A SECURE WEB BASED DATA COLLECTION AND DISTRIBUTION SYSTEM
FOR GLOBAL POSITIONING SYSTEM RESEARCH

A thesis presented to

the faculty of the

Fritz J. and Dolores H. Russ
College of Engineering and Technology
of
Ohio University

In partial fulfillment

of the requirements for the degree

Master of Science

Derek J. Bleyle

August 2004
This thesis entitled

A SECURE WEB BASED DATA COLLECTION AND DISTRIBUTION SYSTEM
FOR GLOBAL POSITIONING SYSTEM RESEARCH

BY

DEREK J. BLEYLE

has been approved

for the School of Electrical Engineering and Computer Science

and the Russ College of Engineering and Technology

Constantinos Vassiliadis
Associate Professor of Electrical Engineering

R. Dennis Irwin
Dean, Fritz J. and Dolores H. Russ
College of Engineering and Technology
ABSTRACT

BLEYLE, DEREK J. M.S. August 2004. Electrical Engineering and Computer Science

A Secure Web Based Data Collection and Distribution System for Global Positioning System Research. (111pp.)

Director of Thesis: Constantinos Vassiliadis

A secure data collection and distribution system has been created for Global Positioning System (GPS) research. Specifically, a program, RxData, has been created, using Borland C++, to produce a database from binary GPS data sent from the LAAS Ground Facility (LGF) data logger at the Ohio University airport. This data is sent through a one-way null modem cable which serves as a hardware firewall. In addition, several programs have been purchased, configured, and implemented to allow access to the binary data via a secure Internet connection. The secure connection has been established using 128-bit encryption between the FTP server, Serv-U FTP v5.0, and the FTP client, FTP Voyager v11.0, and all connections with the server are sent through a software firewall, Sygate Personal Firewall Pro v5.5, which protects the server and database from Internet related attacks.

Approved:

Constantinos Vassiliadis

Associate Professor of Electrical Engineering
DEDICATION

As I come to the conclusion of my second Master’s degree, there are a number of individuals that I’d like to dedicate this work to. It is an honor to do so because it has been the help, support, encouragement, and friendship from each of them that has allowed me to arrive at this point.

I thanked a number of people in my first Master’s thesis entitled *The Performance of Drag Reducing Agents Under Non-Ideal Conditions*. Many of these individuals I wish to thank again as they continue to stand by my side as the years pass by and therefore I dedicate this work to them. These folks include my parents John and Genie Kemp, my sisters and their families, Tara, Manuel, and Christian Pirela, and Meredith, Bruce, and Gabriela Chen, my grandparents, Eugene and Bette Kirkland, all of my cousins, aunts, and uncles, my best friends, Dan Campana and Jon Wilkins, and numerous other friends that have encouraged me, both directly and indirectly, to keep striving to perform my best even in the face of adversity and very hard times.

In addition, I would like to dedicate this work to the thousands of souls that were lost on September 11, 2001, as well as all the men and women in uniform that have dedicated their lives to defending the very freedoms which came under attack on that fateful day. For it is the very freedoms they defend that has allowed me to pursue the education of my choice. I am forever changed due to the events of 9-11 and am more proud than ever to be an American. Thank you to all the men and women in uniform for the countless sacrifices they make on a daily basis. You are not forgotten.
ACKNOWLEDGMENTS

The first person I wish to thank is Professor Chuck Alexander. Dr. Alexander’s persuasion and encouragement to obtain a Master’s degree in Electrical Engineering was a pivotal role in my decision to stay in school at Ohio University. His belief in me, even when others failed to see my potential, was essential to my completion of this degree. Dr. Alexander became the dean of engineering at Cleveland State University (CSU) at the end of my first year of the program and I wish him the best.

Professor Constantinos Vassiliadis must be commended for becoming my advisor when Dr. Alexander moved on to CSU and for being there during some particularly hard times during my three years as an EE student. His support was vital to my staying in the program and I sincerely thank him for helping me obtain work with Professor Frank van Graas.

I cannot thank Dr. van Graas enough, not only for the financial support, but for allowing me to work with him and for giving me the freedom that enabled me to perform my best. His patience and support were critical to all that was accomplished while working on this project. His leadership was inspirational and I learned a great deal from him.

I also owe a great deal of thanks to both Sanjeev Gunawardena and Jonathon Sayre for the instrumental roles they played in my learning Borland C++ and for their generosity in sharing work that allowed me to get a solid footing and complete my program and project.

Finally, I have an incredible amount of respect and admiration for all of the individuals that I had the privilege of working with during my tenure as the president of the Graduate Student Senate. The success of that group would not have been possible without each of you and I thank you all for the lessons I will carry with me throughout my life.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>3</td>
</tr>
<tr>
<td>DEDICATION</td>
<td>4</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>5</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>7</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>8</td>
</tr>
<tr>
<td>ABBREVIATIONS AND ACRONYMS</td>
<td>10</td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>11</td>
</tr>
<tr>
<td>1.1 Statement of Purpose</td>
<td>11</td>
</tr>
<tr>
<td>1.2 Background</td>
<td>11</td>
</tr>
<tr>
<td>1.3 Project Scope</td>
<td>13</td>
</tr>
<tr>
<td>1.4 Other Considerations</td>
<td>17</td>
</tr>
<tr>
<td>1.5 Thesis Outline</td>
<td>18</td>
</tr>
<tr>
<td>2 LITERATURE REVIEW</td>
<td>19</td>
</tr>
<tr>
<td>2.1 Secure Networks</td>
<td>19</td>
</tr>
<tr>
<td>2.1.1 Security Considerations</td>
<td>20</td>
</tr>
<tr>
<td>2.1.2 Security Implementations</td>
<td>27</td>
</tr>
<tr>
<td>3 SYSTEM SETUP</td>
<td>36</td>
</tr>
<tr>
<td>3.1 GPS Satellites</td>
<td>37</td>
</tr>
<tr>
<td>3.2 GPS Antennas and Receivers</td>
<td>38</td>
</tr>
<tr>
<td>3.3 LGF Data Logger</td>
<td>39</td>
</tr>
<tr>
<td>3.4 Hardware Firewall</td>
<td>39</td>
</tr>
<tr>
<td>3.5 FTP Server and Database</td>
<td>40</td>
</tr>
<tr>
<td>3.5.1 Database</td>
<td>40</td>
</tr>
<tr>
<td>3.5.2 FTP Server</td>
<td>75</td>
</tr>
<tr>
<td>3.5.3 Software Firewall</td>
<td>82</td>
</tr>
<tr>
<td>3.6 FTP Client</td>
<td>89</td>
</tr>
<tr>
<td>3.7 FTP Server-Client Interaction</td>
<td>91</td>
</tr>
<tr>
<td>4 RESULTS AND DISCUSSION</td>
<td>97</td>
</tr>
<tr>
<td>5 CONCLUSIONS</td>
<td>104</td>
</tr>
<tr>
<td>5.1 System Overview</td>
<td>104</td>
</tr>
<tr>
<td>5.2 Future Work/Considerations</td>
<td>105</td>
</tr>
<tr>
<td>6 REFERENCES</td>
<td>109</td>
</tr>
<tr>
<td>7 BIBLIOGRAPHY</td>
<td>110</td>
</tr>
<tr>
<td>7.1 Sources Used to Learn About Networks</td>
<td>110</td>
</tr>
<tr>
<td>7.2 Sources Used to Learn C and C++</td>
<td>111</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 3.1 Overview of RxData program components. .......................................................... 51
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Old system of GPS data collection and distribution</td>
<td>15</td>
</tr>
<tr>
<td>1.2</td>
<td>Overview of secure web based data collection system</td>
<td>16</td>
</tr>
<tr>
<td>2.1</td>
<td>Creating a digital signature</td>
<td>30</td>
</tr>
<tr>
<td>2.2</td>
<td>How SSL establishes a secure connection</td>
<td>33</td>
</tr>
<tr>
<td>3.1</td>
<td>RxData at program startup</td>
<td>42</td>
</tr>
<tr>
<td>3.2</td>
<td>RxData port configuration module</td>
<td>43</td>
</tr>
<tr>
<td>3.3</td>
<td>RxData port configuration module showing various baud rates</td>
<td>44</td>
</tr>
<tr>
<td>3.4</td>
<td>Warning message shown if disconnecting from active serial port</td>
<td>44</td>
</tr>
<tr>
<td>3.5</td>
<td>RxData indications that data is being received</td>
<td>45</td>
</tr>
<tr>
<td>3.6</td>
<td>Beginning stages of animation</td>
<td>47</td>
</tr>
<tr>
<td>3.7</td>
<td>Middle stages of animation</td>
<td>47</td>
</tr>
<tr>
<td>3.8</td>
<td>Final stages of animation</td>
<td>47</td>
</tr>
<tr>
<td>3.9</td>
<td>RxData after data transmission has stopped</td>
<td>48</td>
</tr>
<tr>
<td>3.10</td>
<td>RxData upon COM port disconnect</td>
<td>48</td>
</tr>
<tr>
<td>3.11</td>
<td>RxData program information</td>
<td>49</td>
</tr>
<tr>
<td>3.12</td>
<td>Overview of Program RxData</td>
<td>52</td>
</tr>
<tr>
<td>3.13</td>
<td>Flow chart of class TMain in Program RxData</td>
<td>53</td>
</tr>
<tr>
<td>3.14</td>
<td>Flow chart of class TCommConfig in program RxData</td>
<td>55</td>
</tr>
<tr>
<td>3.15</td>
<td>Function OpenPort in class TSerialVdr of program RxData</td>
<td>57</td>
</tr>
<tr>
<td>3.16</td>
<td>COM port setup in function OpenPort of class TSerialVdr</td>
<td>58</td>
</tr>
<tr>
<td>3.17</td>
<td>StartStatusThread function in class TSerialVdr of program RxData</td>
<td>59</td>
</tr>
<tr>
<td>3.18</td>
<td>StatusThreadFunc function in class TSerialVdr of program RxData</td>
<td>61</td>
</tr>
<tr>
<td>3.19</td>
<td>Catch(ECommError) function in class TSerialVdr of program RxData</td>
<td>62</td>
</tr>
<tr>
<td>3.20</td>
<td>ProcessStatusEvent function in class TSerialVdr of program RxData</td>
<td>63</td>
</tr>
<tr>
<td>3.21</td>
<td>Read_COM_port function in class TSerialVdr of program RxData</td>
<td>65</td>
</tr>
<tr>
<td>3.22</td>
<td>ClosePort function in class TSerialVdr of program RxData</td>
<td>67</td>
</tr>
<tr>
<td>3.23</td>
<td>Class TWriteData in program RxData</td>
<td>68</td>
</tr>
<tr>
<td>3.24</td>
<td>Parse_Data function of class TParseData in program RxData</td>
<td>70</td>
</tr>
<tr>
<td>3.25</td>
<td>Process_Serial_Data function in class TParseData of program RxData</td>
<td>71</td>
</tr>
<tr>
<td>3.26</td>
<td>Decode_Message function in class TParseData of program RxData</td>
<td>72</td>
</tr>
<tr>
<td>3.27</td>
<td>Class TAbout of program RxData</td>
<td>74</td>
</tr>
<tr>
<td>3.28</td>
<td>Serv-U FTP Server Program Details</td>
<td>75</td>
</tr>
<tr>
<td>3.29</td>
<td>Domain settings of Serv-U FTP Server</td>
<td>77</td>
</tr>
<tr>
<td>3.30</td>
<td>Session Log in Serv-U FTP server</td>
<td>77</td>
</tr>
<tr>
<td>3.31</td>
<td>User generated SSL certificate in Serv-U FTP Server</td>
<td>78</td>
</tr>
<tr>
<td>3.32</td>
<td>Setting account properties in Serv-U FTP Server</td>
<td>79</td>
</tr>
<tr>
<td>3.33</td>
<td>Setting Serv-U OTP to hash function MD5</td>
<td>80</td>
</tr>
<tr>
<td>3.34</td>
<td>Assigning access controls to user accounts in Serv-U FTP server</td>
<td>81</td>
</tr>
<tr>
<td>3.35</td>
<td>Sygate Personal Firewall Pro details</td>
<td>82</td>
</tr>
<tr>
<td>3.36</td>
<td>Monitoring network activity on Sygate Personal Firewall Pro</td>
<td>83</td>
</tr>
</tbody>
</table>
### ABBREVIATIONS AND ACRONYMS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK</td>
<td>Acknowledgment response to TCP SYN request</td>
</tr>
<tr>
<td>ARPA</td>
<td>Advanced Research Projects Agency</td>
</tr>
<tr>
<td>bps</td>
<td>Bits per second</td>
</tr>
<tr>
<td>Bps</td>
<td>Bytes per second</td>
</tr>
<tr>
<td>C/A</td>
<td>Coarse Acquisition</td>
</tr>
<tr>
<td>COMSEC</td>
<td>Communications Security</td>
</tr>
<tr>
<td>COMPUSEC</td>
<td>Computer Security</td>
</tr>
<tr>
<td>DLL</td>
<td>Dynamic Library Link</td>
</tr>
<tr>
<td>DoD</td>
<td>Department of Defense</td>
</tr>
<tr>
<td>DSA</td>
<td>Digital Signature Algorithm</td>
</tr>
<tr>
<td>ECC</td>
<td>Elliptic Curve Cryptosystems</td>
</tr>
<tr>
<td>FAA</td>
<td>Federal Aviation Administration</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>FTPS</td>
<td>Secure File Transfer Protocol – also SFTP</td>
</tr>
<tr>
<td>GB</td>
<td>Gigabyte</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>Gr</td>
<td>Ground</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HTML</td>
<td>Hyper Text Markup Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hyper Text Transfer Protocol</td>
</tr>
<tr>
<td>ICMP</td>
<td>Internet Control Message Protocol</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>INFOSEC</td>
<td>Information Security</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>ISO</td>
<td>International Organization for Standardization</td>
</tr>
<tr>
<td>kb</td>
<td>Kilobit</td>
</tr>
<tr>
<td>kB</td>
<td>Kilobyte</td>
</tr>
<tr>
<td>LAAS</td>
<td>Local Area Augmentation System</td>
</tr>
<tr>
<td>LGF</td>
<td>LAAS Ground Facility</td>
</tr>
<tr>
<td>NFS</td>
<td>Network File System</td>
</tr>
<tr>
<td>MAC</td>
<td>Media Access Control</td>
</tr>
<tr>
<td>MB</td>
<td>Megabyte</td>
</tr>
<tr>
<td>MHz</td>
<td>Megahertz (1 X 10^6 Hz)</td>
</tr>
<tr>
<td>Net</td>
<td>Short for Internet</td>
</tr>
<tr>
<td>NETSEC</td>
<td>Network Security</td>
</tr>
<tr>
<td>NIC</td>
<td>Network Interface Card</td>
</tr>
<tr>
<td>O/S</td>
<td>Operating System</td>
</tr>
<tr>
<td>OSI</td>
<td>Open Systems Interconnection</td>
</tr>
<tr>
<td>OTP</td>
<td>One Time Passwords</td>
</tr>
<tr>
<td>P2P</td>
<td>Peer-to-Peer</td>
</tr>
<tr>
<td>PKI</td>
<td>Public Key Infrastructure</td>
</tr>
<tr>
<td>PRC</td>
<td>Pseudo Random Code</td>
</tr>
<tr>
<td>Rx</td>
<td>Receive</td>
</tr>
<tr>
<td>RPC</td>
<td>Remote Procedure Call</td>
</tr>
<tr>
<td>SFTP</td>
<td>Secure File Transfer Protocol – also FTPS</td>
</tr>
<tr>
<td>SMTP</td>
<td>Simple Mail Transfer Protocol</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Sockets Layer</td>
</tr>
<tr>
<td>SYN</td>
<td>TCP synchronization request</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>TLS</td>
<td>Transport Layer Security</td>
</tr>
<tr>
<td>Tx</td>
<td>Transmit</td>
</tr>
<tr>
<td>UART</td>
<td>Universal Asynchronous Receiver Transmitter</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>URL</td>
<td>Universal Resource Locator</td>
</tr>
<tr>
<td>VPN</td>
<td>Virtual Private Network</td>
</tr>
<tr>
<td>VCL</td>
<td>Visual Component Library</td>
</tr>
<tr>
<td>WWW</td>
<td>World Wide Web</td>
</tr>
</tbody>
</table>
1 INTRODUCTION

