Net-Secure Application

Submitted to

Amity University Uttar Pradesh

in partial fulfilment of the requirements for the award of the Degree of
Bachelor of Technology in
Name of Department
By

Anand Yati

Under the guidance of

Mr Madan Lal Yadav

Department of Computer Science & Engineering
Amity School of Engineering and Technology

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY
AMITY UNIVERSITY UTTAR PRADESH
NOIDA (U.P.)

DECLARATION

I, Anand Yati, student of B.Tech (Computer Science & Engineering) hereby declare that the project titled "**Net-Secure Application**" which is submitted by me to Department of Computer Science & Engineering, Amity School of Engineering and Technology, Amity University, Uttar Pradesh, Noida, in partial fulfilment of requirement for the award of the degree of Bachelor of Technology in Computer Science, has not been previously formed the basis for the award of any degree, diploma or other similar title or recognition.

Noida
Date:

Anand Yati
Name and signature of Student

CERTIFICATE

On the basis of declaration submitted by Anand Yati, student of B. Tech Computer Science & Engineering I hereby certify that the project titled "**Net-Secure Application**" which is submitted to Department of Computer Science & Engineering, Amity School of Engineering and Technology, Amity University Uttar Pradesh, Noida, in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology in Computer Science, is an original contribution with existing knowledge and faithful record of work carried out by him under my guidance and supervision.

To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Noida

Date:

Mr. Madan Lal Yadav

Project Guide
Department of Computer Science & Engineering
Amity School of Engineering and Technology
Amity University Uttar Pradesh, Noida

Abstract

My main objective behind development of Net Secure application is to provide user with a group of functionalities which secure a user working on a network or surfing net. This application also supports elementary prototype of network attack on a specific IP address on a specific port.

The **functionalities** delivered by my **Net-Secure Application** are as follows :

1.) User can scan and **get risk level** to which his/her is exposed to on internet and network.
2.) User can **check UDP & TCP Traffic** on his machine.
3.) User can **get a list of active services** on his machine which are using internet or network resources.
4.) User can **get a list of active port numbers** on your system.
5.) User can **get number of inbound/outbound gateways** open on your machine.
6.) User can also **get number of Active Protocols, Active Services and Active Application Programs** which uses network resources.
7.) User can **detect the presence of 200+ Trojans and Viruses.**
8.) User can **scan for the active windows private ports.**
9.) **Anti-Phishing** module of this application **compares two URLs to identify** whether **they point to same resource or not** and whether these resources are located on a same website or different.
10.) Anti-Phishing module also **tells the protocol active** on a particular URL ,the **port number** on which that resource is located ,and **exact address** of that resource.
11.) **Network-spy module** of this application is a platform developed to **implement simple network attacks** on a particular IP address and port number.
12.) With the help of Network-spy module user can scan a particular IP address to get a list of open connection ports on which attack is possible.
13.) Users are also provided with **advanced help feature** in Network-Spy module which **suggests user with ports which are vulnerable** to attack.
14.) The **Web Page scanner module** of my application scans a web page to **extract all its control information and the complete coding** behind any web page.
15.) Appropriate **Multi-Threaded** programs are used wherever possible.
16.) All these data is **represented with statistics** like progress bar, percentage displays and appropriate colour change.

In order to deliver these functionalities I have tested my application on different machines and have verified that all of its functionalities are working properly.

Table of Content

INTRODUCTION

1.1 Introduction

Net-Secure Application is a collection of four independent modules, where each module has a separate set of functionalities delivered to its user. The four modules implemented in this application are:

- System Scanner Module
- Anti-Phishing Module
- Network Spy Module
- Web-Page Scanner Module

All of these modules provide user with various threat detection, Network spying and system scanning facilities.

1.1.1 System Scanner Module

This module provides a range of services to its user. This is indeed one of the most useful and prominent module of my application. The main functions performed by this module are:

- Threat detection: It can detect for presence of 200+ Trojans and viruses.
- Active Service detection: It detects all the active network services on your machine.
- Risk Level: It calculated and displays the risk level to which your system is exposed to on internet/network.
- TCP & UDP Traffic: Its shows the TCP & UDP traffic on your system as a percentage of total network traffic.
- Active Ports: It shows the list of active ports on your system.
- Open Gateways: It displays the number of active open gateways in your system which allows inbound/outbound connection to your machine.
- Active Protocols: It displays number of active protocols which are governing your current network usage.
- Active Core Services: It displays number of windows active core services which are using your system's network resources.
- Active Application Programs: It displays number of active application programs which are using your system's network resources.
- All these data is represented in well structured form supported with statistical features like progress bar, percentage displays and appropriate colour change of a progress bar according to the item displayed.

1.1.2 Anti Phishing Module:

This module protects its user from Phishing. Phishing is a term used when a user clicks on a fake link which looks like a genuine one, by clicking a user can lose away all his current session's information which includes but not limited to user id and password. In worse scenarios user can end up downloading a virus or giving away a malicious attacker full access of user's machine. These are the risks imposed by Phishing.

The Anti Phishing module provides following functionalities to its user to safeguard against Phishing:

- Compares two URLs to identify whether they point to same resource or not and whether these resources are located on a same website or different.
- It tells the protocol active on a particular URL , port number on which that resource is located ,and exact address of that resource.

1.1.3 Network Spy Module:

This module helps a user to understand how network attacks are make and also enables a user to make a simple data attack on any of the active IP address and port number in his network or internet.

The various functionalities provided by this module are:

- Implement simple network attacks on a particular IP address and port number.
- User can scan a particular IP address to get a list of open connection ports on which attack is possible.
- Users are also provided with advanced help feature in Network-Spy module which suggests user with ports which are vulnerable to attack.
- Users are also provided with features like single port scanning and multi port scanning of a Target IP address. Where user can check whether a particular port number on a particular IP Address is vulnerable to network attack or not. This scanning can also be applied on a range of port numbers to give the list of vulnerable port numbers.

1.1.4 **Web Page Scanner:**

This module helps its user to extract every possible information about a webpage. With the help of this module user can see what information actually its current web page carries related to the user and web page.

The functionalities provided by webpage scanner are:

- Scans a web page to extract all its control information and the complete coding behind any web page. This control information includes protocol of webpage, creation/expiry date & time, server name, domain name, IP address and many more.

FEASIBILITY REPORT

2.1 Technical Feasibility:

My project is completely feasible as the technology used in its development is globally accepted, acclaimed and free of cost. The tools and technology used in my project are:

- JDK 1.5: Java development kit is a free to use software distributed by oracle systems. Its universally available and a proven/acclaimed platform for development.
- NetBeans IDE 6.5: It is a free of cost IDE used for development of JAVA applications. Its offers a standard development platform and run time environment to its user. Its easy to use and supports its users in development of efficient and error-free codes.
- Windows 2000 or higher operating system: - This is also being used by the current system.

2.2 Operational Feasibility:

Operational feasibility is also there because the proposed solution can fit in with existing operations and right information at right time is provided to users. A single operator is required to work on the project. The knowledge of present operator is sufficient enough to operate the project. The project is easy to work with. The user can easily operate multiple features of this application. In case user finds its difficult to understand the functionality of a module. Appropriate help is provide to him there.

2.3 Economical Feasibility:

Our project is economically feasible because money spent is recovered by saving and user satisfaction. Cost benefit analysis is necessary to determine economic feasibility.

Cost consists of both direct and indirect cost. Most of the software and tools used in development of this application are free to use resulting in a very low cost but yet efficient and powerful solution.

Details of direct cost are:-

- Cost of software. Software used is NetBeans IDE 6.5 & JDK 1.5.
- Cost of peripherals like pen-drive, printer, DVDs etc.

- Cost of stationary.
- Cost of the electrical source (0.2unit/hr & Rs.3.5/unit).
- Cost of printing user manuals and documentation.

Details of indirect cost are:-

Indirect cost will include time spent by the team in discussing problems with each other and gathering information about developing application.

MATERIAL & METHODOLOGY

3.1 Materials Used :

I have used a number of software resources including NetBeans IDE & JAVA technologies which I have mentioned here. This code is fully forward compatible with all the new versions of the software I have used.

3.2 Software Specification:

- NetBeans IDE 6.5
- JDK 1.5
- MS Office(Ms Access)

3.3 Technology Used:

- J2SE Standard Components
- Java Networking Programming
- Java Swings.
- Ms Access Database

3.4 Hardware Requirements:

1. 800 MHz Intel Pentium III Processor (or later)

2. Windows 2000 or Windows XP or later

3. 256 MB RAM

4. 1024x768 16 bit display (32 bit recommended)

5. 1 GB Free HDD space

3.5 Methodology:

1) Design Phase
2) Coding Phase
3) Testing Phase

3.5.1 Design Phase:

Interface of the application is designed keeping in mind various functionalities to be delivered, user friendliness and ease of use. Apart from designing the interface various possible functionalities of the application are figured out and are incorporated in the action plan.

First of all the database for the scanner module was created. Then I created the user interface for various modules. In the end these interfaces were connected to the central control panel.

3.5.2 Coding Phase:

Coding for all the modules is done separately and then they are tested for their proper functioning and then they are integrated into a single working entity. As soon as the user interface was complete for a module I integrated the coding in it. First the database connection was made and then the program logic was implemented.

3.5.3 Testing Phase:

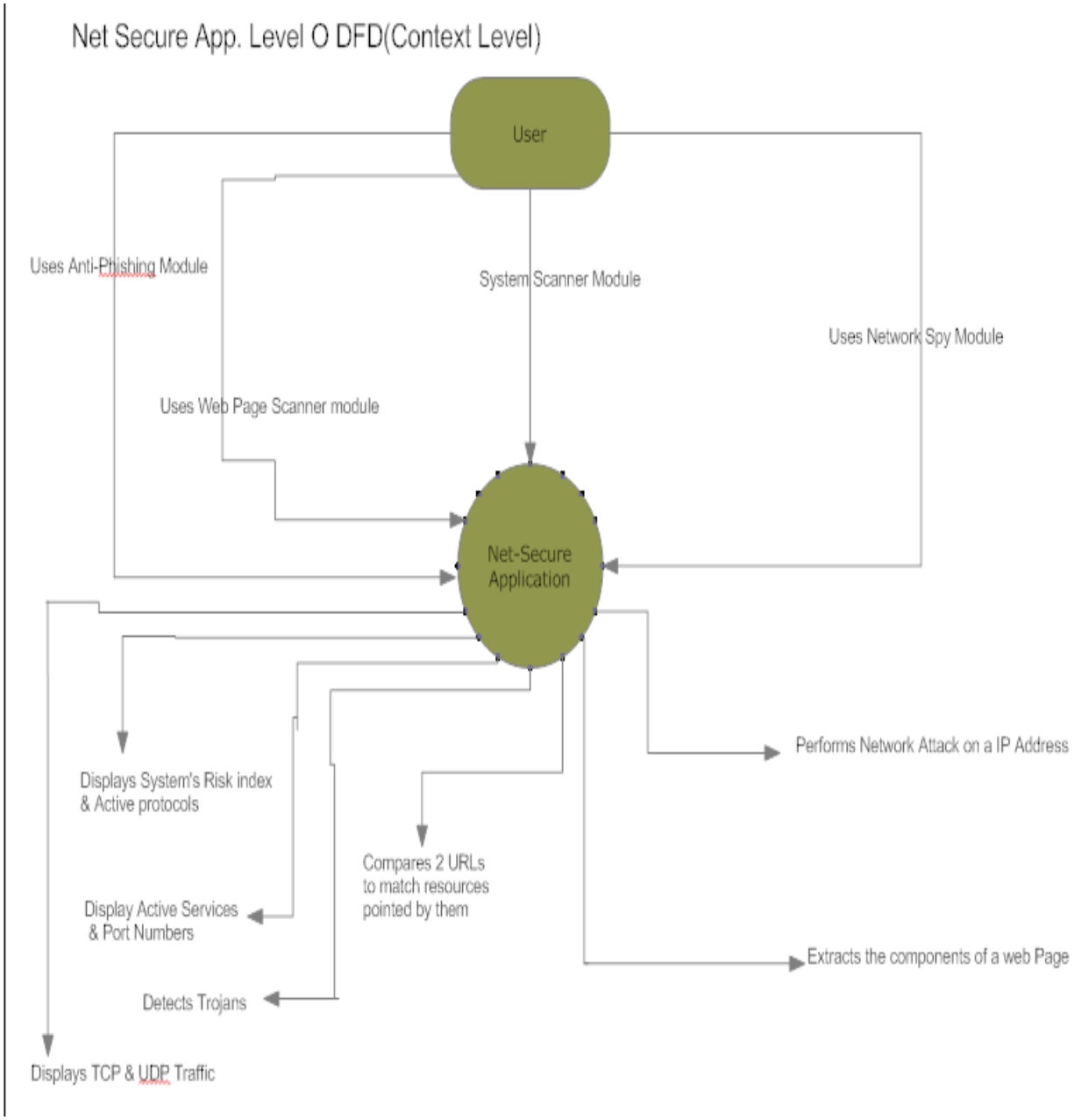
In this phase the code is searched for possible errors or loopholes. The errors or loopholes can be in coding, application of logic or weak functioning. All these issues are jotted down and solved before releasing the final version of product.

