Information Centric Security: Innovative Protections to Mitigate the Insider Threat

Final Report

Office of Naval Research
Small Business Technology Transfer (STTR) Program
Topic# N03-T008
Contract # N00014-03-M-0341
Phase I Base Period Covered: July 2003 – February 2004
Phase I 3 Month Option Available

Principal Investigator:
Dr. Herbert H. Thompson, Director of Security Technology
Security Innovation
1318 South Babcock Street, Melbourne, Florida 32901
Tel: 321-308-0557 x113 Fax: 321-308-0552

Government Technical Monitor:
Ralph Wachter, Office of Naval Research

Corporate Official
Fred Orlando, Chief Operating Officer
Security Innovation
1318 South Babcock Street, Melbourne, Florida 32901
Tel: 321-308-0557 x113 Fax: 321-308-0552

Research Institution PI:
Dr. James A. Whittaker, Professor, Computer Science
Florida Institute of Technology
150 W. University Boulevard Melbourne, Florida 32901
Tel: 321-674-7638 Fax: 321-674-7046

Security Innovation is in the process of being audited to enter into its first cost type government contract with DARPA.

DCAA Auditor:
Joan Bristow
Tel: 321-752-2425
Email: joan.bristow@dcaa.mil

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
02 FEB 2004	**Final**	**01 Jul 2003 - 02 Feb 2004**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Information Centric Security: Innovative Protections to Mitigate the Insider Threat	**N00014-03-M-0341**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
Herbert H. Thompson, James A. Whittaker	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Security Innovation, 1318 S. Babcock St., Melbourne, FL 32901 Florida Institute of Technology, 150 West University Blvd., Melbourne, FL 32901	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
Office of Naval Research ATTN Ralph Wachter Ballston Tower One 800 North Quincy Street, Arlington, VA 22217-5660	**ONR**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) **0001AC**

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT
Security Innovation has been working in partnership with the Florida Institute of Technology to produce designs for tools and technology which will serve to protect sensitive electronic documents from those attackers that operate inside trusted network boundaries. Our focus has been to understand what computing resources and components are used in attacking documents and instrument those resources to log, identify and prevent malicious behavior dynamically. Our overall design protects sensitive documents at three critical times: while on disk, during transmission, and during use. While on disk and during transmission our design augments static cryptographic protections by introducing file locking: the ability to restrict access to documents statically, making cryptographic attacks measurably more difficult by denying access to the encrypted document. The major contribution of this work however is to protect documents when they are most vulnerable: during use. Controls have been designed to protect sensitive documents from attack while their data is being read, edited or executed.

15. SUBJECT TERMS
insider threat, software security, information protection, cryptography, document control

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**UU**	**21**	

Abstract

Security Innovation has been working in partnership with the Florida Institute of Technology to produce designs for tools and technology which will serve to protect sensitive electronic documents from those attackers that operate inside trusted network boundaries. All insider attacks must use local resources (operating system components, installed applications, network appliances and so forth) in order to carry out their purpose. If an attacker is stealing information, they must access files using the operating system; if they want to destroy information, they must run deletion and clean-up utilities. Our focus is therefore to understand what computing resources and components are used in attacking documents and instrument those resources to log, identify and prevent malicious behavior dynamically.

Another point to consider is that information about the activities of rogue insiders is not generated in any existing system log on any major operating system platform. Thus, to monitor and arrest these actions at the host level, we have designed and prototyped a file system filter driver (on the Windows platform) which intercepts all actions on a specific document. Using existing technology, suspicious behavior must be intercepted *before* it is executed on a host computer, otherwise, malicious behaviors will execute without challenge. Our patented technology however will permit suspicious commands to execute so that a more precise determination can be made about the user's intent. Once a sequence of actions has been identified as malicious, we can automatically undo those actions on the machine. This ability to undo is made possible by our precise and patented monitoring technology. Using this mechanism, coupled with process monitoring, we can cripple the ability of a malevolent insider to access protected documents both statically and during use.

Our overall design protects sensitive documents at three critical times: while on disk, during transmission, and during use. While on disk and during transmission there are cryptographic implementations that are provably and measurably secure. For these two conditions our design augments static protections by introducing *File Locking*: the ability to restrict access to documents statically, making cryptographic attacks measurably more difficult by denying access to the encrypted document. The major contribution of this work however is to protect documents when they are most vulnerable: during use. Controls will be introduced to protect sensitive documents from attack while their data is being read, edited or executed. In the next section we summarize the overall problem and our proposed solutions.

1.0 Problem Definition and Proposed Solution

Military, public, and private organizations all have the need to keep sensitive digital information secret. It would be fairly simple if there was but a single classification of secret information and the world could be evenly divided into two groups: those allowed access to the secret and those forbidden from it. For a long time, this was the flawed paradigm that drove information security technology. Most security research and available tools have worked on this paradigm: there are users we trust and those we do not trust. If we can defend against the ones that we don't trust we are safe. The problem is that trust can be exploited: enter the insider threat. With the coalition nature of modern warfare, the military is in desperate need of technological controls to extend mitigated trust to would-be allies. A recent CSI/FBI survey estimates that 70% of losses are from insider attacks as opposed to external hackers. The purpose of this project is to design (Phase I) and build (Phase II) innovative protection mechanisms to reduce the threat of the malicious insider.

Specifically, our work concentrates on the protection of electronic documents from unauthorized access and manipulation. Here we are concerned with *an adversary that gains local access to computers or software that store sensitive information*. With this in mind, there are three critical times when a sensitive document must be protected:

1. A document must be protected while at rest (e.g. on disk, not being used.)
2. A document must be protected during transmission (e.g. an email being sent)
3. A document must be protected during use (e.g. protection from clipboard, stealing contents out of memory, etc.)

Items 1 and 2 both have a well researched, understood and applied solution: cryptography. As an industry, we understand how to encrypt a document such that it is measurably difficult to decipher. The Advanced Encryption Standard (AES) which is now being widely deployed throughout the government has been shown to be an effective implementation.

An enormous amount of money and time has been spent on both developing cryptographic solutions and measuring their effectiveness for static protection and secure communication. In these situations the primary tenants of cryptography hold, meaning that the attacker: has access to the encrypted data (cyphertext); is aware of the algorithm used to create the cipher text; and does not have access to the encryption key. Under these conditions, certain implementations of cryptography are provably strong.

