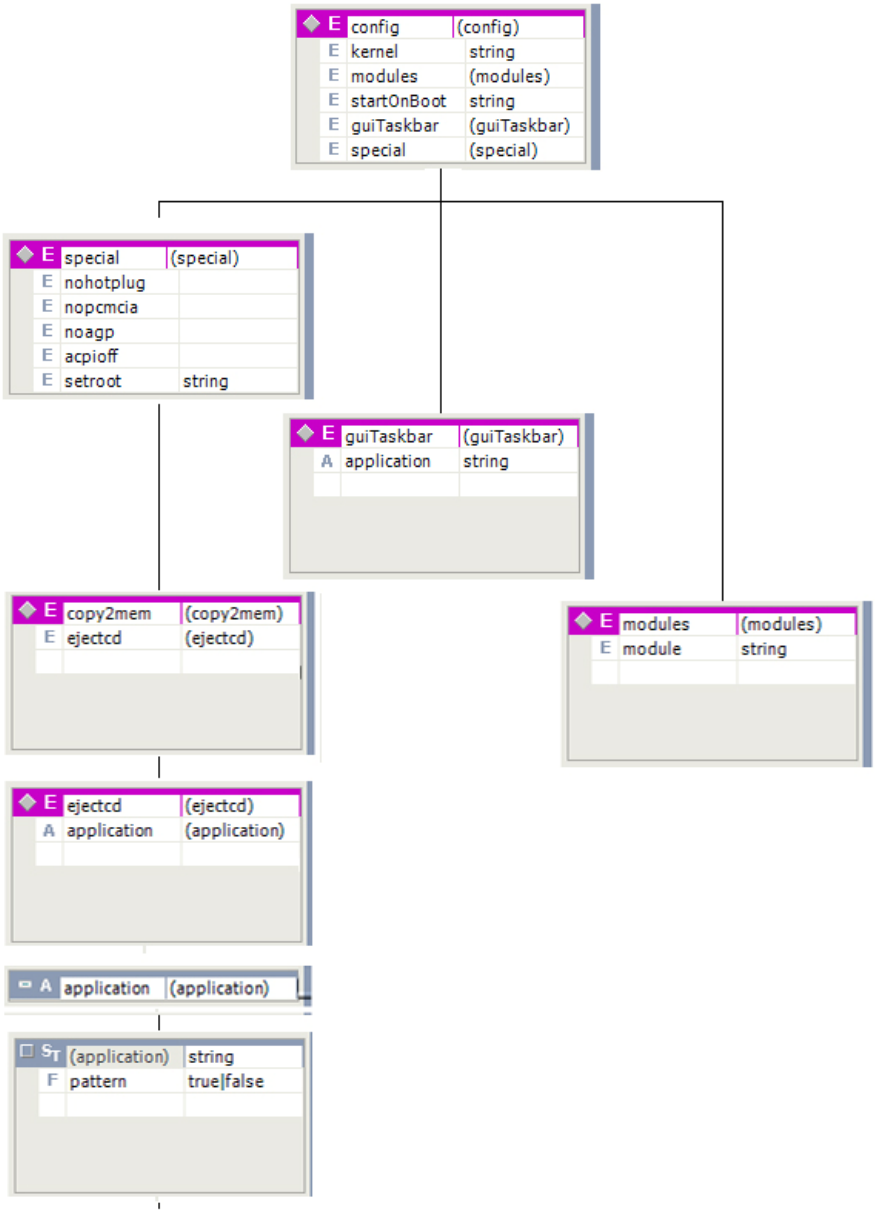


Figure 11. The Configuration Element Relationships within the XML File

3.3.2 FENIX Directory Structure

The FENIX directory structure has been discussed in several areas of this document and I have demonstrated how this directory is used. Although I have discussed Reburn in several sections of this paper, I wish to show the actual Reburn process used to build the FENIX application. Below is the opening page to Reburn. The first few items that stand out are the navigation panel on the left. This navigation panel allows a user to navigate to a particular part of the linear process at any time during the building and customization process. This item is global.

The other global section to the application is the next and back buttons. Those buttons are used by the user to navigate through the process in the order expected. In most cases the user should use these buttons to pass in between the different customization procedures. In between the back and next buttons is a graph used to determine how far along in the customization process a user is. Since this is a screen shot of the first page, the user is at 0%.

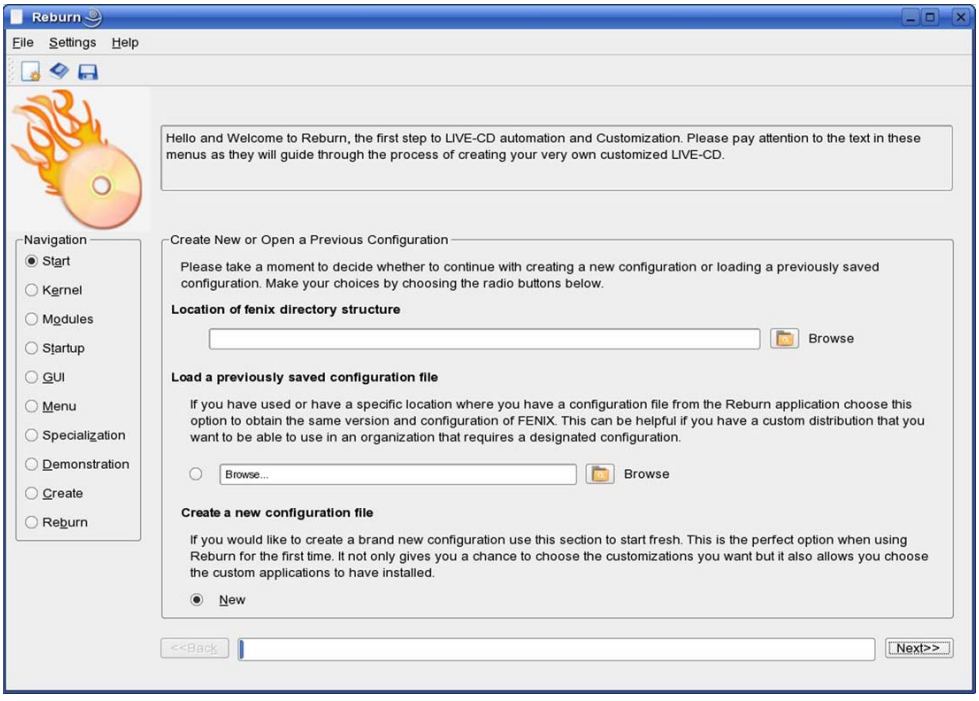


Figure 12. The Start Reburn Menu

The top blank field is the field that is to show the path to the FENIX directory structure. A browse panel on the right allows a user to quickly navigate and find the directory structure to be used. The next blank field that contains the text, “browse…” relates to finding and loading a configuration file. As I mentioned in the previous sections, the configuration file helps to store all available options that a user made in order to reproduce the exact same FENIX version.

3.3.3 Kernel Selection

Once the user clicks the next button a new page comes up. This new page is a selection for kernel choices. This menu is a bit different as there is a rather prominent table being displayed. Above the table is a button. The button is used to connect to a server. If the server is down or for some reason the application is unable to connect to the server a warning will be displayed. If the database does connect the text, “Connected to Database” is shown on the upper left hand corner above the table.

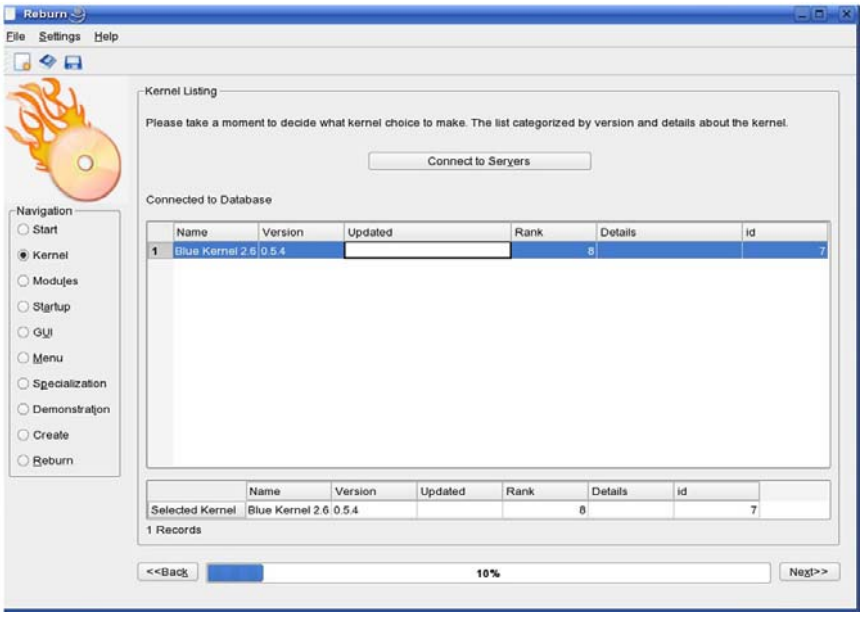


Figure 13. The Kernel Reburn Menu

Once the database has connected to the SQL server a list of kernels are populated into the table. A list is shown and the user will click on one. Once a kernel is highlighted it will be displayed beneath the main table in yet another table. This table has only a single row labeled, “selected kernel.”

3.3.4 Module Selection

Once the next menu appears, a table on the left is displayed. This table contains a list of all modules available for the FENIX OS. A user may search and refine their search with tools shown above the tables. Above the left table a category refinement tool is available. This option will return all modules listed under the category. Above the right table a search field is available. The search field will search through the details of each of the entries and return those that have a rough match.

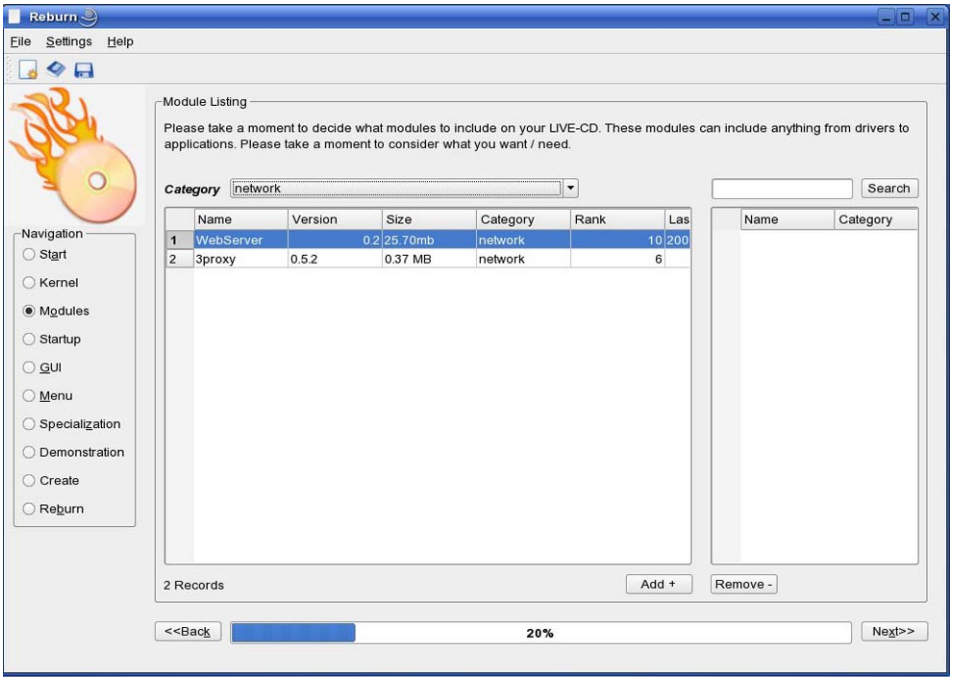


Figure 14. The Modules Reburn Menu

Beneath the two tables are two buttons, “Add +” and “Remove –“. The add button will add an entry from the left table and place it into the table on the right. The remove button

will remove the highlighted entry in the right table. Once the user is done with this page the information stored within the table is kept safe within the internal data structures of the application and used later.

3.3.5 Startup Configuration

The startup configuration page, seeks to help users with their ability to automatically run servers at boot time. At this point in the configuration, Reburn has to do a unique series of processes. Once the user clicks the “Scan for Startup Applications” button, Reburn takes all modules that have been selected in the previous section, mounts them in a temporary folder and unions them. Once this task is complete Reburn will scan the startup application scripts and determine what scripts are available. Upon successfully scanning the applications a list is returned and shown in the table on the left. The user then looks through the list of services, highlights his preferences and selects add. This addition to the right hand table will ensure their ability to start as FENIX loads.

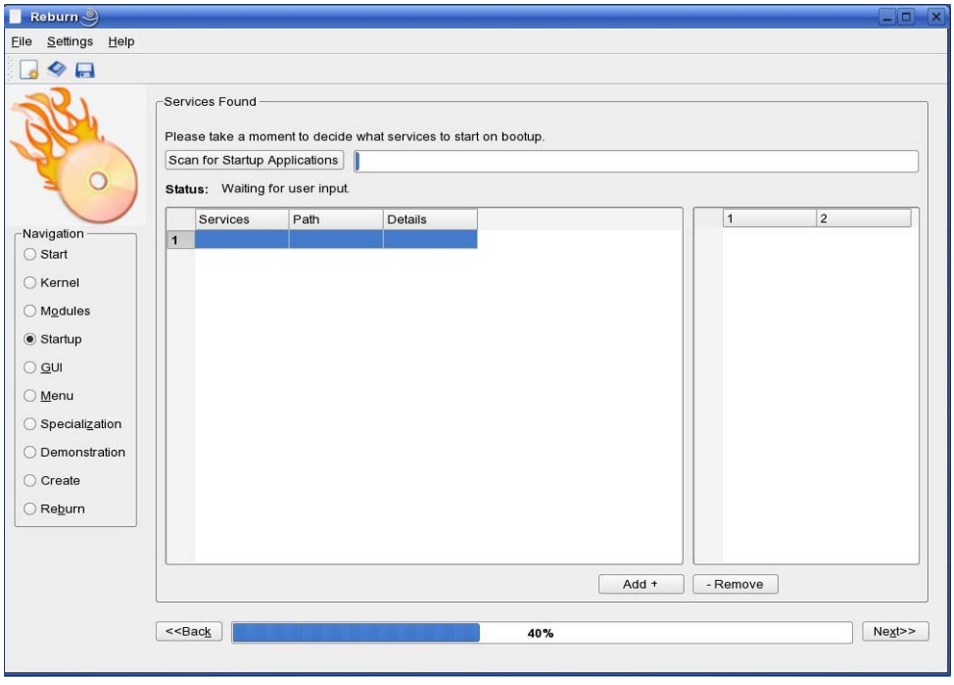


Figure 15. The Service Startup Reburn Menu

3.3.6 KDE GUI Customization

This section was originally planned to be used for enhancements to KDE. However, after some research it was decided that enhancements for KDE was better left in the modules that can be easily downloaded and installed. Instead this section is used for adding user files to FENIX. This can be useful if the user wishs to add backgrounds, songs, programs, investigation forms, or even manuals on how to follow a certain set of procedures. The add and remove buttons work in the common manner. The add button will open a browser dialog to help you locate files on your hard drive. The remove button will remove any item that is highlighted in the table.

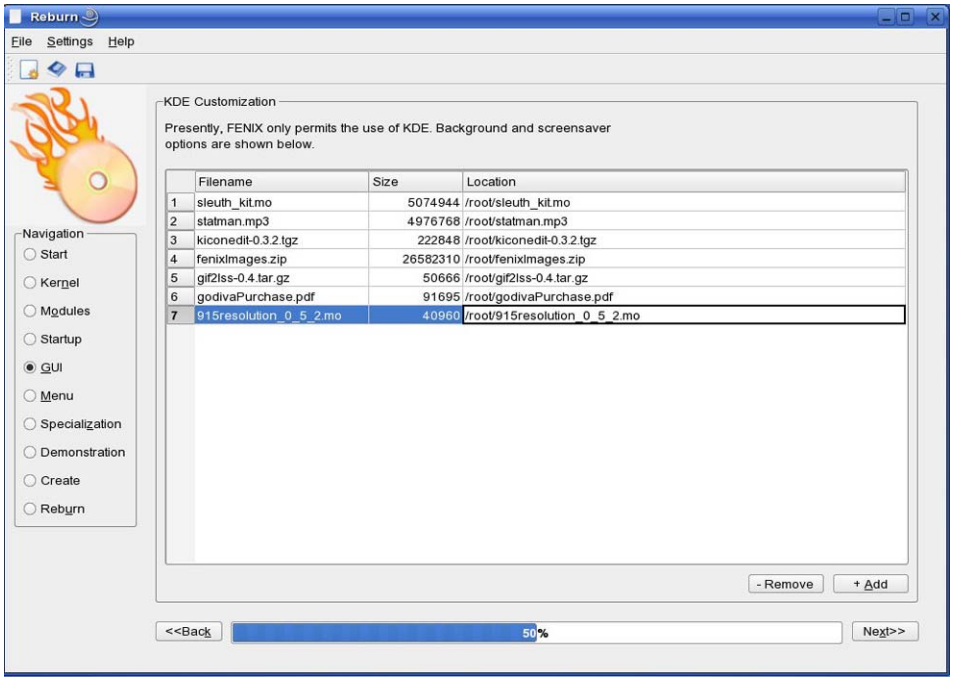


Figure 16. The GUI Configuration Reburn Menu

3.3.7 Taskbar Configuration

At this point the taskbar configuration ability is quite limited and is used only for updating the icons and links on the taskbar for FENIX. This menu is shown below. The image on the top helps to illustrate where the links will go once they are configured. There is a maximum of four links that can be customized. The other two are reserved for the shell and the KDE menu link.

As shown each link has a path to the application and a path to the icon. Both of the paths to the applications and graphic files should be within the virtual CD and not the actual hard drive from which FENIX is being built.

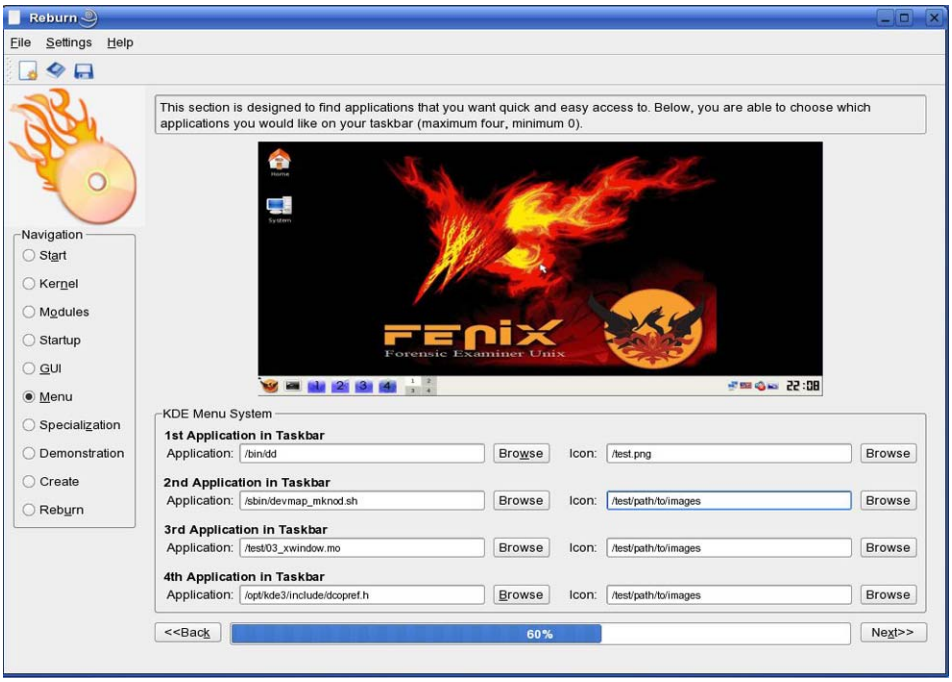


Figure 17. The Taskbar Configuration Reburn Menu

3.3.8 Specialization

The user is now able to specialize the purpose of their LIVE OS. FENIX is intended to be a forensic OS. However, in a forensic lab all types of equipment are necessary. This project has the ability to take on many of these purposes including DVD viewing, forensic acquisitions, system recovery and Windows emulation. Since LIVE OSs offer so many features, profiles have been made to give novice users an easier helping hand in some of their decisions. While this might be alright for novice users, the options are all still singly there for the more advanced users who wish to get intricate with the details of their OS. A screen shot of this menu can be seen below.