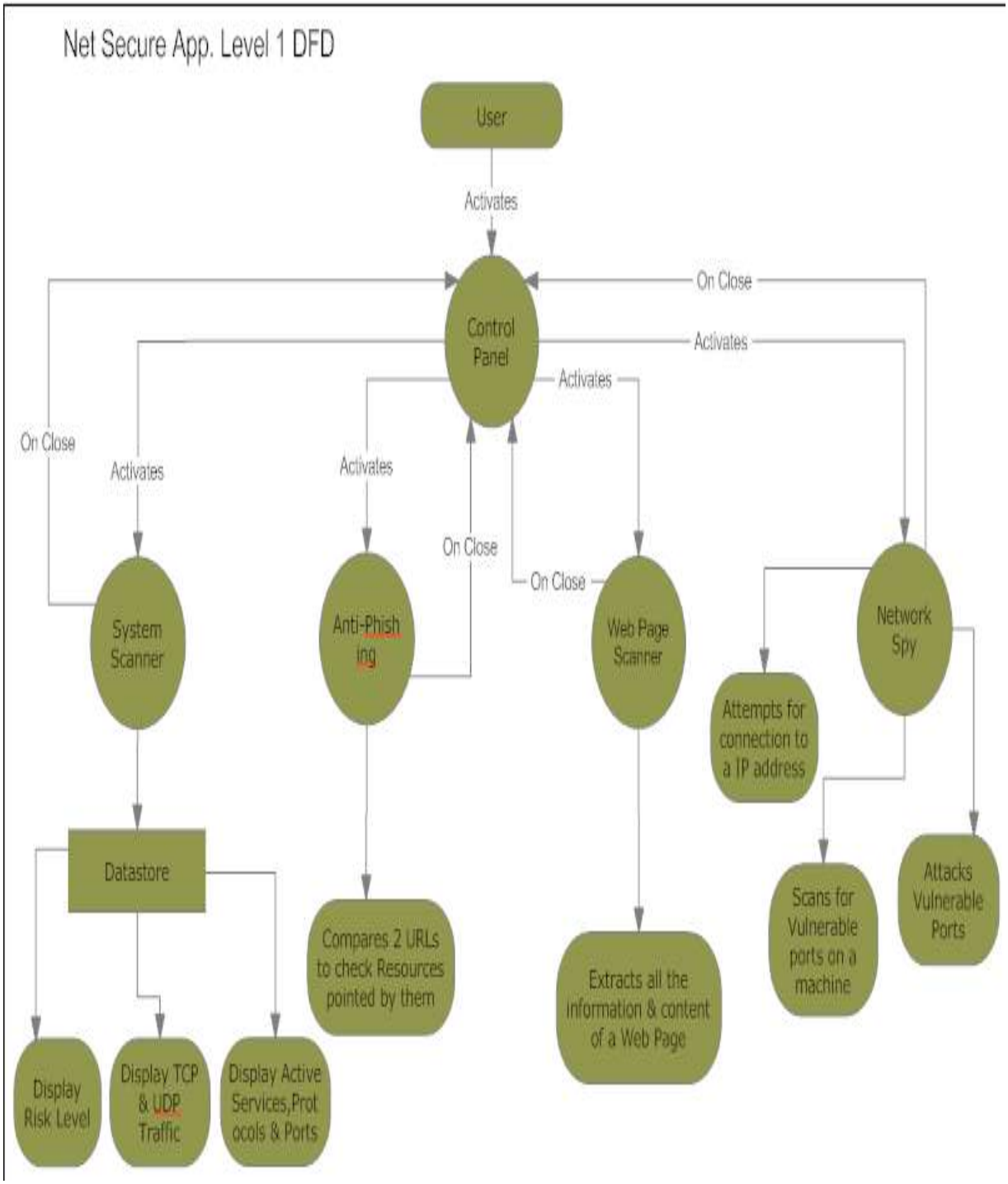
1. Alpha Testing: Alpha testing is done at the developers end in order to find and rectify any errors or abnormal behaviour of code in certain situations. In this case being the developer I did the alpha testing for this code i.e. project. I thoroughly tested each and every functionality of the Net Secure Application and entered various test data to ensure the correct working and integrity of the client. I rectified all the errors if found in this testing phase and hence improved the stability of the Net-Secure Application.

2. Beta Testing: Beta testing is done at the client's end which in these cases were students of Amity University. They tested this code thoroughly and checked for all the functionalities applied. They also tested this code for redundancy and ensured that this code satisfies all minimum specifications in best and most efficient way. This testing made this code even more perfect.

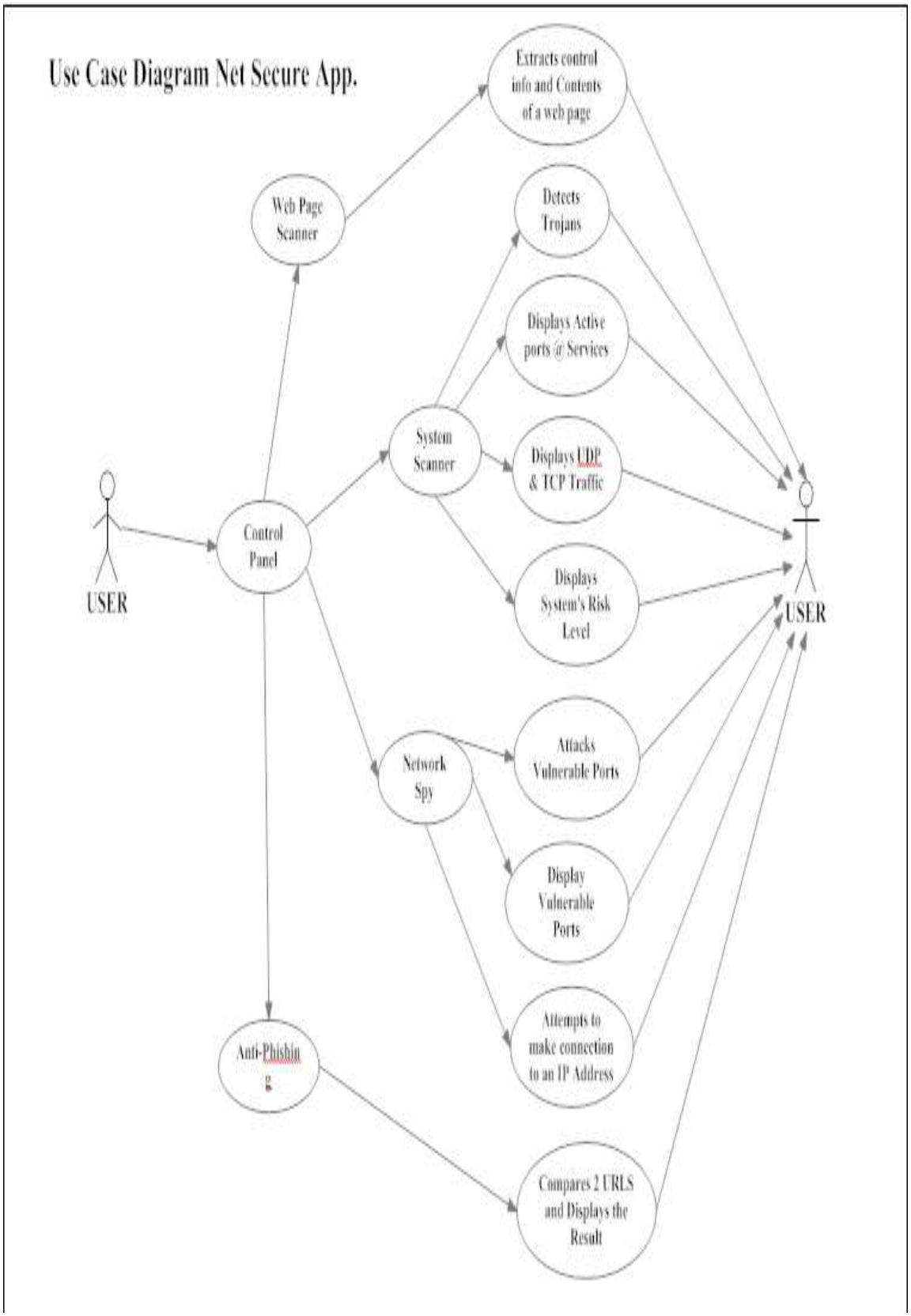
3.6 Data Flow Diagrams:

DATAFLOW DIAGRAM


Net Secure App. Level O DFD(Context Level)
User
Uses Anti-Phishing Module
System Scanner Module
Uses Network Spy Module
Uses Web Page Scanner module
Net-Secure Application
Performs Network Attack on a IP Address
Displays System's Risk index & Active protocols
Compares 2 URLs to match resources pointed by them
Display Active Services & Port Numbers
Extracts the components of a web Page
Detects Trojans
Displays TCP & UDP Traffic


Net Secure App. Level 1 DFD
User
Activates
Control Panel
On Close
Activates
Activates
On Close
Activates
Activates
On Close
On Close
System Scanner
Anti-Phishing
Web Page Scanner
Network Spy
Attempts for connection to a IP address
Datastore
Compares 2 URLs to check Resources pointed by them
Extracts all the information & content of a Web Page
Scans for Vulnerable ports on a machine
Attacks Vulnerable Ports
Display Risk Level
Display TCP & UDP Traffic
Display Active Services,Protocols & Ports

USE CASE DIAGRAM


Use Case Diagram Net Secure App.
Extracts control info and Contents of a web page
Web Page Scanner
Detects Trojans
Displays Active ports @ Services
System Scanner
Displays UDP & TCP Traffic
USER
Control Panel
Displays System's Risk Level
USER
Network Spy
Attacks Vulnerable Ports
Display Vulnerable Ports
Attempts to make connection to an IP Address
Anti-Phishing
Compares 2 URLS and Displays the Result

TECHNOLOGIES USED

4.1 Technologies Used:

In this section I am going to discuss various technologies used in the development of my project. The different technologies used in developing **Net-Secure Application** are :

- J2SE Standard Components
- Java Networking Programming
- Java Swings.
- Ms Access Database

4.2 J2SE Standard Components:

Java Platform, Standard Edition or **Java SE** is a widely used platform for programming in the Java language. It is the Java Platform used to deploy portable applications for general use. In practical terms, Java SE consists of a virtual machine, which must be used to run Java programs, together with a set of libraries (or "packages") needed to allow the use of file systems, networks, graphical interfaces, and so on, from within those programs.

