

UNDERGRADUATE THESIS PROJECT PROPOSAL
School of Engineering and Applied Science
University of Virginia

Web Browser Security Interface Evaluation
for a More Meaningful Design

Submitted by

Jennifer Kahng

Computer Science

TCC 401

Section 4 (11 am)

October 30, 2000

On my honor as a University student, on this assignment I have neither given nor received unauthorized aid as defined by the Honor Guidelines for Papers in TCC Courses.

Signed _____

Approved _____ Date _____
Technical Advisor - David Evans

Approved _____ Date _____
TCC Advisor - Rosanne Welker

TABLE OF CONTENTS

Abstract	1
Glossary	2
I. Rationale and Objectives	3
Rationale	3
Objectives	5
II. Preliminary Impact Statement	6
User Responses	6
Annoying Features	7
Educated Users.....	7
Overall Assessment and Recommendations	8
III. Review of Relevant Literature	8
Security Policies and Users.....	8
User Interface Development Methods	9
Making Security Usable.....	11
IV. Statement of Project Activities	11
A. Activities	12
1. Background Information.....	12
2. Test Application Function Formulation.....	12
3. Create Test Application	13
4. Gather Test Group.....	13
5. Test Group	13
6. Analyze Data.....	13
7. Formulate Design Methods.....	14
8. Final Technical Document.....	14
B. Schedule	15
C. Personnel	16
D. Resources	16
V. Expected Outcomes	17
VI. Works Cited	18
VII. Bibliography	19
Appendix A	20
Budget and Equipment.....	20
Appendix B	21

Abstract

Web browsers allow users to access aspects of the Internet via the World Wide Web. Nearly all types of transactions can occur online today: banking, shopping, education, communication, etc. Web browsers facilitate the completion of these transactions, but they also provide a means of keeping one's personal information private. When users attempt to send information through the World Wide Web for some operation, the web browser displays a security warning informing the user that they may be doing something unsafe. These displays, however, are often unhelpful, vague or inappropriate. Thus, users become frustrated and quickly learn to ignore or remove these warnings. By evaluating various interface designs, it should be possible to determine what design methods are effective for security applications in web browsers.

This project serves to develop methods of security interface design that will capture the user's attention without interfering too heavily in their work. To accomplish this, I plan on constructing an application that generates security messages of a different design and determines its usefulness. Users will be confronted with displays of a different color, size or shape, depending upon the severity of the security problem that arises due to the action they try to accomplish. The effectiveness of these displays will also be tracked to try to determine how long it takes before they are ignored or turned off. The results of this evaluation should serve to help create a set of design rules for web browser security interfaces. It may also be possible to apply this study to other security applications since they also suffer from the same interface issues as web browsers.

Glossary

What follows is a list of definitions for terms that may be useful when reading this proposal.

ActiveX Control - A program used on the World Wide Web developed by the Microsoft Corporation from existing program components. These controls are programs that bring new features to a specific web page and are thus download automatically when a web page using one is loaded [5:263].

CGI (Common Gateway Interface) - CGI scripts are server side programs which have the capability of returning information that is generated based on input (dynamically) rather than already generated information (statically) [5:273].

client - Refers to the local computer or computer system the user is using at that moment.

cookie - A method of providing web browser process history to a server. Cookies are useful for web sites like shopping areas that need to keep track of what a user has ordered between web pages on the web site. Cookies were developed by Netscape Communications [2:193].

Java (and Java Applets) - Java is a programming language developed by Sun Microsystems. Java applets are Java programs that often run on web servers and downloaded when a web page specifies it needs it. Java applets bring new functionality to web pages [2:198]

JavaScript - JavaScript is usually written into the web page being viewed and also adds functionality to the page. It may open new web browser windows or download Java applets for the page's use [2:198].

server - Refers to the remote computer or computer system trying to communicate with the user.