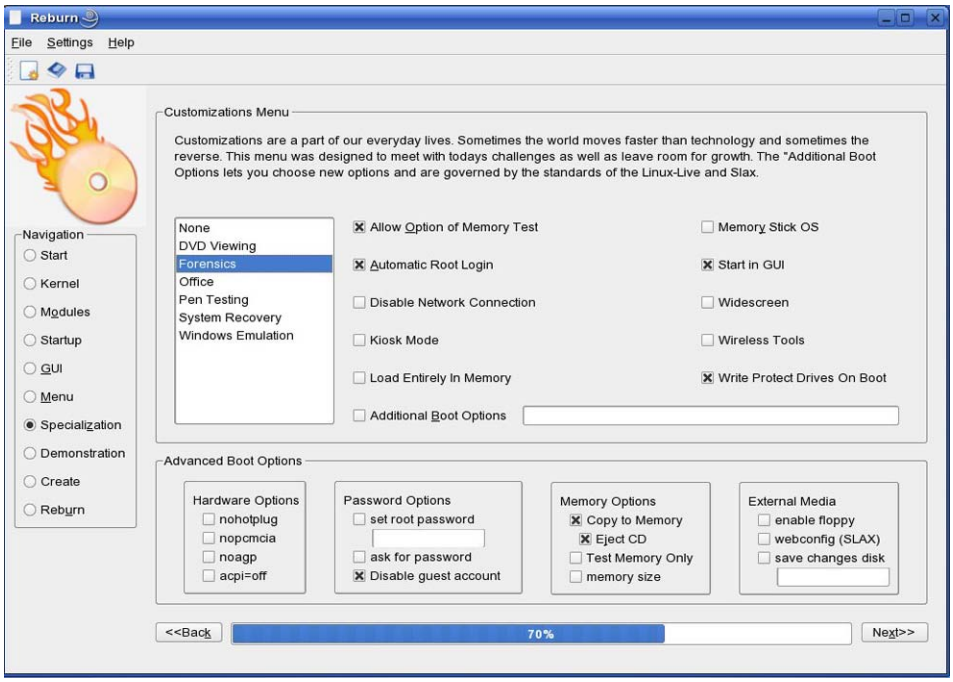


Figure 18. The Specialization Reburn Menu

3.3.8 Image Creation

Once the user has reached this section, all customizations have been completed and it is finally time to begin creating the OS. This menu is designed to help the user find a place to build and store the FENIX OS. The only available format at this time is .iso, unfortunately. However, this is the primary format for images at this time and is a good format.

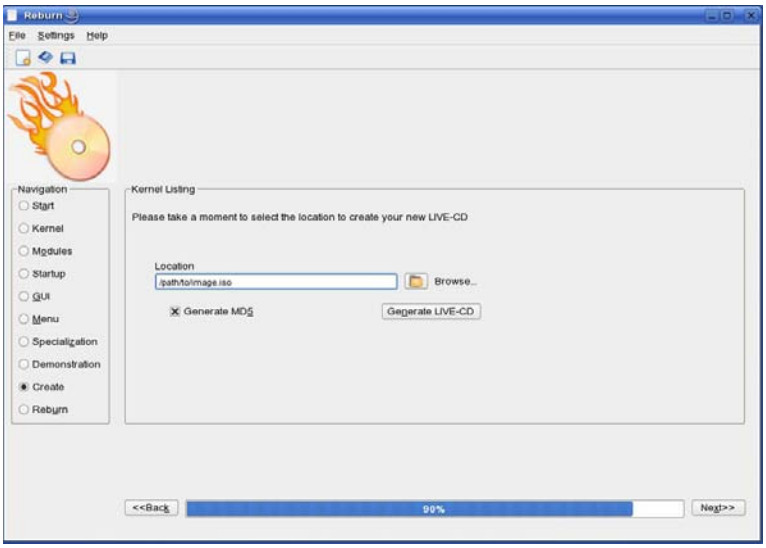


Figure 19. The Image Creation Reburn Menu

3.3.9 Image Demonstration

Once the image has been created a user can test the build with a demonstration section. This section heavily relies upon external use of Qemu. If Qemu is not installed on the machine building FENIX a demonstration will not be run.

Figure 20. The Demonstration Reburn Menu

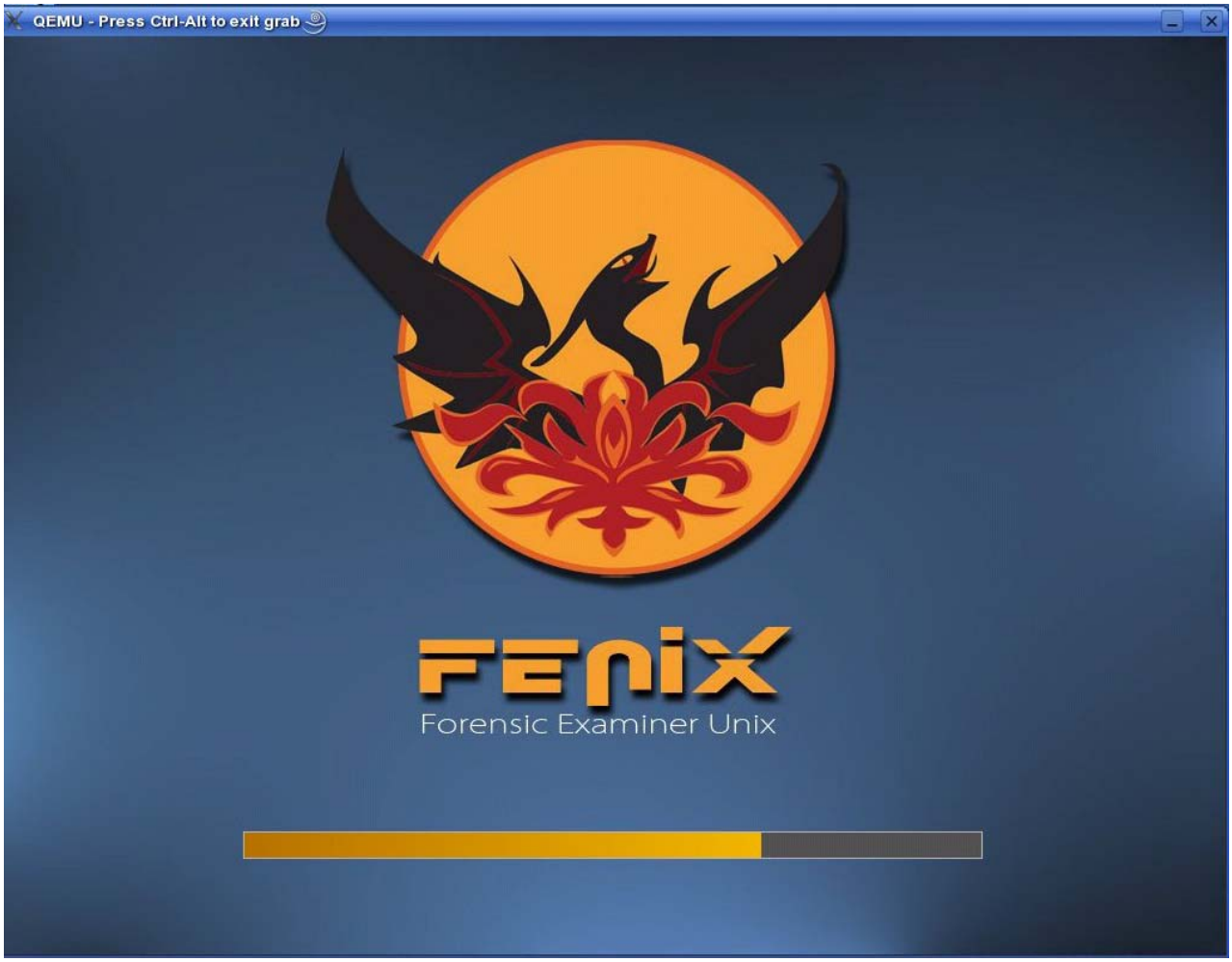


Figure 21. Qemu Loading the FENIX Image

If the user has Qemu installed and clicks the button labeled, “Click for Demonstration,” a window will popup immediately like the one below. FENIX will go through its bootup process where the user can see if the OS is booting how he wants it to boot. If the user is unsatisfied with the OS, he can close the window, click back a few times until he is a the modification screen. A couple of mouse clicks later, the user is back to the demonstration phase and seeing how the new build runs.

Figure 22. Qemu Emulating FENIX

3.3.10 FENIX Production

Once the user is satisfied with their creation, the Qemu window will be closed and the user can click on the next button. In this menu there are only a few options. The first of which is choosing whether to verify the burn and to save the configuration file. Upon clicking burn the configuration file is saved, if the option is requested, and the CD begins to burn. Once the burn is complete the user decides whether to return to the customization process or to end the program with the “Finish” button.

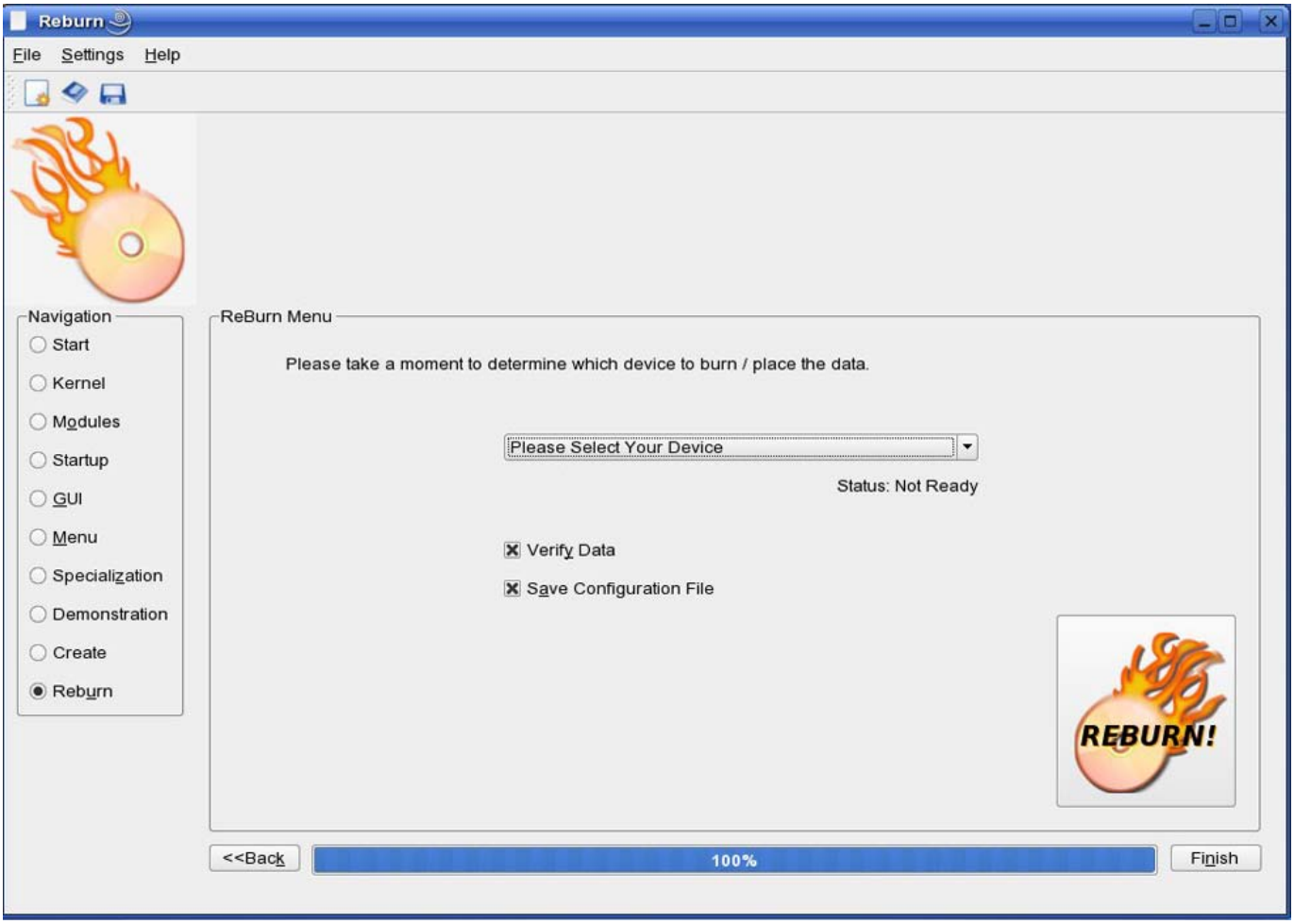


Figure 23. The Image Burning Reburn Menu

3.4 Client Server Structure

Aspects of FENIX are run in a very basic client server model. During the course of the Reburn build process information gets accessed by a remote client that is allotted read only access to a server that remains in a secure environment. This information that is shared with remove clients is uploaded and placed onto the server using a local read-write access account that is provided to users who sign up for the website on the webserver running on the database server. Since these users have been given access to the website through information

provided by them in a registered account, most of these accounts can be considered to be reliable and accountable for their actions.

In addition to this two-fold access to the webserver and database server, FENIX utilizes a few remote administration options that allows forum administrating and module review. These access privileges are handled through the website in the same fashion that regular users are allotted.

3.4.1 Client Server Interactions

The client server interactions that are maintained for FENIX are actually used in the build phase of Reburn. During the 2nd, 3rd , 4th, and 8th stages of Reburn's linear building process server access is managed to control and report data. During the 2nd stage of the process Reburn makes a call to fenix.iseage.org to list kernels located in its database. Upon response the data is read and displayed in a table in order for a user to choose a kernel desired.

During the 3rd stage of the build process a similar interaction occurs but this time the client is able to adjust the requests by category and adding wildcard searches into the SQL database requests. This allows the user to minimize the amount of traffic made between the client and the server.

During the 4th phase all kernels and modules are downloaded and placed into a directory. Unfortunately, this interaction is implemented at a huge expense to the system and may be a cause for concern at a later date if many users are logged into the system at one time.

During the 8th phase if the modules and kernels were not downloaded for the startup scan then the modules and kernels need to be downloaded before the disk image can be made. Again this is at a heavy burden to the server and should be noted as a potential

downfall in the design of this model. Since this model is being used as only a test for the initial design it should not affect performance on a single person usable system.

3.4.2 Client Server Website

The client server website is a basic website design with apache taking most of the traffic and application level work. A user accesses the webpage and downloads informational pages. After some time the user logs into the website using a secure channel and is authenticated against a mySQL database. The mySQL database also stores information regarding the modules for FENIX.

Once the user is logged into the website, the user has advanced features which allow him to navigate special areas of the website such as forums, as well as upload and manage modules that are currently available to other authenticated parties. These abilities require that the user have read-write abilities to the database. Fortunately, the webserver is on the same host as the mySQL server and one can restrict the account to only functioning on the localhost.

3.4.3 Client Server Security

The security of the client server for the mySQL tables that hold all modules and authentication information for users of the FENIX webpage is as simple as restricted access. Users of the Reburn program have one account that allows only read-only access but allots the permission of remote access to the information. The read-only access is allotted to only the tables that hold information about the modules for FENIX.

The users of the website have read-write access but are restricted to only using that account locally. This restricts the ability for an attacker to user a read-write account remotely. In addition to restricted access to the mySQL database, the php input parameters are

validated using PHP's "addslashes" command which will take any special chars and make them non-special in PHP statements.

3.5 FENIX Review

The FENIX project is a complex model from which the components are a series of models, applications, and client server communications all with the single goal of creating a highly customizable LIVE OS that is easy to create, use and is highly portable. The FENIX project accomplishes this goal by using several experimental designs along with several long standing principles and models that have been tested. FENIX has borrowed a few design and methods from common forensic tools and OS design methodologies. A great deal of FENIX design is due to current LIVE OS options that are available presently. However, it seeks to remove their shortcomings by adding other efficient alternatives.

Several of the underlying technologies that make FENIX able to succeed at its goals are the file system technologies (squashfS, unionFS, and tmpFS), Reburn (the linearly designed customization utility), and the back end client server database structures. When these technologies were fused together the FENIX project was born.

CHAPTER 4. TESTING

This section deals with testing the FENIX project. The study of the NIST document outlining testing procedures on forensic tools has become the basis for the testing of FENIX. As such most of the criteria will be used in the testing of FENIX.

4.1 Introduction

NIST provided several very important ideas in their quest for a standardized testing methodology. To recap, NIST introduced the idea of a testing process. It is the goal of FENIX to abide by this process in order to display the finalized test results. The phases that were important within the process are:

1. establish categories of forensic requirements,
2. identify requirements for a specific category,
3. develop test assertions based on requirements,
4. develop test code for assertions,
5. identify relevant test cases,
6. develop testing procedures and method,
7. report test results.

4.1.1 Objective Components

The objective components, or the ones that can be proven by factual data, are some of the more important aspects that are available to legal and investigator entities. This is the section that should be read carefully and studied in order to make sure that the forensic nature of the OS can be used in the court of law.

The key objective components to be tested are the ones that are stated in S. Mocas's paper, ``Building Theoretical Underpinnings for Digital Forensics Research." [1] This document highlights upon the core forensic tool guidelines that should be followed. Those are integrity, authentication, non-interference, minimalization, and reproducibility. Those are attributes that a forensic tool should have in order to uphold key standards in forensics,

however, I have added another standard that is chosen for future work. This new property is flexibility. All of these properties are discussed below in greater detail.

4.1.1.1 Integrity

Mocas defined a digital form of integrity as duplication integrity: which assures that, given a data set, the process of creating a duplicate of the data does not modify the data (either intentionally or accidentally) and the duplicate is an exact bit copy of the original data set [1]. FENIX however does not directly deal with any file acquisitions and uses currently available software applications to handle evidence. As such, integrity does not to be directly tested. However, to ensure that FENIX does not inherently corrupt data, the following requirements and assertions are necessary:

1. FENIX shall not write to the evidence
    a. Test case: attempt to write to the evidence, determine if possible
2. FENIX shall not change acquired evidence
    a. Test case: read the evidence through multiple devices, test checksums before and after to determine if FENIX modifies images
3. FENIX shall be able to read digital evidence from hardware devices
    a. Test case: acquire images of hardware evidence through multiple applications

4.1.1.2 Authentication

According to Mocas, the legal definition of authentication is, “knowing that the electronic evidence is what its proponent claims.” In this definition the tests for FENIX should include both tests for Reburn, FENIX, and the client server authentication. The requirements, assertions, and test cases are listed below.