4.2.1 Nomenclature, standards and specifications

Java SE was known as Java 2 Platform, Standard Edition or J2SE from version 1.2 until version 1.5. The "SE" is used to distinguish the base platform from Java EE and Java ME. The "2" was originally intended to emphasize the major changes introduced in version 1.2, but was removed in version 1.6. The naming convention has been changed several times over the Java version history. Starting with J2SE 1.4 (Merlin), Java SE has been developed under the Java Community Process. JSR 59 was the umbrella specification for J2SE 1.4 and JSR 176 specified J2SE 5.0 (Tiger). Java SE 6 (Mustang) was released under JSR 270.

Java Platform, Enterprise Edition is a related specification which includes all of the classes in Java SE, plus a number which are more useful to programs which run on servers as opposed to workstations. Java Platform, Micro Edition is a related specification intended to provide a certified collection of Java APIs for the

development of software for small, resource-constrained devices such as cell phones, PDAs and set-top boxes.

The Java Runtime Environment (JRE) and Java Development Kit (JDK) are the actual files that are downloaded and installed on a computer in order to run or develop java programs, respectively.

4.2.2 General purpose packages

4.2.2.1 java.lang

The Java package `java.lang` contains fundamental classes and interfaces closely tied to the language and runtime system. This includes the root classes that form the class hierarchy, types tied to the language definition, basic exceptions, math functions, threading, security functions, as well as some information on the underlying native system. This package contains 22 of 32 `Error` classes provided in JDK 6.

The main classes in `java.lang` are:

- `Object` – the class that is the root of every class hierarchy.
- `Enum` – the base class for enumeration classes (as of J2SE 5.0).
- `Class` – the class that is the root of the Java reflection system.
- `Throwable` – the class that is the base class of the exception class hierarchy.
- `Error`, `Exception`, and `RuntimeException` – the base classes for each exception type.
- `Thread` – the class that allows operations on threads.
- `String` – the class for strings and string literals.
- `StringBuffer` and `StringBuilder` – classes for performing string manipulation (`StringBuilder` as of J2SE 5.0).
- `Comparable` – the interface that allows generic comparison and ordering of objects (as of J2SE 1.2).
- `Iterable` – the interface that allows generic iteration using the enhanced `for` loop (as of J2SE 5.0).
- `ClassLoader`, `Process`, `Runtime`,`SecurityManager`, and `System` – classes that provide "system operations" that manage the dynamic loading of classes, creation of external processes, host environment inquiries such as the time of day, and enforcement of security policies.
- `Math` and `StrictMath` – classes that provide basic math functions such as sine, cosine, and square root (`StrictMath` as of J2SE 1.3).
- The primitive wrapper classes that encapsulate primitive types as objects.
- The basic exception classes thrown for language-level and other common exceptions.

Classes in `java.lang` are automatically imported into every source file.

4.2.2.2 java.io

The `java.io` package contains classes that support input and output. The classes in the package are primarily stream-oriented; however, a class for random access files is also provided. The central classes in the package are `InputStream` and `OutputStream` which are abstract base classes for reading from and writing to byte streams, respectively. The related classes `Reader` and `Writer` are abstract base classes for reading from and writing to character streams, respectively. The package also has a few miscellaneous classes to support interactions with the host file system.

Streams

The stream classes follow the decorator pattern by extending the base subclass to add features to the stream classes. Subclasses of the base stream classes are typically named for one of the following attributes:

- the source/destination of the stream data
- the type of data written to/read from the stream
- additional processing or filtering performed on the stream data

The stream subclasses are named using the naming pattern *`XxxStreamType`* where *`Xxx`* is the name describing the feature and *`StreamType`* is one of `InputStream`, `OutputStream`, `Reader`, or `Writer`.

The following table shows the sources/destinations supported directly by the `java.io` package:

Source/Destination	Name	Stream types	In/Out	Classes
`byte` array (`byte[]`)	`ByteArray`	`byte`	in, out	`ByteArrayInputStream`, `ByteArrayOutputStream`
`char` array (`char[]`)	`CharArray`	`char`	in, out	`CharArrayReader`, `CharArrayWriter`
file	`File`	`byte`, `char`	in, out	`FileInputStream`, `FileOutputStream`, `FileReader`, `FileWriter`

string (`StringBuffer`)	`String`	`char`	in, out	`StringReader`, `StringWriter`
thread (`Thread`)	`Piped`	`byte`, `char`	in, out	`PipedInputStream`, `PipedOutputStream`, `PipedReader`, `PipedWriter`

Other standard library packages provide stream implementations for other destinations, such as the `InputStream` returned by the `java.net.Socket.getInputStream()` method or the Java EE `javax.servlet.ServletOutputStream` class.

Data type handling and processing or filtering of stream data is accomplished through stream filters. The filter classes all accept another compatible stream object as a parameter to the constructor and *decorate* the enclosed stream with additional features. Filters are created by extending one of the base filter classes `FilterInputStream`, `FilterOutputStream`, `FilterReader`, or `FilterWriter`.

The `Reader` and `Writer` classes are really just byte streams with additional processing performed on the data stream to convert the bytes to characters. They use the default character encoding for the platform, which as of J2SE 5.0 is represented by the `Charset` returned by the `java.nio.charset.Charset.defaultCharset()` static method. The `InputStreamReader` class converts an `InputStream` to a `Reader` and the `OutputStreamWriter` class converts an `OutputStream` to a `Writer`. Both these classes have constructors that allow the character encoding to use to be specified—if no encoding is specified then the default encoding for the platform is used.

The following table shows the other processes and filters supported directly by the `java.io` package. All of these classes extend the corresponding `Filter` class.

Operation	Name	Stream types	In/Out	Classes
buffering	`Buffered`	`byte`, `char`	in, out	`BufferedInputStream`, `BufferedOutputStream`, `BufferedReader`, `BufferedWriter`

"push back" last value read	`Pushback`	`byte`, `char`	in	`PushbackInputStream`, `PushbackReader`
read/write primitive types	`Data`	`byte`	in, out	`DataInputStream`, `DataOutputStream`
object serialization (read/write objects)	`Object`	byte	in, out	`ObjectInputStream`, `ObjectOutputStream`

Random access

The `RandomAccessFile` class supports *random access* reading and writing of files. The class uses a *file pointer* that represents a byte-offset within the file for the next read or write operation. The file pointer is moved implicitly by reading or writing and explicitly by calling the `seek(long)` or `skipBytes(int)` methods. The current position of the file pointer is returned by the `getFilePointer()` method.

File system

The `File` class represents a file or directory path in a file system. `File` objects support the creation, deletion and renaming of files and directories and the manipulation of file attributes such as *read-only* and *last modified timestamp*. `File` objects that represent directories can be used to get a list of all of the contained files and directories.

The `FileDescriptor` class is a file descriptor that represents a source or sink (destination) of bytes. Typically this is a file, but can also be a console or network socket. `FileDescriptor` objects are used to create `File` streams. They are obtained from `File` streams and `java.net` sockets and datagram sockets.

4.2.2.3 java.net

The `java.net` package provides special IO routines for networks, allowing HTTP requests, as well as other common transactions.

4.2.2.4 java.awt

Main article: Abstract Window Toolkit

The `java.awt`, or Abstract Window Toolkit, provides access to a basic set of GUI widgets based on the underlying native platform's widget set, the core of the GUI event subsystem, and the interface between the native windowing system and the Java

application. It also provides several basic layout managers, a datatransfer package for use with the Clipboard and Drag and Drop, the interface to input devices such as mice and keyboards, as well as access to the system tray on supporting systems. This package, along with `javax.swing` contains the largest number of enums (7 in all) in JDK 6.

4.3 Java Network Programming:

In the last 10 years, network programming has stopped being the province of a few specialists and become a core part of every developer's toolbox. Today, more programs are network aware than aren't. Besides classic applications like email, web browsers, and Telnet clients, most major applications have some level of networking built in. For example:

- Text editors like BBEdit save and open files directly from FTP servers.
- IDEs like Eclipse and IntelliJ IDEA communicate with CVS repositories.
- Word processors like Microsoft Word open files from URLs.
- Antivirus programs like Norton AntiVirus check for new virus definitions by connecting to the vendor's web site every time the computer is started.
- Music players like Winamp and iTunes upload CD track lengths to CDDB and download the corresponding track titles.
- Gamers playing Quake gleefully frag each other in real time.
- Supermarket cash registers running IBM SurePOS ACE communicate with their store's server in real time with each transaction. The server uploads its daily receipts to the chain's central computers each night.
- Schedule applications like Microsoft Outlook automatically synchronize calendars with other employees in the company.

In the future, the advent of web services and the semantic web is going to entwine the network ever more deeply in all kinds of applications. All of this will take place over the Internet and all of it can be written in Java.

Java was the first programming language designed from the ground up with networking in mind. Java was originally designed for proprietary cable television networks rather than the Internet, but it's always had the network foremost in mind. One of the first two real Java applications was a web browser. As the global Internet continues to grow, Java is uniquely suited to build the next generation of network applications. Java provides solutions to a number of problems—platform independence and security being the most important—that are crucial to Internet applications, yet difficult to address in other languages.

One of the biggest secrets about Java is that it makes writing network programs easy. In fact, it is far easier to write network programs in Java than in almost any other language. This book shows you dozens of complete programs that take advantage of the Internet. Some are simple textbook examples, while others are completely functional applications. One thing you'll notice in the fully functional applications is just how little code is devoted to networking. Even in network intensive programs like web servers and clients, almost all the code handles data manipulation or the user

interface. The part of the program that deals with the network is almost always the shortest and simplest.

In brief, it is easy for Java applications to send and receive data across the Internet. It is also possible for applets to communicate across the Internet, though they are limited by security restrictions. In this chapter, you'll learn about a few of the network-centric applications that have been written in Java. In later chapters, you'll develop the tools you need to write your own network programs.

What Can a Network Program Do?

Networking adds a lot of power to simple programs. With networks, a single program can retrieve information stored in millions of computers located anywhere in the world. A single program can communicate with tens of millions of people. A single program can harness the power of many computers to work on one problem.

Network applications generally take one of several forms. The distinction you hear about most is between clients and servers. In the simplest case, clients retrieve data from a server and display it. More complex clients filter and reorganize data, repeatedly retrieve changing data, send data to other people and computers, and interact with peers in real time for chat, multiplayer games, or collaboration. Servers respond to requests for data. Simple servers merely look up some file and return it to the client, but more complex servers often do a lot of processing on the data before answering an involved question. Peer-to-peer applications such as Gnutella connect many computers, each of which acts as both a client and a server. And that's only the beginning. Let's look more closely at the possibilities that open up when you add networking to your programs.

Retrieve Data

At the most basic level, a network client retrieves data from a server. It can format the data for display to a user, store it in a local database, combine it with other data sources both local and remote, analyze it, or all of the above. Network clients written in Java can speak standard protocols like HTTP, FTP, or SMTP to communicate with existing servers written in a variety of languages. However, there are many clients for these protocols already and writing another one isn't so exciting. More importantly, programs can speak custom protocols designed for specific purposes.

Also interesting is the use of existing protocols like HTTP to retrieve data that will be manipulated in new and unique ways. A custom network client written in Java can extract and display the exact piece of information the user wants. For example, an indexing program might extract only the actual text of a page while filtering out the HTML tags and navigation links. Of course, not every file downloaded from a web server has to be loaded into a browser window, or even has to be HTML. Custom network clients can process any data format the server sends, whether it's tab-separated text, a special purpose binary format for data acquired from scientific instruments, XML, or something else. Nor is a custom client limited to one server or document at a time.

Of course, not everything transmitted over HTTP is meant for humans. Web services allow machines to communicate with each other by exchanging XML documents over HTTP for purposes ranging from inventory management to stock trading to airline reservations. This can be completely automated with no human intervention, but it does require custom logic written in some programming language.