1.1 Statement of Purpose

The objective of this project was to, as the title suggests, set up a secure network in which Global Positioning System (GPS) data could be accessed and downloaded over the Internet for analysis by authorized individuals. These authorized individuals include those directly involved in the project, which was sponsored by the Federal Aviation Administration (FAA), as well as those whom are linked to the project within the FAA.

1.2 Background

The grandfather of today’s Internet was known as ARPAnet. ARPAnet was formed in the late 1960’s through the funding of the Advanced Research Projects Agency (ARPA) of the Department of Defense (DoD). ARPAnet originated through the networking of the main computer systems of about a dozen ARPA-funded universities and research institutions across the country. These computer systems were connected with communications operating at a then-stunning 56kbps with the goal of allowing multiple users to send and receive information over the same communication paths at the same time. Packet switching techniques and protocols such as Transmission Control Protocol (TCP) and Internet Protocol (IP) were developed to enable ARPAnet to work. These techniques are still in use today. [Deitel, 2001]

Internet usage was initially limited to universities and research institutions. Then the military began to use it and eventually, the government decided to allow access to the Internet for commercial purposes. The result of commercializing the Internet has been an
explosion in the bandwidth available on the Internet and a plummeting of costs associated with all aspects of the Internet. [Deitel, 2001]

The increase in bandwidth and the decrease in costs of the Internet, as well as the advent of the World Wide Web in 1990, have allowed hundreds of millions of computers around the world to become part of the Internet. This, in turn, has allowed unprecedented access to data at the touch of a keyboard and the click of a mouse. This ease of access, coupled with high speed internet access, has fueled the enormous growth of the Internet over the past decade. As both the number of legitimate uses of the Internet and the number of legitimate users on the Internet has grown, so to have the pitfalls and security risks associated with such use. Security risks are prevalent due to the increase in the number of computer crackers, whom are commonly referred to as, improperly, computer hackers. While the intentions of a cracker are generally malicious, the intentions of a hacker are to gain understanding and improve products and programs. However, this distinction is rarely made and computer users with malicious intent are commonly referred to as hackers. Thus, this thesis will use the term hacker for a computer user with malicious intent.

There are several motives for a hacker to attack a computer system. A hacker’s motivation identifies the purpose of the attempted intrusion and includes: [Maiwald, 2001]

- Challenge
  - most common motivation
  - any Internet connected system is a potential target
- Greed
  - oldest motivation
  - sites that have something of value are primary targets
- Malicious Intent
also called vandalism

o purpose is to deny usage to legitimate users or change the message of the site

o tend to be focused on particular targets

In addition, hackers’ intentions can be broken down into two general areas:

- Those affecting web sites
- Those affecting the data in transit

Examples of the former include:

- Breaking into a web site and manipulating the site in some way
- Breaking into a web site and gaining access to stored critical data, such as users’ private information (e.g. IDs, passwords, credit card numbers, etc.)
- Crashing a web site by overwhelming it with requests

Examples of the latter include:

- Gaining access to data during transmission through the use of packet sniffers and retrieving personal or other important information (e.g. usernames, passwords, etc.)
- Gaining access to data during transmission through some other method

Thus, the need for secure processes while conducting any form of business on the Internet is of the utmost importance so that vital information is inaccessible to unauthorized individuals. Securing web sites is also vitally important so that data is not accessed improperly and in order to prevent manipulation of web sites or web site crashes.

1.3 Project Scope

The goal of this project was to create a secure web based data collection and distribution system for GPS research. The reason such a system was necessary was so data could be monitored, as well as downloaded and processed from remote locations. The
current system for collecting and distributing GPS data is shown in Figure 1.1. This system requires an individual to be on site to transfer data from the LGF (LAAS [Local Area Augmentation System] Ground Facility) data logger to another computer for copying to compact disks. Once these disks are made, they can be packaged and distributed so the data can be analyzed by authorized individuals. This system of data retrieval is inefficient, from both a financial and a time perspective, as compact disks must be created and mailed to individuals for processing. Thus, the new data collection and distribution system (see Figure 1.2) will allow instant access to data and the ability to download the data securely at any location around the globe, so long as the end user is hooked up to the Internet. Once this data is downloaded, it can be processed at the client’s location. So, again, the major benefit of such a system is a time savings. However, a financial savings and a more simplistic way of obtaining the data are seen as well.

Figure 1.2 shows an overview of the system that was created for this purpose. There have been multiple layers of security used in order to implement a secure system and minimize the risk associated with having this system online. The components employed to create this secure data collection and distribution system are as follows:

- Hardware firewall
- Program to capture data and create a database
- Secure-FTP server using 128-bit SSL encryption
- Software firewall
- Secure-FTP client using 128-bit SSL encryption

Each of these will be discussed in more detail later under the System Setup section (Chapter 3) of this work.
Figure 1.1 Old system of GPS data collection and distribution
Figure 1.2 Overview of secure web based data collection system
Thus, the primary considerations in this project, which can more easily be seen by referring to Figure 1.2, were to construct a hardware firewall, to design a software program that would collect binary data through an available serial port, process the data, and produce a database, to select and configure a secure FTP server, to choose and configure a software firewall, and to select a compatible, secure FTP client for use on the end user’s machine. The GPS satellites, GPS antennas and receivers, and the LGF data logger were already in place prior to the start of this project, as a comparison between Figure 1.1 and Figure 1.2 will reveal. The main consideration with respect to the data was to capture and decode the data sent from the LGF data logger and make it available in a database to an FTP server so it could be downloaded. This was all done through a program, RxData, which was developed for this purpose. RxData will be discussed in detail in Chapter 3 as it took the majority of time to create, test, and implement this component of the system.

1.4 Other Considerations

Most of the components used to create this system were purchased “off the shelf” and were configured and implemented according to the needs of this project. The components that were purchased for use in this project include the following:

- Serv-U Version 5.0 FTP Server
- Sygate Personal Firewall Pro Version 5.5
- FTP Voyager Version 11.0 FTP Client
- Borland C++ Builder 6.0 (used to create and deploy RxData)
1.5 Thesis Outline

The goal of writing this thesis is to convey both what has been done and how it has been accomplished. This will be done using a traditional thesis writing format with various sections that convey varying levels of information regarding this project. This section, the Introduction, presents an overview of the project. The Literature Review provides background on the security issues of connecting a computer or network to the Internet and how those issues are resolved. The System Setup section describes how the various components were put together to create a working system. It walks the reader through the emergence of the data from GPS satellites, through the various stages of processing, to the target location of the client computer via the Internet. This section is the heart of this thesis and explains each component in great detail and how that component fits into the overall picture. Results and Discussion discusses the results of the work and the demands placed on the computer which serves as both the server and the database, while the Conclusions section gives both an overview of what has been accomplished and offers suggestions for future system improvements. Finally, the References section lists the sources used in the writing of this thesis, while the Bibliography lists those sources used to learn about networks and programming in C++.
2 LITERATURE REVIEW

2.1 Secure Networks

As was discussed in Chapter 1, today’s Internet emerged from a project funded by the DoD known as ARPAnet. ARPAnet introduced the first protocols that made communication over the Internet possible. These networking protocols, primarily TCP and IP, allowed communication to occur between computers within a network and between computers located in different networks. It wasn’t long after communications over the Internet became more common that it was realized that a secure method of communicating was going to be necessary. The reasons such secure communications are necessary and the current methods used to establish secure connections and secure a communication source (web site server, client computer, company network, etc.) are discussed in the next section.