In the third instance, when the protected document is in use, we rely on something that is far from provably resistant to attack: software. Documents – be they data files, applications or algorithms – must be protected while in use. The half-hearted response to this need by the software industry has been anti-debugging technology. Application vendors have relied on crude methods to stop an attacker from inspecting the execution of an application, but all commercially available anti-debugging methods have been defeated. It is clear that new solutions are needed to protect sensitive data while in use.

While an application is running it faces threats from its environment. We can conceptualize a running application as shown in Figure 1.

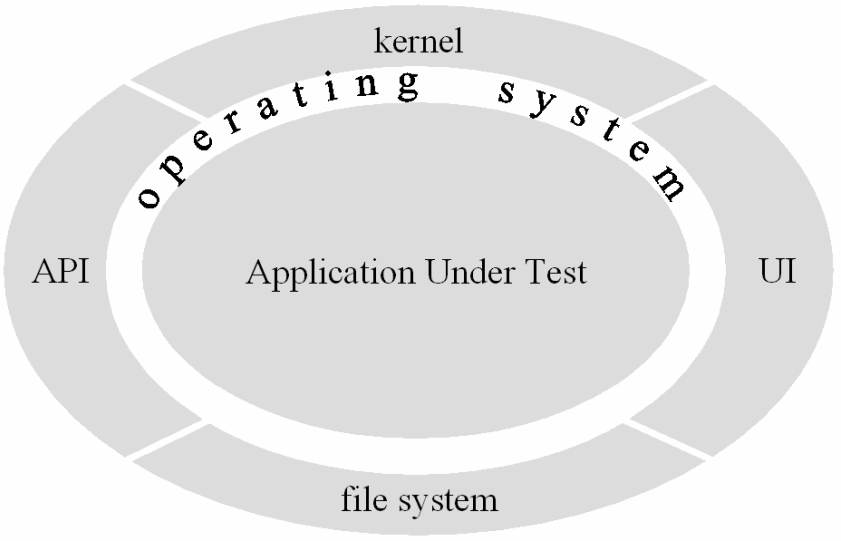

Figure 1 - An Abstraction of an Application's Interaction with its Environment

This fault model describes the extent of software security concerns during execution because such concerns generally relate to the software causing insecure side-effect behaviors that are exploitable via components in the software's environment. The security concerns for each of these interfaces are presented below.

1.1 Threats from the User Interface

User input consists of inputs that originate from a user interface (graphical, command line or menu-driven). Security concerns from the malicious insider through these interfaces include unauthorized access and sabotage.

Our first consideration for the user interface is access control, which is usually implemented via password protection and user authentication. The threat here is from the spy who has access to an application that can decrypt protected files.

Once a user is authenticated, there still may be access controls to consider. Indeed, rarely are all users treated the same. Some users have access to more data and functionality than others and the controls that implement this access must be effective.

1.1.1 Proposed Solutions

Our proposal is to use detailed monitoring of user actions to determine the threat and risk of those actions. Our proposed solution uses our proprietary and patented behavioral monitoring technology primarily embodied in the Hostile Environment Application Tester (HEAT). HEAT (and its companion technology and tool Holodeck) will allow us to record user actions at the lowest level and identify malicious actions through the user interface. We get between a user and a particular application and are able to both intercept and control the signals passed to the application that are responsible for manipulating sensitive documents. One benefit of this monitoring is that we have detailed logs for forensics and prosecution. Perhaps the greatest benefit though is our ability to undo actions that a user has performed once they are identified as malicious. We can, therefore, allow a malicious user to play his or her hand and it will appear as if the changes they have made are permanent. These actions can then be "rolled back" to undo any tampering performed. More detail on our monitoring techniques is included in Section 2 of this report.

1.2 Threats from the File System User

It is often the case that the file system is entrusted to store sensitive data, passwords, and other such persistent information unencrypted. Imagine an application that stores information about itself in the Windows Registry or some other form of central data store. If a malicious user figures out that, for example, license information or encryption keys are stored in the registry then the entire application is compromised. Another point of concern are temporary files that may expose sensitive information contained in documents.

1.2.1 Proposed Solutions

We must be able to control the way in which this data is stored, retrieved, encrypted and managed for security. Using our HEAT technology, we are able to retrofit existing document editors (such as Microsoft Word) to ensure that any information exposed to the file system (including the registry) is strongly encrypted. Our interception technology can therefore ensure that temporary files, registry keys and other vectors of information leakage are protected. Given the generality of our interception technology, no proprietary document manipulation software – such as other editors – are needed: we can retrofit any COTS editor for the Windows or Linux platforms.

1.3 Threats from the Operating System

Any information that an application uses must pass through memory at one time or another. Information that passes through memory in an encrypted form is generally safe, but if it is decrypted and stored even momentarily in memory then it is at risk of being read by insiders with console access. Encryption keys, CD keys, passwords, document controls and other sensitive information must eventually be used in an unencrypted form and its exposure in memory needs to be protected.

Sometimes it is the software itself that must be protected. Many applications have proprietary algorithms or optimizations that give them a strategic advantage over competitors or hostile nations and these secrets need to remain secret.

1.3.1 Proposed Solutions

To protect against sensitive data exposure in memory we must ensure that the data is unattainable by an attacker. Here we take a three pronged approach. First, using our kernel mode interception technology we can dynamically encrypt the contents of memory and decrypt it on access. Initial testing indicates that our current implementation causes a 10% - 15% performance hit on the fortified application. This is likely to not be noticed by the user, but the impact can be further mitigated by identifying areas in memory that are likely to contain sensitive data and restricting the process to those areas. This scheme relies on our second and third defenses: anti-emulation and anti-debugging. Our I^2 (Instruction Interception) technology represents the cutting edge in the field of dynamic application inspection and protection. This technology is currently licensed to several government agencies including the NSA (references available upon request). Should Phase II be awarded we will integrate this capability into our solution. Again, this technology can be used to retrofit existing COTS document editors.

1.4 Threats from other Software

Many applications rely heavily on other software and operating system resources to perform their required functions. Thus, our application is only as secure as the other software that it uses. The attack surface of an application thus includes all external components that our software makes use of. Any interactions that occur along these interfaces are potential entry points for an attacker. Another security concern when dealing with component software environments are the dependencies that naturally exist between software components. Mutual reliance is necessary but mutual trust should not be taken

for granted. Software security is thus a weakest link problem in that when breaches occur in any component, overall application security is likely compromised.