1. Reburn shall create the LIVE image defined by the user.

   a. Test case: Several variations of configurations will be made and tested to see if FENIX was made corresponding to the image.
2. FENIX shall acquire authentic disk images without changing dates, times, or data.
   a. Test case: Autopsy will be used to acquire an image and checked for accurate dates.
3. Only authenticated users will be able to access database information
   a. Test case: A series of accesses will be attempted to the mySQL database.
4. Only privileged webpage users will be able to upload information to the database.
   a. Test case: A series of write procedures will attempted through the webpage without being logged in.
5. Only authentic information will be able to be added to the database.
   a. Test case: A user will attempt to upload information. The information shall not be directly uploaded and passed through a moderator.

4.1.1.3 Non-interference

Mocas defines identifiable interference as, “assuring that when the method (or tool) used to gather and / or analyze digital evidence does change the original data set that the changes are identifiable.” [1]. In this definition the tests for FENIX should include both tests for FENIX and one of its many applications for acquiring an image. The requirements, assertions, and test cases are listed below.

1. FENIX shall not change any data in the process of acquiring it.
   a. A simple disk acquisition shall be made and tested to see if any data was modified.

4.1.1.4 Minimalization

Minimalization allows the investigator to carry out his job without disrupting the jobs of others. Minimalization: assuring that the minimum amount of data was processed (seized and / or examined). This allows for the impacted service to have minimal damage due to the attack and ensuing investigation.

In this definition the tests for FENIX should include both tests for FENIX and its applications. The requirements, assertions, and test cases are listed below. Since this section is more closely related to an application, this will not be tested against FENIX.

4.1.1.5 Reproducibility

In the Daubert ruling [Daubert v. Merrell Dow Pharmaceuticals, 509 U.S. 579 (1993)], used by federal courts to determine the admissibility of expert opinion testimony, is in part based on being able to reliably test the merits of scientific evidence [1]. In this definition, the tests for FENIX should include both tests for Reburn and FENIX. The requirements, assertions, and test cases are listed below.

1. FENIX shall be able to reproduce results following the same process.
    a. A series of double acquisitions shall be made to determine if the same results occur, showing reproducibility.
2. Reburn shall be able to reproduce results following the same process.
    a. Reburn, given the same inputs shall reproduce the same image. The image will be checked via an md5 hash.

4.1.1.6 Flexible

It is my belief that forensic tools should be flexible to new technologies and other software entities. In the case of operating systems, it should be able to have any software on it and it should be easy to update. In most LIVE OSs to date this is not available. This

presents a problem as more tools are coming out daily and different tools are more popular in different areas of the world. Therefore, FENIX needs to be flexible. The requirements, assertions, and test cases are listed below.

1. Reburn shall be able to be easy to modify or change when a user wants to customize their disk. The applications and kernels available for FENIX shall be easily updated or modified with the newest in tools.
    a. FENIX will be configured and tested to see if the proper applications were added upon customization.

4.1.2 Subjective Components

Unfortunately, NIST's testing methods are only good for objective testing. In cases where a users perspective come into question a new set of testing needs to be carried out. For the subjective testing, FENIX will rely on a great deal of polling. The polling will be carried out in five areas of questions. To reduce the level of subjectivity, the questions will be asked in groups of five true/false questions.

In addition to the five question areas, there will be three levels of users. The users will be classified as expert, intermediate, and novice. The classification levels are used to separate the differing opinions of users. In appendix D, a sample poll can be seen.

4.1.2.1 Value

The value that a user places on an operating system or series of utilities is essential in getting a user to start using it. If FENIX is not seen as having any value to an organization or group of people it will not be used. As such, it is important to find out how many people will value FENIX in their organization, work place, or home.

The most important aspects of FENIX's value that are questioned relate to if a user would use the project. This is an essential part of the project and therefore should be important to test whether FENIX has a potential for an ongoing status of research.

4.1.2.2 Easy to Use

FENIX was designed for both advanced and novice users. The key idea is that FENIX is to be made for everyone. As an underlying requirement it is important to question users in regard to the ease of use for the FENIX Project. This line of questioning includes both Reburn and FENIX.

Since Reburn was designed in a simple and easy to understand manner, it is important that the interface is perceived that way. In that respect the majority of the questions are with respect to Reburn rather than to FENIX.

4.1.2.3 Live CD Competition

The LIVE CD race is fast-paced competition between research teams and the public. FENIX is developed off of SLAX technology with forensic design goals that have not yet been implemented into the technology. This is an area that has yet to be tested and developed.

Unfortunately SLAX and FENIX are late in the competition of LIVE CD's. Most users have already sided with Knoppix or Helix. One of FENIX's goals is to migrate users from a locked down set of tools to a more customizable setting by borrowing some of the users from Helix or Knoppix. In addition there are plenty of users out there with commercial tools that have a legion of followers. It is essential that some of these users migrate to FENIX's structure in order for FENIX to show some presence in the LIVE CD world.

4.1.2.4 Effective

Another particular aspect of FENIX that is less superficial than usability or popularity deals with the essence of a user's perspective on if FENIX does what it promises and how well it does it. In addition there are concerns of how Reburn seeks to maintain a level of ease of use but is to too easy at the cost of more boot options. In this line of questioning, I hope to understand more about the perceived effectiveness of FENIX.

4.1.2.5 Security

FENIX is meant to function as an after-the-fact security tool for investigators but can also be expanded to new levels with newer users seeking different purposes. It is in this area that sometimes a tool can be called into question. For one person, nmap and utilities of the like may be used as a security tool. For another the same applications could use for malicious purposes. This area needs to be thought about and every user should have a voice in that.

In addition to the outright purpose of FENIX, the project should be questioned. Does Reburn create security problems? Does FENIX? Does the website? All of these objects within the project should be looked at and questioned.

CHAPTER 5. RESULTS

In addition to the process stated above, NIST listed what should be listed in the report. The listing outline is shown below, for greater detail.

1. a title stating what product was tested
2. identification of the testing environment, i.e., where the tests were run;
3. unique identifier for the test report; [identifier will be repeated on each page in order to ensure that the page is recognized as a part of the test report]
4. the name and address of the vendor;
5. identification of the testing software used;
6. unambiguous identification of the product tested including version, patches, etc.;
7. the test with the criteria for measurement;
8. the name(s), function(s) and signature(s) or equivalent identification of person(s) authorizing the test report;
9. where appropriate and needed, opinions and interpretations;
10. additional information which may be required by specific methods, clients, or groups of clients.

The rest of this report will be carried out as in the above stated NIST approved methodology.

5.1 Introduction

The product to be tested will be the FENIX Project. This includes the FENIX website, the Reburn application, and the LIVE OS: FENIX.

5.1.1 Environment

The environment consists of two computers. The first computer boots from FENIX. The second computer is running Windows XP with the commercial software, Encase. A set of tools were used on the FENIX OS, which will be covered in the software section, below.

Several important facts should noted when discussing the environment. Most notably, the second computer used during testing was there to run Encase only for the integrity checking and reproducibility. The two computers were housed within the same network, away from the Internet.

5.1.2 Vendor Information

FENIX was developed at Iowa State University under the direction of Dr. Doug Jacobson and Sergeant Aaron Delashmutt of investigations at Iowa State University's Police Department. The FENIX project was designed, maintained, and approved as the thesis for Sean Howard.

5.1.3 Testing Software

The testing software used for FENIX were all forensic tools that have proven themselves in real world situations. The following software was used:

- Encase (Guidance Software)

  Encase is a forensic software suite that is used in conjunction with forensic labs that do their lab work on Windows PC's. Encase, at this point in time, is the most widely used commercial tool for forensic acquisitions and analysis.
- linen (Guidance Software)

  Linen is a tool within the Encase suite that is used to acquire image information from a Linux machine.
- Autopsy Forensic Browser (Brian Carrier)

  Autopsy Forensic Browser is the graphical web front to the Sleuth Kit (TSK). It is extremely helpful in navigating acquired devices.

- The Sleuth Kit (Brian Carrier)

  The Sleuth Kit (TSK) is a series of tools that were put together to forensically acquire and analyze devices for forensic investigations. TSK runs in Linux.

- dcfldd (Department of Defense Computer Forensics Lab)

  dcfldd is an enhanced version of GNU dd with features useful for forensics and security. Based on the dd program found in the GNU Coreutils package, dcfldd allows you to hash on the fly, updates the user of its progress during its hashing, bit for bit verification, output to multiple files at the same time, and logging of operations.

- GNU Image Manipulation Program-GIMP (GNU)

  GIMP is a program that is used to edit photos and create images. It was only used for reporting of information during these tests.

5.1.3.1 Operating System Testing

The operating system testing will be done to insure that the OS does not obscure or modify any data, non-interference, authentication, or integrity. In certain test cases the trial will be run in three different applications (dd – for a low level test, autopsy – for a high level test, linen – for a remote test).

5.1.3.1.1 Integrity

1. FENIX shall not write to the evidence
    a. Test case: attempt to write to the evidence, determine if possible

FENIX boots into Linux as any OS normally does, except for FENIX does not auto mount drives into the read-write mode that most do. FENIX makes the user mount his or her own drives. Since not all drives are required during the acquisition this helps save overhead for the system. During this test the following procedure was followed.

Three drives were inserted into the computer's USB ports and hashed using linen. It is important to note that the drives were NOT mounted. Linen allows the computer to analyze the drives without mounting them from the user perspective. Once the hashes were recorded, the user then mounted the drives into read-only status. The hashes were then recorded to verify that the drives hashes did not change from being mounted. After this stage was complete, it was tested to see if a user could write to the evidence using the simple touch command. The results are shown below.

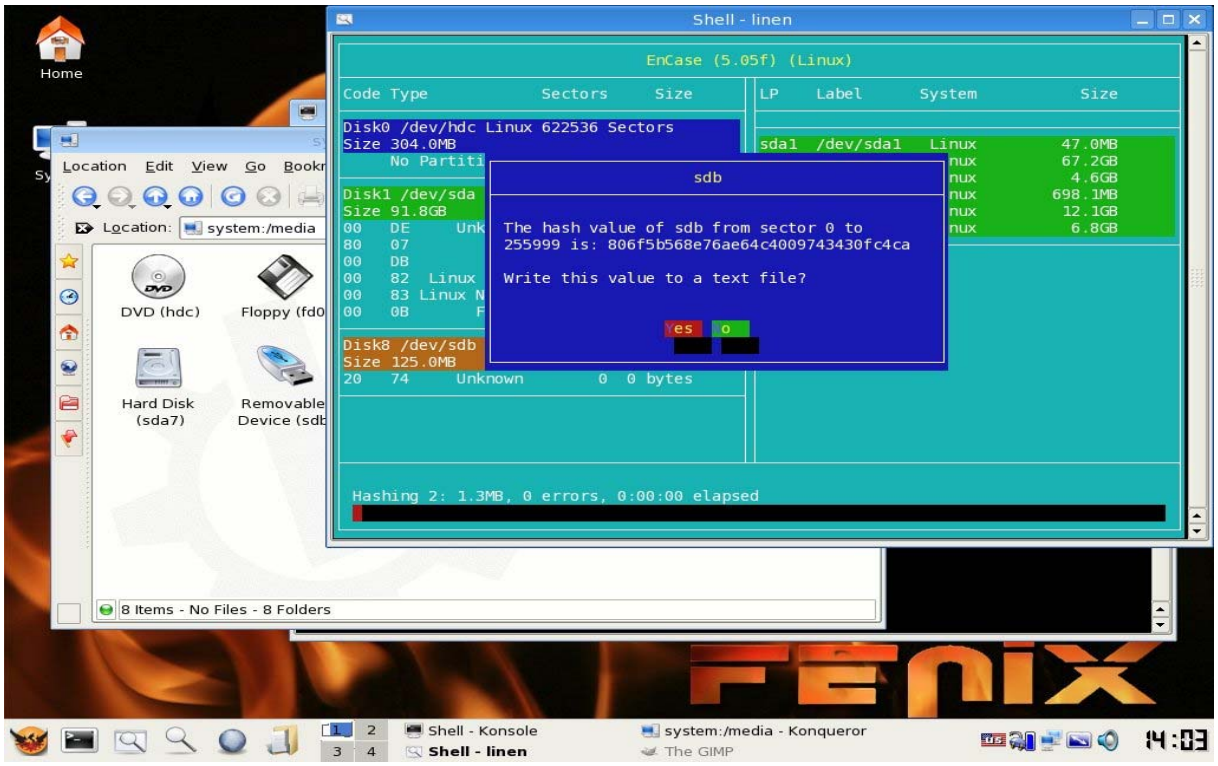


Figure 24: Linen in FENIX showing hash value of /dev/sdb (pre-mount)

Linen is the program running in the teal blue terminal window. The hash reported is: 806f5b568e76ae64c4009743430fc4ca. The drive was then mounted, linen was then run again on the mounted drive with the hash being reported as: 806f5b568e76ae64c4009743430fc4ca. This is seen in the image below.

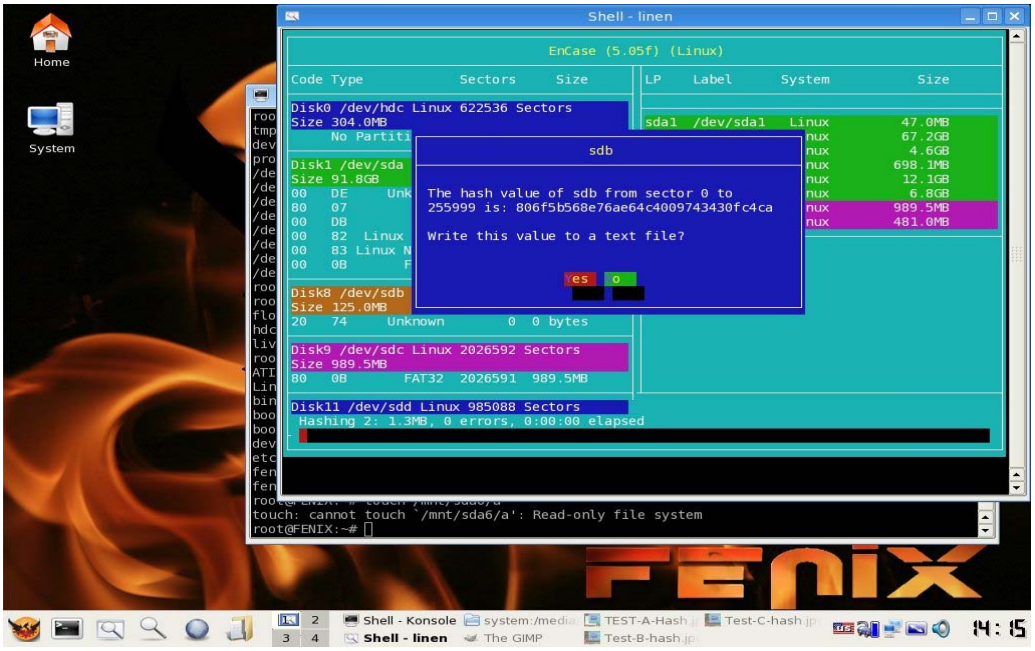


Figure 25: Linen in FENIX showing hash value of /dev/sdb (post-mount)

The screen captures are from one run of three tests. The rest of the tests may be seen in Appendix A. The run results are shown below:

Table 1: Devices with their hash values, before and after being mounted

Device	Size	Hash1 (pre-mount)	Hash2 (post-mount)
/dev/sdb	128mb	806f5b568e76ae64c4009743430fc4ca	806f5b568e76ae64c4009743430fc4ca
/dev/sdd	512mb	23171bf0655f66e3eed28aceff7a7ce6	23171bf0655f66e3eed28aceff7a7ce6
/dev/sdc	1024mb	c1779f5982db344d47e78504673a1436	c1779f5982db344d47e78504673a1436

2. FENIX shall not change acquired evidence
    a. Test case: read the evidence through multiple devices, test checksums before and after to determine if FENIX modifies images

3. FENIX shall be able to read digital evidence from hardware devices
    a. Test case: acquire images of hardware evidence through multiple applications

It is important for evidence to not change once they have been acquired or during their acquisition. In this instance, FENIX will acquire images using two different methods and record their hash values to determine if they have been modified in any way. In addition to completing tests for the requirement for number 2, this test shall prove number 3.

The first method for acquiring an image and reading its hash was done by using Encase. Through linen, Encase acquired the test data over the network and stored it on computer 2, listed in the environment. The hash value of device sdc is shown below.

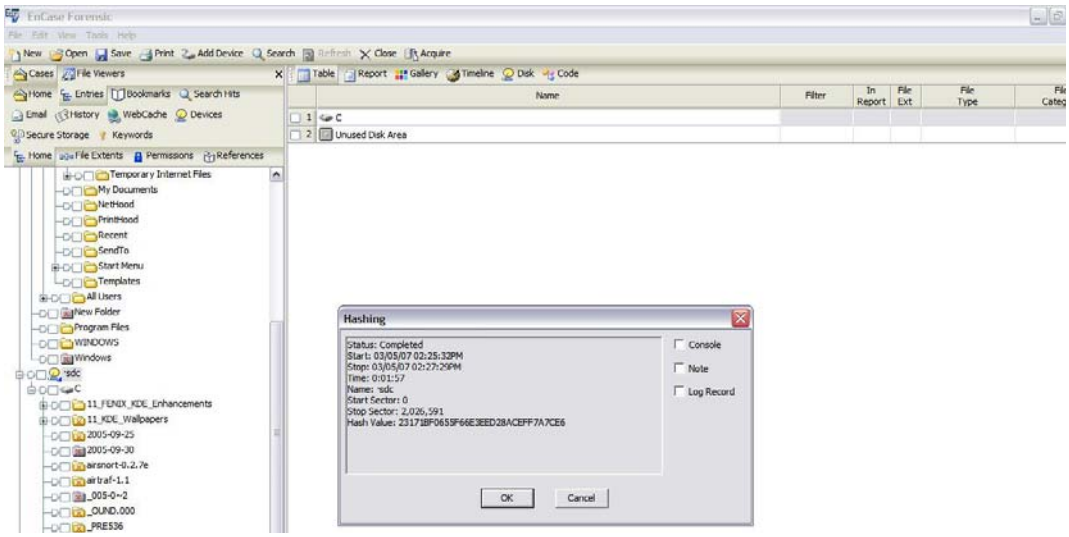


Figure 26: Encase acquired hash value

Encase shows that the hash found is: c1779f5982db344d47e78504673a1436. In the image shown below, Autopsy finds that the hash found is: c1779f5982db344d47e78504673a1436.