SSL (Secure Sockets Layer) - An Internet protocol developed by Netscape Communications. It allows clients and servers to communicate securely over the Internet. SSL also allows both parties to know, for certain, that they are communicating with each other and not a third party impersonating one of the parties [5:133].

user - Refers to the person using a computer or software system.

web browser - Refers to the software that facilitates exploration of the World Wide Web.

I. Rationale and Objectives

In today's world, web browsers are an integral part of our everyday lives. They assist us in our daily errands as well as trying to protect us from malicious users. The methods of protection, however, are not completely adequate and are not used to their full potential by users. This lack of use is perpetuated by the fact that the intended helpful warnings and displays are poorly received and often ignored. If a proper interface can be developed, users will be able to more easily make use of existing security measures. My project will create an application to evaluate different security display types to try to determine the most effective method of offering users this important information.

Rationale

Since its birth, the World Wide Web has been one of the most accessible parts of the Internet. Today, more and more services are available online, making it easier for people to do daily errands from the comfort of their own homes. This convenience also poses a problem for the general public, however, since much of their personal information could be available to malicious parties. Web browsers have a built-in defense against such tactics, providing warnings when users try and send information unencrypted somewhere and configurable security settings with options for different types of online operations. These measures, however, are not enough.

Security settings assume that the user has some knowledge of what they mean, what they do and how they work. People who browse the web often do not understand the underlying concepts of web security and do not care to learn. The default settings usually are not changed, which may not provide an adequate level of protection for online transactions. The obscure nature of the security settings is not the only problem, however. Warning screens display when

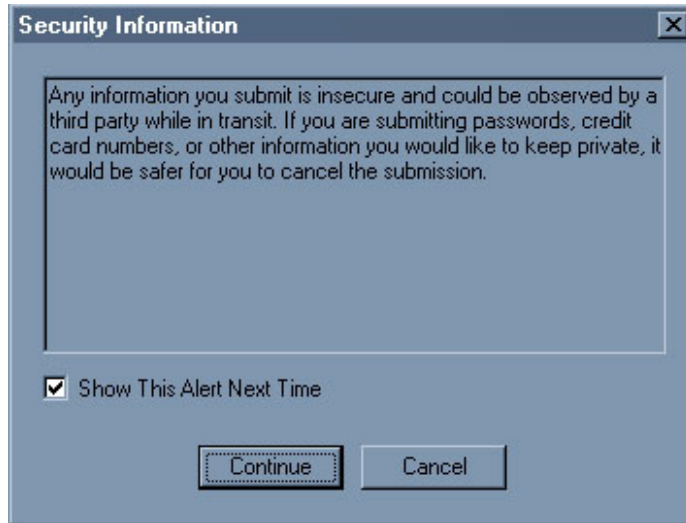


Figure 1. Screen capture of a Netscape® Navigator security warning. 2000.

users do something deemed "unsafe" by the web browser, but these screens are often ambiguous or inappropriate. Figure 1 shows a typical Netscape® Navigator security warning. Entering parameters into a library search field generated this particular warning. I did not feel it was necessary to encrypt those parameters, yet a security information screen displayed. Since security messages in web browsers are general, applying to several cases of possible security breach, the screens display at times where it is not necessary, interrupting the user. More often than not, users become frustrated at the messages because they do not apply to their current operation and disable the alerts completely. This is an undesirable consequence because the alerts are sometimes correct, but if they are disabled, the user is never informed.

The evaluation implemented in this project will provide information to aid in designing a better security interface to make the user take notice of the correct security alerts and takes appropriate action. I believe it is necessary to delve deeper into the mechanisms for determining when and which security alerts display. It is not sufficient to display the same type of warning for different levels of security breaches. All of the security warning windows in web browsers

look alike in shape and color. Thus, it is possible for users to believe they are seeing the same security alert and automatically select the affirmative option without looking, even if the messages may be conveying different information. Perhaps it would be more effective if more severe security breach operations were colored differently. That is, if some web page has an object that exploits a browser's weakness and attempts to tamper with the user's local files, a security warning in a red box would appear, rather than the standard browser color.