1.4.1 Proposed Solutions

Many COTS applications load components that are rarely used by the average consumer. These components represent additional entry points into an application and thus expose additional routes of attack. Take Microsoft Word for example. The average consumer does not employ the use of macros, a feature that allows a document to execute instructions. The same could be said of Word's networking features such as the Help option "Office on the Web". While these features are rarely used, they are loaded and made available every time Microsoft Word is executed and they represent additional attack vectors to an editor that may be used to read or manipulate sensitive documents. To mitigate these concerns we propose *attack surface area reduction*. Using our patented interception technology we can "turn off" unwanted features in COTS applications that open attack vectors. For example, we can block the protected application from loading the libraries and controls that perform these unwanted and risky actions. We have had marked success in implementing this for several COTS applications and believe that this technology will be an integral part of our solution. In addition to attack surface area reduction, we plan to use both monitoring and profiling to identify and arrest malicious actions being performed by other applications or components. The "undo" ability described in Section 1.1.1 will also be used here to allow us to gather more precise signatures of an attack, identify it as malicious, and then undo the actions it performed.

2.0 Completed Work during Phase I

Our first step in Phase I was to identify the attack vectors that exist against protected documents. Through this research we have determined a set of attack vectors against documents by insiders that have been iteratively refined. We have also completed the design and initial implementation of modifications to our existing behavioral interception technology to both capture, log and analyze the activities of a user on a document or set of documents. This monitoring technology will be central to our "in use" document protection to be implemented in Phase II. Finally, in this section we will outline the design of our dynamic file locking mechanism to augment static document protection.

2.1 Attack Vector Research

Our initial research focused on the attack vectors for "documents" on the system, where a document is any file that contains modifiable data such as a source code file, text file, image or binary. To identify malicious behavior, we require more than just monitoring

the actions of individuals on specific documents. To distinguish between legitimate actions on documents and malicious ones, we must consider the document's environment as well as actions on groups of documents. A sampling of the vectors we have uncovered with respect to the categories outlined in the proposal are as follows.

2.1.1 The Document as a Unit

We must observe properties of individual documents as they are accessed. Some of the initial attack vectors we have identified are:

a). File tampering of source code and documents.

There needs to be some controllable mechanism that prevents files from being tampered with by any unauthorized user. This could include a policy that restricts the modification of documents to particular applications. Tampering with documents might not cause any noticeable change, but simply involve the introduction of viruses (i.e. macros). Protection may include preventing certain documents from being modified even by authorized users (e.g. the policy could roam with the document instead of being enforced by the OS file system).

b). Distribution of documents outside of the organization

A mechanism needs to be designed that can protect documents from easily being distributed to other users outside of the organization. This protection should potentially be a part of the files themselves so that the files can only be utilized on authorized systems.

2.1.2 The Document as a Member of a Group

Properties of groups of documents must also be examined. The reasoning here is that actions against a single document may seem normal during some attacks. When the document is seen as a member of a group however, actions on the group can be recognized as malicious. For example, consider the act of taking a document from a network share and saving it to the local machine. This action may not be flagged as suspicious. Now consider the act of downloading the entire directory structure from a network share. This action is more likely to be flagged as suspicious, but if we were only looking at characteristics of individual documents we would likely miss this behavior. Some specific vectors are discussed below:

a). Capturing of Intranet sites and structure

The act of saving and caching a single page from the organization's intranet is a routine occurrence. Many internet browsers perform this action automatically and thus it represents a legitimate, common and benign activity. We have found, however, that the caching of large amounts of intranet web pages is very highly correlated with malicious actions such as distributing proprietary company data and espionage. For this reason, actions against groups of intranet web pages must be monitored and analyzed.

2.1.3 Actions Inside the Document

There are many actions that take place while a document's data is being read or edited. Take, for instance, the use of the clipboard. For documents that contain sensitive information, we must monitor clipboard actions to ensure that data is not siphoned off by an attacker and then pasted into other documents which may be of a lower classification and thus more easily removed from a system. Also, we wish to monitor the altering of data in source code files, to include, for example Easter eggs or back doors to applications that may be under development at the organization.

a). Buffer Overflows.

The document editor[1] and any plug-ins that are allowed to work with it should be protected from buffer overflows that could potentially allow system compromise. A buffer overflow could potentially allow the execution of an attacker's code.

b). Use of sensitive clipboard information

Copying and pasting is a frequent activity for users. The copying of information onto the clipboard from an editor reading a protected document should be monitored and protected. Pasting would only be allowed to authorized applications and such applications would also need to be protected if they then contained the data from the clipboard. This would require some type of dynamic protection that extends not only to the editor but to every other application on the machine that could allow a paste operation.

c). Crashes and exceptions.

[1] Note that here we use the term "document editor" to refer to any application or process which is used to read, edit, access or process any data contained within a sensitive document.

When the document editor or a plug-in misbehaves and causes a crash or exception to occur, the document editor should not produce any memory dump information or allow debugging that could expose sensitive information. Optionally, memory dump information could be allowed, but only by an administrator or in an encrypted form that only an administrative user would be capable of reading.

d). Accidental or malicious deletion or modification of source code and/or documents.

A disgruntled employee or other malicious user on the internal network that has access to source code or other important documents could easily modify (deface) or delete documents. This includes the potential to insert Easter Eggs and other routines in source code that may ship with a vendor's or agency's product without their knowledge. A mechanism for tracking document changes and restoring modified or deleted documents to their original state needs to be placed on the system.

e). "Un-trusted" document editor plug-ins

Plug-ins need to be detected by the protection system and monitored just as closely as the document editor itself. A policy of trusted plug-ins could be developed so that un-trusted plug-ins or add-on applications would not be allowed to execute without the permission of the administrator.

2.1.4 Environmental Properties

In monitoring document use we must also monitor the environment of the system − the processes and applications that access the document. Screen captures are a good example. When the screen is captured in many operating environments, the operating system invokes functions that are not part of the document viewer's process. If we were to only monitor the viewing process, we would miss an important avenue of information theft. In addition we must monitor network interactions of any applications used to view the documents. Any meaningful set of malicious insider activity signatures must take environmental properties into account. Some specific classes of attack vectors are described below:

a). Unauthorized execution or modification of document editor binaries and related DLLs/APIs.