There are a number of ways of producing a secure network. While each method offers a varying degree of protection, when used collectively, the security offered is more than sufficient for most purposes. There are two key components to producing a secure network: the security of the computer and data on the computer, and the security of data in transit between computers.

To secure a computer and its data while connected to the Internet there are both hardware and software firewalls that offer varying levels of protection. They can be configured to the needs of the user and, as such, can either be very secure or offer little protection from attack. Up to date anti-virus software is also important in protecting computers and the information on those computers. When these programs are used in
conjunction with other security measures, they are instrumental in finding malicious code on a computer and preventing the spread of such code.

To protect information in transit, the data must be encrypted. There are a number of ways that data can be encrypted and sent safely over the Internet. These include Virtual Private Networks (VPNs) and Secure Sockets Layer (SSL) encryption. In addition, employing digital signatures solves the problems of authentication and integrity, while using digital certificates and certificate authorities solves the problem of authenticating the parties in a transaction. These techniques are referred to as Public Key Infrastructure (PKI) and the methods used in this project are discussed in the section which follows.

2.1.1 Security Considerations

The only sure way a computer is safe from the attacks of the Internet is to avoid the Internet all together and remain offline. However, this is hardly pragmatic since some degree of business must occur through the Internet in today’s high tech world. Therefore, the best course of action is to protect one’s self as much as possible in order to prevent and/or discourage successful attacks from occurring. One of the best ways to do this is to learn about the various types of threats and how they can best be prevented.

Perhaps the most important contribution to network security, at least from a gateway perspective, is the advent of the firewall. The firewall is analogous to the moat and drawbridge of medieval times. The firewall’s purpose is to ensure that only authorized connections enter or leave a computer, just as the drawbridge was lowered over the moat to allow authorized individuals to enter or leave a castle. Firewalls emerged in the early 1990s as a “countermeasure to the surge in malicious Internet activity.” They performed very well early on and still offer a degree of protection; however, changes in technology are quickly
rendering the firewall less effective. Threats to the protection ensured by a firewall include
the growth of active content—ActiveX and JavaScript in messaging software and Web
browsers—in client software, greater user mobility—wireless connections, and peer-to-peer
technology—coupled with “spyware” and “backdoors” sometimes included in P2P
applications. [Arbaugh, 2003] Thus, by avoiding as many of these technological advances as
is practically possible, a firewall will be more effective at performing its job. For this
project, the vast majority of the technologies that make a firewall less effective at protecting
from an attack will not be used. Therefore, the firewall should be highly effective in
defending the database/server computer from attacks.

There are a multitude of attacks that can be launched on a computer by an attacker. While each
of these attacks occurs in a different way, the purpose of the attack is essentially
the same—to either gain access to the computer being attacked or hinder the performance of
(or even stop—“crash”) that computer.

While it would be impossible to cover every possible type of attack that can occur
(since new ways of attacking a computer emerge almost daily), the following list of attacks
and the manner in which they occur is given to present the reader with the variety of methods
an attacker can use. In turn, this offers some idea of the possible susceptibility of a computer
on the Internet. The following attacks and descriptions have been taken from Kizza, 2002:

- **IP-spoofing**: replaces IP address of the source element in the data packets with a bogus
  address

- **SYN flooding**: utilizes the breakdown of trust relationship by flooding a server with
  SYN packets by bogus addresses to open ports on server. SYN packets are TCP
  synchronization packets that request an acknowledgment (ACK) from the computer they
  are sent to. If enough SYN packets are sent, the server will crash.
• **Sequence numbers attacks**: allows attacker to create a one-way TCP connection with a target element while spoofing another target element by guessing the TCP sequence used.

• **Scanning & probing attacks**: intruders send large quantity of packets from a single location. Involves a Trojan horse remote controlled program with a distributed scanning engine that is configured to scan carefully selected ports.

• **Low bandwidth attacks**: sending of low volume, intermittent series of scanning or probing packets from various locations.

• **Session attacks**: target and break into sessions that are already in progress. Sessions are communication interactions between client and server machines.
  
  o **Packet sniffing**: program on a network element connected to a network to passively receive all Data Link Layer frames passing through the device’s network interface. Sniffed frames can have their contents, message, and header altered, modified, or even deleted and replaced.
  
  o **Buffer overflow**: allows intruder to overrun one or more program variables enabling the intruder to execute arbitrary codes with the privilege of the current user
  
  o **Session hijacking**: intruder sniffs legitimate source IP addresses and hijacks the TCP session without knowledge of the server or client using the stored user ID and access URL on the server machine.

• **DDoS**: (Distributed Denial-of-Service)—nuisance attacks (interrupt services)
  
  o **Ping of death**: ICMP ECHO_REQUESTS are called pings. When the packets sent by the client to the server are large and not fragmented into smaller packets, they can cause havoc.
- **Smurfing**: sends large amount of spoofed ping packets containing victim’s IP address as the source address.

- **Teardrop attack**: attacks the reassembling mechanism of the fragmented IP packets.

- **Land c attack**: Intruder sends TCP SYN packet giving target host’s address as both source and destination addresses.

Any system connected to the Internet and using TCP and UDP services like WWW, Mail, FTP, and Telnet is potentially subjected to DDoS attacks. While there are no meaningful or effective solutions for DDoS attacks, it should be noted that these types of attacks are generally no more than nuisance attacks and do not cause permanent damage to the system (although they may crash the system). [Kizza, 2002]

Each of the attacks described in the list above can be assigned into one of four categories of attack:

- **Access**: an attempt to gain information that the attacker is unauthorized to see. This occurs either where information resides or in transit.

- **Modification**: an attempt to modify information that an attacker is not authorized to modify. This also occurs either where the information resides or while it is in transit and consists of changes to the data, insertions into the data, or deletion of the data.

- **Denial-of-Service**: deny the use of resources to legitimate users of the system, information, or capabilities. This is nothing more than vandalism and can include information, applications, systems, or communications.

- **Repudiation**: an attack against the accountability of the information. Examples include masquerading or denying an event.
As can be seen from the prior lists, there are a large number of ways that a computer on the Internet can be attacked. The reason the Internet makes connected machines so vulnerable to attacks is that it is based on an open architecture system. While this is a strength of the Internet, it is also its greatest weakness, as all computers use essentially the same methods to communicate over the net.

The Internet is based on the Open Systems Interconnection (OSI) standard which was developed in 1984 by the International Organization for Standardization (ISO). The OSI Reference Model is the heart of this standard and is a set of seven layers that define the different stages that data must undergo to travel from one device to another over the network. The seven layers of this model can be broken down into two sets as follows: [Tyson, 2000]

- **Application Set:**
  - **Layer 7: Application** – interacts with the O/S or application whenever files are transferred, messages are read, or other network activities are performed
  - **Layer 6: Presentation** – converts the data from the Application layer into a standard format that other layers can understand.
  - **Layer 5: Session** – establishes, maintains, and ends communication with the receiving device.

- **Transport Set:**
  - **Layer 4: Transport** – also called the host-to-host layer as it maintains the flow control of data and provides error checking and recovery of data between the devices
  - **Layer 3: Network** – uses logical protocols, routing, and addressing to determine the way data will be sent to the recipient device
Layer 2: Data – defines the type of network and the packet sequencing, and assigns the appropriate physical protocol to the data

Layer 1: Physical – the hardware layer that defines the physical characteristics of the network (connections, voltage levels, timing, etc.)

The OSI Reference Model only provides a guideline for computer-to-computer interactions and layers are often combined into a single layer as is done in TCP/IP protocol stack. This stack uses five layers that combine layers of the OSI model as follows: [Tyson, 2000]

- Layer 5: Application – combines the Application, Presentation, and Session layers. Protocols for specific functions reside at this level and include email (SMTP), file transfer (FTP), virtual terminal (TELNET), HTTP, and DNS.

- Layer 4: Transport – corresponds to the Transport layer. This is where another device on the network is asked if it is willing to accept information from the local device. It does this via the TCP or UDP.

- Layer 3: Internet – corresponds to the Network layer. Uses the IP address to determine the address of the device it is communicating with. This layer allows for communication between different networks. It does this via the IP.

- Layer 2: Network Interface – corresponds to the data layer. Routes data between devices (end systems) on the same network and manages the exchange of data with the network and other devices.

- Layer 1: Physical – corresponds to the Physical layer. Concerned with specifying the characteristics of the transmission medium, the nature of the signals, the data rate, etc.

The reason it is important to understand the open architecture of the Internet is because this very “openness” is what makes any computer on the network susceptible to
attack. The types of attacks described earlier target different layers of the OSI model architecture and make use of the way interactions occur over the network (through protocols).

While there are a large number of ways of being attacked while connected to the Internet, there are also ways of minimizing both the number of attacks and the severity of the attacks. There are several approaches to this: [Kizza, 2002]

- Security policy
- Risk management
- Firewalls
- Cryptology
- Authentication and authorization
- Legislation
- Regulation
- Self-regulation
- Education

A security policy sets standards for user ID’s and passwords. These include a standard for the user ID such as length and content, a minimum password length, maximum and minimum password ages, and password content requirements. A security policy also addresses system administration procedures such as software upgrades and updates, performing vulnerability scans, reviewing logs, and performing regular network monitoring. [Maiwald, 2001]

Management of risk involves determining where network vulnerabilities are located and whether those vulnerabilities can be tolerated or not. Risk management also reviews physical vulnerabilities such as server (or other computer) locations and inside threats. [Maiwald, 2001]

By running a software firewall on a computer that is running a server, it helps to ensure that the environment is secure. Since a firewall is a network access control device that is designed to deny all traffic except that which is explicitly allowed, the firewall can block unauthorized login access, file transfer access, and remote command execution. In addition,
it can even deny services such as Rlogin, Telnet, SMTP, NFS, and other RPC services. [Goncalves, 2000, Maiwald, 2001] “Using a firewall is probably the single most effective and easiest way to add security to a small network.” [Deitel, 2001]

Cryptography and authentication and authorization lie under the category of trust. Perhaps the single most important aspect of secure exchanges over the Internet is trust. If a machine cannot be trusted, then there is no way to determine the intent of that machine, and therefore, communicating with that machine can pose a security risk. Thus, a trust relationship must be established through a trust infrastructure.

Legislation, regulation, self-regulation, and education all lie under the category of information. Each of these involves putting together information regarding the dos and don’ts of network usage. The aim of each of these is to educate users of the network about its vulnerabilities and how one can protect oneself, as well as user’s responsibilities when using the network and liabilities for certain types of network activities.

2.1.2 Security Implementations

Information security (INFOSEC) is comprised of communication security (COMSEC), computer security (COMPUSEC), and network security (NETSEC). COMSEC is the security of information in transit. COMPUSEC is the control of access to computer systems, and NETSEC is the security of computer networks. The important thing to understand is that there is no all-encompassing solution to security. In other words, there is no “one size fits all” security solution. A security program consists of: [Maiwald, 2001]

- **Anti-virus software** to protect against and destroy malicious programs
- **Access controls** to protect files on computers
- **Firewalls** which are border security products between internal and external networks and control access of networks, and
- **Encryption** to protect data in transit.