A

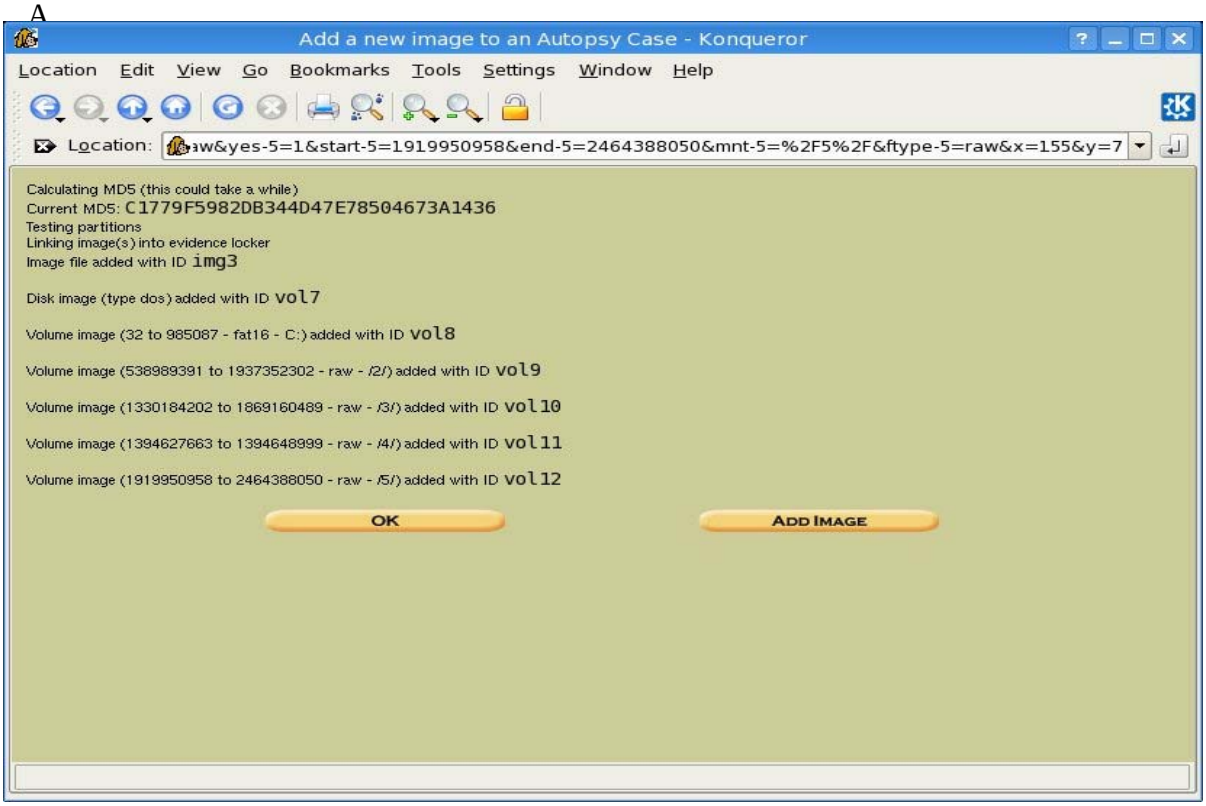


Figure 27: Autopsy acquired hash value

1

he screen captures are from one run of three tests. The rest of the tests were run identical and in series. The rest of the tests may be seen in Appendix A. The run results are shown below:

Table 2: Hash values corresponding to both pre-acquisitions and post-acquisitions

Device	Size	Hash1 (pre-acquisition)	Hash2 (post-acquisition)
/dev/sdb	128mb	806f5b568e76ae64c4009743430fc4ca	806f5b568e76ae64c4009743430fc4ca
/dev/sdd	512mb	23171bf0655f66e3eed28aceff7a7ce6	23171bf0655f66e3eed28aceff7a7ce6
/dev/sdc	1024mb	c1779f5982db344d47e78504673a1436	c1779f5982db344d47e78504673a1436

5.1.3.1.2 Authentication

1. FENIX shall acquire authentic disk images without changing dates, times, or data.
    a. Test case: Autopsy will be used to acquire an image and checked for accurate dates.

In this instance, two different acquisitions will be tested for their dates and times on the same images to determine if they report the same information. As in the earlier tests, 3 different devices will be used to test. The two acquisition software applications will be Autopsy from the Sleuthkit by Brian Carrier and Encase/Linen by Guidance Software. We will look specifically at device /dev/sdd, so as to not have redundancy. The shortened results are shown below and the complete results are shown in Appendix A.

	Name	Last Written	Last Accessed	File Created	File Ext	File Type	File Category	Filter
1	_MLINU~1.PAR	10/26/06 01:17:42AM	10/26/06	10/26/06 01:17:42AM	PAR			
2	_NITRD~1.PAR	10/26/06 01:30:22AM	10/26/06	10/26/06 01:30:22AM	PAR			
3	bootsplash-1024x768.jpg	10/25/06 04:40:46PM	01/03/07	10/25/06 04:40:40PM	jpg	JPEG	Picture	
4	bzImage	10/26/06 12:32:30AM	10/26/06	10/26/06 01:17:42AM				
5	ini-uncomp.gz	10/26/06 01:32:00AM	10/26/06	10/26/06 01:32:20AM	gz	GZIP Compressed	Archive	
6	initrd-Given.gz	10/26/06 01:29:22AM	10/26/06	10/26/06 01:30:22AM	gz	GZIP Compressed	Archive	
7	initrd-My.gz	10/26/06 01:28:32AM	10/26/06	10/26/06 01:30:22AM	gz	GZIP Compressed	Archive	
8	initrd.gz	10/25/06 08:34:46PM	10/25/06	10/25/06 08:37:52PM	gz	GZIP Compressed	Archive	
9	initrdX.gz	10/25/06 08:38:22PM	10/25/06	10/25/06 08:38:52PM	gz	GZIP Compressed	Archive	
10	linux-live-5.5.0	10/26/06 01:25:50AM	10/26/06	10/26/06 01:25:50AM	0			
11	Primary FAT							
12	Secondary FAT							
13	silent-1024x768.jpg	10/25/06 04:36:16PM	01/03/07	10/25/06 04:36:10PM	jpg	JPEG	Picture	
14	squashfs	10/26/06 01:24:24AM	10/26/06	10/26/06 01:24:24AM				
15	Unallocated Clusters							
16	unionfs	10/26/06 01:24:34AM	10/26/06	10/26/06 01:24:34AM				
17	vmlinux.bin	10/26/06 12:32:30AM	10/26/06	10/26/06 01:17:42AM	bin	Raw Binary Format	Windows\DOS	
18	Volume Boot							
19	Volume Slack							

Figure 28: Dates and Times in Encase Acquisition Suite

DEL	Type dir / in	NAME	WRITTEN	ACCESSED	CREATED	SIZE
✓	r/r	_MLINU~1.PAR	2006.10.26 01:17:42 (GMT)	2006.10.26 00:00:00 (GMT)	2006.10.26 01:17:42 (GMT)	0
✓	r/r	_NITRD~1.PAR	2006.10.26 01:30:22 (GMT)	2006.10.26 00:00:00 (GMT)	2006.10.26 01:30:22 (GMT)	0
	r/r	bootsplash-1024x768.jpg	2006.10.25 16:40:46 (GMT)	2007.01.03 00:00:00 (GMT)	2006.10.25 16:40:40 (GMT)	191126
	r/r	bzImage	2006.10.26 00:32:30 (GMT)	2006.10.26 00:00:00 (GMT)	2006.10.26 01:17:42 (GMT)	2947181
	r/r	ini-uncomp.gz	2006.10.26 01:32:00 (GMT)	2006.10.26 00:00:00 (GMT)	2006.10.26 01:32:20 (GMT)	1729748
	r/r	initrd-Given.gz	2006.10.26 01:29:22 (GMT)	2006.10.26 00:00:00 (GMT)	2006.10.26 01:30:22 (GMT)	1455416
	r/r	initrd-My.gz	2006.10.26 01:28:32 (GMT)	2006.10.26 00:00:00 (GMT)	2006.10.26 01:30:22 (GMT)	1789662
	r/r	initrd.gz	2006.10.25 20:34:46 (GMT)	2006.10.25 00:00:00 (GMT)	2006.10.25 20:37:52 (GMT)	1349769
	r/r	initrdX.gz	2006.10.25 20:38:22 (GMT)	2006.10.25 00:00:00 (GMT)	2006.10.25 20:38:52 (GMT)	1790501
	d/d	linux-live-5.5.0/	2006.10.26 01:25:50 (GMT)	2006.10.26 00:00:00 (GMT)	2006.10.26 01:25:50 (GMT)	8192
	r/r	silent-1024x768.jpg	2006.10.25 16:36:16 (GMT)	2007.01.03 00:00:00 (GMT)	2006.10.25 16:36:10 (GMT)	249460
	d/d	squashfs/	2006.10.26 01:24:24 (GMT)	2006.10.26 00:00:00 (GMT)	2006.10.26 01:24:24 (GMT)	8192
	d/d	unionfs/	2006.10.26 01:24:34 (GMT)	2006.10.26 00:00:00 (GMT)	2006.10.26 01:24:34 (GMT)	8192

Figure 29: Dates and times in Autopsy Acquisition Tool

Table 3: Dates and times shown in both Encase and Autopsy

Filename	Written	Accessed	Created
bootsplash-1024x768.jpg	2006.10.25 16:40:46(GMT)	2007.01.03	2006.20.25 16:40:46(GMT)
bzImage	2006.10.26 00:32:30(GMT)	2006.10.26	2006.10.26 01:17:42(GMT)
ini-uncomp.gz	2006.10.26 01:32:00(GMT)	2006.10.26	2006.10.26 01:32:20(GMT)
initrd-Given.gz	2006.10.26	2006.10.26	2006.10.26

	01:29:22(GMT)		01:30:22(GMT)

On all accounts, both Encase and Autopsy agreed with the values. One note should be mentioned. All dates in both Autopsy and Encase were of GMT value. The offset of time will vary with location of the home of the computer being analyzed.

5.1.3.1.3 Non-interference

1. FENIX shall not change any data in the process of acquiring it.
    a. A simple disk acquisition shall be made and tested to see if any data was modified.

In this test, hashes will be recorded of three different thumb drives both before and after acquisition. If they align, FENIX has succeeded in not modifying the images. This test was part of the 2nd integrity test shown above. You should view that section to understand more about this test.

5.1.3.1.4 Minimalization

No tests are related to FENIX in this section.

5.1.3.1.5 Reproducibility

1. FENIX shall be able to reproduce results following the same process.
    a. A series of double acquisitions shall be made to determine if the same results occur, showing reproducibility.

This test was actually completed in the final test of Integrity. The results can be seen in table 3.

5.1.3.1.6 Flexible

There is no test for this section regarding FENIX specifically

5.1.3.2 Application Testing

This section only deals with the Reburn application and ensuring that the software to be loaded was loaded. The results may be seen below.

5.1.3.2.1 Integrity

There is no test for this section regarding Reburn

5.1.3.2.2 Authentication

1. Reburn shall create the LIVE image defined by the user.
    a. Test case: Several variations of configurations will be made and tested to see if FENIX was made corresponding to the image.

Several different variations of FENIX were created with different kernels and only a single application that was different. Each kernel download represented a different bootsplash, which is an easy way to determine if the kernel changed. In addition to FENIX properly creating the bootsplash kernel, the varying applications were different as per requested in the configuration.

Finally, the customizations were tested based upon their specializations. These specializations include: copy to memory, ejectcd, login to GUI, don't login to GUI, write protect, etc. After each customization the configuration was tested to determine if the modifications were shown. In each customization the results were inline with the configuration.

5.1.3.2.3 Non-interference

No tests apply to Reburn in this section.

5.1.3.2.4 Minimalization

1. Reburn shall only download content necessary to building the new OS.
    a. Although this doesn't affect any forensic aspect of the process, Reburn will utilize minimal bandwidth in its creation of FENIX.

Reburn was tested to determine what kind of information was downloaded during the creation process. Reburn downloads the kernel package requested by the user. This kernel package, contains an initialized ramdisk, a kernel image, and a kernel module. Authors can choose whether they want to include any of the 3 items. Unfortunately, at this time, Reburn does not determine which files are necessary to update on the machine and therefore, some software may be downloaded and overwritten that was unnecessary to update. This can be seen in the kernel download section of Reburn. Most kernels of the 2.6.20 linux kernel version, have the same kernel image, however during each kernel selection, this image is downloaded.

Similarly, the application module section will download chosen applications even if they exist on the local machine already. This is the only area of concern where Reburn will download the information if it is already present and is therefore slightly redundant in its process.

However, Reburn does not download the larger portion of the FENIX project. This FENIX skeleton structure must already be downloaded and present on the computer. This saves the server precious resources and bandwidth.

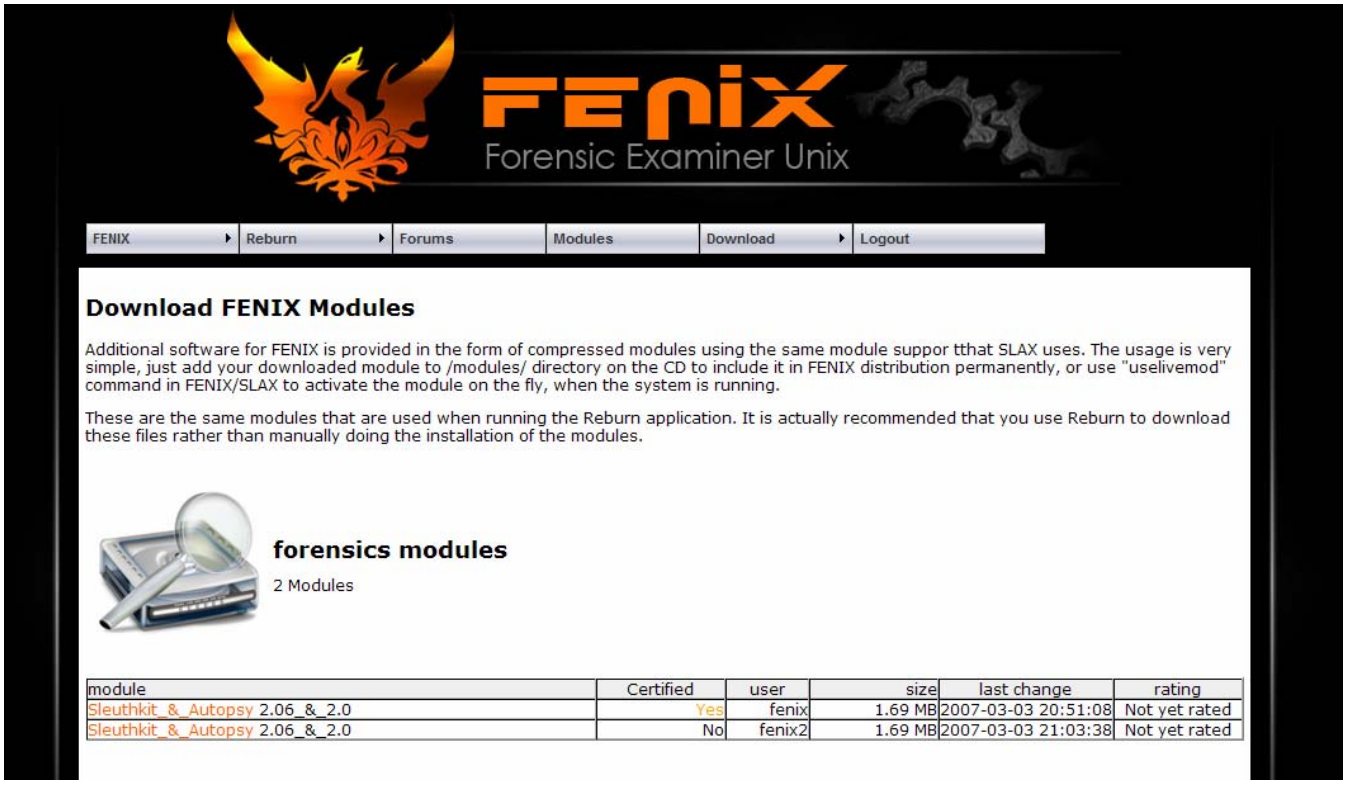
5.1.3.2.5 Reproducibility

1. Reburn shall be able to reproduce results following the same process.
    a. Reburn, given the same inputs shall reproduce the same image. The image will be checked via an md5 hash.

5.1.3.2.6 Flexible

1. Reburn shall be able to be easy to modify or change when a user wants to customize their disk. The applications and kernels available for FENIX shall be easily updated or modified with the newest in tools.
    a. FENIX will be configured and tested to see if the proper applications were added upon customization.

Reburn was started and all steps were followed that were mandatory including: kernel selection, module selection, and FENIX creation. This ultimately created a FENIX disk image with applications that were downloaded onto the disk. All tests showed that this process worked.



5.1.3.3 Website Testing

The only aspects of the website that may be tested deal with website security and its ability to restrict non-privileged users. The test results are below.

5.1.3.3.1 Integrity

1. The website shall not allow anyone to change an upload but the original author
    a. Test case: attempt to overwrite a previously uploaded module uploaded by another user

A module was uploaded using the simple web interface. The 1st module was a module uploaded by the “fenix” user. The second module was uploaded by ”fenix2”. The 1st module was not overwritten by the second, but given a first look at the module listing it seemed like

there was little difference between the two. As such, precautions were taken to allow a user the knowledge of safely downloading one module versus another. The certification of a module was enabled to be shown on the first module reference.