Only authorized users should be allowed to execute the document editor and open protected documents. The modification of document editor binaries should be prevented by even authorized users without the permission of an administrator. This protection would prevent malicious insiders and other software from modifying the document editor binaries even if the individual had console access to the system.

b). Memory reads from "un-trusted" applications.

Applications, other than those specified by the administrator, should be prevented from gaining any access to the memory of the document editor accessing a protected document. This would also prevent espionage software and viruses from reading sensitive information from memory (e.g. during document editor execution when data is being modified).

c). Modification of configuration information by "un-trusted" applications.

A resource policy should be enforced that prevents modification of registry information and editor-specific file configuration data by unauthorized applications even if they are being performed under an authorized user account. This includes the "registry editor" itself. Only an administrator should be allowed to modify the configuration information.

d). Key Loggers

Key loggers installed accidentally either through communication with other users or via email pose a serious risk. Many professionally written key loggers will not be visible to the user and could remain in the system indefinitely without detection. The protection of documents must then extend to either detecting such key loggers, and/or preventing them from obtaining key-stroke data.

e). Screen capturing software

Like key loggers, screen capturing software poses similar risks. The software could potentially exist on the system without the knowledge of the user and allow screenshots of prototype software or sensitive documents to be uploaded to external machines. The protection of document editors should either detect the presence of this software, and/or prevent it from obtaining a screen capture of the editor and other related applications.

f). User-level and kernel mode interception software

User-level and more importantly, kernel-level interception or hooking software could potentially intercept memory function calls, file reads, registry data, and other information passed via parameters to API functions. This capability would allow these hooking utilities to obtain sensitive information. The protection system needs to be able to detect when APIs are being hooked by either user-level or kernel-level software and prevent it from occurring.

g). Virtual machines

A virtual machine could potentially be used to run the entire operating system in an unprotected environment. The protection system needs to be able to detect the presence of a virtual machine and halt the execution and access of the document editor and its sensitive data.

The above attack vectors were used as an initial basis for the design of the protection technologies discussed in sections 2.2 and 2.3.

2.2 Monitoring

To ensure that we can intercept all file accesses on a system we must create a file system filter driver that operates in kernel mode. We have constructed a prototype for the Windows platform and tested by an independent group. Our results so far indicate that we can indeed intercept all file system accesses in a way that is both transparent to the user and cannot be disabled by an unauthorized user once the system is running. There are several issues to still consider. The first is to make the final selections of behaviors to monitor that may individually or in context indicate an attack. This research area is being fueled by our attack vector analysis. The second issue is that of invisibility. For military implementations of this technology it may be desirable for our logging agents to be "cloaked" to the level of not appearing to be a running process in standard Windows views. The third issue is performance. Our goal is to create a monitoring solution that has negligible performance impact on system file accesses. Our initial benchmarks have been favorable and we are continuing to optimize the interception code. All three of these areas will be addressed in the prototype to be constructed in Phase II.

We have also made significant strides in our application monitoring technology. It is essential that we be able to both monitor and control interactions between an application and its environment (reference Figure 1). Figure 2 shows the implementation of this technology in our Holodeck application monitoring and testing tool. The technology shown here will be integrated into our protection solution in Phase II.

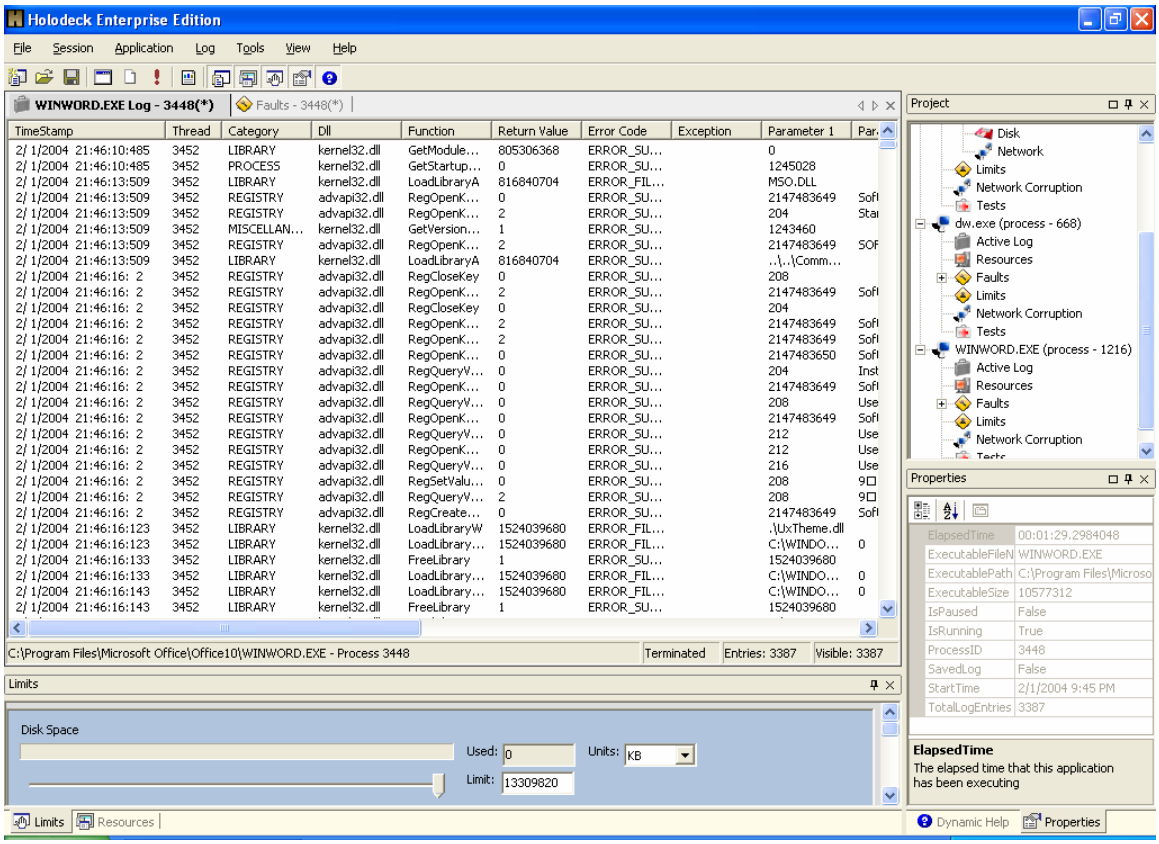

Figure 2 – Security Innovation's application monitoring and control technology is built into our Holodeck product.

