APPLICATIONS OF GUI USAGE ANALYSIS

Except where reference is made to the work of others, the work described in this dissertation is my own or was done in collaboration with my advisory committee. This dissertation does not include proprietary or classified information.

_____

Eric Shaun Imsand

Certificate of Approval:

_____	_____
Cheryl D. Seals Associate Professor Computer Science and Software Engineering	John A. Hamilton, Jr., Chair Assistant Professor Computer Science and Software Engineering
_____	_____
David A. Umphress Associate Professor Computer Science and Software Engineering	Joe F. Pittman Interim Dean Graduate School

APPLICATIONS OF GUI USAGE ANALYSIS

Eric Shaun Imsand

A Dissertation

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Doctor of Philosophy

Auburn, Alabama
May 10, 2008

APPLICATIONS OF GUI USAGE ANALYSIS

Eric Shaun Imsand

_____
Signature of Author

_____
Date of Graduation

DISSERTATION ABSTRACT

APPLICATIONS OF GUI USAGE ANALYSIS

Eric Shaun Imsand

Doctor of Philosophy, May 10, 2008
(M.S. Auburn University, 2003)
(B.S. Auburn University, 2002)

141 Typed Pages

Directed by John A Hamilton, Jr.

In the realm of computer security, a masquerade attack is a form of attack wherein the attacker deceives the victim, causing them to believe they are someone other than who they are. One particularly dangerous form of masquerade attack occurs when an attacker begins using an unattended and unlocked computer workstation. This form of masquerade attack is particularly troubling because it requires no technical expertise to perform. Though proper adherence to organizational security policies can mitigate this risk, new technologies are needed to completely defend against this type of attack.

This dissertation presents the results of a study into the potential suitability of GUI Usage Analysis as an authentication mechanism which can be used as a defense against masquerade attacks. Previous attempts at authenticating the current user of a computer system have focused on typing patterns and mouse movements. GUI Usage

Analysis does not focus on the user's physical interaction with the computer system, but instead on how the user manipulates the windows, icons, menus, and pointers that comprise a graphical user interface.

Results are presented showing the feasibility of employing GUI Usage Analysis as a means of authenticating the user of a computer system. Furthermore, results are also presented demonstrating the effectiveness of using GUI Usage Analysis as a means of identification with the goal of identifying a potential attacker. Finally, the results obtained here are compared to other previously published masquerade detection techniques.

Style manual used	A Manual for Writers of Research Papers, Theses, and Dissertations by Kate Turabian (6th Edition)
Computer software used	Microsoft Office Enterprise 2007, Microsoft Visual Studio 2005, Matlab 7.0

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Most every student that has received formal training in the information security field has been taught that data has three fundamental qualities that describe its overall security: confidentiality, integrity, and availability. If any of these qualities is degraded, then the data in question cannot be considered secure.

Students of information security are also taught to identify vulnerabilities. Generally speaking, vulnerabilities can be grouped into two categories: vulnerabilities that originate outside of an organization and vulnerabilities that originate from within an organization. Much of the collective effort of information security professionals has been directed towards threats from "outsiders", persons not part of the organization. Recent studies, though, have begun to illustrate the considerable damage that can be carried out by insiders (United States Secret Service: National Threat Assessment Center 2004). These findings illustrate the need for the development of additional defensive technologies to protect against insider threats. Please note that the term "insiders" can be used to denote either persons with authorized access to systems/information or persons with physical access to the premises.

It is believed that the results of the studies presented in this dissertation can be utilized by software engineers to develop new technologies to guard against one

particular type of insider attack known as a masquerade attack. It is believed that such a line of products would be of great value to information security professionals, who are currently limited in their ability to detect and prevent insider attacks on confidentiality and integrity.

Masquerade Attacks

Masquerade attacks have been studied in some detail. The general description of a masquerade attack is an instance in which the attacker is able to trick the targeted system into believing that they are someone they are not. In essence, the attacker is impersonating, or masquerading, as another user with legitimate system access. Masquerade attacks are considered dangerous due to the fact that, when successful, the attacker is able to assume the identity of the impersonated user, resulting in the attacker having access to all the resources the impersonated user has access to.

Masquerade attacks can take on many different forms. For example, an attacker that successfully hijacks an authenticated web session is one form of a masquerade attack. In this instance the attacker would appear to the website as the authenticated user. Another form of masquerade attack can occur when an attacker begins using an unattended and unlocked computer workstation. This is the form of masquerade attack investigated by the studies presented in this dissertation.

In the form of masquerade attack considered here, the attacker has physical access to the system and has found it in an unsecured state. In this scenario the system has already authenticated an authorized user and that user has subsequently left the system unattended and unlocked, allowing the attacker to take control of the system without

having to identify or authenticate themselves. In this scenario the system believes the attacker to be the legitimate user, resulting in the attacker being granted access to all of the legitimate users files, e-mail, database tables, etc. As a real-world scenario in which this type of attack might occur, consider an employee who slips into a co-workers office right after they have left for lunch. Even if the targeted employee uses a password on their screensaver, there will still be a window of time in which the system is vulnerable.

As previously mentioned, the overall security of an object is typically measured in three different manners: confidentiality, integrity, and availability. The type of masquerade attack discussed here can affect both the confidentiality and integrity of an object with little chance of being detected. This is because the behaviors exhibited by an attacker seeking to compromise the confidentiality or integrity of an object would closely resemble the normal behavior exhibited by the legitimate user. To illustrate, consider a spreadsheet containing revenue figures on a corporate file server. Suppose that the legitimate user is authorized to view and modify the contents of that file. From the system's point of view, a masquerading attacker will exhibit the same behavior as the known user. The attacker may even modify the file (compromising the integrity of the object) without raising any suspicion that an attack is occurring. Because this is a behavior that the legitimate user regularly engages in, the system will have no way of knowing that the changes being affected have been performed by an attacker.

To detect this type of attack, the system must have some defensive measure that is capable of knowing the actual identity of the user behind the monitor. Furthermore, the identity must be difficult to forge or otherwise defeat. While solutions have been

developed that meet these criterion using hardware devices, software based solutions are also desirable. This dissertation presents several studies that explore the suitability of user interactions with a graphical user interface as a profile to be used in identification and/or authentication.

Description of the Microsoft Windows Graphical User Interface

Though modern computing systems undoubtedly represent vast engineering and scientific achievement, they are ultimately tools; a means to be used in the completion of some greater task. As a result they require some manner of receiving instruction. Typically this instruction comes from human operators who issue commands that are then carried out by the computer. In a continuing effort to make the issuance of these commands more efficient, modern computing environments feature intuitive user interfaces that depict data using pictures instead of plain textual characters, as was the case in the past. The term for these new graphically driven user interfaces is *graphical user interface* and may sometimes be abbreviated to "GUI".

The computing environment used in the studies presented here is Microsoft Windows XP, frequently referred to as simply "Windows". Windows features a graphical user interface which requires users to manipulate objects known as windows, icons, and menus using a tool referred to as a pointer. Thus, the Windows user interface is a sub-category of GUI known as WIMP ("Windows, Icons, Menus, and Pointers"). The vast majority of people, both computer users and non-users, are familiar with WIMP based user interfaces. As of December 2007 some estimates place the market share of Windows XP at nearly eighty percent (Market Share 2007).

Several terms are sometimes used in this dissertation without an accompanying definition. These terms will be defined here.

Table 1. Terminology Used in this Dissertation but not Defined Elsewhere

Term	Definition
Control	An object on which a user takes some action (keystroke, mouse click, double-click, etc.). Examples include text-boxes, buttons, menus, etc.
Handle	An ID number used by the operating system (Windows) to keep track of and refer to a specific control. (Microsoft Corporation 2007)
Event	Strictly speaking, an event is a notification issued by the operating system to indicate to a client application that something specific has occurred. For the purposes of this dissertation, *event* will denote the occurrence of an action performed by the user on some control, such as left mouse button click on a button.
Process	For the purposes of this dissertation, the term *process* describes a computer program that is actively executing (running) on the host computer system.
Message	The term *message* is used to describe a piece of information transmitted from the operating system to a running process, updating the process on the current state of the computing environment

CHAPTER 2

REVIEW OF RELATED WORK

The ideas presented in this dissertation certainly did not occur in a vacuum. Rather, the ideas expressed here are a natural progression of thought built upon other previously published research generated by other authors. The ideas presented here draw from several different separate research sources. These include varying types of behavioral biometrics as well as other system oriented profiling techniques.

The one constant in all of the techniques described here is that they attempt to detect attacks (sometimes referred to as intrusions) by searching for abnormalities in monitored behavior. One of the earliest references to monitoring abnormalities as a means of detecting intrusions was introduced by Denning (Denning 1986). Denning proposed the creation of a network intrusion detection system that monitored the system for abnormal occurrences. The hypothesis espoused by Denning was that abnormal network activity could be strongly indicative of an attack. Many of the modern intrusion/attack detection systems described in this chapter follow a similar architectural style.

Behavioral Biometrics

Biometrics are traits that are generally unique to individual users that can be used as either means of identification or authentication. Biometrics themselves can be further divided into behavioral biometrics and physiological biometrics (Coventry 2005)

(Newbold 2007). Physiological biometrics describes physical characteristics that are used as biometrics. Examples of physiological biometrics include fingerprints, iris and retinal scans, palm prints, etc. Behavioral biometrics describes human behaviors that are generally found to be unique to individual persons, but are not necessarily dependent on any physiological traits. Examples of behavioral biometrics include handwriting, voice analysis, command line profiling, and keystroke dynamics (sometimes known as typing dynamics). It is believed that GUI Usage Analysis is a new form of behavioral biometric. For this reason, only background work concerning behavioral biometrics will be presented in this chapter. Furthermore, only behavioral biometrics that can be analyzed using a computer keyboard and mouse will be discussed. It is believed that while voice analysis, handwriting, and GUI Usage Analysis are all examples of behavioral biometrics, they are still sufficiently different from each other to make their inclusion in this discussion unnecessary.

User/System Interaction Profiling

Several studies have been published that attempt to identify and verify a user's identity based on the current instructions passed to the system. These attempts are somewhat related to GUI Usage Analysis as both techniques utilize passive profiling techniques based on the user's manipulation of the computer system. Because this relationship is somewhat tangential to GUI Usage Analysis, the techniques presented here are not given as much consideration as other profiling technologies detailed in this chapter.

Attempts have been made at profiling users based on the current state of the system. Researchers believe that users tend to utilize computer systems in similar manners on a day to day basis. Because the system is being utilized in a consistent manner from day to day, the underlying components of the system should also be relatively consistent from day to day. Assuming that the same user is operating the computer system in a standard manner, the usage of system resources like processor load and available memory should be relatively constant. Any deviation from these observed norms could represent some sort of attack, either from an intruder having gained unauthorized access to a user's account, or from the user themselves engaging in suspicious behavior.

Such a study was carried out by Li and Manikopoulous (Li and Manikopoulos 2004). Li and Manikopoulous selected key system events such as processor load, process table status, memory consumption, etc., from eight users and utilized them as either legitimate user events or as attacker events. Thirty five separate sessions generated by four of the selected users were used as both training and test data. At the same time, sessions from four additional users were used exclusively as test data. The authors fed the experimental data into a support vector machine classifier. The classification engine correctly detected 63% of attacks and had a false positive rate of 3.7%.

Another approach frequently seen in attempting to detect masquerade attacks is commonly referred to as "command line profiling". Command line profiling attempts to detect masquerading attackers based on the commands issued to the system console. The hypothesis behind command line profiling is very similar to the rationale used for the

system load profiling experiments previously described: users tend to operate computer systems in a consistent manner. For example, an office assistant would commonly be observed issuing console commands to invoke a word processor or spreadsheet application, but not the system command for modifying the system's password file. Such an occurrence would represent a deviation from the observed normal behavior for that user and would most likely indicate an attack of some sort, either from an external attacker or malicious insider.

Multiple studies have been published studying the viability of profiling a user based on the commands entered into a system console (Maxion and Townsend 2002), (Coull, et al. 2003), (Khanna and Liu 2006). Perhaps the most commonly cited work on the subject of command line profiling was presented by a team led by Matthias Schonlau (Schonlau, et al. 2001). Schonlau and his team collected approximately 15,000 commands for each user. A total of fifty users participated in the study. Schonlau then used the collected data and analyzed it using multiple previously published analytical techniques. He concluded that each of the materials could detect intrusions reasonably well, though none of the techniques detected a high enough percentage of attacks to justify its use as the sole means of detecting intrusions.

Keystroke Dynamics

Beginning in the mid 1980s, researchers began to experiment with the possibility that users typing on a keyboard may exhibit patterns in their typing. This general field of research, commonly referred to as "keystroke dynamics" centered on the notion that users had typing patterns that could be used as either the basis for identification or

authentication. Studies frequently focused on the periods of time observed between successive keystrokes. It was generally determined that experienced typists did exhibit patterns in their typing, and that these patterns could be used as a means of authentication. The one caveat was that the user needed to be typing a word or phrase that they frequently typed, such as a password.

One of the initial keystroke dynamics studies was carried out by Umphress and Williams in 1985 (Umphress and Williams 1985). Umphress and Williams performed a study using 17 participants in which each participant completed two typing tests. The first test asked participants to type approximately 1400 characters to serve as the reference profile. The second test asked participants to type approximately 300 characters to use as the test profile. Umphress and Williams analyzed the average latencies occurring between each key press for each user. After comparing the observed latencies in the reference profile to the latencies observed in the test sample, a confidence score of either low, medium, or high was computed. Test samples assigned a value of low confidence were suspected to have originated from a user other than the known reference user. Similarly, test samples assigned a confidence level of "high" were judged to have been generated by the reference user. Umphress and Williams experimental results generated a false positive rate of 12% and a false negative rate of 6%.

Several years after the research published by Umphress and Williams, the team of Joyce and Gupta implemented a follow-up study (Joyce and Gupta 1990). This study was noteworthy because it marked one of the earliest examples of keystroke dynamics research targeting the use of a PIN or password. Using an experimental sample of 33

participants, Joyce and Gupta asked participants to type their user id, password/PIN, first name, and last name. By examining the latencies between consecutive characters, Joyce and Gupta were able to achieve a false negative error rate of 1% and a false positive error rate of 7%.

Following the efforts of Joyce and Gupta, the Italian team of Bergadano, Gunetti, and Picardi published a large scale study designed to verify the utility of keystroke dynamics over dramatically larger samples (Bergadano, Gunetti and Picardi 2002). Bergadano and his team utilized an experimental sample of 154 participants, marking a very large increase over all other prominent keystroke dynamics techniques. The authors of this study were ultimately able to achieve a false positive rate of four percent and a false negative rate of 0.01 percent. The authors of this study utilized a slightly different analytical technique than previous studies to achieve these results. The authors consisted latencies over three character periods rather than the two character period utilized by earlier researchers.

To date, the published research into Keystroke Dynamics has been inconclusive (Peacock, Ke and Wilkerson 2005). Failure to use common data collection and analysis techniques between researchers has made it difficult to make definitive judgments regarding the utility of keystroke dynamics as a means of identification or authentication. The transition of modern computing environments to graphical user interfaces may also have impacted the effectiveness of keystroke dynamics (Garg, et al. 2006), either in their overall usefulness or in terms of the repeatability of keystroke patterns caused by decreased typing.

Despite the apparent conclusion of the academic community regarding the effectiveness of keystroke dynamics, multiple commercial products have been developed utilizing this technique. For example, in 2002 BioPassword Inc. released their first keystroke dynamics based authentication package (Kingsbury 2006). As their name implies, BioPassword Inc. products utilize keystroke dynamics to harden passwords. According to BioPassword, a user is required to not only know a password, but they are required to type it in a manner consistent with an observed reference profile (BioPassword, Inc. 2007). Recently other companies have moved into the marketplace offering competing products (iMagic Software 2007), (ID Control, Inc. 2007), (bioChec 2007). Not surprisingly precise performance data is not readily available for the commercial solutions.

Other GUI Based Authentication Studies

Though the ideas presented in this dissertation are unique, there have been previously published studies that attempt to accomplish the same goals with varying success (Pusara and Brodley 2004), (Garg, et al. 2006). These papers have attempted to differentiate between users based on how users interact with a graphical user interface. This section will describe the work performed by both authors as well as compare and contrast their methodologies. Because only two published studies are considered, this section will differ slightly from the other sections in this chapter, taking advantage of the opportunity to demonstrate direct comparisons between the two studies.

The studies presented here have several key similarities. First, and arguably most importantly, both studies attempted to solve the same problem: verifying the identity of

the user sitting behind the keyboard. Garg and his team referred to this practice as "masquerade detection" while Pusara and Brodley coined the term "user re-authentication". Garg describes a masquerade attack as being an incident in which the operator of a computer is not the rightful owner of the credentials used to access the system. Pusara and Brodley's chosen term, re-authentication, implies a similar problem set: re-verifying that the operator of the computer is the proper owner of the credentials used to access it.

As an aside, Garg's terminology was adopted to describe the research presented in this dissertation. Garg's terminology was chosen over Pusara and Brodley's for two primary reasons. First, an informal literature review seemed to slightly favor the term "masquerade attack" over "re-authentication". Secondly, it is believed that the majority of incidents in which GUI Usage analysis or any similar technique might be applied are best described as attacks.

Both Garg and Pusara and Brodley ultimately focused their research on data gathered from the mouse. Pusara and Brodley appear to have focused on gathering mouse movement data from the beginning, no doubt at least partially inspired by the keystroke dynamics research described earlier in this chapter. Garg, on the other hand, initially captured data from multiple sources including mouse movements as well as keystrokes. As Garg's research progressed he focused exclusively on the movements of the mouse. Garg did not provide a formal explanation for why the other data that was collected was not factored into his final analysis. It is interesting to note that Pusara and

Brodley's work was referenced by Garg, though his findings are not presented as a mere confirmation of Pusara and Brodley's work.

Both authors utilized machine learning techniques to analyze the data that was gathered. Pusara and Brodley utilized a commercial machine learning algorithm known as See5/C5.0 (Rulequest Research 2007). See5 is a form of decision tree learning, a type of machine learning that has been widely studied (Russell and Norvig 2003). Garg selected a different form of machine learning known as Support Vector Machine (SVM) (Vapnik 1995). The specific implementation chosen by Garg is known as SVM-Light, an open source C implementation available free of charge for non-commercial uses (Joachims 2004).

Aside from the difference in machine learning algorithm, the data gathered by both authors was relatively similar. Both examined the distance, angle, and speed observed between consecutive user actions and derived several statistics which were used during the analysis. Both authors calculated the mean and standard deviation of the change between consecutive events. Pusara and Brodley also calculated the third moment observed between consecutive events. Both authors also used a sliding window as part of their analysis to limit the amount of data considered by the machine learning algorithms during any single analytical session.

In spite of all of the similarities found between the studies published by Garg and Pusara and Brodley, key differences can also be observed. The first and most striking difference between the two studies deals with the size of the experimental sample gathered. Pusara and Brodley initially started with a sample of 18 participants, though

that number was eventually decreased to 11. Garg, on the other hand, began with an experimental sample of size three. Garg indicates that no participants were excluded from the analytical data at any time.

The next, and possibly most important, difference between the two studies concerns the activities assigned to the study participants. Garg opted to have the participants utilize their computer systems in an ordinary manner, apparently imposing no limitations on what the participants could and could not do while gathering data. Pusara and Brodley took a different approach and severely limited the activities of the participants in their study. Pusara and Brodley's participants were instructed to read a series of web pages using the Internet Explorer web browser. Pusara and Brodley's participants did not engage in any other activities other than reading the prescribed web pages.

The results of the final experiment presented by Pusara and Brodley were highly comparable to the results of the final experiment presented by Garg. Pusara and Brodley reported a final false negative error rate of 1.75% while Garg and his team reported a false negative error rate of 3.85%. Pusara and Brodley reported a false positive error rate of 0.43% in their final study. Unfortunately Garg and his team did not report a false positive error rate in their findings.

