5-2004

Software integrity management system.

Joseph H. Brown 1980-
University of Louisville

SOFTWARE INTEGRITY MANAGEMENT SYSTEM

By

Joseph H. Brown
B.S. University of Louisville, 2002

A Thesis
Submitted to the Faculty of the
University of Louisville
Speed Engineering School
in Partial Fulfillment of the Requirements
for the Professional Degree of

MASTER OF ENGINEERING

Department of Computer Engineering and Computer Science
University of Louisville

May 2004

SOFTWARE INTEGRITY MANAGEMENT

By

Joseph H. Brown
B.S., University of Louisville, 2002

A Thesis Approved on

May 2004

By the following Thesis Committee:

Dr. James Graham, Thesis Director

Dr. Mehmed Kantardzic

Dr. Larry Tyler

Dedication

This thesis is dedicated to my fiancée Anna Richter and my family who have given me so much support these past years during my attendance at the University of Louisville J.B. Speed School of Engineering.

ACKNOWLEDGMENTS

The author would like to thanks Dr. James H. Graham for his guidance and support through out the entirety of the project. He has shown superb leadership and outstanding encouragement that has lead to the completion of this project. The author would also like to thank Dr. Mehead Kantardzic and Dr. Larry Tyler for being on the thesis overview committee.

Special thanks to Ronald A. Lile who has been an outstanding supervisor and good friend to the author during the author's masters candidacy. Also special thanks to Eric Kramer, Yindong Yu, Jared Baldridge, and Dr. Carol O'Conner for their aid and direction contributions to this project. Thanks also go to the author's family for providing support and encouragement through out the years of attending University of Louisville Speed School of Engineering.

ABSTRACT

The purpose of this thesis is to design, implement, and evaluate a software package that is mutli-platform and will provide software integrity management (SIM). The software package is implemented in Java and will perform two hashing algorithms, Message Digest version 5 (MD5) and Secure Hashing Algorithm 1 (SHA-1), in order to verify the integrity of executable files. These records of executables and their hash value will be stored in flat database files. The database files will be stored off site on multiple servers. Each server will hold a file corresponding to the hash algorithm that was used. By storing the files off site, the users of the SIM package will be guaranteed a certain level of security and assurance that their executable files have not been tampered with.

With the growing threats of security exploits and viruses, it is important for average users to be able to have this level of security. The security of the files off site will be as good as the security of the servers themselves. For this reason the server machines will be Linux machines since they are less susceptible to viruses. The server administrator will still have to keep up with security patches in order to avoid exploits, but the job will be less time consuming without having to worry about virus definitions. Initial testing using the GNU Compiler for Java (GCJ) in the Linux environment showed an increase in computational speed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The modern world today is becoming increasingly dependent on advanced technology. Technology provides solutions to major problems ranging from data storage to data processing. As technology is rapidly growing and changing, so are security threats, system intrusions, and viruses and Trojan horse software. Security exploits and virus attacks were common among home users and companies that had internal workstation computers, prior to widespread internet availability. These systems were isolated from the possibility of an attack or exploit except from that of a normal user of the system.

The majority of these systems were never connected to the Internet, and those that were usually connected by way of modem and did not stay connected for long periods of time. Because of the advancing of technology and lower costs of computer parts, the Internet has grown exponentially as well as the number of home users who connect to it daily. More recently, broadband, low-cost Internet access has become more widely available. As a result, more computers are connected to the Internet nearly twenty-four hours per day. With systems accessible to nearly anyone at any time during the period of a day, someone will eventually find a way to break into that system.

Most home users are not accustomed to updating the critical update components and updating virus definitions; these users are the first to get their system compromised or infected with a virus. Most of time a user is not even aware that their system has been compromised; and the attacker has unlimited access and control over the user's system.

There are several ways to prevent these attacks. The user could install some sort of software firewall solution, install an intrusion detection application, keep up with critical updates, and/or keep virus definitions current. Even if all of these weekly tasks are kept current, viruses and exploits arise almost daily. At this alarming rate, an intruder could break into a system and put Trojan horse programs on a user's machine before the user is even aware of the compromise. Worse the intruder could replace known good programs that are run daily that would allow the intruder access at any time or transmit the user's sensitive data to the intruder.

Some software utilities such as Tripwire use hashing algorithms to ensure that the programs are not switched with Trojan horse programs, but still the clever intruder could gain access and find the database file that holds the hash values and switch some values in order to not alert the user. Usually an offline file is suggested by means of a floppy disk or some other external media, but the average user would either lose the media or just forget about doing scheduled checks. A task such as this should be more automated and invisible to the user.

The solution presented in this thesis is to implement a similar hashing program with standard hashing algorithms that will store the files offline via network communication to secure servers. The program will be implemented in Java but with an interesting aspect. Java can run in multiple environments as long as the Java runtime environment is installed, however, by using GCJ (an open source java compiler) the Java program can be compiled into native machine code that would dramatically reduce processing time. The program will be designed and implemented in a pure object-

oriented language but will be compiled in native operating systems in order to increase the speed of the program.

The second chapter will provide an overview of relevant literature search for this project. The third chapter will discuss the overall design layout for the project including server and client layouts. The fourth chapter will cover the implementation in detail of the project. The fifth chapter will cover the testing approach of the project. The final chapter will review the overall project and the testing conclusions as well as layout a path for future research.

CHAPTER II

LITERATURE REVIEW

2.1 Computer Security

With the explosion of the internet, most of the world has become connected. With all of the connectivity comes an increase in not interoperability, but also possible security intrusions. Since the mid 1990's, logged incidents of security intrusions have become alarmingly numerous. In 2003 alone there were over 3,000 vulnerabilities and more than 115,000 security incidents reported according to CERT [25]. The definition of an intrusion becomes vague because of the many types of security breaches that exist from viruses and worms to program vulnerabilities. There are a number of false reports of people who believe they are being invaded because of the misunderstanding of what a true security intrusion is. Every personal computer has ports or "doors" that any outside program can try to access. Whether or not access is allowed or denied is up to the personal computer hosting these ports.

Each port is labeled with a number such as 21, 22, etc. These certain programs or "services" that use these ports are part of functions such as web servers or ftp servers which are generally geared to server machines; and these ports usually have a great deal activity from outside programs. Every computer communicates on some port when interacting with each other, and when a computer notifies a user of activity on a certain port, sometimes it can lead the user to believe that the machine is being invaded. When

there is activity on a port of a machine, it does not necessarily mean that someone is trying to invade your machine [13].

In the case of someone accessing these ports, it does not necessarily guarantee that the intruder has successfully gained access to the machine. The misunderstanding of what an actual intrusion is considered has caused confusion in society. Many commercial solutions were developed and rushed to the market and gave people a false sense of security. Naturally, when a warning alerts the user that a port is being accessed, the user assumes that they are being invaded by an intruder. These commercial solutions were hurriedly released for single user machines to accommodate the rising worries of consumers that were scared of being invaded or "hacked". This was also a chance for companies to capitalize on security programs. Most users aren't aware that there is no need for a security program to try to lock down everything. Even password rotation is not as effective when users just append letters on the end of their current password. This coping technique dealing with having to change their password is actually degrading security [14]. Classic security solutions such as firewalls or intrusion detection applications are still some of the better solutions for home users pending they are configured properly.

2.1.1 Firewalls

Firewalls have been used for security since the early 1980's. Most of the research done on firewalls was usually geared toward military purposes, but firewalls have become fairly common to the commercial environment and to the home environments.

Firewalls are either hardware or software solutions for keeping some security on home systems or internal private networks. First an examination of the basic technologies of a firewall will be conducted. The two basic technologies of a firewall are *packet filtering* and *application proxy gateways* [9]. Packet filtering is sorting all of the incoming traffic which based on where the packet needs to go. If a packet is going to a port that it should or should not be, then firewall will decide to let the packet pass or not. An application proxy gateway is a more complex system. It will act as middle man between the client and the data received or sent. The application proxy gateway will decide if the data should pass based on the data, not its destination. These technologies are implemented in either hardware architecture or software architecture.

The hardware architecture is a physical machine that is the firewall. It is a physical point between the home user or commercial user and the internet. The firewall will be the user's first line of defense. The software firewall is a program that is installed on the user's machine. The currently most used firewall is software firewalls because they offer easier set up configurations for most users. A hardware solution is usually better because of the greater customization that can be done with a physical firewall. The physical firewall usually intimidates home users by its cost and sometimes quite confusing set up configurations. A technician who is capable of such tasks is usually hired to configure such a solution. Companies are the ones who more often implement hardware solution firewalls. With the technological advances, users are exploring other solutions for security [2]. Some of these solutions include "intelligent" firewalls. The types of firewalls are usually referred to as Intrusion Detection Systems (IDS). Another

popular method of security is ensuring binary file integrity. These two methods will be discussed next.

2.1.2 Intrusion Detection Systems

IDS were developed in the early 1990's when the occurrence of intruder attacks were on the rise. IDSs were developed in order to block or alert the user that the system is being invaded. IDSs are still infantile in research compared other areas of research. The early forms of research in protection from intrusions were firewalls. Perdue University and the Brandenburg University of Technology at Cottbus have made considerable amounts of contribution to the research of IDS. Some common characteristics of an IDS are unsupervised capabilities – the IDS must be able to run without user intervention – , fault tolerance – resistant to a system failure or sudden power loss – , resist subversion – must be able to resist an intruder's attempt to be taken over – , minimal overhead – use little system resource so the machine can operate at optimal performance – , observe deviations – be able to catch when a sudden change occurs within the system – , easily tailored – be simple enough to change or customize – , cope with changing system behavior – if new programs are installed by the user the IDS must recognize it as a normal change – , and difficult to fool – not be tricked by an intruder that they are a valid user [4]. However, there is one major drawback to the IDSs, there needs to be someone monitoring them for a large part of the day. With more advanced tools available for people to create viruses and find ways to exploit existing software, a trained professional would need to be involved with the IDS and be able to distinguish between a real threat from a false alarm. [16]. There are two specific IDS models that will be discussed, anomaly detection and misuse detection.

There are two specific models used when implementing an IDS, they are anomaly detection and misuse detection. The anomaly detection model looks for any deviation of system behavior that is not normal, such as an irregular service that may be running. The misuse model will look for users or intruders that are trying to access parts of a system where permission is otherwise denied. These models are only half of the process. A form of artificial intelligence must be chosen in order to carry the automated tasks of an IDS. Experts Systems and Autonomous Agents are the two most popular choices for implementing such an IDS.

2.1.2.1 Expert Systems

Expert Systems have been in development since the mid 1970's. An expert system is a set of rules that apply to a specific area of expertise, hence expert systems. One would only ask a doctor about fungi diseases, not a chef. So the expert system only has knowledge that is specific to its area of expertise. One of the first Projects developed using the expert system was the MYCIN project [1]. It was system that could produce medical diagnosis of blood infections based on certain answers it was provided. Other systems such as the Ventilator Manager (VM) and TEIRESIAS were also expert systems developed around a medical knowledge. An expert system can be classified as a technique for detecting intrusions, while its counterpart autonomous agents are an actual technology. This technology can implement expert systems or any other type of technique available for detection.

2.1.2.2 Autonomous Agents

Autonomous agents have been around since the 1990's. Agents were mostly introduced for distributed purposes; however, the use of autonomous agents has spread to many other applications. The idea of the autonomous agent is that it is completely independent. The agent should be able to run effectively and efficiently enough to support itself, and at the same time be able to interact with other agents within the same system. A system can be composed of several autonomous agents that can effectively communicate with each other in order to process a job or monitor events [1]. With these artificial intelligent approaches, projects such as Adaptive Intrusion Detection (AID) and Autonomous Agents for Intrusion Detection (AAFID) have been implemented.

2.1.2.3 Adaptive Intrusion Detection (AID)

AID was a research project done at the Brandenburg University of Technology at Cottbus from 1994 to spring 1996 using the UNIX environment. The environment consisted of agents that gathered audit data from workstations and sent the data to a manager where an expert system analyzed the data in order to interpret. The data sent from the agents to manger was transferred using secure RPC. The AID system was platform independent and could process more than 2.5 megabytes of information per minute [26].

2.1.2.4 Autonomous Agents for Intrusion Detection (AAFID)

The AAFID was originally released in 1998 and has released AAFID2 in 1999. This system is developed in a scripting language called Perl version 5. The system has built-in agents and monitoring rules ready to run "out of the box". The system has been tested in the UNIX environment and is currently in testing for the Windows environment. These agents sole job is to monitor either network environment or individual machines alerting the user when there is an anomaly detected. These research projects have progressed tremendously over the last few years, but commercial products are not far behind implementing their own proprietary code to sell on the market.

2.1.3 Binary File integrity

Binary file integrity is any method that ensures that a binary file – such as executables or even office documents – has not been altered in any way. There are many reasons users want to have file integrity. The most important reason is for security. If an executable has been changed to do something other that its original design, a user could suffer loss of control of their machine, or worse loss of data. Also, users can use the methods of file integrity to guarantee that the files received are from other trusted users. Digital signatures and the public key infrastructure are types of authentication strategies that are used. The most common form of file integrity checks is the verification of a hash sum.

Hash sums are mathematical formulas that produce checksum sums of files and produce a hexadecimal format number that is unique to a file. These numbers can be used to verify that the file is the same as when it was checked the last time. There are several hash functions available to use in order to complete an integrity check. The most popular method to use for file integrity is Message Digest version 5 (MD5) that was produced by Professor Ronald L. Rivest of Massachusetts Institute of Technology (MIT). RFC1321 explains that the MD5 algorithm takes an arbitrary file and produces a 128 bit "fingerprint" or "message digest" of the input [27]. The algorithm was originally intended for digital signatures. Other algorithms exist as well such as the Secure Hashing Algorithm (SHA1) and RIPEMD-160 hash function.

These other algorithms have been developed in order to add to the security of creating more unique numbers of a file and guaranteeing its integrity. The SHA1 algorithm produces a 160 bit output of a file. The SHA1 has been officially adopted by government as the Secure Hash standard of the Federal Information Processing Standard (FIPS) and is specified in RFC3174 [28]. The RIPEMD-160 is also 160 bit but it has not become any standard. These hashing algorithms are available for use through various applications. The market has various software applications for computer security.

2.1.4 Commercial Security Products

Available commercial firewall solutions include Symantec's Norton Firewall and Microsoft Windows XP built-in firewall. Also, available commercial IDS products on the market are Symantec Intruder Alert, Microsoft's IDS, etc. Applications that perform

file integrity checks; Tripwire is one of the more popular programs on the market. There are several other products that exist in each category in the market place. Concerning Tripwire, it is both a commercial product and an open source product. Tripwire is a program that will check your programs to make sure that none of them have been modified by an intruder to do something different. This is done by taking a snap shot of your system programs. Tripwire also has Tripwire for servers available with a centralized management console to handle administration and monitoring. The decision of which tools should be used is based on the ease of use, amount of money to spend, and the support and documentation available to the user. Another alternative to spending money is using open source products.

2.1.5 Open Source Products

Open source products are free to use but there are some down sides for the average users. Even though these are free to use, the documentation may not always prove to be complete useful. Some open source projects have great documentation while others appear to lag in that area. Sometimes, numerous searches must be done on the internet to find crucial information about the open source application. Using open source software packages are usually developed for the Linux operating system. There are many versions of open source firewalls, IDSs, and file integrity applications available for use. Tripwire is one of the few applications that have a closed source application and an open source application. All the applications mentioned could also be used to perform computer forensic analysis.

2.2 Computer Forensics

Computer forensics is a growing field in today's society. Computer forensics is the act of analyzing a computer system by a serires of analysis ranging from searching all log files to performing a bit by bit search on a hard drive. As mentioned, intrusions are becoming more and more frequent. Once a machine has been compromised, there are some decisions that must be made. First, depending on the severity of the intrusion, is it worth the time and effort to take a production machine offline in order to analyze it. Second, does the analyst have permission to do so? Third, how will the analyst save volatile data if it is pertinent to the analyst? These decisions are debatable even today [3]. After vigorous decision making, assuming the analyst has the permission to go a head and do the analysis, the analyst must make sure not to modify the data in anyway. If the evidence is to be presented in court, then the analyst must be sure to keep the original state of the information they collected and analyze a copy of the data.

2.2.1 Legal Aspects

More and more legal cases are using evidence that is obtained from a seized computer. The information obtained from these computers can be critical. If an investigator does not follow the exact procedure; or if the investigator accesses the hard drive in such a way that data is lost, the evidence will not be admissible [3]. The investigator needs to be aware that they should be a analyzing a copy and not the original.

There are several ways to make copies of hard drives. There are commercial and open source versions of these solutions.

2.2.2 Commercial Products

One of the more popular choices for forensic tools is Encase suite. Encase is a comprehensive suite of tools that will allow the imaging of hard drives and aid in a detailed analysis of these hard drive images. The analysis would help develop a time line that would aid in the approximation of the point of entry of an intruder. Although popular, Encase is quite expensive. There are several other forensic tools besides Encase such as the computer incident response suite from New Technologies Inc. (NTI) and TapeCat from Sanderson Forensics located in the United Kingdom (U.K.). Many are for commercial use, but there are open source tools as well. Some of the most popular open source tools are offered from the company @stake. This company specializes particularly in security. They offer many security auditing utilities and many forensic utilities. One of the more popular @stake tools is The @stake Sleuth Kit (T.A.S.K) along with Autopsy – a graphical web-based front end for T.A.S.K. [29]

2.2.3 Open Source Products

The T.A.S.K application is a command line driven utility that will analyze images of hard drive. It will support both UNIX/Linux environments and Windows

environments. Within this package, an analyst can look at low level file systems and find evidence that an intruder may have tried to delete files or tried to cover up their tracks with another program. This tool is very versatile in how it can manipulate the image. The Autopsy Forensic browser is a simple web page that will interact with T.A.S.K. making a powerful combination that allows a complete forensic analysis [30]. These tools as well as many other tools are freely downloadable for users or system administrators to use. The author's intent is to create an open source application freely usable to the public. The next chapter will discuss the layout of the open source application Software Integrity Management.

CHAPTER III

SOFTWARE INTEGRITY MANAGEMNENT DESIGN

3.1 Initial Design

The early stages of the application included the creation of a module inserted into the existing forensic tool T.A.S.K to add more functionality. This tool would only be for the Linux/Unix platform. The insert would perform "wizard like" operations to simplify the tedious work of a forensic analyzer. A forensic analyst could spend numerous man hours searching through log files in order to find the point of entry of an intruder without knowing where to begin. This module would aid the analyst in finding a starting point. The module would scan an image of a mounted drive and perform hash functions on common binary files of the mounted image and compare them with the hash of known good binary files. The operation would also complete a comparison analysis in order to determine which operating system distribution was being analyzed. The benefits of knowing this type of information could be used in order to help give the analyst a starting point. To accomplish this type of task there were two major factors that would have to happen. There would have to be a repository of such information that could be easily – and securely – accessed by a client machine. See figure 3.1 for the initial design layout. Many draw backs and limitation to this model design were found from the start.