2.3 *Dynamic File Locking Mechanism*

Our file locking protection capability is meant to be used to augment the static protection cryptography provides to sensitive documents. Sensitive documents will be strongly encrypted and the technology discussed here is meant to restrict access to these files and associate them in a binding way with authorized editors. Once protection is enabled for a file, the protection will remain regardless of whether the application using it is a trusted or un-trusted application. In our current design and prototype implementation, the process for adding a binary file is as follows:

1. Determine the binary file that needs protection.
2. Ensure that the File Encryption Service is running.
3. Open the UI.
4. Drag the file of interest into the list pane of the UI.
5. Add the file as a binary file.

A UI interface will summarize this information allowing the user/administrator to check/uncheck the files that should be protected. CRC tracking and

information will be integrated into this system in Phase II in order that files added from one machine may automatically be detected on another machine so that the administrator does not have to go to each individual machine to setup the protection of applications. This is ideal for the environment where homogenous configurations are used.

We currently have a prototype CRC implementation, but it has not been integrated into the current design. This integration is slated for Phase II.

2.3.1 Prototype UI Design

In the final product implementation, the UI features will be integrated with the Explorer shell so that right clicking on a file will reveal protected service options. These will include 1) Add as protected file, 2) Add as protected binary, and 3) Add as trusted application. In addition, a system tray icon will provide status information such as server status and protection system warnings. Currently, our prototype of the Secure Integrated Document Engine (SIDE) operates as a GUI which interacts with our running file encryption service.

2.3.1.1 Adding Files to the Protected File System

To add files, our current implementation allows you to drag the file from an Explorer window to the UI and a dialog will appear requesting information:

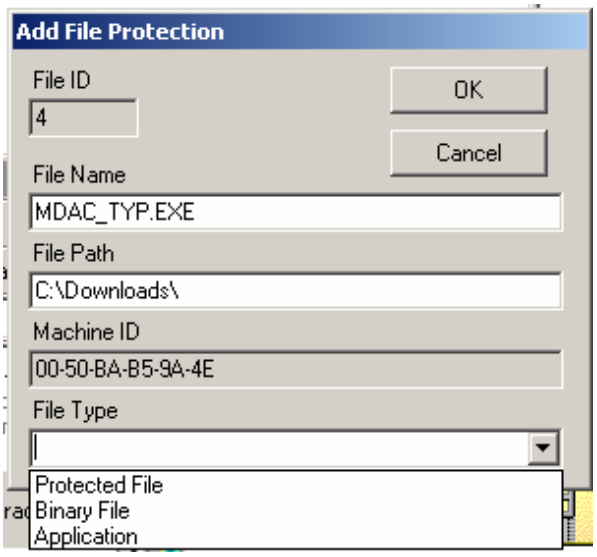

Figure 3: A file is added through the UI to be protected.

The file can be added as a protected file, binary file, or application. The UI itself places these files into separate bins as shown in Figure 4.

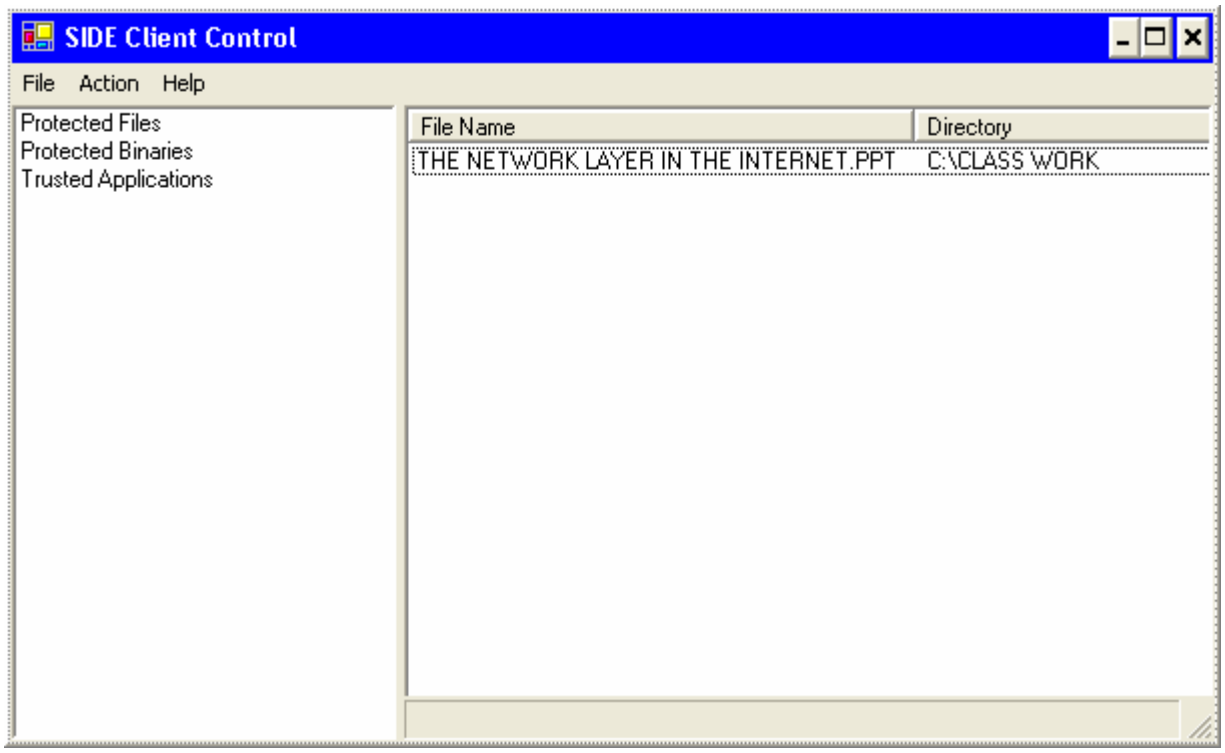

Figure 4: The interface to our File Locking mechanism shows files and editors that are under its protection.