Previous GUI Usage Analysis Research

The experiments presented in this dissertation are not the first attempts at using GUI Usage Analysis as a means of authentication. Prior to conducting the experiments presented here, two additional studies were performed using differing data analysis

techniques and participants. These studies attempted to determine the utility of using GUI Usage Analysis as a means of authentication for an overall participant pool, as well as the impacts that demographical traits might have on the effectiveness of GUI Usage Analysis.

In (Imsand and Hamilton, GUI Usage Analysis for Masquerade Detection 2007), Imsand and Hamilton present the results of an initial study into the viability of using GUI Usage Analysis as a means of detecting masquerade attacks. The authors of this study utilized an experimental sample of ten subjects, all of whom were undergraduate students majoring in either business of computer science. These ten subjects were asked to complete a specific set of tasks three times with a period of two days between data collection sessions. In all experiments two of the three sessions were used as training data, while the third session was used as a test session.

Imsand and Hamilton utilized a compositional analysis method similar to the techniques described at other points in this dissertation. The authors calculated the number of times each pair of consecutively occurring events was found in the reference sample and compared that number to a similarly calculated total obtained from the test sample. The total number of discrepancies between the reference sample and the test sample was then calculated, indicating the amount of dissimilarity observed between the reference sample and the test sample.

The authors exhaustively tested each possible combination of training and test data for each user and reached a tentative conclusion that GUI Usage Analysis could be used as a means of authentication and/or part of a masquerade attack detection scheme.

This tentative conclusion was based on an observed false negative attack rate of 40%. The authors used an "attack threshold" which indicated the maximum amount of dissimilarity that could be observed between two sets of data without signaling an attack. By tuning the attack threshold for each user the false positive rate was zero percent.

The authors of this study made one additional interesting observation regarding this research. It was noted that, in comparison to other GUI based profiling techniques, GUI Usage Analysis required much less data to make a classification decision. It was determined that other GUI based profiling schemes would require up to 72% more data in order to operate as designed.

The results in (Imsand and Hamilton, GUI Usage Analysis for Masquerade Detection 2007) showed that some users experienced much higher rates of masquerade attack detection than some other study participants. An investigation was conducted in an effort to determine if there was any characteristic that was unique to users who experienced better attack detection. In (Imsand and Hamilton, Impact of Daily Computer Usage on GUI Usage Analysis 2007), Imsand and Hamilton conducted a study that sought to determine if such a characteristic could be found.

A very small amount of demographical data was gathered by the authors of (Imsand and Hamilton, GUI Usage Analysis for Masquerade Detection 2007), with none of the solicited characteristics corresponding to a significantly higher attack detection rate. For this reason, the authors of (Imsand and Hamilton, Impact of Daily Computer Usage on GUI Usage Analysis 2007) gathered an entirely new set of participants and created a much larger participant survey. The task list that each participant was asked to

complete remained the same, as did the algorithm used to classify test users (i.e. attacker vs. legitimate user). While the second study mirrored the first study in many ways, there were several key experimental differences. First, the participant pool grew from ten participants to sixteen participants. Secondly, the period of time waited between experimental runs was shortened from two days to at least one hour. The other significant difference observed between these two studies was the pre-experiment briefing given to participants. In the first study participants were not briefed on the overall objective of the research prior to their participation. Participants of the second study were given a briefing describing the overall goals of the research. This disclosure was done at the encouragement of the Auburn University office of Human Subjects Research.

The overall attack detection rate observed in the second study dropped slightly. The initial study produced an attack detection rate of 60% with no false positives, while the second study found an attack detection rate of 52% with no false positives. The authors provided several suspected reasons for the dip in attack detection rate, though no conclusive determinations were drawn. As previously mentioned, the goal of the study was to find characteristics that might indicate a user may be better protected by GUI Usage Analysis. The data presented by the authors indicated that users that spend more than six hours per day using a computer enjoyed significantly better protection from GUI Usage Analysis. Other characteristics demonstrated slightly better performance for members of certain groups, though the observed improvements were not nearly as pronounced as in cases in which a user spent more than six hours a day operating a computer. Unfortunately the overall number of participants in the study prevented the

authors from drawing definitive conclusions regarding computer usage and the protection offered by GUI Usage Analysis.

Multiple deficiencies found in the first two studies of GUI Usage Analysis were identified and corrected prior to the commencement of work on this dissertation. Most notable of these corrections was the inclusion of many more participants. Both of the previous studies featured, at most, sixteen participants. The formal findings presented in this dissertation are based on studies featuring thirty one participants. Furthermore, the analytical techniques utilized in prior GUI Usage Analysis studies were unsatisfactory. Neither study featured proven data mining and classification techniques, instead opting for custom developed solutions. This deficiency has also been addressed in the final findings of this dissertation.

CHAPTER 3

EXPERIMENTAL SETUP

The study presented here was designed to assess the feasibility of using GUI Usage Analysis as both a means of authentication and identification. To accomplish this great care was taken in the design of the performed experiments. As already noted, other studies did not present a clear path to be followed when conducting this study. No single recording tool was used, nor was there a common set of tasks to be performed by participants. This was partially due to the varying nature of what previous studies were attempting to prove; clearly investigators assessing the suitability of command line input profiling would gather different data from individuals investigating mouse movements. Unfortunately there are discrepancies in the literature more fundamental than these differences. Multiple studies observed participants performing the same set of prescribed tasks (Schonlau, et al. 2001), (Pusara and Brodley 2004). Other studies (Garg, et al. 2006) monitored their participants while using a computer as they ordinarily would, completing their everyday tasks.

In (Schonlau, et al. 2001), Schonlau argues that no definitive conclusions about identifying characteristics can be drawn unless the only variable in the study are the identifying characteristics themselves. In other words, unless everything else, including the student's task list is considered to be a constant, no definitive conclusions can be drawn. It is difficult to find fault with this analysis and as a result, the general method

proposed by Schonlau was the method that was adopted for this study. Users were monitored in the same software environment using the same monitoring suite while performing the same set of tasks.

The remainder of this chapter is organized in the following manner. Section 3.1 presents a discussion of the custom monitoring software that was used in the collection of data for this study. Section 3.2 discuses the criteria used to determine who was eligible for participation in this study. Section 3.3 provides a brief outline of the environment that participants used when participating in this study. Section 3.4 discuses the task list that participants were asked to complete and provides the rationale used in some of the limitations imposed on the participants (such as the frequency of completed runs, amount of time between runs, etc.).

Monitoring Software

One of the most important aspects of assessing the viability of identifying or authenticating users by their GUI interaction patterns is the actual collection of user's interactions. It was determined that the following pieces of information had to be gathered from users actively using a graphical user interface:

- The action the user performed
- The application the user was interacting with
- The specific control (i.e. button, text-box, toolbar, etc.) that the user acted on

It was believed that failure to collect any of these pieces of information would prevent successful classification (identification and/or authentication) from occurring at acceptable levels. To illustrate the rationale behind this belief, suppose that the specific

control that a user was interacting with was not collected. It would then be impossible to determine if a user's left click was performed on the "File" menu, a button on the toolbar, relocating the cursor within a document, etc.

It was determined that the utility *Spy++* provided by Microsoft Corporation as part of its Visual Studio application provided the desired functionality. Unfortunately the interface for *Spy++* was determined to be too complex for the average user to interact with.

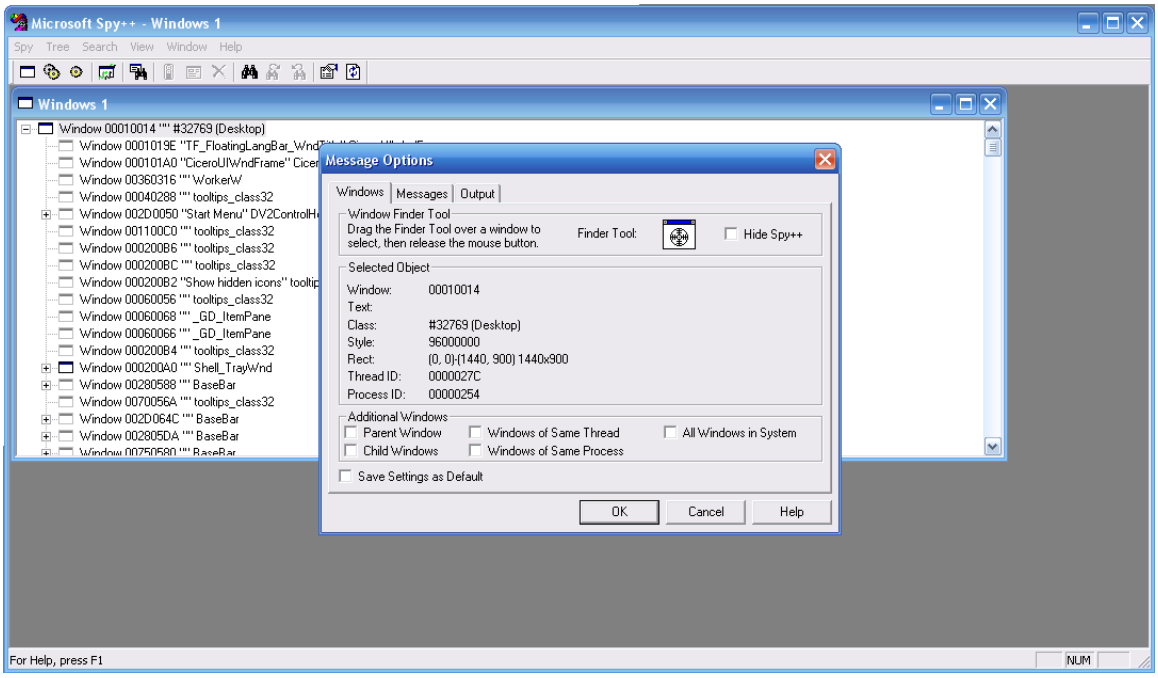

Figure 1. *Spy++* User Interface

The difficulties provided by the unwieldy interface provided by *Spy++* could have been overcome. Unfortunately *Spy++* was unsuitable for this study for other reason as well. There was no way to automate the setup and configuration of *Spy++* between experimental runs. Proper configuration of *Spy++* required several highly specific interactions with the interface. Concern that human error could lead to the correction of

incorrect or incomplete data was high. Furthermore, the application could not be placed in a stealth mode in which it was hidden from users. There was concern that the complex interface presented by the application might serve to distract or confuse the users while they were participating in the experiment.

When the decision was made not to use *Spy++* alternatives were sought out. Unfortunately no freely available utilities were found at the time that provided all of the desired data. The decision was made to create a custom utility to use as a data collection instrument in this study.

Research was commenced on how to log keystrokes and mouse-clicks in the Microsoft Windows Operating System. This research quickly yielded two key pieces of information. The first was that the Microsoft Windows operating system uses messages sent from the OS to applications to inform them of user actions that should be responded to (Microsoft Corporation 2007). The second key piece of information was that Windows provides a construct that Microsoft terms "hooks" that enable an application to intercept these messages (Microsoft Corporation 2007a). According to Microsoft, a hook is, "a point in the system message-handling mechanism where an application can install a subroutine to monitor the message traffic in the system and process certain types of messages before they reach the target window procedure."

The Windows XP operating system provides a wide variety of hooks that can be installed depending on which messages an application seeks to capture. Also, hooks can be configured to capture only the messages for a single application, or for all applications on the system. For this study it was determined that the *WH_GETMESSAGE* hook was

appropriate as that hook provided the ability to capture all messages sent by the OS to the application (Microsoft Corporation 2007a). Other hooks were considered, such as the *WH_KEYBOARD_LL* hook as well as the *WH_MOUSE_LL* hook. Unfortunately experimentation showed that the only hook that provided all of the information needed for full analysis was *WH_GETMESSAGE*.

The actual mechanics involved in the capture of Windows messages is somewhat complex. The Microsoft Developer Network entry on hooks states:

> "The system maintains a separate hook chain for each type of hook. A *hook chain* is a list of pointers to special, application-defined callback functions called *hook procedures*. When a message occurs that is associated with a particular type of hook, the system passes the message to each hook procedure referenced in the hook chain, one after the other." (Microsoft Corporation 2007a)

Stated more clearly, a hook allows an application to inject a specified sub-routine into the message processing machinery of each process. Conceptually this can be thought of as simply replacing the message handling sub-routine for each application running on the system.

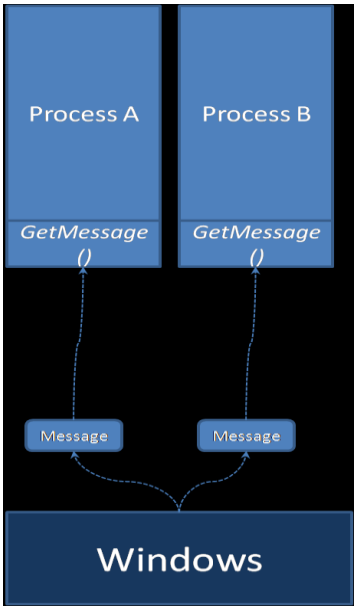

Figure 2. Message Processing Prior to "Hooking"

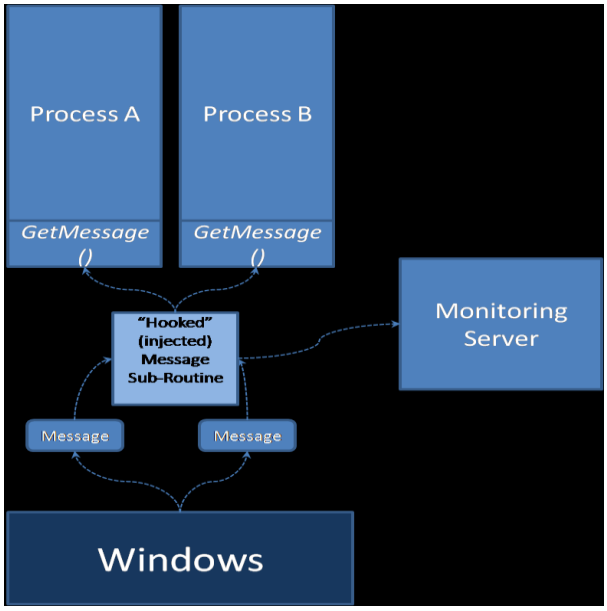

Figure 3. Message Processing After "Hooking"

Clearly capturing the data sent by the operating system to each application did not completely meet the requirements needed for the data collection phase of this study. The other difficulty that had to be overcome by the development of the monitoring software was a logging procedure to store the data for later retrieval. Ideally all of the collected information would be stored in a file stored locally and then retrieved later. In theory it would have been simple for the "hooked" sub-routine to simply copy all captured messages to a log file before allowing the target application to process the data. Unfortunately certain characteristics of the hooking process prevented this simple scenario from being executed. It was discovered that the injected sub-routine ran in the context of the actual application that was to receive the message from the operating system. For this reason the sub-routine could not attempt to use a single common file to record the information, due to the fact that it would be impossible to pass an open file pointer into the function prior to hooking (Newcomer 2003). With simplest design

eliminated due to technical problems, other data collection facilities were considered, such as transmitting data to a database stored either locally or on a server located elsewhere on the Internet.

It was decided that the "hooked" sub-routine (the sub-routine that was injected into each process) would pass copies of all intercepted messages back to a server application via UDP datagrams. Since the server application was also running on the local host, it was determined that the likelihood of packet loss was low, making the reliability of TCP not worth the added overhead. This solution was chosen because it removed the need for an external database stored locally or remotely.

In its final form the monitoring software consisted of two components: a server application and a library that was injected into each application running on the system. The final system was initially based on an open source project published on the internet (Newcomer 2003). That code was then heavily modified to suit the specific requirements of the monitoring software needed for this study.

The server application was responsible for invoking the API that caused Windows to inject the library into each executable. It was also responsible for "unhooking" the DLL from each executable. The final responsibility of the server application was to display an easy to use graphical interface that study participants could easily understand and work with.

Figure 4. Interface for of the monitoring software

Up to now the messages have been discussed as if they automatically contained all of the needed data to perform an analysis. Unfortunately this was not the case. The messages transmitted by Windows typically contain the following pieces of information:

- A handle to the "window" the message is addressed to
- The message itself
- A pointer to additional data about the message. The contents of the additional data were dependent on the message itself.

Of the three things that were needed to conduct the analysis, only a single item was provided by Windows in the original intercepted message. Fortunately the Windows XP API contains several functions that could be used to deduce the remaining desired information. The *GetClassName* function is provided by Windows XP and returns a string containing the name of the class that the specified window is an instance of (Microsoft Corporation 2007). As outlined earlier, the ideal dataset would have enabled the analysis to compare precisely what each user was clicking or typing on. The information obtained by examining the classes of the controls acted upon does not quite provide that same level of specificity. For instance, two different users could click on two different buttons located on the toolbar of an application. Because the two buttons are instances of the same class, though, the analysis method used in this study had no way

to tell that these were not identical actions. Unfortunately a reliable manner of obtaining the precise data that was desired was never found. It is not believed that relying on this slightly less specific data dramatically affected the results of this study.

The other item of information that was not contained in the original message sent by the operating system was the actual application that the message was destined for. Ordinarily, when applications are not attempting to inspect all messages transmitted by the operating system, this piece of information is not important. When an event occurs Windows determines the active window, looks up the message processing chain associated with that window, and transmits the message accordingly. In this study, though, that information was not enough. A method of logging the application that received the action was required. Fortunately, as outlined earlier, the library function that inspected all of the message traffic ran in the context of the application that was to receive the message. Using the *GetCommandLine* function provided by the Windows XP API, the library function was able to discover the name of the application that it was currently executing in (Microsoft Corporation 2007). This information was transmitted to the server portion of the monitoring software, along with the name of the class of the control to receive the event.

Participant Eligibility

One of the underlying suspicions used in the design and implementation of this study is that users have certain routines or habits that are unique to each person. It was suspected at the outset of this study that, on average, two different people completing the same set of tasks would undoubtedly use a different set of actions to accomplish their goals. For GUI Usage Analysis to be suitable as a means of either identification or

authentication, though, each person would have to have a certain degree of consistency to their actions. For this reason it was determined that the first and most important criteria that study participants were required to meet was that they be moderately proficient at the use of a computer.

To illustrate why this limitation was placed, consider a novice user who is completing some task for the first time. When asked to complete a task like "spell check the current document, correcting all errors," the user may be able to "stumble" their way through this task. The interface of modern computing system is designed to be intuitive and the novice user takes advantage of this. The user, reasoning that the spell check system is a "tool" offered by the word processor, searches under the "Tools" drop down menu and successfully locates the spell check system. On the third session the user happens to place their mouse over the spell check shortcut and a tool tip appears indicating that pressing the current button will invoke the spell checker. The novice user, not having any particular manner in which they are accustomed to doing things, might decide to use this execution path to invoke the spell checker because it requires fewer mouse clicks. Because the user is a novice they are continually searching and learning, attempting to complete each task. As they rapidly acquire new skills their behavior will undoubtedly change.

Demographical Make-up of the Participant Population

The participants used in this study were all non-students, meaning that they were all full time employees of some organization. This was considered advantageous in order to ensure a participant population that was not overly skewed towards any particular

demographic. The following pieces of information were obtained from the study participants:

- Profession
- Highest degree of academic achievement
- Gender
- Age range
- Self-assessment of computer skills
- Average daily computer usage
- Source of computer knowledge (self-taught or through a book/instruction)

There were a variety of reasons behind the collection of the demographical information from the participants. Some questions, particularly inquiries regarding gender and age, were made with no prior indication that those pieces of data would yield any interesting findings. They were collected for the sake of completeness in case the initial beliefs were proven concerning the utility of that data were proven to be incorrect. The remaining pieces of information were quite purposefully collected from the participants.

Prior to commencement of the study, it was hypothesized that an individual's profession might greatly influence their suitability for GUI Usage Analysis. For instance, it was hypothesized that a user's skill at operating a computer might have a great impact on their suitability for GUI Usage Analysis. It was believed that more skilled users may know of more efficient, less conspicuous manners of accomplishing tasks. In this way their higher skill set might lead to a more unique behavioral thumbprint.