To build a secure web based data collection system for GPS research, several components were put in place to protect the system from the Internet, as well as protecting information in transit between the FTP server and the FTP client. The software firewall is the first line of defense in protecting the system from malicious attacks. Ports have been limited on the server computer to only those needed by the FTP server. A hardware firewall was put in place between the LGF data logger and the database and server computer. This “firewall” allows data to flow in one direction and, therefore, even if the software firewall is breached, the most important component of the data collection system, the LGF data logger, is protected.

To protect information in transit, a trust infrastructure needed to be established between the FTP server and the FTP client. The fundamental goals of establishing a trust infrastructure, which in turn allows for a successful, secure transaction, are:

- **Authentication**: Users must be able to assure themselves that they are communicating with a real entity
- **Confidentiality**: Sensitive Internet communications and transactions must be kept private
- **Data integrity**: Communications must be protected from undetectable alteration by third parties in transmission on the Internet.
- **Non-repudiation**: It should not be possible for a sender to reasonably claim that he or she did not send a secured communication. [verisign.com, 2003; Deitel et al, 2001]
The authentication issue is a question of how the sender and receiver of a message prove their identities to one another. A digital signature is a method of authenticating electronic information by using encryption. This type of protection uses an encryption method known as public-key cryptography. Public-key cryptography is an asymmetric technology that uses two inversely-related keys, known as a key pair—a public key and a private key. The public key is freely distributed and is published with information as to who the owner is. The private key is kept secret by the owner of the key pair. When authenticating, the owner of the key pair encrypts the information with the private key. Only the correct published public key can correctly decrypt the information. Thus, only the owner of the key pair could have sent the information and authentication is provided. The defining property of the secure public-key algorithm is that it is computationally infeasible to deduce the private keys from the public key. This algorithm was created in 1976 by Whitfield Diffie and Martin Hellman and is, aptly, referred to as the Diffie-Hellman algorithm. Other public key algorithms exist as well. These include RSA, Elgamal, Digital Signature Algorithm (DSA), and Elliptic Curve Cryptosystems (ECC). [Deitel, 2001, Maiwald, 2001]

A digital signature takes public key cryptography one step further and protects the information from modification after it has been received and decrypted. A digital signature is created as displayed in Figure 2.1. The digital signature is the electronic equivalent of a written signature and serves to authenticate the sender’s identity, as previously indicated. It is also, as is a written signature, difficult to forge. A digital signature is created by first taking plaintext information and passing it through a hash function. The hash function is a mathematical calculation that creates a checksum of the information, known as a hash value, or message digest. The message digest is then encrypted by the sender’s private key which
creates a digital signature and authenticates the sender. The encrypted message digest is sent, along with the information that has been encrypted by the receiver’s public key, to the receiver of the information. The receiver uses the sender’s public key to decipher the original digital signature and reveal the message digest. Next the original message is deciphered using the receiver’s private key. Finally, the receiver applies the same hash function as that used by the sender to the original message. If the message digest matches the one included in the signature then the information has not changed. [Maiwald, 2001, Deitel, 2001]
The confidentiality, or privacy, issue is a question of how you ensure that information transmitted over the Internet is not captured or passed on to a third party without your knowledge. Public key cryptography is used as it was for authentication. However, the keys used to encrypt and decrypt messages are reversed. Thus, encryption is performed with the receiver’s public key since only the receiver’s private key will be able to decrypt the message. This ensures confidentiality of the data.

The integrity issue is a question of how you ensure the information you send or receive has not been altered or compromised. The integrity of data is also determined through the use of a digital signature as was described for the authentication process. Since the digital signature method requires the hash values, or message digests, to match up on both the sender and receiver ends for the message to be unaltered, this method provides for a means to check the data integrity. If the message digest is the same on the sender and receiver end, the data has not been altered and there is data integrity.

The non-repudiation issue is a question of how you legally prove that a message was sent or received. In other words, it is a question of accountability. Digital signatures do not provide proof that a message has been sent. However, once a document has been digitally signed, it can be timestamped. Timestamping binds a time and date to a digital document via a third party called a timestamping agency. The timestamping agency receives the encrypted, digitally signed message and affixes the time and date of receipt to the message. It digitally signs the whole package with the timestamping agency’s private key and cannot be altered by anyone except the timestamping agency. If the sender does not report its private key to be compromised before the document is timestamped, the sender cannot legally prove the
document was signed by a third party. The sender can also require the receiver to digitally sign and timestamp the message as proof of receipt. [Deitel, 2001]

2.1.2.1 Secure Sockets Layer

SSL was originally developed by Netscape in 1994 at the same time that the original web browser, the Netscape Navigator was launched. In 1996, Netscape turned over SSL version 3.0 to the Internet Engineering Task Force (IETF). The IETF has “officially” renamed SSL to TLS (Transport Layer Security) and is seeking wider adoption of the TLS protocol and approach.

The SSL protocol runs above the TCP/IP protocol stack and below higher-level protocols located within the application layer (FTP, HTTP, Telnet, SMTP, etc). It uses TCP/IP on behalf of the higher-level protocols, and in the process allows an SSL-enabled server to authenticate itself to an SSL-enabled client, allows the client to authenticate itself to the server, and allows both machines to establish an encrypted connection. [Netscape.com, 2004]

Figure 2.2 describes how an SSL connection is established between a client and a server machine. This interaction between the client and server is referred to as an SSL handshake. This figure was created using a combination of sources—Schultz, 2001, Verisign², 2003, and netscape.com. The steps involved to establish the SSL connection have been taken directly from netscape.com and are as follows:

1. The client sends the server the client's SSL version number, cipher settings, randomly generated data, and other information the server needs to communicate with the client using SSL.
2. The server sends the client the server's SSL version number, cipher settings, randomly generated data, and other information the client needs to communicate with the server over SSL. The server also sends its own certificate and, if the client is requesting a server resource that requires client authentication, requests the client's certificate.

3. The client uses some of the information sent by the server to authenticate the server. If the server cannot be authenticated, the user is warned of the problem and informed
that an encrypted and authenticated connection cannot be established. If the server can be successfully authenticated, the client goes on to Step 4.

4. Using all data generated in the handshake so far, the client (with the cooperation of the server, depending on the cipher being used) creates the **premaster secret** for the session, encrypts it with the server's public key, and sends the encrypted premaster secret to the server.

5. If the server has requested client authentication, the client also signs another piece of data that is unique to this handshake and known by both the client and server. In this case the client sends both the signed data and the client's own certificate to the server along with the encrypted premaster secret.

6. If the server has requested client authentication, the server attempts to authenticate the client. If the client cannot be authenticated, the session is terminated. If the client can be successfully authenticated, the server uses its private key to decrypt the premaster secret, then performs a series of steps (which the client also performs, starting from the same premaster secret) to generate the **master secret**.

7. Both the client and the server use the master secret to generate the **session keys**, which are symmetric keys used to encrypt and decrypt information exchanged during the SSL session and to verify its integrity—that is, to detect any changes in the data between the time it was sent and the time it is received over the SSL connection.

8. The client sends a message to the server informing it that future messages from the client will be encrypted with the session key. It then sends a separate (encrypted) message indicating that the client portion of the handshake is finished.
9. The server sends a message to the client informing it that future messages from the server will be encrypted with the session key. It then sends a separate (encrypted) message indicating that the server portion of the handshake is finished.

10. The SSL handshake is now complete, and the SSL session has begun. The client and the server use the session keys to encrypt and decrypt the data they send to each other and to validate its integrity.

The ciphers discussed in the steps above can be any number of ciphers that are in use today. These include DES, DSA, KEA, MD5, RSA, and Triple-DES, among others. The server used in this project uses a 128-bit cipher which is far more secure than the 40-bit ciphers which are still in use today.
3 SYSTEM SETUP

The system consists of a number of components that are tied together to form the secure web based data collection and distribution system. Each of these components plays a critical role in how the data is collected, decoded, stored, transmitted, and processed. Please refer to Figure 1.2 throughout this section for a schematic of the system.

There are currently 29 GPS satellites that circumnavigate the earth. Each of these satellites transmits data via a carrier signal and that data is collected at the Ohio University airport through specially developed GPS receivers. The data obtained on these receivers is sent to the LAAS (Local Area Augmentation System) Ground Facility (LGF) data logger where the data is processed. Once processed, the data is sent through a hardware firewall. This firewall is a two-wire null modem cable that consists of only a transmit (Tx)-receive (Rx) wire and a ground (Gr) wire. After passing through the hardware firewall, the data enters the computer which serves as both the database and the FTP server.

The computer housing the FTP server and database is a Dell Inspirion 4550 with a 2.53 GHz processor, 1 GB 333 MHz DDR ram, a 120 GB hard drive, and a 10/100 Bit Ethernet LAN port. This computer runs the Windows XP O/S and contains the following programs:

- a data receiving program—RxData—that was written in Borland C++ Builder v6.0
- a secure FTP server—Serv-U v5.0
- a software firewall—Sygate Personal Firewall Professional v5.5

The majority of time and effort was spent developing the software program RxData. All other programs were purchased on the market and were setup and installed on the server.
The FTP server is connected to Ohio University’s Local Area Network through an RJ-45 Ethernet cable, which, in turn, is connected to the Internet. On the other end of this system are the client computers that are required to run some version of a secure FTP client that will interact with the Serv-U SFTP Server. FTP Voyager v11.0 is a secure FTP client that uses 128-bit encryption to ensure secure transfers and is offered by the same company that produced the Serv-U SFTP Server. It has been tested and is the recommended software for use with this system. It is, however, limited to computers that are running the Windows O/S. Therefore, if a client is running another O/S, it will need to run a product that is capable of interacting with the Serv-U software. In other words, it must be capable of secure 128-bit encrypted transfers.

3.1 GPS Satellites

The GPS is a worldwide radio-navigation system formed from a constellation of satellites and their ground stations. The first GPS satellites were launched in 1973 by the Department of Defense (DoD). At the time of this writing, a total of 27 healthy satellites are in one of 6 orbits that hover 11,000 miles above the Earth’s surface and, collectively, cover the entire face of the globe at any given time. A total of 29 satellites are currently in orbit; however, the number of healthy satellites—those that can be used for collecting GPS data—varies from day to day. Because of their distance from the Earth and the spacing between each satellite, a minimum of five satellites are in view from every point on the globe and allow the system to function accurately. The satellites travel at a velocity of 4 km/s and transmit a total of 2 carrier signals. One is an L-1 carrier signal that is transmitted at 1575.42 MHz, while the other is an L-2 carrier signal that is transmitted at 1227.6 MHz. The L-1 carrier signal modulates a Coarse Acquisition (C/A) Pseudo-Random Code (PRC) which is
transmitted at 1.023 MHz and a navigation message that is transmitted at 50 bps. These signals contain information on both distance (measured by the speed of L-1 signals between 3 satellites and a receiver) and time (by taking a measurement from a fourth satellite since GPS satellites contain atomic clocks) and are picked up by GPS receivers. The PRC is used to place the GPS receivers in sync with the GPS satellites, which in turn, allows the system to accurately indicate where someone or something is located on the planet. The navigation message contains information on where the satellites are located within their orbits and in relation to ground stations. Errors in measurements are minimized using all three signals that are sent—the carrier-phase, the code-phase, and the navigation message. Finally, the L-2 carrier signal that was mentioned is used primarily for military purposes, but is also used by high-end GPS receivers to account for errors and improve GPS accuracy. [trimble.com, ngs.noaa.gov]

3.2 GPS Antennas and Receivers

There are a total of four GPS antennas located at the Ohio University airport that gather data from the GPS satellites. Each antenna is actually a system of two antennas—a vertically-oriented linear array composed of stacked dipoles for tracking low-elevation satellites (5-30 degrees) and a high zenith (High Z) antenna for tracking high-elevation satellites (30-90 degrees)—and, as such, there are eight dedicated receiver subsystems that collect data from the antennas and send it to the LGF data logger. There are anywhere from 7 to 12 satellites located above the horizon (and thus available to gather data from) at any given time that data are collected from. These multiple measurements are made over the same signal frequency using different channels on the receiver. Since each GPS satellite has a unique PRC, the receiver can determine which satellite is sending that signal and will
funnel it to the appropriate channel of the receiver. Hence, a single receiver operating over a single frequency can pick up and distinguish signals from multiple satellites.