Navigating to each bin in the left hand pane will update the list on the right with files contained in those areas. In the above example, a file has already been added to the protected files bin. If we try to open this file, we will get an error. For example, if we attempt to open our protected file in Microsoft Notepad the error shown in Figure 5 is generated.

Figure 5: When we attempt to open one of our protected files in an untrusted editor – in this case Microsoft Notepad – we get an error.

This means that this file is inaccessible. *Any* application will respond in a similar manner when this file is attempted to be opened. In order to open the file with Notepad (even though it is not a Notepad file), we must add Notepad as a trusted application and associate it with the file. This is done by dragging the Notepad executable to the UI and adding it as an Application (Figure 6).

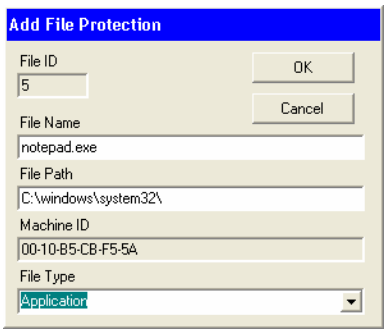

Figure 6: We can now add Notepad as a Trusted Application

Now, looking in the Trusted Applications bin, notepad will appear (Figure 7).

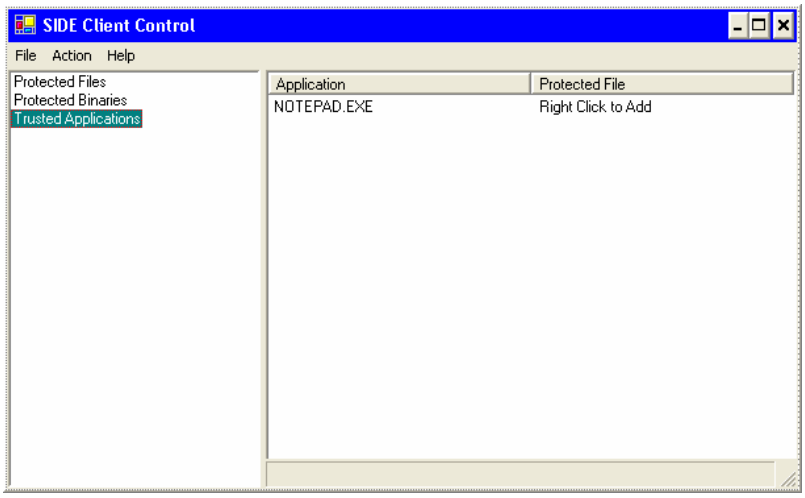

Figure 7: The File Locking mechanism then displays Notepad as a Trusted Application

Trusted applications have their own pools of protected files they are allowed to access. For Notepad, we must define what protected files it will be allowed access to. To do this, we create an association for Notepad to the file we just added. This is done using right-click menus on the entry in the right pane (Figure 8).

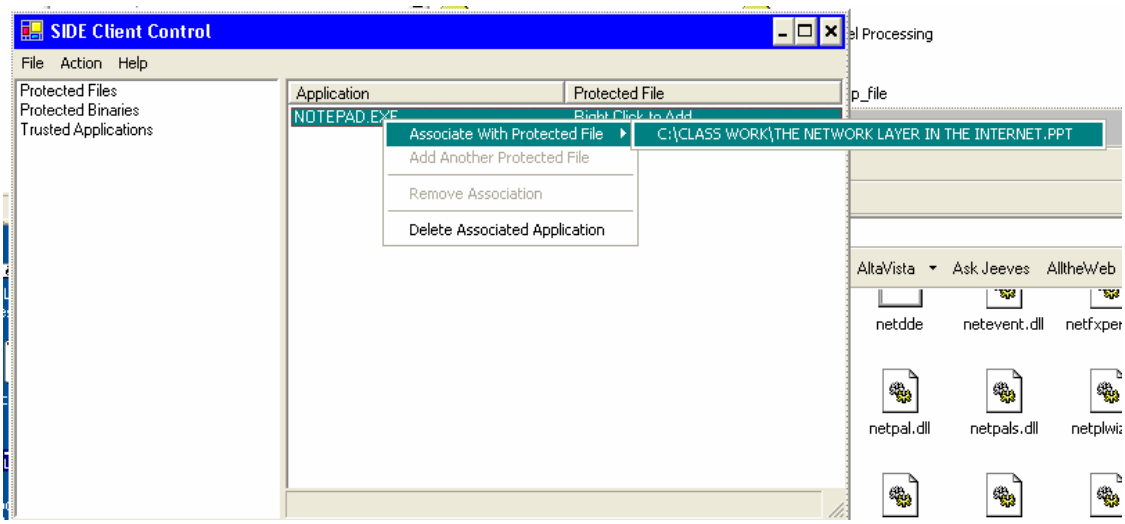

Figure 8: One of the key features of our solution is the ability to bind a protected file to a particular trusted application.

Associating Notepad with the file will now allow Notepad to open the file. We can also delete the Notepad application as a trusted application using the "Delete Associated Application" option. Once an application has been associated with a protected file, it will appear in the list (Figure 9).