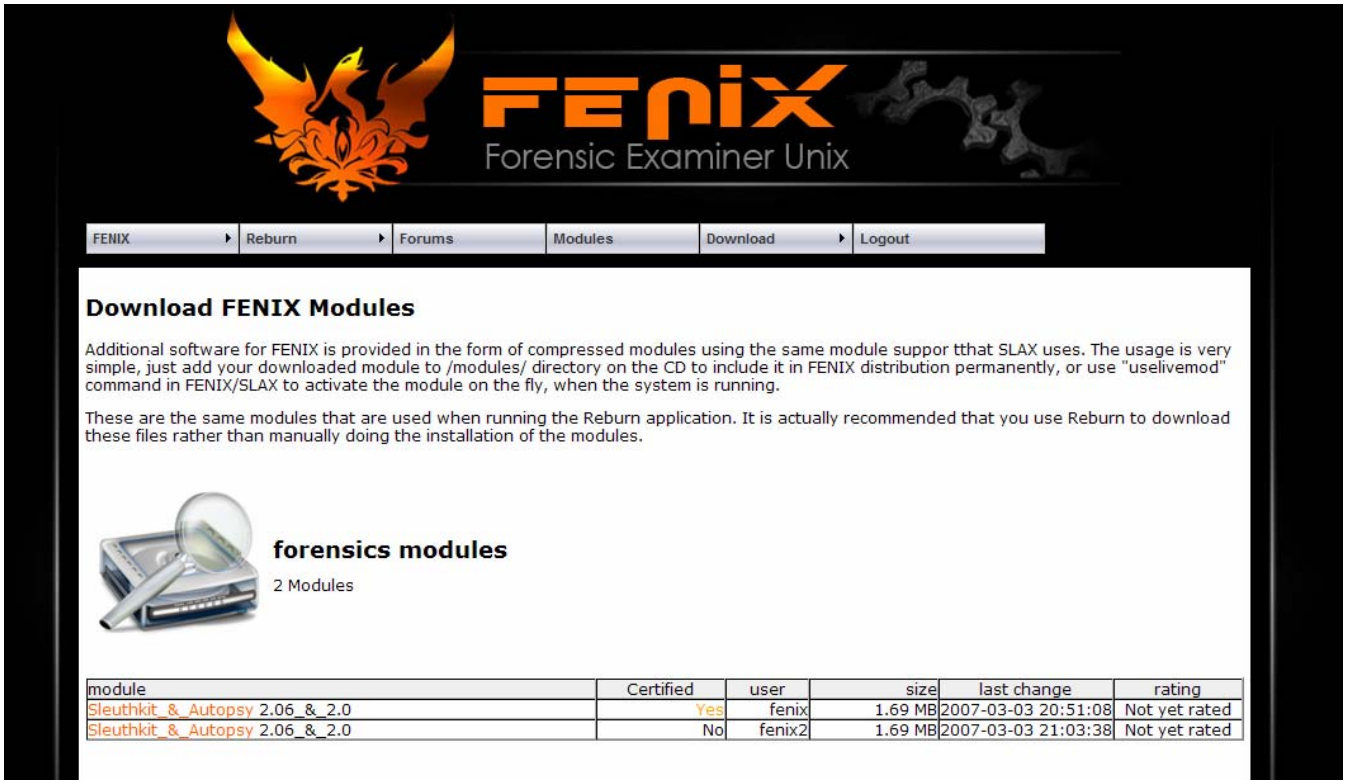
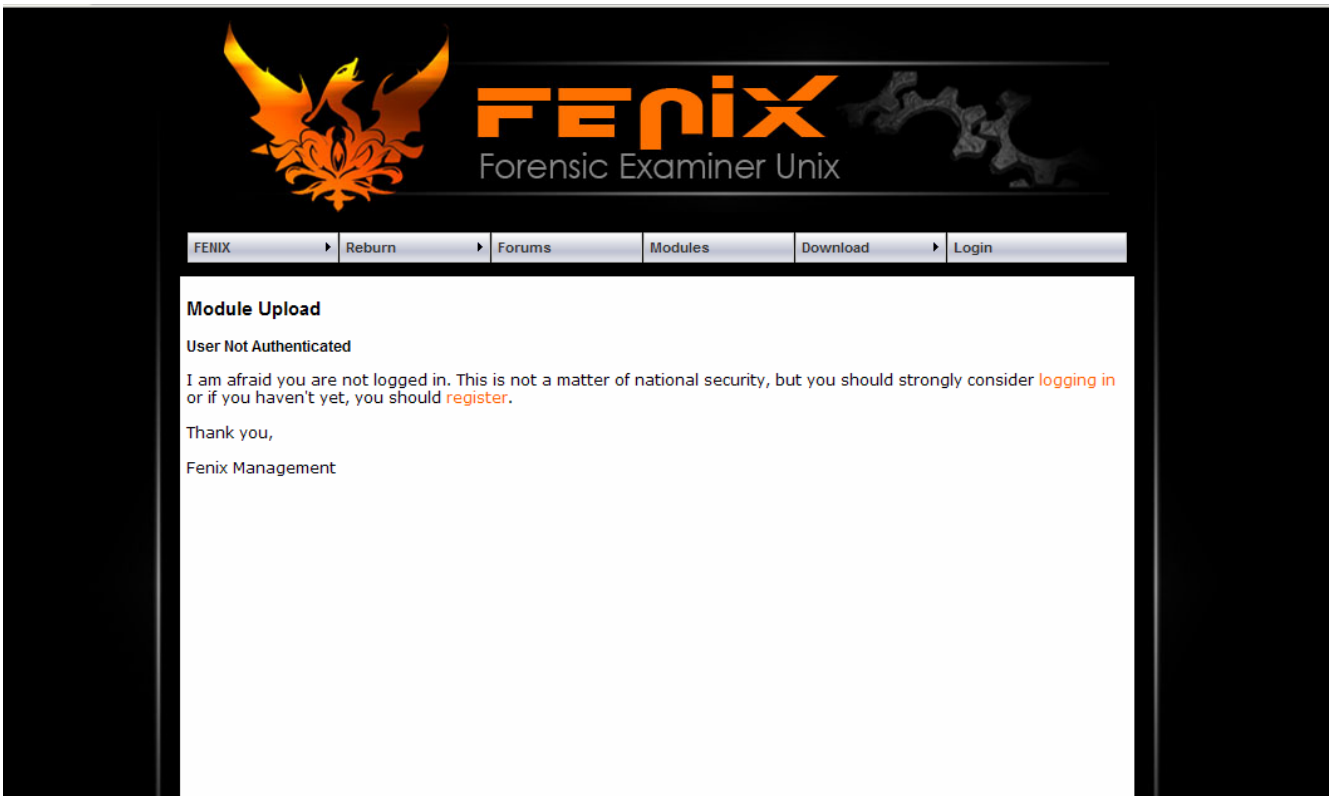


Figure 30: A duplicate file was uploaded to test if an overwrite occurs

5.1.3.3.2 Authentication

1. Only privileged webpage users will be able to upload information to the database.
    a. Test case: A series of write procedures will attempted through the webpage without being logged in.



In the above picture, the Fenix website rejects a request to access a webpage based upon user session information. Since the user is not logged in, he cannot access the page to upload data. Similarly, if a user attempts to post information to the page even though the upload information is accessed, the user will be unable to do so as the post of information also requests session information to determine the users abilities.

2. Only authentic information will be able to be added to the database.
    a. Test case: A user will attempt to upload information. The information shall not be directly uploaded and passed through a moderator.

Default uploads are set to a boolean value of false for certification. once a moderator has reviewed the content of the upload, it will be judged as having passed certification or removed.

The method for automatic acceptance of modules was chosen to enable the fasted possible spread of information and new software. If the Fenix Project is widely used, on busy days even an army of moderators won't be able to keep up. As such, it is deemed important to allow users to download uncertified modules, but at the same time require a limiting agent by moderating modules and certifying those that pass as both being the same software as is labeled and safe.

5.1.3.3.3 Non-interference

There is not test available for this section regarding the website.

5.1.3.3.4 Minimalization

There is no test available for this section regarding the website.

5.1.3.3.5 Reproducibility

There is no test available for this section regarding the website.

5.1.3.3.6 Flexible

There is no test available for this section regarding the website.

5.1.4 Project Testing Information

The project was tested in several aspects of and relating to its webpage, its OS, and its application. Each was tested against forensic aspects and its ability to handle those testing.

5.1.4.1 Version

The test were run against version 1.0 of FENIX, 1.0 of Reburn, and its first iteration of the webpage.

5.1.4.2 Kernel

The kernel used within FENIX was 2.6.20 with a slightly modified and reconfigured kernel. The new modifications involve support for squashFS (LZMA Compression) and unionFS (AUFS). In addition to the filesystem modifications, this kernel allows for bootsplash. Other than that the kernel was not modified in any other way.

5.1.4.4 Image Availability

The image that was tested can be viewed at http://fenix.iseage.org/images/fenix-forensic-1.0.iso.

5.1.4.5 Subjective Component

In cases where a users perspective come into question a new set of testing needs to be carried out. For the subjective testing, FENIX will rely on a great deal of polling. The polling received from the polls is shown below.

5.1.4.5.1 Value

80% of the users said they would use Reburn or FENIX for some occasion.

100% of the users said they might know someone who would use FENIX or Reburn.

60% of the users have used a forensic utility before.

80% saw a need for a forensic utility such as FENIX.

Overall Score (4.33/5)

5.1.4.5.2 Easy to Use

100% of the users were able to create a LIVE OS using Reburn.

60% of the users didn't run into some problem while creating FENIX.

80% of the users thought Reburn ran the way it was expected to.

80% of the users thought Reburn wasn't overly complicated

80% of the users thought FENIX was easy to use

Overall Score (4/5)

5.1.4.5.3 Live CD Competition

100% of the users said they would rather use FENIX than HELIX

100% of the users said they would rather use FENIX than Knoppix

80% of the users said they would rather use FENIX than a commercial LIVE OS

100% said they would rather use FENIX than the Penguin Sleuth Kit

100% said they found the advanced features of FENIX to be useful

Overall Score (4.8/5)

5.1.4.5.4 Effective

80% said that Reburn wasn't overly simplified

100% were able to create, customize, and build a LIVE OS

80% were able to create an OS that allowed them to perform the actions they wished.

80% were able to add their own custom modules

100% were able to connect to the database to download a kernel or module

Overall Score (4.4/5)

5.1.4.5.5 Security

80% saw FENIX as being non-destructive

60% were able to establish that the filesystem was in fact not modified

80% saw Reburn as not having security issues

60% saw FENIX as not having security issues

60% saw the FENIX website as not being a security concern

Overall Score (3.4/5)

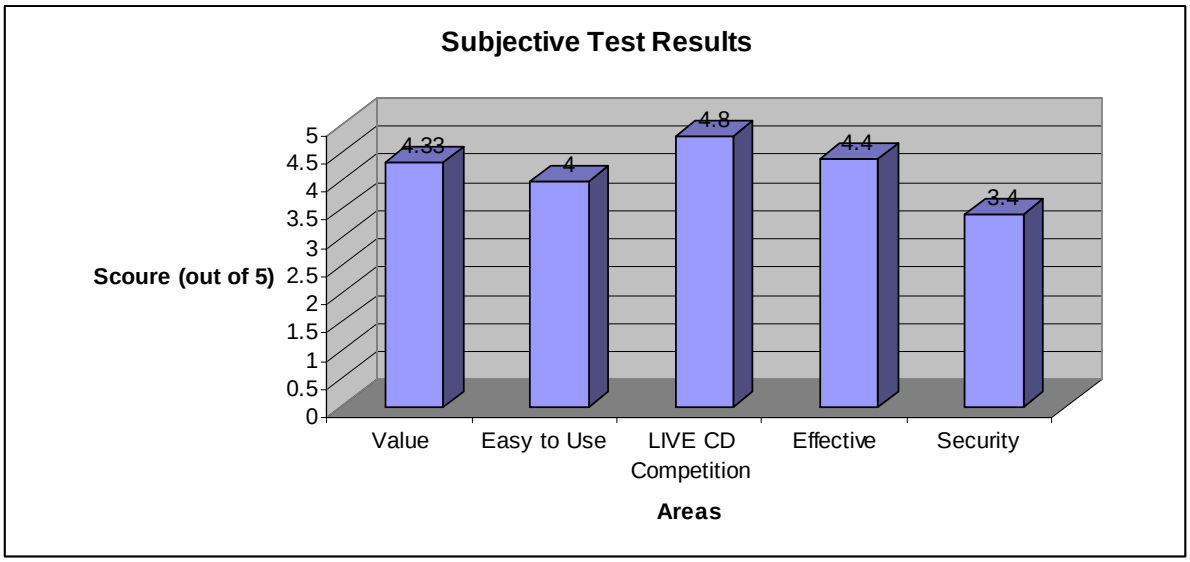


Figure 31: Subjective Score Chart, showing user opinions

5.2 Criteria Review

FENIX, Reburn, and the FENIX website were all tested against the current standards in forensics and testing practices defined by NIST. Overall FENIX upheld against all tests. Reburn itself held up quite well against its tests in particular the subjective testing. The website itself still has some security concerns as it has not been fully tested and there are

some theoretical areas of weakness that could occur. More work needs to be done on the project in general, these areas include:

- FENIX's ease of use
- Reburn's high system resource requirements
- Website Security
- Website integrity of modules
- The amount of modules available for FENIX
- The lack of auto-mount in read-only for FENIX

CHAPTER 6. SUMMARY AND DISCUSSION

In this paper I have discussed the FENIX project and the majority of parts that go into it. I have taken a look at the background research that was used to build FENIX and the inner workings of how FENIX functions. It is the goal of this section to bring it all together and discuss FENIX's important contributions, the success of the project, and any future work that can be implemented.

6.1 Introduction

The FENIX Project was designed to suit the needs of the Iowa Department of Criminal Investigations and their need to have an easy to use and customizable forensic CD for first responder functions. Under this request FENIX has demonstrated its abilities to be both easy to use and customizable. In addition to fulfilling the subjective requirements, FENIX has shown that it is in compliance with present forensic standards. With these properties combined it has made FENIX an asset to any organization wishing to use FENIX as a forensic tool.

6.2 Important Contributions

Presently, FENIX is the only known user-friendly, highly customizable, modular, forensic LIVE CD available. The FENIX Project has contributed many ideas that have both been considered and not considered for forensic work. This is both a success and a failure in part by FENIX and shall be discussed in greater detail.

6.1.1 The FENIX Project Successes

The FENIX Project has successfully implemented the Linux-Live tools into a Slackware installation. Although this process has been previously done with both GoblinX

and SLAX, this is the first time it has been done for a forensically sound system that utilizes the live-tools in such a way as not write post data to any drives connected to the machine.

In addition to incorporating proper forensic standards, FENIX has used the modularity of Linux-Live to allow a heightened level of customization. By removing the difficulties in task and time involved, Reburn has made updating a LIVE CD much easier. Now, options for bootup are but a click away. The kernel selection and kernel update of an OS is a click away. New tools and applications are a click away. This simplifies everything and is able to speed up the process of LIVE OS development.

6.1.2 The FENIX Project Failures

Unfortunately, FENIX doesn't have a large group of people working on its implementation. As such certain security concerns arise; most notably, the authentication of modules. How does a user know that the modules downloaded are what they say they are and not a Trojan horse?

Presently authentication is done through manual download and verification through a human. The upload is restricted and documented, but a user may create another email account and re-authenticate himself to another account in order to upload more troublesome Trojan horses. Since the man-power does not exist to create all modules by trusted parties, I must still allow the 3rd party uploading of certain applications.

To combat this forensic module will be built by trusted members and certified by FENIX. The other modules will be marked as being developed by a 3rd party. This should help a user to acknowledge that a module may be potentially harmful.

6.2 Conclusion

The FENIX project has shown that it is worthy of forensic work and analysis. It has been researched to uphold the requirements of forensic tools and currently available

operating systems. FENIX has been tested both objectively and subjectively against these requirements and has come out passing all of the tests.

In the developer's viewpoint, FENIX has accomplished a great deal of success but remains lacking in a few areas, described earlier as success and failures. Although FENIX exhibited a high success to failure ratio, there is still work to be done in order to achieve perfection. Although FENIX is an asset to anyone who wants to have a first response kit, customized to their needs and readily available, questions about the security of modules and website authentication continually need to be queried to create a heightened level of security.

6.3 Future Work

New areas of research are opening up every day within the world of digital forensics. Unfortunately, the areas of research are not taken up as fast as they should be due to startup costs and training. It is my hope that FENIX will help to enlighten forensic professionals into new schools of thought that will help them pass these challenges as they are brought to them.

These new challenges include:

- Module creation of new libraries / tools
- Enhanced options available for OSs
- New hardware support
- A greater level of security within the website
- A deeper understanding of digital forensics and secure acquisitions
- An advanced routine within Reburn which allows more advanced customizations

APPENDIX A. FENIX TEST DATA

Three different tests that encompass the forensic facets of FENIX are shown below in their entirety. Images were taken at every step so as to prove the forensic integrity of FENIX.

/dev/sdb

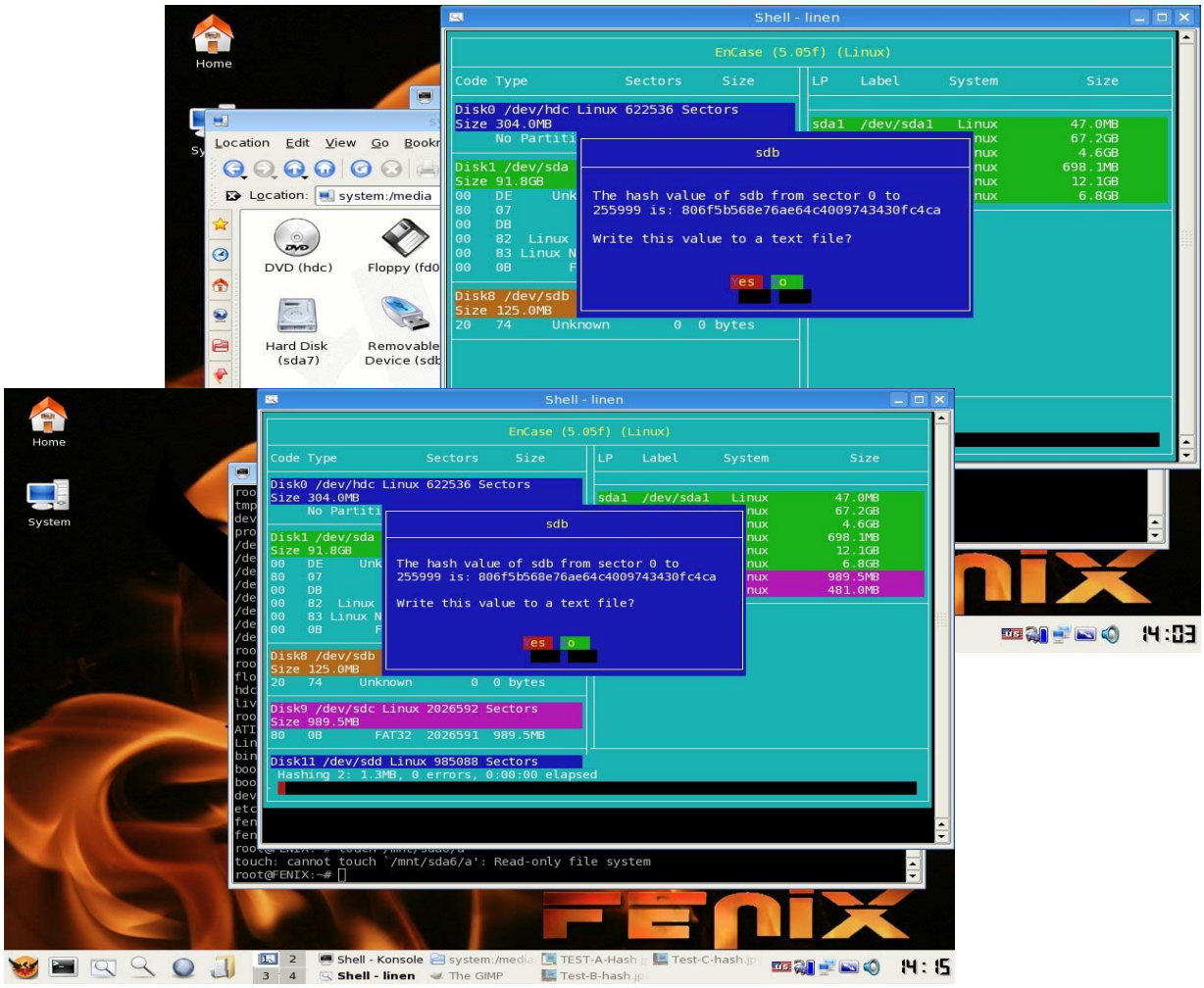


Figure 32: /dev/sdb before and after mounting (notice time difference)