**3.3 LGF Data Logger**

The data logger is a Ziatech dedicated data processing computer with the following features:

- 166 MHz processor
- 10 serial ports
- Attached 1-GB Jaz drive (used in original system setup)
- Industrial components that make this machine extremely durable and reliable

Each of the serial ports is connected to a GPS receiver and thus, data is received through these ports. Once this data enters the computer it is processed and sent through the hardware firewall to the computer that servers as the FTP server and database.

**3.4 Hardware Firewall**

The connection between the LGF Data Logger and the FTP server and database is established using a 2-wire null modem cable. This connection allows for a Tx-Rx connection and a Gr-Gr connection. Consequently, data *can* be transmitted from the data logger to the FTP server and database, but *cannot* be transmitted from the FTP server and database to the data logger. As such, this connection serves as a hardware firewall and will prevent any malicious activity from reaching the data logger. This will prevent any attack on the data logger and therefore, prevent a crash on that machine caused by such an attack. This is very important, as the data logger is the heart of the data collection system and any crash of the data logger would cause the loss of crucial data. Thus, the system can withstand a crash in
the FTP server and database with the primary outcome being a disruption in online service. This is primarily a nuisance since no critical data will be lost (since the data logger stores data for a certain amount of time) and the FTP server and database will only need to be reset to get it back online.

3.5 FTP Server and Database

The computer running the FTP server also hosts RxData (the program which creates the database), the database itself, and a software firewall to protect the computer from Internet attacks. This computer is a Dell Dimension 4550 with the following features:

- Intel Pentium 4-2.53 GHz microprocessor
- 4x AGP
- Intel 845PE system chip set
- Floppy drive
- 1 GB DDR 333-MHz SDRAM memory
- DVD +R/RW drive
- 120 GB ATA-100 Ultra DMA hard drive
- DVD ROM drive
- RJ-45 NIC connector port
- 1 parallel port
- 6-2.0 compliant USB connectors
- 1 serial port
- Microsoft Windows XP Operating System (O/S)
- 100-MB Zip drive
- Integrated Intel 10/100 Ethernet Network Interface Card (NIC)

3.5.1 Database

The database portion of this project was created through a program dubbed RxData. It has been named as such because the primary purpose of the program is to capture, or receive (hence the “Rx” portion of the name), data (hence the “Data” portion of the name) as it enters the computer via the serial port. This program was created entirely in house using Borland C++ Builder Version 6.0. Through the use of Borland’s Integrated Development
Environment (IDE) and its Visual Component Library (VCL), RxData Version 1.0 was created. It should be noted that a large amount of help was received in getting started with this project from both Sanjeev Gunawardena and Jonathon Sayre. Sanjeev was instrumental in providing an example of communicating with serial ports using C++ in a Windows environment, while Jonathon was crucial in communicating his knowledge of C++ when problems were encountered. These gentlemen cannot be thanked enough for the knowledge they provided that was instrumental in getting this program started.

When RxData is started, it automatically sets the serial port parameters and connects to the serial port. Thus, the program is ‘listening’ for data once the program starts. The screen that is seen upon startup is shown in Figure 3.1. When RxData starts up, a lot of information is presented to the user. The first thing the program does is present the user with instructions on program use. Next, information is given regarding the status of the system. The program lets the user know:

- If a connection to the serial port was successfully created
- If the database directory already existed or whether it needed to be created (and if so, whether it was created successfully or not)
- If the file that will receive data was successfully created
- The current serial port properties
- If the serial port UART buffers were successfully purged
In addition, the program lets the user know:

- The date and time the program was opened
- The number of the day of the year (used for GPS data filename)
- The system status
- The time the serial port was connected
- The name of the file and where it is located on the computer

All of this information is presented to the user so he/she has this information available and is able to understand what is happening with the program ‘behind the scenes.’

If the serial port parameters need to be changed, ‘Port Setup’ is clicked on the main module (Figure 3.1) and the RxData serial port configuration module appears (Figure 3.2). The COM port and the baud rate are the serial port parameters that can be changed by the user. While the number of data bits, parity, and number of stop bits are presented in the port
configuration module, they are for informational purposes only and their values cannot be changed. The ports available to the user through this program are COM ports one through four, but only those available on the local machine will be displayed. A variety of baud rates are available to the user (see Figure 3.3) and range from 300 bps to 230400 bps. This program will be connected to the LGF data logger at 115200 bps (which is the highest transfer rate of the 16550C UART). The most important consideration when connecting two serial ports is that the baud rate on each machine should be set at the same value.

If RxData has established a serial port connection, and the user clicks ‘Rx Port Disconnect’, a warning message is shown (see Figure 3.4). This message is presented to prevent the user from accidentally disconnecting from the serial port while the program is operating, and, therefore, is done to prevent data loss. If the user clicks yes, the program will disconnect from the serial port. If the user clicks no, the program will remain connected to
Figure 3.3 RxData port configuration module showing various baud rates

Figure 3.4 Warning message shown if disconnecting from active serial port
the serial port and will continue to receive and process data (which it continues to do even when the warning message is displayed).

Once data is received on the serial port, RxData gives several indications to the user that data is being received and processed. As soon as data is received, the program displays the message “Bytes are being received and processed” as seen in Figure 3.5. In addition the

![RxData indications that data is being received](image)

program indicates the following:

- The time the data transfer started
- The amount of time that has elapsed since data started being received (from first byte received to last byte received)
- The message “Bytes have been received and are being processed” is displayed in the lower portion of the program
• The total bytes read from the COM port is displayed
• The total raw data bytes written to a temporary file is displayed
• The total decoded data bytes written to disk is displayed
• The number of messages that have been decoded is displayed

Concurrently, as long as data is being received, the program offers the user visual cues to indicate that data is being received and processed. These cues can be seen in Figure 3.6, Figure 3.7, and Figure 3.8 and are:

• Animated dots following the “Bytes are being received and processed” message
• Animated arrows pointing to the total bytes read from the COM port and the total bytes written to both temporary and permanent files
• Incrementing of elapsed time, total bytes received, and messages decoded

In the event that data is no longer being received over the serial port (i.e. all data has been transmitted by the LGF data logger), the program animation would stop and the program values would stop being updated. The program would appear as it does in Figure 3.9. The primary things to note are that the elapsed time stops once data transfer ceases and the total bytes read from the COM port, the raw data bytes written, the decoded data bytes written, and the number of messages decoded stop updating and retain their last updated values (but not their final values). This is particularly important to note as the program would seem to indicate that the bytes read from the COM port, the raw bytes written, and the decoded data bytes written are all different from one another. This wouldn’t make sense and could only be possible if data was being lost after it was received on the COM port. This, in turn, would indicate that there was either a problem with the program, or that the data was corrupted and was being discarded by the program.
Figure 3.6 Beginning stages of animation

Figure 3.7 Middle stages of animation

Figure 3.8 Final stages of animation
Once data transfer has ceased and “Rx Port Disconnect” has been selected, several aspects of the program are updated. This is shown in Figure 3.10. The program indicates...
that the COM port was successfully closed and the serial UART buffers have been purged. The system status is updated and indicates that the port is disconnected. The final elapsed time is shown and is the same as indicated in Figure 3.9. Again, this occurs because the elapsed time ceases to increment once data is no longer being received. The program also:

- Displays the calculated data transfer rate
- Updates the bytes read from the COM port, the raw bytes written, and the decoded data bytes written and offers the visual cue that the values are the final values. Note that these three values are now equivalent (as compared to Figure 3.9 where they differed from one another) which indicates that the program is operating correctly and that no data was corrupted and/or lost during transmission.
- Displays the final value of the number of messages decoded

Finally, if “Help” is clicked, the program information is displayed as shown in Figure 3.11.

![Figure 3.11 RxData program information](image)
3.5.1.1 RxData Program Flowchart

Since the Windows O/S is a proprietary operating system and since Windows is intended to cover all aspects of computer usage, the O/S program is not only complex, but information on how the software controls computer hardware is difficult to find. A lot of information exists on how to control various aspects of the serial ports, but the majority of the information was written in Microsoft Visual Basic, and, therefore, was of little use in writing this program. In addition, information that was found and was applicable to C or C++ programming was generally written for controlling devices or transmitting data between systems where the hardware in one system ‘talks’ to the hardware in the other system (handshaking, etc.). Most of this information had limited applicability to this program since the system being created has a one-way serial connection between the receiving (FTP server and database) computer and the sending (LGF data logger) computer.

The flow charts which follow illustrate not only how RxData works and interacts with the serial port to collect data and produce a database, but also the complexity of such interactions, primarily due to working with the Windows XP O/S.

Figure 3.12 displays an overview of RxData. The program is composed of 6 classes- TMain, TCommConfig, TSerialVdr, TWriteData, TParseData, and TAbout-and a header file, Defines. An overview of the classes and header file and their purpose within the program is given in Table 3.1.
<table>
<thead>
<tr>
<th>Component</th>
<th>Function of Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMain</td>
<td>Imports Defines.h information into program. Interacts with all other classes. Initializes program and variables and creates program interface.</td>
</tr>
<tr>
<td>TCommConfig</td>
<td>Creates form to set serial port parameters. Interacts with TMain and TSerialVdr classes.</td>
</tr>
<tr>
<td>TSerialVdr</td>
<td>Deals with all aspects of the serial port. Interacts with TMain and TWriteData classes.</td>
</tr>
<tr>
<td>TWriteData</td>
<td>Deals with all aspects of creating files and writing data to hard drive. Interacts with TSerialVdr and TParseData classes.</td>
</tr>
<tr>
<td>TParseData</td>
<td>Parses and decodes data received from serial port. Interacts only with class TWriteData.</td>
</tr>
<tr>
<td>TAbout</td>
<td>Creates interface and displays information about RxData. Interacts only with class TMain.</td>
</tr>
<tr>
<td>Defines</td>
<td>Defines universal program parameter values and is imported into class TMain.</td>
</tr>
</tbody>
</table>

Table 3.1 Overview of RxData program components.

By viewing Figure 3.13, it can be seen that once program RxData is started, class TMain initializes the serial port and creates the Graphical User Interface (GUI) shown in Figure 3.1 which the user interacts with. At this point, the program waits for input from the user until one of four options is selected. These options are:

- Port Setup
- Help
- Rx Port Connect/Rx Port Disconnect
- Close ("X")

If “Port Setup” is selected, class TCommConfig is called and TMain is left. If "Rx Port Connect” is selected, function OpenPort is called in class TSerialVdr and TMain is left. If “Rx Port Disconnect” is selected, the user is presented with a choice as to whether to really disconnect or not. If “yes” is selected, function ClosePort is called in class TSerialVdr and TMain is left. If “no” is selected, the program returns to waiting for user input. If “Help” is selected, class TAbout is called and TMain is left. Finally, if the user chooses to close the
Figure 3.12 Overview of Program RxData
Figure 3.13 Flow chart of class TMain in Program RxData
program and “X” is selected, a series of checks is performed by the program and either closes or returns to waiting for input from the user, depending on the choice made by the user as displayed in Figure 3.13.