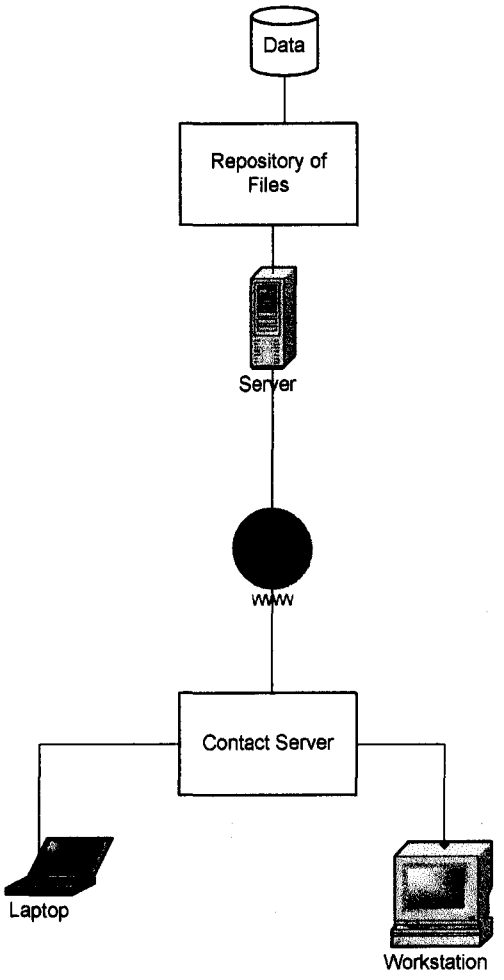

Figure 3.1 – Initial Design Layout

First, with the various distributions of Linux/Unix it would be very hard to find a comprehensive repository of the binary files for one distribution. If such a repository existed, the amount of information searched would be become numerous and slow down processing time. Also, there is no redundancy for the repository. If the machine where the repository was stored ever failed, the module would be useless. Also, not every user of a personal computer would have the knowledge and experience of a security expert. For these reasons, an alternate design was needed in order to address these issues. Also the design should include a way to run on Windows compliant machines as well as Linux

compliant machines. Most users currently use Windows, and it would be beneficial for these users to have access to an application that would serve as a computer security utility and/or forensic helper. The alternate design created would include these enhancements and become a stand-alone application.

3.2 Alternate Design

The new design created was titled Software Integrity Management (S.I.M.). The S.I.M. application would solve all of the drawbacks of the initial design. The layout for S.I.M. is a distributed model. There will be an ability to have multiple clients that will calculate two hash functions for each binary file on their system – message digest version 5 and secure hashing algorithm 1 – and send the files off site to multiple servers. There will be two servers – one for each hash function respectively – that will hold a flat database file for each client node it serves, see figure 3.1. The files will be read only in order to keep write protection limited to the server portion of the application. Also, each file has a date last modified field that will help ensure that the files have not been altered by any outside source. Each client node will have a database file on each server that is uniquely identified by a combination of the client's IP address, the client's machine name, and hash sum of the client's IP address. The dual files imitate a mock redundancy for each client node which will provide fault tolerance in the case of a hardware failure of either the client or the server; or a compromise of one of the server nodes or client machine. There were specific goals in mind while designing the architecture of the application.

The intended of the application were the following. First, to create a software package that would support multi-platform usage seamlessly with client level control. Secondly, to have an interface that would allow ease of use for any user with any level of experience or knowledge. Lastly, to the lay ground work for future implementations of an intelligent analyzer that will help users maintain the integrity of their operating system files. In order to address the goals of the author, the application was implemented in the following way.

The menu interface will have as few choices as possible to allow less confusion and more understanding of the application function. The chosen language to implement the application was java since the class files generated are created in byte code and can be executed on any given machine with the aid of a virtual machine. The classes intended to be used for the software package would be a client class, server class, and menu class. These few classes were designed in order to add ease of use, flexibility, and reusability for the application; however, having such few classes and needing a virtual machine caused some limitations to the application package.

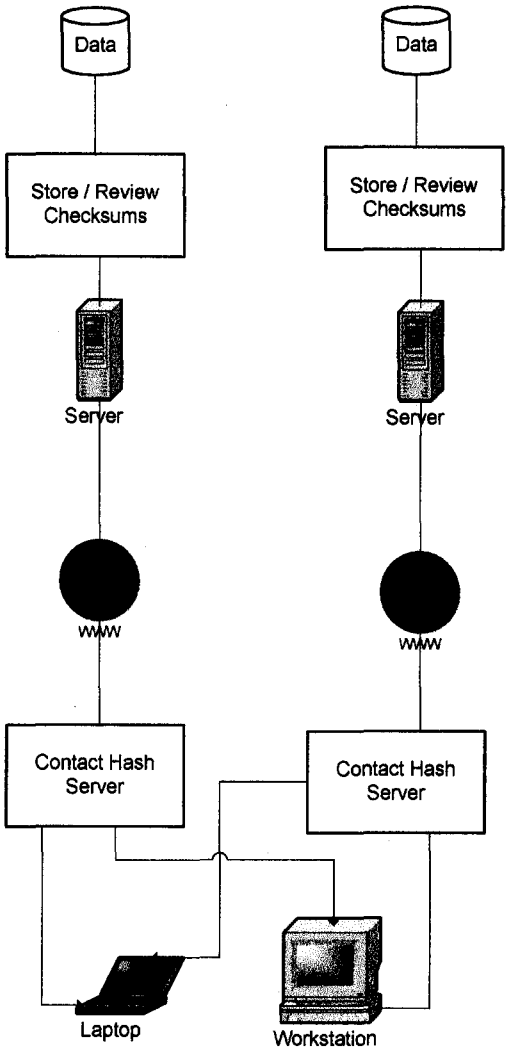

Figure 3.2 – Alternative Design Layout

3.3 Limitations

Running class files using a virtual machine is generally slower since the byte code has to be interpreted by a virtual machine. If the code was converted into native machine language, then the processing time could be saved when computing the hash functions. Particularly Windows has many executable files in the root system folder which is either "WINDOWS" or "WINNT". Linux binary files have multiple directories with normal

user binary files and administrative user binary files; however, the time to calculate hash sequences of the Linux binary files is usually less than the time it takes to calculate the hash sequences on Windows binary files. Since the application's goal is to be efficient on a multi-platform level, an element of speed would need to be incorporated.

The class design was also a limiting factor because the amount of methods that filled each class became numerous. More often than not another "useful method" would be thought of to make the application more user-friendly. The classes became overwhelming to sort through all of the methods. Because of "bloated" classes, confusion about the state of the program would be encountered. The classes would need to be further broken down into more classes in order to establish the reusability of the application as well as the organization and fundamental characteristics of being an object oriented program. With these two major revisions, the software package would be able to run in less time with more efficiency.

3.4 Revised Design

The revisions will consist of two main factors. The software will be compiled into machine executables for Windows and Linux; and the application will be broken into more classes for more reusability and efficiency. The gnu GCJ compiler will be used to compile the software into native machine code. The gnu GCJ compiler is packaged with the gnu GCC compiler version 3. This compiler will convert Java code into native machine code for Windows or Linux. The windows version is of GCJ can be used with

Cygwin – a windows compliant tool kit with Linux utilities. By creating the binary files, process time will shortened immensely.

The classes will be broken into the following classes: client, md5Server, sha1Server, secureHash, integrityCheck, transport, generateList, menu, menuChoices, menu, keyboard, functions, directory, choice1, choice2, choice3, reget, resend, and drivers for the client and server side applications. The same basic system from section 3.1 is still applicable even when several classes exist. By expanding the class list, future enhancements of each choice is simplified. Also, if more choices needed to be added for future functionality, they could be added easily. A more detailed explanation of each class and implementation is covered in the next chapter.

CHAPTER IV

IMPLEMENTATION

The choice of object oriented language for the implementation was Java. Java is a language that was designed by Sun Microsystems in order to be a multiplatform programming solution for creating applications. With this flexibility, the project could be written in such a way to be compatible with Windows compliant machines and Linux compliant machines.

4.1 Initial Implementations

Some of the early implementations included using PERL to create and insert a module into the T.A.S.K program created by the @stake organization. This PERL module would be a "wizard" for the T.A.S.K utility used to search a mounted image and help complete a forensic analysis. The module would scan all binary files in the common system paths and calculuate several hash values of the files. These files would be used to compare to a repository of known good system binary files in order to narrow down what operating system was being analyzed and if there were any changed binary files before the system was shut down. Later, the program became a standalone Java program that would perform the forensic analysis without the aid of T.A.S.K. The next phase would implement servers that would store large database files of different workstations for the Linux / Unix operating systems only and their corresponding hash value. There would be

a total of four hash algorithms used. The last revision before the final project was solidified into its current state was to make the a program that would manage software integrity by using four hashing algorithms to check the integrity of the binary files and storing the database files offline on four servers each corresponding to a hash algorithm. The total number of algorithms used was reduced to two since the remaining two algorithms were available in the Java language.

After final revisions of design layouts and choice of language for the final project, an initial implementation of the project had begun using Borland Jbuilder version 8 personal edition. This Integrated Development Environment (IDE) was chosen because of three reasons. First the author was more familiar with this particular IDE. Secondly, the IDE generally had a faster response time than previously used IDEs. Thirdly, this particular IDE included the Java Library version 1.4.1 that would be used to implement the application.

4.2 Java Libraries

The Java Library used was version 1.4.1. This version of the library has many useful features that the project took advantage of. Some features that were used were standard built-in hash methods of Message Digest version 5 (MD5) and Secure Hashing Algorithm 1 (SHA1) that are part of the Java Security package. Another useful method that the String class offered was the "split" method. This method would split a long string based on the regular expression or character (s) given in the argument. These methods were essential to the functionality of the project; however, a special split

function was written in order to remedy compatibility issues in a GNU compiler used that will be discussed in section 4.5. The overall project was programmed into a Java package. The Software Integrity Management (SIM) package was created in order to make the usability and portability of the project simple.

4.3 S.I.M. package

The S.I.M. package's organizational structure is composed of the sim directory and two sub directories named agents and interfaces, see figure 4.1. In the agents directory there are a total of seven classes – client, md5Serer, sha1Server, generateList, secureHash, integrityCheck, and transport – which are the vital components of the S.I.M. application. The interfaces directory consists of ten classes – menu, menuChoices, choice1, choice2, choice3, keyBoard, functions, directory, reget, and resend – which is the rest of the S.I.M. application. The interface directory of the package is responsible for the interfacing with the user. Information is received through the interface and the necessary agents are created for execution from the agents directory. The class functionalities are as follows.

The menu class handles displaying the menu for the program's command line interface choices. The menuChoice class handles the choices passed from the menu class. Each choice class handles the respective choice. The keyBoard class is responsible for handling keyboard input. The functions class is responsible for handling any special function needed by the S.I.M. application. The directory class handles functions performed for adding, editing, and deleting custom directories; however it will

not be implemented in this version of S.I.M. because of the added complexity from the user perspective. This will be discussed in the next section. The reget and resend classes are responsible for handling if files needed to be re-sent to the servers, or re-received from the servers. The client class handles the interfacing of secureHash class and integrityCheck class with the client machine. The md5Server class and sha1Server class are used to run remote servers. The secureHash class handles the hashing of the binary files for the S.I.M. application. The integrityCheck class handles the checking functions used to detect any integrity breaches. The generateList class is used by the secureHash class and the integrityCheck class in order to generate the list of common binary files of the client machine. Method summaries for each class can be found in Appendix IV.

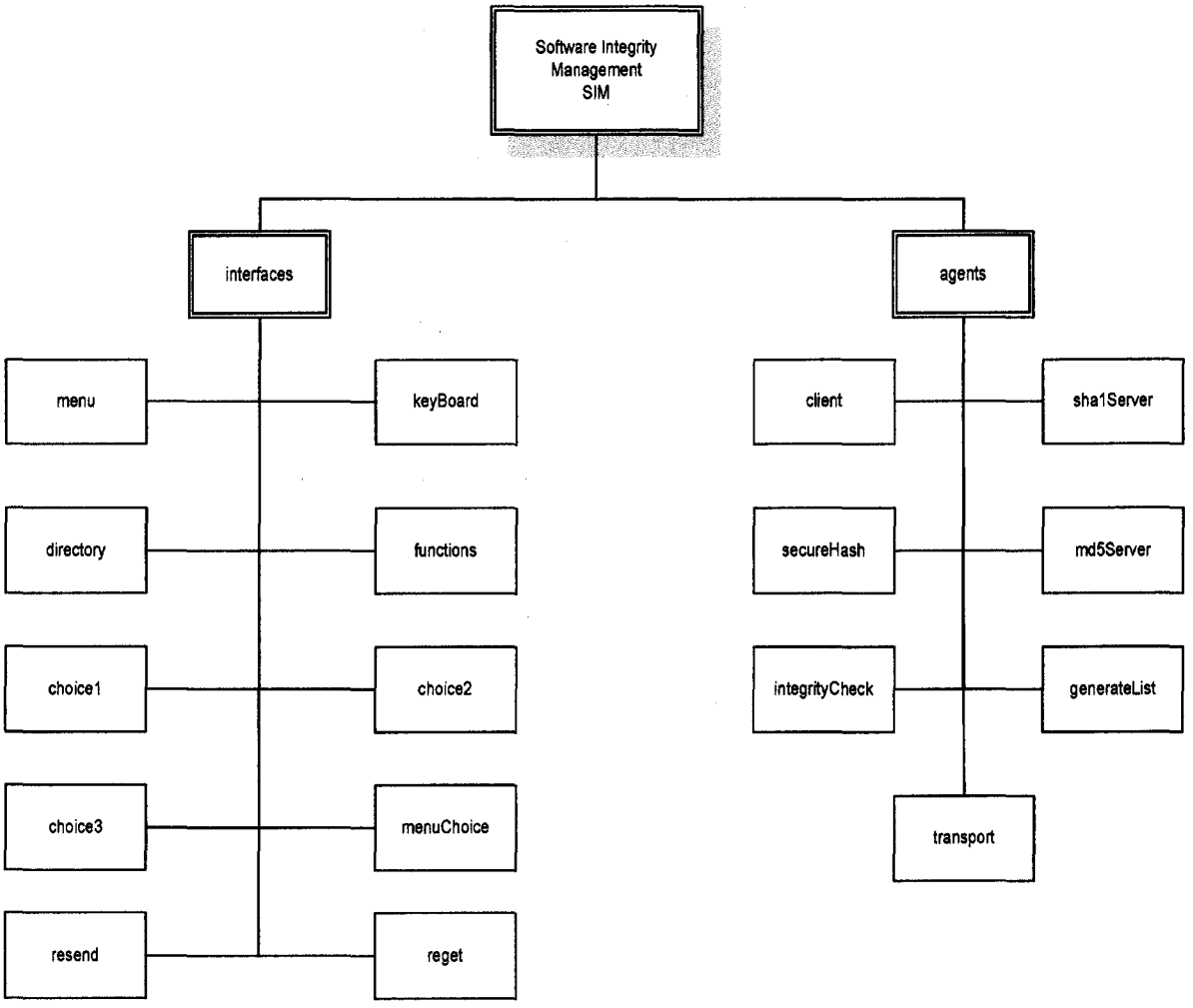

Figure 4.1 – Organizational Structure of S.I.M.

4.4 User Interface

The user interface for the SIM package is command line driven. There are 2 main options. 1.) Begin Secure Hash, and 2.) Check Integrity. Figure 4.2 shows a screen shot of the interface menu. The option to add custom directories has been omitted because this version of S.I.M. is focusing on the simplest interface for users; however the feature

is note worthy to mention since there are several future features that can be implemented with the

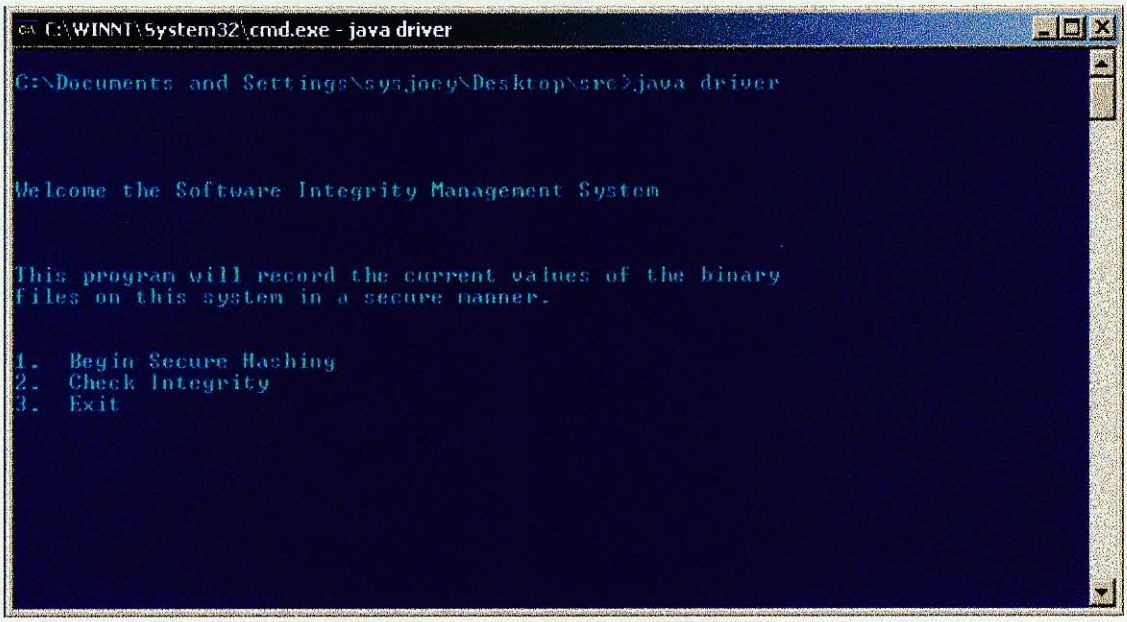

Figure 4.2 – S.I.M. User Menu

directory class. These options provide flexibility to the user so that custom directories for user programs and binary files may be added. The overall performance of the application will be decent and use a larger portion of processing time since the application is dependent on a virtual machine. The slightly slower processing of Java comes from the fact that the class files are in byte format and they have to be interpreted rather than a program written in C/C++ that uses the processor directly. See Figure 4.3 for option 1 execution example.

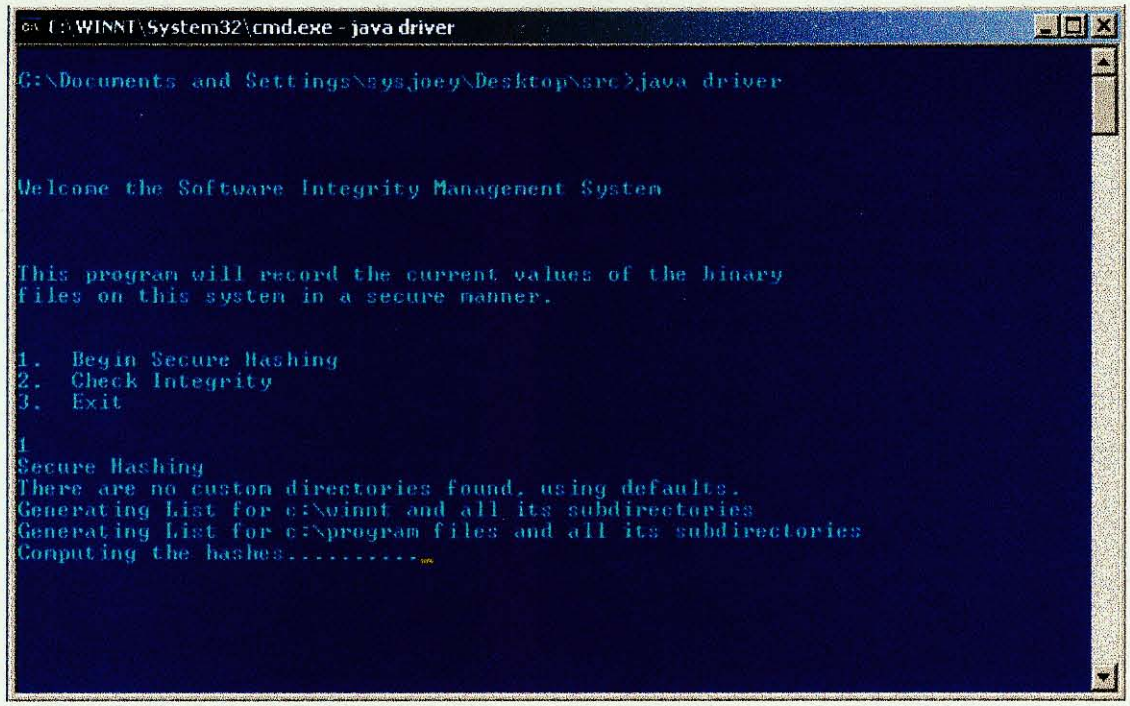

Figure 4.3 – Execution of Option 1

An alternate method to the virtual machine is to compile the program using the GNU compiler for java (GCJ). The next section covers more detail about GCJ. A Graphical User Interface (GUI) is planned for the next version of the application. If a GUI is developed for this version, benefits of the desired compiler GCJ will not be realized since the current version cannot create window components from Javax library.