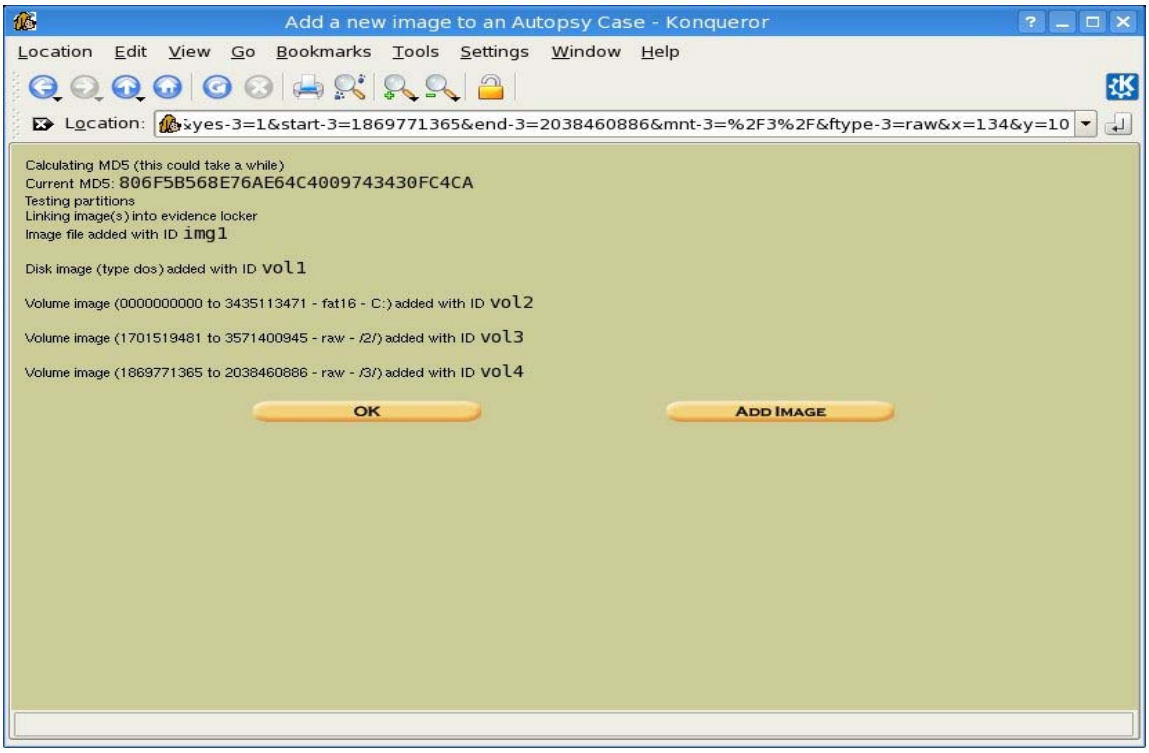


Figure 33: Hash value acquired by Autopsy on FENIX

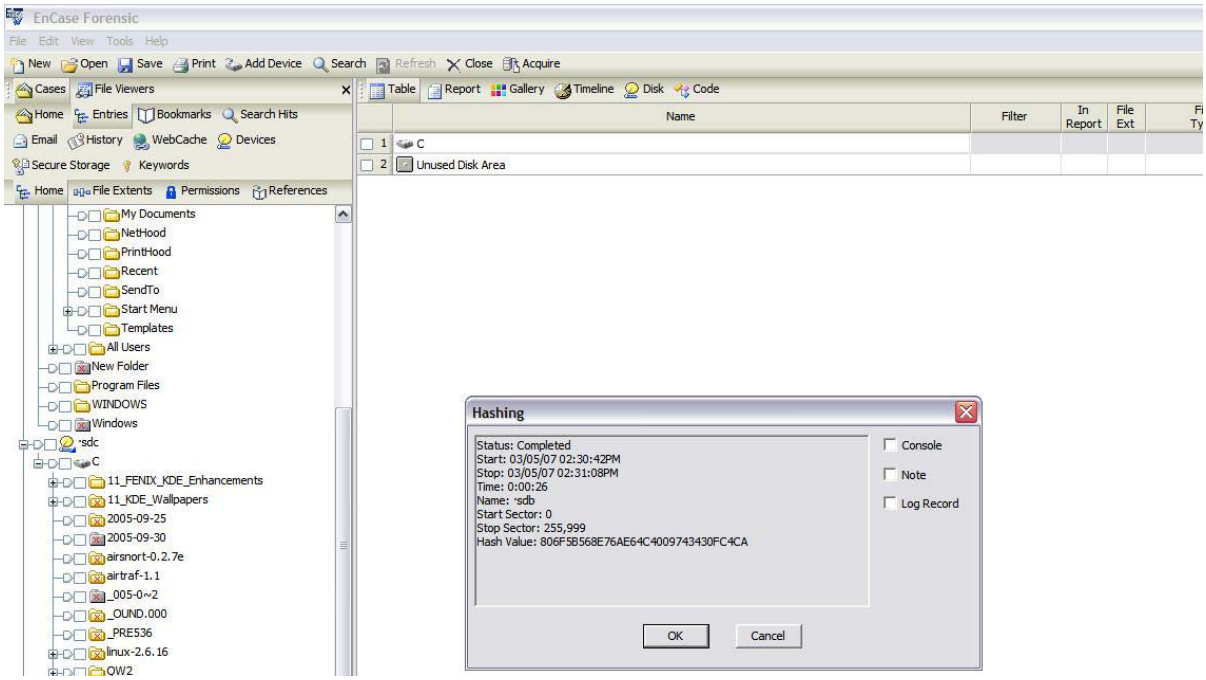


Figure 34: Hash value acquired by Encase through Linen on FENIX

DEL	Type dir / in	NAME	WRITTEN	ACCESSED	CREATED	SIZE	UID	GID	META
✓	r / r	4501_9d24f95ed4_m.jpg	2006.03.07 17:57:20 (GMT)	2006.03.07 00:00:00 (GMT)	2006.03.07 17:57:18 (GMT)	0	0	0	25
✓	r / r	4501_9d24f95ed4_m.text	2006.03.07 17:57:20 (GMT)	2006.03.07 00:00:00 (GMT)	2006.03.07 17:57:18 (GMT)	7417	0	0	28
✓	r / r	_kitty_porn_big_embrace.jpg	2006.03.07 17:57:34 (GMT)	2006.03.07 00:00:00 (GMT)	2006.03.07 17:57:32 (GMT)	0	0	0	32
	r / r	_kitty_porn_big_embrace.text	2006.03.07 17:57:34 (GMT)	2006.03.07 00:00:00 (GMT)	2006.03.07 17:57:32 (GMT)	13571	0	0	36
✓	r / r	_OVE.jpg	2006.03.07 17:56:58 (GMT)	2006.03.07 00:00:00 (GMT)	2006.03.07 17:56:56 (GMT)	0	0	0	21
✓	r / r	_OVE.jpg	2006.03.07 17:56:58 (GMT)	2006.03.07 00:00:00 (GMT)	2006.03.07 17:56:56 (GMT)	53645	0	0	22
✓	r / r	_WRD0472.tmp	2006.02.24 08:17:32 (GMT)	2006.02.24 00:00:00 (GMT)	2006.02.24 08:17:30 (GMT)	22016	0	0	6
✓	r / r	_WRD3838.tmp	2006.02.28 09:02:20 (GMT)	2006.02.28 00:00:00 (GMT)	2006.02.28 09:02:14 (GMT)	87552	0	0	14
	r / r	amoeba.jpg	2006.02.20 23:04:26 (GMT)	2007.02.26 00:00:00 (GMT)	2006.02.20 23:11:36 (GMT)	135294	0	0	75
	r / r	appender.jpg	2006.02.20 23:07:20 (GMT)	2007.02.26 00:00:00 (GMT)	2006.02.20 23:11:36 (GMT)	108254	0	0	76
	r / r	AUTOEXEC.BAT	2005.09.06 13:22:12 (GMT)	2006.03.07 00:00:00 (GMT)	2006.03.07 18:00:14 (GMT)	16	0	0	45
	r / r	boot.ini	2005.09.28 18:05:30 (GMT)	2006.03.07 00:00:00 (GMT)	2006.03.07 18:00:14 (GMT)	211	0	0	46
✓	r / r	briesleeps.jpg	2006.03.07 17:58:00 (GMT)	2006.03.07 00:00:00 (GMT)	2006.03.07 17:57:58 (GMT)	0	0	0	39
✓	r / r	briesleeps.text	2006.03.07 17:58:00 (GMT)	2006.03.07 00:00:00 (GMT)	2006.03.07 17:57:58 (GMT)	53148	0	0	42
	r / r	cavity-nonfraction.jpg	2006.02.20 23:06:04 (GMT)	2007.02.26 00:00:00 (GMT)	2006.02.20 23:11:36 (GMT)	93188	0	0	80

Figure 36: The dates and times of files within /dev/sdb in Autopsy

Table | Report | Gallery | Timeline | Disk | Code

	Name	Last Written	Last Accessed	File Created	File Ext	File Type	File Category
1	4501_9d24f95ed4_m.jpg	03/07/06 05:57:20PM	03/07/06	03/07/06 05:57:18PM	jpg	JPEG	Picture
2	4501_9d24f95ed4_m.text	03/07/06 05:57:20PM	03/07/06	03/07/06 05:57:18PM	text		
3	_kitty_porn_big_embrace.jpg	03/07/06 05:57:34PM	03/07/06	03/07/06 05:57:32PM	jpg	JPEG	Picture
4	_kitty_porn_big_embrace.text	03/07/06 05:57:34PM	03/07/06	03/07/06 05:57:32PM	text		
5	_OVE.JPG	03/07/06 05:56:58PM	03/07/06	03/07/06 05:56:56PM	JPG	JPEG	Picture
6	_OVE.JPG	03/07/06 05:56:58PM	03/07/06	03/07/06 05:56:56PM	JPG	JPEG	Picture
7	_WRD0472.TMP	02/24/06 08:17:32AM	02/24/06	02/24/06 08:17:30AM	TMP	Windows Temporary	Windows
8	_WRD3838.TMP	02/28/06 09:02:20AM	02/28/06	02/28/06 09:02:14AM	TMP	Windows Temporary	Windows
9	AMOEBA.JPG	02/20/06 11:04:26PM	02/26/07	02/20/06 11:11:36PM	JPG	JPEG	Picture
10	APPENDER.JPG	02/20/06 11:07:20PM	02/26/07	02/20/06 11:11:36PM	JPG	JPEG	Picture
11	AUTOEXEC.BAT	09/06/05 01:22:12PM	03/07/06	03/07/06 06:00:14PM	BAT	Batch	Code\Executable
12	BOOT.INI	09/28/05 06:05:30PM	03/07/06	03/07/06 06:00:14PM	INI	Initialization	Windows
13	briesleeps.jpg	03/07/06 05:58:00PM	03/07/06	03/07/06 05:57:58PM	jpg	JPEG	Picture
14	briesleeps.text	03/07/06 05:58:00PM	03/07/06	03/07/06 05:57:58PM	text		
15	cavity-nonfraction.jpg	02/20/06 11:06:04PM	02/26/07	02/20/06 11:11:36PM	jpg	JPEG	Picture
16	CAVITY.JPG	02/20/06 11:05:08PM	02/26/07	02/20/06 11:11:36PM	JPG	JPEG	Picture
17	compressor.jpg	02/20/06 11:08:50PM	02/26/07	02/20/06 11:11:36PM	jpg	JPEG	Picture
18	CprE 537XmidtermStudyGuide.doc	02/28/06 09:02:16AM	02/28/06	02/28/06 09:02:14AM	doc	Word Document	Document
19	dependencies.jpg	02/20/06 08:56:44PM	02/20/06	02/20/06 08:56:42PM	jpg	JPEG	Picture
20	dependencies.jpg	02/20/06 08:56:48PM	02/26/07	02/20/06 08:56:42PM	jpg	JPEG	Picture
21	dependencies.jpg	02/20/06 08:56:44PM	02/20/06	02/20/06 08:56:42PM	jpg	JPEG	Picture
22	Documents and Settings	03/07/06 06:01:04PM	03/07/06	03/07/06 06:01:02PM			
23	DougPapers.doc	02/24/06 08:17:32AM	02/24/06	02/24/06 08:17:30AM	doc	Word Document	Document
24	DougPapers.doc	02/24/06 08:17:32AM	02/24/06	02/24/06 08:17:30AM	doc	Word Document	Document
25	IO.SYS	02/24/05 04:34:10PM	03/07/06	03/07/06 06:00:14PM	SYS	Device Driver	Code\Executable
26	kitty_pr0n.jpg	03/07/06 05:56:42PM	03/07/06	03/07/06 05:56:40PM	jpg	JPEG	Picture

Figure 35: The dates and times of files within /dev/sdb on Encase

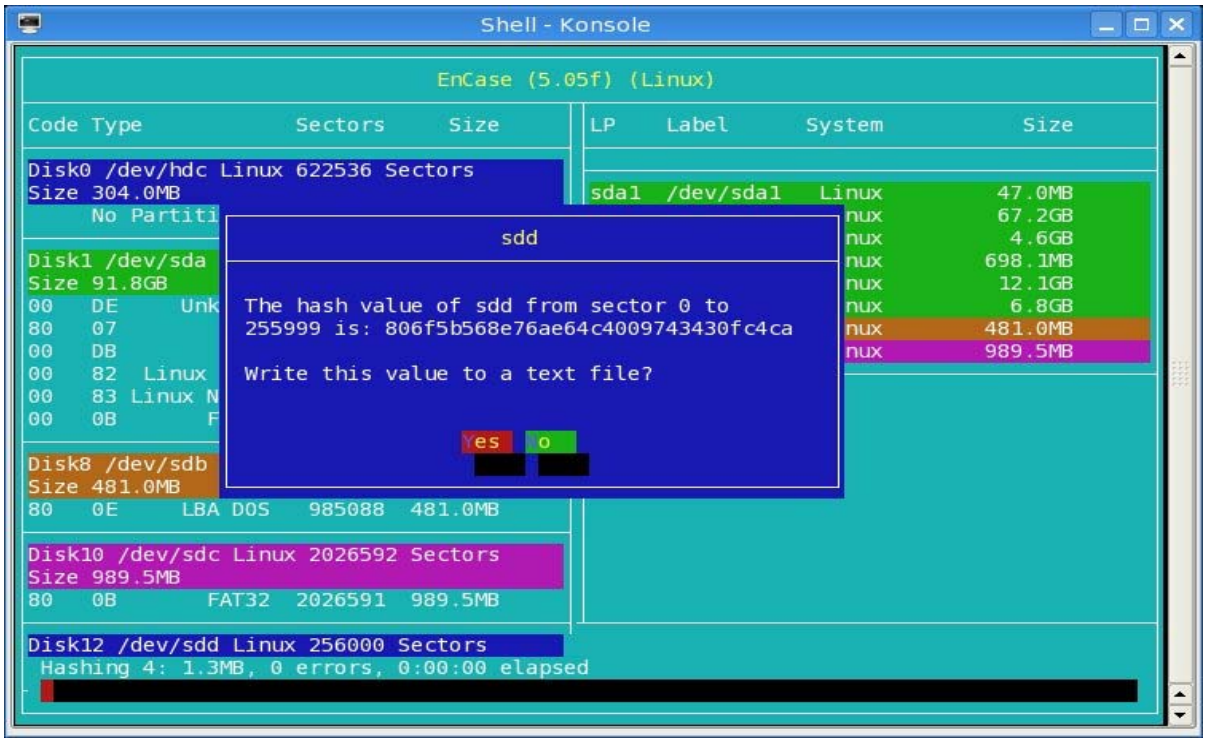


Figure 37: After reboot, names of devices are changed, /dev/sdb is moved to /dev/sdd but hash value remains unchanged

/dev/sdc

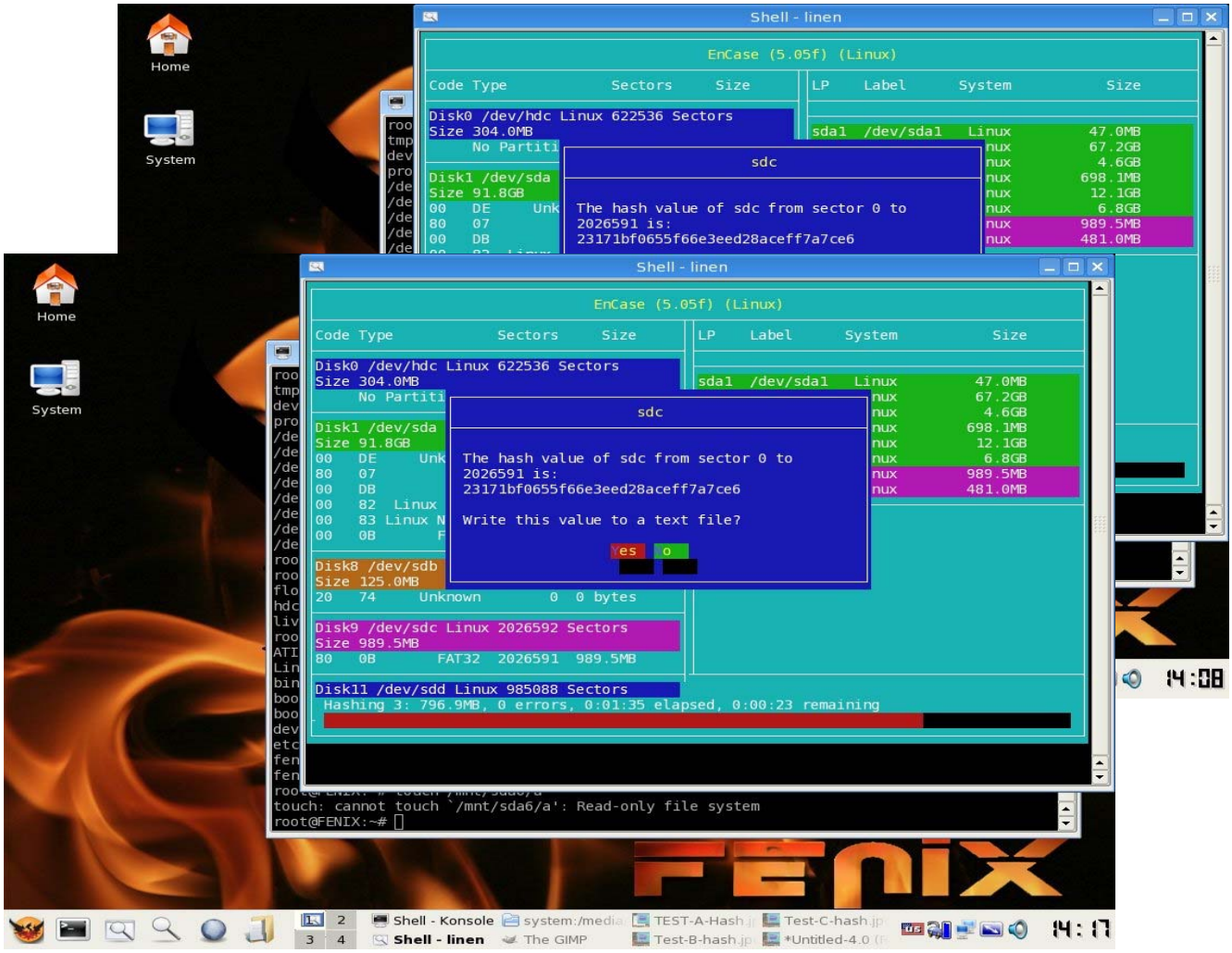


Figure 38: Linen tests hash values both pre-mount and post-mount to determine of mounting changes hash

