CounterMeasures:

An Interactive Game for Security Training

A Major Qualifying Project Report

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

in Computer Science and Interactive Media and Game Development

by

Craig Jordan _____

Matt Knapp _____

Dan Mitchell _____

Date: March 2011

Approved:

_____

Prof. Mark Claypool, Co-Advisor

_____

Prof. Kathi Fisler, Co-Advisor

Abstract:

Computer security has become vital for protecting users, applications and data from harm. However, the United States is facing a severe deficit in the number of security professionals needed to adequately protect our computer systems. Often learning practical knowledge about computer security from textbooks and academic papers is less engaging and more time consuming then through simulations or games. Our hypothesis is that participating in a training simulation that closely emulates real-world systems is a better platform for learning security concepts than reading about security concepts in technical documents. The purpose of this MQP is to test our hypothesis through a game, called CounterMeasures, that teaches the basics of computer security. CounterMeasures provides the user with a real, interactive shell with which to practice security skills and challenges that are outlined through a series of objectives and help screens.

The CounterMeasures architecture is client-server based, with the client utilizing Adobe Flex and AIR, and the server consisting of BlazeDS, which uses a Tomcat webserver. The client connects via ssh to the virtual machine on the server after pulling the user information from a database. This ssh connection allows the user to interact with the machine and run commands, while the client displays the terminal responses, as well as their objectives and help screens.

To evaluate our system, we had 20 users participate in a study, with 10 users playing CounterMeasures and another 10 participating in a control group that received security reading materials. Pre- and post-questionnaires and measures of performance showed there was no significant difference in the knowledge gain between users in the control group and user playing CounterMeasures. However, users who played CounterMeasures could complete their goals in less time and also enjoyed the learning process more. Even users in the control group indicated that they would have preferred to have played a security game versus reading security material.

Table of Contents

I: Introduction

The increase in computer and network usage, with over 146% growth in users since 2000, along with the overall lack of technical knowledge by the public, has made computer security vital for protecting legitimate users and applications from harm [6]. From the average home user to International corporations and banks, securing computers and the networks on which they run is becoming increasingly important. United States federal officials have estimated there are only about 1,000 security specialists in the United States where between 10 and 30 thousand are needed [12]. As technology continues to spread at a rapid rate, security becomes increasingly important, due to the inability of security specialists to keep up with new attacks as well as the increasing skill levels and capabilities of the attackers.

Despite the importance of learning about security, it can be difficult to begin learning the relevant skills. Information about security and hacking techniques, such as breaking into and securing servers, is scattered and difficult to find at best. On top of this, much of the material is technical and specific to a particular technique, and thus not suitable as an entry point for partly technical beginners (such as college students). Adding to this problem is the lack of ways to practice cyber security related skills. Few texts or resources teach practical application of security techniques, instead teaching the theory behind the techniques. Also, practicing hacking techniques in a legal and harmless way is difficult. To practice attacks or to check if defenses are set up properly, various exploits must be used. It is illegal to break into systems that others own, and it can even be illegal to try and subvert software that was legally purchased. Practicing attacks or defenses also runs the risk of damaging the system on which the exploits or patches are run. The final problem with current security resources is the large knowledge base needed to even begin to learn from the resources. Many security resources require significant background in programming, networking, system administration, knowledge of different platforms and operating systems, and in many cases previous knowledge of security topics.

Games are becoming a popular tool for education in many subjects [15]. Games can engage and interest learners for long periods of time and can both teach and test complicated concepts. Computer security is an abstract, heavily rule based subject which is well suited for simulations and games. While there are a small number of games that teach security and a small number of games that allow for practicing security, to the best of our knowledge there are no games that teach the player computer security, allow the player to practice what they learn, and then test the player on what they have learned.

A short time ago, we participated in a capture the flag/wargame being run online [14]. Wargames are competitions between two or more teams, where each team gets a computer and must implement several different services and keep the services running securely, all while trying to attack the other teams' computers. The teams are then scored after each service implementation deadline and each team

receives a set number of points for correctly implementing the service. If at any time a team could prove a break into an opposing system they are awarded a large number of points and the opposing team would be given time to re-secure their system. While we were relatively experienced in programming and networking, worked with Linux systems before, and had some basic security background, when the game started we did not know where to start helping our team or what to start doing. Later, we were able to help with the programming portion of the challenges, but we knew little about the system administration aspects involved in security, as well as defense and offensive measures that we needed to take. Many of those involved were too busy with their own tasks to help out anyone who was new. Although the game was a valuable experience, it also pointed out many of the inadequacies in currently available materials.

We seek to address the lack of games teaching cyber security by creating a game that teaches specific techniques used by security experts and allows the player to practice in a game environment. The game we created, CounterMeasures, provides a single resource that teaches some security basics. The game assumes little previous security knowledge, provides a real, hands-on application of the skills being taught, all while making the content fun and engaging by putting it into a game environment.

Our first hypothesis is that participating in a training simulation that closely emulates real-world systems is a better platform for learning security concepts than reading about security concepts in technical documents. Our second hypothesis is that learning practical knowledge about cyber security from textbooks and academic papers is less engaging and more time consuming than learning through simulations or games.

In order to test our hypothesis, we designed a cyber security game that allows players to use the actual tools and techniques of security experts to complete four missions. The missions in CounterMeasures consist of objectives that require the player to utilize the skills they learn from the hints accompanying each mission. The training missions are straightforward and assist the player while separately teaching and testing some basic security skills. The normal missions provide less help and require the combination of skills, but the player is allowed to view additional hints at the cost of a lower game score.

To build CounterMeasures, we created a game client using Flex/Flash that runs on the player's computer using Adobe Air. The client connects to a Java server sending commands to a virtual machine and receiving mission information from XML files. We implemented 3 training missions teaching scanning, buffer overflows, and format strings and 1 normal mission testing the use of the 3 previously learned skills.

To assess whether CounterMeasures achieved its goals, we ran user testing over 3 days and attracted 20 participants. We split up the participants into an experimental group which played

CounterMeasures and a control group that read from a packet of condensed cyber security information. Both groups were told to complete the same final mission and both took a pre- and post-test questionnaire to measure their learning and attitude toward computer security.

The results of the user experiment show that the control group took on average roughly double the amount of time as the experimental group to complete the mission even though both groups show approximately the same learning. The experimental group reported more engagement using the game over the textbook learning of the control group.

The rest of this document is organized as follows: Section II is the related work where other forms of security learning are discussed and their contributions to CounterMeasures are discussed. Section III describes the game mechanics and content as well as the various technologies used in building the game. Section IV has the details of the user experiment design and testing. Section V provides the relevant data used to draw conclusions. Section VI contains the problems identified with the project but not fully solved. Section VII contains the conclusions, both stating and explaining the results of the project. Section VIII has the future work which describes how the project can be continued and other projects that would expand upon the goals of this project.

II: Related Work

This section covers books, games, and other work that is related to our project.

2.1 Computer Security Books

There are numerous books that cover computer security topics. The books range from basic books that discuss security for average users, such as setting strong passwords and not writing the password down, to advanced books that delve deeply into code and exploits behind security issues. Some of the simpler books can be found in the "For Dummies" series, including books on Hacking, Firewalls, and other security topics [16]. These books range greatly in the amount of background knowledge expected, as well. Some expect little previous knowledge, while others give almost no background and are only intended for people already experienced in the area.

One book on this subject is Hacking: The Art of Exploitation by Jon Erickson [3]. This book goes into the details of the code behind many exploits, down to C programming and writing shellcode. It covers the basics of programming in C, debugging with gdb, basic exploits and how to spot exploits, such as buffer overflows, format string vulnerabilities, networking, shellcode, countermeasures, and even crypology. Erickson's book does not assume too much background knowledge and builds from the ground up. However, if the reader is not technically inclined they may find themselves unable to understand the subject matter. Erickson also provides the source code for his examples and a Linux environment in which to run the examples. This is a good start, but CounterMeasures attempts to expand on this by providing a more guided environment, as well as providing all of the required resources bundled together.

Another series, Hacking Exposed, has several books covering various aspects of security. The books take a slightly higher level approach, describing many types of attacks and how they are carried out, but mostly avoiding specific code examples. The books also do not provide as much background on topics, assuming the reader has a good deal of knowledge about computers, systems, and networks.

These books provide a good basis for learning about security and hacking, which is similar to what this project is trying to provide, although they do not always give enough background. Also, while some of the books have samples to try, a lot of their material is just theory, with little practice behind it. Our project attempts to provide the same theory as the books but to also give the user some practical experience to aid in learning the concepts.

2.2 Computer Security Games

There are about a dozen games that use computer security concepts to form gameplay mechanics. These games range from simulations that utilize real applications and exploits to puzzle games where hacking is simply the theme of the game and is trivial to the core gameplay.