4.5 GNU Complier for Java

The GNU GCJ compiler is available at http://www.gcj.org for free usage under the General Public License (GPL). The project has become part of the GCC suite since GCC version 3. GCJ has the ability to compile Java code into native machine code whether Windows or Linux/Unix. By doing this, the virtual machine is eliminated. Processing time is tremendously reduced when the virtual machine is eliminated and the Java code is converted into native machine code such as C or C++. There are a few draw backs to using GCJ. One is that GCJ will only convert Java code compatible with library version 1.3.1. Also, the compiler does not convert Javax or swing components into native windows of the operating system environment. This severely limits the code to command line interface only, and some of the Java functionality will be unusable. Overall, GCJ is good to use to speed up the processing time of the program. Detailed testing is discussed in the following chapter.

CHAPTER V

TESTING

Although there were many initial testing intervals to test the functionality of each class developed during the creation of the S.I.M. application, the major testing of the application consisted of three phases. The first phase was testing the speed of computation for the binary files on a machine in order to confirm the overall performance of reading and calculating the desired hash sums. The second phase consisted of verifying the accuracy of the S.I.M application in terms of its ability to detect extra binary files installed and/or changed binary files on the system. The final phase consisted of testing the interaction of different client versions, i.e. using the virtual machine version or using the compiled binary version, with the servers. Several pieces of equipment were needed to complete these phases.

The equipment used in the testing phase included two Dell Dimensions 2100 machines, one Libranet Linux 2.8.1 machine, and one Debain 3.0 Linux machine. The two Dell machines were Intel Celeron 1.1 GHz processors, 256 Megabytes of RAM, and a 40 Gigabyte hard drive. The two server machines hardware configurations were not as important because the performance was analyzed on the client side. The two servers were available to client machines via a 10/100 megabit network. Bandwidth was of no concern since very limited client machines would be accessing the servers. Also, during the first two phases the server machines were not used so a controlled environment could be created for the test clients. The above hardware used in the testing phases has some

effect on phase I because not all machines are of equal speed. All results of phase I following is based on these hardware configurations. The following section will discuss the computational speed of each of the versions of the S.I.M. application.

5.1 Testing Phase I – Computational Speed Test

Phase I was set up as follows, two client machines – one with Windows 2000 Professional Edition and the other with Debian Linux 3.0 Testing version – were configured with minimal installation options as well as the Java Runtime Environment version 1.4.2_04. Both ran the S.I.M. application fifty times each with the runtime environment and both ran with a compiled binary executable without the assistance of any virtual machine. Table 5.1 shows sample data of the time trial runs. The time is in the format of milliseconds in the table. For example Trial 1 under column 2 completed the hash computation in 285,871 milliseconds or in approximately 4 minutes, 45 seconds, and 87 tenths of a second. For the complete set of test data refer to Appendix I.

Table 5.1 – Sample Trial Times in Milliseconds

Test Runs	Windows with Java Virtual Machine	Debian Linux with Java Virtual Machine	Windows without Java Virtual Machine	Debian Linux without Java Virtual Machine
1	285871	166120	266564	208432
2	250770	154417	216451	207559
3	250209	156357	214778	222299
4	252663	155404	216241	121199
5	251602	156871	215009	120506
6	252673	155415	214719	120792
7	250520	154867	215570	120536
8	251481	155590	215050	121292
9	250501	155129	216411	120459
10	251883	155162	216392	121116

Each time was acquired by use of the Calendar class in order to get an instance of the time before the hash method was invoked. After the hash method was finished, the Calendar class was used to take another snapshot of the current system time and compute the difference between the two times. The times were then appended to a file in the local directory in order to view later. There were some difficulties implementing the time capturing function since the capturing had to be compatible across multiple platforms. The main difficulty was discovering how to find a class that would return milliseconds and operate correctly in both environments. The other difficulty was making sure that the classes used were implemented in the version of GCJ used to create the binary files.

Four T distribution tests were conducted to test the significance of the variables in relation to whether or not the virtual machine had an influence, the operating system had and influence, or a combination of the two factors had an influence. The results from the Minitab output are as follows.

Two-Sample T-Test and CI: Time1, Windowswith/withoutVM
Two-sample T for Time1

Windowsw	N	Mean	StDev	SE Mean
0	50	217648	7396	1046
1	50	252332	5167	731

Difference = mu (0) - mu (1)
Estimate for difference: -34684
95% CI for difference: (-37220, -32148)
T-Test of difference = 0 (vs not =): T-Value = -27.18 P-Value = 0.000 DF = 87

Two-Sample T-Test and CI: Time2, LinuxwithwithoutVM

Two-sample T for Time2

Linuxwit	N	Mean	StDev	SE Mean
0	50	126372	22114	3127
1	50	158646	14126	1998

Difference = mu (0) - mu (1)
Estimate for difference: -32274
95% CI for difference: (-39655, -24893)
T-Test of difference = 0 (vs not =): T-Value = -8.70 P-Value = 0.000 DF = 83

Two-Sample T-Test and CI: Time3, VM

Two-sample T for Time3

VM	N	Mean	StDev	SE Mean
0	50	158646	14126	1998
1	50	252332	5167	731

Difference = mu (0) - mu (1)
Estimate for difference: -93686
95% CI for difference: (-97939, -89432)
T-Test of difference = 0 (vs not =): T-Value = -44.04 P-Value = 0.000 DF = 61

Two-Sample T-Test and CI: Time4, NONVM

Two-sample T for Time4

NONVM	N	Mean	StDev	SE Mean
0	50	126372	22114	3127
1	50	217648	7396	1046

Difference = mu (0) - mu (1)
Estimate for difference: -91276
95% CI for difference: (-97875, -84678)
T-Test of difference = 0 (vs not =): T-Value = -27.68 P-Value = 0.000 DF = 59

Essentially the four means are significantly different from each other according to Minitba. Figure 5.1 shows a graphical display of the results.

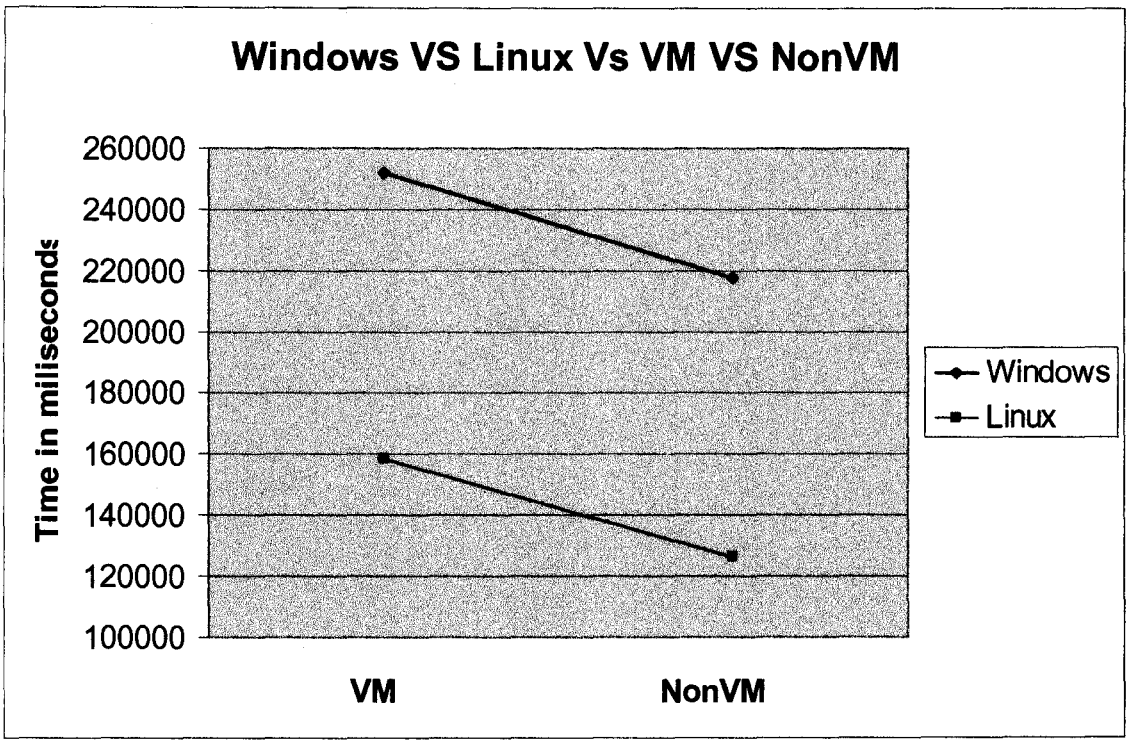

Figure 5.1 – Results of Comparing the Time Averages with the Four Factors

Most cases the test results were consistent as far as computational time is concerned. The average running time for Windows with the Java virtual machine was 252332 milliseconds. The average time for Windows without the virtual machine was 217648. The client machine was newly installed and had few programs installed except for the Java runtime environment. Most users have several thousand binary files in their Windows directory and Program Files directory. As the number of files increase, it would be likely to see an increase in the time for computing the hashes. The binary file was compiled using GCJ – a suite of Linux development tools that are Windows compliant – and then copied over to the client machine. It should be noted that the classpath had to be set to "." in order to make GCJ function properly. The commands used to create the executable file are as follows:

```
export CLASSPATH=.
gcj -c driver.java sim\agnets\*.java sim\interfaces\*.java
gcj --main-driver -o driver.exe *.o
```

Next the Linux Environment and its time trial phase will be discussed.

Linux has traditionally been an environment for development and this made it easier to make changes and add some nicer features when dealing the Linux environment. The average time for the S.I.M. application running with a virtual machine was 247,024 milliseconds, and the average time for without a virtual machine was 216,910 milliseconds. This was truly significant since it was the opposite with the Windows environment and the outcome was what was expected. However, the Linux environment had some issue of its own.

To start, not every Linux machine has GCJ although it is quickly becoming adapted by the GNU organization and being integrated with the standard GCC suite. But having GCJ is only part of the issue for Linux machines. The developers of GCJ are implementing the Java libraries to into code that can be converted to machine code at compilation time. The full library is not yet implemented, but most of the features the S.I.M. uses are standard enough to not need anything special. For that very reason, the functions class was created to handle things that were not available.

When the time came to test the Linux binary, everything went well except for the fact that no directories could be read for binary files. With further investigation, it was discovered that the Java security library had only been implemented in GCL version 3.3.0 or higher. This was definitely a problem since not every Linux machine is the same. With the help Yindong Yu, a configuration file was created using autoconf in Linux that would test for these packages to make sure the machine was able to run the S.I.M.

application. In the process a Makefile is created that will be able to "clean" or "make" the client or the server binary file. Once these files and source code files were put together, a gzipped tar file was created in order to be portable to any Linux machine in order to run the S.I.M. application. There is some noteworthy information that should be taken into consideration as far as the speed is concerned.

The Windows client machine only read two directories while the Linux client machine read six directories and still computed the hash sums faster with and with the Java virtual machine. The fact must be stressed that these both were very basic installs and these time trials were based on minimal amount of binary files installed. With regards to Windows, there are several binary files that get installed with general web browsing and the fact that many people like to download different programs and try them out. Keeping this in mind it was realized quickly that the amount of binary would become numerous and take an extreme amount time process all of them. Generally in the Linux environment only needed tools are installed, however, there are some users that may like to install several programs to test them out. Both operating systems would experience high computations time with the increase of binary files, but Windows would suffer extremely more since most users just install programs here and there. Usually a Linux operator will like to compile things by source; and if the user prefers to use binary installation files, generally there are not a lot of binary files that have to be installed. Testing phase I was a success in proving that Linux provides a bit more robustness for processing and computation. After the completion of the testing phase I, the accuracy of the integrity check was to be tested in phase II. The next section will discuss the results of testing phase II.

5.2 Testing Phase II – Accuracy of Integrity Check

Testing phase II consisted of running the application and changing or adding/deleting binary files. This phase was only conducted on the Linux system rather than its counterpart Windows. There were two reasons for this decision. First, the S.I.M. was written in Java so it has cross-platform capabilities which means the same integrity check algorithm will be used for either operating system. The second reason is for the obvious speed the Linux environment was capable of producing. It should be noted that these tests were also conducted on the client machine entirely to have a controlled environment. The Java code was simply modified to not delete the files to be sent, and only use the local machine for sending files to the server and receiving files from the server.

In order to test the accuracy of the integrity check, there were two things to keep in mind. First the original files of the machine scan would not be altered or deleted in order to have a base line of the good binary files. Second, the files altered were copies from the original base line which were used during the integrity check. Realistically, the files would reside on two different servers and the local file that contains the good binary files would also be deleted upon completion of the transmission to server. There are several possible combinations in which the check could be completed. There is the possibility that only the md5 file is good and all decisions are based on that file; or there is the possibility that only the sha1 file is good and al decisions are based on the file. The last combination is that both are checked simultaneously and a decision is based on that

comparison. These three main groups of comparisons are dictated by the initial check of the integrityCheck class.

When the files are transferred to the client an initial check of those files are checked to make sure they are 1) the same length and 2) have the same binary file paths. If they do not meet the criteria, the servers are contacted and the server that possesses the file with a date that is least recent will be the file used for the comparison. From then on, decisions are based on the integrity of that particular file. This unique feature about this process is that one of the files from the multiple servers will generally be guaranteed its integrity. This means that it is unusual for both servers to be compromised at the same time. Once the decision about which file is going to be used, there are four possible decisions to be made. The system is either 1) good and resubmits both good files to both servers replacing the old files, 2) the system has detected extra binary files and advises the use to make sure no one has installed something without their permission, 3) the system has detected altered binary files and advise immediate action for the use, or 4) a combination of two and three.

These options are also available in the event the files pass the initial file check of the server files. Naturally, there were three main tests for this testing phase. There is the test of using the md5 file solely, using the sha1 file solely, and using both files to determine if the system binary files are valid. Table 5.2 has sample information about the test results from these various test. Only ten tests of each segment were completed. The full set of results can be found in Appendix II.

Table 5.2 – Sample Results from Accuracy Test

Trial	Detected extra binary files	Detected changed binary files	Detected both extra files and changed files
1	Pass	Pass	Pass
2	Pass	Pass	Pass
3	Pass	Pass	Pass
4	Pass	Pass	Pass

It was not as extensive as the time trials since integrity checks should not result in detecting changes frequently; however, it is an important test to make sure all of the functionality is working properly. Also, while difficult to quantify the accuracy test, it should be noted that more testing samples from outside sources should be used to calululate a more quantifiable number. It was discovered that one of the methods had cause the application to crash because of index issue with an array. Phase II was helpful in testing the overall performance and validity of Java of the Java code of the integrityCheck class if nothing else. Lastly, testing phase III – client interoperability – will be discussed in the next section.

5.3 Testing Phase III – Compatibility Check of Client Versions

Testing phase III consisted of testing the client interaction with the server over a true networking environment. Two Linux servers were used for the servers, and the Windows and Linux client machines were used to test the client functionality. Each of the four clients was used to interact with the servers to make sure there were no compatibility issues between client versions versus server version. The results of this testing phase proved there were yet more factors that affect the networking environment.

All four clients could successfully compute check sums and send the files to the network server machines flawlessly. It was only when changing the server files and testing the integrity option that one issue arose.

The issue was concerning the last date modified aspect of the process. When the files are changed they are time stamped by the server with the time of the last modification. This is an important feature since the S.I.M. application will use those dates in the event the files do not pass the server check. If the files fail the server check, then the last date modified is used to decide which hash file will be used to decide the security status. If the last date modified is wrong, then the integrity of the files is lost. This made it clear the servers had to have the exact same date. Most servers use a centralized time server such as the server from the National Institute of Standards and Technology (NIST). For these test cases, ntp.louisville.edu was used as the time server for both of the servers. Once the date was remedied, the integrity checks went as well as the accuracy checks in testing phase II. Table 5.3 shows the results of mixing the client and server versions. All interactions had passing results. There was also some software flaws discovered during this testing phase.

Table 5.3 – Client Interoperability Test Results

Trial	Java Client and Java Server	Java Client and Binary Server	Binary Client and Java Server	Binary Client and Binary Server
1	Passed	Passed	Passed	Passed
2	Passed	Passed	Passed	Passed
3	Passed	Passed	Passed	Passed
4	Passed	Passed	Passed	Passed
5	Passed	Passed	Passed	Passed

During test phase III there was also a bug in the program. There was a class that would take care of any re-submission of hash files in the event of one or more servers

being unreachable, and it functioned as expected with option number 1. However, when this class was used to re-submit files during an overwrite procedure in option number when one of the servers was down; it was discovered that the resend class would not work properly in the event of this error. This error was not critical because the server would have to go down directly after be asked for the date last modified of the server file. It is not impossible that this could happen, but it is a very low possibility.

Test phase III proved not only useful for the client interoperability, but it also revealed some of the pitfalls that server administrators would have to be aware of as well as some program bugs. Both servers could have approximately the same time, but it is recommended that both servers have the same time server and update simultaneously in order to prevent the wrong actions from happening in the integrity check option. Overall, all testing phases were successful. Problems were encountered throughout all three phases, but there were enough resources available that quickly helped remedy these problems. The final chapter will overview the results with a conclusion with a section suggesting some future research possibilities.

CHAPTER VI

CONCLUSIONS AND FUTURE RESEARCH

This chapter will overview and summarize the testing results of the three phases, and it will also comment on the discoveries made during these testing phases. Also, the author will explain details for future research that the S.I.M. application could create for other areas of research.

6.1 Conclusions

Overall the S.I.M. application functioned well during the time trials, accuracy test, and the client interoperability test. First the time trials will be discussed, then the accuracy test will be discussed, and finally, the client interoperability test will be discussed. The time trial test results show that the client applications are suitable for both Windows systems and Linux systems. However, the Window systems can have several binary files installed that increase the amount of time to compute the hash sums. The performance of the Windows system will be evaluated and some recommendations will be suggested.

The performance of the client side of S.I.M. for the Windows platform was successful. The Java virtual machine and the stand alone executable both functioned properly during the testing phase. Test results show that the binary executable was faster

while processing the hash algorithms. This result becomes important since the speed of computing will become an issue for machines with several binary files. There are a few recommendations that will help with the usability of S.I.M.

The recommendations from the performance testing are as follows. The user should run S.I.M. as a background job while other applications are accessed. Task manager wizard can be used to create a job that the S.I.M. application can run periodically. Last, it is recommended by the author for the Windows client machines to use the binary executable file. This client is chosen for two main reasons. First, the user can easily take the single binary file to any Windows platform machine and run it like any a normal program. The distribution would be effectively achieved by a common network share where the file could reside. The second reason is because when GCJ is released in later versions, more of the overall functionality of the Java libraries will be available. It also means that a graphical user interface would more feasible because the window will be able to run optimally on the native OS. Next, the time trial tests on the Linux platform will be discussed.