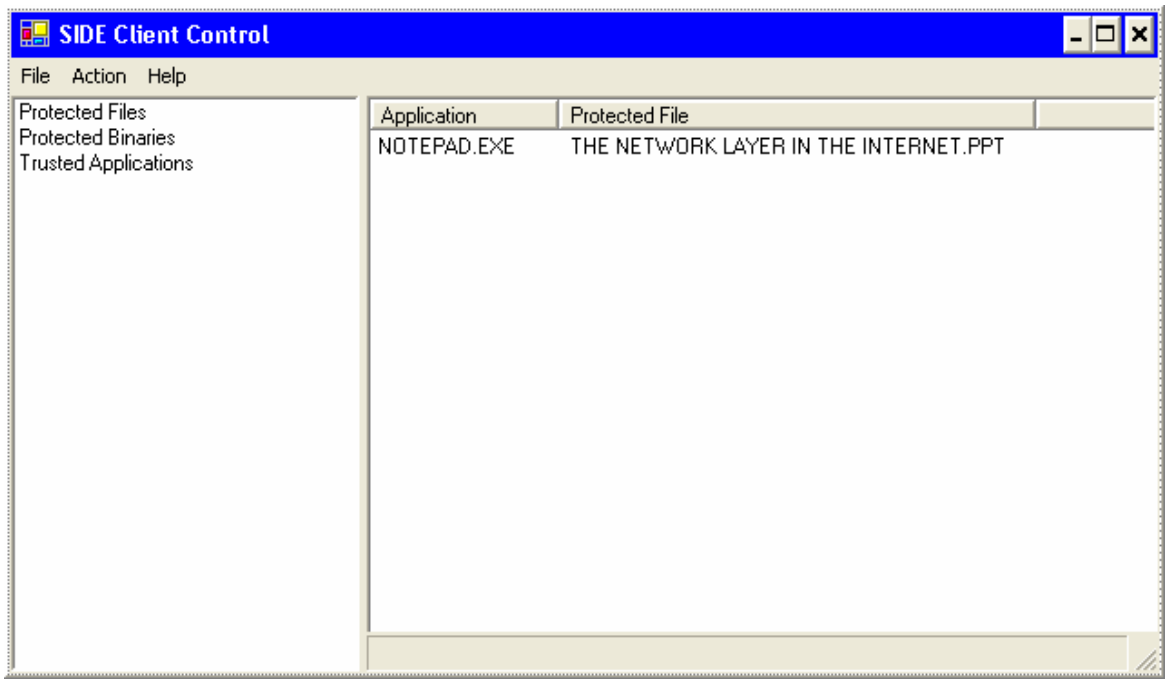

Figure 9: Now our protected file can only be opened with Notepad

Multiple additional protected files may also be associated with that application using the right-click menu. If we want to remove the association, we just select that option in the right click menu (Figure 10).

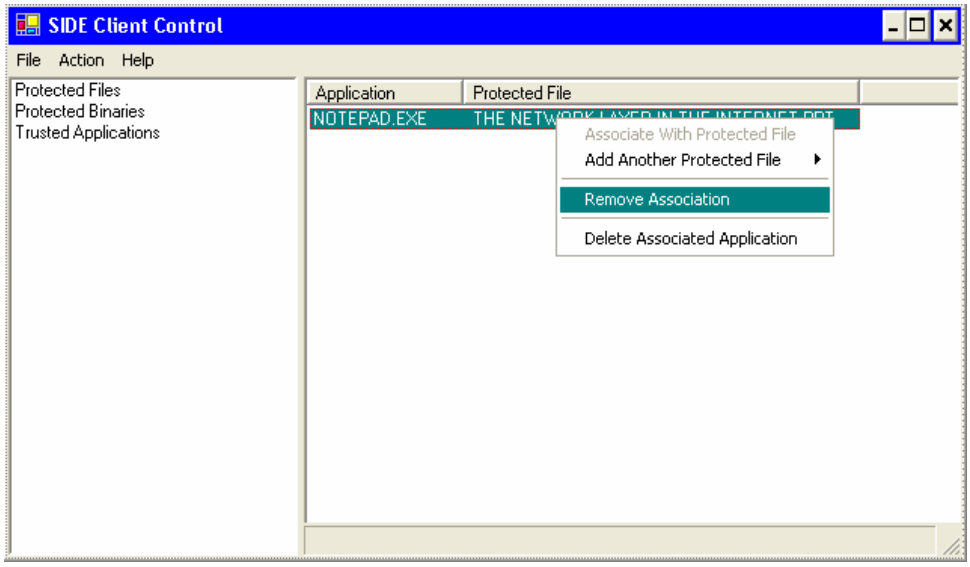

Figure 10: An authorized administrator can easily remove such associations

Protected binaries behave differently than protected files. Once a protected file has been added, it cannot be renamed, deleted, written to, or read from. However, a protected binary can still be read but not modified in any way. To see this in action, we will add a text file as a *protected binary* (Figure 11).

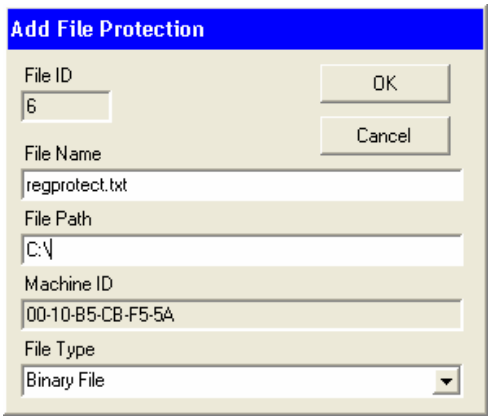

Figure 11: Any file can also be added as a protected binary

This file is now added to the binary file list (Figure 12).

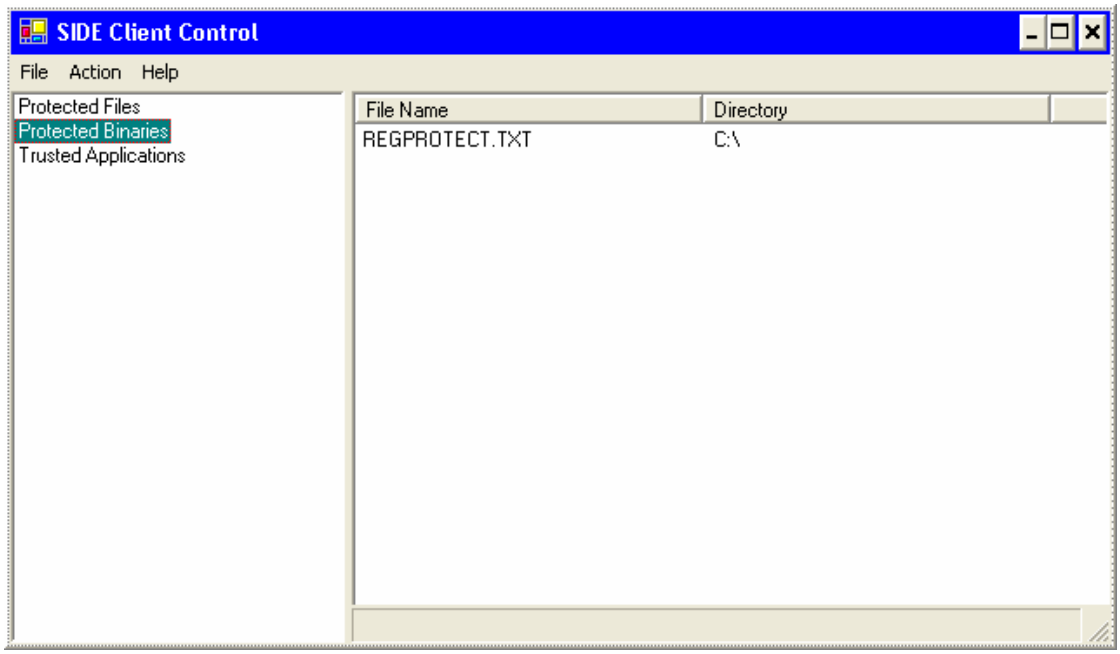

Figure 12: We see our target file added as a protected binary

If we try to open this file using Notepad, it will succeed, but, attempting to save to the protected file will not succeed (Figure 13).