Java network clients are flexible because Java is a fully general programming language. Java programs see network connections as streams of data that can be interpreted and responded to in any way necessary. Web browsers see only certain kinds of data streams and can interpret them only in certain ways. If a browser sees a data stream that it's not familiar with (for example, a response to an SQL query), its behavior is unpredictable. Web sites can use server-side programs written in Java or other languages to provide some of these capabilities, but they're still limited to HTML for the user interface.

Writing Java programs that talk to Internet servers is easy. Java's core library includes classes for communicating with Internet hosts using the TCP and UDP protocols of the TCP/IP family. You just tell Java what IP address and port you want, and Java handles the low-level details. Java does not support NetWare IPX, Windows NetBEUI, AppleTalk, or other non-IP-based network protocols, but in the first decade of the new millennium, this is a non-issue. TCP/IP has become the lingua franca of networked applications and has effectively replaced pretty much all other general-purpose network protocols. A slightly more serious issue is that Java does not provide direct access to the IP layer below TCP and UDP, so it can't be used to write programs like ping or traceroute. However, these are fairly uncommon needs. Java certainly fills well over 90% of most network programmers' needs.

Once a program has connected to a server, the local program must understand the protocol the remote server speaks and properly interpret the data the server sends back. In almost all cases, packaging data to send to a server and unpacking the data received is harder than simply making the connection. Java includes classes that help your programs communicate with certain types of servers, most notably web servers. It also includes classes to process some kinds of data, such as text, GIF images, and JPEG images. However, not all servers are web servers, and not all data is text, GIF, or JPEG. As a result, Java lets you write protocol handlers to communicate with different kinds of servers and content handlers that understand and display different kinds of data. A web browser can automatically download and install the software needed by a web site it visits using Java WebStart and the Java Network Launching Protocol (JNLP). These applications can run under the control of a security manager that prevents them from doing anything potentially harmful without user permission.

Send Data

Web browsers are optimized for retrieving data: they send only limited amounts of data back to the server, mostly through forms. Java programs have no such limitations. Once a connection between two machines is established, Java programs can send data across the connection just as easily as they can receive from it. This opens up many possibilities.

File storage

Applets often need to save data between runs—for example, to store the level a player has reached in a game. Untrusted applets aren't allowed to write files on local disks, but they can store data on a cooperating server. The applet just opens a network connection to the host it came from and sends the data to it. The host may accept the data through HTTP POST, FTP, SOAP, or a custom server or servlet.

Massively parallel computing

There've always been problems that are too big for one computer to solve in a reasonable period of a time. Sometimes the answer to such a problem is buying a faster computer. However, once you reach the top of the line of off-the-shelf systems you can pick up at CompUSA, price begins to increase a lot faster than performance. For instance, one of the fastest personal computers you can buy at the time of this writing, an Apple PowerMac with two 2.5GHz processors, will set you back about $3,000 and provide speeds in the ballpark of a few gigaflops per second. If you need something a thousand times that fast, you can buy a Cray X1 supercomputer, which will cost you several tens of million dollars, or you can buy a thousand or so PowerMacs for only a few million dollars—roughly an order of magnitude less. The numbers change as the years go by. Doubtless you can buy a faster computer for less money today, but the general rule holds steady. Past a certain point, price goes up faster than performance.

At least since the advent of the cheap microcomputer a quarter of a century ago, programmers have been splitting problems across multiple, cheap systems rather than paying a lot more for the supercomputer of the day. This can be done informally by running little pieces of the problem on multiple systems and combining the output manually, or more formally in a system like Beowulf. There's some overhead involved in synchronizing the data between all the different systems in the grid, so the price still goes up faster than the performance, but not nearly as much faster as it does with a more traditional supercomputer. Indeed, cluster supercomputers normally cost about 10 times less than equally fast non-cluster supercomputers. That's why clusters are rapidly displacing the old style supercomputers. As of June 2004, just under 60% of the world's top 500 publicly acknowledged supercomputers were built from clusters of small, off-the-shelf PCs, including the world's third-fastest. There are probably a few more computers worthy of inclusion in the list hidden inside various government agencies with black budgets, but there's no reason to believe the general breakdown of architectures is different enough to skew the basic shape of the results.

When it comes to grid computing, Java is uniquely suited to the world of massively parallel clusters of small, off-the-shelf machines. Since Java is cross-platform, distributed programs can run on any available machine, rather than just all the Windows boxes, all the Solaris boxes, or all the PowerMacs. Since Java applets are secure, individual users can safely offer the use of their spare CPU cycles to scientific projects that require massively parallel machines. When part of the calculation is complete, the program makes a network connection to the originating host and adds its results to the collected data.

There are numerous ongoing efforts in this area. Among them is David Bucciarelli's work on JCGrid (http://jcgrid.sourceforge.net/), an open source virtual filesystem and grid-computing framework that enables projects to be divided among multiple worker machines. Clients submit computation requests to the server, which doles them out to the worker systems. What's unique about JCGrid compared to systems like Beowulf implemented in C is that the workers don't have to trust the server or the client. Java's security manager and byte code verifier can ensure the uploaded computation tasks don't do anything besides compute. This enables grids to be established that allow anyone to borrow the CPU cycles they need. These grids can be campus-wide, company-wide, or even worldwide on the public Internet. There is a lot of unused computing power wasting electricity for no reason at any given time of day on the world's desktops. Java networking enables researchers and other users to take advantage of this power even more cheaply than they could build a cluster of inexpensive machines.

Peer-to-Peer Interaction

The above examples all follow a client/server model. However, Java applications can also talk to each other across the Internet, opening up many new possibilities for group applications. Java applets can also talk to each other, though for security reasons they have to do it via an intermediary proxy program running on the server they were downloaded from. (Again, Java makes writing this proxy program relatively easy.)

Servers

Java applications can listen for network connections and respond to them, so it's possible to implement servers in Java. Both Sun and the W3C have written web servers in Java designed to be as fully functional and fast as servers written in C, such as the Apache HTTP server and Microsoft's Internet Information Server. Many other kinds of servers have been written in Java as well, including IRC servers, NFS servers, file servers, print servers, email servers, directory servers, domain name servers, FTP servers, TFTP servers, and more. In fact, pretty much any standard TCP or UDP server you can think of has probably been ported to Java.

More interestingly you can write custom servers that fill your specific needs. For example, you might write a server that stores state for your game applet and has exactly the functionality needed to let players save and restore their games, and no more. Or, since applets can normally only communicate with the host from which they were downloaded, a custom server could mediate between two or more applets that need to communicate for a networked game. Such a server could be very simple, perhaps just echoing what one applet sent to all other connected applets. WebCollab uses a custom server written in Java to collect annotations, notes, and slides from participants in the teleconference and distribute them to all other participants. It also stores the notes on the central server. It uses a combination of the normal HTTP and FTP protocols as well as its custom WebCollab protocol.

Along with classical servers that listen for and accept socket connections, Java provides several higher-level abstractions for client-server communication. Remote method invocation allows objects located on a server to have their methods called by

clients. Servers that support the Java Servlet API can load extensions written in Java called servlets that give them new capabilities. The easiest way to build a multiplayer game server might be to write a servlet rather than an entire server.

Searching the Web

Java programs can wander through the Web, looking for crucial information. Search programs that run on a single client system are called spiders. A spider downloads a page at a particular URL, extracts the URLs from the links on that page, downloads the pages referred to by the URLs, and repeats the process for each page it downloads. Generally, a spider does something with each page it sees, from indexing it in a database to performing linguistic analysis to hunting for specific information. This is more or less what services like Google do to build their indices. Building your own spider to search the Internet is a bad idea because Google and similar services have already done the work, and a few million private spiders would soon bring the Net to its knees. However, this doesn't mean you shouldn't write spiders to index your own local Intranet. In a company that uses the Web to store and access internal information, a local index service might be very useful. You can use Java to build a program that indexes all your local servers and interacts with another server program (or acts as its own server) to let users query the index.

The purposes of agents are similar to those of spiders (researching a stock, soliciting quotations for a purchase, bidding on similar items at multiple auctions, finding the lowest price for a CD, finding all links to a site, and so on), but whereas spiders run a single host system to which they download pages from remote sites, agents actually move themselves from host to host and execute their code on each system they move to. When they find what they're looking for, they return to the originating system with the information, possibly even a completed contract for goods or services. People have been talking about mobile agents for years, but until now, practical agent technology has been rather boring. It hasn't come close to achieving the possibilities envisioned in various science fiction novels. The primary reason for this is that agents have been restricted to running on a single system—and that's neither useful nor exciting. In fact, through 2003, the only successful agents have been hostile code such as the Morris Internet worm of 1989 and the numerous Microsoft Outlook vectored worms.

These cases demonstrate one reason developers haven't been willing to let agents go beyond a single host: they can be destructive. For instance, after breaking in to a system, the Morris worm proceeded to overload the system, rendering it useless. Letting agents run on a system introduces the possibility that hostile or buggy agents may damage that system, and that's a risk most network managers haven't been willing to take. Java mitigates the security problem by providing a controlled environment for the execution of agents that ensure that, unlike worms, the agents won't do anything nasty. This kind of control makes it safe for systems to open their doors to agents.

The second problem with agents has been portability. Agents aren't very interesting if they can only run on one kind of computer. It's sort of like having a credit card for Nieman-Marcus: a little bit useful and has a certain snob appeal, but it won't help as much as a Visa card if you want to buy something at Sears. Java provides a platform-

independent environment in which agents can run; the agent doesn't care if it's visiting a Sun workstation, a Macintosh, a Linux box, or a Windows PC.

An indexing program could be implemented in Java as a mobile agent: instead of downloading pages from servers to the client and building the index there, the agent could travel to each server and build the index locally, sending much less data across the network. Another kind of agent could move through a local network to inventory hardware, check software versions, update software, perform backups, and take care of other necessary tasks. A massively parallel computer could be implemented as a system that assigns small pieces of a problem to individual agents, which then search out idle machines on the network to carry out parts of the computation. The same security features that allow clients to run untrusted programs downloaded from a server lets servers run untrusted programs uploaded from a client.

4.4 Swings (JAVA)

Swing is the primary Java GUI widget toolkit. It is part of Sun Microsystems' Java Foundation Classes (JFC) — an API for providing a graphical user interface (GUI) for Java programs.

Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit. Swing provides a native look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform.

The Swing Architecture

Swing is a platform-independent, *Model-View-Controller* GUI framework for Java. It follows a single-threaded programming model, and possesses the following traits:

Foundations

Swing is platform independent both in terms of expression (Java) and implementation (Look-and-Feel).

Extensible

Swing is a highly partitioned architecture, which allows for the "plugging" of various custom implementations of specified framework interfaces: Users can provide their own custom implementation(s) of these components to override the default implementations. In general, Swing users can extend the framework by extending existing (framework) classes and/or providing alternative implementations of core components.

Swing is a component-based framework. The distinction between objects and components is a fairly subtle point: concisely, a component is a well-behaved object with a known/specified characteristic pattern of behaviour. Swing objects asynchronously fire events, have "bound" properties, and respond to a well-known set of commands (specific to the component.) Specifically, Swing components are Java

Beans components, compliant with the Java Beans Component Architecture specifications.

Customizable

Given the programmatic rendering model of the Swing framework, fine control over the details of rendering of a component is possible in Swing. As a general pattern, the visual representation of a Swing component is a composition of a standard set of elements, such as a "border", "inset", decorations, etc. Typically, users will programmatically customize a standard Swing component (such as a JTable) by assigning specific Borders, Colors, Backgrounds, opacities, etc., as the properties of that component. The core component will then use these properties (settings) to determine the appropriate renderers to use in painting its various aspects. However, it is also completely possible to create unique GUI controls with highly customized visual representation.