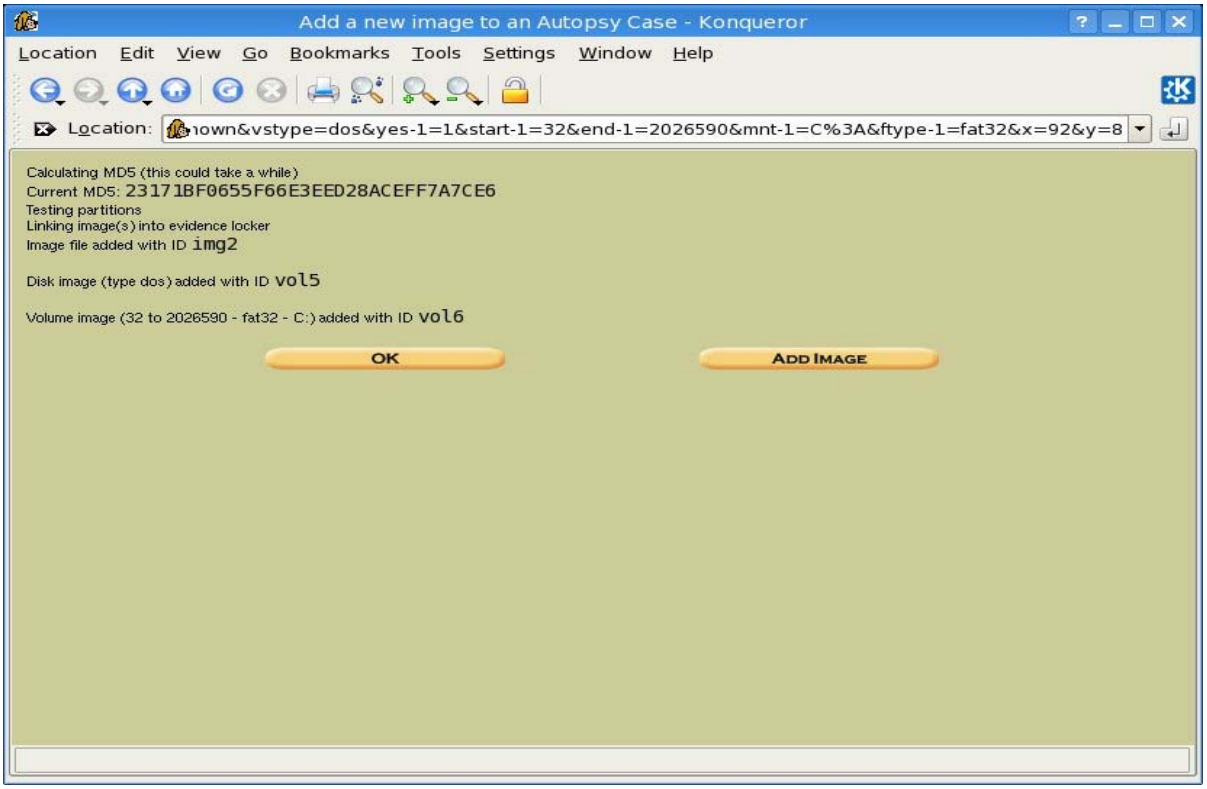


Figure 39: Hash value after acquisition by Autopsy

	Name	Last Written	Last Accessed	File Created	File Ext	File Type	File Category	Filter
1	_MLINU~1.PAR	10/26/06 01:17:42AM	10/26/06	10/26/06 01:17:42AM	PAR			
2	_NITRD~1.PAR	10/26/06 01:30:22AM	10/26/06	10/26/06 01:30:22AM	PAR			
3	bootsplash-1024x768.jpg	10/25/06 04:40:46PM	01/03/07	10/25/06 04:40:40PM	jpg	JPEG	Picture	
4	bzImage	10/26/06 12:32:30AM	10/26/06	10/26/06 01:17:42AM				
5	ini-uncomp.gz	10/26/06 01:32:00AM	10/26/06	10/26/06 01:32:20AM	gz	GZIP Compressed	Archive	
6	initrd-Given.gz	10/26/06 01:29:22AM	10/26/06	10/26/06 01:30:22AM	gz	GZIP Compressed	Archive	
7	initrd-My.gz	10/26/06 01:28:32AM	10/26/06	10/26/06 01:30:22AM	gz	GZIP Compressed	Archive	
8	initrd.gz	10/25/06 08:34:46PM	10/25/06	10/25/06 08:37:52PM	gz	GZIP Compressed	Archive	
9	initrdX.gz	10/25/06 08:38:22PM	10/25/06	10/25/06 08:38:52PM	gz	GZIP Compressed	Archive	
10	linux-live-5.5.0	10/26/06 01:25:50AM	10/26/06	10/26/06 01:25:50AM	0			
11	Primary FAT							
12	Secondary FAT							
13	silent-1024x768.jpg	10/25/06 04:36:16PM	01/03/07	10/25/06 04:36:10PM	jpg	JPEG	Picture	
14	squashfs	10/26/06 01:24:24AM	10/26/06	10/26/06 01:24:24AM				
15	Unallocated Clusters							
16	unionfs	10/26/06 01:24:34AM	10/26/06	10/26/06 01:24:34AM				
17	vmlinux.bin	10/26/06 12:32:30AM	10/26/06	10/26/06 01:17:42AM	bin	Raw Binary Format	Windows\DOS	
18	Volume Boot							
19	Volume Slack							

Figure 41: Encase dates and times of acquisition

DEL	Type dir / in	NAME	WRITTEN	ACCESSED	CREATED	SIZE
✓	r / r	_MLINU~1.PAR	2006.10.26 01:17:42 (GMT)	2006.10.26 00:00:00 (GMT)	2006.10.26 01:17:42 (GMT)	0
✓	r / r	_NITRD~1.PAR	2006.10.26 01:30:22 (GMT)	2006.10.26 00:00:00 (GMT)	2006.10.26 01:30:22 (GMT)	0
	r / r	bootsplash-1024x768.jpg	2006.10.25 16:40:46 (GMT)	2007.01.03 00:00:00 (GMT)	2006.10.25 16:40:40 (GMT)	191126
	r / r	bzImage	2006.10.26 00:32:30 (GMT)	2006.10.26 00:00:00 (GMT)	2006.10.26 01:17:42 (GMT)	2947181
	r / r	ini-uncomp.gz	2006.10.26 01:32:00 (GMT)	2006.10.26 00:00:00 (GMT)	2006.10.26 01:32:20 (GMT)	1729748
	r / r	initrd-Given.gz	2006.10.26 01:29:22 (GMT)	2006.10.26 00:00:00 (GMT)	2006.10.26 01:30:22 (GMT)	1455416
	r / r	initrd-My.gz	2006.10.26 01:28:32 (GMT)	2006.10.26 00:00:00 (GMT)	2006.10.26 01:30:22 (GMT)	1789662
	r / r	initrd.gz	2006.10.25 20:34:46 (GMT)	2006.10.25 00:00:00 (GMT)	2006.10.25 20:37:52 (GMT)	1349769
	r / r	initrdX.gz	2006.10.25 20:38:22 (GMT)	2006.10.25 00:00:00 (GMT)	2006.10.25 20:38:52 (GMT)	1790501
	d / d	linux-live-5.5.0/	2006.10.26 01:25:50 (GMT)	2006.10.26 00:00:00 (GMT)	2006.10.26 01:25:50 (GMT)	8192
	r / r	silent-1024x768.jpg	2006.10.25 16:36:16 (GMT)	2007.01.03 00:00:00 (GMT)	2006.10.25 16:36:10 (GMT)	249460
	d / d	squashfs/	2006.10.26 01:24:24 (GMT)	2006.10.26 00:00:00 (GMT)	2006.10.26 01:24:24 (GMT)	8192
	d / d	unionfs/	2006.10.26 01:24:34 (GMT)	2006.10.26 00:00:00 (GMT)	2006.10.26 01:24:34 (GMT)	8192