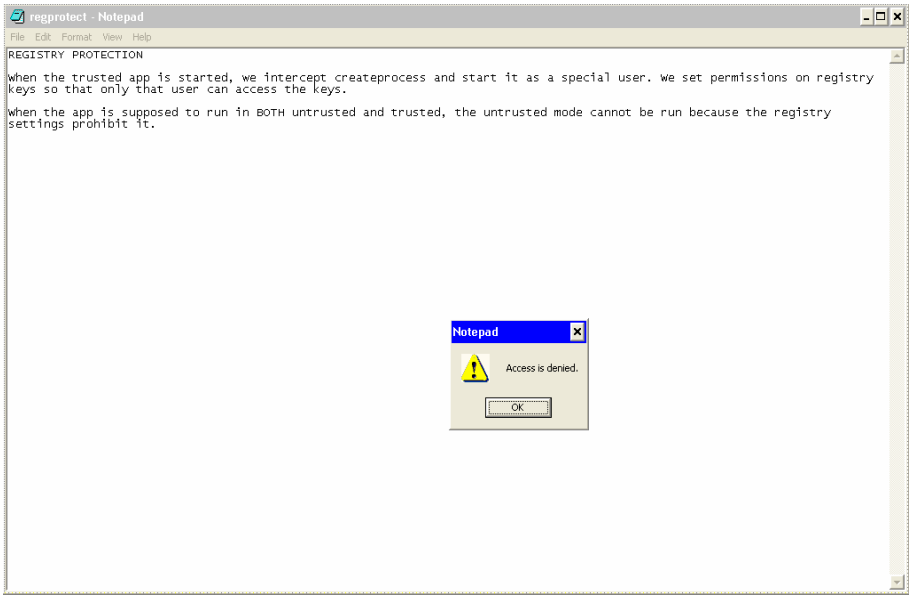

Figure 13: Protected binaries can be viewed but not altered

We also cannot rename the file or delete it (Figure 14).

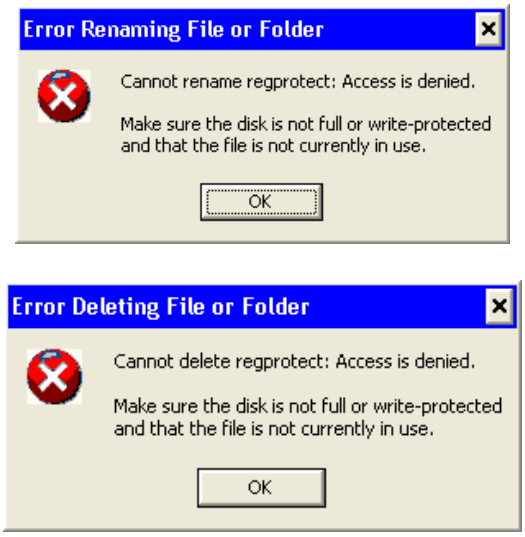

Figure 14: An error message results if we try to either delete or rename a protected binary

During Phase II this technology will be integrated into our overall document protection solution.

2.4 Option Proposal [3 Month] – Design of a test-bed and related metrics to determine the effectiveness of the Phase II implementation

It is not useful to create a new defense against the insider threat unless some claim can be made as to its effectiveness. For the 3 month option we have proposed to design a test-bed and benchmarks to be implemented in Phase II to determine the effectiveness of our defensive solution at stopping insider attacks and measure other key attributes such as cost of implementation and performance impact. Previous experience indicates that metrics from the fields of software complexity and reliability may be meaningful in this context. Where necessary, we will design novel measures to accurately reflect the effectiveness of a given defense. Security Innovation, Florida Tech and the Principal Investigators have been involved with software security metrics for some time and are currently funded by the Air Force Research Lab (AFRL grant # F33615-02-C-1299) to develop measures of security for software. The measures and benchmarks developed under this option will allow not only our solution to be evaluated but will serve as a tool to evaluate a broad range of security defenses and controls.

At the end of the three month period, we will deliver a whitepaper detailing the design of a test-bed to measure the effectiveness of defensive mechanisms to mitigate the insider threat. The whitepaper will also discuss what metrics will be used and the nature of the experiments – possibly involving so called hacking "red teams" – to validate these metrics.

3.0 Project Summary

This final report has presented the design of a document protection solution to mitigate the insider threat. If funded, the design presented here will be implemented in Phase II. The three month option described above will serve to create effectiveness measures of the Phase II implementation.

Below is a list of the major project milestones along with anticipated completion dates. The following chart covers activities in the base (6 month) contract period and the optional extension (3 months shown in grey). The progress of each of these is listed in percentages:

Status	Delivery Date	Item Description
100% complete	09-01-2003	Analysis of attack vectors by malicious insiders to sensitive electronic documents.
100% complete	10-01-2003	Survey existing protection mechanisms for electronic documents against the insider threat.
100% complete	11-01-2003	Determine where gaps exist in current protection technologies for sensitive documents against malicious insiders.
100% complete	12-01-2003	Design protection mechanisms for electronic documents while on disk
100% complete	01-01-2004	Design a protection architecture for documents while in-use
100% complete	01-01-2004	Develop a design for a protection infrastructure which protects electronic documents both statically and while in-use.
Not Started	04-01-2004	3 Month Option - Design a test bed, a red-team plan and metrics to determine the effectiveness of document protection mechanisms against malicious insiders.