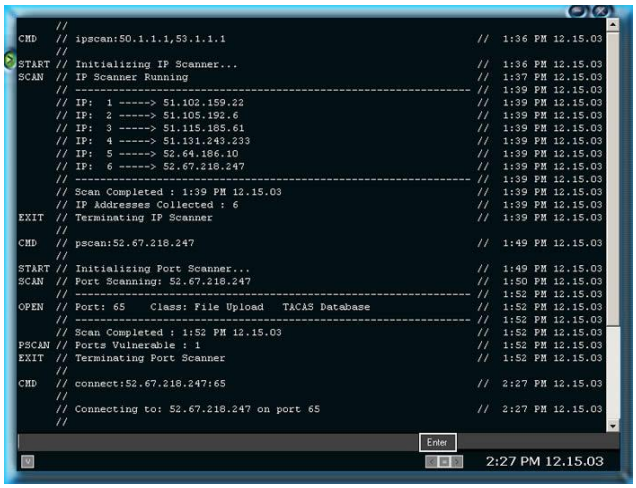

Figure 1: Street Hacker Main GUI

Street Hacker is a single player game developed by VirtuWeb Interactive and released in 2004. Street Hacker is played in a virtual environment in which the player follows a story that is told through mission-based gameplay. The interface, shown in Figure 1, is that of a fictional operating system which has similar functionality to many distributions of Linux. It is one of the more realistic games in that the operating system and file structure used in the virtual environment are similar to Linux and many of the commands the player uses perform the same function as the real commands of the same name used in an actual operating system. Our project also has mission-based gameplay and a realistic interface and emulates the realistic aspects of Street Hacker. However, CounterMeasures takes this a step further by making all of the commands real and providing an actual Linux shell for user interaction. The missions in

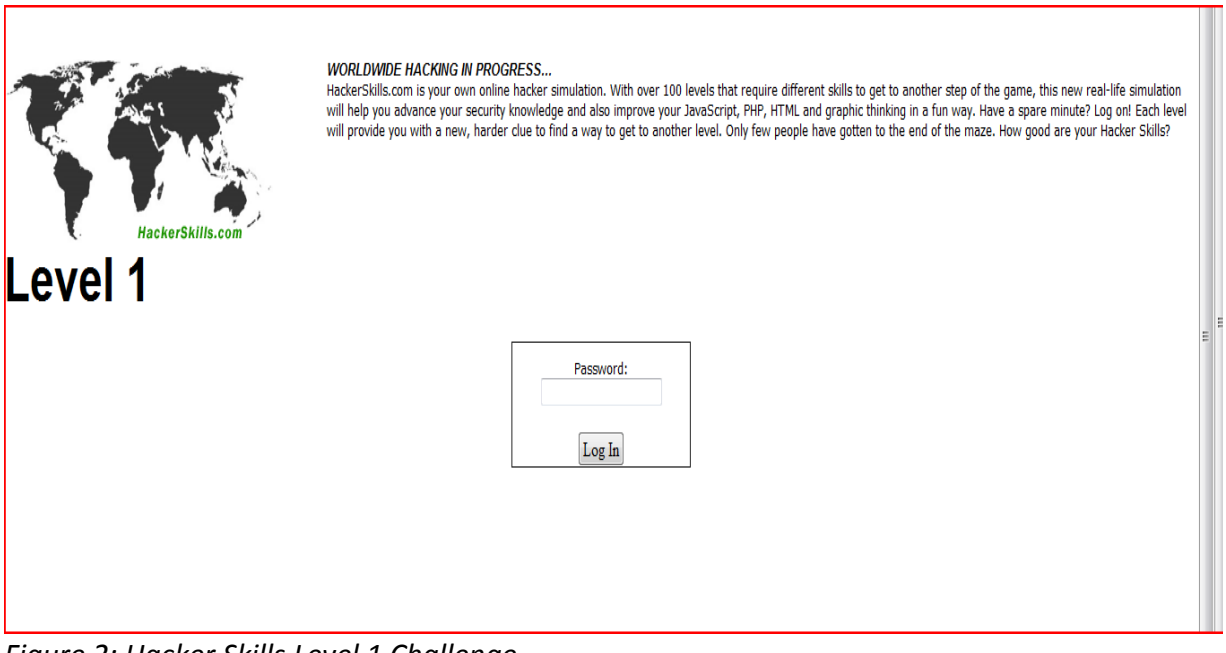

Figure 2: Hacker Skills Level 1 Challenge

CounterMeasures also use real code examples and systems, making the knowledge gained practical and immediately applicable to real systems.

Hacker Skills is a browser-based hacking simulation with gameplay varying from actual exploits to logic puzzles utilizing Web languages and development practices. The interface is minimal so as not to distract the player or give clues as to the solution of each puzzle. The gameplay featured in Hacker Skills both keeps the scope of the puzzles contained within Web development and promotes self-learning in that player receives only clues without explanations even if they do manage to solve a puzzle. Our project is about teaching, however, and provides explanations for exploits as the player uses each exploit; in addition our game maintains a base of knowledge about exploits that updates as the player progresses.

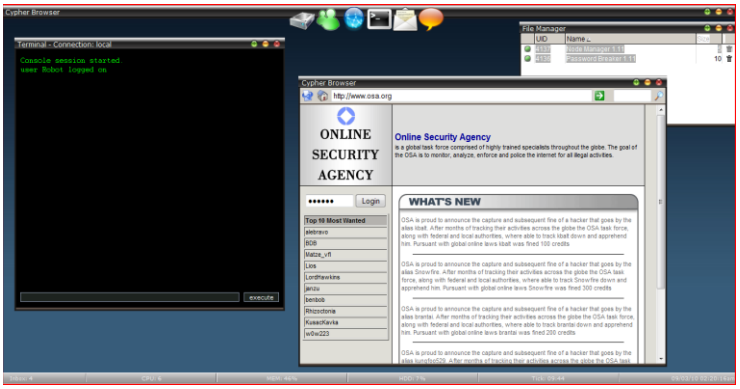

Figure 3: Cypher Mission Screen

Cypher is a browser-based multiplayer game developed by eXtremeCast in which players play alone or in teams to complete mission-based gameplay. The gameplay here is not realistic and computer security is simply a theme used to inform the story and interface. A fictitious operating system is used to keep the player under a single application while still providing functionality and variety for the game. Gameplay involves using various pseudo-programs (e.g. password crackers, encrypters, decrypters, file copiers) to break into different systems, steal passwords, or retrieve data from a server. All of these actions are used in the various scenarios the missions outline. More relevant to our project is Cypher's structure for missions in which not all objectives in a mission have to be completed. Completing extra objectives in a mission reward the player with additional information on future missions and our project has a similar incentive system.

There are many other games related to computer security, but the majority of these games are puzzle games with a hacking theme and offer no real education about hacking. Others are closer to simulations where education is the goal but all of those games keep a narrow scope similar to Hacker Skills. There are virtually no games which teach computer security with real exploits in place of abstract

gameplay. Our project attempts to turn a normal wargame into a single player simulation with a single interface to tie the information and functionality needed to learn about and practice computer security.

2.3 Computer Security Skill Testing

There are also several Web sites that provide sets of challenges designed to test various security knowledge or hacking skills. One example is http://www.hackthissite.org/ [4], a Web site set up with various challenges designed to teach various aspects of hacking, specifically Web site hacking (the ultimate challenge is to try and break into the site itself). The challenge categories are: Basic missions, Realistic missions, Application missions, Programming missions, Extbasic missions, Javascript missions, and Stego missions. Basic missions are designed to be straight forward challenges that teach the fundamentals of client-server HTTP hacking. Realistic missions are complete Web sites with built in security flaws. The skills learned in Realistic missions are meant to be able to be directly applied to real-world situations. Application missions are more focused on programming and operating system level security. Extbasic missions are about finding exploits through reading code. Javascript missions teach vulnerabilities in javascript. Stego missions are about finding where hidden information is stored. Some other sites are http://smashthestack.org/wargames.php [13] and http://www.overthewire.org/wargames/ [8] which focus on hosting wargames. http://www.crackmes.de/ [1] is a site focused specifically on reverse engineering.

The challenges found on http://www.hackthissite.org/ [4] are similar to the single player mode that our project provides. Their challenges focus on various aspects of security, help teach users techniques and give the user practical experience using what they learned about computer security. However, many of the computer security challenges do not offer much in the way of help, advice, or hints.

2.4 Computer Security Competitions

Several organizations and individuals run Web sites and events that provide a similar experience as to what we are trying to provide. One of the most well-known organizations is DEFCON [17]. DEFCON runs a CTF (capture the flag) event, where teams capture computers, keep services running, and protect the computers they have captured. Rather than having two or more teams fighting for each other's computer, there are other computers and the teams compete to get the most. There is an open round for everyone, and then the "real" event, where there is a qualifier beforehand, and the top 9 teams from the qualifier, plus last years winner, compete.

Another group that hosts computer security events is the SANS institute [18]. They provide various classes, both online and in person, teaching various security topics in a classroom setting. The SANS institute also provides several competitions that are open to everyone. One of the biggest competitions run by SANS is the SANS Netwars competition, aimed at high school, college, and even post-college individuals. The competition deals with various aspects of security, with some elements of CTF games.

One individual runs a Web site: http://p6drad-teel.net/~windo/wargame/ [14] where sometimes team versus team wargames are held. The wargames are generally between two to four teams, where each team is given a virtual machine to secure. The teams then attempt to break into other teams' virtual machines while maintaining control of their own virtual machine. To provide more of a challenge and more attack surfaces, each team has to keep services running during the course of the game. The services can vary between a simple Web site or ftp server to allowing the team to have a shell that can run some basic commands.

The general purpose of a wargame is to gain experience in securing and compromising servers in a situation similar to the real world. The games generally last until one team owns the boxes of all the other team(s). The games consist of a series of "cycles," with each team implementing a proposed service and attempting to compromise the other team's box. Some example services could be a Web server with forum software, an anonymous ftp server, or a public ssh server. Once a machine is compromised, the team has 12 hours to secure the compromised machine at which point they must put the machine back online and keep it online for 24 hours. If the team can keep all the services running for 24 hours without the box being compromised the team wins.

Competitions are helpful for learning practical applications, along with hacking and security techniques. In the CTF and wargame events at both DEFCON and on http://p6drad-teel.net/~windo/wargame/, teams must use real exploits on live systems that are actively being secured by other teams. However, many wargames have little or no introduction to the material, and participants are expected to already know most if not all of the techniques they need to succeed. This is especially true in the CTF and wargame events mentioned above. Players in CTF and wargame competitions are given no introduction, and the only help is from team members. Our project helps to emulate the practice found in the games, but also has a strong focus on helping users to get started and learn security topics, rather than expecting the users to know all about the topics beforehand.

2.5 Other Computer Security Sources

Another attempt at security education similar to our game is in the form of an operating system, called Damn Vulnerable Linux [19]. DVL was designed to be a vulnerable version of Linux, and also to have several old, highly vulnerable versions of software. DVL also has many built in security tools that are useful when attempting to secure or penetrate a system. The difference between our system and DVL is that DVL is an unstructured system whereas our game is a structured system that provides help in learning security techniques.

III: Architecture and Implementation

This section outlines the main elements of the game and their implementation. We begin by covering the basics of the game and how it is designed in section 3.1. Next we discuss the significant technical specifications and elements. Following this in section 3.3, we describe the gameplay of CounterMeasures, supported with screenshots that also demonstrate the user interface and its significant features. In section 3.4, we discuss the missions that are currently implemented in CounterMeasures and used for testing purposes.