It was also hypothesized that the source of a user's computer skills might also greatly influence their suitability for GUI Usage Analysis. It was believed that users that

taught themselves might be more likely to have unique behavioral fingerprints when compared with those who learned from a common book or instruction. The downside to this question is that it may not be possible to fully investigate and answer with the data collected in this study, as the participants that indicated they learned from a book or instruction presumably did not all use the same book/instructor.

Daily computer usage was also considered to be a trait that might impact the performance of the algorithm in this study. It was hypothesized that persons that use a computer for longer periods of the day may have behavioral patterns that are more firmly entrenched than users that spend less time in a day using a computer. Though this initial description sounds similar to the rationale behind the collection of data concerning computer skills, the two pieces of information are different. Consider a clerk who routinely enters data into Microsoft Word or Excel files. This clerk knows how to do his/her job well and has learned to complete most common tasks in those two applications quite efficiently. This person's skill set is confined, though, to the use of Microsoft Excel and Word, meaning that this person hardly qualifies as an expert computer user.

The final, unaddressed piece of data gathered from the study participants concerned their academic achievement. It was hypothesized that an individual's native intelligence might be a more accurate indicator of success with GUI Usage Analysis than any of the other items previously addressed. It was believed that intelligent users may be more likely to learn methods of accomplishing tasks, be classified as expert users, as well as being heavy computer users. While hardly conclusive, the degree academic

achievement does have a natural link to an individual's native intelligence, explaining why it was chosen as an item to be collected from the participants of this study.

Though 31 participants completed the experiment correctly, only 29 completed the required questionnaire. The information provided by those 29 participants is listed in Table 2. Initial inspection seems to show the participant population as being heavily comprised of individuals in two different classes: IT and Education. It is important to realize, though, that the individuals that listed their field as "Education" are faculty at a particular high school. These individuals were chosen because of their diverse specialties. While it may be factually correct to list their field as "Education", each of them is equally proficient in a secondary skill such as mathematics, literature, physical science, etc.

Table 2. Demographic Survey Results

Participant Number	Current Profession	Highest Degree of Academic Achievement	Gender	Age	Self Assessment of Computer Skills	Avg. Amt Of Daily Comp. Usage	Source of Comp. Skill
1	Marketing / PR	Some Undergrad.	F	40-65	Average	8+	Course / Instruct.
2	Accounting	Some Grad.	F	40-65	High	8+	Self Taught
3	IT	Some Undergrad	M	25-39	Average	4-6	Self Taught
4	IT	Associates Degree	M	18-24	High	6-8	Self Taught
5	IT	Some Grad.	M	25-39	Above Average	8+	Self Taught
6	Engineering	Graduate Degree	M	25-39	High	6-8	Self Taught
7	IT	Some Undergrad.	M	18-24	Average	2-4	Self Taught
8	IT	Some Undergrad.	M	25-39	High	8+	Self Taught

Table 2 – Continued

Participant Number	Current Profession	Highest Degree of Academic Achievement	Gender	Age	Self Assessment of Computer Skills	Avg. Amt Of Daily Comp. Usage	Source of Comp. Skill
9	Accounting	Associates Degree	F	25-39	High	6-8	Course / Instruct.
10	IT	Associates Degree	M	40-65	Average	4-6	Self Taught
11	Engineering	Associates Degree	M	40-65	Average	6-8	Self Taught
12	IT	Some Undergrad.	M	40-65	Above Average	2-4	Self Taught
13	IT	Some Undergrad.	M	25-39	Above Average	8+	Self Taught
14	IT	Associates Degree	M	18-24	High	8+	Self Taught
15	Education	Some Grad.	F	40-65	Above Average	2-4	Self Taught
16	Education	Graduate Degree	F	40-65	Average	2-4	Course / Instruct.
17	Education	Graduate Degree	F	40-65	High	2-4	Course / Instruct.
18	clerical	Some Undergrad	F	40-65	High	8+	Self Taught
19	Education	Graduate Degree	F	40-65	High	6-8	Self Taught
20	Education	Graduate Degree	F	40-65	Above Average	4-6	Self Taught
21	Education	Graduate Degree	M	25-39	Above Average	2-4	Self Taught
22	Education	Undergrad. Degree	F	40-65	Above Average	2-4	Self Taught
23	Education	Graduate Degree	F	25-39	Above Average	2-4	Course / Instruct.
24	Education	Undergrad. Degree	F	40-65	Above Average	2-4	Self Taught
25	Education	Graduate Degree	F	40-65	Above Average	2-4	Self Taught
26	IT	Graduate Degree	M	25-39	High	6-8	Self Taught

Table 2 – Continued

Participant Number	Current Profession	Highest Degree of Academic Achievement	Gender	Age	Self Assessment of Computer Skills	Avg. Amt Of Daily Comp. Usage	Source of Comp. Skill
27	Education	Graduate Degree	M	40-65	Above Average	6-8	Self Taught
28	Education	Graduate Degree	F	40-65	High	8+	Self Taught
29	IT	Graduate Degree	M	40-65	High	8+	Course / Instruct.

Experimental Environment

The data collected in this study utilized an identical software configuration on all systems. The systems were configured as follows:

- Microsoft Windows XP Professional, Service Pack 2
- Microsoft Office 2003
- Microsoft Internet Explorer 6.0

Users completed the experiment using a standard three button mouse and standard keyboard.

Experimental Task List

Creating the task list was a surprisingly difficult endeavor. The nature of this study placed a premium on tasks that normal users would automatically know how to do. Unfortunately coming up with tasks that the majority of users knew how to do proved difficult, as evidenced by the need to suppress all interactions with Microsoft Excel.

Many computer proficiency books were examined in an attempt to find a suitable task list. Many computer literacy tests, offered both online and in print form, were also examined. It was determined that none of the tests examined met the needs of this study. Frequently the tests were judged to be too complex and containing tasks that the average user would not know how to complete. As a result the task list used in this study was created by the investigators and comprised of tasks that, in the estimation of the investigators, most users would automatically know how to complete.

The task list itself consisted of a total of ten steps, with some steps having sub-tasks that had to be completed. The complete task list is shown in Figure 5.

User Identification Research Experiment
Eric Imsand
imsanes@auburn.edu
(901) 338-9323

This experiment that you are being asked to participate in is designed to determine whether the ways in which we use a computer are unique to each person. If this is the case then it may be possible to identify people by how they work with a computer, not just the username and password that they typed in. In order to determine this a small amount of experimental data must be collected. You will shortly be asked to complete some routine tasks using a computer. While you are completing these tasks a piece of monitoring software will record which buttons you click on, keys you press, etc. At no time will **any** personal information be collected from you.

Participation in this experiment is voluntary. If you have any questions about the scope of this experiment or what kind of data is to be collected, please contact Eric Imsand.

NOTE: **Please close Microsoft Outlook before starting the experiment.**
Directions:

1. Click on *Start -> Programs -> E Imsand -> Recording Software*

2. Click the "Record" button to start recording.
   (NOTE: You may minimize the recording software window if you like)
3. Create a folder in the "My Documents" folder. The folder should be named "CompTest"
4. Download the file "sample_document" from http://www.auburn.edu/~imsanes/research/. **When asked, please choose to save**
   the document to the folder you created in step #3.
5. Open the document you downloaded in the previous step if it is not already opened.
   a. Change the font for the entire document to Times New Roman with a 12pt. font size.
   b. Center the title (the first line of text on the first page), and increase its font size to 16 pt., also making it bold and italics.
   c. Move the last paragraph in the document to be the third paragraph in the document.
   d. Spell check the entire document, correcting all errors.
   e. Save the document as "research document" in the folder created in step #3.
   f. Close Microsoft Word.
6. Download the file "Data" from http://www.auburn.edu/~imsanes/research/.
7. Open the document you downloaded in the previous step if it is not already opened
   a. Calculate the average of columns A & B for all rows (i.e. calculate the average of A2 & B2, A3 & B3, etc.) and store the value in column C (labeled "Overall average") of that row.
   b. Save the document as "research data" in the folder created in step #3.
   c. Print the document (if your computer is configured to print to more than one system, use the default printer).
   d. Close Microsoft Excel.
8. Perform the following file and directory operations:
   a. Change the name of the file created/saved in #4 from "research document" to "rd#####" where ##### is your actual student ID or your initials.
   b. Move the folder created in step #1 and all of its contents to the Desktop.
9. 9. Using **Internet Explorer** (not Netscape, Fire Fox, etc.) please navigate to the following websites.
   - MSNBC
   - CNN
   - Google
   - Auburn University
   - Randolph School

10. Click the "Stop" button on the monitoring software. Close the recording software. If a survey is displayed, please complete it.

Figure 5. Complete Task List provided to Experimental Participants

As previously alluded to, undergraduate and graduate students were specifically avoided in the recruitment of test subjects for this study. The fact that all participants were working professionals provided several advantages. Perhaps the most significant of these advantages was that the participants' employment allowed verification that the chosen participants were capable of completing the task list prior to commencement of the study. Each participant's supervisor was consulted concerning the participant's computer skills and their ability to successfully complete all of the tasks on the list. The supervisors provided verification that all of the invited participants were capable of completing the tasks on the list, with one possible exception. The assignment list, which is fully enumerated later in this chapter, asked users to complete some tasks in Microsoft Excel. Several supervisors indicated that their participants might be incapable of completing the Excel tasks. The decision was made to ask the participants to complete the tasks in Microsoft Excel and then conduct a post-experiment interview to determine if there were any tasks on the list that they were incapable of completing. Approximately half of the participants indicated that they were not able to complete the Microsoft Excel portion of the study. All of the interactions with Microsoft Excel were removed from the data set prior to analysis.

Administration of the Experiment

Participants in this study were required to complete the task list a total of five times. The selection of five runs was a compromise between the need to obtain enough information to conduct the experiment without placing an unreasonable burden on the participants. Participants were asked to complete the task list five times on a single day

with at least half an hour between runs. The selection of one half hour as an interval between runs was another compromise between the need participants' need for flexibility and a need for some interval between runs. The interval was added to the experimental procedure in the hopes that participants would not remember how they completed the task list. It was hoped that even if the participants could remember *what* they did they would not be able to remember precisely *how* they did it.

There was some consideration given to the amount of time the administration of the experiment should cover. It was initially believed that the experiment would be carried out over a series of days in an effort to minimize the likelihood that a participant might adjust their behavior to better suit themselves to the need of the study. On the other hand, most users do learn new skills and change the manners in which they consciously complete a task. In the end it was decided that the best approximation of what an actual, deployed system might encounter would be to have the participants perform the experiments in a single day.

The overall requirements of the study definitely had an adverse impact on participation. Over 55 users originally volunteered to participate in the study. Of these approximately 55 users, only 31 successfully completed the study in the prescribed manner. Of the 31 that successfully completed the tasks as prescribed by the assignment list, only 29 completed the survey on their demographics. These users have been excluded from parts of the analysis dealing with portions of the subpopulation.

As previously stated, participants in this study completed the task list a total of five times. It was believed that, due to the artificial nature of the testing environment (i.e.

taking users out of their ordinary environment) two "test" runs should be administered to users prior to the collection of data actually used in this study. This allowed users to become familiar with the testing environment as well as to refresh their skill set. It was suspected that at least some of the participants in this study would have a working knowledge of how to complete each task on the list but, due to the nature of their daily computer usage, would not immediately recall precisely how to perform each step. Consider a network administrator who spends a great deal of time working with router configuration menus and operating system security policies. This user has used the spell check functionality of a word processor many times in the past but does not need to make use of that particular skill on a daily basis. By allowing two runs before actual data collection, this user is able to refresh his/her memory and recall precisely how they typically invoke the spell check system.

Ignoring the first two sessions that were collected had other advantages as well. Many users expressed concern about their performance, frequently making statements such as, "I'm not sure if I'm doing this the correct way," or "I don't remember exactly how to do this." To ease these concerns users were informed that their first two runs would not be used in the final analysis. It was believed that telling the participants that their initial two runs would not be analyzed would have little impact on overall result of the study.

Pre-experimental Participant Briefing

Prior to participation in the experiment, the participants were briefed and consent was obtained. The briefing consisted of both a brief oral presentation, typically lasting

less than 2 minutes, as well as information contained in the informed consent form they were required to sign. In accordance with the recommended procedures provided by the Auburn University Office of Human Subjects Research (Auburn University 2007), participants were given a broad overview of the research that was to take place. It is believed that any impact this briefing may have had on the performance of the subjects is minimal. The complete informed consent document that was obtained from all participants is included in Appendix A.

CHAPTER 4

USE OF GUI USAGE ANALYSIS AS AN AUTHENTICATION MECHANISM

After the data was collected from the study participants, a variety of analytical techniques were applied to the data. Multiple techniques were used in the analysis for two reasons: first to determine the suitability of GUI Usage Analysis as a means of authentication and secondly to determine which analytical method yielded the best results. This chapter discusses the varying analytical techniques and the results produced by each of them.

Authentication is typically considered to be the second phase of an access control system. While the identification phase is responsible for determining the user's supposed identity, authentication is concerned with verifying that claim. Authentication typically centers around requiring the user to provide proof in one of three manners: the user provides something (s)he has, the user states something that (s)he knows, or the user provides some physical characteristic (i.e. something the user "is") (Pfleeger and Pfleeger 2003) (Apple, Inc. 2007).

Prior to discussing each technique in great detail, some terminology should be clarified. During this discussion, the term "event" is used to describe the entire set of information yielded from a single user action. In other words, an event is considered to be the combination of the actual user action (left mouse click, 'A' key depressed, etc.), the Windows class of the object the user was interacting with (button, text-box, etc.), and

the executable the user was using at the time. The final piece of information may, at first glance, appear redundant. It is not uncommon for applications developed by differing software vendors to have unique class names for the objects present in their user interfaces. However, since the majority of the applications used by the participants of this study were developed by Microsoft Corporation it was decided to include the final piece of information in order to avoid any confusion that could occur if any two applications happened to share class names.

As previously detailed, each participant generated three sessions which were used as part of this study. When simulating attacks for this study, one user was selected to be the "known" user. Two of the known user's sessions were fed into the classification engine to serve as training data. The other session was omitted to serve as a possible test session. Attacks were simulated by taking a third session and feeding it into the classification engine as well. The classification engine then made a determination as to whether or not the three sessions were generated by the same user, or a different user. The third, unknown session could be either the third session from the "known" user, or it could be a session from a randomly selected user. Unless otherwise noted, all possible combinations of sessions were tested as both training and testing data.

The remainder of this chapter is organized as follows: section 4.1 discusses the overall rationale behind the choices of analytical methods used. Section 4.2 discusses the results of a TF-IDF inspired analysis that was performed with a "window" of size 2. Section 4.3 discusses the results of a using Jaccard analysis. Section 4.4 covers the results of analysis performed on varying sub-portions of the participant population.

Rationale behind Analytical Techniques Used in this Study

When initially considering the data produced by the participants in this study, it became clear that there were some unique considerations. First and foremost, there were few, if any, restrictions on the output that could be generated. Generating an experimental alphabet that could represent all events generated during the course of the study would be highly difficult. Consider the process of enumerating all objects within all of the applications the participants could possibly interact with. Next consider all the possible actions that users might perform on the enumerated objects. Clearly this yields a very large number of possible events.

For this reason, it was decided to investigate analytical methods that were originally designed for processing and analyzing human generated text. Consider the similarities between the two sets of data. When analyzing a set of human generated text there are few limitations on the text that will be produced. As previously noted, this is very similar to the set of data generated by the users in this study.

One additional note should be made prior to considering the differing methods that were used in this study. When evaluating the effectiveness of intrusion detection systems it is customary to measure their performance in terms of false positives and false negatives. "False positives" and "false negatives" are informal terms used to describe type 1 and type 2 errors. A false positive, or type 1 error, is said to have occurred when the null hypothesis was incorrectly rejected. In the case of most of the analyses presented here, the null hypothesis states that a masquerade attack is occurring. This means that a false positive, or type 1 error, occurs when the classification engine incorrectly

determines that an attack is not occurring. Inversely, a false negative, or type 2 error occurs when the classification engine incorrectly determines that an attack is occurring.

For the sake of consistency many of the analysis techniques presented here are quantified using the familiar terminology of "false positives" and "false negatives". For several of the analyses presented, particularly analyses based on variants of the TF-IDF algorithm, the terms "false positive" and "false negative" are not entirely accurate. Those analyses typically assess the degree of similarity between two samples, labeling samples that are sufficiently different as being "attacks". Thus the TF-IDF inspired analyses are reported as having a false positive rate of 0%. In these instances the threshold was set at a level where false negatives could not occur.

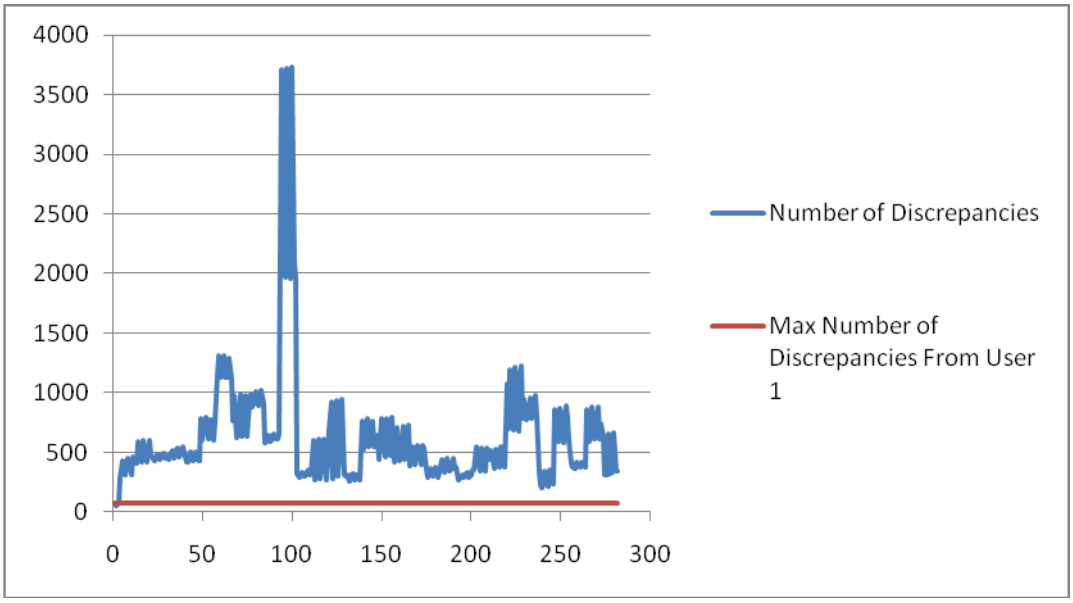

Figure 6. Illustration of Attack Threshold

Figure 6 illustrates the rationale used graphically. The red line in the graph depicts the maximum dissimilarity between any two sessions generated by the same user, user 1 in

this case. The points in blue illustrate the number of discrepancies observed between any random session generated by user one and a session generated by some other user.

Simple Dissimilarity Analysis

As previously described, TF-IDF (term frequency – inverse document frequency) based analysis seeks to determine the importance of an event to the overall corpus. In the context of this discussion the corpus is the larger session of data that is being evaluated. The TF-IDF weight is calculated for each word in a body of text. The weight is equal to the number of occurrences of that word, divided by the total number of words in a document. For example, suppose that the TF-IDF weight of the word "cat" was desired. It would be calculated in the following manner:

$$TF - IDF(\text{cat}) = \frac{count(\text{cat})}{m}$$

Where *m* is the size of the entire document, in number of words.

It is not completely accurate to term the analyses performed here as being TF-IDF analyses. TF-IDF analysis is typically used when seeking to determine the importance of a single term or event to a larger body of data, not for determining the dissimilarity of two sets of data. The term "TF-IDF inspired" will be used in this discussion to demonstrate the role the algorithm played in inspiring the analytical methods described here.