Once the user clicks “Port Setup” from the main program interface, the program enters class TCommConfig (Figure 3.14). This class immediately creates a form which displays a GUI with options for changing the serial port parameters. When the form is created, the current serial port parameters are shown as the currently selected items in the drop down boxes. This is shown in Figure 3.2.

The first scenario in class TCommConfig is if the user makes no changes to the parameters. If this is the case, the user can select either the “OK” or the “Cancel” button since none of the other buttons are active. The ultimate end is that the form is closed and the user returns to class TMain which is the main program interface. This can be seen in Figure 3.14.

If any of the values in the form are changed, then all the buttons become active and the user has a choice between five buttons. The user can cancel the changes, which will close the GUI, return the program to class TMain, and return the user to the main program interface.

If the user selects the “Set as Default” button, the currently selected parameters are written to an initialization file and will be used whenever the program reconnects to the serial port. The values are not immediately applied to the serial port unless the “Apply” button is selected.

If the “Restore Defaults” button is selected, the program will replace the currently selected parameter values with those from the initialization file.
Figure 3.14 Flow chart of class TCommConfig in program RxData
When the “Apply” button is selected, the program goes through a series of checks as indicated in Figure 3.14 and either disconnects from the serial port (if necessary) by calling function ClosePort in class TSerialVdr and applies the changes to the port by calling function SetPortProperties in class TSerialVdr, or makes no changes to the currently active port and exits back to the main program interface (back to class TMain).

If the “OK” button is selected, the program goes through a series of checks and will either exit back to the main program interface without applying any changes (if any were made) or will apply the changes to the deactivated serial port and exit back to the main program interface (from class TCommConfig back to class TMain). All of this can be seen in Figure 3.14.

When RxData connects to the serial port either upon program startup or through a user directive, a complex series of interactions between RxData, Windows, and the computer hardware ensues. This is illustrated in Figure 3.15 and Figure 3.16. Class TMain calls function OpenPort in class TSerialVdr. This entire process is transparent to the user unless an error occurs while attempting to set the serial port parameters. The user remains on the main program interface while this happens. The main things to note are the number of steps that are necessary to change the default serial port parameters that are set by Windows with the user-defined port parameters set by RxData, and the number of times the program must check for errors to ensure that no problems were encountered by Windows when setting or retrieving the port parameters.

Figure 3.17 shows what happens when the OpenPort function calls the StartStatusThread function. A thread is started in Windows to monitor the status of the serial port and then the StatusThreadFunc is called. Upon returning, if no errors were found, the function returns to the OpenPort function of class TSerialVdr.
Figure 3.15 Function OpenPort in class TSerialVdr of program RxData
From TSerialVdr::OpenPort

Get original COM state and save settings

Error? Yes Error handling
       No Throw ECommError

Set COM state with user-defined settings

Error? Yes Error handling
       No Throw ECommError

Set COM port buffer sizes

Error? Yes Error handling
       No Throw ECommError

Get original COM time-outs and save settings

Error? Yes Error handling
       No Throw ECommError

Set COM time-outs with user-defined settings

Error? Yes Error handling
       No Throw ECommError

COM port open

Purge COM port buffers

Go to TSerialVdr::OpenPort

Figure 3.16 COM port setup in function OpenPort of class TSerialVdr
Figure 3.17 StartStatusThread function in class TSerialVdr of program RxData
When the StatusThreadFunc function is called, a complex series of events occur in order for the program to monitor the serial port for errors or characters which have been received (Figure 3.18). A status loop is entered which continually checks for either of these events. If an error is thrown, function Catch(ECommError) is called. If a character is received, function ProcessStatusEvent is called. When this status loop is broken because an error has occurred, the function is left and returns back to the StartStatusThread function.

When an error is thrown because a problem has occurred with a Windows function, the error is caught by the function Catch(ECommError) as indicated in Figure 3.19. A message is displayed to the user to indicate the cause of the problem or the location within the program where the error occurred and the handle to the event which monitors the O/S status is closed. This Windows error, in turn, causes the program to stop functioning properly and will either cause RxData to crash, or the user must restart the program (or both). Most of these errors occur during the debugging phase of building the program and, therefore, any fatal Windows errors encountered should be very infrequent while using this program.

When a character is received in the serial port UART InQueue buffer, the ProcessStatusEvent function is called in class TSerialVdr. Once the character is received, a timer is started by the program to determine the elapsed time and, ultimately, the data transfer rate (see Figure 3.10). The elapsed time is displayed on the main program interface (see Figure 3.5) and is updated every time this function is called and a character has been received. This is displayed in Figure 3.20. Once the elapsed time is updated, the program checks the message time stamp to determine if it has changed from the previous message.
Figure 3.18 StatusThreadFunc function in class TSerialVdr of program RxData
Figure 3.19 Catch(ECommError) function in class TSerialVdr of program RxData
Figure 3.20 ProcessStatusEvent function in class TSerialVdr of program RxData
time stamp. If it has, the program closes the currently open files and calls function OpenFile in class TWriteData to create new files with updated names that coincide with the new message timestamp. Upon returning from function OpenFile, or if no file name update was necessary, the function Read_COM_Port is called in class TSerialVdr. If a line status error has been received in function ProcessStatusEvent, an error is thrown and the program returns to function StatusThreadFunc.

Once it has been indicated by the function ProcessStatusEvent that a character has been received, function Read_COM_Port is called. This function is the heart of class TSerialVdr in terms of processing the data that is received through the serial port. As such, this function will be covered in detail. Figure 3.21 shows an overview of the Read_COM_Port function. The first thing that occurs is that an event is created in Windows in order to read from the COM port. Assuming no errors occur when creating this event, the program checks to see if there are bytes left in the serial port UART InQueue buffer. If bytes remain, COM errors are checked for and if found, the error is either cleared and a message displayed or an error is thrown which terminates the function. If no COM errors have occurred, RxData checks the Windows function WaitingOnRead. If Windows is waiting, this indicates that the result is overlapped. An overlapped result simply indicates that the process is asynchronous. Since this has been accounted for in the program, Windows functions are called to obtain the overlapped results. However, if Windows is not waiting to read data from the serial port, the data is simply moved to a buffer created by Windows to store the data until it is removed by RxData for processing. Once the data is obtained, the number of bytes read is displayed in the main program interface and the function Parse_Data is called in class TParseData in order to process the data. It should be pointed out that if data is unable to
Figure 3.21 Read_COM_port function in class TSerialVdr of program RxData
be moved to the Windows buffer (indicating an error) and Windows is unable to determine why, RxData will encounter a critical error and will automatically be terminated by Windows. As noted previously, the occurrence of such an error should be extremely infrequent, if it occurs at all.

When a call is made by either another function or by the user to close the serial port, the function ClosePort in class TSerialVdr is accessed. This function (see Figure 3.22) first checks to ensure that the port is not already closed. If it is still open, a series of Windows checks are made and then the original serial port settings (before RxData was started) are restored. Once the original settings are restored, the port is closed, the buffers are purged, all open files are closed, and final results are displayed on the main program interface. In addition, the calculated transfer rate is displayed as well. If any errors were generated during this process, they are sent to function Catch(ECommError).

Once the OpenPort function is called in class TSerialVdr, it calls class TWriteData. This class handles all the opening of files and all of the writing of both data and diagnostic files as seen in Figure 3.23. One thing that needs to be noted regarding this flowchart is that when the function is called from OpenPort, two files are created, a temporary file for raw data and a permanent file for decoded data. These temporary files are kept for two weeks and are automatically deleted by the program after this point. They are created to ensure that if any decoded data is corrupted, there will be a backup copy stored for a reasonable amount of time that can be retrieved if necessary. They are also created to deal with any unforeseen events that may occur that may lead to a loss of or corruption of the decoded data. Another thing to note is that function Parse_Data accesses TWriteData to write both the raw data and the decoded data to the hard drive. Finally, both function Process_Serial_Data and
Figure 3.22 ClosePort function in class TSerialVdr of program RxData
Figure 3.23 Class TWriteData in program RxData
Decode_Message in class TParseData access class TWriteData to create and write to diagnostics files.

When function Read_COM_Port calls function Parse_Data in class TParseData, the data is copied to another buffer and class TWriteData is accessed to write the raw data. If bytes remain in the buffer, function Process_Serial_Data is called. If no bytes remain in the buffer, the program returns to Read_COM_Port. Upon returning from function Process_Serial_Data, the program checks whether the data has been decoded. If it has been, class TWriteData is accessed to write the decoded data. This can all be seen in Figure 3.24.

Figure 3.25 indicates what occurs when function Process_Serial_Data is called. This function ensures that the buffer has not overflowed, checks to see if any messages remain in the buffer, determines whether there are enough bytes in the buffer for an entire message, decides whether a message header has been found, and then verifies if enough bytes exist in the buffer for a full message (based on information found in the message header). If these conditions have been met, function Decode_Message is called. If any of these conditions have not been met, various steps are taken as indicated in Figure 3.25. Upon returning from Decode_Message, the function examines whether the message was decoded or not. If it has been, the message is removed from the buffer and the process starts again. If the message was not decoded, the message header is removed and the process begins anew. When no more messages exist in the buffer, the program returns to the Parse_Data function.

Once Decode_Message is called, a checksum if found in the message and compared with a calculated checksum that is based on the message length. If the two values are equivalent, the message has been decoded (see Figure 3.26) and the message is moved to another buffer which is sent to class TWriteData to be written to the hard drive. The number
Figure 3.24 Parse_Data function of class TParseData in program RxData
Figure 3.25 Process_Serial_Data function in class TParseData of program RxData
From TParseData::
Process_Serial_Data

Determine checksum_rcvd

Calculate checksum_calc

checksum_rcvd = checksum_calc?

No → Message NOT decoded

Yes → Move data from Process_buffer to OutputBuffer

Buffer overflow?

Yes → Reset buffer

To TWriteData

No → Message decoded

Display number of messages decoded

To TParseData::
Process_Serial_Data

From TWriteData

Figure 3.26 Decode_Message function in class TParseData of program RxData
of messages that have been decoded during the current session is calculated and the results are displayed on the main program interface. At this point the program returns to function Process_Serial_Data.

Finally, if the user clicks “Help” from the main program interface, class TAbout is called and a form is created which gives details regarding program RxData. The user can either click “OK” or click “X” in the upper right hand corner of the box to close the form and return to the main program interface. This can all be seen in Figure 3.27.
Figure 3.27 Class TAbout of program RxData
3.5.2 FTP Server

The software selected to run the FTP server was Serv-U FTP Server Version 5.0. A screenshot showing the details of this program can be seen in Figure 3.28. It should be noted that an FTP server was chosen over an HTML server due to the nature of this project. Since the purpose of this server is to transfer binary data files, an FTP server is not only more practical, but allows for a more secure computing environment. In addition, configuring an FTP server was far more simplistic than configuring an HTML server for allowing access to data files, while minimizing security concerns. To reiterate, the following reasons explain why an FTP server was chosen over an HTML server:

- Makes more sense for binary data transfers
- More secure than an HTML server
• Easier to configure to meet the needs of the project
• Greater ability to control access rights of end users

There are a number of reasons the Serv-U FTP server was chosen over its competitors. These include the following:

• Ease of setup/configuration
• Remote administration
• Ease of use
• Features
• Cost
• SSL Support
• Scalability
• Reputation

Figure 3.29 shows a screen shot of the domain settings for this FTP server. Note that a static IP address has been assigned and only SSL or TLS sessions are allowed in the current configuration. The use of only secure connections is called an implicit connection and this prevents any insecure or incompatible connection from being established. If explicit connections were allowed, both secure and insecure connections would be allowed between the server and the client. In addition, all communication with the server is done via port 990, which isn’t a common port for an FTP server (as they tend to function on ports 21 and 22). This helps decrease attacks on the machine.