3.1 Game Elements and Design

CounterMeasures is an educational game that can be used as a learning tool for computer security in a single player environment. It features missions that focus on a security concept and allows players to actually practice the attacks they are learning about in the mission. By utilizing a real system console, CounterMeasures emphasizes realism in both the scenarios behind and attacks being used in each mission.

CounterMeasures is designed as a single-player game. The player is guided through several missions, each of which teaches a new aspect of hacking or security. Each mission has a title, a description, a score for completing the mission, a skill that is the focus of the mission, objectives that must be accomplished to complete the mission, help that is given to guide the user, and a list of commands learned during the mission. The missions build upon each other, allowing the player to utilize previously learned skills in each new mission. Players are given a fully functional shell that allows the player to run commands and complete missions. Some commands are blacklisted, or not allowed, to prevent the user from causing too much damage to the test environment.

The scoring system for CounterMeasures is based on the static score associated with each mission. When a player completes a mission, their score is incremented by the score associated with the mission. Additionally, for the more difficult missions, some help is optional. The user is warned that using any additional help will subtract from the total score than can receive for completing the mission.

CounterMeasures assumes some previous technical knowledge from players. This consists primarily of Linux command line familiarity, since the system imitates or uses the same tools as are used for real attacks which are often run through a Linux command line. Players are also expected to have some familiarity with running and interacting with C programs in a Linux environment, as well as basic knowledge of how computers are connected by networks. The players are not required to have previous computer security knowledge to play CounterMeasures as they learn all security related knowledge necessary throughout the missions.

3.2 Technical Specifications

CounterMeasures uses a client-server architecture. The CounterMeasures client is installed on users' machines. This client application is developed using Adobe Flex [http://www.adobe.com/products/flex/], which utilizes the Java programming language. Flex is an Adobe product designed to allow for easy Web GUI development. AIR [http://www.adobe.com/products/air/], another Adobe product, allows Flex applications to run as desktop applications. If desired, CounterMeasures can be ported to a Web application, as the interface is already written in Flex. This could provide an easier way to allow users to play CounterMeasures, as well as a simpler system to update CounterMeasures. AIR and Java also have the benefit of ensuring the client is easily portable to multiple systems, including Windows, Linux, and Mac. The client program contains the graphical user interface (GUI), with which the user interacts. The CounterMeasures server handles most calculations, retrieval of data, and running commands.

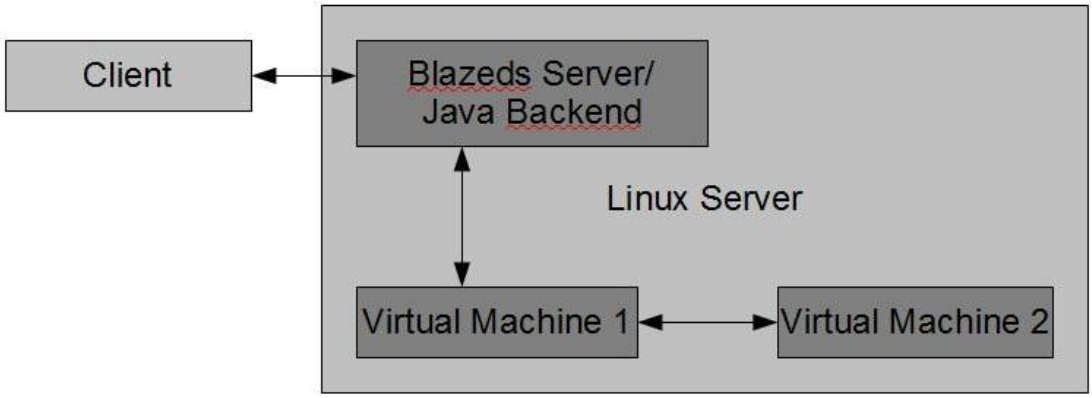

Figure 4: System Diagram

The CounterMeasures server is a Linux PC, running the newest and fully patched version of Ubuntu (10.04, Lucid Lynx). This machine is set up to allow incoming connections from clients. The

CounterMeasures backend on the server has several pieces. First, the backend has Java code that runs, acting as the server side to which the client programs connects. The server contains the logic and calculations behind CounterMeasure and is powered by BlazeDS [http://opensource.adobe.com/wiki/display/blazeds/BlazeDS] and a Tomcat Web server [http://tomcat.apache.org/]. BlazeDS is a server-based Java technology allowing for easy integration with Adobe Flex and Adobe AIR. BlazeDS utilizes Tomcat, which is an open source Java Servlet and JavaServer Pages technology.

Second, there are two virtual machines, powered by VirtualBox [http://www.virtualbox.org/], which is an open source virtualization product. Each machine is running BackTrack 4 [http://www.backtrack-linux.org/], a Linux distribution designed to be used for penetration testing and with several built in security tools and vulnerable versions of programs. Each virtual machine is set up to allow ssh so that connections can be established. One machine is used for client connections; player's actions run on this machine. The second machine is used for attacking purposes, as some missions call for attacking or scanning remote systems. These machines have the programs that players use while playing the game. These programs have been compiled with the ***–fno-stack-protector*** option, to allow stack smashing, where appropriate.

Third, CounterMeasures utilizes a database which contains player information, such as login, password, and mission information for the player. The database can be established on the same machine, but for our purposes, the database was hosted on WPI oracle servers.

When a user starts their client and connects to the server, several events happen. First, the Flex client establishes a connection to the BlazeDS server through Tomcat Web server. After this JDBC is used to connect from the BlazeDS server to the Oracle database to gather relevant user information, such as current score, completed missions, and current mission. Next, the server uses jsch [http://www.jcraft.com/jsch/], a Java secure shell library, to connect to one of the virtual machines. This established shell allows users to type commands into a prompt on their client, which are then piped to the server, which sends the commands to the virtual machine to be executed, returning the results. The process of sending commands and receiving results is accomplished using a stream from client to server, so the user appears to have an active shell in their client GUI. The setup and lifetime is shown in Figure 8.

Another important implementation decision is how the mission information is stored. We wanted the missions to be easily updated, so that we and others could add more content to CounterMeasures. All of the mission information is stored in an XML document, which is defined by an XML schema document. This schema is compiled using XML Beans [http://xmlbeans.apache.org/], which provides

classes and methods to access any XML document conforming to the XML schema document. This allows the Java server code easy access to the mission information. Each mission has a title, description, score, skill, a list of objectives, a list of help content, and a list of commands that the player learns during the mission. The objectives stored in the XML file are simply the text for each objective. The logic for checking for completed objectives is stored in the Java server side code.

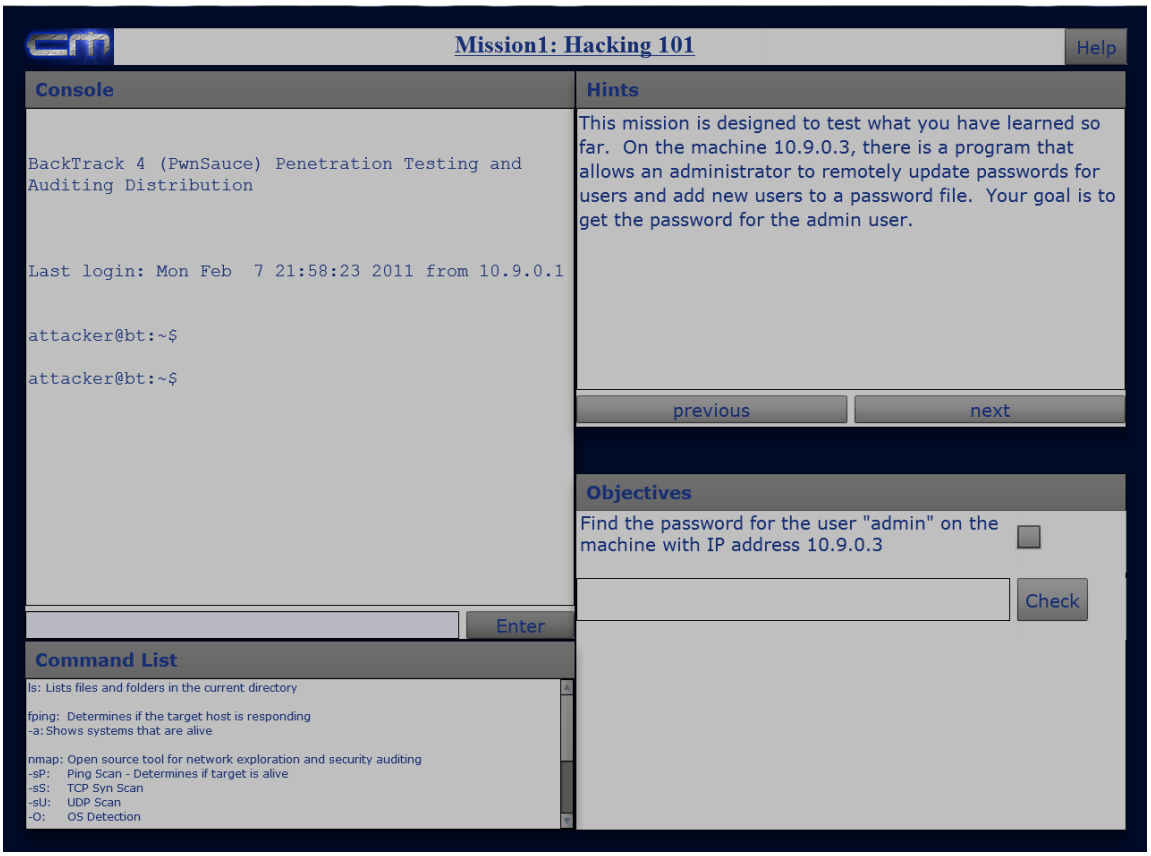

Figure 5: Interface in CounterMeasures

3.3 Game-play and Interface Design

Figure 5 shows the interface for CounterMeasures. The console is in the upper left of the screen, where the player enters the commands to run and the output from the commands is displayed. The hints section to the right of the console contains up-to-the minute help that gives the player hints about the usage of commands and insight as to how to go about completing the next objective. The objectives section tells the player the next goal and the overall goal for completing the mission. A list of all the commands the player has learned in previous missions appears in a window below the console.