A total of three different simple dissimilarity analyses were performed in this study, focusing on two different attributes. As previously documented, the simple

dissimilarity analyses presented here determine whether or not a masquerade attack is occurring by comparing the number of discrepancies or differences between the known and unknown sessions. If the degree of dissimilarity is greater than some attack threshold *a* then an attack is determined to have occurred. The first dissimilarity based analytical method studied used a static attack threshold for all participants. The second dissimilarity based method used a custom attack threshold for each user as well as using an event window of size one. The final dissimilarity based analysis method used a variable attack threshold for each user as well as using an event window of size 2, meaning that each event and its predecessor were analyzed jointly.

Simple Dissimilarity Analysis with a static Attack Threshold

As a scholarly experiment, GUI Usage Analysis provides the basis for an interesting investigation. For the data obtained to be of any use to the general populace, though, GUI Usage Analysis must eventually be implemented in some sort of software product designed to be used by the general public. The research described in this section outlines one attempt to assess difficulties that might be encountered if GUI Usage Analysis were to be ported to a commercial product.

Ease of deployment is a key consideration for any organization that seeks to utilize behaviorally based systems. A system using GUI Usage Analysis would undoubtedly fall under this category. It is believed that a system using a statically determined attack threshold would be far simpler to deploy as opposed to a system that used a variable attack threshold for each user. A system with a static threshold could potentially bypass the training period. Any training period that was required would

undoubtedly be simpler when using a static threshold as opposed to a dynamic threshold that adjusted for each user.

For this reason, an analysis was performed using a simple dissimilarity analysis and a static attack threshold. In other words, the same dissimilarity threshold was used for all users when determining whether or not a session was generated by the known user or an attacker. If a single threshold can be obtained experimentally then that would represent a finding of great value to any organization that might seek to develop a usable software system incorporating GUI Usage Analysis.

A variety of static thresholds were experimented with. The results of these experiments are shown in Table 3. Please note that the attack detection rate is simply the inverse of the false negative rate. It is included here for the sake of completion.

Table 3. Dissimilarity Analysis with a Static Attack Threshold

Attack Threshold	False Positive Rate	False Negative Rate	Attack Detection Rate
428.5	35.175%	17.24%	82.75%
450	31.94%	20.58%	79.41%
650	24.42%	49.07%	50.92%
700	22.27%	55.6%	44.39%

The results listed in Table 3 indicate that several things. First, a decrease in false positive errors was accompanied by an increase in false negative errors, as expected. Second, the relationship between the two error rates was not linear; an increase of *X* percent in the

false negative error rate did not lead to an equivalent drop in the false positive error rate. A chart plotting the relative ascent and decent of each data rate is given in Figure 7.

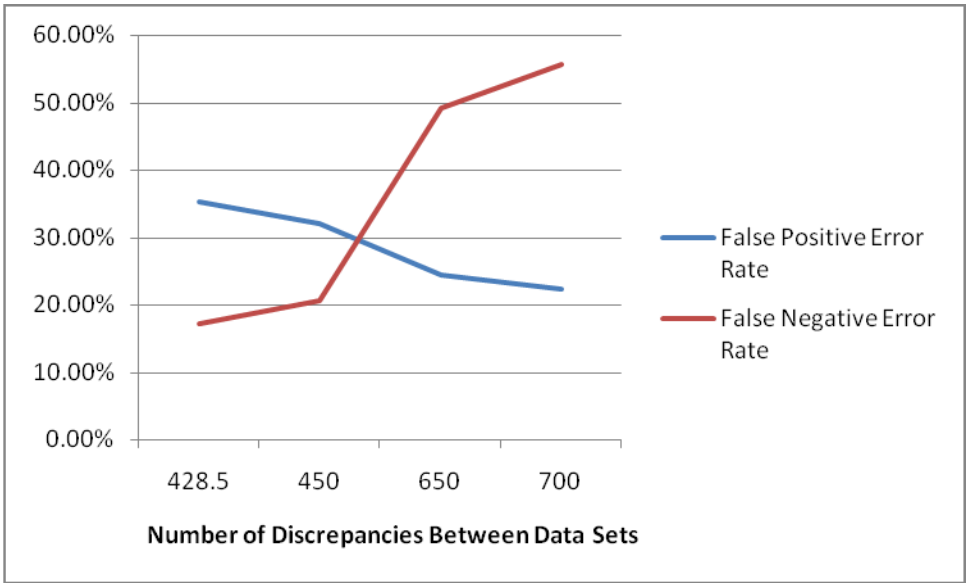

Figure 7. Relationship between False Positive Error Rates and False Negative Error Rates

Simple Dissimilarity Analysis with a variable Attack Threshold

The analytical method that produced the most promising results was a TF-IDF inspired analysis with a window of size one. As previously described this indicates that events were considered singularly, without any consideration to the event that preceded it or followed it. This analysis was conducted in the following steps:

- Calculate how often each individual event occurs in the two known samples
- Average the number of occurrences by dividing by 2.
- Calculate how often each individual event occurs in the unknown sample
- Calculate *A,* the difference between how often a particular event occurs in the averaged known sample in comparison to the unknown sample
- Calculate *B,* the occurrences of events that were present in the known sample

- Calculate *C*, the occurrences of events that were present in the unknown sample.

The total number of discrepancies between the known sample and unknown sample was calculated as: $\sum A + B + C$, where *A* is considered to be the difference between the number of times a particular event occurred in the known and unknown sample.

The results achieved using the simple dissimilarity analysis with a window of size 1 were generally very good. Strictly speaking this analysis produced an attack detection rate of 91.34%. In terms of false positives and false negatives, there was a false positive rate of 0% and a false negative rate of 8.66%. Performance of particular users is listed in Table 4.

Table 4. Attack Detection Rate for TF-IDF Inspired with Window Size 1

ID Number	Attack Detection Rate
1	100.00%
2	100.00%
3	98.58%
4	78.72%
5	100.00%
6	99.29%
7	65.96%
8	100.00%
9	60.28%
10	100.00%
11	73.40%
12	100.00%
13	97.87%

Table 4 – Continued

ID Number	Attack Detection Rate
14	100.00%
15	98.58%
16	88.65%
17	91.84%
18	89.01%
19	98.58%
20	77.66%
21	86.88%
22	99.65%
23	100.00%
24	82.62%
25	99.65%
26	71.63%
27	95.74%
28	97.16%
29	98.58%
30	81.21%
31	100.00%

As illustrated in Table 4, there were a total of 31 participants that successfully completed the experiment at least five times. Student's *t* distribution was used to approximate the overall success rate for the entire population. Using an α of 0.05 with 30 degrees of freedom yields a *t* value of 2.04. The observed data had a standard deviation

of 11.63. This data was used in the following calculations to estimate the interval for the estimated attack detection rate of the entire population, μ:

$$X \pm t_{n-1}\frac{S}{\sqrt{n}}$$

$$91.34 \pm 2.04\frac{11.63}{\sqrt{31}}$$

$$87.08 \leq \mu \leq 95.6$$

Attempted Simulation of Real-World Performance

Obviously three samples is not a large enough sample of data to declare the results achieved here as definitive. If the participants submitted to fourth, fifth, and sixth experimental runs it would not be surprising for a wider degree of dissimilarity to be observed between any two experimental runs. For this reason additional analysis was performed with a threshold of dissimilarity set at a different level.

As already discussed the similarity threshold used previously was the maximum dissimilarity seen between two sessions of the same user. A subsequent analysis was performed with the threshold being adjusted by one standard deviation. In other words, a standard deviation was added to the attack threshold to make it appear that the user supplied data was less consistent. The decision to adjust by one standard deviation was driven by the belief that three data points represented too small a number to estimate the actual distribution that participants' data might present. The choice of one standard deviation was admittedly arbitrary. It was felt, though, that the choice of a single standard deviation was as valid as any other amount of adjustment because of the large

number of unforeseen variables that are present when considering any human-centric system in deployed "in the field". In other words, the author felt that it would not be possible to accurately account for all of the differences that might be encountered when transitioning the system from controlled settings into a real-world deployment. The addition of one standard deviation to the attack threshold simply represents a "good faith" estimation of the difference in performance that might be encountered outside of a laboratory.

Obviously adding one standard deviation to the results caused the attack detection rate to fall. Whereas the first method resulted in an average attack detection rate of ~91%, the secondary analysis resulted in an average attack detection rate of ~81%. The performance of each individual when adding a standard deviation to the attack threshold is indicated in Table 5.

Table 5. Attempted Extrapolation of Real-World Attack Detection Rates

ID Number	Attack Detection Rate
1	98.58%
2	100.00%
3	83.69%
4	48.94%
5	99.29%
6	97.16%
7	49.29%
8	100.00%
9	39.36%
10	100.00%

Table 5 - Continued

ID Number	Attack Detection Rate
11	49.29%
12	100.00%
13	91.13%
14	100.00%
15	97.16%
16	70.57%
17	66.31%
18	66.31%
19	88.65%
20	60.64%
21	65.60%
22	98.23%
23	100.00%
24	62.06%
25	97.16%
26	60.99%
27	79.79%
28	92.91%
29	89.72%
30	63.12%
31	97.87%

As previously alluded to, while it appears that the attach detection rate is normally distributed across the entire population, it was not at all clear that the dissimilarity

between sets generated by the same user was normally distributed. It was felt that having only three data points was too few to speculate as to the precise nature of the distribution. This was the rationale behind the decision to conduct additional analysis by adding one standard deviation to the attack threshold as opposed to generating a confidence interval based on *t* values.

These calculations were performed, though, and are presented here for the sake of completeness. Using the three known dissimilarity scores, an upper limit for each user's attack threshold was calculated with an $\alpha = 0.05$. The average attack detection rate when analyzing the data in this manner was 83.95%, slightly better than the average detection rate of 81% observed when adjusting each user's attack threshold by one standard deviation.

Likelihood of Individual Users Successfully Perpetrating a Masquerade Attack

After determining the overall likelihood that any given session might represent a masquerade attack it was decided to investigate the likelihood that each individual user might be able to perpetrate a masquerade attack on a separate user. In other words, it was hypothesized that the majority of false negative errors for any particular user might be generated by a relatively small portion of the population. It was believed that some users might possibly be able to impersonate another user with high accuracy. Appendix B shows the individual results obtained for each possible combination of known user and attacker.

Simple Dissimilarity Analysis with Variable Attack Threshold and Event Window of Size 2

The high success rate achieved when using a simple dissimilarity analysis and an attack threshold that varied for each user was encouraging. Further analysis was performed in an attempt to better these results. The initial analysis was concerned with the overall composition of two sets of data, with little regard for the order in which the events occurred. It was hypothesized that the results might be improved by attempting to factor chronological ordering into the analysis.

This method can be described as a simple dissimilarity analysis with an event window of size two. The analysis links each event with the event that preceded it in an attempt to cause the analysis to consider the chronological order in which events occurred. This was the only change from the analytical method described in section titled, "Simple Dissimilarity Analysis with a static Attack Threshold".

To better understand the analytical method used here, suppose that the following events are executed by user 1: *A C E B G D*. The analytical method used in section one would have simply compared the data provided by user 1 against an unknown sample. The analysis itself simply consisted of comparing how often event *A* occurs in the datasets provided by the known and unknown users. The method used in this section is slightly different, though. Instead of comparing how often each event occurred in isolation, pairs of events were compared instead. Instead of focusing on how often event *E* occurred in each dataset, this method instead focused on how often events (*C, E)* occurred in tandem in the data sets provided by both the known and unknown users.

Stated differently, an event window of size two caused the algorithm to compare how often (*A, C*), (*C, E*), (*E, B*), (*B, G*), and (*G, D*) occurred in both the known and unknown datasets.

Surprisingly, the results achieved using this method were markedly worse than the results achieved in the "Simple Dissimilarity Analysis with a static Attack Threshold" experiment. The overall attack detection rate was found to be 50.72%. As always false positives were prevented from occurring. The false negative error rate, which was simply *1 – {Attack Detection Rate}* was 49.28%. The standard deviation observed in this analysis was also much higher than the results achieved in the section titled, "Simple Dissimilarity Analysis with a static Attack Threshold" (32.53% for this experiment in comparison to 11.63% for the previous experiment). The results for each participant are provided in Table 6.

Table 6. Dissimilarity Analysis with a Static Attack Threshold and an Event Window of 2

User ID	Attack Detection Rate
1	100.00%
2	0.61%
3	23.48%
4	17.99%
5	67.68%
6	99.39%
7	42.68%
8	97.26%
9	88.72%

Table 6 - Continued

User ID	Attack Detection Rate
10	99.39%
11	35.37%
12	71.34%
13	93.60%
14	64.94%
15	10.98%
16	14.63%
17	37.80%
18	84.45%
19	42.68%
20	13.11%
21	43.90%
22	21.65%
23	25.00%
24	25.91%
25	45.12%
26	7.32%
27	51.22%
28	61.59%
29	84.15%
30	14.02%
31	86.28%

Simple Dissimilarity Analysis with a static Attack Threshold and Event Window of Size Two

The results achieved in section 4.3.4 were considered to be so discouraging that an analysis using dissimilarity analysis, a static attack threshold, and an event window of two was not conducted. This suspicion was based on the fact that the results achieved when using a static attack threshold and an event window of size one were worse than the results achieved when a variable attack threshold was used. This result was not surprising at the time. It was strongly suspected that a similar trend would be observed when analysis was performed using an event window of size two.

Jaccard Co-efficient Analysis

Originally developed by Paul Jaccard in the early 20th century, the Jaccard co-efficient is often used to determine the similarity between two sets of data (Kumar, et al. 2006) (He, Chen-Chuan Chang and Han 2004). The Jaccard Index for two sets of data is computed as: $(A, B) = \frac{|A \cap B|}{|A \cup B|}$. The Jaccard Index is very similar to the basic dissimilarity computation featured in the previous section. It supplements the simple dissimilarity scale by factoring in the total number of events in the sets.

The Jaccard co-efficient was computed from the data collected in this study in the following manner:

1. The number of events that the two data sets had in common was computed and used as $A \cap B$. If an event was found to have occurred a differing number of times in the two datasets, the lower number of occurrences was used. If event *A*

occurred 7 times in dataset 1 and 9 times in dataset 2, seven was computed as part of the overall sum used to indicate the intersection of the two datasets.

2. The total number of events was calculated and used as $A \cup B$. In the opposite manner of the intersection operator, if an event was found to occur in both datasets, the larger of the two values was used as part of the sum used to represent the union of the two datasets.
3. The intersection computed in phase 1 and the intersection computed in phase 2 was divided to yield the Jaccard co-efficient used in this study.

Jaccard Co-efficient Analysis with Universal Attack Threshold

The analysis performed with simple dissimilarity using a static threshold produced less than desirable results. Both the false positive and false negative rates were higher than would be desired in a commercial system. Despite the lack of success demonstrated with the static attack threshold when used with simple dissimilarity analysis, a similar experiment was conducted using the Jaccard Index. A suitable attack threshold that could be used for all users would undoubtedly be easier to manage than dynamic thresholds individualized for each user.

The mathematical properties of the Jaccard Index increase the likelihood of finding a single attack threshold that might be suitable. By dividing the intersection of two data sets by the union of those same two datasets, the Jaccard Index normalizes its values making comparisons easier to conduct. For example, a static attack threshold implemented on a simple dissimilarity analysis would probably not be successful. The simple length of the datasets produced by different users make a simple count of the

discrepancies difficult to use as a threshold. This problem is addressed by the Jaccard Index as previously described.

The value for the static attack threshold was experimentally derived. As expected, there was a direct inverse effect between the false positive and false negative error rates. An increase in one error rate coincided with a decrease in the other error rate. The false positive, false negative, and attack detection rates for certain illustrative attack thresholds are listed in Table 7. Please note that the attack detection rate is simply $1 - < False\ Negative\ Error\ Rate >$ and is included in the table only to enhance comprehensibility.

Table 7. Jaccard Index with a Universal Attack Threshold

Attack Threshold	False Positive Rate	False Negative Rate	Attack Detection Rate
0.70	52.69%	0.90%	99.1%
0.675	45.16%	1.21%	98.79%
0.65	37.63%	1.92%	98.08%
0.625	33.33%	3.53%	96.47%
0.60	26.88%	5.77%	94.23%
0.575	19.35%	9.05%	90.95%
0.55	12.90%	13.26%	86.74%
0.525	7.53%	18.54%	81.46%
0.50	3.23%	25.53%	74.47%

Unlike the performance of the simple dissimilarity comparison algorithm, the attack detection rate found when using the Jaccard Index was, in the estimation of the researcher, reasonable. Most encouraging was the fact that, unlike the simple dissimilarity analysis, the false negative error rate climbed at a relatively slow rate. Figure 8 provides a graphical representation of the relative rise and fall of the false positive and false negative error rates.

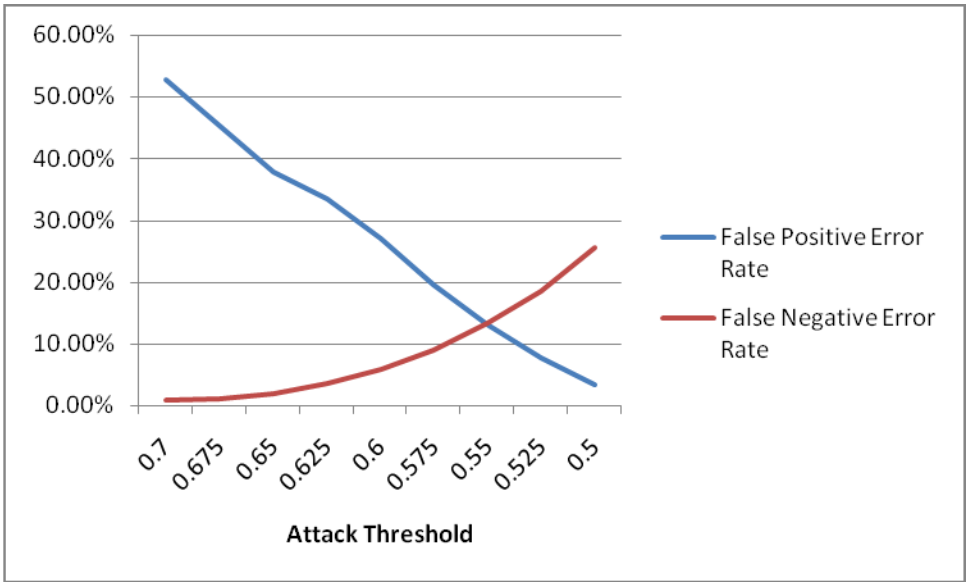

Figure 8. Relationship between False Positive Error Rates and False Negative Error Rates using the Jaccard Index

The overall performance was determined by the researcher to be satisfactory. The suitability of these statistics is ultimately up to any customer were this to someday be developed into a commercial product. Assuming that the majority of businesses would prefer the results achieved with an attack threshold of 0.525, the specific performance of each user is provided in Table 8.

Table 8. Individual Performances with a Universal Attack Threshold using Jaccard Analysis

User ID	False Positive Rate	False Negative Rate	Attack Detection Rate
1	0.00%	0.00%	100.00%
2	0.00%	11.70%	88.30%
3	0.00%	24.47%	75.53%
4	0.00%	32.62%	67.38%
5	0.00%	37.94%	62.06%
6	0.00%	33.69%	66.31%
7	33.33%	5.32%	94.68%
8	0.00%	24.82%	75.18%
9	66.67%	17.38%	82.62%
10	0.00%	15.96%	84.04%
11	0.00%	20.21%	79.79%
12	0.00%	21.28%	78.72%
13	0.00%	39.36%	60.64%
14	0.00%	16.67%	83.33%
15	0.00%	33.69%	66.31%
16	33.33%	0.71%	99.29%
17	33.33%	0.35%	99.65%
18	33.33%	3.55%	96.45%
19	0.00%	0.71%	99.29%
20	0.00%	14.18%	85.82%
21	0.00%	25.53%	74.47%
22	0.00%	25.18%	74.82%

Table 8 - Continued

User ID	False Positive Rate	False Negative Rate	Attack Detection Rate
23	0.00%	0.35%	99.65%
24	0.00%	29.43%	70.57%
25	0.00%	30.14%	69.86%
26	33.33%	18.79%	81.21%
27	0.00%	15.60%	84.40%
28	0.00%	11.70%	88.30%
29	0.00%	0.00%	100.00%
30	0.00%	28.37%	71.63%
31	0.00%	35.11%	64.89%

Jaccard Co-efficient Analysis with Individual Attack Thresholds

Just as was the case with the simple dissimilarity analysis performed in the previous section, an analysis was also conducted using attack thresholds customized for each user. The results obtained were slightly better than the results obtained from the simple dissimilarity analysis. Just as with the dissimilarity analysis the attack threshold was set at a level to prevent false positives from occurring. The overall average attack detection rate was 93.73%. The standard deviation observed when finding masquerade attacks using the Jaccard Co-efficient was also lower than observed during the comparable dissimilarity analysis. The simple dissimilarity analysis yielded a standard deviation of 9.6 for the Jaccard Co-efficient analysis and a standard deviation of 11.6 when using simple dissimilarity analysis.