Aesthetic modifications may not work, however. The user would not be used to seeing a different kind of dialog box displayed and would take notice the first time, but the goal is to make sure the user pays attention all the time. User interface design methods must be taken into account, but these methods are not completely adequate for security applications. When designing user interfaces in general, it is not necessary for the developer to understand the inner workings of the product to make it useful. With security applications, however, the developer should know how the product works, in terms of security, in order to develop an interface that will take advantage of those features. Additionally, if the developer does not understand how the application should work, then the interface may not interact with the security part of the software correctly resulting in the user's data not actually being secure. Research into interface design for security applications is fairly new, and there is not much literature available on this subject [10]. This project may help broaden the field, or at least add positively to the community.

Objectives

There are several goals in this project, most of which deal with evaluating users' reactions to different types of interfaces. The other aspect of this project involves interpreting the results from user testing. These objectives are summarized as follows:

- Create an experiment that has a user perform some task while running a Java test application in the background that generates false security messages. The messages should be false so as not to actually inflict harm onto the user's computer and also to implement the changes necessary to observe the user's reactions.
- Gather a group of test users of different computing ability.
- Develop a list of evaluation criteria. Items should include situations such as how long it takes before a particular type of security message is ignored, whether or not the test user investigates further when a severe warning occurs, etc.
- Develop a list of recommended design issues to focus on for more effective web browser security interfaces.

II. Preliminary Impact Statement

Users of web browsers are the primary audience for this project and will encounter any changes that are made due to the design methods developed. These users only represent a specific group of people that this project will affect and it may be possible to use the results for other security applications as well. Changing the interface that web browser users see when confronted with security issues may also cause more problems than intended. If the interface is constructed well, however, users may learn more about what they are doing in their online transactions and take more interest in their personal protection.

User Responses

In a broader view, the development of a meaningful interface for security settings can be scaled for other uses, especially when a user's input is necessary. Software that requires

notifying users of changes or potential hazards also needs to employ the same tactics evaluated in this project to get the user's attention and evoke the proper response. In other security applications, not all situations require high priority, just as in web browser security warnings. When a sufficiently severe situation arises, however, the user not only needs to see the alert, but also needs to understand why it appears and what to do when given the alert.

Annoying Features

If this evaluation results in a redesign of web browser alerts, the changes to the interface may not produce intended effects. Different companies may also implement a new design in different ways. In many cases, users of popular software are used to the way the program behaves and are easily annoyed at new features or existing features brought on by a new design. If the design of the security settings interface or its behavior is substantially different from what exists in current web browsers, users may be annoyed with those changes and pay even less attention to the warnings or signs that something is amiss.

Educated Users

If the application I create can help determine what security interface designs are effective, and new designs are implemented, users will be able to better protect themselves online. Warnings that catch user's attention properly will stop them from making mistakes during online transactions and may compel them to be more cautious. Also, since it is the user's personal information that is at stake, if they can see that something potentially bad could happen to them, they may take a more active interest in research and development for online security.

Overall Assessment and Recommendations

This project has the potential to affect a wide range of people. Since this project will only evaluate certain aspects of security interface design, I recommend that a further and broader study occur to complement the information gained here. It may turn out that the results I compile may only effectively apply to web browsers, but I hope that it will at least induce others contribute more for the research of security application interfaces.

III. Review of Relevant Literature

The design of practical user interfaces for security applications is difficult for computer software developers. In many cases, the actual functionality of the security enhancements is lost or the interface itself is inadequate to convey the security protection offered. The problem increases when users of varying computer abilities use the same product. Security programs require some knowledge of the protocols used to effectively protect users and are often difficult for "beginning" or "intermediate" computer users to utilize. This section of the proposal will describe security policies existing on the Internet and how they affect users. Current methods of user interface development will also be discussed, as well as why those methods are not suitable for security enhancing programs. I will also present information regarding current ideas on making security more usable for the general public without changing the user's normal behavior.