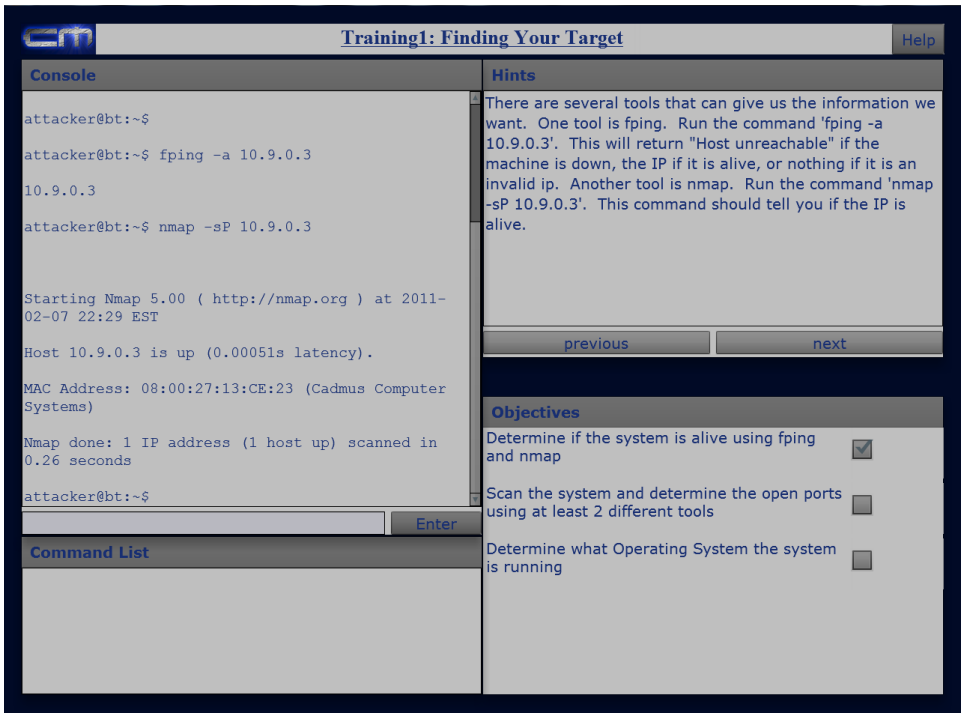

Figure 6: Interface after the first objective was completed

The sample of gameplay shown in the following three Figures, 6, 7, and 8, is from the first mission. The first mission deals with different methods that can be used to scan a target system. The hints describe how to determine if the target is alive by using the ***fping*** command, giving the usage for the command. The hints also suggest that the same information could be discoverable using the ***nmap*** command. Both ***fping*** and ***nmap*** can be used to make the target system respond in some way. The user is supposed to run the commands to assess whether the target machine is active.

The screen in Figure 6 shows what happens after the user has run the ***fping*** and ***nmap*** commands. There is now output in the console showing that the target system is alive and the objective for determining if the system is alive has been marked off by the system.

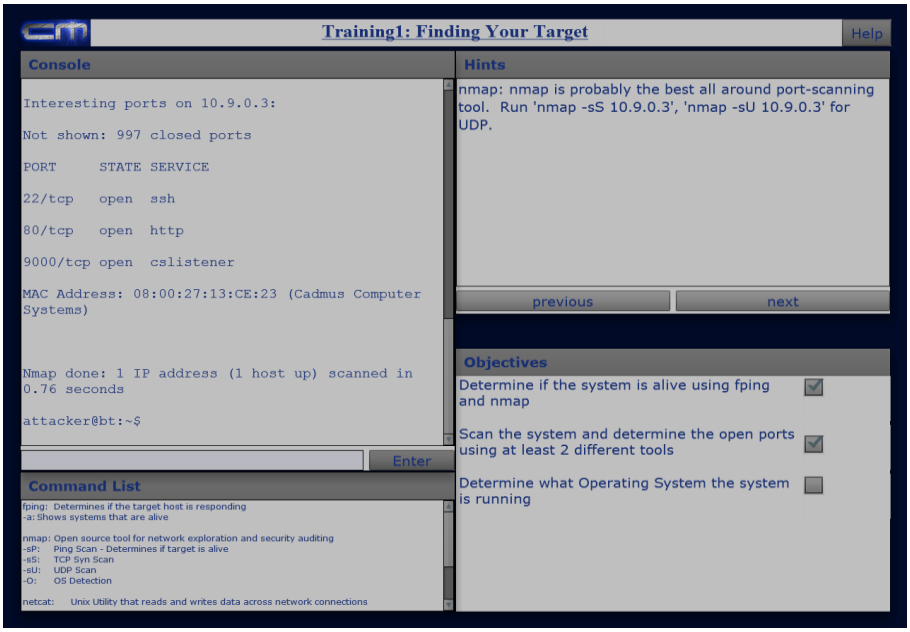

Figure 7: Interface after the second objective was completed

Figure 7 shows the aftermath of the user running both the ***netcat*** and ***nmap -sS*** commands. Netcat and nmap are multi-use IP address and port scanning utilities which can determine the running services and operating system on a target system. The console shows which ports the scan found to be open as well as nmap's best guess as to the service that is running on that port.

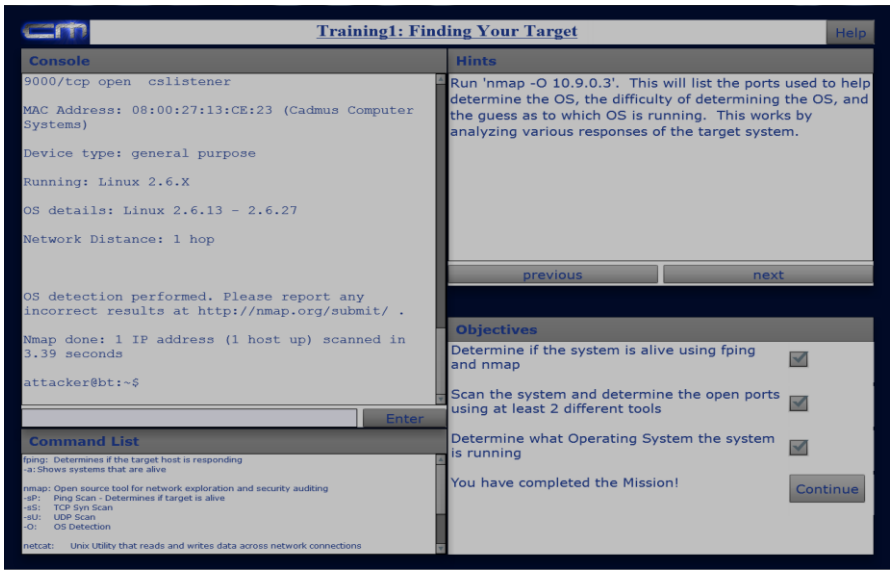

Figure 8: Interface after the third objective was completed

Figure 8 shows the result of the user running the ***nmap*** command with the ***-O*** option. The ***nmap*** command run with the ***-O*** option is used to determine what operating system the target

machine is running or at least the command's best guess. The console output shows that the ***nmap -O*** command determined the operating system to be Linux. The objective for determining the operating system has been checked off. The hints screen now describes what the output in the console means and informs the user that the end of the mission has been reached. Now that the final objective has been marked as completed the Continue button appears and the user can continue to the next mission.

3.4 Mission Content

Gameplay is broken up into two levels, Training and Missions. Training levels are simple missions that focus on one specific aspect of security with the objectives spelled out for the player in the help section. Missions are more complicated and generally combine several aspects of security that were already covered in other missions or training levels. Missions only have basic help, leaving the players to figure out how to complete the mission objectives own their own. The following list of missions describes the description and objectives as they are stated to the player.

Training 1: Finding Your Target

Description:

This training mission teaches the basics of scanning a remote system given its IP. You will learn how to determine if a system is alive, what OS it is running, and what ports are running/what services are running on those ports.

Objectives:

- Determine if the system is alive using fping and nmap
- Scan the system and determine the open ports using at least 2 different tools
- Determine what Operating System the system is running

Training 2: Bypassing Authentication

Description

This training mission teaches the basics of a buffer overflow attack. You are given a program to exploit using this technique.

Objectives

- Get the key protected by the program "adminKey".

Training 3: Discovering Passwords

Description

This training mission teaches the basics of format string vulnerabilities. You learn how to take advantage of insecure code and use format strings to access program memory.

Objectives

- Get the database password used in the program "access_database".

Mission 1: Hacking 101

Description

This mission tests you on all of the skills you have learned so far.

Objectives

- Find the password for the user "admin" on the machine with IP address 10.9.0.3

IV: Evaluation

This section discusses how the evaluation was designed and how it accomplishes the goal of testing whether participating in a training simulation or game can teach security concepts better, be more engaging, and less time consuming then learning through textbooks and academic papers. Section 4.2 covers the player demographic for which our study will apply. Section 4.3 discusses the design of the evaluation and how we conducted the user study. Section 4.4 provides the environment setup and location, and section 4.5 describes the method of obtaining users to participate in the study.

4.1 Main Goal

The main goal of the project is to educate users about cyber security in a way that is fun. The content is designed to be as close to real situations as possible by emulating real world systems and to provide a hands-on experience, so that users can take the knowledge from the single player game and immediately apply it to real-world or competitive situations, such as a CTF game. The process is intended to be fun, so that users are more likely to continue through the game and seek to further their security skills and knowledge with outside sources. There are two aspects used to evaluate the success of this project: *education* and *fun*. Education refers to how much the game was able to teach the participants in the study. Fun refers to how enjoyable playing the game was for the participants in the study.

4.2 Demographic

The target demographic for this project is people who have a general technical background. This includes students studying Computer Science, Information Technology, or related fields at the college

level. Players are expected to be technically knowledgeable, have some basic knowledge of Linux/Unix systems, and know simple concepts of how programs work in a computing environment.