The S.I.M. application ran successfully in the Linux environment. Both the Java client and the binary file functioned accordingly. The binary file is not referred to as a stand alone because it is not a type of binary that could run on any Linux machine. As it was shown in testing phase I, there were compatibility issues between GCJ versions. Also, when Linux binaries are created, they are linked to shared library files that reside on that particular system. For this reason a Linux binary could not be moved to another Linux machine because of the chance a library file might not exist or the chance that the

versions might differ. Most of these issues were resolved when the configure file and the Makefile were created in order to make the distribution of the S.I.M. application easier.

The binary file was faster – as was expected – during the time trials in the Linux environment. Considering that the Linux client read four directories as opposed to just two from the Windows client is a significant point. The Linux environment proved to be more robust and efficient – even with the compatibility issues encountered – for the client to operate. Any client type would work for the Linux environment with no problem, but the binary client would be the first choice of the author. Both write the hash files in the same manner, so the clients have the ability to be used interchangeably. Also, the binary version does not exempt the same symptoms as the Windows binary version. If the developers of GCJ updated the library files, it easier for a Linux machine to run a command such as *apt-get upgrade* in order to update the GCJ version and have the latest implementation of the Java library. Then the Makefile could be used to clean up the old configuration and recreate the new binary file with the new available Java library implementations. The overall recommendation is to use the Linux environment; however, the average user is using some version of Windows. For the Windows users, the binary file will be the best choice. Next, the overview and recommendations of phase II – the accuracy test.

The accuracy test was performed with successful results. There were many scenarios that were possible for an integrity breach, and the S.I.M. application detected them all very well. Realistically, not all of the situations could happen in such close time intervals that was tested; but conducting the test in this manner shows that the S.I.M. application is capable of handling more than could realistically handle. An example of a

realistic situation is when a person may compromise one of your servers. An intruder would have to know exactly which other machine is running the other server in order to corrupt both hash files. More than likely, an integrity check would be scheduled before the intruder could find the other server. Once this check occurs, then both servers are replaced with the verified files. By that time, an administrator would see that one of the servers was compromised and take necessary action to protecting the other servers and rebuilding the compromised server. The integrity of the workstation has been kept.

The above example demonstrates events that could happen in any network environment. The S.I.M. was armed with possible outcomes of a scenario in anticipation of an intelligent intruder that could bring the whole network down. But that scenario is highly unlikely. S.I.M. has built-in fault tolerance as well as the ability to be modified to add even more server to provide an even greater fault tolerant resistance to intruders. The accuracy test was a success on both platforms and that demonstrates that the S.I.M. application is suitable for any client. The last phase of the testing will be discussed next.

Phase III was a success and very helpful in discovering design flaws. Overall, the four client versions worked as expected; however, there were some errors that occurred that raise interesting questions. There were two main issues that arose from this testing phase. The first issue was that of server communication. When the testing phase was initiated, only one server could be contacted. At first it was not apparent as to what the problem could be. After some investigation, it was concluded that the server could not accept any transmissions from the clients because of the *shorewall* application. This Linux server was configured slightly different than the other intended server that was to be used. The *shorewall* application is a front end for *iptables* that does port filtering for

the purpose of firewalls and other routing related tasks. After realizing that *shorewall* would not allow the S.I.M. to accept hash files, another Linux server was chosen in its place. Both servers could function properly as the S.I.M. servers. The second issue was that of the time stamp.

When the clients initiated the integrity checks, the decisions made were not what were intended. The error was traced to a portion of the code where the last date modified field of the files was used to decide which option to use in the process. This error showed that the two servers used obviously had different system times. The time stamp is a key role in the S.I.M. application, so if these parameters were incorrect, the S.I.M. application becomes useless. In order to make sure the two servers had the same time, and application called *ntpdate* was installed on one of the servers. One server already synchronized its clock with ntp.louisville.edu, so the second server had to do the same. The *ntpdate* application connected to ntp.louisville.edu and synchronized its system clock. Now that both servers had approximately the same time the testing resumed. All testing phases were successful and proved the S.I.M. application could be used over a network environment. Testing data results can be viewed in Appendix II. The next section will cover some of the future research projects that could be spawned from the S.I.M. application or appended to the S.I.M. application.

6.2 Future Research

The S.I.M. application provides a multitude of directions for future research in the area of computer security, computer forensics, and intelligent systems. First in the area of computer security, the S.I.M. application lays ground work for homogenous applications that improve the area of usability and security. The user interface was designed to extremely simply for any user to operate. With later versions, there will be more choices, but the menu interface should still be simplistic. Even in the implementation of a Graphical Interface, the layout will need to be eye pleasing and be able to be navigated easily. With regard to security, more hashing algorithms can be implemented in order to create more hash files for more redundancy. Keep in mind that by adding more hashing algorithms more processing time would be required to run the client application. Concerning the server side of S.I.M., there are four possible improvements on the security.

There could be more multiple servers to create redundancy in case of server intrusions and/or server failures. Second, the files could be hashed by one of the algorithms, then the output hashed by the same algorithm or another hashing algorithm. This hierarchy could be implemented in a number of ways that would only be known to designer. Even on client level, this custom hierarchy could be used to do two and three round hash sums of the binary files. This also would increase the processing time required to run the S.I.M. application. Security can also be implemented on a file level as well. The third improvement can be on the database file the server contains for each client. It can be stored in a more secure manner than just a flat database file where a

simple text editor can interpret it. Writing the file in binary format and doing bit shifting operations before decrypting the contents can help protect the contents from being changed by outside sources. Lastly, the S.I.M. application can be modified to use the Secure Sockets Layer during communication in order to give an extra layer of security to guarantee that the transmission is not being intercepted by intruders. Next areas concerning computer forensics will be discussed.

The S.I.M. application can be used to help when conducting an analysis on a compromised system. When a system has been compromised, usually an image of the hard driver copied and analyzed in order to help prevent the event from happening again. The S.I.M. application could possibly be modified to connect to a repository of known good hash sums to check against the compromised hard to isolate where a Trojan program could reside. This would also be useful in helping to derive a timeline of when an attacker compromised a client system. Users browse the web for work and leisure time, and this browsing can allow viruses and Trojan programs to be installed without the user's knowledge. It is beneficial to have a type of utility such as S.I.M. in order to ensure that their binary files are safe to execute. This type of application use would be well suited for small business network environments as well as home usage for the security conscience. Lastly, in the area of intelligent systems the S.I.M application can be used in many beneficial ways.

The S.I.M. application can be modified with autonomous agents thereby improving on the monitoring of client binary utilities. Using autonomous agents would greatly improve on the user friendliness of the S.I.M. application as well as its flexibility. Agents could be scheduled to run integrity check jobs periodically in a background

process and alert the user if any programs look like they may have been changed. For a more dynamic approach, the agents could scan the system $PATH, and store the paths in a file to use later in a secure hash option. Also the S.I.M. application could be modified to use an expert system to find integrity violations more effectively and efficiently. The highlighted points previously mentioned are major areas where future research would be greatly beneficial; and there are many more enhancements that could be added to the S.I.M. application improving its usability, functionality, and flexibility.

REFERENCES

[1] Luger, George F. Artificial Intelligence: Structures and Strategies for Complex Problem Solving. United States: Pearson Education, Inc, 2002

[2] McClure, Stuart, Joel Scambray, and George Kurtz. Hacking Exposed. Berkley, California: Osborne/McGraw-Hill, 2001

[3] Caloyannides, Michael A. Computer Forensics and Privacy. Norwood, MA: Artech House, Inc 2001

[4] Alberts, Christopher and Dorofee, Audrey. Managing Information Security Risks: The Octave[sm] Approach. Upper Saddle River, NJ: Pearson Education, Inc., 2003

[5] Kizza, Joseph Migga. Computer Network Security and Cyber Ethics. Jefferson, North Carolina: McFarland & Company, Inc., Publishers, 2002

[6] Ghosh, Sumit. Principles of Secure Network Systems Design. Hoboken, NJ: Springer-Verlag New York, Inc., 2002

[7] Ramachandran, Jay. Designing Security Architecture Solutions. New York, NY: John Wiley & Sons, Inc., 2002

[8] Kruse II, Warren G. and Heiser, Jay G. Computer Forensics: Incident Response Essentials. Indianapolis, IN: Pearson Education, Inc., 2002

[9] Musaji, Yusufali F. Auditing and Security: AS/400, NT, UNIX, Networks, and Disaster Recovery Plans. New York, NY: John Wiley & Sons, Inc., 2001

[10] Ursino, Domenico. Extraction and Exploitation of Intensional Knowledge from Heterogeneous Information Sources. Springer-Verlag Berlin Heidelberg New York, 2002

[11] Chen, Guanorong, and Pham, Trung T. "Some Applications of Fuzzy Logic in Rule – Based Systems." *Experts Systems* Vol. 19 No. 4 (2002):208 – 222

[12] Chen, Guanorong, and Pham, Trung T. "Modeling for an Expert System and a Parameter Validation." *Experts Systems* Vol. 19 No. 5 (2002):285 – 294

[13] Caloyannides, Michael A. Desktop Witness: the do's and don'ts of personal computer security. West Sussex, England: John Wiley & Sons Ltd, 2002

[14] Sandhu, Ravi. "Good-Enough Security: Toward a Pragmatic Business-Driven Discipline." IEEE Internet Computing Volume 7 Issue 1 (2003): 66-68

[15] Thompson, Herbert H. "Why Security Testing is Hard." IEEE Security & Privacy Volume 1 Issue 4 (2003): 83-86

[16] Schneier, Bruce. "The Speed of Security." IEEE Security & Privacy Volume 1 Issue 4 (2003): 96

[17] Bishop, Matt. "What is Computer Security?." IEEE Security & Privacy Volume 1 Issue 1 (2003): 67-69

[18] Arce, Ivan. "The Weakest Link Revisted." IEEE Security & Privacy Volume 1 Issue 2 (2003): 72-76

[19] Caloyannides, Michael. "Privacy vs. Information Technology." IEEE Security & Privacy Volume 1 Issue 1 (2003): 100-103

[20] Whittaker, James. "Why Secure Applications are Difficult to Write." IEEE Security & Privacy Volume 1 Issue 2 (2003): 81-83

[21] Smith, S.W. "Humans in the Loop." IEEE Security & Privacy Volume 1 Issue 3 (2003): 75-79

[22] Cowan, Crispin. "Software Security for Open-Source Systems." IEEE Security & Privacy Volume 1 Issue 1 (2003): 38-45

[23] Gong, Li. "Why Cross-Platform Security." IEEE Internet Computing Volume 7 Issue 3 (2003): 95-96

[24] Yasinsac, Alec, Erbacher, Robert F., Marks, Donald G., Pollit, Mark M., and Sommer, Peter M. "Computer Forensics Education." IEEE Security & Privacy Volume 1 Issue 3 (2003): 15-23

[25] Arce, Ivan. "More Bang for the Bug: An Account of 2003's Attack Trends." IEEE Internet Computing Volume 2 Issue 1 (2004): 66-68

[26] http://www.securityfocus.com/tools/55 Accessed February 2004

[27] http://userpages.umbc.edu/~mabzug1/cs/md5/md5.html Accessed March 2004

[28] http://www.bytefusion.com/products/ens/secexmd5+/index_secexmd5.htm?ussecurehashalgorithm1(sh.htm Accessed March 2004

[29] http://stake.com/research/tools/forensic/ Accessed December 2003

[30] http://www.sleuthkit.org/ Accessed February 2004

APPENDIX I. Test Phase I Time Trials Data

Time in Milliseconds Format

Test Runs	Windows with Java Virtual Machine	Debian Linux with Java Virtual Machine	Windows wihtout Java Virtual Machine	Debian Linux without Java Virtual Machine
1	285871	166120	266564	208432
2	250770	154417	216451	207559
3	250209	156357	214778	222299
4	252663	155404	216241	121199
5	251602	156871	215009	120506
6	252673	155415	214719	120792
7	250520	154867	215570	120536
8	251481	155590	215050	121292
9	250501	155129	216411	120459
10	251883	155162	216392	121116
11	252453	156360	215330	120521
12	250730	255555	216021	121475
13	250570	159785	221899	120817
14	254656	159750	216512	121116
15	251211	160010	217543	120840
16	245844	156042	216251	120866
17	251802	159858	215911	120603
18	250200	159516	215290	121032
19	250060	159358	215680	120903
20	257600	159930	216241	121392
21	249739	156388	217633	120545
22	251762	156540	218034	121108
23	251782	156641	217122	120559
24	249880	156133	217012	121126
25	251101	156670	216401	120380
26	251081	156679	219235	121064
27	250270	156554	216811	120475
28	255016	156184	215690	120962
29	250770	156753	215420	121068
30	250871	156339	215791	121644
31	252893	156204	215279	120364
32	254917	157034	215711	120873
33	252173	156323	217302	120534
34	253794	155422	217153	121190
35	252453	156004	217072	120551
36	251102	155926	215700	121096
37	249970	155508	215810	120520
38	250710	155555	215480	120993

39	249699	155786	216281	120769
40	251812	155284	215740	120952
41	251873	156188	217183	120495
42	252002	155712	217242	120974
43	254836	155641	216922	120338
44	253655	156315	216301	121398
45	252823	156111	218023	120325
46	250871	154713	215429	120954
47	250260	155905	216852	120514
48	251211	155878	216472	120908
49	251301	155246	229280	120840
50	252674	155184	214168	121331

APPENDIX II. Test Phase II Accuuracy Data

		Decision Test For Passing Server Verification	
Trial	**Detected extra binary files**	**Detected changed binary files**	**Detected both extra files and changed files**
1	Pass	Pass	Pass
2	Pass	Pass	Pass
3	Pass	Pass	Pass
4	Pass	Pass	Pass
5	Pass	Pass	Pass
6	Pass	Pass	Pass
7	Pass	Pass	Pass
8	Pass	Pass	Pass
9	Pass	Pass	Pass
10	Pass	Pass	Pass

		MD5 Decision Test Results	
Trial	**Detected extra binary files**	**Detected changed binary files**	**Detected both extra files and changed files**
1	Pass	Pass	Pass
2	Pass	Pass	Pass
3	Pass	Pass	Pass
4	Pass	Pass	Pass
5	Pass	Pass	Pass
6	Pass	Pass	Pass
7	Pass	Pass	Pass
8	Pass	Pass	Pass
9	Pass	Pass	Pass
10	Pass	Pass	Pass

		SHA1 Decision Test Results	
Trial	**Detected extra binary files**	**Detected changed binary files**	**Detected both extra files and changed files**
1	Pass	Pass	Pass
2	Pass	Pass	Pass
3	Pass	Pass	Pass
4	Pass	Pass	Pass
5	Pass	Pass	Pass
6	Pass	Pass	Pass
7	Pass	Pass	Pass
8	Pass	Pass	Pass
9	Pass	Pass	Pass
10	Pass	Pass	Pass