Configurable

Swing's heavy reliance on runtime mechanisms and indirect composition patterns allows it to respond at runtime to fundamental changes in its settings. For example, a Swing-based application can change its look and feel at runtime. Further, users can provide their own look and feel implementation, which allows for uniform changes in the look and feel of existing Swing applications without any programmatic change to the application code.

Lightweight UI

Swing's configurability is a result of a choice not to use the native host OS's GUI controls for displaying itself. Swing "paints" its controls programmatically through the use of Java 2D APIs, rather than calling into a native user interface toolkit. Thus, a Swing component does not have a corresponding native OS GUI component, and is free to render itself in any way that is possible with the underlying graphics APIs.

However, at its core every Swing component relies on an AWT container, since (Swing's) `JComponent` extends (AWT's) Container. This allows Swing to plug into the host OS's GUI management framework, including the crucial device/screen mappings and user interactions, such as key presses or mouse movements. Swing simply "transposes" its own (OS agnostic) semantics over the underlying (OS specific) components. So, for example, every Swing component paints its rendition on the graphic device in response to a call to component.paint(), which is defined in (AWT) Container. But unlike AWT components, which delegated the painting to their OS-native "heavyweight" widget, Swing components are responsible for their own rendering.

This transposition and decoupling is not merely visual, and extends to Swing's management and application of its own OS-independent semantics for events fired within its component containment hierarchies. Generally speaking, the Swing Architecture delegates the task of mapping the various flavors of OS GUI semantics onto a simple, but generalized, pattern to the AWT container. Building on that

generalized platform, it establishes its own rich and complex GUI semantics in the form of the `JComponent` model.

Loosely-Coupled and MVC

The Swing library makes heavy use of the Model/View/Controller software design pattern,which conceptually decouples the data being viewed from the user interface controls through which it is viewed. Because of this, most Swing components have associated *models* (which are specified in terms of Java interfaces), and the programmer can use various default implementations or provide their own. The framework provides default implementations of model interfaces for all of its concrete components. The typical use of the Swing framework does not require the creation of custom models, as the framework provides a set of default implementations that are transparently, by default, associated with the corresponding `JComponent` child class in the Swing library. In general, only complex components, such as tables, trees and sometimes lists, may require the custom model implementations around the application-specific data structures. To get a good sense of the potential that the Swing architecture makes possible, consider the hypothetical situation where custom models for tables and lists are wrappers over DAO and/or EJB services..

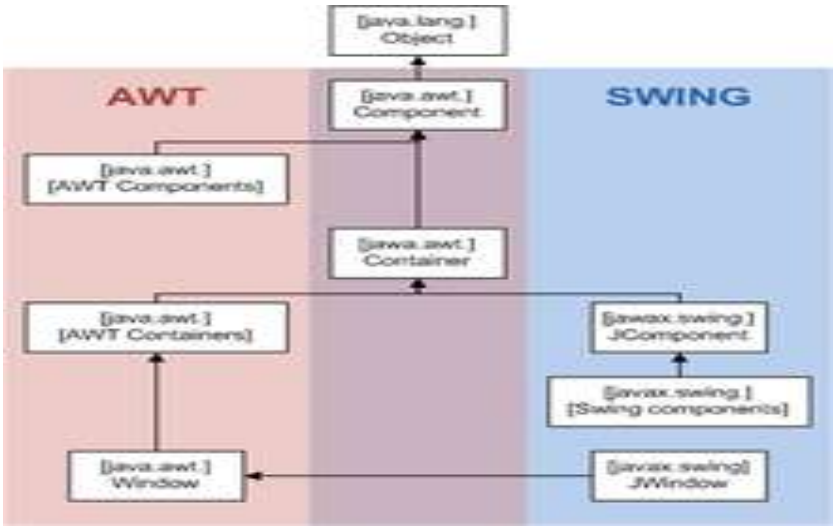
Typically, Swing component model objects are responsible for providing a concise interface defining events fired, and accessible properties for the (conceptual) data model for use by the associated JComponent. Given that the overall MVC pattern is a loosely-coupled collaborative object relationship pattern, the model provides the programmatic means for attaching event listeners to the data model object. Typically, these events are model centric (ex: a "row inserted" event in a table model) and are mapped by the JComponent specialization into a meaningful event for the GUI component.

For example, the `JTable` has a model called `TableModel` that describes an interface for how a table would access tabular data. A default implementation of this operates on a two-dimensional array.

The view component of a Swing JComponent is the object used to graphically "represent" the conceptual GUI control. A distinction of Swing, as a GUI framework, is in its reliance on programmatically-rendered GUI controls (as opposed to the use of the native host OS's GUI controls). Prior to Java 6 Update 10, this distinction was a source of complications when mixing AWT controls, which use native controls, with Swing controls in a GUI (see Mixing AWT and Swing components).

Finally, in terms of visual composition and management, Swing favors relative layouts (which specify the positional relationships between components) as opposed to absolute layouts (which specify the exact location and size of components). This bias towards "fluid"' visual ordering is due to its origins in the applet operating environment that framed the design and development of the original Java GUI toolkit. (Conceptually, this view of the layout management is quite similar to that which informs the rendering of HTML content in browsers, and addresses the same set of concerns that motivated the former.)

Relationship to AWT



AWT and Swing class hierarchy

Since early versions of Java, a portion of the Abstract Window Toolkit (AWT) has provided platform-independent APIs for user interface components. In AWT, each component is rendered and controlled by a native peer component specific to the underlying windowing system.

By contrast, Swing components are often described as *lightweight* because they do not require allocation of native resources in the operating system's windowing toolkit. The AWT components are referred to as *heavyweight components*.

Much of the Swing API is generally a complementary extension of the AWT rather than a direct replacement. In fact, every Swing lightweight interface ultimately exists within an AWT heavyweight component because all of the top-level components in Swing (`JApplet`, `JDialog`, `JFrame`, and `JWindow`) extend an AWT top-level container. Prior to Java 6 Update 10, the use of both lightweight and heavyweight components within the same window was generally discouraged due to Z-order incompatibilities. However, later version of Java have fixed these issues, and both Swing and AWT components can now be used in one GUI without Z-order issues.

The core rendering functionality used by Swing to draw its lightweight components is provided by Java 2D, another part of JFC.

Relationship to SWT

The Standard Widget Toolkit (SWT) is a competing toolkit originally developed by IBM and now maintained by the Eclipse community. SWT's implementation has more in common with the heavyweight components of AWT. This confers benefits such as more accurate fidelity with the underlying native windowing toolkit, at the cost of an increased exposure to the native platform in the programming model.

The advent of SWT has given rise to a great deal of division among Java desktop developers, with many strongly favouring either SWT or Swing. Sun's development on Swing continues to focus on platform look and feel (PLAF) fidelity with each

platform's windowing toolkit in the approaching Java SE 7 release (as of December 2006).

There has been significant debate and speculation about the performance of SWT versus Swing; some hinted that SWT's heavy dependence on JNI would make it slower when the GUI component and Java need to communicate data, but faster at rendering when the data model has been loaded into the GUI, but this has not been confirmed either way. A fairly thorough set of benchmarks in 2005 concluded that neither Swing nor SWT clearly outperformed the other in the general case.

SWT serves the Windows platform very well but is considered by some to be less effective as a technology for cross-platform development. By using the high-level features of each native windowing toolkit, SWT returns to the issues seen in the mid 1990s (with toolkits like zApp, Zinc, XVT and IBM/Smalltalk) where toolkits attempted to mask differences in focus behaviour, event triggering and graphical layout. Failure to match behavior on each platform can cause subtle but difficult-to-resolve bugs that impact user interaction and the appearance of the GUI.

4.5 Ms Access:

Microsoft Office Access, previously known as **Microsoft Access**, is a relational database management system from Microsoft that combines the relational Microsoft Jet Database Engine with a graphical user interface and software-development tools. It is a member of the Microsoft Office suite of applications, included in the Professional and higher editions or sold separately. In mid-May 2010, the current version of Microsoft Access 2010 was released by Microsoft in Office 2010; Microsoft Office Access 2007 was the prior version.

Access stores data in its own format based on the Access Jet Database Engine. It can also import or link directly to data stored in other applications and databases.[1]

Software developers and data architects can use Microsoft Access to develop application software, and "power users" can use it to build simple applications. Like other Office applications, Access is supported by Visual Basic for Applications, an object-oriented programming language that can reference a variety of objects including DAO (Data Access Objects), ActiveX Data Objects, and many other ActiveX components. Visual objects used in forms and reports expose their methods and properties in the VBA programming environment, and VBA code modules may declare and call Windows operating-system functions.

Uses

Microsoft Access is used to make databases.

When reviewing Microsoft Access in the real world, it should be understood how it is used with other products. An all-Access solution may have Microsoft Access Forms and Reports managing Microsoft Access tables. However, Microsoft Access may be used only as the 'front-end', using another product for the 'back-end' tables, such as Microsoft SQL Server and non-Microsoft products such as Oracle and Sybase. Similarly, some applications will only use the Microsoft Access tables and use

another product as a front-end, such as Visual Basic or ASP.NET. Microsoft Access may be only part of the solution in more complex applications, where it may be integrated with other technologies such as Microsoft Excel, Microsoft Outlook or ActiveX Data Objects.

Access tables support a variety of standard field types, indices, and referential integrity. Access also includes a query interface, forms to display and enter data, and reports for printing. The underlying Jet database, which contains these objects, is multiuser-aware and handles record-locking and referential integrity including cascading, updates and deletes.

Repetitive tasks can be automated through macros with point-and-click options. Microsoft Access is popular among non-programmers and professional developers alike. Non-programmers can create visually pleasing and relatively advanced solutions with very little or no code. It is also easy to place a database on a network and have multiple users share and update data without overwriting each other's work. Data is locked at the record level which is significantly different from Excel which locks the entire spreadsheet.

Microsoft offers a wide range of template databases within the program and for download from their website. These options are available upon starting Access and allow users to enhance a database with pre-defined tables, queries, forms, reports, and macros. Popular templates include tracking contacts, assets, issues, events, projects, and tasks. Templates do not include VBA code.

Microsoft Access also offers the ability for programmers to create solutions using the programming language Visual Basic for Applications (VBA), which is similar to Visual Basic 6.0 (VB6) and used throughout the Microsoft Office programs such as Excel, Word, Outlook and PowerPoint. Most VB6 code including the use of Windows API calls, can be used in VBA. Power users and developers can extend basic end-user solutions to a professional solution with advanced automation, data validation, error trapping, and multi-user support.

Database solutions created entirely in Microsoft Access are well suited for individual and workgroup use across a network. The number of simultaneous users that can be supported depends on the amount of data, the tasks being performed, level of use, and application design. Generally accepted limits are solutions with 1 GB or less of data (Access supports up to 2 GB) and performs quite well with 20 or fewer simultaneous connections (255 concurrent users are supported). This capability is often a good fit for department solutions. If using an Access database solution in a multi-user scenario, the application should be "split". This means that the tables are in one file called the back-end (typically stored on a shared network folder) and the application components (forms, reports, queries, code, macros, linked tables) are in another file called the front end[*disambiguation needed*]. The linked tables in the front end point to the back end file. Each user of the Access application would then receive their own copy of the front end file.

Applications that run complex queries or analysis across large datasets would naturally require greater bandwidth and memory. Microsoft Access is designed to scale to support more data and users by linking to multiple Access databases or using

a back-end database like Microsoft SQL Server. With the latter design, the amount of data and users can scale to enterprise-level solutions.

Microsoft Access' role in web development prior to version 2010 is limited. User interface features of Access, such as forms and reports, only work in Windows. In versions 2000 through 2003 an Access object type called Data Access Pages created publishable web pages. Data Access Pages are no longer supported. The Microsoft Jet Database Engine, core to Access, can be accessed through technologies such as ODBC or OLE DB. The data (i.e., tables and queries) can be accessed by web-based applications developed in ASP.NET, PHP, or Java.