Figure 40: Autopsy dates and times of acquisition

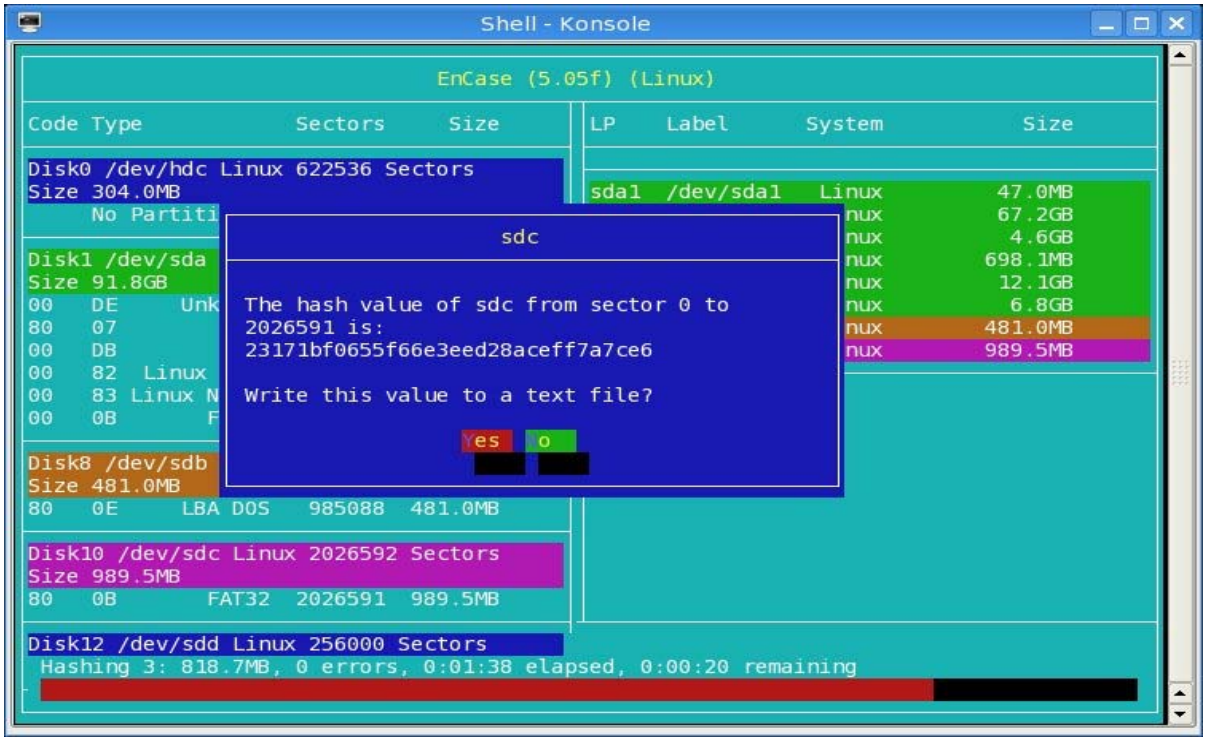


Figure 42: Hash value of device doesn't change after reboot

/dev/sdd

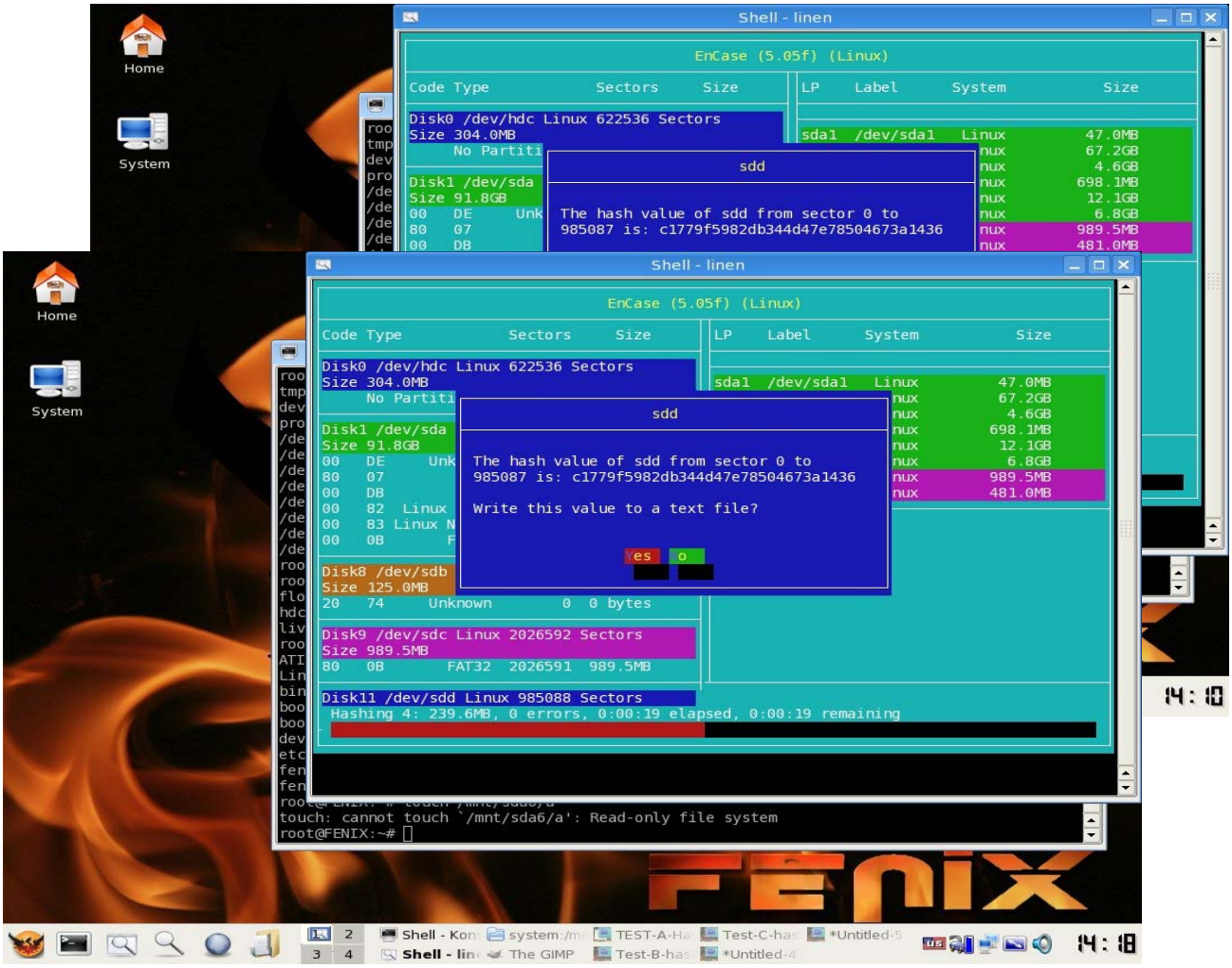


Figure 43: Hashes before and after mounting the device, notice the times

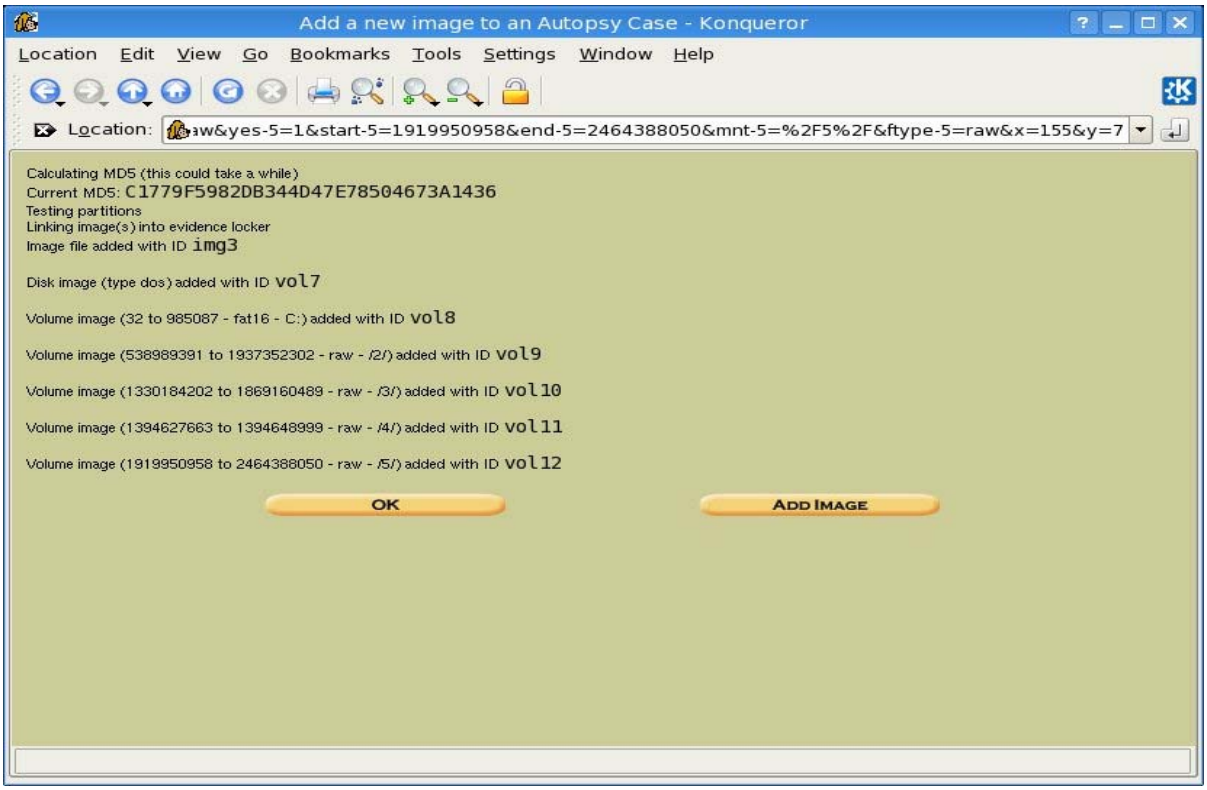


Figure 45: Hash value of data in Autopsy

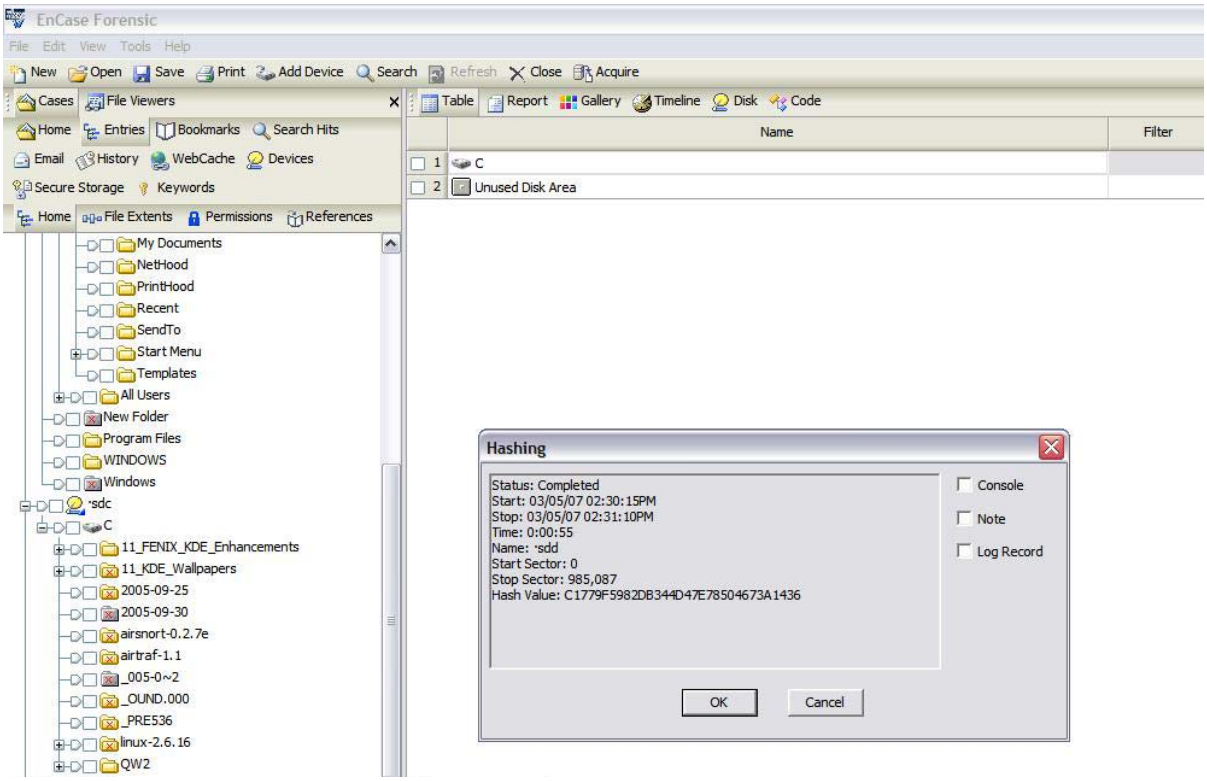


Figure 44: Hash value of data in Encase

DEL	Type dir / in	NAME	WRITTEN	ACCESSED	CREATED	SIZE	UID	GID	META
✓	r / r	10_FENIX_KDE_Enhancements.lzm	2007.02.23 13:30:34 (GMT)	2007.02.23 00:00:00 (GMT)	2007.02.23 13:30:34 (GMT)	2830336	0	0	50
✓	r / r	10_KDE_Enhancements.lzm	2007.02.23 13:30:14 (GMT)	2007.02.23 00:00:00 (GMT)	2007.02.23 13:30:14 (GMT)	2830336	0	0	46
	r / r	11_FENIX_KDE_Enhancements.lzm	2007.02.23 14:03:40 (GMT)	2007.02.23 00:00:00 (GMT)	2007.02.23 14:03:40 (GMT)	2859008	0	0	78
	d / d	11_FENIX_KDE_Enhancements/	2007.02.23 14:10:04 (GMT)	2007.02.23 00:00:00 (GMT)	2007.02.23 14:10:04 (GMT)	4096	0	0	81
✓	d / d	11_KDE_Wallpapers/	2007.02.23 14:11:16 (GMT)	2007.02.23 00:00:00 (GMT)	2007.02.23 14:11:16 (GMT)	4096	0	0	43
✓	d / d	2005-09-25/	2005.10.02 23:35:46 (GMT)	2005.10.02 00:00:00 (GMT)	2005.10.02 23:35:44 (GMT)	4096	0	0	86
✓	d / d	2005-09-30/	2005.10.02 23:35:34 (GMT)	2005.10.02 00:00:00 (GMT)	2005.10.02 23:35:32 (GMT)	897024	0	0	84
✓	d / d	_005-0~2/	2005.10.02 23:35:26 (GMT)	2005.10.02 00:00:00 (GMT)	2005.10.02 23:35:24 (GMT)	2760704	0	0	82
✓	r / r	_OOTEX.LOG	2007.01.02 12:20:16 (GMT)	2007.01.02 00:00:00 (GMT)	2007.01.02 12:20:14 (GMT)	1540	0	0	12
✓	d / d	_OUND.000/	2007.01.02 12:20:14 (GMT)	2007.01.02 00:00:00 (GMT)	2007.01.02 12:20:14 (GMT)	4096	0	0	10
✓	d / d	_pre536/	2005.10.12 23:26:04 (GMT)	2005.10.12 00:00:00 (GMT)	2005.10.12 23:26:02 (GMT)	4096	0	0	87
✓	r / r	_tmp123.tmp	2005.10.18 16:55:44 (GMT)	2005.10.18 00:00:00 (GMT)	2005.10.18 16:52:32 (GMT)	8181945	0	0	96
✓	r / r	_tmp123.tmp	2005.10.18 23:04:10 (GMT)	2005.10.18 00:00:00 (GMT)	2005.10.18 23:03:56 (GMT)	5340108	0	0	109
✓	r / r	_UICKW~1.PAR	2006.12.14 15:58:10 (GMT)	2006.12.14 00:00:00 (GMT)	2006.12.14 15:58:04 (GMT)	4718592	0	0	15
✓	r / r	_w2.zip	2006.12.14 15:54:24 (GMT)	2006.12.14 00:00:00 (GMT)	2006.12.14 15:54:12 (GMT)	9818954	0	0	11

Figure 47: Dates and Times as recorded by Autopsy

	Name	Last Written	Last Accessed	File Created	File Ext	File Type	File Category	Filter
1	10_FENIX_KDE_Enhancements.lzm	02/23/07 01:30:34PM	02/23/07	02/23/07 01:30:34PM	lzm			
2	10_KDE_Enhancements.lzm	02/23/07 01:30:14PM	02/23/07	02/23/07 01:30:14PM	lzm			
3	11_FENIX_KDE_Enhancements	02/23/07 02:10:04PM	02/23/07	02/23/07 02:10:04PM				
4	11_FENIX_KDE_Enhancements.lzm	02/23/07 02:03:40PM	02/23/07	02/23/07 02:03:40PM	lzm			
5	11_KDE_Wallpapers	02/23/07 02:11:16PM	02/23/07	02/23/07 02:11:16PM				
6	2005-09-25	10/02/05 11:35:46PM	10/02/05	10/02/05 11:35:44PM				
7	2005-09-30	10/02/05 11:35:34PM	10/02/05	10/02/05 11:35:32PM				
8	_005-0~2	10/02/05 11:35:26PM	10/02/05	10/02/05 11:35:24PM				
9	_OOTEX.LOG	01/02/07 12:20:16PM	01/02/07	01/02/07 12:20:14PM	LOG	Log	Document	
10	_OUND.000	01/02/07 12:20:14PM	01/02/07	01/02/07 12:20:14PM	000	DoubleSpace Volume	Windows	
11	_PRE536	10/12/05 11:26:04PM	10/12/05	10/12/05 11:26:02PM				
12	_TMP123.TMP	10/18/05 11:04:10PM	10/18/05	10/18/05 11:03:56PM	TMP	Windows Temporary	Windows	
13	_TMP123.TMP	10/18/05 04:55:44PM	10/18/05	10/18/05 04:52:32PM	TMP	Windows Temporary	Windows	
14	_UICKW~1.PAR	12/14/06 03:58:10PM	12/14/06	12/14/06 03:58:04PM	PAR			
15	_W2.ZIP	12/14/06 03:54:24PM	12/14/06	12/14/06 03:54:12PM	ZIP	ZIP Compressed	Archive	
16	airsnort-0.2.7e	10/24/05 06:24:24PM	10/24/05	10/24/05 06:24:22PM	7e			
17	airtraf-1.1	10/24/05 06:24:50PM	10/24/05	10/24/05 06:24:48PM	1			
18	arianavenger-Cover4-BLACK.jpg	10/18/05 11:03:52PM	10/18/05	10/18/05 11:03:50PM	jpg	JPEG	Picture	
19	arianavenger-Cover4-BLACK.jpg	10/18/05 11:04:10PM	06/08/06	10/18/05 11:03:56PM	jpg	JPEG	Picture	
20	arianavenger-Cover4-BLACK.jpg	10/18/05 11:03:52PM	10/18/05	10/18/05 11:03:50PM	jpg	JPEG	Picture	
21	arianavenger-Cover4-BLACK.png	10/18/05 04:52:02PM	10/18/05	10/18/05 04:52:00PM	png	Portable Networks Gra	Picture	
22	arianavenger-Cover4-BLACK.png	10/18/05 04:55:44PM	06/08/06	10/18/05 04:52:32PM	png	Portable Networks Gra	Picture	
23	arianavenger-Cover4-BLACK.png	10/18/05 04:52:06PM	10/18/05	10/18/05 04:52:00PM	png	Portable Networks Gra	Picture	
24	dmesg[1311]	03/07/00 05:29:46AM	10/20/05	10/20/05 07:10:16PM				
25	dmesg[1311].g3	03/07/00 05:29:46AM	10/27/05	10/20/05 07:10:16PM	g3			
26	Fenix_icon128x128.png	02/23/07 12:58:06PM	02/23/07	02/23/07 12:58:06PM	png	Portable Networks Gra	Picture	

Figure 46: Dates and times as recorded by Encase

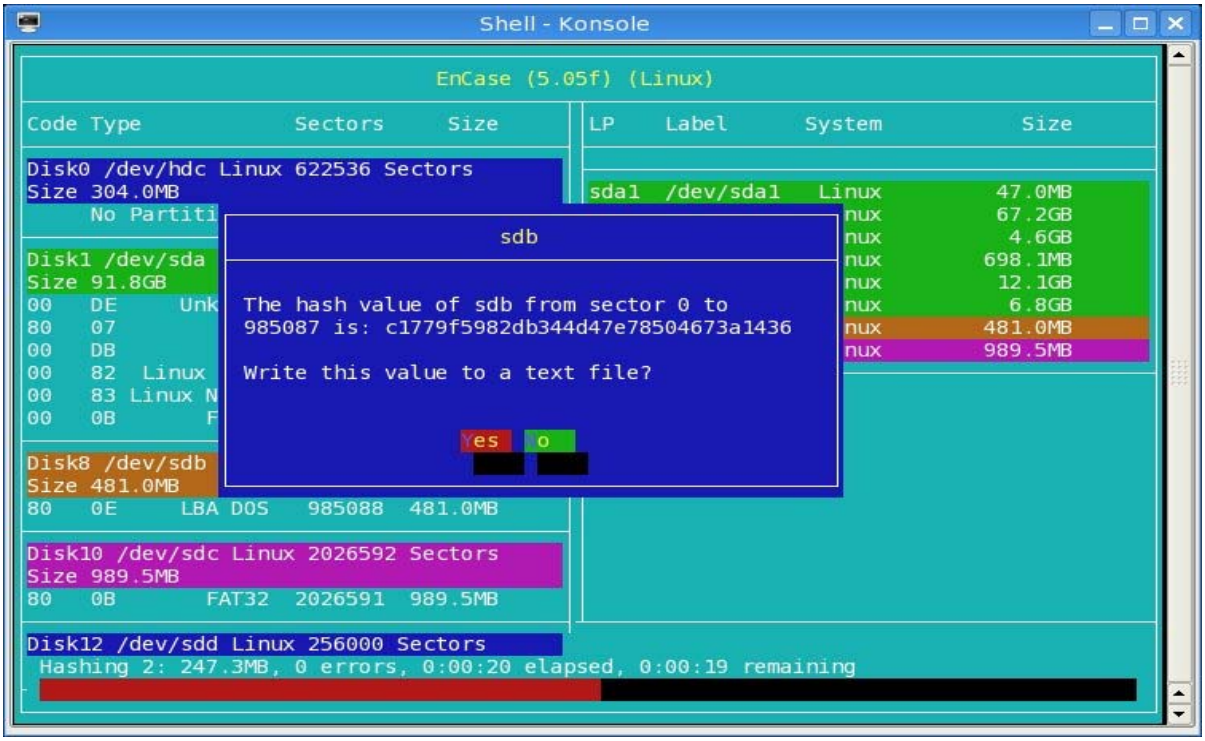


Figure 48: Hash of device after reboot. Device names changed, but hash values did not

Table 4: Devices and their hash values

Device	Size	Hash1 (pre-acquisition)
/dev/sdb	128mb	806f5b568e76ae64c4009743430fc4ca
/dev/sdd	512mb	23171bf0655f66e3eed28aceff7a7ce6
/dev/sdc	1024mb	C1779f5982db344d47e78504673a1436

APPENDIX B. XML SCHEMA

```
<?xml version="1.0"?><xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="config">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="kernel" type="xsd:string" minOccurs="1"
maxOccurs="1"/>
      <xsd:element name="modules">
   <xsd:complexType>
     <xsd:sequence>
      <xsd:element name="module" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
     </xsd:sequence>
   </xsd:complexType>
  </xsd:element>
  <xsd:element name="startOnBoot" type="xsd:string"
maxOccurs="unbounded"/>
  <xsd:element name="guiTaskbar" maxOccurs="4" minOccurs="1">
    <xsd:complexType>
         <xsd:simpleContent>
           <xsd:extension base="xsd:string">
            <xsd:attribute name="application" type="xsd:string"/>
           </xsd:extension>
         </xsd:simpleContent>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="special">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="nohotplug"/>
        <xsd:element name="nopcmcia"/>
        <xsd:element name="noagp"/>
        <xsd:element name="acpioff"/>
        <xsd:element name="setroot" type="xsd:string"/>
        <xsd:element name="askpass"/>
        <xsd:element name="disableguest"/>
        <xsd:element name="copy2mem">
          <xsd:complexType>
            <xsd:sequence>
               <xsd:element name="ejectcd">
                <xsd:complexType>
                 <xsd:simpleContent>
                   <xsd:extension base="xsd:string">
                      <xsd:attribute name="application">
                          <xsd:simpleType>
                             <xsd:restriction base="xsd:string">
                                <xsd:pattern value="true|false"/>

     </xsd:restriction>
                          </xsd:simpleType>

                       </xsd:attribute>
```

```
                    </xsd:extension>
                  </xsd:simpleContent>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="testMem"/>
        <xsd:element name="memSize" type="xsd:string"/>
        <xsd:element name="floppy"/>
        <xsd:element name="webconfig"/>
        <xsd:element name="saveChanges" type="xsd:string"/>

        <xsd:element name="memTestAvail"/>
        <xsd:element name="disableNet"/>
        <xsd:element name="kiosk"/>
        <xsd:element name="additional" type="xsd:string"/>
        <xsd:element name="memStick"/>
        <xsd:element name="startGUI"/>
        <xsd:element name="widescreen"/>
        <xsd:element name="wireless"/>
        <xsd:element name="writeProtect"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="createLoc" type="xsd:string" minOccurs="1"
maxOccurs="1"/>
   </xsd:sequence>
</xsd:complexType>
</xsd:element></xsd:schema>
```

APPENDIX C. XML CONFIGURATION FILE

```
<config xsi:noNamespaceSchemaLocation="http://fenix.iseage.org/fenix.config.xsd">
        <kernel>1</kernel>
        <modules>
                <module>12</module>
                <module>14</module>
                <module>4</module>
                <module>38</module>
        </modules>
        <startOnBoot>/etc/init.d/www</startOnBoot>
        <startOnBoot>/etc/init.d/ftpd</startOnBoot>
        <guiTaskbar application="/forensics/sleuthkit">
                /opt/kde3/share/doc/HTML/en/kdelibs-apidocs/favicon.ico
        </guiTaskbar>
        <guiTaskbar application="/bin/oocalc">/bin/ico/ooffice/calc.ico</guiTaskbar>
        <guiTaskbar application="/bin/etherape">/share/fenix-X/etherape.ico</guiTaskbar>
        <special>
                <nohotplug/>
                <nopcmcia/>
                <noagp/>
                <acpioff/>
                <setroot>password</setroot>
                <askpass/>
                <disableguest/>
                <copy2mem>
                        <ejectcd application="true"/>
                </copy2mem>
                <testMem/>
                <memSize>80%</memSize>
                <floppy/>
                <webconfig/>
                <saveChanges>true</saveChanges>
                <memTestAvail/>
                <disableNet/>
                <kiosk/>
                <additional>Additional Boot Options</additional>
                <memStick/>
                <startGUI/>
                <widescreen/>
                <wireless/>
                <writeProtect/>
        </special>
        <createLoc>/root/path/to/fenix.iso</createLoc>
</config>
```

APPENDIX D. FENIX POLL FOR SUBJECTIVE TESTING

Table 5. FENIX Poll for Subjective Testing

FENIX Project Poll – http://fenix.iseage.org		
Name:		
Level: Expert Intermediate Novice		
Value		
Would you ever use FENIX or Reburn?	True	False
Would anyone you know use FENIX or Reburn?	True	False
Have you any need for a LIVE, bootable CD or memory based OS?	True	False
Have you ever used a forensic utility?	True	False
Do you see any need for a forensic utility or OS?	True	False
Easy to Use		
Were you able to create FENIX?	True	False
Did you run into any problems when creating FENIX?	True	False
Did Reburn work the way you expected it to?	True	False
Was Reburn overly complicated?	True	False
Was FENIX easy to use?	True	False
LIVE CD Competition		
Would you use FENIX, rather than Helix?	True	False
Would you use FENIX, rather than Knoppix?	True	False
Would you use FENIX, rather than any commercial Tool?	True	False
Would you use FENIX, rather than the Penguin Sleuth Kit?	True	False
Do you consider the advanced features of FENIX to be useful?	True	False
Effective		

Was Reburn overly simplified?	True	False
Were you able to create, customize, and redefine your LIVE CD?	True	False
Were you able to create a LIVE CD that allowed you to perform the actions you wished?	True	False
Were you able to add your own module?	True	False
Were you able to connect to the database to download a kernel or module?	True	False
Security		
Do you see FENIX as being non-destructive?	True	False
Were you able to establish that the filesystem was not modified? (during forensic acquisition mode)	True	False
Do you see Reburn as having a security issue?	True	False
Do you see FENIX as having a security issue?	True	False
Do you see the FENIX website as being a security concern?	True	False

BIBLIOGRAPHY

[1] S. Mocas, ``Building Theoretical Underpinnings for Digital Forensics Research", , *Proceedings of the Digital Forensic Research Workshop (DFRWS)* , August, 2003.

[2] J. Kornblum, Preservation of Fragile Digital Evidence by First Responders, Digital Forensic Research Workshop, Syracuse, NY, August 2002.

[3] (ITL Staff); General Test Methodology for Computer Forensic Tools"; (Word), Nov, 2001. http://www.cftt.nist.gov/Test%20Methodology%207.doc

[4] CSI/FBI Survey 2005, CSI Institute, 2003, http://www.gocsi.com

[5] Johnston, Dick. A Unique Challenge Requiring an Innovative Response. Abstracts from Cyber Crime Summit. http://www.crime-research.org/library/Abstracts%20for%20Cyber%20Crime%20Summit.doc

[6] Broucek, Vlasti. Turner, Paul. Computer Incident Investigations: e-forensic Insights on Evidence Acquisition . School of Information Systems, University of Tasmania.

http://forensics.utas.edu.au/files/EICAR2004.pdf

[7] Willassen, Svein Yngvar. Mjolsnes , Stig Frode. Digital Forensics Research.

http://www.telenor.com/telektronikk/volumes/pdf/1.2005/Page_092-097.pdf

[8] *Accuracy (Trueness and Precision) of Measurement Methods and Results – Part 1: General Principles and Definitions, ISO 5725-1:1994, Technical Corrigendum 1* Published 1998-02-15, International Organization for Standardization, Geneva, Switzerland, 17p.

[9] *Accuracy (Trueness and Precision) of Measurement Methods and Results – Part 2: Basic Method for the Determination of Repeatability and Reproducibility of a Standard*

Measurement Method, ISO 5725-2:1994, First Edition 1994-12-15, International Organization for Standardization, Geneva, Switzerland, 42p.

[10] "Best Practices for Seizing Electronic Evidence v.2.0" (handbook). International Association of Chiefs of Police, 2002.

[11] "January 2005 Internet Usage". Nielsen//NetRatings. http://www.nielsennetratings.com/reports.jsp?section=pub_reports&report=usage&period=monthly&panel_type=3. January 2005.

[12] Stewart-Rattray, Jo. *Business Computing Forensics.* Australian Computer Society. March 2005.

[13] Wikipedia. *El Torito (CD-ROM standard).* *http://en.wikipedia.org/wiki/El_Torito_%28CD-ROM_standard%29*

[14] Wikipedia. *GNU GRUB.* http://en.wikipedia.org/wiki/GRUB

[15] IBM. *Inside the Linux Boot Process.* http://www-128.ibm.com/developerworks/linux/library/l-linuxboot/fig1.gif

[16] Gleason, BJ and Fahey, Drew. Helix 1.7 for Beginners: Manual Version 2006.03.07. http://www.e-fense.com/helix/

[17] Knoppix Design Team. First Responder Guide for Law Enforcement and Corrections Officers. 2003.07.01.

[18] Grundy, Barry. The Law Enforcement and Forensic Examiner Introduction to Linux A Beginner's Guide. 2004.01.01

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis. First and foremost I would like to thank Dr. Doug Jacobson for his belief in me and his support throughout the research and completion of FENIX. I would also like to thank my committee members for their efforts and contributions to this work: Dr. Thomas Daniels and Dr. Steffen Schmidt. I would additionally like to thank a few of my colleagues for their support and interest in the project; Lt. Aaron Delashmutt, Tony Atilano, Julie Rursch, Noah Korba, Rachael McCormick, Chris Shoun, and Eric Anders. Finally I would like to thank my parents, John and Cindy Howard, for encouraging my further education while instilling a persistent attitude in my work ethic.