Security Policies and Users

The number of users connected to the Internet grows daily. The World Wide Web, the most visible aspect of the Internet, provides millions of opportunities for malicious users to try and gain access to information they should not view. The media often glorifies the most spectacular cases, such as a government website that was defaced or an E-Mail virus that

crippled large companies, but places less emphasis on the common people who are more at risk for a security breach than anyone else. Existing web browsers attempt to court users by having the latest features before their competitor, but this race leaves less time for testing and verification resulting in unstable applications or security holes that attackers quickly exploit. These features, such as cookies, Java applets, JavaScript, CGI scripts, or ActiveX controls, were created to help make Web activities more versatile, but initially produced undesired effects. One such exploit occurred in 1996 when researchers discovered that it was possible to program a Java applet to delete files on a user's hard drive without the user's knowledge [8:61].

More technical methods of web security have been offered to help combat against security and privacy breaches. SSL works within the Internet's main communications protocol to provide a secure method for clients and servers to communicate. This is especially useful for online stores that need to obtain user information like credit card numbers securely. The idea of having anonymous users also appears. This protocol allows users to access a third party web site which retrieves the information the user wishes to view and sends it back to the user without the server knowing who originally requested the information. Anonymous services help protect user's privacy [5:332].

User Interface Development Methods

An interface is dependent on its application and intended functionality, thus no standard for interface development exists. This is evident in user interface trends today since users are more familiar with learning how to use a piece of software rather than developers spending a large amount of time creating an acceptable interface [4:43]. There have been studies, however, to try and determine how user interfaces should be designed. Critical systems where human life is at stake, for instance, should have simple interfaces for an operator to use properly while under

stress, yet not too simple as to facilitate catastrophic errors. The software should also cater towards users of different technical backgrounds, skill levels, and needs [7:20]. A methodology to try and create adequate interfaces exists by utilizing ideas like user observation and constant feedback analysis [1:145].

Despite the methods of creating practical user interfaces, they are often inadequate for security applications. Jef Raskin, the leader of the Macintosh project, proposes an idea that says pop-up messages should be kept at a minimum [6:178]. When applied to security messages in a web browser, however, this does not apply since the messages should only occur when a security breach occurs, thus there is no real method of minimizing the number of windows that display when they are all potentially important. The security settings are also often buried under menus or located in an area where users are unlikely to look. While the latter is a result of trying to separate security settings from regular settings, it goes against normal human practices. For most software applications, it is unnecessary to know the details of how the software works in order to use it. When a user uses a word processor, he or she does not need to know how the word processor stores its information in order to create a document, modify an existing document, etc. Security applications are different in this regard. A study on the usability of PGP (Pretty Good Privacy, a cryptographic system that allows users to send E-Mail securely) showed that users who did not understand how PGP worked, even on a basic level, combined with an interface, that was not as helpful as originally thought, were more prone to mistakes than users who took the time to learn the details of the program [10]. Thus, it was clear that even programs with seemingly adequate interfaces failed when it came to informing the user exactly what he or she should do to use it correctly.

Making Security Usable

The area of security application user interfaces is relatively new. Groups like the Special Interest Group for Human-Computer Interactions have specialists who help create user interfaces for various applications, but often lack the expertise to design interfaces for security applications. There are many reasons why user interfaces for security applications are difficult to develop. First and foremost, the developer must understand the underlying security principles behind the application. The developer also cannot afford to misunderstand how the application should be used or the user cannot fully utilize the protection the software offers [9:3]. There are methods used in practice to try and determine what users will respond to as well as what they want out of their software. These methods are much like those discussed earlier, user observation and feedback analysis, but also required users to understand how the security software worked and how it would react in certain situations [3:4]. It is clear that more study into user interfaces for security applications, as well as a method of educating users, is needed to successfully create a practical user interface.

IV. Statement of Project Activities

In order to complete my thesis, a detailed set of activities and an associated schedule are detailed. These items will serve to measure my progress. It may not be possible to strictly follow the schedule since circumstances always arise when some sections may get delayed.