4.3 Design

The user study has two main methods for evaluation: performance statistics and a questionnaire. Users are divided into two groups, an experimental group and a control group. Users are split between the experimental and control group based on arrival. We alternate putting users in the control or experimental group. Each group is given an identical questionnaire at the beginning, consisting of basic security related questions and is asked to answer to the best of their abilities. The control group is given a selection of reading material covering the security topics covered in the game and asked to spend approximately 25 minutes (the approximate duration of the game) reading the material. The experimental group gets a chance to play through all of the training missions in the game. Then, both groups are given the final challenge of the game the experimental group plays through the first real mission, and the control group is given the same instructions and a shell on a PC to use. The time for completion and whether or not each player completes the mission is recorded, as well as the number of help screens needed.

At the end of the experiment, users from both groups are given another identical questionnaire. This questionnaire contains security-related questions similar to those on the first questionnaire, with additional questions related to the users' enjoyment of the experiment. The questionnaire also asks whether or not the user plans on continuing to learn more about security and is interested in future missions for the game.

4.4 Environment Setup

The study took place in the Fossil Laboratory at WPI. There were two laptops setup running CounterMeasures and two other computers with a shell setup for users in the control group. The computers with the shells were for users in the control group for the experiment. Each computer was separated, so that a user at one computer could not see what a user at another computer was doing. As users arrived, they were given the pre-study questionnaire and then shown to either a laptop running CounterMeasures or a computer with the shell. After completing the study, all users were given a post-questionnaire to complete. This study was carried out between 3 and 5 PM EST each day between, 2/16/2011 – 2/18/2011.

4.5 Obtaining users

To obtain users for this experiment, we sent an email to students at WPI who were majoring in Computer Science and Interactive Media and Game Development. We also solicited participants from a sophomore/junior Operating Systems class. Students were informed that we needed users to participate in a study for our MQP relating to security. Students were told that they could expect the study to take approximately 30 minutes and that pizza would be provided to those who participated.

V: Analysis

In this section we analyze the data that we collected from the experimental and control groups. We had 20 participants total, with 10 participants in the control group and 10 participants in the experimental group.

5.1 Demographics

All of the users that participated in our experiment were undergraduate students at WPI and most users came from the Operating Systems course. All had Computer Science backgrounds and were interested in computer security but virtually none had previous computer security experience.

5.2 Test Scores

The maximum possible score for both pre- and post-tests were 14 points. The maximum achieved score in the pre-test was 11 in the control group and 10 in the experimental group. The minimum achieved score in the pre-test was 2 in the control group and 1 in the experimental group. The maximum achieved score in the post-test was 13 in the control group and 14 in the experimental group. The minimum achieved score in the post-test was 9 in the control group and 7 in the experimental group.

Maximum Possible Score (Pre- and Post-test)		14
Experimental Pre-Test	Min Score	1
	Max Score	10
Experimental Post-Test	Min Score	7
	Max Score	14
Control Pre-Test	Min Score	2
	Max Score	11
Control Post-Test	Min Score	9
	Max Score	13

Table 1 Ranges of Scores for Experimental and Control Pre- and Post-Tests

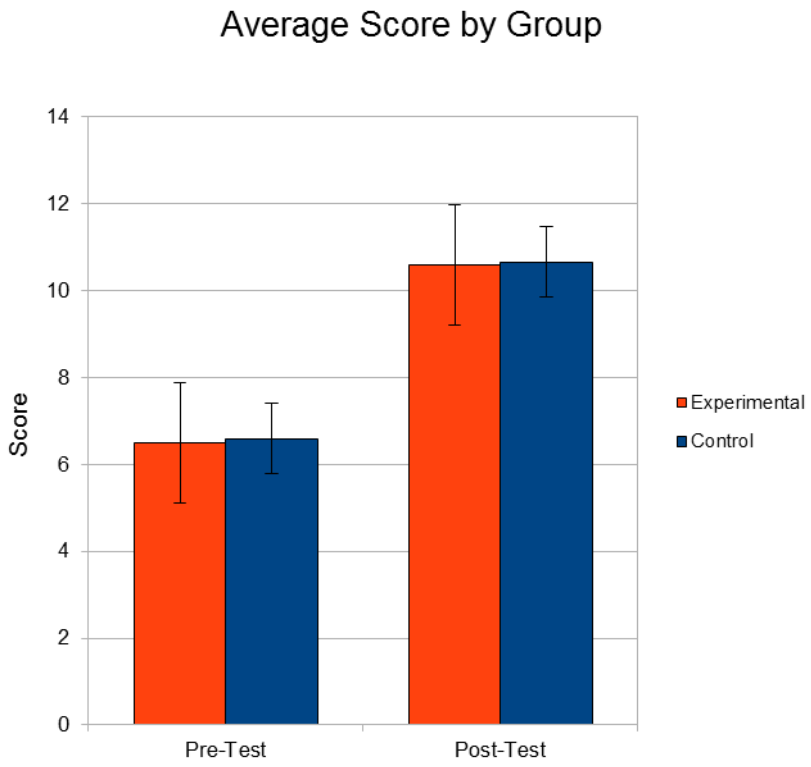

Figure 9 Graph that shows the Average Scores on the Pre- and Post-tests for the Experimental and Control Groups with 95% Confidence Intervals

In Figure 9 the y-axis indicates the score and the x-axis indicates the pre-test or the post-test. The two bars are the averages of the scores the users got on the pre- and post-test questionnaires. The scores in the control and experimental groups were approximately the same in both the pre- and post-test. All of the participants in the experiment improved their scores by playing CounterMeasures or by reading the text. For both groups the most common mistakes in the post-test were on questions 5 and 6, both of which had multiple answers. Most users got the question partially correct meaning they circled some of the answers but not all.

5.3 Time

In the control group the minimum time to complete the mission was 38 minutes and the maximum time was 86 minutes. In the experimental group the minimum time was 11 minutes and the maximum time was 63 minutes.

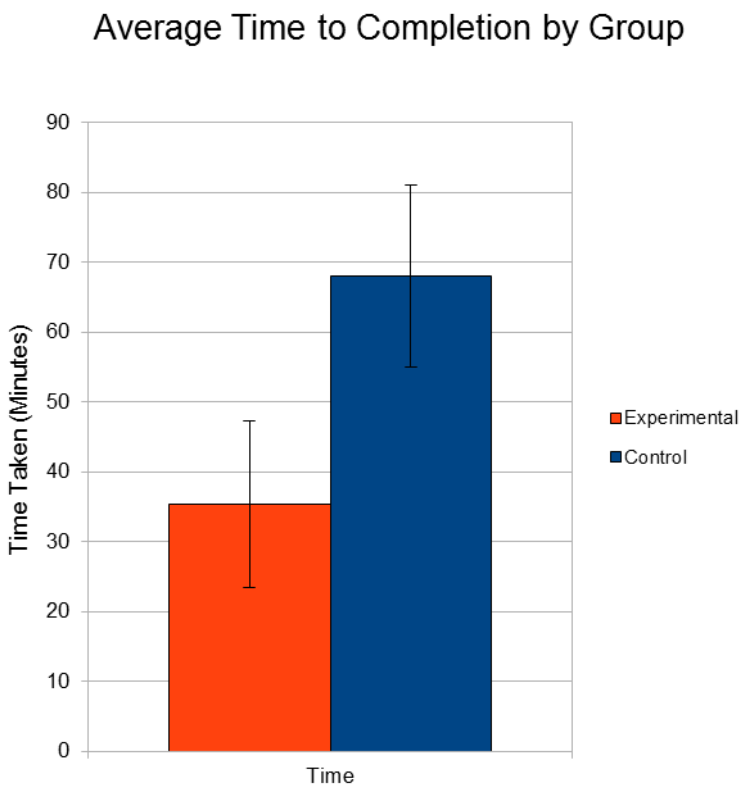

Figure 10 Graph that Shows the Average Time to Completion for the Experimental and Control Groups with 95% Confidence Intervals

In Figure 10 the y-axis is the average amount of time taken to complete the experiment and the two different bars represent the experimental and control groups. The error bars show the 95% confidence intervals for each value. Users in the experimental group took on average about half the time it took users in the control group to complete their given missions.

5.4 User Opinion

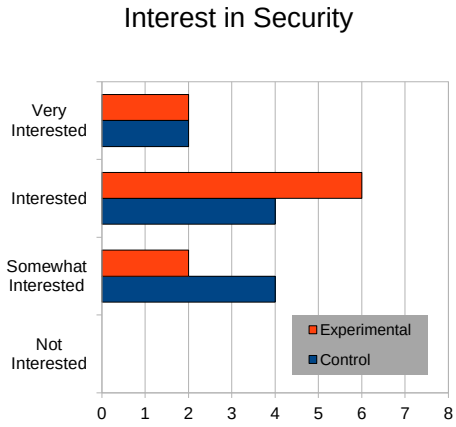

Figure 11 Interest in computer security before experiment

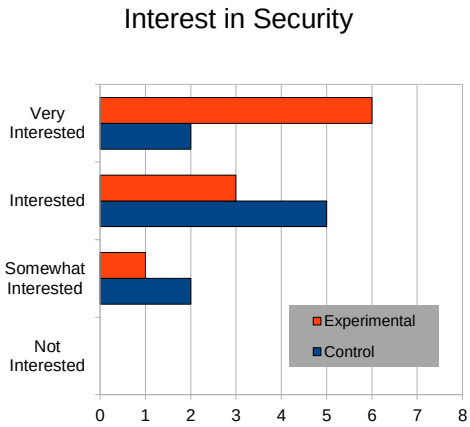

Figure 12 Interest in computer security after experiment

Figure 11 shows how interested participants were in computer security before the experiment. Figure 12 shows how interested participants were in computer security after the experiment. The y-axis for both graphs shows the different choices available for the participant to pick. The x-axis for both graphs shows the number of participants who chose that response. Users in the experimental group were more interested in security after playing CounterMeasures while interest users in the control group remained the same.