The attack detection rates for each user are shown in Table 9. The 95% confidence interval obtained from the data analyzed using Jaccard's Index was $90.19\% \leq \mu \leq 97.26\%$.

Table 9. Individual Performance with Customized Attack Thresholds using Jaccard Index

ID Number	Attack Detection Rate
1	100.00%
2	96.81%
3	95.74%
4	82.27%
5	100.00%
6	100.00%
7	92.20%
8	100.00%
9	69.50%
10	100.00%
11	91.13%
12	100.00%
13	98.23%
14	100.00%
15	95.74%
16	61.35%
17	100.00%
18	93.26%
19	99.29%
20	93.26%

Table 9 - Continued

ID Number	Attack Detection Rate
21	92.91%
22	97.87%
23	100.00%
24	82.98%
25	99.65%
26	76.24%
27	97.87%
28	98.23%
29	100.00%
30	91.13%
31	100.00%

It is interesting to note that only one user had their attack detection rate drop using Jaccard's Index as a measure of similarity when compared to the results obtained by simply analyzing the number of discrepancies between sets. Several users, though, had a dramatic rise in their attack detection rate. Users 7 (49.29% to 92.20%), 4 (48.94% up to 82.27%), and 11 (49.29% up to 91.13%) are several examples of users that experienced dramatic increases of at least 30% in their attack detection rate. Only user 16's attack detection rate fell when analyzed using the Jaccard Index. A comparison of the two methods and the performance for each user is shown in Table 10.

Table 10. Comparison of Dissimilarity and Jaccard Analysis

Overall Average Detection Rate	Simple Dissimilarity Analysis	Jaccard Analysis
	91.34%	93.73%
User ID Number	Simple Dissimilarity Analysis	Jaccard Analysis
1	98.58%	100.00%
2	100.00%	96.81%
3	83.69%	95.74%
4	48.94%	82.27%
5	99.29%	100.00%
6	97.16%	100.00%
7	49.29%	92.20%
8	100.00%	100.00%
9	39.36%	69.50%
10	100.00%	100.00%
11	49.29%	91.13%
12	100.00%	100.00%
13	91.13%	98.23%
14	100.00%	100.00%
15	97.16%	95.74%
16	70.57%	61.35%
17	66.31%	100.00%
18	66.31%	93.26%
19	88.65%	99.29%
20	60.64%	93.26%

Table 10 - Continued

User ID Number	Simple Dissimilarity Analysis	Jaccard Analysis
21	65.60%	92.91%
22	98.23%	97.87%
23	100.00%	100.00%
24	62.06%	82.98%
25	97.16%	99.65%
26	60.99%	76.24%
27	79.79%	97.87%
28	92.91%	98.23%
29	89.72%	100.00%
30	63.12%	91.13%
31	97.87%	100.00%

Impact of Varying Demographics on Authentication

As described in Chapter 3, in the section titled, "Demographical Makeup of the Participant Population", the participants in this study were surveyed and a variety of demographical characteristics were recorded. This data was collected to be used to determine if any particular portion of the experiment population was better suited for using GUI Usage Analysis as a means of authentication. Because no single sub-population contained enough members to be considered statistically significant, the findings presented here are submitted as indicators, not definitive evidence.

The findings presented here are assessed in terms of overall attack detection rate when using the Jaccard Index to analyze and detect masquerade attacks. The majority of the characteristics collected from the participants centered on the suspicion that computer proficiency would positively impact the attack detection rate. It was believed that surveying the participants on a variety of characteristics was the most efficient manner in which to determine computer proficiency, short of administering a potentially biased computer proficiency exam to participants.

Unless otherwise noted, the analyses presented here utilize the data yielded from section the section titled, "Jaccard Co-efficient Analysis with Individual Attack Thresholds". Furthermore, because the number of respondents falling into a particular group was often small, confidence intervals with $\alpha = 0.05$ have also been included to demonstrate how close these results might mirror results obtained from the overall population. Please also note that any categories for which only a single participant was a member were excluded from the following analysis.

Impact of Vocation on Authentication

The data was analyzed to determine the degree to which an individual's vocation affects the attack detection rate when using Jaccard Index and GUI Usage Analysis. It was hypothesized by the author that fields that relied on the use of computers might have higher attack detection rates than members of professions that did not. Only vocations that were indicated on more than three occasions were included in the analysis. The differences in attack detection rates between the varying vocations was judged to be too small to be of any significant consequence. The results are shown in Table 11.

Table 11. Impact of Vocation on Attack Detection

Profession	Num of Respondents	Attack Detection	Confidence Interval
IT	7	93.82%	$84.83\% \leq \mu < 100\%$
Education	12	93.88%	$86.70\% \leq \mu < 100\%$
Engineering	3	97.04%	$84.33\% \leq \mu < 100\%$
Technician	3	97.40%	$86.21\% \leq \mu < 100\%$

Impact of Educational Achievement on Attack Detection

The next demographical characteristic examined was the degree to which educational achievement impacted the attack detection rate. The author hypothesized that individuals with greater educational achievement would possess greater proficiency at using a computer. This belief was based on the anecdotally supported notion that professional people are more likely to require a computer to complete their daily tasks. The results of this analysis are shown in Table 12. The data seems to indicate a slight trend towards higher achievement producing a higher attack detection rate.

Table 12. Impact of Educational Achievement on Attack Detection

Highest Degree Achieved	Num of Responses	Attack Detection	Confidence Interval
Some Undergraduate Coursework	5	97.87%	$93.83\% \leq \mu < 100\%$
Associates Degree	5	88.58%	$72.50\% \leq \mu < 100\%$
Undergraduate Degree	3	94.56%	$77.26\% \leq \mu < 100\%$
Some Graduate Coursework	4	98.05%	$92.91\% \leq \mu < 100\%$
Graduate Degree	11	93.46%	$84.99\% \leq \mu < 100\%$

Impact of Self-Reported Computer Skill on Attack Detection

As previously stated, the author suspected that the greater a user's proficiency at using a computer, the higher their attack detection rate would be. In an effort to determine the accuracy of this hypothesis each participant was surveyed and asked to indicate their proficiency at operating a computer. The results are shown in Table 13. The results show a slight trend indicating that users with above average computer skills enjoy higher attack detection rates.

Table 13. Impact of Self-Reported Computer Skill on Attack Detection

Computer Skill	Num of Respondents	Attack Detection	Confidence Interval
High	12	93.38%	$86.52\% \leq \mu < 100\%$
Above Average	11	96.84%	$93.47\% \leq \mu < 100\%$
Average	6	90.60%	$75.00\% \leq \mu < 100\%$

Impact of Daily Computer Usage on Attack Detection

The average amount of time spent each day using a computer was collected from each participant. Subjects were asked to indicate the amount of in two hour blocks (0-2,

2-4, etc.). The results of this analysis are provided in Table 14. The results show a slight trend indicating that users that spend more time each day using a computer enjoy a slightly higher attack detection rate.

Table 14. Impact of Daily Computer Usage on Attack Detection

Daily Use	Num of Respondents	Attack Detection	Confidence Interval
8+ Hours	9	98.86%	$97.14\% \leq \mu < 100\%$
6-8 Hours	7	88.30%	$75.68\% \leq \mu < 100\%$
4-6 Hours	3	97.40%	$88.4\% \leq \mu < 100\%$
2-4 Hours	10	92.94%	$84.13\% \leq \mu < 100\%$

Impact of Computer Skill Source on Attack Detection

The final piece of information that was solicited from each participant was the source of their computing skills. Participants were asked to indicate where they acquired the majority of their computing skills, either through a course or through self instruction. It was hypothesized by the author that users that had taught themselves to use a computer would be better suited for GUI Usage Analysis and consequently have a higher attack detection rate. The results of this analysis are shown in Table 15. The results are consistent with the expectations and indicate a trend towards users that teach themselves having higher attack detection rates. It is interesting to note that a relatively small number of users indicated that they had obtained their computing skills from an organized course or instruction. It is possible that a user's source of computing skill might be a great indicator of a user's suitability for GUI Usage Analysis but offer little practical value because of the relatively small number that obtained their skills from a course.

Table 15. Impact of Computer Skill Source on Attack Detection

Educational Method	Num of Respondents	Attack Detection	Confidence Interval
Self Taught	23	95.59%	$92.78\% \leq \mu \leq 98.49\%$
Through a course or instruction	6	88.47%	$69.54\% \leq \mu < 100\%$

Impact of Age on Attack Detection

The participants' age was another item that was analyzed to determine the degree to which it could be used as a predictor of suitability for GUI Usage Analysis. It was hypothesized by the author that older users might be less comfortable with the use of a computer, possibly impacting the effectiveness of GUI Usage Analysis. The results of this analysis are provided in Table 16. The results of this analysis were quite interesting. Contrary to expectations, the average attack detection rate increased with the participants' age, though the improvement was rather modest.

Table 16. Impact of Age on Attack Detection

Age	Num of Respondents	Attack Detection	Confidence Interval
18-24	3	91.48%	$69.41\% \leq \mu < 100\%$
25-39	9	93.06%	$84.22\% \leq \mu < 100\%$
40-65	17	95.13%	$90.12\% \leq \mu < 100\%$

In all, it was felt that the demographical findings, while interesting, could only be used to indicate trends. There were simply not enough participants in each category, regardless of the characteristic being examined, to strongly infer how each subpopulation

might actually perform in a laboratory environment. While secondary to the investigations presented here, studies into which particular user groups perform better in GUI Usage Analysis studies might be an area of future study.

Summary of Findings

The results obtained here indicate that GUI Usage Analysis can be used as an effective means of authentication to defend against masquerade attacks. The overall attack detection rate was found to be higher in instances in which customized attack thresholds were used for each user. Furthermore, the attack detection rate was found to be higher when analyzed using the Jaccard Index as opposed to other simple dissimilarity analyses.

Finally, the participant pool was analyzed to determine if a single demographic provided significant insight to the suitability of GUI Usage Analysis for a particular user. The number of participants in each portion of the population was too small to draw any definitive conclusions. The data indicated that users using a computer eight or more hours a day enjoyed higher attack detection rates.

CHAPTER 5

GUI USAGE ANALYSIS AS AN IDENTIFICATION MECHANISM

The ability to correctly spot an attack is always highly prized by administrators. Traditionally, the second part of coping with an attack is recovery from the attack. For many system administrators, part of the recovery process is the ability to examine the evidence and attempt to place blame or responsibility for the attack; a virtual "whodunit". In Chapter 4 a study was presented demonstrating the utility of GUI Usage Analysis as a means of authentication. An authentication system enables someone to determine that an attack is occurring. In order to determine who the attacker was, GUI Usage Analysis must be adapted to serve a different purpose: identification.

Authentication systems can traditionally be thought of as producing two possible outputs: either the current user is authenticated, indicating that the proper user is the one operating the computer, or the current user is not authenticated, indicating an attack is occurring. Identification, on the other hand, is a much more difficult function to construct. Whereas authentication functions returned one of two values (TRUE, FALSE), identification functions will return one of *N* values, where *N* is the number of users known to the system.

This chapter will present the results of two different attempts at performing identification using GUI Usage Analysis data. The first attempt uses the Jaccard Index

presented in Chapter 4 as part of an identification scheme. The second attempt involves the use of Artificial Neural Networks as part of an identification scheme.

Identification Using Jaccard Index

User Identification

As explained in Chapter 4, the Jaccard Index is a mathematical representation of the similarity between two sets of data. The Jaccard Index is defined as:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

The result of the Jaccard Index can then be used to determine which sets are more similar to each other, as was done in Chapter 4.

In Chapter 4 the Jaccard Index was used to authenticate a simulated unknown user using a computer terminal. The result of that operation was binary: the user's claimed identity was either confirmed (the user was authenticated) or the user's claimed identity was rejected. As previously described, identification is a function that can produce up to *N* outputs, where *N* represents the number of users known to the system. As the first step of using GUI Usage Analysis as a means of identification, the Jaccard Index authentication procedure introduced in Chapter 4 was modified.

As previously illustrated, when two sessions are compared to each other using the Jaccard Index a similarity score is generated. The results of the authentication study presented in Chapter 4 indicate that, generally speaking, sessions generated by the same user have higher Jaccard Index values than sessions generated by differing users. Using

this rationale, a study was performed to determine if the Jaccard Index could be implemented as part of an identification study.

To test whether or not the Jaccard Index could be used to identify users from their GUI interaction data, the following algorithm was used:

```
For each user A
        Max_Jaccard_Index = 0.0
        Training = Select_training_sessions(A, 2)
        For each user B
                If A = B
                        Test = Select_test_session(A, Training)
                Else
                        Test = Select_test_session(B)
                End If
                Current_Jaccard_index = Jaccard(Training, Test)
                If Current_Jaccard_index > Max_Jaccard_Index
                        Max_Jaccard_Index = Current_Jaccard_Index
                        Suspected_Identity = B
                End If
        End For
        Print("Suspected Identity: ", B)
End For
```

Figure 9. Pseudo-code of Jaccard Index Identification Function

The algorithm selects a known user *A* and examines each session not present in the training data. The algorithm returns the user *B* that generated the maximum Jaccard Index when analyzed with the training data extracted from *A*. That user is assumed to be the same person as the one that provided the training data. In this instance, the Jaccard Index is used to identify the donor of an unknown sample. It should be noted that every possible combination of known and unknown user was experimented with. Furthermore, all possible combinations of training and test data were also exhaustively tested.

This identification experiment was performed on the same thirty one individuals that participated in the authentication study presented in Chapter 4. Overall, the Jaccard

based identification algorithm successfully identified the unknown user in 77.1% of trials. The raw data resulting from this study is presented in Appendix C. The performance for each individual user (i.e. successful identification rate) is shown in Table 17.

Table 17. Successful Identification Rate by User

User Number	Identification Success Rate
1	100
2	100
3	100
4	33.33
5	100
6	100
7	33.33
8	100
9	33.33
10	100
11	33.33
12	66.66
13	100
14	100
15	100
16	66.66
17	100
18	66.66
19	100
20	66.66
21	100
22	100
23	100
24	33.33
25	66.66
26	0
27	66.66
28	100
29	66.66
30	33.33
31	100

The standard deviation of the individual identification performance is quite large, averaging 30.05. This is due to the fact that only three test sessions were possible from the data being used. Therefore, a single failed identification would reduce the success rate from 100% to 66.66%.

For this reason, it may not be reasonable to say on an individual by individual basis that one user definitively performed better, or that their behavioral patterns more uniquely identify them in comparison to fellow participants. However, when examined as a whole, this study featured 93 different trials, with successful identification occurring in 77.1% of the cases. It is reasonable to conclude that this number is probably highly comparable to the overall identification success rate that would be experienced in the general population.

Demographic Analysis

Just as was the case when GUI Usage Analysis was analyzed as a means of authentication, an analysis was conducted to determine if any single user trait predisposed users to enjoying more success with GUI Usage Analysis. Also, it must again be stated that the findings here are simply indications of possible trends. The number of participants in each group is too small to draw any definitive conclusions regarding each trait's impact on the use of GUI Usage Analysis as a means of identification.

The findings presented here are assessed in terms of overall successful identification rate when using the Jaccard Index to analyze and detect masquerade attacks. The majority of the characteristics collected from the participants centered on

the suspicion that computer proficiency would positively impact the attack detection rate. It was believed that surveying the participants on a variety of characteristics was the most efficient manner in which to determine computer proficiency, short of administering a potentially biased computer proficiency exam to participants.

The results presented here are based on the number of times members of a particular group were successfully identified based on their GUI Usage patterns. Confidence intervals with $\alpha = 0.05$ are provided in an effort to indicate what the attack detection rate for the entire sub-population might be. Finally, please note that any category which had one or fewer respondents were not included in these analyses. Almost universally, the standard deviation amongst respondents of a particular sub-group was simply too large to make any reliable inferences regarding the likelihood of using GUI Usage Analysis as a reliable means of identification within some sub-group of the population. The resulting analyses do show trends in the data, but these must also be taken with a proverbial "grain of salt".

Impact of Vocation on Authentication

The data was analyzed to determine the degree to which an individual's vocation affects the identification rate when using Jaccard Index and GUI Usage Analysis. The author hypothesized that fields that relied on the use of computers might have higher identification rates than members of professions that did not. Only vocations that were indicated on more than three occasions were included in the analysis. In total the difference in identification rate between members of the varying professions was judged by the author to be relatively small, particularly given the few number of participants that

listed their vocation as “Engineering” or “Technician”. The results are shown in Table 18.

Table 18. Impact of Vocation on Identification Rate

Profession	Num of Respondents	Identification Rate	Confidence Interval
IT	7	71.43%	$33.96\% \leq \mu < 100\%$
Education	12	83.33%	$69.05\% \leq \mu \leq 97.61\%$
Engineering	3	66.66%	$0\% < \mu < 100\%$
Technician	3	77.78%	$0\% < \mu < 100\%$

Impact of Educational Achievement on Identification Rate

The next demographical characteristic examined was the degree to which educational achievement impacted the identification rate. The author hypothesized that individuals with greater educational achievement would possess greater proficiency at using a computer. This belief was based on the anecdotally supported notion that professional people are more likely to require a computer to complete their daily tasks. The results of this analysis are shown in Table 19. The data seems to indicate a slight trend towards higher achievement producing a higher identification rate. In particular, the largest gap seems to be between individuals with at least some training at a traditional four year institution.

Table 19. Impact of Educational Achievement on Identification Rate

Highest Degree Achieved	Num of Responses	Identification Rate	Confidence Interval
Some Undergraduate Coursework	5	80.00%	$42.98\% \leq \mu < 100\%$
Associates Degree	5	60.00%	$14.66\% \leq \mu < 100\%$
Undergraduate Degree	3	77.78%	$0\% < \mu < 100\%$
Some Graduate Coursework	4	91.67%	$65.14\% \leq \mu < 100\%$
Graduate Degree	11	78.79%	$58.08\% \leq \mu \leq 99.48\%$

Impact of Self-Reported Computer Skill on Identification Rate

As previously stated, it was suspected that the greater a user's proficiency at using a computer, the higher their rate of successful identification would be. In an effort to determine the accuracy of this hypothesis each participant was surveyed and asked to indicate their proficiency at operating a computer. The results are shown in Table 20. Unlike the results observed during the authentication study, self-reported computer skill seems to have had a slightly larger impact on the identification rate of individuals.

Table 20. Impact of Self-Reported Computer Skill on Identification Rate

Computer Skill	Num of Respondents	Identification Rate	Confidence Interval
High	12	75.00%	$52.64\% \leq \mu \leq 97.35\%$
Above Average	11	81.82%	$66.42\% \leq \mu \leq 97.21\%$
Average	6	72.22%	$37.82\% \leq \mu < 100\%$

Impact of Daily Computer Usage on Identification Rate

The average amount of time spent each day using a computer was collected from each participant. Subjects were asked to indicate the amount of in two hour blocks (0-2, 2-4, etc.). This data was then analyzed to determine if there might be any evidence linking daily computer usage to successful identification rates. The results of this analysis are provided in Table 5. Unlike the results observed during the authentication study, the data gathered here indicates that identification success seems to be dependent on either high daily usage or low daily usage. Users that utilized computer systems for intermediate amounts of time on a daily basis appear to enjoy reduced identification rates, particularly when accounting for the fact that only three individuals indicated between four and six hours of use daily (and thus might represent an aberration). It is possible that the amount of daily computer usage corresponds to overall levels of actual computer proficiency (contrast with *self-reported* proficiency described in the previous section). In this instance it may be the case that being either an expert user or a novice user may result in distinctive patterns, generated by either copious computer knowledge or a dearth of computer knowledge. The results are shown in Table 21.