APPENDIX III. Selected Source Code

```
package sim.agents;
import java.io.*;
import sim.interfaces.*;

/**
 * <p>Title: Software Integrity Management</p>
 * <p>Description: Security Software</p>
 * <p>Copyright: Copyright (c) 2004</p>
 * <p>University: University of Louisville, J.B. Speed School of
Engineering</p>
 * @author Joseph H. Brown
 * @version 1.0 Revision 1
 */

public class integrityCheck {
  public integrityCheck() { }

  private String md5NoMatch = "";
  private String sha1NoMatch = "";
  private String md5Extras = "";
  private String sha1Extras = "";
  private int md5Yes = 0, md5No = 0, sha1Yes = 0, sha1No = 0;
  private int md5BinaryYes = 0, md5BinaryNo = 0, sha1BinaryYes = 0,
sha1BinaryNo = 0;
  private functions f1 = new functions();


  /**
   * Responsible for performing the integrity check.  This method will
call
   * others suppport methods to help in determining the validity of the
system.
   *
   * @param  serverMD5
   *             The file downloaded from the md5 server.
   *         md5Client
   *             The client file of the stored md5 results.
   *         serverSHA1
   *             The file downloaded from the sha1 server.
   *         sha1Client
   *             The client file of the stored sha1 results.
   *         date
   *             The date of the md5 file used for both file's last
date modified.
   * @return Nothing
   * @throws IOException
   */
  public void performIntegrityCheck(String serverMD5, String md5Client,
String serverSHA1, String sha1Client, long date) throws IOException {
    int status = 0, status2 = 0, index = 0, index2 = 0;
    long date1, date2;
    String[] md5Array, sha1Array, md5ExtraArray, sha1ExtraArray;
```

```
    System.out.println("Verifying server files.....");
    status = this.checkServerFiles(serverMD5, serverSHA1);
    if (status == 0) {
      //file not the same size or there are binaries that do not match
      System.out.println("The file sizes are not the same length or
have different binaries, using the last modified date.");
      transport t1 = new transport();
      date1 = t1.getLastModifiedMD5();
      //if date == -1 need to by pass operation - not likely though
      //System.out.println("date1 = " + date1);
      date2 = t1.getLastModifiedSHA1();
      //if date == -1 need to by pass operation - not likely though
      //System.out.println("date2 = " + date2);

      /************* Use the MD5 file for the integrity check
*************/
      if (date1 < date2) {
        System.out.println("Using the date from the MD5 server");
        System.out.println("Initiating MD5 secure check.");
        status = this.checkIntegrity(md5Client, serverMD5,0);
        if (status == 3) {
          //Successful match
          resend r1 = new resend();
          int close = -1, tryMe = -1;
          System.out.println("The md5 check has verified all binary
files. "
                             + "Both the md5 and sha1 servers will have
the "
                             + "most up to date secure files.");
          System.out.println("Sending to the md5 server.....");
          t1 = null;
          t1 = new transport(date, md5Client, sha1Client);
          t1.overrideMD5Server(md5Client);
          System.out.println("Sending to the sha1 server.....");
          t1.overrideSHA1Server(sha1Client);

          /*******************************************************/
          //create resend here!! - later version must have a class to
test for this.
          //send the md5
          /*tryMe = t1.overrideMD5Server(md5Client);
          //status = t1.sendToMD5Server(this.md5Address, this.md5Port);
          if (tryMe == 1)
            System.out.println("MD5 File sent successfully");
          else if (tryMe == -1) {
            //System.out.println("Chosen not to overwrite md5 ");
            //delmd5 = true;
          }
          else {
            //do the create resend
            r1.createResend(md5Client, date);
            r1.setFlag();
            close = 1;
            System.out.println("Saving the md5 job for later.");
          }
          //send the sha1
```

```
            tryMe = t1.overrideSHA1Server(sha1Client);
            //status = t1.sendToSHA1Server(this.sha1Address,
this.sha1Port);
            if (tryMe == 1) {
              System.out.println("SHA1 File sent successfully");
            }
            else if (tryMe == -1) {
              //System.out.println("Not overwrite.");
              //delsha1 = true;
            }
            else {
              //do the resend send sha1
              r1.addToResend(sha1Client, date);
              close = 1;
              System.out.println("Saving sha1 job for later.");
            }
            if (close == 1)
              r1.completeResend();
            r1 = null;
            t1 = null;
            /*******************************************************/

          }
          else if (status == 2) {
            //extra binaries and no matches
            System.out.println("There are extra binaries detected on your
system since "
                               + "the last secure hash.  Please make
sure that no program(s) "
                               + "have been installed without your
knowledge.  These binaries "
                               + "are usually trojan programs or spy ware
installed without your knowledge.");
            //List the extra binaries.
            String[] temp = f1.split(this.md5Extras, "!");
            System.out.println("The following is a list of the extra
binaries detected");
            for (int i = 0; i < temp.length; i++)
              System.out.println(temp[i].substring(temp[i].indexOf(" "),
temp[i].length()));
            temp = null;
            temp = f1.split(this.md5NoMatch, "!");
            System.out.println("The system has also detected that some
binary programs "
                               + "may have been changed.  Sometimes extra
programs installed "
                               + "can change other binary programs, or
the worst situation "
                               + "is that there has been an unauthorized
breech in the system programs "
                               + "Please verify the changes were made by
you and update the files, or perform "
                               + "an analysis for intrusion detection");
            System.out.println("The following is a list of the changed
binaries detected");
            for (int i = 0; i < temp.length; i++)
              System.out.println(temp[i]);
```

```
          temp = null;
        }
        else if (status == 1) {
          //extra binaries
          System.out.println("There are extra binaries detected on your
system since "
                             + "the last secure hash.  Please make
sure that no program(s) "
                             + "have been installed without your
knowledge.  These binaries "
                             + "are usually trojan programs or spy ware
installed without your knowledge. ");
          System.out.println("The main suggestion is to double check
and make sure your secure hash "
                             + "file was up to date with any programs
you may have installed.");
          //List the extra binaries.
          String[] temp = f1.split(this.md5Extras, "!");
          System.out.println("The following is a list of the extra
binaries detected");
          for (int i = 0; i < temp.length; i++)
            System.out.println(temp[i].substring(temp[i].indexOf(" "),
temp[i].length()));
          temp = null;
        }
        else if (status == 0) {
          //no mathces
          System.out.println("The system has detected that some binary
programs "
                             + "may have been changed.  Sometimes extra
programs installed "
                             + "can change other binary programs, or
the worst situation "
                             + "is that there has been an unauthorized
breech in the system programs "
                             + "Please verify the changes were made by
you and update the files, or perform "
                             + "an analysis for intrusion detection");
          String[] temp = f1.split(this.md5NoMatch, "!");
          System.out.println("The following is a list of the altered
binaries detected");
          for (int i = 0; i < temp.length; i++)
            System.out.println(temp[i]);
          temp = null;
        }
      }
      /************** Use the SHA1 file for the integrity check
**************/
      else if (date1 > date2) {
        System.out.println("Using the date from the SHA1 server");
        System.out.println("Initiating SHA-1 secure check.");
        status2 = this.checkIntegrity(sha1Client, serverSHA1,1);
        System.out.println(status2);
        if (status2 == 3) {
          int tryMe = -1, close = -1;
          resend r1 = new resend();
```

```
            System.out.println("The sha1 check has verified all binary
files.\n"
                               + "Both the md5 and sha1 servers will have
the "
                               + "most up to date secure files.\n\n");
            System.out.println("Sending to the md5 server.....\n");
            t1 = null;
            t1 = new transport(date, md5Client, sha1Client);
            t1.overrideMD5Server(md5Client);
            System.out.println("\nSending to the sha1 server.....\n");
            t1.overrideSHA1Server(sha1Client);

            /*******************************************************/
            //create resend here!! - later version must have a class to
test for this.
            //send the md5
            /*tryMe = t1.overrideMD5Server(md5Client);
            //status = t1.sendToMD5Server(this.md5Address, this.md5Port);
            if (tryMe == 1)
              System.out.println("MD5 File sent successfully");
            else if (tryMe == -1) {
              //System.out.println("Chosen not to overwrite md5 ");
              //delmd5 = true;
            }
            else {
              //do the create resend
              r1.createResend(md5Client, date);
              r1.setFlag();
              close = 1;
              System.out.println("Saving the md5 job for later.");
            }
            //send the sha1
            tryMe = t1.overrideSHA1Server(sha1Client);
            //status = t1.sendToSHA1Server(this.sha1Address,
this.sha1Port);
            if (tryMe == 1) {
              System.out.println("SHA1 File sent successfully");
            }
            else if (tryMe == -1) {
              //System.out.println("Not overwirte.");
              //delsha1 = true;
            }
            else {
              //do the resend send sha1
              r1.addToResend(sha1Client, date);
              close = 1;
              System.out.println("Saving sha1 job for later.");
            }
            if (close == 1)
              r1.completeResend();
            r1 = null;
            t1 = null;
            /*******************************************************/

          }
          else if (status2 == 2) {
            //extra binaries and no matches
```

```
          System.out.println("There are extra binaries detected on your
system since "
                                + "the last secure hash.  Please make
sure that no program(s) "
                                + "have been installed without your
knowledge.  These binaries "
                                + "are usually trojan programs or spy ware
installed without your knowledge.");
          //List the extra binaries.
          String[] temp = f1.split(this.sha1Extras, "!");
          System.out.println("The following is a list of the extra
binaries detected");
          for (int i = 0; i < temp.length; i++)
            System.out.println(temp[i].substring(temp[i].indexOf(" "),
temp[i].length()));
          temp = null;
          System.out.println("The system has detected that some binary
programs "
                                + "may have been changed.  Sometimes extra
programs installed "
                                + "can change other binary programs, or
the worst situation "
                                + "is that there has been an unauthorized
breech in the system programs "
                                + "Please verify the changes were made by
you and update the files, or perform "
                                + "an analysis for intrusion detection");
          temp = f1.split(this.sha1NoMatch, "!");
          System.out.println("The following is a list of the altered
binaries detected");
          for (int i = 0; i < temp.length; i++)
            System.out.println(temp[i]);
          temp = null;
        }
        else if (status2 == 1) {
          System.out.println("There are extra binaries detected on your
system since "
                                + "the last secure hash.  Please make
sure that no program(s) "
                                + "have been installed without your
knowledge.  These binaries "
                                + "are usually trojan programs or spy ware
installed without your knowledge.");
          System.out.println("The main suggestion is to double check
and make sure your secure hash "
                                + "file was up to date with any programs
you may have installed.");
          //List the extra binaries.
          String[] temp = f1.split(this.sha1Extras, "!");
          System.out.println("The following is a list of the extra
binaries detected");
          for (int i = 0; i < temp.length; i++)
            System.out.println(temp[i].substring(temp[i].indexOf(" "),
temp[i].length()));
          temp = null;
        }
        else if (status2 == 0) {
```

```
          System.out.println("In the 0, all good");
          //no mathces
          System.out.println("The system has detected that some binary
programs "
                             + "may have been changed.  Sometimes extra
programs installed "
                             + "can change other binary programs, or
the worst situation  "
                             + "is that there has been an unauthorized
breech in the system programs  "
                             + "Please verify the changes were made by
you and update the files, or perform  "
                             + "an analysis for intrusion detection");
          String[] temp = f1.split(this.sha1NoMatch, "!");
          System.out.println("The following is a list of the altered
binaries detected");
          for (int i = 0; i < temp.length; i++)
            System.out.println(temp[i]);
          temp = null;
        }
      }
      else if (date1 == date2) {
        //something went reel wrong!!
        System.out.println("The System has performed an error -
integrity Class.  System Failure.");
        System.exit(1);
      }
      else {
        //do nothing
        System.out.println("The application has exhibited bad system
behavior, exiting!");
        System.exit(1);
      }
    }

    /************ Start scenario where files are of same length
**********/
    else if (status == 1) {
      //it is all good
      System.out.println("Initiating MD5 secure check.");
      status = this.checkIntegrity(md5Client, serverMD5,0);
      System.out.println("Initiating SHA-1 secure check.");
      status2 = this.checkIntegrity(sha1Client, serverSHA1,1);
      //System.out.println(status + " and " + status2); //Debugging
Statement
      //Both files matched successfully
      if ( (status == 3) && (status2 == 3) ) {
        System.out.println("Both files returned and verified
successfully.  "
                           + "Your system binary files are good.");
      }
      //Found good MD5 hash sums and bad SHA1 sums
      if ( (status == 3) && (status2 == 0) ) {
        transport t1;
        System.out.println("The md5 check has verified all binary
files, but the"
```

```
                            + "sha1 check has failed.  The md5 file will
be used for replacement.\n"
                            + "Both the md5 and sha1 servers will have
the "
                            + "most up to date secure files.\n\n");
        System.out.println("Sending to the md5 server.....\n");
        t1 = new transport(date, md5Client, sha1Client);
        t1.overrideMD5Server(md5Client);
        System.out.println("\nSending to the sha1 server.....\n");
        t1.overrideSHA1Server(sha1Client);
        t1 = null;
        //create resend here!! - later version must have a class to
test for this.
      }
      //Found good SHA1 hash sums and bad MD5 sums
      if ( (status == 0) && (status2 == 3) ) {
        transport t1;
        System.out.println("The sha1 check has verified all binary
files, but the"
                            + "md5 check has failed.  The sha1 file will
be used for replacement.\n"
                            + "Both the md5 and sha1 servers will have
the "
                            + "most up to date secure files.\n\n");
        System.out.println("Sending to the md5 server.....\n");
        t1 = new transport(date, md5Client, sha1Client);
        t1.overrideMD5Server(md5Client);
        System.out.println("\nSending to the sha1 server.....\n");
        t1.overrideSHA1Server(sha1Client);
        t1 = null;
        //create resend here!! - later version must have a class to
test for this.
      }
      //extra MD5 binaries and no md5 matches && extra SHA1 binaries
and no SHA1 matches
      if ( (status == 2) && (status2 == 2) ) {
        System.out.println("There are extra binaries detected on your
system since "
                            + "the last secure hash.  \nPlease make sure
that no program(s) "
                            + "have been installed without your
knowledge.  \nThese binaries "
                            + "are usually trojan programs or spy ware
installed without your knowledge.");
        System.out.println("The main suggestion is to double check and
make sure your secure hash "
                            + "file was up to date with any programs you
may have installed.");
        //List the extra binaries.
        String[] temp = f1.split(this.sha1Extras, "!");
        System.out.println("The following is a list of the extra
binaries detected");
        for (int i = 0; i < temp.length; i++)
          System.out.println(temp[i].substring(temp[i].indexOf(" "),
temp[i].length()));
        temp = null;
        //no matches below here
```

```
        System.out.println("The system has also detected that some
binary programs "
                           + "may have been changed.  Sometimes extra
programs installed "
                           + "can change other binary programs, or
the worst situation  "
                           + "is that there has been an unauthorized
breech in the system programs");
        //List the compromised binaries.
        temp = f1.split(this.sha1NoMatch, "!");
        System.out.println("  The following is a list of the no match
binaries detected according to the sha1 hash.");
        for (int i = 0; i < temp.length; i++)
          System.out.println(temp[i]);
        temp = null;
        System.out.println(" ");
        temp = f1.split(this.md5NoMatch, "!");
        System.out.println("The following is a list of the no match
binaries detected according to the md5 hash.");
        for (int i = 0; i < temp.length; i++)
          System.out.println(temp[i]);
        temp = null;
      }
      //extra MD5 binaries and no md5 matches && extra SHA1 binaries
      if ( (status == 2) && (status2 == 1) ) {
        System.out.println("There are extra binaries detected on your
system since "
                           + "the last secure hash.  Please make sure
that no program(s) "
                           + "have been installed without your
knowledge.  These binaries "
                           + "are usually trojan programs or spy ware
installed without your knowledge.");
        System.out.println("The main suggestion is to double check and
make sure your secure hash "
                           + "file was up to date with any programs you
may have installed.");
        //List the extra binaries.
        String[] temp = f1.split(this.sha1Extras, "!");
        System.out.println("The following is a list of the extra
binaries detected");
        for (int i = 0; i < temp.length; i++)
          System.out.println(temp[i].substring(temp[i].indexOf(" "),
temp[i].length()));
        temp = null;
        //no matches below here
        System.out.println("The system has also detected that some
binary programs "
                           + "may have been changed.  Sometimes extra
programs installed "
                           + "can change other binary programs, or
the worst situation  "
                           + "is that there has been an unauthorized
breech in the system programs");
        //List the compromised binaries.
        temp = f1.split(this.md5NoMatch, "!");
```

```
        System.out.println("  The following is a list of the no match
binaries detected according to the md5 hash.");
        for (int i = 0; i < temp.length; i++)
          System.out.println(temp[i]);
        temp = null;
      }
      //extra MD5 binaries && extra SHA1 binaries and no SHA1 matches
      if ( (status == 1) && (status2 == 2) ) {
        System.out.println("There are extra binaries detected on your
system since "
                           + "the last secure hash.   Please make sure
that no program(s) "
                           + "have been installed without your
knowledge.   These binaries "
                           + "are usually trojan programs or spy ware
installed without your knowledge.");
        System.out.println("The main suggestion is to double check and
make sure your secure hash "
                           + "file was up to date with any programs you
may have installed.");
        //List the extra binaries.
        String[] temp = f1.split(this.sha1Extras, "!");
        System.out.println("The following is a list of the extra
binaries detected");
        for (int i = 0; i < temp.length; i++)
          System.out.println(temp[i].substring(temp[i].indexOf(" "),
temp[i].length()));
        temp = null;
        //no matches below here
        System.out.println("The system has also detected that some
binary programs "
                           + "may have been changed.  Sometimes extra
programs installed "
                           + "can change other binary programs, or
the worst situation "
                           + "is that there has been an unauthorized
breech in the system programs");
        //List the compromised binaries.
              temp = f1.split(this.sha1NoMatch, "!");
        System.out.println("  The following is a list of the no match
binaries detected according to the sha1 hash.");
        for (int i = 0; i < temp.length; i++)
          System.out.println(temp[i]);
        temp = null;
      }
      //extra MD5 binaries && extra SHA1 binaries
      if ( (status == 1) && (status2 == 1) ) {
        System.out.println("There are extra binaries detected on your
system since "
                           + "the last secure hash.   Please make sure
that no program(s) "
                           + "have been installed without your
knowledge.   These binaries "
                           + "are usually trojan programs or spy ware
installed without your knowledge.");
        System.out.println("The main suggestion is to double check and
make sure your secure hash "
```

```
                            + "file was up to date with any programs you
may have installed.");
        //List the extra binaries.
        String[] temp = f1.split(this.sha1Extras, "!");
        System.out.println("The following is a list of the extra
binaries detected");
        for (int i = 0; i < temp.length; i++)
          System.out.println(temp[i].substring(temp[i].indexOf(" "),
temp[i].length()));
        temp = null;
      }
      //MD5 binaries with no match && SHA1 binaries with no match
      if ( (status == 0) && (status2 == 0) ) {
        System.out.println("MD5 binaries with no match and || SHA1
binaries with no match");
        //no matches
        System.out.println("The system has detected that some binary
programs "
                           + "may have been changed. Sometimes extra
programs installed "
                           + "can change other binary programs, or
the worst situation "
                           + "is that there has been an unauthorized
breech in the system programs");
        //List the compromised binaries.
        String[] temp = f1.split(this.sha1NoMatch, "!");
        System.out.println("The following is a list of the no match
binaries detected according to the sha1 hash.");
        for (int i = 0; i < temp.length; i++)
          System.out.println(temp[i]);
        temp = null;
        System.out.println("");
        temp = f1.split(this.md5NoMatch, "!");
        System.out.println("The following is a list of the no match
binaries detected according to the md5 hash.");
        for (int i = 0; i < temp.length; i++)
          System.out.println(temp[i]);
        temp = null;
      }
    } //End the else if stastus == 1
    else {
      System.out.println("Error in the integrity class - code 148.
System will exit now");
      System.exit(1);
    }
  }


  /**
   * Checks the server files to make sure they are the same length and
have the
   * same binary files.
   *
   * @param  serverMD5
   *             The file from the md5 server.
   *         serverSHA1
   *             The file from the sha1 server.
```

```
    * @return int
    *             An integer that returns the result of the server files
verification.
    * @throws IOException
    *             File not found exception.
    */
  private int checkServerFiles(String serverMD5, String serverSHA1) {
    try {
      BufferedReader b = new BufferedReader(new FileReader(serverMD5));
      BufferedReader b2 = new BufferedReader(new
FileReader(serverSHA1));
      String temp = "", one = "", two = "";
      String[] arrayOne, arrayTwo;
      int count1 = 0, count2 = 0, flag = 0, index = 0, index2 = 0;

      while (! (temp = b.readLine()).equals("##**##&&?")) {
        //System.out.println(one);
        one = one + temp + "!";
        count1++;
      }
      b.close();
      while (! (temp = b2.readLine()).equals("##**##&&?")) {
        //System.out.println(two);  //Debugging statement
        two = two + temp + "!";
        count2++;
      }
      b2.close();
      //System.out.println(count1 + " and " + count2);  //Debugging
statement
      if (count1 != count2) {
        //Return the file lengths do not match.
        return (0);
      }
      else {
        arrayOne = f1.split(one, "!");
        arrayTwo = f1.split(two, "!");
        for (int i = 0; i < arrayOne.length; i++) {
          //Compare the binary files only.
          index = arrayOne[i].indexOf(" ");
          for (int k = 0; k < arrayTwo.length; k++) {
            index2 = arrayTwo[k].indexOf(" ");
            if ( arrayOne[i].substring(index+1,
arrayOne[i].length()).equals(arrayTwo[k].substring(index2+1,
arrayTwo[k].length())) ) {
              flag = 1;
              break;
            }
          }  //End for loop with k.
          if (flag != 1) {
            flag = 0;
            //Return the binaries do not match
            return (0);
          }
          flag = 0;
        } //End the main for loop.
        //else return status good
        return (1);
```

```
      }
    }
    catch (FileNotFoundException f1) {
      System.out.println("There has been a file reading error, error
code: server - integrity check");
      return (-1);
    }
    catch (IOException e) {
      System.out.println("There has been an IO error in the server
verification method - inegrity check.");
      return (-1);
    }
  }


  /**
   * Checks the integrity of the local file compared to the server
file.  The
   * hash type is indicated by the value of the integer passed.
   *
   * @param  localFile
   *             The file name of the local saved results.
   *         serverFile
   *             The file name of the server saved results.
   * @return int
   *             An integer that returns the result of the file
verification.
   * @throws IOException
   *             File could not be found.
   */
  public int checkIntegrity(String localFile, String serverFile, int
indicator) throws IOException {
    try {
      BufferedReader b = new BufferedReader(new FileReader(localFile));
      BufferedReader b2 = new BufferedReader(new
FileReader(serverFile));
      String one = "", two = "", temp = "", warning = "";
      String[] oneArray, twoArray;
      int yesBinary = 0, yesSum = 0, noBinary = 0, noSum = 0;
      int flag = 0, index = 0, index2 = 0;
      //System.out.println("Reading the md5 files for comparison");
//Debugging statement
      while (! (temp = b.readLine()).equals("##**##&&?"))
        one = one + temp + "!";
      b.close();
      while (! (temp = b2.readLine()).equals("##**##&&?"))
        two = two + temp + "!";
      b2.close();
      oneArray = f1.split(one, "!");  //Function written to accomadate
GCJ
      twoArray = f1.split(two, "!");  //Function written to accomadate
GCJ
      //oneArray = one.split("!");  //Built-in String function for
regular Java
      //twoArray = two.split("!");  //Built-in String function for
regular Java
      temp = "";
```

```
      System.out.println("Running Secure Check");
      for (int i = 0; i < oneArray.length; i++) {
        //Compare the binary files only.
        index = oneArray[i].indexOf(" ");
        for (int k = 0; k < twoArray.length; k++) {
          index2 = twoArray[k].indexOf(" ");
          if ( oneArray[i].substring(index+1,
oneArray[i].length()).equals(twoArray[k].substring(index2+1,
twoArray[k].length())) ) {
            if (indicator == 0)
              this.md5BinaryYes++;
            if (indicator == 1)
              this.sha1BinaryYes++;
            //yesBinary++;  //Debugging statement
            flag = 1;
            //Next compare the sums
            if ( oneArray[i].substring(0, index-
1).equals(twoArray[k].substring(0, index2-1)) ) {
              if (indicator == 0)
                this.md5Yes++;
              if (indicator == 1)
                this.sha1Yes++;
              //yesSum++;  //Debugging statement
            }
            else {
              warning = warning + oneArray[i].substring(index+1,
oneArray[i].length()) + "!";
              if (indicator == 0)