One of the benefits of this FTP server is its user-friendly interface and useful features. One such feature is the logs which allow you to monitor the server’s status in real time. These logs are also archived so they can be viewed at a later time. Figure 3.30 shows an example of a session log which gives the details of the entire server session from start up until shut down. This log enables an administrator to determine which ports the server is listening on, enabling changes to be made to increase security or improve compatibility with an end user’s FTP client. Note that a domain log is also available, but it gives less
Figure 3.29 Domain settings of Serv-U FTP Server

Figure 3.30 Session Log in Serv-U FTP server
information about the server as all the information it shows is related to the domain in question.

In order to establish a secure, trusted connection with an FTP client, Serv-U FTP server uses a user-generated SSL certificate (Figure 3.31). This certificate is displayed on the client’s machine when originally establishing a connection with the server. Once this certificate is approved by the end user, it is saved to the client machine. This enables secure SSL sessions between the client and server to take place via the principals of Public Key Infrastructure (PKI) as discussed in Chapter 2.

Another important feature of Serv-U is the ability of the administrator to control where the client may navigate on the server machine. This is done by locking the user in the
home directory (see Figure 3.32) and thus, prevents them from leaving the directory that they log in to. In the case shown in Figure 3.32, this directory is c:\database.

In addition, the administrator may assign another layer of security, beyond SSL, to the exchange of passwords between the server and the client. This is done by using a technology called OTP (One-Time Passwords), which works by never sending a password over the Internet. Instead, it sends a one-way encrypted version of the password called ‘hash’ over the net. There is no way for a human to retrieve this password and even if it is decoded, the same ‘hash’ function is never used twice, so it is very safe. Hence the name OTP. [Serv-U Help files].
The Serv-U FTP Server supports a popular form of OTP called S/KEY. This form of OTP comes in two variants, MD4 and MD5, and the server supports both of these. These variants simply refer to the name of the hash function being used when encrypting the password. This can be seen in Figure 3.33 where the password type has been set to OTP S/KEY MD5. Once this change has been made the user’s password must be reentered and applied to the account since Serv-U needs to know the password when using S/KEY. The password already stored in the account cannot be decrypted by Serv-U and, therefore, must be reentered.

Finally, various levels of administrative controls can be assigned to each user’s account (see Figure 3.34). These access controls can be broken down as follows:
Figure 3.34 Assigning access controls to user accounts in Serv-U FTP server

- Read – allows user to read files in assigned directories
- Write – allows user to write files in assigned directories
- Append – allows user to append to files in assigned directories
- Delete – allows user to delete files in assigned directories
- Execute – allows user to execute program files in assigned directories
- List – allows user to see a list of directories on server
- Create – allows user to create directories on server
- Remove – allows user to remove directories on server
- Inherit – allows sub-directories to inherit all of user’s primary directory access attributes

Note that the vast majority of these access controls would fall under the category of administrative controls. Thus, they enable an administrator to control various functions of
the FTP server from a remote location. This is a very important feature of this software since the server will be physically located in a remote location.

3.5.3 Software Firewall

![Sygate Personal Firewall Pro Details](image)

Figure 3.35 Sygate Personal Firewall Pro details

The software selected to run the software firewall was Sygate Personal Firewall Pro Version 5.5. A screenshot showing the details of this program can be seen in Figure 3.35.

There are a number of reasons this software firewall was chosen over its competitors. These include the following:

- Ease of setup
- Ease of use
• Features
  o Traffic history and attack history graphs
  o Log creation
  o Backtracing
• Customer ratings and technical software reviews

Figure 3.36 shows the firewall interface which allows for real-time network activity monitoring. This screen is very useful if an attack on the computer is suspected since incoming and outgoing network traffic can be monitored instantly. In addition, known attacks will be displayed in the attack history graph. Thus, if an attack occurs when an
administrator is on hand, the attack can be resolved by blocking the ports being attacked, by preventing a response to inquiries from the attacker, or by some other means made possible through the firewall. In addition, all running applications are displayed and an icon indicates if the program is allowed to access the Internet, must ask for permission to access the Internet, or is blocked from accessing the Internet. This is a very powerful tool that allows an administrator to enable certain programs to gain access to the Internet (e.g. Serv-U FTP Server), while denying unnecessary and/or high risk programs from accessing the Internet (e.g. Internet Explorer).

In order to aid the system administrator in keeping up with potential security threats and/or problems, Sygate Personal Firewall has a set of logs, as seen in Figure 3.37, which record essential data to be monitored in either real time or at a later time. This is a benefit if the administrator is unable to monitor the system and the system has undergone an attack. By logging this information, the firewall allows the user to analyze these security threats at a later time and determine what course of action to take.

An example of one of these logs, the “Traffic Log”, is shown in Figure 3.38. This log records all incoming, outgoing, and blocked traffic and alerts the user to the security threat that each poses. Much useful information can be found in these logs and another powerful feature of Sygate Personal Firewall is the ability to back trace to the origin of the threat to determine the attacking computer’s IP address, who maintains the machine, and where the computer is located. This is very useful if there is a need to report the attacker to high level administrators, to the authorities, or to the attacker himself/herself.

In addition to the logs, the firewall software also allows the system administrator to be contacted through email in the event of an attack or security breach. This important
Figure 3.37 Sygate Personal Firewall showing logs

Figure 3.38 Traffic Log in Sygate Personal Firewall
feature could prevent the attacker from crashing the computer or otherwise damaging the machine. This is set up as shown in Figure 3.39.

In addition, a number of security features can be set according to preference to produce a more secure environment or allow for more freedom in computer usage. This is accomplished as shown in Figure 3.40. The features listed and their associated functions are as follows:

- **Enable Intrusion Detection System:** Analyzes network packets and compares them with known attacks and patterns of attack and then blocks those attacks.

- **Anti-MAC spoofing:** prevents Media Access Control (MAC) which is a spoofing attempt to hijack a communication session between two computers

- **Portscan Detection:** detects port scans of the system
Figure 3.40 Setting up Security Options in Sygate Personal Firewall

- **Anti-IP spoofing**: prevents Internet Protocol (IP) spoofing which attempts to hijack a communication session between two computers
- **Driver-Level Protection**: blocks protocol drivers from accessing the network unless they are given permission by the user
- **OS fingerprint masquerading**: keeps programs from detecting the O/S of a computer
- **Stealth mode browsing**: keeps computer hidden from web servers while on a network
- **NetBIOS Protection**: blocks all communication from computers located outside the Personal Firewall’s local subnet range
- **DoS Protection**: checks incoming traffic for Denial of Service (DoS) patterns and protects against them
- **Anti-Application Hijacking**: checks for malicious applications that work by interjecting DLLs and Windows hooks into Windows applications and blocks them
• **Block Universal Plug and Play traffic:** blocks this feature to eliminate the vulnerabilities associated with it.

• **Block attackers IP address for XXX seconds:** blocks all communication from a host source once an attack has been detected.

• **Block all traffic when the service is not loaded:** prevents any traffic from entering or leaving before the firewall is up and running.

• **Enable DLL authentication:** firewall determines which DLLs are used by which trusted applications and stores that information. Applications running DLLs not trusted are blocked.

Each of these features helps combat the numerous types of attacks that the computer might face while connected to the Internet. The different types of attacks that can occur while a computer is connected to the Internet have been discussed in Chapter 2.
3.6 FTP Client

The software selected as the FTP client was FTP Voyager Version 11.0. Details regarding the program are shown in Figure 3.41. There are a number of reasons this FTP client was chosen over its competitors. These include the following:

- Ease of setup/configuration
- Features
- Ease of use
- Secure FTP (FTPS) support using SSL
- Cost
- Password encryption
- Produced by same company as the Serv-U FTP Server (RhinoSoft)

A screen shot of FTP Voyager is shown in Figure 3.42. What should be noted is that the top portion of the screen displays the interaction between the Serv-U server and the FTP
Figure 3.42 FTP Voyager in "Simple" Mode connecting with Serv-U Server
client, the lower left-hand portion shows the files on the local machine, and the lower right-hand corner displays the files available for downloading. Note that when the client connects to the server, an SSL connection is established before the username and password are transmitted. This enables these to be transmitted securely. Once the connection is established, the server enters passive (PASV) mode and reestablishes an SSL connection so files are transferred securely. Notice that the client shows the files located on the server as being located in the directory “/” and, as such, prevents the user from knowing where they are located on the server. This is another security feature of Serv-U and FTP Voyager and can be seen in the lower right-hand portion of the screen.

In addition to the screen shown in Figure 3.42, FTP Voyager also offers a “regular” or “complex” mode which puts more information at the user’s fingertips. This can be seen in Figure 3.43. Note that information about the server-client interaction is now located in the bottom left of the screen, information about the files located on the server is shown in the top right of the screen and information about the local machine is shown on both the center left and right hand portions of the screen. The top left shows the machine the client is connected to and the bottom right displays the status of files that are being or have been downloaded to the client. It should be noted that the change from ‘regular’ to ‘simple’ mode only affects the look and feel of the screen. Thus, it is up to the end user to choose the mode he/she is most comfortable using.

3.7 FTP Server-Client Interaction

For a successful session to occur between server and client machines, several things must occur. First, a secure connection must be established and then a secure connection through a PASV port must be established to transfer files through the firewall protected
server. Once these connections are successfully established, the client may download files from the server. These transactions can be seen in the “Domain Log” of the “Activity” portion of the “OU Airport Database” domain in Serv-U (Figure 3.44) and the upper portion of the screenshot in Figure 3.42.

Either during or following a transmission between the server and the client, a number of server operational statistics can be viewed in Serv-U under the “Users” tab of the “Activity” portion of the “OU Airport Database” domain as shown in Figure 3.45. These stats include the time a user has been connected to the server, the average download speed to the client and the amount of data downloaded to the client. Other operational statistics are also
Figure 3.44 Domain log showing session between the server and client machines

Figure 3.45 Information regarding user download as seen in Serv-U
shown which could prove useful to the server administrator in the event that troubleshooting a problem or limiting certain aspects of user accounts (e.g. download speed, etc) becomes necessary.

From the client standpoint, once a secure connection has been established, files can be downloaded. The user simply selects the files he/she would like to download from the server and selects “download” after right-clicking the selection with the mouse. FTP Voyager then displays an additional screen which reports the status of the download and estimated time for the entire download to complete (Figure 3.46). The option is also given for the user to cancel the download, which could prove very valuable should the download take too much time, etc.

Figure 3.46 Screenshot showing a download from Serv-U to FTP Voyager
Figure 3.47 shows the FTP Voyager program following a successful download of files from Serv-U. Notice that following each file transfer, the server and client reestablish an SSL connection to securely transfer the next requested file. Following a transfer FTP Voyager gives information regarding the total bytes transferred, total transfer time, and average transfer rate during the SSL session, as well as the location of the downloaded file. All of this can be seen in the upper section of the figure. The files are now listed in the local directory as well as seen in the lower left-hand portion of the figure.
Finally, a screenshot of the firewall software is shown just after a transfer has completed. Notice, in Figure 3.48, that both the incoming and outgoing traffic history portions of the screen both register transferred bytes during a display. However, the amount of outgoing traffic is on the megabyte level whereas the amount of incoming traffic is on the kilobyte level. The reason data is received by the server during a download is due to the handshaking, SSL communications, and other types of communication between the client and server during a transfer.