Table 21. Impact of Daily Computer Usage on Identification Rates

Daily Use	Num of Respondents	Identification Rate	Confidence Interval
8+ Hours	9	92.59%	$81.29\% \leq \mu < 100\%$
6-8 Hours	7	52.38%	$17.42\% \leq \mu \leq 87.34\%$
4-6 Hours	3	88.89%	$39.95\% \leq \mu < 100\%$
2-4 Hours	10	76.66%	$57.03\% \leq \mu < 96.29\%$

Impact of Computer Skill Source on Identification Rates

Information was solicited from each participant regarding the source of their computing skills. Participants were asked to indicate where they acquired the majority of their computing skills, either through a course or through self instruction. The author hypothesized that users that had taught themselves to use a computer would be better suited for GUI Usage Analysis and consequently have a higher identification rate. The results of this analysis are shown in Table 22. Unlike the authentication study, the results do not show much of a trend between the source of computer usage and identification rates, though the relatively small number of respondents indicating they obtained their skills through a course or instruction makes it impossible to draw definitive conclusions.

Table 22. Impact of Computer Skill Source on Identification Rates

Educational Method	Num of Respondents	Identification Rate	Confidence Interval
Self Taught	23	76.81%	$63.65\% \leq \mu \leq 90.36\%$
Through a course or instruction	6	73.33%	$44.05\% \leq \mu < 100\%$

Impact of Age on Identification Rates

The participants' age was another item that was analyzed to determine the degree to which it could be used as a predictor of suitability for GUI Usage Analysis. The author hypothesized older users might be less comfortable with the use of a computer, possibly impacting the effectiveness of GUI Usage Analysis. The results of this analysis are provided in Table 23. The results of this analysis were quite interesting. Contrary to expectations, the identification rate increased with the participants' age, though the improvement was rather modest. It is suspected that a larger population would yield results indicating that a user's age offers no indication of his/her suitability for GUI Usage Analysis.

Table 23. Impact of Age on Identification Rates

Age	Num of Respondents	Identification Rate	Confidence Interval
18-24	3	55.55%	$0\% \leq \mu < 100\%$
25-39	9	81.48%	$52.52\% \leq \mu < 100\%$
40-65	17	78.43%	$65.87\% \leq \mu < 90.98\%$

Analysis Using Artificial Neural Networks

In an effort to further confirm the findings of the Jaccard Index experiment, another identification experiment was performed. This experiment made use of a form of machine learning known as artificial neural networks, commonly referred to as "neural networks". Neural networks were selected for experimentation because of their history

as a proven machine learning technique capable of analyzing many different problems in a wide variety of problem domains (Mathworks Corporation 2007). A diagram showing a conceptual neural network may be found in Figure 10.

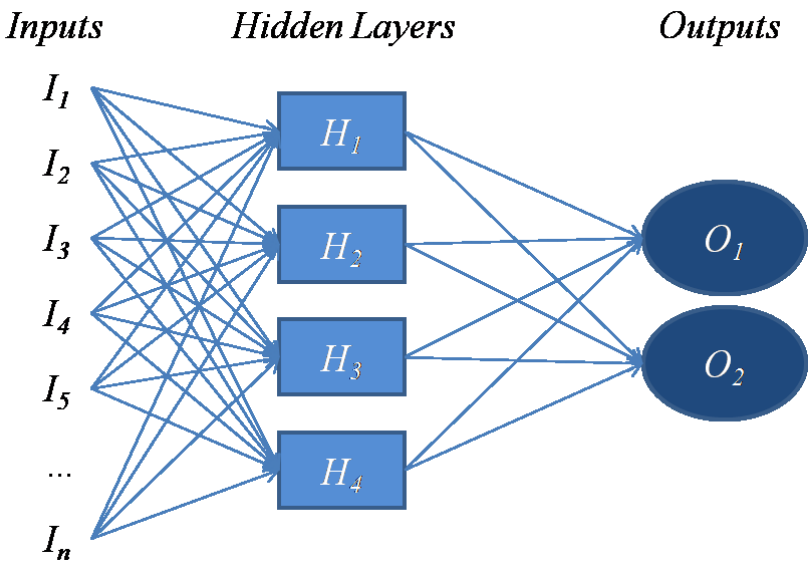

Figure 10. Conceptual Diagram of a Neural Network with *N* Input Units, four Hidden Units, and two Ouputs

Several challenges were encountered in the adaptation of neural networks to the data collected in this study. For example, neural networks are typically designed to accept multiple numerical values in parallel as inputs. This posed an implementation challenge in this instance, since the data gathered in this study was more analogous to a corpus of text. Furthermore, when neural networks are used to analyze text documents, the input is frequently categorized by word frequency (i.e. how often each word occurs in the document). Often the number of distinct words in a document is a relatively small number that the neural network can cope with. Applying this model to the data gathered in this study, each event (action, control, executable) becomes analogous to a word in a text document. Unfortunately, this precise implementation method was not practical for

use in this study. In this study, tens of thousands of distinct events were observed, translating to tens of thousands of "words" and resulting in the neural network needing to accommodate the same number of parallel inputs. This is generally considered to be too many inputs for a single neural network to process effectively, leading to a condition known as *overfitting* (Russell and Norvig 2003). Overfitting occurs in instances in which the number of inputs to a neural network greatly outpaces the amount of training data provided to the network. A neural network that has been overfitted has been trained not only to recognize the important characteristics in the data, but also ends up "memorizing" the unimportant data (noise).

To illustrate the problem of overfitting, consider the following simple example. Suppose a neural network has been designed to identify graduate students in Computer Science. The network has been configured to accept inputs concerning the following observable qualities of a person:

- Height
- Weight
- Age
- Gender
- Ethnicity
- Hair color
- Clothing style
- Build (skinny, stocky, etc.)

The neural network is then presented with three examples of Computer Science graduate students to be used as training data. Following the training period the network is presented with its first test case. The network then proceeds to classify the first test case incorrectly, labeling an English major as a CS graduate student. Inspection of the data yields the problem: the test case, like all of the training data, was male. In this overly simple case, overfitting caused the network to incorrectly believe that being male was the most important factor in identifying Computer Science graduate students.

The workaround crafted for this problem still relied on a count of the number of times some object was involved in an event. Instead of considering an event as an atomic unit, though, events were instead broken up into their smaller components. This yielded a neural network structure that was given the following inputs:

- # of times a user action (left click, double click, 'A' press, etc.) was observed
- # of times a particular type of control (scroll_bar, text_box, etc.) was observed
- # of times a particular process (Winword.exe, Iexplore.exe, etc.) was observed

The number of distinct inputs being passed to the neural network was still quite large (over one thousand) but the network seemed capable of coping with this volume of input sources.

The neural network implementation used in this study was the Artificial Neural Network Toolbox (ANN) distributed as part of the Matlab numerical analysis suite (Mathworks Corporation 2007). The Neural Network Toolbox was relatively robust and easy to use. Furthermore, it is believed that the use of pre-written, commercially available analysis packages prevented coding errors that may have occurred in the

implementation of a relatively complex algorithm such as back-propagation neural networks.

Other than specifying the number of hidden processing layers contained in the neural network, the recommended default settings provided by the Matlab developers were used in this study. The neural network that was developed contained one hidden processing layer of 45 neurons. The number of neurons used in the final experiment was derived through experimentation. Using more than 45 neurons resulted in slightly enhanced accuracy at the cost of much longer processing times. The use of fewer than 45 neurons resulted in significantly degraded accuracy with only modest processing time savings. The neural network featured *N* outputs, with one output corresponding to each participant in the sample.

The network was constructed by randomly selecting two data sessions from each user for use as training data. The remaining data session was set aside to be used as test data. The neural network was trained for five hundred epochs. This number was arbitrarily selected, though this did not appear to matter. Training of the network was aborted by the Matlab package prior to reaching the training limit when the performance of the network hit a proverbial ceiling (i.e. the network was trained to its fullest possible extent).

The performance of the neural network was found to be inferior to the performance of the Jaccard Index based algorithm. Using the neural network, only 12 of 31 participants were correctly identified, yielding a successful identification rate of 38.7%. This compares quite poorly with the successful identification rate of the Jaccard Index based identification algorithm described earlier in this chapter. It is believed that

overfitting was the primary cause of the poor performance, resulting from the large disparity between the number of training cases (62) and the number of inputs (3185).

One of the interesting aspects of the design of neural networks is that all output units produce output on all input cases. This means, in the instance of this output unit, that all *N* output units yield a number between 0 and 1 after each test. This number corresponds to the similarity that the network perceived between the test sample and the reference sample provided for that individual.

On a practical level this structure allows an investigator to easily determine which user the network incorrectly identified in instances in which its analysis produced an incorrect result. Furthermore, this structure allows for determine how many more "guesses" would have been required before correctly identifying the right user. The results of this type of analysis are presented in Table 24. The column labeled "User Rank" indicates the number of guesses or attempts that the network required prior to correctly making the identification. Stated differently, this column represents the number of users that were incorrectly identified by the network prior to arriving at the correct result.

On average, the neural network did not correctly determine the identity of the unknown user until the eighth attempt. Because the neural network faired poorly in identifying the unknown user, demographic data was not analyzed in conjunction with neural network performance. It was determined that such an analysis would not yield noteworthy findings.

Table 24. Average User Rank in Identification with Neural Network

User ID	User Rank
1	3
2	4
3	1
4	30
5	16
6	1
7	8
8	12
9	21
10	1
11	21
12	6
13	7
14	1
15	1
16	4
17	1
18	9
19	1
20	11
21	3
22	1
23	1
24	11
25	1
26	15
27	11
28	11
29	1
30	1
31	24

Summary of Findings

This chapter has presented the results of a study of using GUI Usage Analysis data as part of an identification scheme. The results indicate that Jaccard Index based identification algorithms perform with reasonable accuracy, correctly identifying the

unknown user in 77% of cases. This level of accuracy, while certainly insufficient for legal uses, arguably offers enough accuracy so that its findings can be used in conjunction with other evidence in an effort to identify an attacker. The artificial neural network, though, was not able to correctly interpret the information often enough, and as a result, performed much worse than its Jaccard based counterpart. The results of this analysis indicates that the Jaccard method was much better suited to handle the type and quantity of data gathered in this study.

CHAPTER 6

COMPARISON TO OTHER PUBLISHED TECHNIQUES

As discussed in Chapter 2, many studies have been conducted seeking to study the use of a variety of user traits as possible means of either identification or authentication. Unfortunately many of the traits studied in earlier studies are better suited for command-line driven interfaces, not modern graphical user interfaces. For example, the effectiveness of keystroke dynamics as a means of authentication or identification is inherently linked to the amount of time a user spends typing.

Efforts to profile users based on their interactions with modern computing environments first centered around the users mouse movements. Most likely inspired by the previously published studies focusing on users keyboard patterns (i.e. keystroke dynamics), these studies sampled several traits seen by users using the mouse. Two separate studies have been published analyzing the utility of using mouse movements as a form of authentication (Garg, et al. 2006), (Pusara and Brodley 2004). Both of these studies will be examined here, with the published results being compared to the results of the authentication experiments presented in this dissertation. Unfortunately a direct comparison to either study is not possible since the type and quantity of data gathered for the experiments presented here are incompatible with the data analysis techniques used in either of the other studies.

Comparison to Pusara & Brodley's Mouse Movement Profiling

In (Pusara and Brodley 2004), Pusara and Brodley present a technique for conducting "re-authentication" by studying a user's mouse movements. Their technique performed authentication by comparing several traits extracted from an unknown user's mouse movements to a reference profile associated with the legitimate user. Pusara and Brodley's work showed promise, though the application of their technique was ultimately limited to users who made heavy use of the mouse.

Pusara and Brodley organized an experimental sample of 18 undergraduate college students. They employed a monitoring package that recorded the following information during the user's session:

- Mouse location (sampled every 100 ms)
- Number of left clicks
- Number of right clicks
- Number of double-clicks
- Number of "non-client area" clicks
- Time of each event (mouse movement, mouse clicks, etc.)

Pusara and Brodley's participants were observed while reading the same set of webpages. As a result, the participants interacted exclusively with the Internet Explorer web-browser.

The aforementioned traits were analyzed and several features were extracted. The overall number of each of the traits was calculated. Furthermore, the mean, standard deviation, and third moment were calculated for the distance, angle, and speed observed

between events. Finally, the mean, standard deviation, and third moment were all calculated between cursor locations (as sampled every 100 ms by the monitoring software).

Pusara and Brodley's participants were asked to submit to data collection one time. The participants were monitored until 10,000 observations were recorded by the monitoring software. This single set of data was then divided up into multiple subsets which were used as either training data or test data.

The data gathered was fed into the commercial decision tree classification algorithm known as See5 (Rulequest Research n.d.). The resulting decision tree was used to perform an authentication study over the entire population. The results of Pusara and Brodley's initial experiment are presented in Table 25, along with the results of the Jaccard analysis presented in Chapter 4.

Table 25. Comparison of GUI Usage Analysis and Initial Mouse Movement Analysis

Study	Sample Size	False Positive Rate	False Negative Rate
GUI Usage Analysis	31	0%	6.27%
Pusara and Brodley's Mouse Movement Analysis	18	27.5%	3.06%

Following this initial analysis, Pusara and Brodley reached the conclusion that many of the false positives observed were due to a subset of their experimental population. This subset was found to generate fewer mouse events than other members of the experimental population. For this reason, this subset of 7 users was excluded from further analysis by Pusara and Brodley.

Following this initial experiment, Pusara and Brodley began experimenting with several attributes of the decision tree algorithm in an effort to tune the algorithm to the

performance of each individual user. Pusara and Brodley customized both the alarm threshold for each user as well as the window size. The alarm threshold represented the number of anomalous events that the system could observe before concluding that an attack was under way. The window size represented the period, or number of consecutive events, that were analyzed on an experimental run. The aforementioned exclusion of certain participants combined with the tuning of algorithm attributes resulted in improved performance. These improved numbers are provided in Table 26.

Table 26. Comparison of GUI Usage Analysis and Revised Mouse Movement Analysis

Study	Sample Size	False Positive Rate	False Negative Rate
GUI Usage Analysis	31	0%	6.27%
Pusara and Brodley's Mouse Movement Analysis (Revised Decision Tree)	11	0.43%	1.75%

Several of the attributes generated by the monitoring software used by Pusara and Brodley were not able to be reconstructed using the data gathered in this study. This was primarily due to the different objectives of the two studies: analyzing mouse movements (Pusara and Brodley) versus analysis of overall interaction with the user interface using both keyboard and mouse (GUI Usage Analysis). More specifically, the data used in the experiments presented in this dissertation was collected with the intention of analyzing GUI Usage patterns, not reconstructing previously published studies.

As an example of data that was neither collected nor derivable, consider the "non-client area events" utilized by Pusara and Brodley. Pusara and Brodley calculated the number of mouse events that occurred in the menu bar portion of Internet Explorer,

events that they termed “non-client area events”. This data was not gathered by the monitoring software used to collect the data presented in this dissertation, simply because it was not believed to be necessary to study the effectiveness of GUI Usage Analysis. Furthermore, the “non-client area events” could not be derived from the data that was collected because the cursor coordinates associated with each observed event were absolute, not relative to the current window. To accurately recreate the type of data used by Pusara and Brodley, the monitoring software would have needed to record the screen location of each window. This data was not recorded. Other difficulties also existed, most notably the amount of data gathered from each user. Pusara and Brodley’s users only used Internet Explorer. The participants in the GUI Usage Analysis study also used Internet Explorer, though only for a portion of the experiment.

The results produced by each method are relatively comparable, though any comparisons between the results presented in this dissertation and Pusara and Brodley’s results are admittedly weak. Pusara and Brodley’s approach detected 6.9% more attacks but also registered 0.43% more false positives. Of more importance than the simple performance statistics, though, is the fact that Pusara and Brodley’s method was not reported to be effective for the entire user population. Finally, as previously stated, the relatively small sample size used by Pusara and Brodley makes it difficult to draw any definitive conclusions.

Comparison to Garg’s Mouse Movement Profiling

In (Garg, et al. 2006) Garg presents the results of a study that was very similar to the study presented by Pusara and Brodley. Garg also sought to use GUI interactions as a

means of performing user authentication. Much of the analysis performed by Garg is very similar to the Pusara and Brodley analysis, with some differences.

One difference that was interesting to note between Garg's work and the work of Pusara and Brodley was that Garg's monitoring software initially captured keystroke data. The keystroke data that was captured did not appear to be used in the final analysis in any way, nor was an explanation provided for why it was not included. Still, this is interesting as it seems to indicate that Garg initially considered keyboard input when organizing his study.

Garg and his team captured the following traits from his sample users:

- Mouse Clicks (left click and right click)
- Distance between events
- Mouse speed
- The angles between succeeding events

In addition to those raw events, Garg also calculated the mean and standard deviation for all of the aforementioned raw features. It is interesting to note that Garg and Pusara and Brodley eventually used almost identical feature sets when conducting their analysis (Pusara and Brodley's paper was referenced by Garg). Finally, whereas Pusara and Brodley used the commercially distributed See5 decision tree classifier, Garg chose to support vector machines (SVM) as to provide machine learning functionality.

The results reported by Garg are presented in Table 27. The results of the GUI Usage Analysis authentication study are also presented in Table 27 for the sake of comparison.

Table 27. Comparison of GUI Usage Analysis and Garg's Mouse Movement Analysis

Study	Sample Size	False Positive Rate	False Negative Rate
GUI Usage Analysis	31	0%	6.27%
Garg et al.'s Mouse Movement Analysis	3	Not Reported	3.85%

One of the most striking characteristics of Garg's research is the lack of a reported false positive rate. The reader is left to draw their own conclusions regarding this omission. The other striking quality of Garg's research is the sample size of three participants. Such a small sample size makes it extremely difficult to draw any definitive conclusions from the results. It is suspected that, if a larger sample were obtained, results matching Pusara and Brodley's might ultimately be obtained by Garg. This belief is based on the striking similarity of the two feature sets that the studies employed. It is believed that the performance between the two machine learning techniques employed (SVM vs. decision tree learning) would ultimately have resulted in only small discrepancies in false positive and false negative rates. When comparing Garg's research to the results of the GUI Usage Analysis study presented here, the numbers are relatively similar, with Garg's technique boasting a slightly higher attack detection rate and his false positive rate remaining a mystery. Unfortunately, as was the case with the study by Pusara and Brodley, the comparisons between Garg's results and the results yielded by GUI Usage Analysis are relatively weak. There are simply too many environmental variables at play to draw definitive conclusions between the two studies.

Ultimately Garg's research was not reproduced here because it was deemed to be scientifically invalid. It is included in this discussion primarily because it represents one of only two known papers attempting to authenticate users based on how they interact

with a modern GUI driven computer system. Careful inspection of Garg's study yields several problems. First among these are the tasks that Garg allowed his participants to perform. Unlike both Pusara and Brodley's study as well as the GUI Usage Analysis studies presented in this dissertation, Garg did not require his participants to complete a uniform task list. Failure to take this step resulted in an experiment with multiple variables: different users completing different tasks. It is impossible to conclusively determine whether the support vector machine in Garg's study was able to differentiate between users based on their interaction patterns or the tasks they performed. To further illustrate this problem, suppose user A is a student working to complete their final thesis prior to graduation, while user B is a marketing professional. User A will conceivably spend much of their energies preparing their manuscript in a word processor, while User B may spend a great deal of their time performing market research on the Internet. It is entirely possible, if not likely, that User A may make much heavier use of the keyboard, not out of a behavioral trait, but simply because of the work they happened to be tasked with at that given moment in time.