                this.md5No++;
              if (indicator == 1)
                this.sha1No++;
              noSum++;
            }
            break;
          }
        }  //End for loop with k.
        if (flag != 1) {
          if (indicator == 0)
            this.md5No++;
          if (indicator == 1)
            this.sha1No++;
          noBinary++;
          temp = temp + oneArray[i] + "!";
          flag = 0;
        }
        flag = 0;
      } //End the main for loop.
      if ( (noSum > 0) && (temp.equals("")) ) {
        //no matches but all binaries found
        if (indicator == 0)
          this.md5NoMatch = warning;
        if (indicator == 1)
          this.sha1NoMatch = warning;
        return (0);
      }
      if ( (!(temp.equals(""))) && (noSum == 0) ) {
        //Found Extra Binaries
```

```
        if (indicator == 0)
          this.md5Extras = temp;
        if (indicator == 1)
          this.sha1Extras = temp;
        return (1);
        //write to file insecure.log - extra feature for future version
      }
      if ( (noSum > 0) && (!(temp.equals(""))) ) {
        //found no matches and extra binaries
        if (indicator == 0) {
          this.md5Extras = temp;
          this.md5NoMatch = warning;
        }
        if (indicator == 1) {
          this.sha1Extras = temp;
          this.sha1NoMatch = warning;
        }
        return (2);
      }
      if ( (noSum == 0) && (temp.equals("")) ) {
        System.out.println("The files have been verified
successfully.");
        return (3);
      }
      return (-1);
    }
    catch (FileNotFoundException e) {
      System.out.println("File is not found, internal error 4568." +
                         "System will now quit.");
      return (-1);
    }
  }
}

package sim.agents;
import java.io.*;
import java.util.*;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import sim.interfaces.*;

/**
 * <p>Title: Software Integrity Management</p>
 * <p>Description: Security Software</p>
 * <p>Copyright: Copyright (c) 2004</p>
 * <p>University: University of Louisville, J.B. Speed School of
Engineering</p>
 * @author Joseph H. Brown
 * @version 1.0 Revision 1
 */

public class secureHash {
  public secureHash() { }

  private Vector results = new Vector();
  private String md5File = "";
```

```
  private String sha1File = "";
  private long mainDate = -1;
  private functions f1 = new functions();


  /**
   * Decides which version of the OS is found and calls the necessary
method.
   *
   * @param  os
   *             The string value of the OS name.
   * @return String
   *             This string indicates weather or not the OS can be
used with S.I.M.
   * @throws IOException
   */
  public String hash(String os) throws IOException {
    if (os.indexOf("Windows") > -1) {
      this.results = this.beginWindowsHash();
      this.saveResults("windowsmd5", "windowssha1");
    }
    else if (os.indexOf("CYGWIN") > -1) {
      this.results = this.beginWindowsHash();
      this.saveResults("windowsmd5", "windowssha1");
    }
    else if (os.indexOf("Linux") > -1) {
      this.results = this.beginLinuxHash();
      this.saveResults("linuxmd5", "linuxsha1");
    }
    else
      return ("os_error");
    return ("complete");
  }


  /**
   * Sets the returning Vector with paths of the windows executables
and their
   * md5 hashed values in Hex format and their sha1 hashed values in
Hex format.
   *
   * @return theFiles2
   *                This vector will contain all the executables with
their
   *                full paths and md5 hashed value in Hex.
   * @throws NoSuchAlgorithmException
   *                Thrown when the MessageDigest cannot find the
desired hashing
   *                algorithm to compute the hash function.
   * @throws IOException
   *                Thrown when File IO exceptions occurs.
   */
  private Vector beginWindowsHash() throws IOException
  {
    String digest = "", digest2 = "", temp = "", in = "";
    Vector theFiles = new Vector();
    Vector theFiles2 = new Vector();
```

```
    byte[] hash, hash2;
    String[] dirList;
    generateList g1 = new generateList();

    //Check for Custom Directory File that will be created from the
menu
    try {
      BufferedReader custom = new BufferedReader(new
FileReader("dircust"));
      while ((temp = custom.readLine()) != null)
        in = in + temp + "!";
      custom.close();
    }
    catch (FileNotFoundException ferror) {
      System.out.println("There are no custom directories found, using
defaults.");
    }
    try {
      MessageDigest md5 = MessageDigest.getInstance("MD5");
      MessageDigest sha1 = MessageDigest.getInstance("SHA-1");
      int i = 0, status;
      File test = new File("c:\windows");
      if (test.exists())
        theFiles = g1.readWindows("c:\windows");
      else {
        theFiles = g1.readWindows("c:\winnt");
        //System.out.println("It is windows NT architecture");
//Debugging statement
        theFiles2 = g1.readWindows("c:\program files");
        theFiles.addAll(theFiles2);
        theFiles2.removeAllElements();
      }
      //Add the custom directories to the hash file
      if ( (!in.equals("")) && (!in.equals(" ")) ) {
        dirList = f1.split(in, "!");  //Function written to accomadate
GCJ
        //dirList = in.split("!");  //Built-in String function for
regular Java
        for (int k = 0; k < dirList.length; k++) {
          theFiles2 = g1.readWindows(dirList[k]);
          theFiles.addAll(theFiles2);
          theFiles2.removeAllElements();
        }
        theFiles2.removeAllElements();
      }
      System.out.print("Computing the hashes");
      status = (theFiles.size())/16;
      //System.out.println(status);  //Debugging statement
      while (i < theFiles.size()) {
        FileInputStream fReader = new
FileInputStream(theFiles.elementAt(i).toString());
        int ch;
        while ( (ch = fReader.read()) != -1) {
          md5.update( (byte) ch);
          sha1.update( (byte) ch);
        }
        hash = md5.digest();
```

```
        hash2 = sha1.digest();
        //Convert to Hex Value.
        for (int j = 0; j < hash.length; j++) {
          int v = hash[j] & 0xFF;
          if (v < 16)
            digest += "0";
          digest += Integer.toString(v, 16).toUpperCase();
        }
        //SHA-1 computation
        for (int j = 0; j < hash2.length; j++) {
          int v = hash2[j] & 0xFF;
          if (v < 16)
            digest2 += "0";
          digest2 += Integer.toString(v, 16).toUpperCase();
        }
        if ( (i % status) == 0 )
          System.out.print(".");
        theFiles2.add(i, digest + " " +
theFiles.elementAt(i).toString() + " sha1 " + digest2 + " " +
theFiles.elementAt(i).toString());
        //System.out.println(digest + " " +
theFiles.elementAt(i).toString() + " sha1 " + digest2 + " " +
theFiles.elementAt(i).toString());  //Debugging statement
        digest = "";
        digest2 = "";
        i++;
      }
      System.out.print("complete ");
      return(theFiles2);
    }  //End try
    catch (NoSuchAlgorithmException e) {   }
    return(theFiles2);
  }


  /**
   * Sets the returning Vector with paths of the linux executables and
their
   * md5 hashed values in Hex format and their sha1 hashed values in
Hex format.
   *
   * @return theFiles2
   *                This vector will contain all the executables with
their
   *                full paths and md5 hashed value in Hex.
   * @throws NoSuchAlgorithmException
   *                Thrown when the MessageDigest cannot find the
desired hashing
   *                algorithm to compute the hash function.
   * @throws IOException
   *                Thrown when File IO exceptions occurs.
   */
  private Vector beginLinuxHash() throws IOException {
    String digest = "", digest2 = "", temp = "", in = "";
     Vector theFiles = new Vector();
     Vector theFiles2 = new Vector();
     byte[] hash, hash2;
```

```
      String[] dirList;
      generateList g1 = new generateList();

      //Check for Custom Directory File that will be created from the
menu
      try {
        BufferedReader custom = new BufferedReader(new
FileReader("dircust"));
        while ((temp = custom.readLine()) != null)
          in = in + temp + "!";
        custom.close();
      }
      catch (FileNotFoundException ferror) {
        System.out.println("There are no custom directories found, using
defaults.");
      }
      try {
        MessageDigest md5 = MessageDigest.getInstance("MD5");
        MessageDigest sha1 = MessageDigest.getInstance("SHA-1");
        int i = 0, status;
        theFiles = g1.readLinux("/sbin");
        theFiles2 = g1.readLinux("/bin");
        theFiles.addAll(theFiles2);
        theFiles2.removeAllElements();
        theFiles2 = g1.readLinux("/usr/sbin");
        theFiles.addAll(theFiles2);
        theFiles2.removeAllElements();
        theFiles2 = g1.readLinux("/usr/bin");
        theFiles.addAll(theFiles2);
        theFiles2.removeAllElements();
        theFiles2 = g1.readLinux("/usr/local/sbin");
        theFiles.addAll(theFiles2);
        theFiles2.removeAllElements();
        theFiles2 = g1.readLinux("/usr/local/bin");
        theFiles.addAll(theFiles2);
        theFiles2.removeAllElements();
        //Add Custom Directories
        if ( (!in.equals("")) && (!in.equals(" ")) ) {
          dirList = f1.split(in, "!");
          //dirList = in.split("!");
          for (int k = 0; k < dirList.length; k++) {
            theFiles2 = g1.readLinux(dirList[k]);
            theFiles.addAll(theFiles2);
            theFiles2.removeAllElements();
          }
          theFiles2.removeAllElements();
        }
        System.out.print("Computing the hashes");
        status = (theFiles.size())/16;
        //System.out.println(status);  //Debugging statement
        while (i < theFiles.size()) {
          FileInputStream fReader = new
FileInputStream(theFiles.elementAt(i).toString());
          int ch;
          while ( (ch = fReader.read()) != -1) {
            md5.update( (byte) ch);
            sha1.update( (byte) ch);
```

```
        }
        hash = md5.digest();
        hash2 = sha1.digest();
        //Convert to Hex Value.
        for (int j = 0; j < hash.length; j++) {
          int v = hash[j] & 0xFF;
          if (v < 16)
            digest += "0";
          digest += Integer.toString(v, 16).toUpperCase();
        }
        //SHA-1 computation
        for (int j = 0; j < hash2.length; j++) {
          int v = hash2[j] & 0xFF;
          if (v < 16)
            digest2 += "0";
          digest2 += Integer.toString(v, 16).toUpperCase();
        }
        if ( (i % status) == 0 )
          System.out.print(".");
        theFiles2.add(i, digest + " " +
theFiles.elementAt(i).toString() + " sha1 " + digest2 + " " +
theFiles.elementAt(i).toString());
        //System.out.println(digest + " " +
theFiles.elementAt(i).toString() + " sha1 " + digest2 + " " +
theFiles.elementAt(i).toString());  //Debugging statement
        digest = "";
        digest2 = "";
        i++;
      }
      System.out.print("complete ");
      return(theFiles2);
    }  //End try
    catch (NoSuchAlgorithmException e) {   }
    return(theFiles2);
  }


  /**
   * Saves the private data member results to a md5 file and a sha1
file.
   *
   * @param  md5
   *             The file name of the saved md5 results.
   * @param  sha1
   *             The file name of the saved sha1 results.
   * @return Nothing
   * @throws IOException
   *             File could not be created.
   */
  private void saveResults(String md5, String sha1) throws IOException
{
    int i = 0;
    int index;
    long mainDate = -1;

    //System.out.println("I am in the write file");  //Debugging
statement
```

```
    try {
      this.md5File = md5;
      this.sha1File = sha1;
      BufferedWriter b = new BufferedWriter(new FileWriter(md5));
      BufferedWriter b2 = new BufferedWriter(new FileWriter(sha1));

      System.out.println("Writing the md5 file to " + this.md5File +
". " +
                         "Writing the sha-1 file to " + this.sha1File+
".");
      //System.out.println("this.results.size = " +
this.results.size());  //Debugging statement
      while( i < this.results.size()) {
        index = (this.results.elementAt(i).toString()).indexOf(" sha1
");
        //System.out.println(index + "String = " +
this.results.elementAt(i).toString() + " i = " + i);  //Debugging
statement
        if (index == -1) {
          System.out.println("File not formatted correctly" + i);
          break;
        }
        b.write( ((this.results.elementAt(i).toString()).substring(0,
index) ) + " ");
        b2.write(
((this.results.elementAt(i).toString()).substring(index+6,
(this.results.elementAt(i).toString()).length()) ) + " ");
        i++;
      }
      b.write("##**##&&?");
      b2.write("##**##&&?");
      b.close();
      b2.close();
      //set the last modified date of both files equal
      File f = new File(this.md5File);
      this.mainDate = f.lastModified();
      f.setLastModified(this.mainDate);
      f.setReadOnly();
      f = new File(this.sha1File);
      f.setLastModified(this.mainDate);
      f.setReadOnly();
      f =null;
    }
    catch (IOException e){
      System.out.println("There was some file permission and/or reading
error: " + e);
      System.exit(1);
    }
  }


  /**
   * Returns the name of the md5 file that was written to disk.
   *
   * @return this.md5File
   *             The name of the md5 file that was written to disk
after the
```

```
     *              hash function has completed.
     */
    public String getMD5File() {
      return(this.md5File);
    }


    /**
     * Returns the name of the sha1 file that was written to disk.
     *
     * @return this.sha1File
     *              The name of the sha1 file that was written to disk
after the
     *              hash function has completed.
     */
    public String getSHA1File() {
      return(this.sha1File);
    }


    /**
     * Returns the date of the md5 file that was written to disk.
     *
     * @return this.mmainDate
     *              The date of the md5 file that was written to disk
after the
     *              hash function has completed.  This date will be used
for both
     *              the md5 last date modified and sha1 last date
modified.
     */
    public long getDate() {
      return(this.mainDate);
    }
}

package sim.agents;
import java.io.*;
import java.net.*;
import sim.interfaces.*;

/**
 * <p>Title: Software Integrity Management</p>
 * <p>Description: Security Software</p>
 * <p>Copyright: Copyright (c) 2004</p>
 * <p>University: University of Louisville, J.B. Speed School of
Engineering</p>
 * @author Joseph H. Brown
 * @version 1.0 Revision 1
 */

public class transport {
  //used when using the integrity option
  public transport() {  }

  //used for sending, must use this constructor when doing secure hash
```

```
  public transport(long date, String md5, String sha1) {
    this.md5File = md5;
    this.sha1File = sha1;
    this.mainDate = date;
  }

  private String md5File = "";
  private String sha1File = "";
  private long mainDate = -1;
  private keyBoard k1 = new keyBoard();

  //Default values
  private String md5Address = "ox.slug.louisville.edu";
  private int md5Port = 7001;
  private String sha1Address = "gradoff8.spd.louisville.edu";
  private int sha1Port = 7007;


  /**
   * Contacts the server and sends the client md5 file to the server
and deletes
   * the local file.
   *
   * @return integer
   *             An integer to signify if the transfer was successful
or not.
   * @throws IOException
   *             Could not properly communicate with the server.
   */
  public int sendToMD5Server() throws IOException {
    Socket clientSocket = null;
    PrintWriter out = null;
    BufferedReader in = null;
    BufferedReader inFromServer = null;
    int index;
    String temp = "";

    try {
      clientSocket = new Socket(this.md5Address, this.md5Port);
      out = new PrintWriter(clientSocket.getOutputStream(), true);
      in = new BufferedReader(new FileReader(this.md5File));
      inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
      String LINE = "";
      int keyStroke;
      System.out.println("Contacting the md5 server.....");
      LINE = inFromServer.readLine();
      if (LINE.equals("Hello from Server_md5")) {
        //Send Operation
        out.println("1");
        //Create file name
        InetAddress add = InetAddress.getLocalHost();
        LINE = add.toString();
        index = LINE.indexOf("/");
        temp = LINE.substring(index+1);
        if (add.hashCode() < 0) {
          int num = add.hashCode();
```

```
          num = num*-1;
          LINE = LINE.substring(0, index) + "." + temp + "." + "(" +
num + ")" + "-" + this.mainDate;
        }
        else
          LINE = LINE.substring(0, index) + "." + temp + "." +
add.hashCode() + "-" + this.mainDate;
          //System.out.println(LINE);  //Debugging statement
        out.println(LINE);
        //Wait to see if it exists
        LINE = inFromServer.readLine();
        if (LINE.equals("no")) {
          System.out.println("Sending the file......");
          while ( (LINE = in.readLine()) != null)
            out.println(LINE);
          in.close();
          File f = new File(this.md5File);
          f.delete();
          f = null;
          return (1);
        }
        if (LINE.equals("yes")) {
          System.out.println("The server has determined that there is a
file " +
                             "already created for this machine.  It is
" +
                             "recommended that you run an integrity
check or " +
                             "if you are 100% positive that you want
to overwrite this " +
                             "file press 1 and enter to overwrite.  If
you overwrite you " +
                             "potentially have corrupted binaries and
will not be aware " +
                             "of their presence.  Press any other key
and then enter to quit.\n" +
                             "You should use the update option if you
have installed any new programs.");
          keyStroke = k1.readInt();
          if (keyStroke == 1) {
            out.println("go");
            System.out.println("Sending the file......");
            while ( (LINE = in.readLine()) != null)
              out.println(LINE);
            in.close();
            File f = new File(this.md5File);
            f.delete();
            f = null;
            return(1);
          }
          else {
            //System.out.println("I am sending -1 back");  //Debuggin
statement
            out.println("stop");
            in.close();
            return (-1);
          }
```

```
        }
      }
      else {
        System.out.println("Did not get correct response from server, "
+
                           "terminating.");
        return (0);
      }
    }
    catch (NumberFormatException n1) {
      //System.out.println("I am sending -1 back");  //Debuggin
statement
      out.println("stop");
      in.close();
      out.close();
      return ( -1);
    }
    catch (UnknownHostException e) {
      System.err.println("Could not contact Agent_md5, agent may be
down. ");
      return (0);
    }
    catch (IOException e) {
      System.err.println("Couldn't get I/O for the connection to
Agent_md5.");
      return (0);
    }
    return (1);
  }


  /**
   * Contacts the server and sends the client md5 file to the server
and deletes
   * the local file.
   *
   * @param address
   *             The address - ipaddress or dns - of the md5 server.
   *        port
   *             The port of the md5 server that will be used for
communication.
   * @return integer
   *             An integer to signify if the transfer was successful
or not.
   * @throws IOException
   *             Could not properly communicate with the server.
   */
  public int sendToMD5Server(String address, int port) throws
IOException {
    Socket clientSocket = null;
    PrintWriter out = null;
    BufferedReader in = null;
    BufferedReader inFromServer = null;
    int index;
    String temp = "";

    try {
```

```
      clientSocket = new Socket(address, port);
      out = new PrintWriter(clientSocket.getOutputStream(), true);
      in = new BufferedReader(new FileReader(this.md5File));
      inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
      String LINE = "";
      int keyStroke;
      System.out.println("Contacting the md5 server.....");
      LINE = inFromServer.readLine();
      if (LINE.equals("Hello from Server_md5")) {
        //Send Operation
        out.println("1");
        //Create file name
        InetAddress add = InetAddress.getLocalHost();
        LINE = add.toString();
        index = LINE.indexOf("/");
        temp = LINE.substring(index+1);
        if (add.hashCode() < 0) {
          int num = add.hashCode();
          num = num*-1;
          LINE = LINE.substring(0, index) + "." + temp + "." + "(" +
num + ")" + "-" + this.mainDate;
        }
        else
          LINE = LINE.substring(0, index) + "." + temp + "." +
add.hashCode() + "-" + this.mainDate;
          //System.out.println(LINE);
        out.println(LINE);
        //Wait to see if it exists
        LINE = inFromServer.readLine();
        if (LINE.equals("no")) {
          System.out.println("Sending the file.....");
          while ( (LINE = in.readLine()) != null)
            out.println(LINE);
          in.close();
          File f = new File(this.md5File);
          f.delete();
          f = null;
          return (1);
        }
        if (LINE.equals("yes")) {
          System.out.println("The server has determined that there is a
file " +
                             "already created for this machine.\nIt is
" +
                             "recommended that you run an integrity
check or " +
                             "if you are 100% positive that you want
to overwrite this " +
                             "file press 1 and enter to overwrite.\nIf
you overwrite you " +
                             "potentially have corrupted binaries and
will not be aware " +
                             "of their presence.\nPress any other key
and then enter to quit.\n" +
                             "You should use the update option if you
have installed any new programs.");
```

```
            keyStroke = k1.readInt();
            if (keyStroke == 1) {
              out.println("go");
              System.out.println("Sending the file.....");
              while ( (LINE = in.readLine()) != null)
                out.println(LINE);
              in.close();
              File f = new File(this.md5File);
              f.delete();
              f = null;
              return(1);
            }
            else {
              //System.out.println("I am sending -1 back");  //Debugging
statement
              out.println("stop");
              in.close();
              return ( -1);
            }
          }
        }
        else {
          System.out.println("Did not get correct response from server, "
+
                             "terminating.");
          return (0);
        }
      }
      catch (NumberFormatException n1) {
        //System.out.println("I am sending -1 back");  //Debugging
statement
        out.println("stop");
        in.close();
        out.close();
        return (-1);
      }
      catch (UnknownHostException e) {
        System.err.println("Could not contact Agent_md5, agent may be
down. ");
        return (0);
      }
      catch (IOException e) {
        System.err.println("Couldn't get I/O for the connection to
Agent_md5.");
        return (0);
      }
      return (1);
    }


    /**
     * Contacts the server and sends the client sha1 file to the server
and deletes
     * the local file.
     *
     * @return integer
```

```
    *             An integer to signify if the transfer was successful
or not.
    * @throws IOException
    *             Could not properly communicate with the server.
    */
  public int sendToSHA1Server() throws IOException {
    Socket clientSocket = null;
    PrintWriter out = null;
    BufferedReader in = null;
    BufferedReader inFromServer = null;
    int index;
    String temp = "";

    try {
      clientSocket = new Socket(this.sha1Address, this.sha1Port);
      out = new PrintWriter(clientSocket.getOutputStream(), true);
      in = new BufferedReader(new FileReader(this.sha1File));
      inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
      String LINE = "";
      int keyStroke;
      System.out.println("Contacting the sha1 server......");
      LINE = inFromServer.readLine();
      if (LINE.equals("Hello from Server_sha1")) {
        //Send Operation
        out.println("1");
        //Create file name
        InetAddress add = InetAddress.getLocalHost();
        LINE = add.toString();
        index = LINE.indexOf("/");
        temp = LINE.substring(index+1);
        if (add.hashCode() < 0) {
          int num = add.hashCode();
          num = num*-1;
          LINE = LINE.substring(0, index) + "." + temp + "." + "(" +
num + ")" + "-" + this.mainDate;
        }
        else
          LINE = LINE.substring(0, index) + "." + temp + "." +
add.hashCode() + "-" + this.mainDate;
          //System.out.println(LINE);
        out.println(LINE);
        //Wait to see if it exists
        LINE = inFromServer.readLine();
        if (LINE.equals("no")) {
          System.out.println("Sending the file......");
          while ( (LINE = in.readLine()) != null)
            out.println(LINE);
          in.close();
          File f = new File(this.sha1File);
          f.delete();
          f = null;
          return (1);
        }
        if (LINE.equals("yes")) {
          System.out.println("The server has determined that there is a
file " +
```

```
                                "already created for this machine.  It is
" +
                                "recommended that you run an integrity
check or " +
                                "if you are 100% positive that you want
to overwrite this " +
                                "file press 1 and enter to overwrite.  If
you overwrite you " +
                                "potentially have corrupted binaries and
will not be aware " +
                                "of their presence.  Press any other key
and then enter to quit.  " +
                                "You should use the update option if you
have installed any new programs.");
          keyStroke = k1.readInt();
          if (keyStroke == 1) {
            out.println("go");
            System.out.println("Sending the file......");
            while ( (LINE = in.readLine()) != null)
              out.println(LINE);
            in.close();
            File f = new File(this.sha1File);
            f.delete();
            f = null;
            return(1);
          }
          else {
            //System.out.println("I am sending -1 back");  //Debugging
statement
            out.println("stop");
            in.close();
            return ( -1);
          }
        }
      }
      else {
        System.out.println("Did not get correct response from server, "
+
                           "terminating.");
        return (0);
      }
    }
    catch (NumberFormatException n1) {
      //System.out.println("I am sending -1 back");  //Debugging
statement
      out.println("stop");
      in.close();
      out.close();
      return (-1);
    }
    catch (UnknownHostException e) {
      System.err.println("Could not contact Agent_sha1, agent may be
down.  ");
      return (0);
    }
    catch (IOException e) {
```

```
      System.err.println("Couldn't get I/O for the connection to
Agent_sha1.");
      return (0);
    }
    return (1);
  }


  /**
   * Contacts the server and sends the client sha1 file to the server
and deletes
   * the local file.
   *
   * @param address
   *              The address - ipaddress or dns - of the sha1 server.
   *        port
   *              The port of the sha1 server that will be used for
communication.
   * @return integer
   *              An integer to signify if the transfer was successful
or not.
   * @throws IOException
   *              Could not properly communicate with the server.
   */

  public int sendToSHA1Server(String address, int port) throws
IOException {
    Socket clientSocket = null;
    PrintWriter out = null;
    BufferedReader in = null;
    BufferedReader inFromServer = null;
    int index;
    String temp = "";

    try {
      clientSocket = new Socket(address, port);
      out = new PrintWriter(clientSocket.getOutputStream(), true);
      in = new BufferedReader(new FileReader(this.sha1File));
      inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
      String LINE = "";
      int keyStroke;
      System.out.println("Contacting the sha1 server.....");
      LINE = inFromServer.readLine();
      if (LINE.equals("Hello from Server_sha1")) {
        //Send Operation
        out.println("1");
        //Create file name
        InetAddress add = InetAddress.getLocalHost();
        LINE = add.toString();
        index = LINE.indexOf("/");
        temp = LINE.substring(index+1);
        if (add.hashCode() < 0) {
          int num = add.hashCode();
          num = num*-1;
          LINE = LINE.substring(0, index) + "." + temp + "." + "(" +
num + ")" + "-" + this.mainDate;
```

```
        }
        else
          LINE = LINE.substring(0, index) + "." + temp + "." +
add.hashCode() + "-" + this.mainDate;
          //System.out.println(LINE);
        out.println(LINE);
        //Wait to see if it exists
        LINE = inFromServer.readLine();
        if (LINE.equals("no")) {
          System.out.println("Sending the file......");
          while ( (LINE = in.readLine()) != null)
            out.println(LINE);
          in.close();
          File f = new File(this.sha1File);
          f.delete();
          f = null;
          return (1);
        }
        if (LINE.equals("yes")) {
          System.out.println("The server has determined that there is a
file " +
                             "already created for this machine.  It is
" +
                             "recommended that you run an integrity
check or " +
                             "if you are 100% positive that you want
to overwrite this " +
                             "file press 1 and enter to overwrite.  If
you overwrite you " +
                             "potentially have corrupted binaries and