Access 2010 allows databases to be published to SharePoint 2010 web sites running Access Services. These web-based forms and reports run in any modern web browser. The resulting web forms and reports, when accessed via a web browser, don't require any add-ins or extensions (e.g. ActiveX, Silverlight.

In enterprise environments, Microsoft Access is particularly appropriate for meeting end-user database needs and for rapid application development. Microsoft Access is easy enough for end users to create their own queries, forms and reports, laying out fields and groupings, setting formats, etc. This capability allows professional developers, as well as end users, to develop a wide range of applications to fulfill the needs of an organization or commercial purpose. Many technology departments enjoy Access's ease of use, thus allowing departmental users the ability to create highly focused applications, while allowing the technology departments to focus on the enterprise level systems that provide the information (enterprize data) to supported departments.

A compiled MDE or ACCDE version of an Access database can be created to prevent users from getting to the design surfaces to modify module code, forms, and reports. This is often used in environments where end-user modifications are discouraged or the application's code should be kept private.

Microsoft offers a runtime version of Microsoft Access 2007 for download. This allows people to create Access solutions and distribute it for use by non-Microsoft Access owners (similar to the way DLLs or EXEs are distributed). Unlike the regular version of Access, the runtime version allows users to use the Access application but they cannot use its design surfaces.

Microsoft also offers developer extensions for download to help distribute Access applications, create database templates, and integrate source code control with Microsoft Visual SourceSafe.

Features

Users can create tables, queries, forms and reports, and connect them together with macros. Advanced users can use VBA to write rich solutions with advanced data manipulation and user control.

The original concept of Access was for end users to be able to "access" data from any source. Other uses include: the import and export of data to many formats including

Excel, Outlook, ASCII, dBase, Paradox, FoxPro, SQL Server, Oracle, ODBC, etc. It also has the ability to link to data in its existing location and use it for viewing, querying, editing, and reporting. This allows the existing data to change and the Access platform to always use the latest data. It can perform heterogeneous joins between data sets stored across different platforms. Access is often used by people downloading data from enterprise level databases for manipulation, analysis, and reporting locally.

There is also the Jet Database format (MDB or ACCDB in Access 2007) which can contain the application and data in one file. This makes it very convenient to distribute the entire application to another user, who can run it in disconnected environments.

One of the benefits of Access from a programmer's perspective is its relative compatibility with SQL (structured query language) — queries can be viewed graphically or edited as SQL statements, and SQL statements can be used directly in Macros and VBA Modules to manipulate Access tables. Users can mix and use both VBA and "Macros" for programming forms and logic and offers object-oriented possibilities. VBA can also be included in queries.

Microsoft Access offers parameterized queries. These queries and Access tables can be referenced from other programs like VB6 and .NET through DAO or ADO. From Microsoft Access, VBA can reference parameterized stored procedures via ADO.

The desktop editions of Microsoft SQL Server can be used with Access as an alternative to the Jet Database Engine. This support started with MSDE (Microsoft SQL Server Desktop Engine), a scaled down version of Microsoft SQL Server 2000, and continues with the SQL Server Express versions of SQL Server 2005 and 2008.

Microsoft Access is a file server-based database. Unlike client–server relational database management systems (RDBMS), Microsoft Access does not implement database triggers, stored procedures, or transaction logging. Access 2010 includes table-level triggers and stored procedures built into the ACE data engine. Thus a Client-server database system is not a requirement for using stored procedures or table triggers with Access 2010. Tables, queries, Forms, reports and Macros can now be developed specifically for web base application in Access 2010. Integration with Microsoft SharePoint 2010 is also highly improved.

SNAPSHOTS OF WORKING APPLICATION

Fig. 1: Control Panel

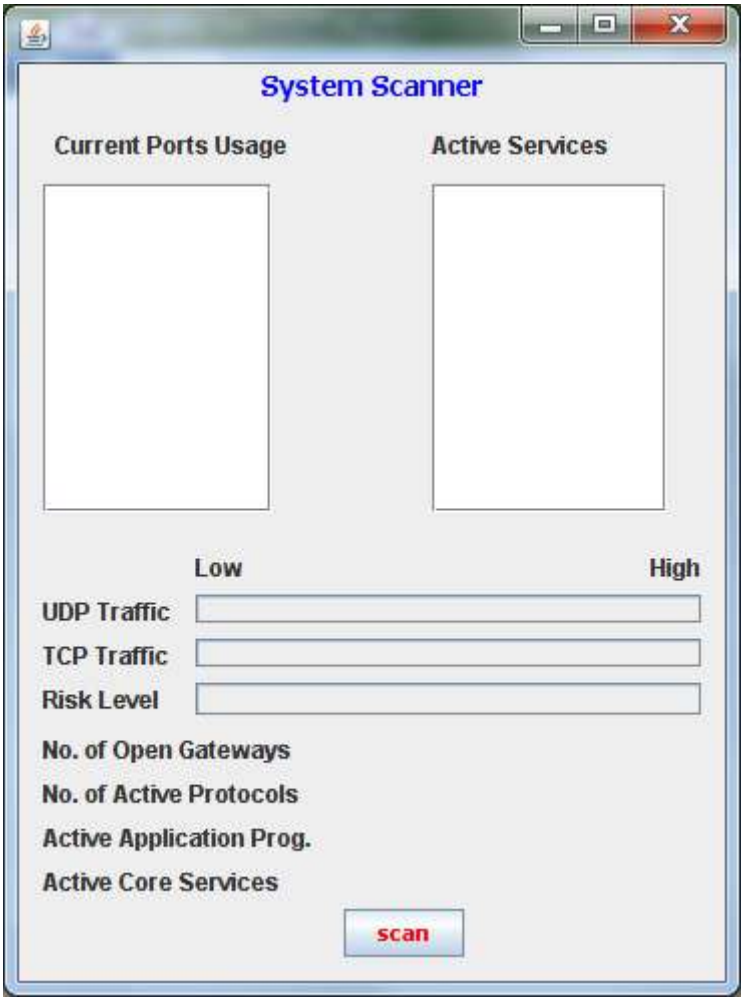


Fig.2: Blank Scanner Module Window

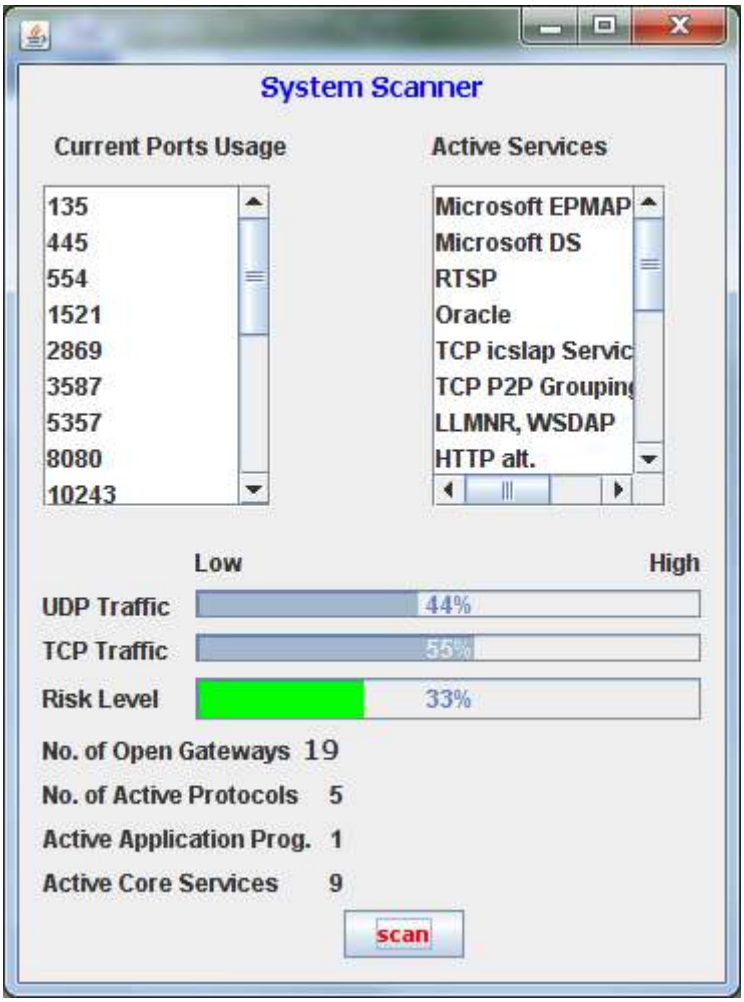


Fig. 3: Scanner Detecting Low Risk and other System Info.