Summary of Comparative Findings

As previously stated, it is ultimately believed that the performance of Garg's technique might prove to be very comparable to the results achieved by Pusara and Brodley. It is also believed that Garg's method might ultimately encounter the same difficulties that Pusara and Brodley encountered: the discovery that mouse movement analysis is only effective for a portion of the overall computer using population. In comparison, the results of the GUI Usage Analysis study did not indicate a need to exclude any user from the study. Certainly the technique may be more effective for some

users, but the findings presented here indicate that all users would enjoy some protection from GUI Usage Analysis as a means of authentication. It is worth noting, however, that users who are suited for Pusara and Brodley's technique may enjoy slightly higher attack detection rates. At the same time, it should be noted once again that Pusara and Brodley's experiment focused only on the interactions of users operating a web browser. While speculative, it is not unreasonable to think that the performance of Pusara and Brodley's system might degrade if their techniques were applied to all activities performed on a computer, as was the case in the GUI Usage Analysis studies.

The findings of the studies presented in this dissertation offer one additional finding that might prove to be important in future research. The experiment conducted in this dissertation gathered samples from the participants at different chronological times. Pusara and Brodley's study, by comparison, gathered all of its data from the participants during a single session. The research presented in this dissertation demonstrates that user behavior is consistent over a longer period of time, and between activity sessions.

A side by side comparison of all three studies is presented in Table 28.

Table 28. Comparison of all Presented Techniques

Study	Sample Size	False Positive Rate	False Negative Rate
GUI Usage Analysis	31	0%	8.66%
Pusara and Brodley's Mouse Movement Analysis (Revised Decision Tree)	11	0.43%	1.75%
Garg et al.'s Mouse Movement Analysis	3	Not Reported	3.85%

CHAPTER 7

POTENTIAL VULNERABILITIES OF GUI USAGE ANALYSIS

All common access control schemes have practical weaknesses that make the systems they protect vulnerable to attack. For example, traditional password based authentication schemes have been shown to be vulnerable in a variety of ways:

- The password may be forgotten
- The user may write the password down making it vulnerable to theft
- The password may be stored in cleartext making it vulnerable to hackers
- The password might be compromised through dictionary and brute force attacks

New access control methods also possess vulnerabilities. Biometric systems may experience high false positive rates, denying access to legitimate users. An attacker that can exploit this trait can use it as a form of a denial of service attack. Some biometric systems are also susceptible to forged credentials – consider finger print readers, some of which have extensively documented weaknesses to fake fingerprints.

Given these facts, it would be naïve to assume that GUI Usage Analysis is not similarly vulnerable to some attack. This chapter presents discussion designed to illuminate some manners in which GUI Usage Analysis might be vulnerable. A great deal of this discussion is hypothetical in nature. Demonstration of any of the potential vulnerabilities outlined here is outside the scope of this document.

GUI Usage Analysis, as presented here, is envisioned as being used in defensive manners, protecting critical systems from masquerade attacks. If this vision is correct, it is reasonable to conclude that any attempt to defeat GUI Usage Analysis would most likely occur in cases in which a single person was attempting to access a computer system they were not intended or authorized to access.

Because of the relatively specific nature of the defense offered by GUI Usage Analysis, the potential attacks are also fairly specific. It is believed that any attempt to subvert GUI Usage Analysis would have to take the form of one user that attempts to behave like another. In other words, an attacker would almost certainly have to attempt to impersonate the victim in order to defeat GUI Usage Analysis.

The studies presented in this dissertation have considered two different defensive uses of GUI Usage Analysis: as a means of authentication and as a means of identification. The remainder of this chapter is divided accordingly, with section 7.1 focusing on authentication vulnerabilities and section 7.2 focusing on possible identification vulnerabilities.

It should be noted that the vulnerabilities described here may not represent the complete set of vulnerabilities found in any actual system implementing GUI Usage Analysis. Furthermore, it is also worth noting that the single greatest vulnerability faced by GUI Usage Analysis would almost certainly be a system that fails to perform accurately. Just as is the case with other intrusion detection systems, a high number of false positives will ultimately result in the system either being disabled or desensitized, most likely causing an increase in an attacker's success rate.

Vulnerabilities of GUI Usage Analysis as an Authentication Scheme

Chapter 4 presented the results of a study in which GUI Usage Analysis was analyzed to determine its utility as an authentication scheme. The results of this study indicated that GUI Usage Analysis could be used to accurately authenticate a user of a system. Because authentication is a function that produces only two possible outputs (legitimate user vs. potential attacker) the methods in which GUI Usage Analysis are similarly constrained. The only objective for an attacker would be to convince the system that they were the legitimate user.

It is believed that an attacker could convince the system that they were the legitimate user in one of two manners. First, the attacker might observe the targeted user for some period of time, making observations about the victim's mannerisms and methods of interaction. The data gathered by the studies presented here do not offer any indication of what the likelihood of success for such attempts might be.

The other alternative is that the attacker can rely on luck and hope that their own interaction patterns closely resembled the interaction patterns of the victim. If the attacker's interaction patterns truly resembled the patterns of the victim it is possible that the system might mistake the attacker for the victim. The data gathered by the studies presented here indicate that the odds of this happening are low, with the likelihood of this form of attack succeeding being 5.7%. It is worth noting, though, that it is conceivable that real-world systems may possess lower accuracy, thus bettering the attacker's chances.

Vulnerabilities of GUI Usage Analysis as an Identification Scheme

Any organization that attempted to base a defense system around GUI Usage Analysis would most likely attempt to implement this scheme not only as an authentication system, but also as a system capable of making identifications. Being able to prevent masquerade attacks and other unauthorized computer use is one thing; being able to identify the attacker is another, more tempting possibility. Unfortunately implementing GUI Usage Analysis as an identification scheme offers additional vulnerabilities not present when considering this technique as an authentication system.

As previously stated, authentication functions produce only two possible outputs: the user is either authenticated or (s)he is rejected and their access denied. Identification, on the other hand, is a function that can produce *N* outputs, where *N* represents the number of users known to the system. This key difference might potentially allow a defensive technology to be used offensively by an attacker.

First, an identification system can be attacked by impersonation in the same manner that an authentication system can. In other words, an attacker can still attempt to make the system believe that they are the legitimate user. However, unlike authentication systems, identification systems are vulnerable to other forms of attack. In this example of a hypothetical attack, if the attacker does not successfully impersonate the legitimate user the system will conceivably take action to protect itself. It stands to reason that if an identification system has determined that the current user is an attacker/impostor, the system must also have made some determination about the real identity of the attacker (otherwise the system would be performing simple authentication). This is where another

way to defeat the system can be introduced: misidentification. If the system does not correctly identify the impostor, then the attacker has successfully implicated someone else in their crime. It stands to reason that an attacker might even use this form of attack as a way to frame a rival in the hopes that they (the rival) might be punished.

CHAPTER 8

SUMMARY OF FINDINGS AND IMMINENT FUTURE RESEARCH

This dissertation has presented findings from two different studies as well as provided a comparison to published performance statistics for other GUI based behavioral profiling models. A review of these findings is presented here as well as research that will be performed in the imminent future. Chapter 9 presents long term research objectives and plans based on the findings made here.

Utility of GUI Usage Analysis as an Authentication Scheme

GUI Usage Analysis was examined for its utility as a means of authenticating users and preventing masquerade attacks. It was determined that GUI Usage Analysis was an effective authentication scheme in laboratory settings. Sessions from a known user were compared to a session from an unknown user and the similarity was calculated. The similarity was calculated using two different algorithms: TF-IDF ("term frequency – inverse document frequency") and Jaccard Similarity.

Two different methods of determining an attack were experimented with: a static threshold and a variable threshold. In experiments using a static threshold, a single value was used for all users to determine whether an attack had occurred. In experiments using a variable threshold, the threshold used for labeling a session as being attack was customized for each user.

Ultimately the Jaccard coefficient was found to be the most effective at correctly identifying attacks. Not surprisingly, using a variable attack threshold also provided the greatest accuracy. The combination of using Jaccard coefficient with a variable attack threshold resulted in a false positive rate of 0% and a false negative rate of 8.66%.

Finally, the performance of each participant was analyzed in relation to their responses to survey data. This analysis was performed in an effort to see if there were any behavioral traits that could serve as a predictor of attack detection rate. While the number of participants in each statistical group was too small to make any definitive conclusions, the data gathered here indicates that users who work with a computer system eight or more hours a day enjoy a higher attack detection rate.

Utility of GUI Usage Analysis as an Identification Scheme

Following the study investigating the use of GUI Usage Analysis as an authentication scheme, a second study was performed. This study sought to determine the suitability of GUI Usage Analysis as a means of identification. The results of the study indicate that, in laboratory settings, GUI Usage Analysis can be used as a means of identification with reasonable confidence.

The use of GUI Usage Analysis as an identification scheme was investigated using two different approaches. The initial approach used the Jaccard coefficient as introduced in Chapter 4. Each unknown session was compared to the reference sample for each user in an effort to determine which user had provided the unknown sample. The reference sample that maximized the Jaccard coefficient was considered to be the

supposed donor of the unknown sample. This method resulted in correct identification in 77.1% of all trials.

Following the use of the Jaccard coefficient as a means of identification, a traditional machine learning algorithm was used. Artificial Neural Networks (ANNs) have been used in many different problem domains to learn and predict a wide variety of functions. An artificial neural network was constructed using the Matlab Artificial Neural Network toolbox. The network was trained using two sessions from each user. The remaining sessions from each user were then passed to the network. The network correctly identified the donor of the unknown sample in 39% of trials. It is believed that the poor performance of the neural network in comparison to the experiment using the Jaccard coefficient was most likely due to overfitting of the neural network.

Comparison between GUI Usage Analysis and Other Published Techniques for Authentication

Prior to conducting the studies presented in this dissertation, two studies were published demonstrating the effectiveness of using mouse movements as a means of authentication. The results of the Jaccard coefficient analysis with a variable attack threshold were compared to the results published in both studies. Though a definitive comparison of attack detection methods was not possible, some hypotheses were formed based on the results of those comparisons.

The first paper, written by Pusara and Brodley, had 18 participants completing a single session of viewing webpages. The data gathered from each user was divided into sub-sessions which was then fed into a decision tree learning algorithm. Pusara and

Brodley initially excluded seven of their participants from their final results because these users did not make heavy use of the mouse. The results obtained from the remaining eleven participants were relatively comparable to the results obtained using GUI Usage Analysis. Pusara and Brodley's technique resulted in a false positive error rate of 0.43% and a false negative error rate of 1.75%. Contrast these results with the results obtained from the GUI Usage Analysis method, which resulted in a false positive error rate of 0% and a false negative error rate of 8.66%. While Pusara and Brodley's stated performance would generally be regarded as superior to the performance of GUI Usage Analysis, it is worth pointing out that Pusara and Brodley's method was not found to be usable on approximately 40% of the participant pool.

Garg et al. published a second study that was very similar to the study published by Pusara and Brodley. Garg and his team extracted a virtually identical feature set to the one extracted by Pusara and Brodley, with Garg opting for support vector machine (SVM) analysis instead of the decision tree algorithm chosen by Pusara and Brodley. It was demonstrated that the scientific validity of Garg's study was somewhat suspect, making any comparison somewhat ill-advised.

Imminent Research Objectives

Following completion of the studies presented here, subsequent studies are planned that will seek to refine the experimental processes developed for this dissertation. New participants will be recruited and will be asked to supply more sessions than the participants in this study were asked to. Participants will also have data collected over

multiple days, as opposed to the single day collection period used in the studies presented here.

It is very possible that the participants may require compensation in some form to ensure greater cooperation than what was achieved in these studies. As previously described, the participants in these studies received no compensation. It is believed that the lack of compensation harmed retention efforts, leading to many participants failing to complete the study satisfactorily.

The ultimate goal of the additional data is to analyze several variables that were not considered in these studies. For example, it should be determined how consistent a user's behavior is over a longer period of time (days instead of hours). Also, additional data is required in order to accurately recreate the other published studies in the area of GUI based authentication. Finally, a second set of users will serve to validate the findings presented here.

CHAPTER 9

FUTURE APPLICATIONS OF THIS RESEARCH

New technologies and discoveries are of little value if they cannot be applied to real-world problems and needs in a meaningful way. Though the original researcher may find the data and results to be of some value by themselves, if the findings cannot impact the general populace in some positive way it can be argued that the research has ultimately been fruitless. This chapter will outline some ways in which GUI Usage Analysis could possibly be applied in order to produce technologies that will serve some purpose.

Information Assurance Applications

When GUI Usage Analysis was initially conceived, it was thought of as a potential defensive technology. GUI Usage Analysis was originally targeted as a means of helping to secure computer systems against one of the most difficult types of attack, the previously described masquerade attack on unattended, unlocked workstations. The research presented in this dissertation shows that masquerade attacks may be detected using GUI Usage Analysis in a controlled environment. This research indicates that GUI Usage Analysis can be used as both a means of identification and authentication with reasonable rates of success. As previously stated, though, these findings are of little value to the general population without a vision of how to apply these new discoveries.

Masquerade Attack Research

One of the unfortunate truths regarding masquerade detection research is that, to date, no comprehensive study has been performed to determine exactly how many masquerade attacks typically occur. Stated differently, information security researchers have no empirical data that can be used to indicate exactly how much of a threat masquerade attacks actually pose to the typical organization. The ability to generate some amount of empirical evidence to fill this knowledge vacuum is certainly one valuable application of this research.

Conducting a study that attempts to quantify the number of masquerade attacks that may occur presents several difficulties that must be addressed. For example, soliciting participation from differing users could prove to be difficult. As previously mentioned, the participation attrition rate encountered in this study was approximately fifty percent. Any study seeking to gather the type of comprehensive data described here would most likely need to run for a longer period of time using more participants than the study presented in this dissertation. It is likely that, without mandatory participation, finding willing participants could pose a difficulty.

The other obstacle to be avoided in conducting this sort of future research regards the amount of data gathered. The log files generated by each participant were, on average, approximately 2-3 megabytes. This amount of data poses logistical problems. Storing the log files on each system could begin to burden local systems. Unfortunately transmitting event data over a local network also creates difficulties. Not only would transmitting a user's actions over the network cause a sharp increase in the amount of

data a network would have to cope with, but it would also be vulnerable to eavesdroppers on most networks. Furthermore, the current technique used to defeat eavesdropping attacks over shared network mediums, encryption, frequently results in even higher computational and network resource requirements.

Masquerade Detection Research

When considering future applications of this research, most people immediately consider the ability to detect masquerade attacks, often in the form of some type of intrusion detection system offering real-time alerting. Unfortunately, many difficult hurdles must be overcome prior to the creation of a real-time masquerade detection system.

The first and possibly most difficult obstacle to be conquered regards online processing. As previously described, the data processing performed in the study presented in this dissertation involved offline data mining techniques. Achieving the same accuracy in a real-time system could prove to be difficult. Furthermore, the issue of *where* the data processing might occur is also important to consider. Processing the data in an effort to detect attacks locally on each system could impose enough overhead to affect the user's experience. Processing all data remotely on a separate system could lead to scalability problems as the number of systems being monitors grows.

The usability of any real-time monitoring system also becomes critical. Most anomaly based intrusion detection systems suffer from a measurable false positive rate (recall that a false positive occurs when the system incorrectly believes an attack is occurring). It is reasonable to assume that any deployed masquerade detection system

would also suffer some number of false positives. How the system responds to those false positives will be critical. Several possibilities exist, offering varying levels of inconvenience. For example, in the event of a suspected attack, the system could require intervention from a system administrator. System administrators could end up devoting a great deal of time to tracking down and confirming/rejecting suspected attacks, particularly given the steps for investigating an attack. Another possible intervention would be to have the system lock the workstation that is supposedly under attack. This approach will most likely thwart a real attack, provided that the attacker does not have the user's password. The downside to this approach is that the burden for false positives is placed on the end user, possibly leading to complaints about the software. Of the two approaches to responding to false positives, it is believed that it makes more sense to employ the second approach, locking the workstation. It is believed that the burden experienced by an end user who is required to unlock their workstation is lower than the burden placed on system administrators tasked with responding individually to every false positive.

Another usability issue confronts the subject of user training. While the data presented here indicates that users develop habits in the way they accomplish tasks, it also stands to reason that users may adjust their behavior over time. Users will conceivably learn new, more efficient techniques for accomplishing tasks. Users may also simply change their behavior over time. Furthermore, a user's physical body may change as the result of injury or illness. Obviously the system should have some sort of manner to deal with these eventualities.

Yet another issue faced by a real-time detection system can be described as *session marking*. Consider the most illustrative example of a masquerade attack presented earlier in this dissertation: the unattended, unlocked workstation. In this case, the proper user has, at some point in the past, gotten up and left their workspace. Some time later, the attacker begins using the system. The two sessions (the one generated by the authorized user and the one generated by the attacker) can be delimited by the gap in time during which the system received no user input. The question is how much inactivity the system should interpret as the end of a session. This value will have to be determined by any group constructing an intrusion detection system using GUI Usage Analysis.

Human-Computer Interaction Applications

Much of the discussion in this dissertation has focused on the use of GUI Usage Analysis in a manner designed to enhance information security. There are, though, other disciplines that could potentially benefit from a successfully implemented GUI Usage Analysis product. One of these disciplines is the field of Human-Computer Interaction. More specifically, Usability specialists might find great utility in applying GUI Usage Analysis.

To illustrate, consider two computer users who share the same computer. User A is a 16 year old male who has used computers his entire life. User B is a 48 year old female who recently began using computers as a new requirement for her job. User A is quite comfortable manipulating graphical user interfaces and finds the normal tasks required to operate a GUI system to be second nature. User B is intimidated by the

keyboard, mouse, and the computer in general. User B lives in fear of "clicking the wrong thing."

Given their different backgrounds, it comes as no surprise that User A and User B process information differently. When given a screen to interact with, User A's mind immediately seizes upon the contextual clues placed by the interface designers. Because his mind is accustomed to processing these contextual clues, he is able to process the information displayed on the screen quite quickly. In short, he knows exactly where to look in order to find the information that he is after. User B, though, is accustomed to acquiring information from traditional printed materials. Her mind has not been trained to pick up on the contextual clues placed by the interface designers. As a result, she is not able to immediately ignore portions of the screen that do not contain the information she is after. Instead, she reads the entire screen like a book, progressing from left to right, starting at the top row and working downward.

Now, it is quite clear that while User A will accomplish the tasks much more quickly than User B, both users will eventually process the contents of the computer screen and find the information they were looking for. However, because User B has taken so much longer to complete the task, she finds the whole process of using a computer to be arduous and difficult. Her desire to use a computer in the future is greatly reduced. This represents the start of a self-perpetuating cycle in which User B doesn't use the computer because she is not good at it. Because she never uses it, she never improves.

It is conceivable that GUI Usage Analysis could be applied in this situation to alter how information is displayed to these two users. The system could conceivably custom tailor the interface for each user. The interface for User A could remain unaltered, displaying information in the traditional manner in which he is accustomed. The screen could potentially be re-arranged for User B, though, and structured to look more like a traditionally printed material. Critical information could be repositioned to the top left portion of the screen, where User B would be more likely to encounter it quickly.

There are other manners in which GUI Usage Analysis could potentially be used to enhance usability for users. Many users might enjoy an interface that automatically adjusts the display to their preferences after a few seconds of use. Users with physical handicaps might particularly benefit from this feature. In short, any computer system that is used by multiple people might potentially enjoy enhanced usability by utilizing GUI Usage Analysis as a means of tailoring the user interface "on the fly".

REFERENCES

Apple, Inc. *Performing Privileged Operations with Authorization Services.* 2007. http://developer.apple.com/documentation/Security/Conceptual/authorization_concepts/glossary/chapter_5_section_1.html (accessed September 24, 2007).

Auburn University. "Informed Consent for Adults." *OVPR - Human Subjects Research.* 2007. http://www.auburn.edu/research/vpr/ohs/forms/IC%20for%2019%20or%20older.doc (accessed September 16, 2007).