will not be aware " +
                             "of their presence.  Press any other key
and then enter to quit.  " +
                             "You should use the update option if you
have installed any new programs.");
          keyStroke = k1.readInt();
          if (keyStroke == 1) {
            out.println("go");
            System.out.println("Sending the file......");
            while ( (LINE = in.readLine()) != null)
              out.println(LINE);
            in.close();
            File f = new File(this.sha1File);
            f.delete();
            f = null;
            return(1);
          }
          else {
            //System.out.println("I am sending -1 back");  //Debugging
statement
            out.println("stop");
            in.close();
            return ( -1);
          }
        }
      }
      else {
```

```
        System.out.println("Did not get correct response from server, "
+
                           "terminating.");
        return (0);
      }
    }
    catch (NumberFormatException n1) {
      //System.out.println("I am sending -1 back");  /Debugging
statement
      out.println("stop");
      in.close();
      out.close();
      return ( -1);
    }
    catch (UnknownHostException e) {
      System.err.println("Could not contact Agent_sha1, agent may be
down. ");
      return (0);
    }
    catch (IOException e) {
      System.err.println("Couldn't get I/O for the connection to
Agent_sha1.");
      return (0);
    }
    return (1);
  }


  /**
   * Contacts the md5 server and gets the md5 file from the server.
   *
   * @param  address
   *             The address - ipaddress or dns - of the md5 server.
   *         port
   *             The port of the md5 server that will be used for
communication.
   *         serverFile
   *             The name of the file that will store the results from
the md5 server.
   * @return integer
   *             An integer to signify if the transfer was successful
or not.
   * @throws IOException
   *             Could not properly communicate with the server.
   */
  public int getFromServerMD5(String address, int port, String
serverFile) throws IOException {
    Socket clientSocket = null;
    PrintWriter out = null;
    BufferedReader in = null;
    BufferedReader inFromServer = null;
    int index;
    String temp = "", outputLine = "";

    try {
      clientSocket = new Socket(address, port);
      out = new PrintWriter(clientSocket.getOutputStream(), true);
```

```
      inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
      BufferedWriter bWriter = new BufferedWriter(new
FileWriter(serverFile));
      String LINE = "";
      System.out.println("Contacting the md5 server.....");
      LINE = inFromServer.readLine();
      if (LINE.equals("Hello from Server_md5"))
      {
        //Send Operation
        out.println("2");
        //Create file name
        InetAddress add = InetAddress.getLocalHost();
        LINE = add.toString();
        index = LINE.indexOf("/");
        temp = LINE.substring(index+1);
        if (add.hashCode() < 0) {
          int num = add.hashCode();
          num = num*-1;
          LINE = LINE.substring(0, index) + "." + temp + "." + "(" +
num + ")" + "-" + this.mainDate;
        }
        else
          LINE = LINE.substring(0, index) + "." + temp + "." +
add.hashCode() + "-" + this.mainDate;
          //System.out.println(LINE);  //Debugging statement
        out.println(LINE);
        //Wait to see if it exists
        LINE = inFromServer.readLine();
        if (LINE.equals("no")) {
          System.out.println("There is no file stored on the server
associated " +
                             "with your machine.  Please use the Secure
Hash option");
          bWriter.close();
          return (-1);
        }
        if (LINE.equals("yes")) {
          System.out.println("Receiving the file.......");
          while (! (outputLine =
inFromServer.readLine()).equals("##**##&&?"))
            bWriter.write(outputLine + "\n");
          bWriter.write(outputLine);
          bWriter.close();
        }
      }
      else {
        System.out.println("Did not get correct response from server, "
+
                           "terminating.");
        in.close();
        out.close();
        return (-3);
      }
    }
    catch (UnknownHostException e) {
```

```
      System.err.println("Could not contact Agent_md5, agent may be
down.\n");
      return (0);
    }
    catch (IOException e) {
      System.err.println("Couldn't get I/O for the connection to
Agent_md5.");
      return (0);
    }
    return (1);
  }


  /**
   * Contacts the md5 server and gets the md5 file from the server.
   *
   * @param
   *          serverFile
   *              The name of the file that will store the results from
the md5 server.
   * @return integer
   *              An integer to signify if the transfer was successful
or not.
   * @throws IOException
   *              Could not properly communicate with the server.
   */
  public int getFromServerMD5(String serverFile) throws IOException {
    Socket clientSocket = null;
    PrintWriter out = null;
    BufferedReader in = null;
    BufferedReader inFromServer = null;
    int index;
    String temp = "", outputLine = "";

    try {
      clientSocket = new Socket(this.md5Address, this.md5Port);
      out = new PrintWriter(clientSocket.getOutputStream(), true);
      inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
      BufferedWriter bWriter = new BufferedWriter(new
FileWriter(serverFile));
      String LINE = "";
      System.out.println("Contacting the md5 server.....");
      LINE = inFromServer.readLine();
      if (LINE.equals("Hello from Server_md5")) {
        //Send Operation
        out.println("2");
        //Create file name
        InetAddress add = InetAddress.getLocalHost();
        LINE = add.toString();
        index = LINE.indexOf("/");
        temp = LINE.substring(index+1);
        if (add.hashCode() < 0) {
          int num = add.hashCode();
          num = num*-1;
          LINE = LINE.substring(0, index) + "." + temp + "." + "(" +
num + ")" + "-" + this.mainDate;
```

```
        }
        else
          LINE = LINE.substring(0, index) + "." + temp + "." +
add.hashCode() + "-" + this.mainDate;
          //System.out.println(LINE);  //Debugging statement
        out.println(LINE);
        //Wait to see if it exists
        LINE = inFromServer.readLine();
        if (LINE.equals("no")) {
          System.out.println("There is no file stored on the server
associated " +
                             "with your machine.  Please use the Secure
Hash option");
          bWriter.close();
          return (-1);
        }
        if (LINE.equals("yes")) {
          System.out.println("Receiving the file.......");
          while (! (outputLine =
inFromServer.readLine()).equals("##**##&&?"))
            bWriter.write(outputLine + "  ");
          bWriter.write(outputLine);
          bWriter.close();
        }
      }
      else {
        System.out.println("Did not get correct response from server, "
+
                           "terminating.");
        in.close();
        out.close();
        return (-3);
      }
    }
    catch (UnknownHostException e) {
      System.err.println("Could not contact Agent_md5, agent may be
down.  ");
      return (0);
    }
    catch (IOException e) {
      System.err.println("Couldn't get I/O for the connection to
Agent_md5.");
      return (0);
    }
    return (1);
  }


  /**
   * Contacts the sha1 server and gets the sha1 file from the server.
   *
   * @param  address
   *             The address - ipaddress or dns - of the sha1 server.
   *         port
   *             The port of the sha1 server that will be used for
communication.
   *         serverFile
```

```
   *            The name of the file that will store the results from
the sha1 server.
   * @return integer
   *            An integer to signify if the transfer was successful
or not.
   * @throws IOException
   *            Could not properly communicate with the server.
   */
  public int getFromServerSHA1(String address, int port, String
serverFile) throws IOException {
    Socket clientSocket = null;
    PrintWriter out = null;
    BufferedReader in = null;
    BufferedReader inFromServer = null;
    int index;
    String temp = "", outputLine = "";

    try {
      clientSocket = new Socket(address, port);
      out = new PrintWriter(clientSocket.getOutputStream(), true);
      inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
      BufferedWriter bWriter = new BufferedWriter(new
FileWriter(serverFile));
      String LINE = "";

      System.out.println("Contacting the sha1 server.....");
      LINE = inFromServer.readLine();
      if (LINE.equals("Hello from Server_sha1")) {
        //Send Operation
        out.println("2");
        //Create file name
        InetAddress add = InetAddress.getLocalHost();
        LINE = add.toString();
        index = LINE.indexOf("/");
        temp = LINE.substring(index+1);
        if (add.hashCode() < 0) {
          int num = add.hashCode();
          num = num*-1;
          LINE = LINE.substring(0, index) + "." + temp + "." + "(" +
num + ")" + "-" + this.mainDate;
        }
        else
          LINE = LINE.substring(0, index) + "." + temp + "." +
add.hashCode() + "-" + this.mainDate;
          //System.out.println(LINE);  //Debugging statement
        out.println(LINE);
        //Wait to see if it exists
        LINE = inFromServer.readLine();
        if (LINE.equals("no")) {
          System.out.println("There is no file stored on the server
associated " +
                             "with your machine.  Please use the Secure
Hash option");
          bWriter.close();
          return (-1);
        }
```

```
        if (LINE.equals("yes")) {
          System.out.println("Receiving the file.......");
          while (! (outputLine =
inFromServer.readLine()).equals("##**###&&?"))
            bWriter.write(outputLine + "\n");
          bWriter.write(outputLine);
          bWriter.close();
        }
      }
      else {
        System.out.println("Did not get correct response from server, "
+
                           "terminating.");
        in.close();
        out.close();
        return (0);
      }
    }
    catch (UnknownHostException e) {
      System.err.println("Could not contact Agent_sha1, agent may be
down.\n");
      return (0);
    }
    catch (IOException e) {
      System.err.println("Couldn't get I/O for the connection to
Agent_sha1.");
      return (0);
    }
    return (1);
  }


  /**
   * Contacts the sha1 server and gets the sha1 file from the server.
   *
   * @param  address
   *             The address - ipaddress or dns - of the sha1 server.
*
   * @return integer
   *             An integer to signify if the transfer was successful
or not.
   * @throws IOException
   *             Could not properly communicate with the server.
   */
  public int getFromServerSHA1(String serverFile) throws IOException {
    Socket clientSocket = null;
    PrintWriter out = null;
    BufferedReader in = null;
    BufferedReader inFromServer = null;
    int index;
    String temp = "", outputLine = "";

    try {
      clientSocket = new Socket(this.sha1Address, this.sha1Port);
      out = new PrintWriter(clientSocket.getOutputStream(), true);
      inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
```

```
      BufferedWriter bWriter = new BufferedWriter(new
FileWriter(serverFile));
      String LINE = "";
      System.out.println("Contacting the sha1 server.....");
      LINE = inFromServer.readLine();
      if (LINE.equals("Hello from Server_sha1")) {
        //Send Operation
        out.println("2");
        //Create file name
        InetAddress add = InetAddress.getLocalHost();
        LINE = add.toString();
        index = LINE.indexOf("/");
        temp = LINE.substring(index+1);
        if (add.hashCode() < 0) {
          int num = add.hashCode();
          num = num*-1;
          LINE = LINE.substring(0, index) + "." + temp + "." + "(" +
num + ")" + "-" + this.mainDate;
        }
        else
          LINE = LINE.substring(0, index) + "." + temp + "." +
add.hashCode() + "-" + this.mainDate;
          //System.out.println(LINE);  //Debugging statement
        out.println(LINE);
        //Wait to see if it exists
        LINE = inFromServer.readLine();
        if (LINE.equals("no")) {
          System.out.println("There is no file stored on the server
associated " +
                             "with your machine.  Please use the Secure
Hash option");
          bWriter.close();
          return (-1);
        }
        if (LINE.equals("yes")) {
          System.out.println("Receiving the file.......");
          while (! (outputLine =
inFromServer.readLine()).equals("##**##&&?"))
            bWriter.write(outputLine + "\n");
          bWriter.write(outputLine);
          bWriter.close();
        }
      }
      else {
        System.out.println("Did not get correct response from server, "
+
                           "terminating.");
        in.close();
        out.close();
        return (0);
      }
    }
    catch (UnknownHostException e) {
      System.err.println("Could not contact Agent_sha1, agent may be
down.\n");
      return (0);
    }
```

```
    catch (IOException e) {
      System.err.println("Couldn't get I/O for the connection to
Agent_sha1.");
      return (0);
    }
    return (1);
  }


  /**
   * Contacts the md5 server and gets the last modified date from md5
file on
   * the server.
   *
   * @return date
   *             The last date modified or an integer to signify that
the
   *             server is down.
   * @throws IOException
   *             Could not properly communicate with the server.
   */
  public long getLastModifiedMD5() throws IOException {
    long date = 0;
    Socket clientSocket = null;
    PrintWriter out = null;
    BufferedReader in = null;
    BufferedReader inFromServer = null;
    int index;
    String temp = "";

    try {
      clientSocket = new Socket(this.md5Address, this.md5Port);
      out = new PrintWriter(clientSocket.getOutputStream(), true);
      inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
      String LINE = "";
      System.out.println("Contacting the md5 server.....");
      LINE = inFromServer.readLine();
      if (LINE.equals("Hello from Server_md5")) {
        //Send Operation
        out.println("3");
        //Create file name
        InetAddress add = InetAddress.getLocalHost();
        LINE = add.toString();
        index = LINE.indexOf("/");
        temp = LINE.substring(index+1);
        if (add.hashCode() < 0) {
          int num = add.hashCode();
          num = num*-1;
          LINE = LINE.substring(0, index) + "." + temp + "." + "(" +
num + ")" + "-" + this.mainDate;
        }
        else
          LINE = LINE.substring(0, index) + "." + temp + "." +
add.hashCode() + "-" + this.mainDate;
        out.println(LINE);
        //Wait to see if it exists
```

```
        LINE = inFromServer.readLine();
        if (LINE.equals("no")) {
          System.out.println("The file does not exist according to get
last modified - "
                             + "transport class error");
          System.exit(1);
        }
        if (LINE.equals("yes"))  {
          LINE = inFromServer.readLine();
          date = Long.parseLong(LINE);
          in.close();
        }
      }
      else {
        System.out.println("Did not get correct response from server,
terminating NOW!");
        System.exit(1);
        return (0);
      }
    }
    catch (UnknownHostException e){
      System.err.println("Could not contact Agent_md5, agent may be
down. ");
      return (-1);
    }
    catch (IOException e) {
      System.err.println("Couldn't get I/O for the connection to
Agent_md5.");
      return (-1);
    }
    return (date);
  }


  /**
   * Contacts the md5 server and gets the last modified date from md5
file on
   * the server.
   *
   * @param  address
   *             The address - ip address or dns - of the md5 server.
   *         port
   *             The port of the sha1 server that will be used for
communication.
   * @return date
   *             The last date modified or an integer to signify that
the
   *             server is down.
   * @throws IOException
   *             Could not properly communicate with the server.
   */
  public long getLastModifiedMD5(String address, int port) throws
IOException {
    long date = 0;
    Socket clientSocket = null;
    PrintWriter out = null;
    BufferedReader in = null;
```

```
    BufferedReader inFromServer = null;
    int index;
    String temp = "";

    try {
      clientSocket = new Socket(address, port);
      out = new PrintWriter(clientSocket.getOutputStream(), true);
      inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
      String LINE = "";
      System.out.println("Contacting the md5 server.....");
      LINE = inFromServer.readLine();
      if (LINE.equals("Hello from Server_md5")) {
        //Send Operation
        out.println("3");
        //Create file name
        InetAddress add = InetAddress.getLocalHost();
        LINE = add.toString();
        index = LINE.indexOf("/");
        temp = LINE.substring(index+1);
        if (add.hashCode() < 0) {
          int num = add.hashCode();
          num = num*-1;
          LINE = LINE.substring(0, index) + "." + temp + "." + "(" +
num + ")" + "-" + this.mainDate;
        }
        else
          LINE = LINE.substring(0, index) + "." + temp + "." +
add.hashCode() + "-" + this.mainDate;
        out.println(LINE);
        //Wait to see if it exists
        LINE = inFromServer.readLine();
        if (LINE.equals("no")) {
           System.out.println("The file does not exist according to get
last modified - "
                              + "transport class error");
          System.exit(1);
         }
        if (LINE.equals("yes")) {
          LINE = inFromServer.readLine();
          date = Long.parseLong(LINE);
          in.close();
        }
      }
      else {
        System.out.println("Did not get correct response from server,
terminating NOW!");
        System.exit(1);
        return (0);
      }
    }
    catch (UnknownHostException e) {
      System.err.println("Could not contact Agent_md5, agent may be
down. ");
      return (-1);
    }
    catch (IOException e) {
```

```
      System.err.println("Couldn't get I/O for the connection to
Agent_md5.");
      return (-1);
    }
    return (date);
  }


  /**
   * Contacts the sha1 server and gets the last modified date from sha1
file on
   * the server.
   *
   * @return date
   *             The last date modified or an integer to signify that
the
   *             server is down.
   * @throws IOException
   *             Could not properly communicate with the server.
   */
  public long getLastModifiedSHA1() {
    long date = 0;
    Socket clientSocket = null;
    PrintWriter out = null;
    BufferedReader in = null;
    BufferedReader inFromServer = null;
    int index;
    String temp = "";

    try {
      clientSocket = new Socket(this.sha1Address, this.sha1Port);
      out = new PrintWriter(clientSocket.getOutputStream(), true);
      inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
      String LINE = "";
      System.out.println("Contacting the sha1 server.....");
      LINE = inFromServer.readLine();
      if (LINE.equals("Hello from Server_sha1")) {
        //Send Operation
        out.println("3");
        //Create file name
        InetAddress add = InetAddress.getLocalHost();
        LINE = add.toString();
        index = LINE.indexOf("/");
        temp = LINE.substring(index+1);
        if (add.hashCode() < 0) {
          int num = add.hashCode();
          num = num*-1;
          LINE = LINE.substring(0, index) + "." + temp + "." + "(" +
num + ")" + "-" + this.mainDate;
        }
        else
          LINE = LINE.substring(0, index) + "." + temp + "." +
add.hashCode() + "-" + this.mainDate;
        out.println(LINE);
        //Wait to see if it exists
        LINE = inFromServer.readLine();
```

```
        if (LINE.equals("no")) {
           System.out.println("The file does not exist according to get
last modified - "
                              + "transport class error");
           System.exit(1);
         }
        if (LINE.equals("yes"))
        {
          LINE = inFromServer.readLine();
          date = Long.parseLong(LINE);
          in.close();
        }
      }
      else {
        System.out.println("Did not get correct response from server,
terminating NOW!");
        System.exit(1);
        return (0);
      }
    }
    catch (UnknownHostException e) {
      System.err.println("Could not contact Agent_sha1, agent may be
down.");
      return (-1);
    }
    catch (IOException e) {
      System.err.println("Couldn't get I/O for the connection to
Agent_sha1.");
      return (-1);
    }
    return (date);
  }


  /**
   * Contacts the sha1 server and gets the last modified date from sha1
file on
   * the server.
   *
   * @param  address
   *             The address - ip address or dns - of the sha1 server.
   *         port
   *             The port of the sha1 server that will be used for
communication.
   * @return date
   *             The last date modified or an integer to signify that
the
   *             server is down.
   * @throws IOException
   *             Could not properly communicate with the server.
   */
  public long getLastModifiedSHA1(String address, int port) {
    long date = 0;
    Socket clientSocket = null;
    PrintWriter out = null;
    BufferedReader in = null;
    BufferedReader inFromServer = null;
```

```
    int index;
    String temp = "";

    try {
      clientSocket = new Socket(address, port);
      out = new PrintWriter(clientSocket.getOutputStream(), true);
      inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
      String LINE = "";
      System.out.println("Contacting the sha1 server.....");
      LINE = inFromServer.readLine();
      if (LINE.equals("Hello from Server_sha1")) {
        //Send Operation
        out.println("3");
        //Create file name
        InetAddress add = InetAddress.getLocalHost();
        LINE = add.toString();
        index = LINE.indexOf("/");
        temp = LINE.substring(index+1);
        if (add.hashCode() < 0) {
          int num = add.hashCode();
          num = num*-1;
          LINE = LINE.substring(0, index) + "." + temp + "." + "(" +
num + ")" + "-" + this.mainDate;
        }
        else
          LINE = LINE.substring(0, index) + "." + temp + "." +
add.hashCode() + "-" + this.mainDate;
        out.println(LINE);
        //Wait to see if it exists
        LINE = inFromServer.readLine();
        if (LINE.equals("no")) {
           System.out.println("The file does not exist according to get
last modified - "
                              + "transport class error");
          System.exit(1);
         }
        if (LINE.equals("yes")) {
          LINE = inFromServer.readLine();
          date = Long.parseLong(LINE);
          in.close();
        }
      }
      else {
        System.out.println("Did not get correct response from server,
terminating NOW!");
        System.exit(1);
        return (0);
      }
    }
    catch (UnknownHostException e) {
      System.err.println("Could not contact Agent_sha1, agent may be
down. ");
      return (-1);
    }
    catch (IOException e) {
```

```
      System.err.println("Couldn't get I/O for the connection to
Agent_sha1.");
      return (-1);
    }
    return (date);
  }


  /**
   * Sends the verified client file up to the sha1 server overwriting
the old server
   * file.
   *
   * @param  file
   *             The client file to upload and overwrite the server
file.
   * @return integer
   *             An integer to signify if the transfer was successful
or not.
   * @throws IOException
   *             Could not properly communicate with the server.
   */
  public int overrideSHA1Server(String file) throws IOException {
    Socket clientSocket = null;
    PrintWriter out = null;
    BufferedReader in = null;
    BufferedReader inFromServer = null;
    int index;
    String temp = "";

    try {
      clientSocket = new Socket(this.sha1Address, this.sha1Port);
      out = new PrintWriter(clientSocket.getOutputStream(), true);
      in = new BufferedReader(new FileReader(file));
      inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
      String LINE = "";
      int keyStroke;
      System.out.println("Contacting the sha1 server.....");
      LINE = inFromServer.readLine();
      if (LINE.equals("Hello from Server_sha1")) {
        //Send Operation
        out.println("1");
        //Create file name
        InetAddress add = InetAddress.getLocalHost();
        LINE = add.toString();
        index = LINE.indexOf("/");
        temp = LINE.substring(index+1);
        if (add.hashCode() < 0) {
          int num = add.hashCode();
          num = num*-1;
          LINE = LINE.substring(0, index) + "." + temp + "." + "(" +
num + ")" + "-" + this.mainDate;
        }
        else
          LINE = LINE.substring(0, index) + "." + temp + "." +
add.hashCode() + "-" + this.mainDate;
```

```
        out.println(LINE);
        //Wait to see if it exists
        LINE = inFromServer.readLine();
        if (LINE.equals("no")) {
          System.out.println("Sending the file.....");
          while ( (LINE = in.readLine()) != null)
            out.println(LINE);
          in.close();
          out.close();
          File f = new File(file);
          f.delete();
          f = null;
          return (1);
        }
        if (LINE.equals("yes")) {
          out.println("go");
          System.out.println("Sending the file.....");
          while ( (LINE = in.readLine()) != null)
            out.println(LINE);
          in.close();
          out.close();
          File f = new File(file);
          f.delete();
          f = null;
          return(1);
        }
      }
      else {
        System.out.println("Did not get correct response from server, "
+
                           "terminating.");
        return (0);
      }
    }
    catch (UnknownHostException e) {
      System.err.println("Could not contact Agent_sha1, agent may be
down.");
      return (0);
    }
    catch (IOException e) {
      System.err.println("Couldn't get I/O for the connection to
Agent_sha1.");
      return (0);
    }
    return (1);
  }


  /**
   * Sends the verified client file up to the md5 server overwriting
the old server
   * file.
   *
   * @param  file
   *             The client file to upload and overwrite the server
file.
   * @return integer
```

```
   *            An integer to signify if the transfer was successful
or not.
   * @throws IOException
   *             Could not properly communicate with the server.
   */
  public int overrideMD5Server(String file) throws IOException {
    Socket clientSocket = null;
    PrintWriter out = null;
    BufferedReader in = null;
    BufferedReader inFromServer = null;
    int index;
    String temp = "";

    try {
      clientSocket = new Socket(this.md5Address, this.md5Port);
      out = new PrintWriter(clientSocket.getOutputStream(), true);
      in = new BufferedReader(new FileReader(file));
      inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
      String LINE = "";
      int keyStroke;
      System.out.println("Contacting the md5 server.....");
      LINE = inFromServer.readLine();
      if (LINE.equals("Hello from Server_md5")) {
        //Send Operation
        out.println("1");
        //Create file name
        InetAddress add = InetAddress.getLocalHost();
        LINE = add.toString();
        index = LINE.indexOf("/");
        temp = LINE.substring(index+1);
        if (add.hashCode() < 0) {
          int num = add.hashCode();
          num = num*-1;
          LINE = LINE.substring(0, index) + "." + temp + "." + "(" +
num + ")" + "-" + this.mainDate;
        }
        else
          LINE = LINE.substring(0, index) + "." + temp + "." +
add.hashCode() + "-" + this.mainDate;
        out.println(LINE);

        //Wait to see if it exists
        LINE = inFromServer.readLine();
        if (LINE.equals("no")) {
          System.out.println("Sending the file.....");
          while ( (LINE = in.readLine()) != null)
            out.println(LINE);
          in.close();
          out.close();
          File f = new File(file);
          f.delete();
          f = null;
          return (1);
        }
        if (LINE.equals("yes")) {
          out.println("go");
```

```
          System.out.println("Sending the file.....");
          while ( (LINE = in.readLine()) != null)
            out.println(LINE);
          in.close();
          out.close();
          File f = new File(file);
          f.delete();
          f = null;
          return(1);
        }
      }
      else {
        System.out.println("Did not get correct response from server, "
+
                           "terminating.");
        return (0);
      }
    }
    catch (UnknownHostException e) {
      System.err.println("Could not contact Agent_md5, agent may be
down. ");
      return (0);
    }
    catch (IOException e) {
      System.err.println("Couldn't get I/O for the connection to
Agent_md5.");
      return (0);
    }
    return (1);
  }


  /**
   * Returns md5 server address which is the value of this.md5Address.
   *
   * @return this.md5Address
   *             The value of the md5 server address in the instance of
this class.
   */
  public String getmd5Address() {
    return (this.md5Address);
  }


  /**
   * Returns md5 server port which is the value of this.md5Port.
   *
   * @return this.md5Port
   *             The value of the md5 server port in the instance of
this class.
   */
  public int getmd5Port() {
    return (this.md5Port);
  }


  /**
```

```
    * Returns sha1 server address which is the value of
this.sha1Address.
    *
    * @return this.sha1Address
    *             The value of the sha1 server address in the instance
of this class.
    */
  public String getsha1Address() {
    return (this.sha1Address);
  }


  /**
    * Returns sha1 server port which is the value of this.sha1Port.
    *
    * @return this.sha1Port
    *             The value of the sha1 server port in the instance of
this class.
    */
  public int getsha1Port() {
    return (this.sha1Port);
  }
}
```