A. Activities

The activities described here outline the steps I plan to take to complete my project successfully. The time ranges specified are only estimates and each section may take more or less time to complete depending on my proficiency in each and the actual difficulty of the task.

1. Background Information

Before beginning the evaluation project, I must familiarize myself with a number of items. The test application will most likely be written in Java, since Java can easily interact with other software, like a web browser. I do not know Java; however my knowledge of C++ should facilitate its learning. I anticipate that it should not take more than two or four weeks to understand the basics of Java and begin to design the application. I must also understand what policies web browsers use to determine security issues and how they are handled. Additionally, I need to research the kinds of security breaches that may be possible through web scripts or controls so I can properly simulate them in my test application. This activity should take two to four weeks as well.

2. Test Application Function Formulation

The Java test application needs to have a specific set of functions to perform. What types of security errors should it simulate? How do I determine severity levels for the different kinds of security breaches? What kinds of design differences should I incorporate: only one kind, like color changes, or multiple variations? These questions must be answered before the application is created and testing can begin. Determining application functionality should take approximately two weeks.

3. Create Test Application

The test application must be built and tested to make sure it functions as expected and has no unintended consequences. This application should run in the background so the user has no knowledge that a supplementary software program is running while they are testing the system. The application should also have some mechanism uniquely identify each user and record their actions regarding security screens. Application creation may take three to four weeks, perhaps longer if major bugs arise.

4. Gather Test Group

An appropriate test group must be collected in order to use my application. Ideally, the test group should not know the true task they are performing, lest they be inclined to think more about what they are doing. The group should probably consist mostly of "novice" users, but "expert" users must also participate since the Internet is comprised of all kinds of users. It should take a week or two to gather a suitable test audience.

5. Test Group

The test group must be given a task to perform which will utilize my application. It may be best to separate test users so that they do not influence each other's decisions when confronted with a security operation. Testing should be spread across two or three weeks.

6. Analyze Data

The data from testing must be analyzed to determine user habits and whether or not the different types of displays were any more or less effective than current designs. Analysis should take one to two weeks.

7. Formulate Design Methods

Based on the results from the data analysis, I will form a set of design rules regarding what will and will not work. Some further analysis should also be done to determine if the scope of the evaluation could be scaled for other security applications. This process should take one to two weeks.

8. Final Technical Document

Throughout the process of my project, I will gather data for my final technical paper.

The majority of the work will probably take the latter two months of the entire project to complete as various data arrives.

Since my project hinges directly on whether or not the test application can be built, the amount of time allotted to testing it may be too short to glean any real meaningful results. Programming takes variable amounts of time and is very difficult to specify precisely, especially when the language is unfamiliar, which is the case for me. I will hopefully be able to test my application on at least a small group of users, if I do not have time to test on a wider range of users. In the event that my application takes overly long to create and cuts into testing time, it may be necessary to hand the formal testing aspect of this project to someone else.

B. Schedule

The following chart graphically represents my activities and the length of time I anticipate each activity will run.

Task	October	November	December	January	February	March
1						
2						
3						
4						
5						
6						
7						
8						

Task Number	Description
1	Background Information
2	Test Application Function Formulation
3	Create Test Application
4	Gather Test Group
5	Test Group
6	Analyze Data
7	Formulate Design Methods
8	Final Technical Document

C. Personnel

During the course of my project, I will need the assistance of several people. Most notable, my technical advisor, David Evans of the Computer Science Department, will help me refine my project and help me contact the right people so that I can test my application properly. Additionally, Rosanne Welker of the TCC Department will help me understand how my thesis project should proceed and possible aide in the generation of my technical report.

I have also contacted some people outside of the University who may help my project. Since Mozilla is a web browser whose source is open for anyone to see, I asked some of the security developers for assistance. Mitchell Stoltz from Netscape has offered to assist me in understanding security policies and how web browsers use them, if possible. Additionally, those who correspond regularly on the newsgroup netscape.public.mozilla.security should be able to help as well.