Bergadano, Francesco, Daniele Gunetti, and Claudia Picardi. "User Authentication through Keystroke Dynamics." *ACM Transactions on Information and Systems Security* 5, no. 4 (2002): 367-397.

bioChec. *bioChec - Keystroke Biometrics.* 2007. http://www.biochec.com/ (accessed December 16, 2007).

BioPassword, Inc. *Enterprise and Network Authenticaiton Software Products from BioPassword.* 2007. http://biopassword.com/network-authentication-software.php (accessed December 16, 2007).

Coull, Scott, Joel Branch, Boleslaw Szymanski, and Eric Breimer. "Intrusion Detection: A Bioinformatics Approach." *Annual Computer Security Applications Conference.* Las Vegas, Nevada, USA: Applied Computer Security Associates, 2003.

Coventry, Lynne. "Usable Biometrics." In *Security and Usability - Designing Secure Systems that People Can Use*, by Lorrie Faith Cranor and Simson Garfinkel, 175-198. Sebastaopol, California, USA: O'Reilly Media, 2005.

Denning, Dorothy. "An Intrusion-Detection Model." *IEEE Transactions on Software Engineering* (IEEE Press) 13, no. 2 (1986): 222-232.

Garg, Ashish, Ragini Rahalkar, Shambhu Upadhyaya, and Kevin Kwiat. "Profiling Users in GUI Based Systems for Masquerade Detection." *2006 IEEE Workshop on Information Assurance.* Piscataway, New Jersey, USA: IEEE Press, 2006. 48-54.

He, Bin, Kevin Chen-Chuan Chang, and Jiawei Han. "Discovering complex matchings across web query interfaces: a correlation mining approach." *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining.* Seattle, WA, USA: ACM Press, 2004. 148-157.

ID Control, Inc. "KeystrokeID." *ID Control - Strong, Affordable, and Easy Authentication.* 2007. http://www.idcontrol.com/index.php?option=com_content&task=category&sectionid=6&id=34&Itemid=55 (accessed December 16, 2007).

iMagic Software. "FAQ Trustable Passwords." *iMagic Software - Home of Trustable Passwords.* 2007. http://www.imagicsoftware.com/FAQ.htm (accessed December 16, 2007).

Imsand, Eric, and John A Hamilton. "GUI Usage Analysis for Masquerade Detection." *IEEE Workshop on Information Assurance.* Piscataway, New Jersey, USA: IEEE Press, 2007. 270-276.

Imsand, Eric, and John A. Hamilton. "Impact of Daily Computer Usage on GUI Usage Analysis." *InfoSecCD Conference.* New York, New York, USA: ACM Press, 2007. 196-205.

Joachims, Thorsten. *SVMLight Support Vector Machine.* February 9, 2004. http://svmlight.joachims.org/ (accessed December 12, 2007).

Joyce, Rick, and Gopal Gupta. "Identity Authentication Based on Keystroke Latencies." *Communications of the ACM* 33, no. 2 (1990): 168-176.

Khanna, Rahul, and Huaping Liu. "System Approach to Intrusion Detection Using Hidden Markov Model." *International Wireless Communications and Mobile Computing.* Vancouver, British Columbia, Canada: ACM Press, 2006. 349-354.

Kingsbury, Kathleen. "Telltale Fingertips." *Time Magazine.* December 10, 2006. http://www.time.com/time/insidebiz/article/0,9171,1568467,00.html (accessed December 16, 2007).

Kumar, Deept, Naren Ramakrishnan, Richard Helm, and Malcolm Potts. "Algorithms for Storytelling." *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* Philadelphia, PA, USA: ACM Press, 2006. 604-610.

Li, Ling, and Constantine Manikopoulos. "Windows NT One-class Masquerade Detection." *IEEE Workshop on Information Assurance.* Piscataway, New Jersey, USA: IEEE Press, 2004. 82-87.

Market Share. *Operating System Market Share for November 2007.* December 6, 2007. http://marketshare.hitslink.com/report.aspx?qprid=10 (accessed December 6, 2007).

Mathworks Corporation. "Neural Network Toolbox." *The Mathworks: Accelerating the Pace of Engineering and Science.* 2007. http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/ (accessed December 18, 2007).

Maxion, Roy, and Tahlia Townsend. "Masquerade Detection Using Truncated Command Lines." *IEEE International Conference on Dependable Systems and Networks.* Pittsburgh, Pennsylvania, USA: IEEE Press, 2002. 219-228.

Microsoft Corporation. "About Hooks." *Microsoft Developer Network.* 2007a. http://msdn2.microsoft.com/en-us/library/ms644959.aspx (accessed 8 31, 2007).

—. "GetClassName Function." *Microsoft Developer Network.* 2007. http://msdn2.microsoft.com/en-us/library/ms633582.aspx (accessed September 1, 2007).

—. "GetCommandLine (Windows)." *Microsoft Developer Network.* 2007. http://msdn2.microsoft.com/en-us/library/ms683156.aspx (accessed September 1, 2007).

—. "Handles and Objects (Windows)." *Microsoft Developer Network.* November 1, 2007. http://msdn2.microsoft.com/en-us/library/ms724457.aspx (accessed December 6, 2007).

—. "Hooks." *Microsoft Development Network.* 2007. http://msdn2.microsoft.com/en-us/library/ms632589.aspx (accessed 08 31, 2007).

Newbold, Richard D. *Newbold's Biometric Dictionary: For Military and Industry.* Bloomington, Indiana: AuthorHouse, 2007.

Newcomer, J. "Hooks and DLLs." *FlounderCraft, Ltd.* March 20, 2003. http://www.flounder.com/hooks.htm (accessed September 1, 2007).

Peacock, Alen, Xian Ke, and Matt Wilkerson. "Identifying Users from Their Typing Patterns." In *Security and Usability - Designing Secure Systems that People Can Use*, by Lorrie Faith Cranor and Simson Garfinkel, 199-220. Sebastopol, California, USA: O'Reilly Media, 2005.

Pfleeger, Charles, and Shari Pfleeger. *Security in Computing.* Prentice Hall PTR, 2003.

Pusara, M, and C Brodley. "User Re-authentication via Mouse Movements." *Computer and Communications Security.* Washington, DC, USA: ACM Press, 2004. 1-8.

Rulequest Research. "Data Mining Tools See5 and C5.0." *Rulequest Research - Data Mining Tools.* November 2007. http://www.rulequest.com/see5-info.html (accessed December 16, 2007).

—. *Data Mining Tools See5 and C5.0.* http://www.rulequest.com/see5-info.html (accessed December 7, 2007).

Russell, Stuart, and Peter Norvig. *Artificial Intelligence - A Modern Approach.* Upper Saddle River, NJ, United States: Prentice Hall, 2003.

Schonlau, Matthias, William DuMouchel, Wen-Hua Ju, Alan Karr, Martin Theus, and Yehuda Vardi. "Computer Intrusion: Detecting Masquerades." *Statistical Science* 16, no. 1 (2001): 1-17.

Umphress, D, and G Williams. "Identity Verification through Keyboard Characteristics." *International Journal of Man-Machine Studies* (Academic Press) 23 (1985): 263-273.

United States Secret Service: National Threat Assessment Center. *Insider Threat Study: Computer System Sabotage in Critical Infrastructure Sectors.* Washington D.C.: United States Secret Service, 2004.

United States Secret Service: National Threat Assessment Center. *Insider Threat Study: Illicit Cyber Activity in the Banking and Finance Sector.* Washington D.C.: United States Secret Service, 2004.

Vapnik, Vladamir. *The Nature of Statistical Learning Theory.* Springer-Verlag, 1995.

APPENDIX A

INFORMED CONSENT LETTER

Figure A1 contains the contents of the informed consent document distributed to all participants prior to participation in this study.

INFORMED CONSENT
for a Research Study Entitled
Employing WIMP Usage Patterns for Masquerade Detection

You are invited to participate in a research study investigating new ways to identify individual computer users. This study is being conducted by Eric S. Imsand, a student in the Department of Computer Science and Software Engineering, under the supervision of Dr. John A. Hamilton, Jr. Associate Professor in the Department of Computer Science and Software Engineering. We hope to learn if the manner in which people use a computer – the icons they click on, keyboard shortcuts they use, etc. – can be used as a means of identification. You were selected as a possible participant because you are an individual over 19 years of age and have been identified as a person who is at least mildly proficient at the normal operation of a computer running the Windows XP operating system.

If you decide to participate, we will first install a piece of software on your personal computer that is designed to record which icons, menus you click on, any keyboard shortcuts you may use, and so on. After the monitoring software has been installed you will be asked to complete a series of tasks once a day for five consecutive days while the monitoring software is running. Finally you will be asked to allow the monitoring software to record your actions over a full two day period of time.

If you decide to participate in this study, there is a chance that monitoring software could record data that you might otherwise prefer not be logged. This risk is greatest when the software is recording all actions performed by you while using your computer in a typical manner. The result of this risk is that your privacy could be breached if the experimental data were stolen.

To minimize these risks several precautions have been taken. First, the information recorded by the monitoring software is encoded so that it can only be interpreted using a custom piece of computer software. Secondly, the information will be stored

on your computer in a secure manner so that anyone seeking to access it would have to provide your username and password (if applicable).

Third, you may always temporarily pause the recording software (i.e. pause recording) while you are doing work that you prefer not be recorded. Finally, if you should decide to cease participation in this study, the monitoring software can be permanently removed from your computer at any time by using the "Add/Remove Programs" menu in the Windows control panel. Technical assistance will be provided if you cannot remove the software yourself. Please note that all of the data that is collected will be destroyed no later than 6 months following the conclusion of our study. Files stored electronically will be erased and any physical media will be destroyed.

If our research is successful, we hope to be able to apply these findings to the creation of a new breed of computer security software. This new generation of software will be able to tell whether the person using your computer is actually you. It would provide an extra layer of defense if your personal computer's password were ever stolen, or if someone were to begin using your computer while you were away from your desk. We cannot promise you that you will receive any or all of the benefits described.

Any information obtained in connection with this study and that can be identified with you will remain confidential. Only the authors of this study will handle the data gathered and all data will be stored on a non-networked secured computer workstation. Information collected through your participation may be used in the completion of educational requirements (i.e. doctoral dissertations), published in a professional journal, and/or presented at a professional meeting, etc. If so, none of your identifiable information will be included.

Any personally identifiable data gathered during the course of this study will be destroyed no later than 12/31/2007. Please know that you may withdraw from participation at any time, without penalty, and that you may withdraw any data which has been collected about yourself, as long as that data is identifiable.

Your decision whether or not to participate will not jeopardize your future relations with Auburn University or the Department of Computer Science and Software Engineering.

If you have any questions we invite you to ask them now. If you have questions later, please contact Eric Imsand (Phone: 901-338-9323, e-mail: imsanes@auburn.edu) or Dr. Hamilton (Phone: 334-844-6360, e-mail: hamilton@eng.auburn.edu) and they will be happy to answer them. You will be provided a copy of this form to keep.

For more information regarding your rights as a research participant you may contact the Auburn University Office of Human Subjects Research or the Institutional

Review Board by phone (334)-844-5966 or e-mail at hsubjec@auburn.edu or IRBChair@auburn.edu.

HAVING READ THE INFORMATION PROVIDED, YOU MUST DECIDE WHETHER OR NOT YOU WISH TO PARTICIPATE IN THIS RESEARCH STUDY. YOUR SIGNATURE INDICATES YOUR WILLINGNESS TO PARTICIPATE.

_____	_____
Participant's signature Date	Investigator obtaining consent Date
_____	_____
Print Name	Print Name
_____	_____
Parent's or Guardian Signature Date (if appropriate)	Co-investigator's signature Date (if appropriate)
_____	
_____	

Figure A1. Text of the Informed Consent Document Provided to Study Participants

APPENDIX B

SIMULATED ATTACK SUCCESS RATE FOR INDIVIDUAL USERS

Table A1 contains the results of simulated attacks for all possible combinations of known users and attacker.

Table A1. Simulated Attack Rate for Known Users and Attackers

	User ID										
		1	2	3	4	5	6	7	8	9	10
Attacker ID	1		0%	0%	0%	0%	0%	0%	0%	0%	0%
	2	0%		0%	0%	0%	0%	0%	0%	0%	0%
	3	0%	0%		11%	0%	0%	11%	0%	56%	0%
	4	0%	0%	0%		0%	0%	33%	0%	78%	0%
	5	0%	0%	0%	89%		0%	11%	0%	56%	0%
	6	0%	0%	0%	56%	0%		0%	0%	22%	0%
	7	0%	0%	0%	0%	0%	0%		0%	44%	0%
	8	0%	0%	0%	33%	0%	0%	0%		22%	0%
	9	0%	0%	0%	0%	0%	0%	44%	0%		0%
	10	0%	0%	0%	0%	0%	0%	0%	0%	0%	
	11	0%	0%	0%	67%	0%	0%	56%	0%	67%	0%
	12	0%	0%	0%	0%	0%	0%	0%	0%	56%	0%
	13	0%	0%	0%	22%	0%	0%	0%	0%	67%	0%
	14	0%	0%	0%	56%	0%	0%	0%	0%	11%	0%
	15	0%	0%	33%	11%	0%	0%	0%	0%	33%	0%
	16	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	17	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	18	0%	0%	0%	0%	0%	0%	0%	0%	11%	0%
	19	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	20	0%	0%	0%	11%	0%	0%	0%	0%	11%	0%
	21	0%	0%	0%	11%	0%	0%	0%	0%	11%	0%
	22	0%	0%	0%	0%	0%	0%	0%	0%	22%	0%
	23	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	24	0%	0%	0%	0%	0%	0%	0%	0%	78%	0%
	25	0%	0%	0%	44%	0%	0%	11%	0%	44%	0%
	26	0%	0%	0%	0%	0%	0%	0%	0%	22%	0%
	27	0%	0%	0%	0%	0%	0%	11%	0%	44%	0%
	28	0%	0%	0%	67%	0%	0%	33%	0%	78%	0%
	29	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	30	0%	0%	0%	11%	0%	0%	0%	0%	22%	0%
	31	0%	0%	0%	67%	0%	0%	33%	0%	100%	0%

Table A1 - Continued

	User ID										
		11	12	13	14	15	16	17	18	19	20
Attacker ID	1	0%	0%	0%	0%	0%	67%	0%	0%	0%	0%
	2	0%	0%	0%	0%	0%	44%	0%	0%	0%	0%
	3	11%	0%	0%	0%	0%	33%	0%	0%	0%	0%
	4	67%	0%	0%	0%	0%	11%	0%	44%	0%	0%
	5	44%	0%	0%	0%	0%	56%	0%	11%	0%	11%
	6	11%	0%	0%	0%	0%	89%	0%	0%	0%	67%
	7	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	8	0%	0%	0%	0%	0%	100%	0%	0%	0%	22%
	9	0%	0%	11%	0%	0%	33%	0%	0%	0%	11%
	10	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%
	11		0%	0%	0%	0%	0%	0%	11%	0%	0%
	12	0%		0%	0%	0%	67%	0%	0%	0%	0%
	13	0%	0%		0%	0%	100%	0%	0%	0%	0%
	14	0%	0%	0%		0%	0%	0%	22%	0%	0%
	15	0%	0%	11%	0%		78%	0%	0%	0%	0%
	16	0%	0%	0%	0%	0%		0%	0%	0%	0%
	17	0%	0%	0%	0%	0%	0%		0%	0%	0%
	18	0%	0%	0%	0%	0%	11%	0%		0%	0%
	19	0%	0%	0%	0%	0%	0%	0%	0%		0%
	20	0%	0%	0%	0%	0%	33%	0%	0%	0%	
	21	0%	0%	0%	0%	0%	0%	0%	22%	0%	11%
	22	0%	0%	0%	0%	0%	100%	0%	0%	0%	11%
	23	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	24	0%	0%	0%	0%	0%	44%	0%	0%	0%	11%
	25	22%	0%	0%	0%	0%	67%	0%	0%	0%	44%
	26	0%	0%	0%	0%	0%	0%	0%	33%	0%	0%
	27	22%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	28	44%	0%	0%	0%	0%	0%	0%	67%	0%	0%
	29	0%	0%	0%	0%	0%	56%	0%	0%	0%	0%
	30	0%	0%	0%	0%	33%	89%	0%	0%	0%	0%
	31	56%	0%	33%	0%	0%	33%	0%	0%	0%	22%

Table A1 - Continued

	User ID											
		21	22	23	24	25	26	27	28	29	30	31
Attacker ID	1	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	2	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	3	0%	0%	0%	56%	0%	22%	0%	0%	0%	0%	0%
	4	22%	0%	0%	0%	0%	56%	0%	44%	0%	22%	0%
	5	44%	0%	0%	22%	11%	67%	0%	0%	0%	22%	0%
	6	0%	0%	0%	0%	0%	67%	0%	0%	0%	67%	0%
	7	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	8	0%	0%	0%	0%	0%	56%	0%	0%	0%	22%	0%
	9	0%	0%	0%	0%	0%	33%	0%	0%	0%	11%	0%
	10	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	11	0%	0%	0%	0%	0%	11%	0%	11%	0%	0%	0%
	12	0%	0%	0%	67%	0%	44%	0%	0%	0%	0%	0%
	13	0%	0%	0%	78%	0%	56%	0%	0%	0%	22%	0%
	14	0%	0%	0%	0%	0%	33%	0%	0%	0%	0%	0%
	15	44%	0%	0%	67%	0%	44%	0%	0%	0%	33%	0%
	16	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	17	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	18	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	19	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	20	0%	0%	0%	11%	0%	33%	0%	0%	0%	0%	0%
	21		0%	0%	22%	0%	0%	11%	0%	0%	0%	0%
	22	0%		0%	22%	0%	0%	0%	0%	0%	11%	0%
	23	0%	0%		0%	0%	0%	0%	0%	0%	0%	0%
	24	11%	0%	0%		0%	56%	56%	0%	0%	0%	0%
	25	11%	0%	0%	0%		33%	0%	0%	0%	56%	0%
	26	0%	0%	0%	33%	0%		0%	0%	0%	0%	0%
	27	11%	0%	0%	67%	0%	11%		0%	0%	0%	0%
	28	0%	0%	0%	0%	0%	22%	0%		0%	0%	0%
	29	0%	0%	0%	0%	0%	0%	0%	0%		0%	0%
	30	0%	0%	0%	0%	0%	11%	0%	0%	0%		0%
	31	56%	0%	0%	67%	0%	56%	0%	0%	0%	11%	

APPENDIX C.

INDIVIDUAL PERFORMANCE ON IDENTIFICATION EXPERIMENTS

The following represent the raw data obtained from the identification experiment based on the Jaccard Index, as described in Chapter 5. The first column represents the actual donor of the test data sample. The second column represents the user the identification algorithm suspects provided the unknown sample, based on analysis of the known training sets.

Table A2. Results of User Identification Experiment showing which Users were mistakenly Identified

ID Number of Actual User	ID Number of Suspected User
1	1
1	1
1	1
2	2
2	2
2	2
3	3
3	3
3	3
4	28
4	5
4	4
5	5
5	5
5	5
6	6
6	6
6	6
7	7
7	11
7	9

Table A2 - Continued

ID Number of Actual User	ID Number of Suspected User
8	8
8	8
8	8
9	9
9	7
9	13
10	10
10	10
10	10
11	28
11	11
11	9
12	24
12	12
12	12
13	13
13	13
13	13
14	14
14	14
14	14
15	15
15	15
15	15
16	16
16	16
16	10
17	17
17	17
17	17
18	18
18	28
18	18
19	19
19	19
19	19
20	25
20	20
20	20
21	21
21	21
21	21
22	22
22	22
22	22

Table A2 - Continued

ID Number of Actual User	ID Number of Suspected User
23	23
23	23
23	23
24	9
24	27
24	24
25	8
25	25
25	25
26	14
26	12
26	24
27	27
27	27
27	24
28	28
28	28
28	28
29	8
29	29
29	29
30	30
30	30
30	30
31	31
31	31
31	31