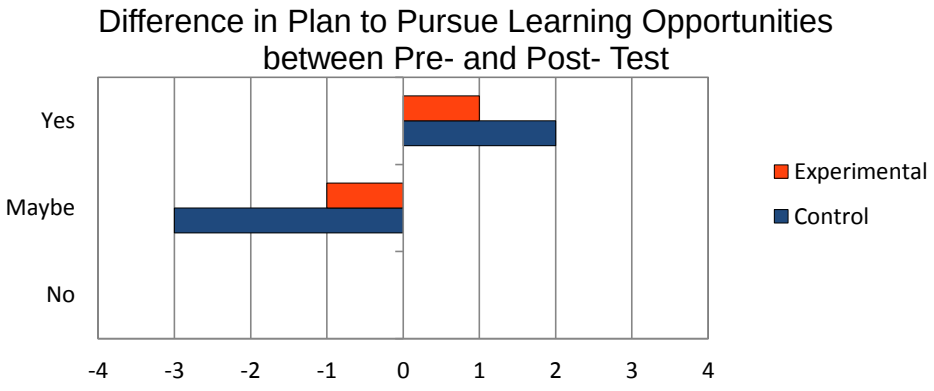

Figure 13 Difference in User's Plan to Pursue Learning Opportunities between Pre- and Post-Tests

Figure 13 shows the change in users' plans as to whether or not to pursue learning opportunities in security. The y-axis shows the three choices available for the user to pick: yes, maybe, or no. The x-axis shows the change in aggregate values between the pre- and post-tests. Both the users in the control and experimental groups plan to pursue learning opportunities after the experiment is over.

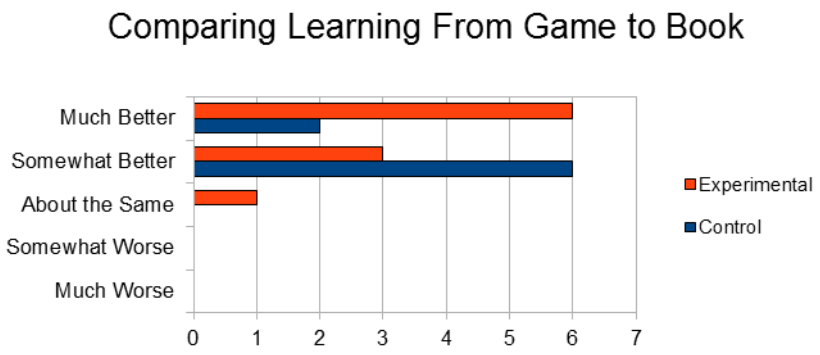

Figure 14 Users' opinions on learning from game compared to learning from book

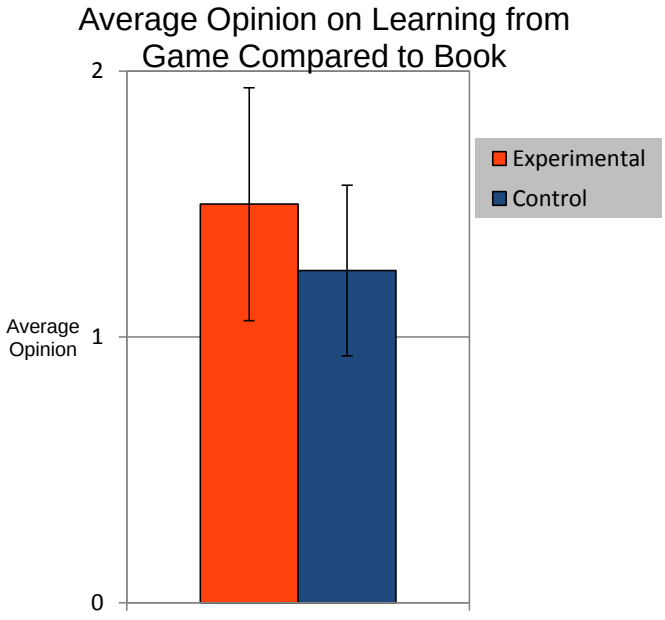

Figure 15 Average User Opinions on Learning from a Game Compared to Learning from a Book

Figure 14 shows the count of the users' opinions on comparing learning from a game to learning from a book. Figure 15 shows the average opinion for the users of each group on a 5-point scale with much worse being a -2 and much better being a 2. The error bars in Figure 15 show the 95% confidence intervals. All of the participants reported that they enjoyed, or would have enjoyed in the case of the control group, learning computer security more from a game environment than from a book, but more experimental users rated learning from a game as much better than learning from a book.

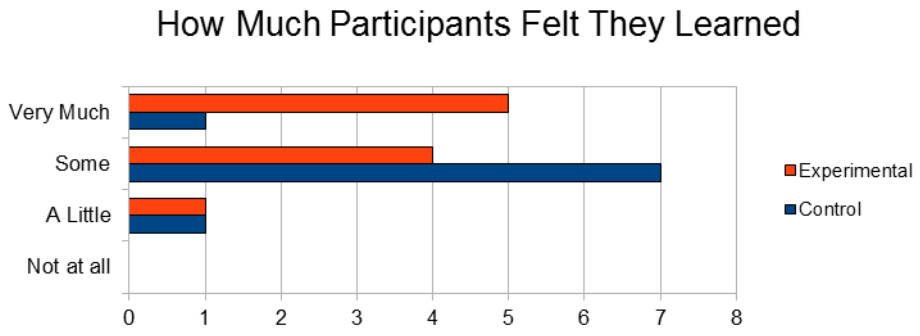

Figure 16 Graph of how much Participants felt they Learned

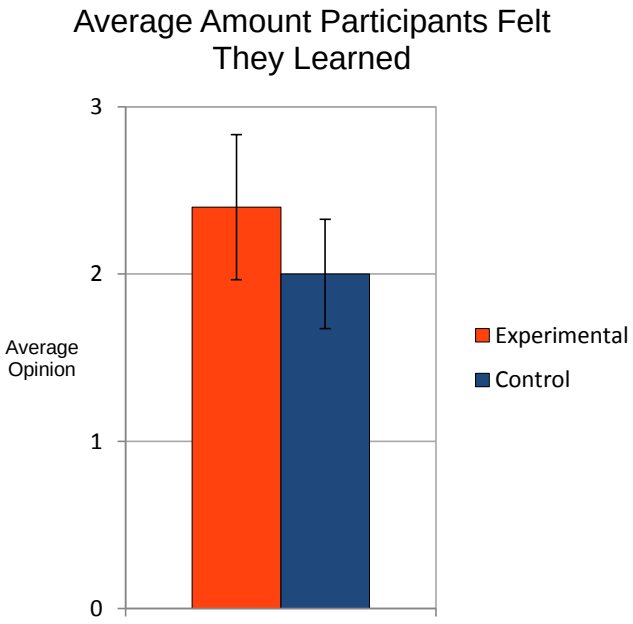

Figure 17 Graph of the Average Amount Participants felt that they learned

Figure 16 shows how much the participants in the study felt that they learned. The y-axis is the possible responses the users could give and the x-axis is count of how many times that response was given. Figure

17 shows the average of the amount that the users felt they learned on a scale of 0 to 3 with 0 being the "not at all" response and 3 being the "very much" response. The experimental users were more confident than the users in the control group in the amount of computer security knowledge that they perceived themselves to have learned over the course of the game. However, the quiz scores shown in Figure 11 did not back up this perception.

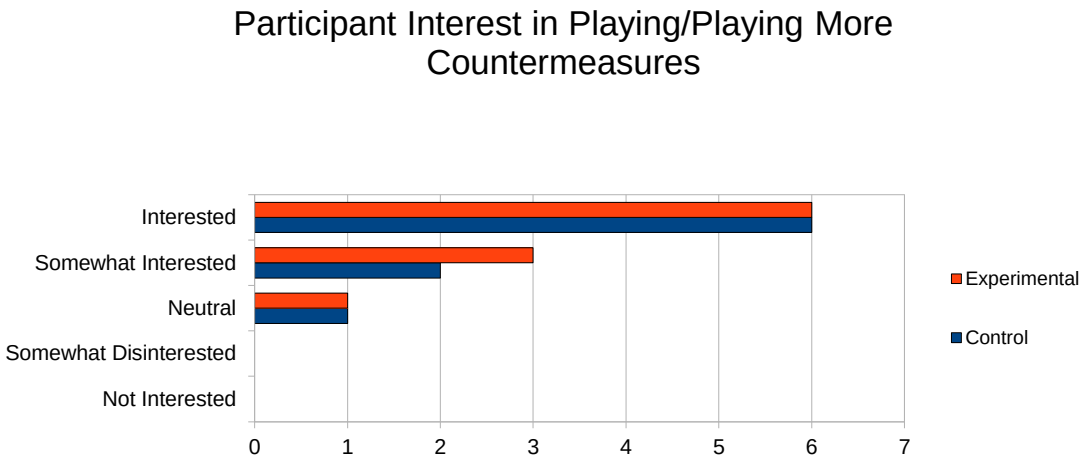

Figure 18 Graph showing interest in additional computer security games

Figure 18 shows how interested were in playing/playing more CounterMeasures. The y-axis is the possible responses that users could give and the x-axis is the number of times that response was given. Both the experimental and control users were interested in playing additional games to learn about computer security.

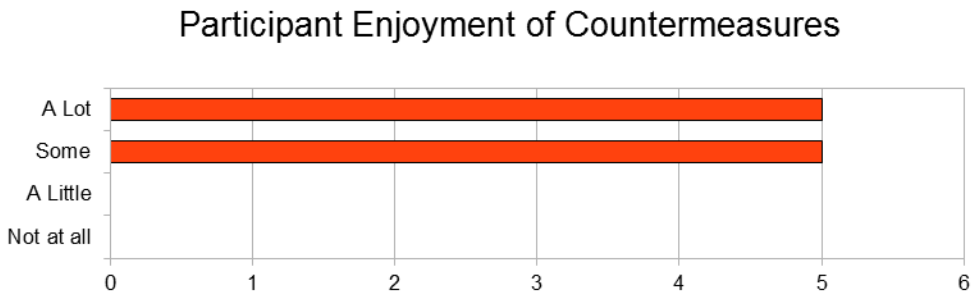

Figure 19 Graph showing how much users enjoyed playing CounterMeasures

Figure 19 shows how much users enjoyed playing CounterMeasures. The y-axis is the possible responses that the users could give in response to the question and the x-axis is the number of times that the response was given. The experimental users overall gave responses that indicated they enjoyed playing CounterMeasures.