Depending on if this projects becomes more complicated than I anticipate, I may have to ask for the assistance of others in completing my tasks.

D. Resources

For this project, I will need the use of a computer and a means in which to compile Java applications. I have my own computer and I can easily obtain a Java compiler. I will also need to gather Java resources since I must learn Java on my own. Sun Microsystems will have much information on Java and Java compilers that I can look into, since they developed Java.

V. Expected Outcomes

After I finish writing an application that tracks how users react to security messages, I will learn what methods are effective in interface design. It will be difficult, as many users are already used to dismissing current messages without a second glance. If my application can successfully divide security issues into subcategories, display them differently based on severity and track the user's responses, then the application itself is complete. The analysis of the data determines the completeness of the project. Ideally, the information should show what users react to and for how long. That is, since users are predisposed to ignoring web browser security messages, how long will it take for them to ignore different looking messages? If the results I find reveal that showing different kinds of messages based on the security problem catches the user's attention and makes them think about what they are doing, I will have found some design methods for this particular aspect of security interfaces. If the results show the opposite, then I will know what aspects of interfaces are ineffective and should be avoided in future implementations.

VI. Works Cited

1. Borenstein, Nathaniel S. *Programming as if People Mattered: Friendly Programs, Software Engineering, and Other Noble Delusions*. Princeton: Princeton University Press. 1991.
2. Fisch, Eric A., and Gregory B. White. *Secure Computers and Networks: Analysis, Design, and Implementation*. New York: CRC Press. 2000.
3. Holmström, Ursula. *User-centered Design of Security Software*. Human Factors in Telecommunications. Copenhagen, Denmark. May 1999. 18 Oct. 2000. <<http://www.tcm.hut.fi/Research/TeSSA/Papers/Holmstrom/hft99.ps>>
4. Olsen, Dan R. *Developing User Interfaces*. San Francisco: Morgan Kaufmann Publishers, Inc. 1998.
5. Oppliger, Rolf. *Security Technologies for the World Wide Web*. Boston: Artech House. 2000.
6. Raskin, Jef. *The Humane Interface: New Directions for Designing Interactive Systems*. Reading: Addison Wesley Longman, Inc. 2000.
7. Shneiderman, Ben. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. New York: Addison-Wesley Publishing Company. 1992.
8. Tiwana, Amrit. *Web Security*. Boston: Digital Press. 1999.
9. Whitten, Alma, J.D. Tygar. *Usability of Security: A Case Study*. Technical Report CMU-CS-98-115. Carnegie Mellon University, School of Computer Science. December 1998. 18 Oct. 2000. <<http://reports-archive.adm.cs.cmu.edu/anon/1998/CMU-CS-98-115.pdf>>
10. Whitten, Alma, and J. D. Tygar. "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0." Proceedings of the 8th USENIX Security Symposium, August 1999. 12 Sept. 2000. <<http://www.cs.cmu.edu/~alma/johnny.pdf>>

VII. Bibliography

- Borenstein, Nathaniel S. *Programming as if People Mattered: Friendly Programs, Software Engineering, and Other Noble Delusions*. Princeton: Princeton University Press. 1991.
- Fisch, Eric A., and Gregory B. White. *Secure Computers and Networks: Analysis, Design, and Implementation*. New York: CRC Press. 2000.
- Holmström, Ursula. *User-centered Design of Security Software*. Human Factors in Telecommunications. Copenhagen, Denmark. May 1999. 18 Oct. 2000. <<http://www.tcm.hut.fi/Research/TeSSA/Papers/Holmstrom/hft99.ps>>
- Olsen, Dan R. *Developing User Interfaces*. San Francisco: Morgan Kaufmann Publishers, Inc. 1998.
- Oppliger, Rolf. *Security Technologies for the World Wide Web*. Boston: Artech House. 2000.
- Raskin, Jef. *The Humane Interface: New Directions for Designing Interactive Systems*. Reading: Addison Wesley Longman, Inc. 2000.
- Shneiderman, Ben. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. New York: Addison-Wesley Publishing Company. 1992.
- Thomas, Richard C. *Long Term Human-Computer Interaction: An Exploratory Perspective*. New York: Springer. 1998.
- Tiwana, Amrit. *Web Security*. Boston: Digital Press. 1999.
- Whitten, Alma, J.D. Tygar. *Usability of Security: A Case Study*. Technical Report CMU-CS-98-115. Carnegie Mellon University, School of Computer Science. December 1998. 18 Oct. 2000. <<http://reports-archive.adm.cs.cmu.edu/anon/1998/CMU-CS-98-155.pdf>>
- Whitten, Alma, and J. D. Tygar. "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0." Proceedings of the 8th USENIX Security Symposium, August 1999. 12 Sept. 2000. <<http://www.cs.cmu.edu/~alma/johnny.pdf>>