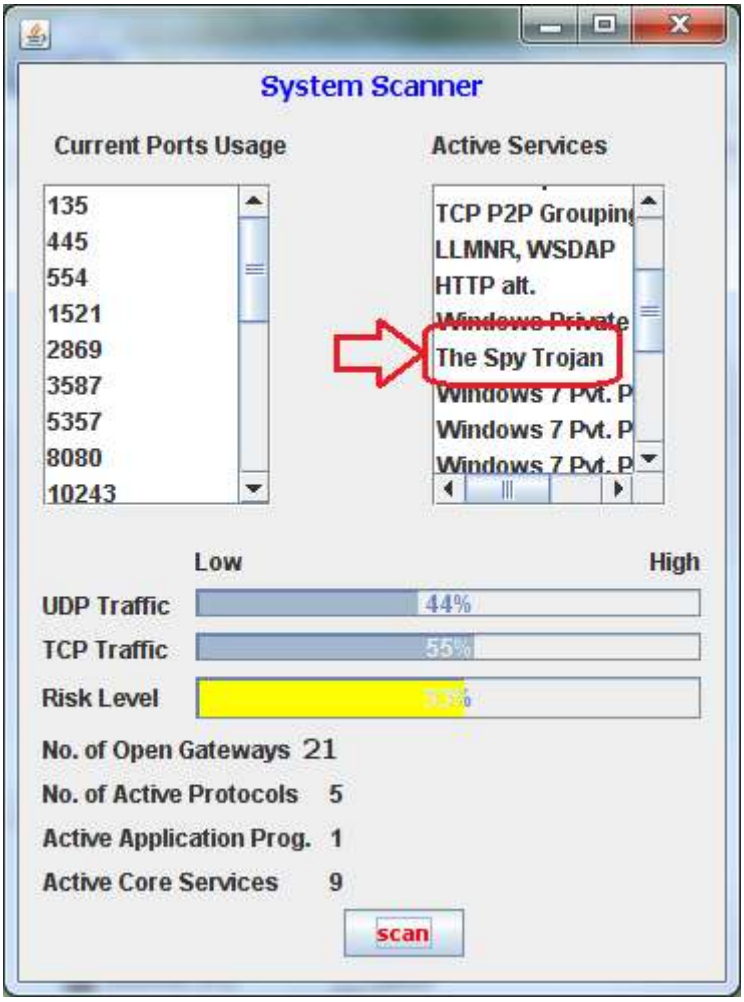


Fig. 4: Scanner Module on Medium Risk and Detected Trojan

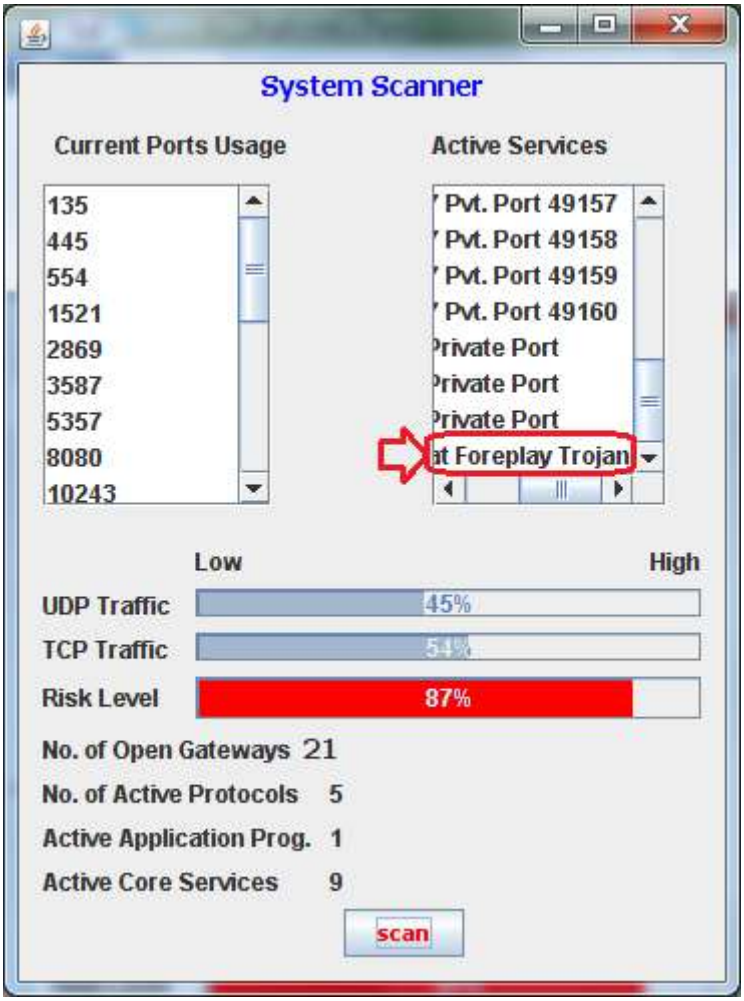


Fig. 5 : Scanner on Displaying High Risk and detected Trojan

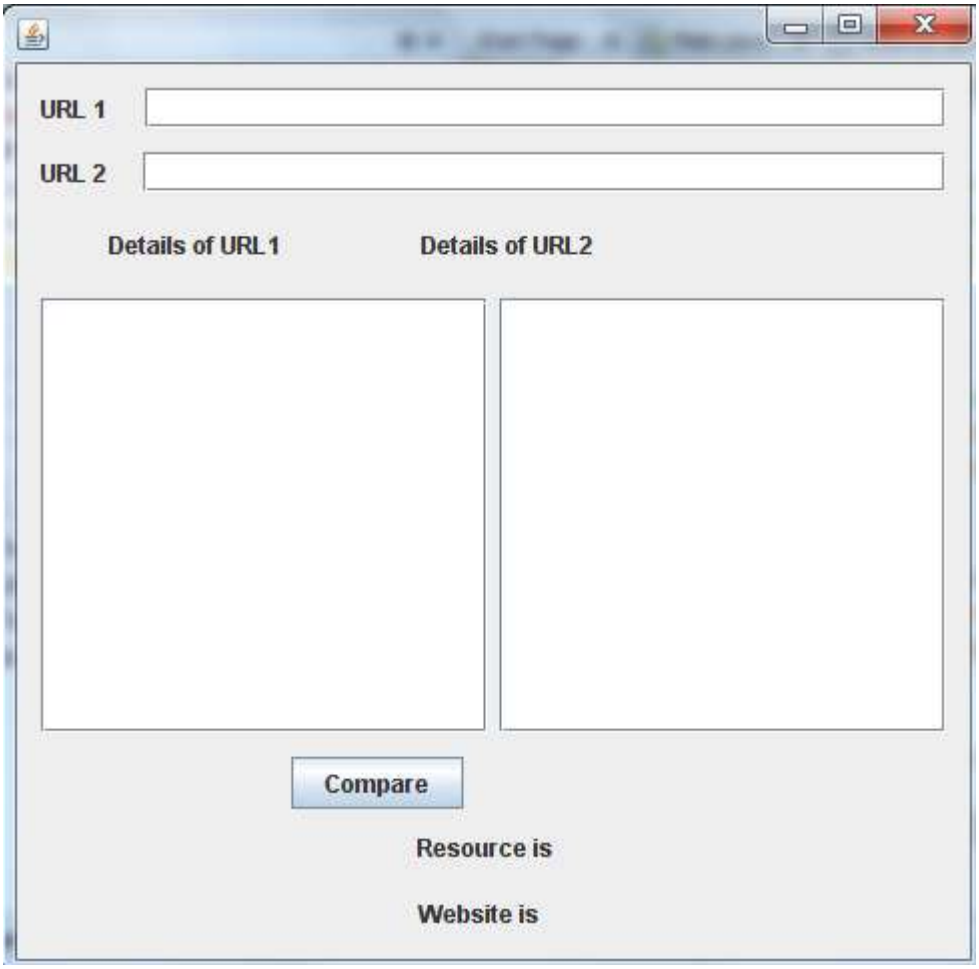


Fig. 6: Blank Anti-Phishing Window

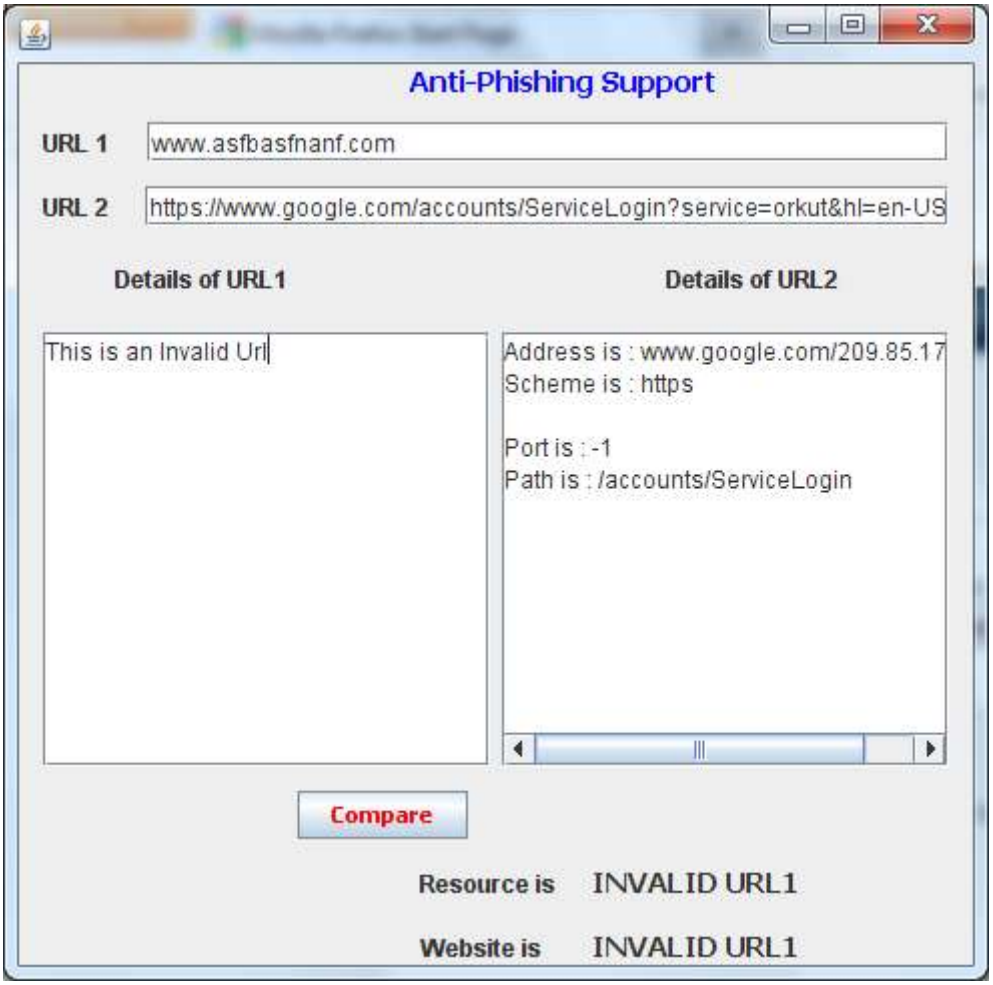


Fig. 7 :Anti-Phishing detecting a wrong input

Fig.8: Anti-Phishing detecting same website but different resources

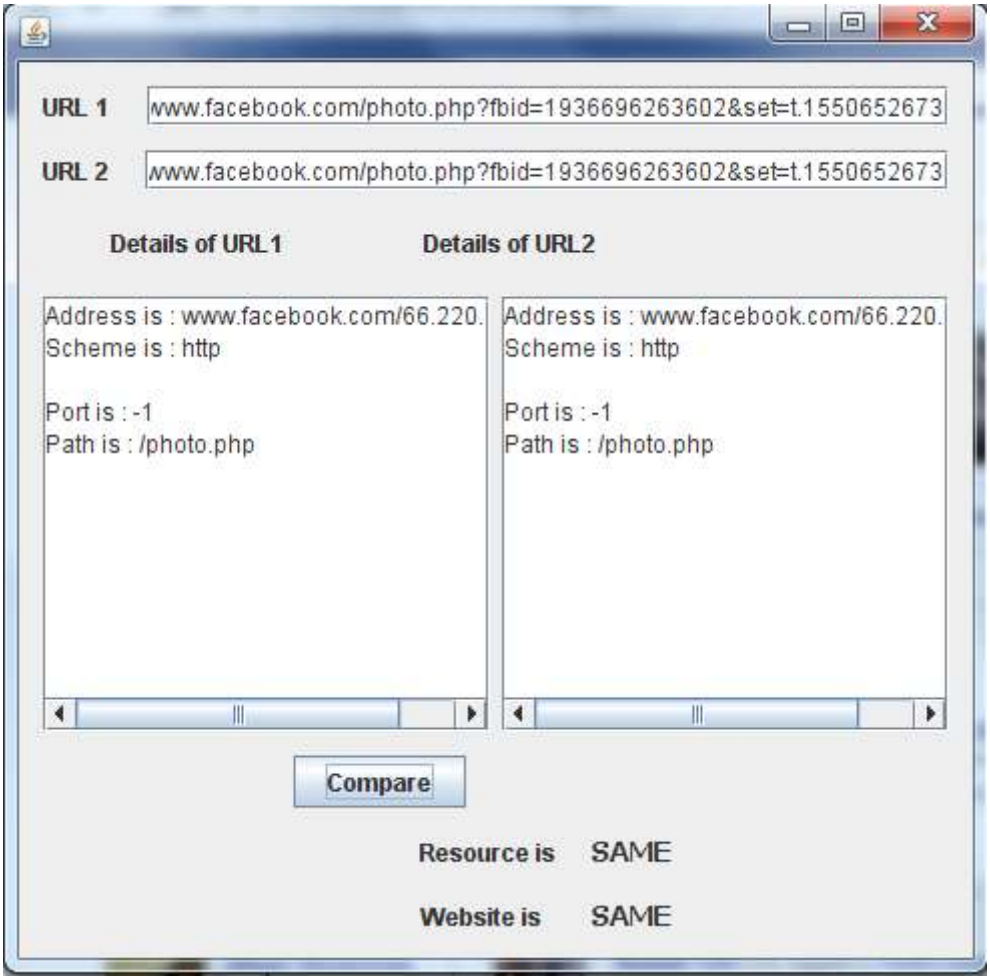


Fig. 9: Anti-Phishing Detecting same website and resources.