5.5 Summary

The results showed that both the control and experimental groups learned about the same amount with both groups averaging about 6.5 on the pre-test and 10.6 on the post-test. However, although both groups learned about the same amount the experimental users averaged about 35 minutes to complete the experiment while the control group averaged about 70 minutes. Both groups showed an interest in either playing CounterMeasures, for the control users or playing more CounterMeasures, for the experimental users.

VI: Unresolved Issues

This section deals with issues and known bugs in the game that we were unable to fix. All of the major unresolved issues with the functionality of the game stem from the shell employed by users to enter commands. The shell was designed to act as a fully functional Linux shell with some restricted commands, and to prevent the user from executing commands that could negatively impact game-play for themselves or other users. However, some commands did not work fully on the shell we created, including man pages. Another limiting factor was that the user could not use the shell to open documents or use a text editor. Most commands that required additional input past the first command or did not return a prompt seemed to cause trouble for the shell. Also, if a command was entered incorrectly or a program hung, this would sometimes cause the shell to stop functioning, and there was no way to kill the process or command. If the shell stopped functioning, the game had to be restarted. The final limiting factor of the shell was that at times some text was not fully returned, for reasons unknown.

VII: Conclusion

Computer security has become vital for protecting users, applications and data from harm due to increased computer and network usage by users with a lack of technical knowledge. The purpose of this project was to teach computer security in a new way that is more accessible to users than reading textbooks full of information. The platform that was tested here teaches computer security through a game. We developed a system using a Flash front-end and a Java back-end that guides players through a

few tutorials designed to teach about scanning systems, buffer overflows and string format vulnerabilities and then apply that knowledge to extract a password from a program that had been written to allow exploitation. The Experiments were conducted to determine if the game, CounterMeasures, was as effective a tool at teaching security concepts as was the reading in textbooks to learn the concepts.

The results of the experiments indicate that CounterMeasures teaches the same measurable amount of knowledge about computer security but in a shorter period of time and with a higher rate of user engagement. CounterMeasures also made players more confident in their knowledge, but this may be due to the fact that the experimental group had less background knowledge than the control group. This might have caused the control group to feel as if there was more information presented than they were tested on while the experimental group was only tested on the knowledge the game provided. The game was more immersive than a book. All participants in the experiment were initially interested in computer security and all reported wanting to pursue learning opportunities after the experiment was over. After playing, the users were more likely to want to learn about computer security in the future. This response is attributed to the experiment piquing the user's interest in computer security by having the users actively practice computer security skills.

VIII: Future Work

There are several different directions this project could take in the future. One possibility would be to directly extend CounterMeasures. This would primarily include adding more content in the form of extra and more complex missions with more training to prepare the players for the added missions. Missions consist of an entry in an XML document. The entry describes the mission name, the objectives, the number of points the mission is worth, and the help available to the user for the mission. . This effort would also need to include a fix for the shell, as several of the unresolved issues mentioned will seriously hinder any attempts to create more advanced missions.

Another possibility would be to keep CounterMeasures as it is now, but remove the shell. Instead, have the player use their own shell (or a tool like *Putty* for Windows) and have the rest of the game not take up the full screen. A feature could even be implemented to automatically open up a shell and ssh into the proper machine. This path would keep the essence of the game, but lose the complications of trying to provide a fully working shell in Java and Flex. Negatively, it would be more difficult for the users because they would have to perform setup steps before playing the game that the system would otherwise handle.

Another direction that could be taken would be to create a Linux environment within which the player would play the game. The environment would be customized to have the missions and hints easily

accessible, and otherwise be a fully functional Linux operating system. The game environment could be distributed in the form of a disk image, and there could be servers set up for the users to attack or scan. A Web server could be set up to keep track of scoring. This system would be similar to the way NetWars is currently implemented, but could be enhanced to use the basic mission and help features we have outlined in CounterMeasures.

One final idea would be to take the game in a multiplayer direction. This could be stand-alone, or it could be built on top of one of the other ideas, where the player is trained in single player mode first, and then advanced into the multiplayer game. The multiplayer game could follow a Capture the Flag Format, where competing teams try to gain access to various insecure servers setup for the game. Structure could be provided in the forms of hints on how to defend and attack systems, as well as the various services that will be running on each system and vulnerable to attack. The mission structure would be used to guide teams through the game, rather than individual missions.

In addition, more gameplay elements could be added to make CounterMeasures into more like a game and less like a tutorial. Some of the elements that could be added are things such as adding a time constraint to each mission or adding a story for the player to discover as they play through the game. Another way to enhance the game would be to implement a more complicated scoring system instead of the simple format of getting a static number of points for completing each mission.

IX: References

[1] *Crackmes.de*. Web. Sept. 2010. <http://www.crackmes.de/>.

[2] *Cypher!* Web. Sept. 2010. <http://cypher.extremecast.com/index.php>.

[3] Erickson, Jon. *Hacking the Art of Exploitation*. San Francisco, Calif: No Starch, 2008.

[4] *Hack This Site!* Web. Sept. 2010. <http://www.hackthissite.org/>.

[5] *HackerSkills.com*. Web. Sept. 2010. <http://www.hackerskills.com/>.

[6] "Internet Usage Statistics, Population and Telecom Reports for the Americas." *Internet World Stats - Usage and Population Statistics*. 30 June 2010. Web. 24 Feb. 2011. <http://www.internetworldstats.com/stats2.htm>.

[7] Lockhart, Andrew. *Network Security Hacks.* Farnham: O'Reilly, 2006.

[8] *OverTheWire - Wargames*. Web. Sept. 2010. <http://www.overthewire.org/wargames/>.

[9] Peikari, Cyrus, and Anton Chuvakin. *Security Warrior*. Beijing: O'Reilly & Associates, 2004.

[10] Scambray, Joel, Stuart McClure, and George Kurtz. *Hacking Exposed: Network Security Secrets & Solutions*. Berkeley, CA: Osborne/McGraw-Hill, 2001. Print.

[11] *Street Hacker.com*. Web. Sept. 2010. <http://www.streethacker.com/>.

[12] "U.S. Cyber Challenge – Major Shortage of Cyber Security Workforce Professionals | USADefenseIndustryJobs.com." *USADefenseIndustryJobs.com | Jobs and Careers – Defense Systems*. 30 Jan. 2011. Web. 24 Feb. 2011. <http://www.usadefenseindustryjobs.com/2011/01/30/u-s-cyber-challenge-major-shortage-of-cyber-security-workforce-professionals/>.

[13] "Wargames." *SmashTheStack Wargaming Network*. Web. Sept. 2010. <http://smashthestack.org/wargames.php>.

[14] Web. Sept. 2010. <http://p6drad-teel.net/~windo/wargame/>.

[15] “Digital Game-Based Learning” Marc Prensky. Oct. 2003. Web. Feb 2011. <http://210.240.189.212/dctelearning/type_resources/01_papers/9612_digital_papers/2_english/BIT095103/digital%20game-based%20learning.pdf>.

[16] "Security - For Dummies." *How-To Help and Videos - For Dummies*. Web. Sept. 2010. <http://www.dummies.com/store/Computers-Internet/Security.html>.

[17] *DEF CON® Hacking Conference - The Hacker Community's Foremost Social Network.* Web. Sept. 2010. <http://www.defcon.org/>.

[18] *SANS: Computer Security Training, Network Security Research, InfoSec Resources*. Web. Sept. 2010. <http://www.sans.org/>.

[19] *Damn Vulnerable Linux*. Web. Sept. 2010. <http://www.damnvulnerablelinux.org/>.

X: Appendices

10.1 Pre-Questionnaire

Login:

Pre-Study Questionnaire

Demographic Information:

1) What year are you?

Freshman Sophomore Junior Senior Grad other

2) Circle all classes you have taken.

Object Oriented Design

Systems Programming Concepts

Machine Org. & Assem. Lang.

OS

Distributed/Networks/Computer architecture

Soft Eng/OOAD

Databases

Algorithms

Foundations/Theory/Languages

AI

Graphics/Animation

Soft. Security Eng.

3) How much security background do you have?

None A Little Some A Lot

4) Please list security experiences/classes/jobs.

5) How interested are you in security?

Not Interested Somewhat Interested Interested Very Interested

6) Do you plan to pursue opportunities to learn about security?

No Maybe Yes

<u>Technical Questions:</u>

1) If you are trying to gain unauthorized access over a network to a remote system, which of the following would you do first?

a. Identify open ports and running processes on the remote system

b. Launch a denial of service attack to try and disrupt the system

c. Search online for vulnerabilities in their running processes

d. Use a buffer overflow attack to gain access

2) What are two different useful tools for scanning remote systems?

_____ _____

3) Circle all the things you CANNOT learn when scanning a remote system.

a. The passwords of some of the users

b. If the system is "alive," or running and connected to the network

c. Which ports are open on the target system

d. What operating system is running on the system

4) Name two different types of vulnerabilities that can be created by programming mistakes.

_____ _____

5) Which of the following are typical effects of a buffer overflow? Circle ALL that apply.

a. Segmentation fault

b. Computer crash

c. Executing arbitrary code

d. Over-writing memory

e. Shutdown ports

6) What can typically be accomplished using format string parameters? Circle ALL that apply.

a. Read from memory

b. Write to memory

c. Shutdown ports

d. Computer crash

7) What is a good indication that a program might contain a format string vulnerability?

a. Text that the user entered is printed verbatim to the screen

b. The user can input a very large string

c. The program takes several command line arguments

d. The program responds differently based on the user's input

10.2 Post-Questionnaire for Experimental Group

Login:

Post-Study Questionnaire

Technical Questions:

1) If you are trying to gain unauthorized access over a network to a remote system, which of the following would you do first?

a. Identify open ports and running processes on the remote system

b. Launch a denial of service attack to try and disrupt the system

c. Search online for vulnerabilities in their running processes

d. Use a buffer overflow attack to gain access

2) What are two different useful tools for scanning remote systems?