Appendix A

This appendix describes the cost of the resources I will need for my project.

Budget and Equipment

As stated in Section IV Part D, I have my own computer to use to develop the application in this project. The cost of the Java tools I will need are unknown at this time, since I do not know all of the tools I need. In any case, I do not anticipate the cost being outside of my own personal range or my department's, should I deem it necessary to request the resources from them. Java tools are readily available online and offline, thus I should have no problems obtaining them during the time frame of this project.

Appendix B

A description of my academic and work related background is presented here as my résumé.

Jennifer N. Kahng

Current Address:

201 Smith
Charlottesville, VA
22904-2233
(804) 243-2141
e-mail: jnk7s@virginia.edu

Permanent Address:

36 Valmoore Drive
Poquoson, VA
23662-1244
(757) 868-4692

Education

- **University of Virginia** Charlottesville, VA (9/96 - present)

Degree: *Bachelor of Science, Computer Science* 5/2001, GPA 3.004/4.00

Academic program emphasized software development and design methods. Curriculum included: Programming, Software Development Methods, Computer Architecture, Algorithms, Digital Logic Design, Cryptography and Network Security, and Networks.

Work Experience

- **Energy Research Undergraduate Laboratory Fellowship** Berkeley, CA (6/00 - 8/00)

Research Center: Ernest Orlando Lawrence Berkeley National Laboratory

Position: Summer intern

Duties: Primary project was to create a user interface that controlled servo motors for the Advanced Light Source's beamlines. The motors controlled the intensity of radiation by varying the amount of light allowed through the beamline. Secondary project involved creating a user interface to control a stepper motor which rotated a mirror reflecting linearly polarized light. The laboratory's researchers wished to have circularly polarized light, thus a data acquisition aspect of the system was developed to aid in reaching the goal.

- **Summer Internships in Science and Engineering** Stanford, CA (6/99 - 8/99)

Research Center: Stanford Linear Accelerator Center

Position: Summer intern

Duties: Responsibilities included programming a software interface between a desktop computer running Windows NT Workstation 4.0 and an ESR Spectrometer in standard ANSI C. Serial port programming through the GUI interface was required. A small software package was prepared for possible integration with other software.

- **Langley Aerospace Research Summer Scholars** Hampton, VA (6/98 - 8/98)

Position: Summer intern

Duties: Primary projects dealt with web page design for the Composites and Polymers Branch and web available documents for the Technologies Applications Group. Previous experience in the field of chemistry facilitated the transition between technical information to a more readable, marketable product. The move to make technology transfer more accessible resulted in the need for its web availability.

Skills/Tools/Languages

C/C++ (ANSI/Visual), VHDL (FPGA Advantage), Perl, HTML, ASM (Programmer's WorkBench, Code View), Unix (AIX, Sun Solaris, Linux, FreeBSD, System V), LabWindows/CVI, information/security servers (httpd, ftpd, sshd), Windows 3.x/9x/NT 4.0, Macintosh OS 7.x/8.x, Corel WordPerfect Suite, MS Office 2000, Adobe PhotoShop/PageMill