APPENDIX IV. Selected Method Summaries

sim.agents

Class integrityCheck

```
java.lang.Object
  |
  +--sim.agents.integrityCheck
```

Method Summary	
int	**checkIntegrity**(String localFile, String serverFile, int indicator) Checks the integrity of the local file compared to the server file.
void	**performIntegrityCheck**(String serverMD5, String md5Client, String serverSHA1, String sha1Client, long date) Responsible for performing the integrity check.

sim.agents

Class secureHash

```
java.lang.Object
  |
  +--sim.agents.secureHash
```

Method Summary	
long	**getDate**() Returns the date of the md5 file that was written to disk.
String	**getMD5File**() Returns the name of the md5 file that was written to disk.
String	**getSHA1File**() Returns the name of the sha1 file that was written to disk.
String	**hash**(java.lang.String os) Decides which version of the OS is found and calls the necessary method.

sim.agents

Class transport

```
java.lang.Object
   |
   +--sim.agents.transport
```

Method Summary	
int	**getFromServerMD5**(String serverFile) Contacts the md5 server and gets the md5 file from the server.
int	**getFromServerMD5**(String address, int port, java.lang.String serverFile) Contacts the md5 server and gets the md5 file from the server.
int	**getFromServerSHA1**(String serverFile) Contacts the sha1 server and gets the sha1 file from the server.
int	**getFromServerSHA1**(String address, int port, String serverFile) Contacts the sha1 server and gets the sha1 file from the server.
long	**getLastModifiedMD5**() Contacts the md5 server and gets the last modified date from md5 file on the server.
long	**getLastModifiedMD5**(String address, int port) Contacts the md5 server and gets the last modified date from md5 file on the server.
long	**getLastModifiedSHA1**() Contacts the sha1 server and gets the last modified date from sha1 file on the server.
long	**getLastModifiedSHA1**(String address, int port) Contacts the sha1 server and gets the last modified date from sha1 file on the server.
String	**getmd5Address**() Returns md5 server address which is the value of this.md5Address.
int	**getmd5Port**() Returns md5 server port which is the value of this.md5Port.
String	**getsha1Address**() Returns sha1 server address which is the value of this.sha1Address.
int	**getsha1Port**() Returns sha1 server port which is the value of this.sha1Port.
int	**overrideMD5Server**(String file) Sends the verified client file up to the md5 server overwriting the old server file.
int	**overrideSHA1Server**(String file) Sends the verified client file up to the sha1 server overwriting the old server file.

int	**sendToMD5Server**() Contacts the server and sends the client md5 file to the server and deletes the local file.
int	**sendToMD5Server**(String address, int port) Contacts the server and sends the client md5 file to the server and deletes the local file.
int	**sendToSHA1Server**() Contacts the server and sends the client sha1 file to the server and deletes the local file.
int	**sendToSHA1Server**(String address, int port) Contacts the server and sends the client sha1 file to the server and deletes the local file.

APPENDIX V. Miscellaneous Source Code

```
import java.io.*;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

/**
* <p>Title: Software Integrity Management</p>
* <p>Description: Security Software</p>
* <p>Copyright: Copyright (c) 2004</p>
* <p>University:: University of Louisville, J.B. Speed School of
Engineering</p>
* @author Joseph H. Brown
* @version 1.0 Revision 1
*/

public class Test {
  public Test() {  }

  public static void main(String[] args) throws IOException {
    Test test = new Test();
    String os = "";

    //Test for java security hash methods
    try {
      MessageDigest md5 = MessageDigest.getInstance("MD5");
    }
    catch (NoSuchAlgorithmException a) {
      System.out.println("NOMD5");
    }
    try {
      MessageDigest sha1 = MessageDigest.getInstance("SHA-1");
    }
    catch (NoSuchAlgorithmException a) {
      System.out.println("NOSHA1");
    }

    //Test to make sure the OS is supported
    os = System.getProperty("os.name");
    if ( (os.indexOf("Windows") < 0) && (os.indexOf("Linux") < 0) &&
(os.indexOf("CYGWIN") < 0) ) {
      System.out.println("UNKNOWNOS");
    }
  }
}
```

VITA

NAME: Joseph H. Brown

ADDRESS: Department of Computer Engineering and Computer Science
University of Louisville
Louisville, KY 40292

DOB: Louisville, KY – January 29, 1980

EDUCATION &
TRAINING: B.S. Computer Engineering and Computer Science
University of Louisville
1998 - 2002

AWARDS: Presidential Scholarship, The Arthur M. Riehl Award, Academic Dean's List

PROFESSIONAL SOCIETIES:

- National Chapter of Association of Computing Machinery
- Student Chapter of Association of Computing Machinery
  - Linux Users Special Interest Group
  - Mac Users Special Interest Group
- Upsilon Pi Epsilon Computer Engineering Honor Society
- Tau Beta Pi Engineering Honor Society
- Institute of Electrical and Electronic Engineers
- Institute of Electrical and Electronic Engineers Computer
- Order of the Engineer
- Student Chapter of Technology Network of Greater Louisville