_____ _____

3) Circle all the things you CANNOT learn when scanning a remote system.

a. The passwords of some of the users

b. If the system is "alive," or running and connected to the network

c. Which ports are open on the target system

d. What operating system is running on the system

4) Name two different types of vulnerabilities that can be created by programming mistakes.

_____ _____

5) Which of the following are typical effects of a buffer overflow? Circle ALL that apply.

a. Segmentation fault

b. Computer crash

c. Executing arbitrary code

d. Over-writing memory

e. Shutdown ports

6) What can typically be accomplished using format string parameters? Circle ALL that apply.

a. Read from memory

b. Write to memory

c. Shutdown ports

d. Computer crash

7) What is a good indication that a program might contain a format string vulnerability?

a. Text that the user entered is printed verbatim to the screen

b. The user can input a very large string

c. The program takes several command line arguments

d. The program responds differently based on the user's input

Ratings:

1) Compare learning from a self-contained, game atmosphere to learning from a book or paper. The game atmosphere is:

Much Worse Somewhat Worse About the Same Somewhat Better Much Better

2) How much do you feel you learned?

Not at all A Little Some Very Much

3) How interested are you in playing more CounterMeasures or more of a similar type of security game?

Not Interested Somewhat Disinterested Neutral Somewhat Interested Interested

4) How interested are you in security after playing?

Not Interested Somewhat Interested Interested Very Interested

5) Do you plan to pursue opportunities to learn about security after playing?

No Maybe Yes

6) How much did you enjoy CounterMeasures?

Not at all A Little Some A Lot

7) What would you improve/change about CounterMeasures?

8) What were the most useful features/parts of CounterMeasures?

10.3 Post-Questionnaire for Control Group

Login:

Post-Study Questionnaire

Technical Questions:

1) If you are trying to gain unauthorized access over a network to a remote system, which of the following would you do first?

a. Identify open ports and running processes on the remote system

b. Launch a denial of service attack to try and disrupt the system

c. Search online for vulnerabilities in their running processes

d. Use a buffer overflow attack to gain access

2) What are two different useful tools for scanning remote systems?

_____ _____

3) Circle all the things you CANNOT learn when scanning a remote system.

a. The passwords of some of the users

b. If the system is "alive," or running and connected to the network

c. Which ports are open on the target system

d. What operating system is running on the system

4) Name two different types of vulnerabilities that can be created by programming mistakes.

_____ _____

5) Which of the following are typical effects of a buffer overflow? Circle ALL that apply.

a. Segmentation fault

b. Computer crash

c. Executing arbitrary code

d. Over-writing memory

e. Shutdown ports

6) What can typically be accomplished using format string parameters? Circle ALL that apply.

a. Read from memory

b. Write to memory

c. Shutdown ports

d. Computer crash

7) What is a good indication that a program might contain a format string vulnerability?

a. Text that the user entered is printed verbatim to the screen

b. The user can input a very large string

c. The program takes several command line arguments

d. The program responds differently based on the user's input

Ratings:

1) Compare learning from a self-contained, game atmosphere to learning from a book or paper. The game atmosphere is:

Much Worse Somewhat Worse About the Same Somewhat Better Much Better

2) How much do you feel you learned?

Not at all A Little Some Very Much

3) How interested are you in playing CounterMeasures or a game about security?

Not Interested Somewhat Disinterested Neutral Somewhat Interested Interested

4) How interested are you in security after playing?

Not Interested Somewhat Interested Interested Very Interested

5) Do you plan to pursue opportunities to learn about security after playing?

No Maybe Yes

10.4 Instructions provided to Control Group

Instructions:

First, take some time to read the information you have been given. This will teach you some useful security techniques. What you want to specifically focus on is scanning, buffer overflows, and string format vulnerabilities. When you are done (reading should take ~15 to 20 minutes) you can use the shell you have been given to work on the following mission:

```
This mission is designed to test what you have learned so far.  On the
machine 10.9.0.3, there is a program that allows an administrator to remotely
update passwords for users and add new users to a password file.  Your goal
is to get the password for the admin user. The only additional thing you
should need to know is that you can use "telnet 10.9.0.3 port#" to connect to
a specific port on the specified IP. Good luck! When you find the password,
let one of us know.
```

10.5 Collected Data

Test Scores:

Experimental Group Pre- and Post-Test Scores

User	Pre-Study							Score (Out of 14)	Post-Study							Score (Out of 14)
	Q1	Q2	Q3	Q4	Q5	Q6	Q7		Q1	Q2	Q3	Q4	Q5	Q6	Q7	
u1	2	0	2	0	2	1	0	7	2	2	2	2	2	1	0	11
u2	2	0	2	1	1	1	0	7	2	2	2	2	1	1	2	12
u3	2	0	2	0	2	1	0	7	2	2	2	1	1	1	2	11
u4	2	0	2	0	1	2	2	9	2	2	2	2	1	1	2	12
u5	2	0	0	0	1	2	0	5	2	2	2	2	2	2	2	14
u6	0	0	2	0	1	1	2	6	0	0	2	1	1	1	2	7
u7	0	0	1	0	0	0	0	1	2	1	2	0	1	0	2	8
u8	0	0	2	0	1	2	0	5	2	2	0	1	2	1	0	8
u9	2	0	2	0	1	1	2	8	2	2	2	2	1	1	2	12
u10	2	2	2	1	2	1	0	10	2	2	2	2	2	1	0	11

Control Group Pre- and Post-Test Scores

User	Pre-Study							Score (Out of 14)	Post-Study							Score (Out of 14)
	Q1	Q2	Q3	Q4	Q5	Q6	Q7		Q1	Q2	Q3	Q4	Q5	Q6	Q7	
t1	0	0	2	2	1	2	2	9	2	0	2	2	1	1	2	10
t2	0	0	2	2	2	2	1	9	2	2	2	2	2	2	1	13
t3	2	2	2	2	1	2	0	11	2	2	2	2	2	1	0	11
t4	2	1	0	2	1	1	1	8	2	2	0	2	1	1	2	10
t5	0	0	0	0	1	1	2	4	2	2	2	2	1	1	1	11
t6	0	0	2	0	1	1	0	4	-	-	-	-	-	-	-	-
t7	0	0	0	0	1	1	0	2	2	1	2	2	1	1	0	9
t8	2	0	2	2	1	0	0	7	2	1	0	2	2	2	0	9
t9	0	0	2	0	1	2	1	6	2	2	2	1	1	2	1	11
t10	0	0	2	0	2	2	0	6	2	2	2	2	1	2	1	12

Time to Complete Experiment:

Experimental Group Completion Times

User	Times (Minutes)					Notes
	Overall	Mission 1	Mission 2	Mission 3	Mission 4	
u1	59	16	8	4	31	
u2	27	5	2	3	17	
u3	39	6	4	8	21	
u4	22	14	3	1	4	
u5	21	7	3	3	8	
u6	11	7	1	1	2	
u7	7	4	3	-	-	Didn't Finish
u8	63	6	6	14	37	
u9	47	12	6	6	23	
u10	29	17	1	7	4	

Control Group Completion Times

User	Times (Minutes)	Notes
	Overall	
t1	52	
t2	38	
t3	70	
t4	40	
t5	83	Didn't Finish
t6	-	Didn't Start
t7	95	
t8	82	
t9	86	
t10	81	

User Opinions/Ratings:

Participant Interest In Security Before Experiment

	Control	Experimental
Not Interested	0	0
Somewhat Interested	4	2
Interested	4	6
Very Interested	2	2

Participant Interest in Security after Experiment

	Control	Experimental
Not Interested	0	0
Somewhat Interested	2	1
Interested	5	3
Very Interested	2	6

Participants Plans on Pursuing Learning Opportunities About Security Before the Experiment

	Control	Experimental
No	0	0
Maybe	7	5
Yes	3	5

Participants Plans on Pursuing Learning Opportunities About Security After the Experiment

	Control	Experimental
No	0	0
Maybe	4	4
Yes	5	6

Participants Opinions on Learning about Security from a Game compared to a Book

	Control	Experimental
Much Worse	0	0
Somewhat Worse	0	0
About the Same	0	1
Somewhat Better	6	3
Much Better	2	6

How Much Participants Felt they Learned about Security during the Experiment

	Control	Experimental
Not at all	0	0
A Little	1	1
Some	7	4
Very Much	1	5

Participant Interest in Playing/Playing More CounterMeasures

	Control	Experimental
Not Interested	0	0
Somewhat Disinterested	0	0
Neutral	1	1
Somewhat Interested	2	3
Interested	6	6

Experimental Groups Rating of Enjoyment of Playing CounterMeasures

Not at all	0
A Little	0
Some	5
A Lot	5

Demographics:

Control

Grade		Classes Taken		Security Background	
Freshman	1	Object Oriented Design	9	None	2
Sophomore	5	Systems Programming Concepts	10	A Little	7
Junior	1	Machine Org + Assembly Lang.	7	Some	1
Senior	3	Operating Systems	8	A Lot	0
Grad	0	Distributed Systems/Networking/Computer Arch.	3		
Other	0	Soft Eng/OOAD	4		
		Databases	2		
		Algorithms	6		
		Foundations/Theory/Languages	3		
		AI	0		
		Graphics/Animation	2		
		Soft. Security Eng.	2		

Experimental

Grade		Classes Taken		Security Background	
Freshman	2	Object Oriented Design	10	None	7
Sophomore	5	Systems Programming Concepts	10	A Little	3
Junior	3	Machine Org + Assembly Lang.	8	Some	0
Senior	0	Operating Systems	7	A Lot	0
Grad	0	Distributed Systems/Networking/Computer Arch.	1		
Other	0	Soft Eng/OOAD	1		
		Databases	3		
		Algorithms	4		
		Foundations/Theory/Languages	1		
		AI	0		
		Graphics/Animation	0		
		Soft. Security Eng.	0		