![Figure 3.48 Screen shot of Sygate firewall during a file transfer](image-url)
4 RESULTS AND DISCUSSION

A number of components were configured to allow individuals to securely access a database of binary GPS data and download the data for analysis in a secure fashion. The database was created using RxData, a program engineered in C++ to capture the data from the serial port and create the database. A secure FTP server, Serv-U, was used to allow secure connections and transfers with the computer. A software firewall, Sygate Personal Firewall Pro, was used to guard the computer from attacks while a hardware firewall was created to protect the LGF data logger. FTP Voyager was chosen as the secure FTP client due to its compatibility with the SFTP server. RxData, Serv-U, and Sygate Personal Firewall Pro are all located on the same computer.

One concern when setting up this system was housing the database and server on the same computer. Generally, when constructing such a system, the units are placed on separate machines to ensure that the computers can keep up with the demands placed on them. However, due to the relatively low number of expected users of this system at any given time, the benefits of using a single computer to house the multiple components outweigh any constraints such a configuration presents. To ensure there were no major issues before going online, the system was tested extensively to check for limits in the computer’s hardware components through the use of Windows “Performance Monitor”. Performance Monitor is a Windows administrative tool that allows various components of the computer to be monitored. Performance Monitor was run while RxData was collecting, decoding, and storing data, while the Sygate firewall was running, and while Serv-U was either running and
not communicating (for baseline data) or was communicating and transferring data to one or more users.

Results indicate that no problems will be seen while a single user is using the server. Results also indicate that few, if any, problems will be encountered when there are multiple server sessions in progress. The most likely problem would be a bandwidth limitation if enough users were transferring data from the server simultaneously. The problem could be corrected, if necessary, by restricting user’s downloading bandwidth. Graphical results from the Windows Performance Monitor are discussed below.

Figure 4.1 shows baseline data for the FTP server and database computer. During this test, RxData was running, Serv-U was running but there were no connections and no data transfers, and Sygate firewall was running. Several parameters were monitored including the:

- Committed RAM (shown in MB X 10)
- kBytes Sent/sec (X100)
- Current Bandwidth (in MB)
- Packets Sent/sec (X10)
- Threads
- Average Disk Queue Length (X0.01)
- % Processor Time
- Interrupts/sec (X100)
- Current Connections

It should be noted that in this and all the other figures that follow, the units on the ordinate are dependent on the parameter being monitored. The parameters were fashioned in such a way so they could all be viewed on the same graph. As a result, several of the parameters
must be multiplied by a factor to obtain the correct value for that parameter. These multiplication factors are shown in the list above and should become clear in the discussion that follows.

During the baseline test the committed RAM was 180 MB and the threads ranged between 40 and 45, while the current bandwidth was 10 MB. None of these changed when data arrived via the serial port and RxData began capturing and processing this data. The percent of processor time increased to approximately 13 while data was being received and captured and the interrupts/sec rose to approximately 1900. Both of these decreased following the stop of data arrival. All other parameters remained at or near 0.

Figure 4.1 Windows System Performance Monitor—RxData only (No Serv-U downloads)

Figure 4.2 shows data for the FTP server and database computer when RxData was running, Serv-U was running and there was one connection and download, and the Sygate
firewall was running. It can be seen that once Serv-U starts transferring data, there is an increase in the processor time from 13% to between 22% and 37% (average of 31%), an increase in interrupts/sec to 2500, an increase in packets/sec to 250, an increase in kBytes Sent/sec to approximately 1050, an increase in the average disk queue length to approximately 0.05 seconds, and an increase in connections to 1. The committed ram, threads, and current bandwidth exhibited no change.

![Figure 4.2 Windows System Performance Monitor—1 Serv-U User](image)

Data for the FTP server and database computer when RxData was running, Serv-U was running and there were two connections and downloads, and Sygate firewall was running is shown in Figure 4.3. It should be noted that there are no users logged in or downloading from Serv-U from the time the test begins to 110 seconds and from 1150 to 1240 seconds, there is one user from 120 to 230 seconds and from 980 to 1140 seconds, and there are two
users from 240 to 970 seconds. All of this can be seen by monitoring the current connections. Note that the processor time increases from 13 % when there are no users downloading from Serv-U to approximately 31 % when one user is downloading. When two users are downloading the processor time increases slightly to 35 %. The number of interrupts increased from 2500 to 2700 per second and the number of packets increased from 250 to 270 per second when the number of users downloading from Serv-U increased from one to two. The number of kBytes Sent per second increased slightly from 1050 to 1130 when the downloads increased from one to two. The committed ram, threads, and current bandwidth exhibited no change as the number of Serv-U downloads changed.

Figure 4.3 Windows System Performance Monitor—multiple Serv-U Users (2 downloads)
Finally, data for the FTP server and database computer when RxData was running, Serv-U was running and there were three connections and downloads, and Sygate firewall was running is shown in Figure 4.4. It should be noted that there are no users logged in or downloading from Serv-U from the time the test begins to 130 seconds and from 1190 to 1330 seconds, there is one user from 140 to 230 seconds and from 1110 to 1180 seconds, there are two users from 240 to 490 seconds and from 1010 to 1100, and there are three users from 500 to 1000 seconds. This can be seen by monitoring the current connections. Note that the processor time increases from 13 % when there are no users downloading from Serv-U to approximately 32 % when one user is downloading. When two users are downloading the processor time increases slightly to 34 % and when three users are downloading it increases even more slightly to just above 35 %. The number of interrupts increased from 2500 to 2700 per second and the number of packets increased from 250 to 270 per second when the number of users downloading from Serv-U increased from one to two. However, when the number of users downloading increased from two to three, the interrupts decreased to 2500 per second while the number of packets remained steady at 270 per second. The reason for this is unclear, but had no bearing on system performance. The number of kBytes Sent per second increased from 1050 to 1130 when the downloads increased from one to two and only increased slightly to 1135 when the number of downloads increased from two to three. The committed ram, threads, and current bandwidth exhibited no change as the number of Serv-U downloads changed. It is also unclear why the average disk queue length varied throughout the test, but it had no impact on system performance.

The important thing to note when comparing these graphs is that the increase in processor time is very slight as the number of users downloading from Serv-U increases. The
reason this is believed to be the case is that the program or network seems to be bandwidth limited as can be seen by comparing the kBytes Sent/sec. There is a modest increase when the number of downloads increases to two, but there is almost no increase when the number of downloads increases to three. As a result there is a large amount of processor power left to perform other functions on the computer, even when multiple users are downloading from Serv-U. Thus, the results shown and discussed above indicate that the computer is more than adequate to serve as both the database and FTP server while running the software firewall. No problems are expected to be encountered due to this configuration.
5 CONCLUSIONS

5.1 System Overview

A secure web-based data collection and distribution system for GPS research was created. This system was built in order to make data obtained from GPS satellites available over the Internet to authorized individuals. The benefits of such a system are numerous, but the primary benefit is a time savings since data is instantly accessible as it is placed online.

In order to create such a system, a number of factors had to be taken into account. The primary consideration was the security of the LGF data logger at the Ohio University Airport. Since this system is the heart of the data collection system and can’t easily be replaced, it was of the utmost importance that access was not able to be gained into this system. The safety of this system was ensured by creating a two-wire null modem cable that would only allow data to be transferred in a one-way direction—from the data logger to the database. Thus, access cannot be gained to this system through the Internet. Therefore, putting this system online poses no threat to the LGF data logger.

Another consideration was the security of the computer running the database and server. The safety of this system was ensured through the use of several software programs. A secure FTP server, as opposed to an HTML server, was run to decrease the susceptibility of the server. The number of ports being used by the server was minimized to decrease those available for attack and a software firewall was configured to severely hamper any program other than the server from accessing the Internet. In addition, access to the server is only granted to clients that approve of the server-generated security certificate and those that have created a client-side security certificate on their computer. These certificates verify the
authenticity of the server and client and allow SSL transactions to take place. These SSL transactions are the only type allowed between the server and the client, and as such, only 128-bit encrypted transactions transpire through the server. In addition, user passwords are protected using an S/KEY hash function known as MD5 that creates a one time password (OTP). So even if a hacker intercepts a password and successfully decrypts it, the password will prove useless since it will be unable to provide access to the server. While the only sure way of protecting this computer would be to keep it offline, the steps taken to guard the computer, data, and passwords should provide a high level of security that will prevent the vast majority of attacks from occurring successfully on the system.

Another consideration when putting this system together was the data transfer speed needed between the LGF data logger and the database to ensure that data doesn’t accumulate on the data logger and isn’t lost during the transfer. Based on the amount of data currently collected and the probability of adding GPS receivers in the future to collect more data, it was assumed that up to 1 GB of data would be collected and transferred per 24 hour period. Thus, the throughput of the serial ports would have to be approximately 115.7 kbps. This is above the maximum transfer rate of 115.2 kbps and well above the working transfer rates of between 110 and 113 kbps. However, the current transfer speeds are more than adequate to handle the amount of data that is currently generated and should allow for a significant increase in GPS receivers before problems are encountered in this area. Thoughts on improvements in this area of the system are given below.

5.2 Future Work/Considerations

One possible concern is whether the Serv-U FTP server allocates bandwidth equally to users during simultaneous downloads. The server seems to allocate the majority of the
bandwidth to the client with the least “resistance” and whatever portion of the bandwidth remains goes to the remaining clients. Thus, one client may obtain download speeds of between 900 and 1000 kBps while the other may only obtain speeds between 50 and 100 kBps.

This problem may be resolved when the server is relocated to the OU Airport and a static IP address is obtained. Since upload speeds tend to be limited, more bandwidth may be allocated when the server is set up. This is the preferred option as it would be more practical during the times when a single user is downloading data from the server and the maximum possible transfer rate is desired.

Another option may be to limit the maximum bandwidth per user and thus, leave a larger portion of the bandwidth available to other users so simultaneous downloads are more equivalent. The problem with this approach would be the limitations placed on client download speeds even when there is a single user connected to the server. Therefore, this is not the preferred solution to the problem.

In any case, this is a portion of the system that can be improved in the future.

Another area of concern is the role of the administrator in overseeing the system. The minimum duties that would be required of the administrator to keep the system up and running would be, in terms of the system component:

- Microsoft Windows XP
  - Apply security updates and patches as soon as they are available
  - Update and run virus software routinely to protect computer and users
  - Remove and/or backup data files to DVD-R or other media to ensure hard disk doesn’t fill up
- RxData
o Ensure program is operating properly
o Make improvements in program as necessary
o Watch out for data being lost when transferred from LGF data logger

- Sygate Personal Firewall Pro
  o Monitor activity between the computer and network
  o Log and report attacks on computer
  o Keep software updated

- Serv-U FTP Server
  o Create new user accounts
  o Delete old user accounts
  o Monitor activity and adjust user and/or server parameters, as necessary, to ensure optimal server performance
  o Keep software updated

In order to perform these duties and learn about the multitude of programs running the system, it is suggested that a manual be put together for each component describing the basics and requirements of each program. This would enable an administrator to get up to speed on the system and perform effectively with a minimum investment in time. This would also enable, in theory, somebody unfamiliar with the system to act as an interim administrator should the primary administrator be unable to get to the computer during or after an emergency has emerged.

Also, as was mentioned in the prior section, the amount of data that is collected by the GPS receivers may exceed the transfer capabilities of the serial ports in the future. Therefore, an upgrade to either a parallel, USB or some other type of port would be necessary to meet these demands. This would entail reprogramming both the data logger and the
database computers. However, by making such an upgrade, the data transfer limitations that currently exist would be largely removed due to the increased capacity of these other types of ports as they all have 10 or more times the capacity of the serial port.
6 REFERENCES


7 BIBLIOGRAPHY

7.1 Sources Used to Learn About Networks


7.2 Sources Used to Learn C and C++