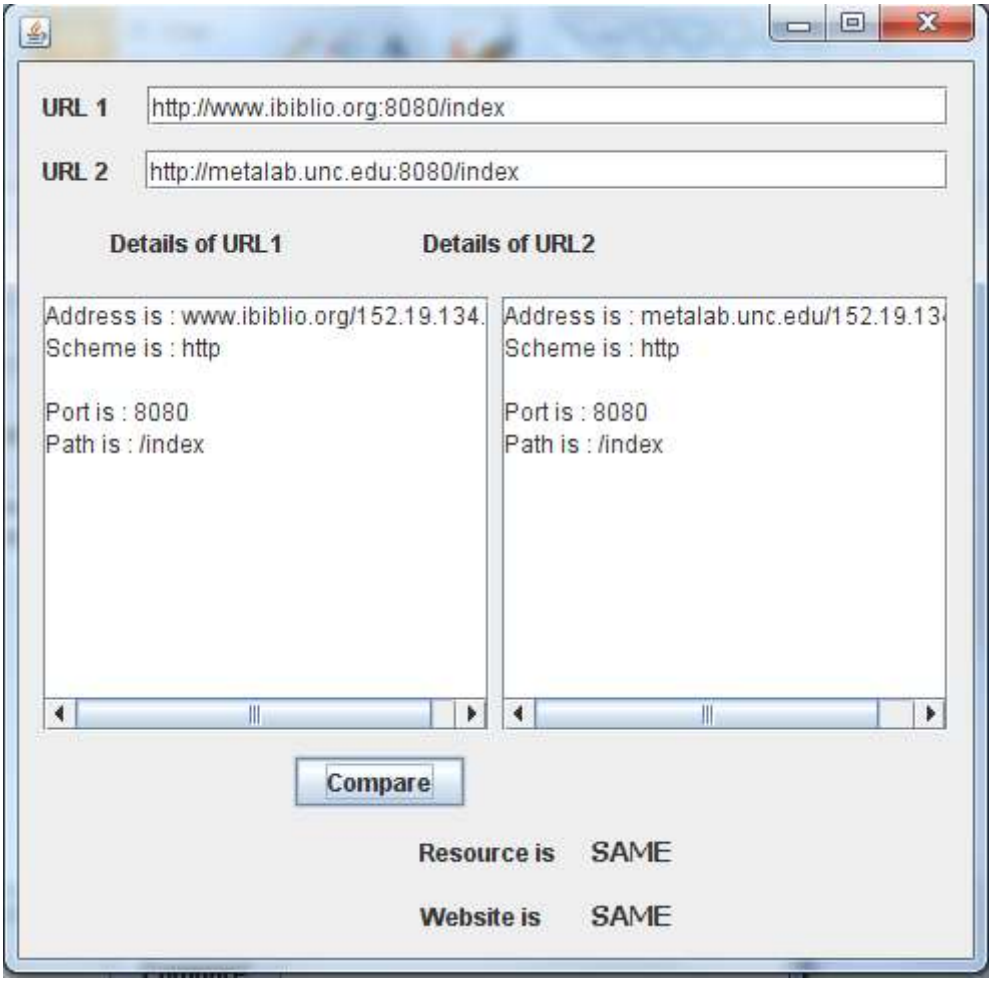


Fig. 10: Anti-Phishing Detecting same resource and website although names are different.

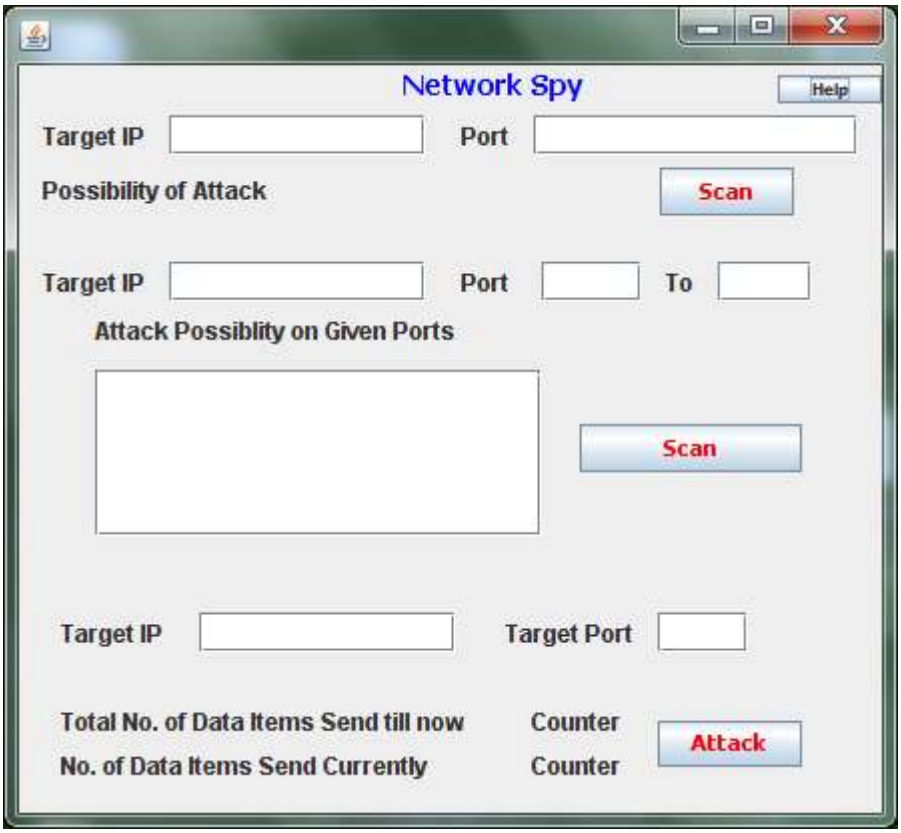


Fig. 11: Blank Network Spy Module

Fig. 12: Help Window of Network Spy

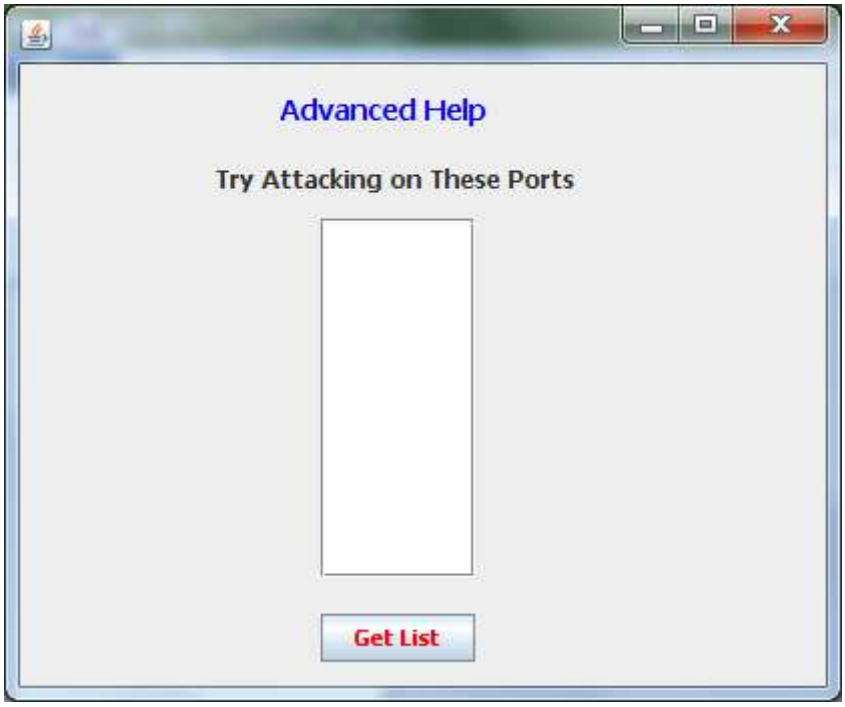


Fig. 13: Blank Advanced Help of Network Spy

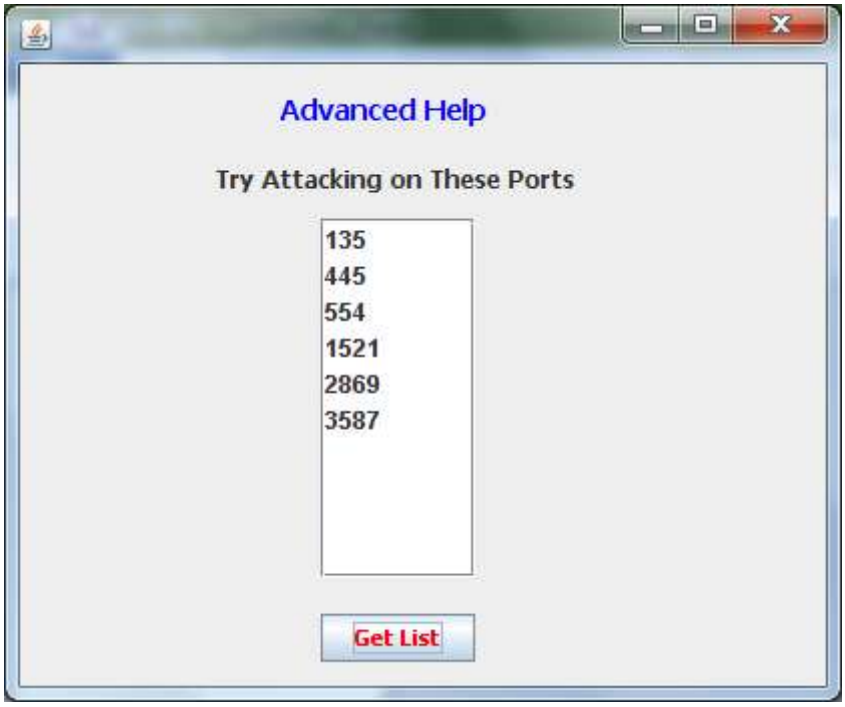


Fig.14 : Working Advanced help window of Network Spy

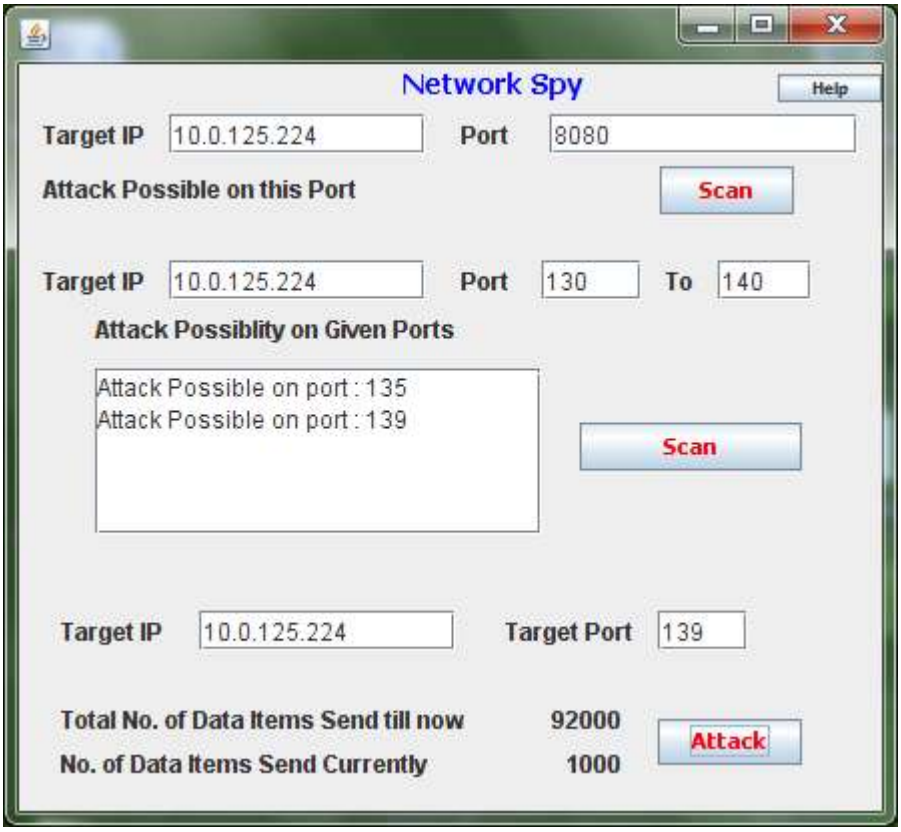


Fig. 15: Working Network Spy Module

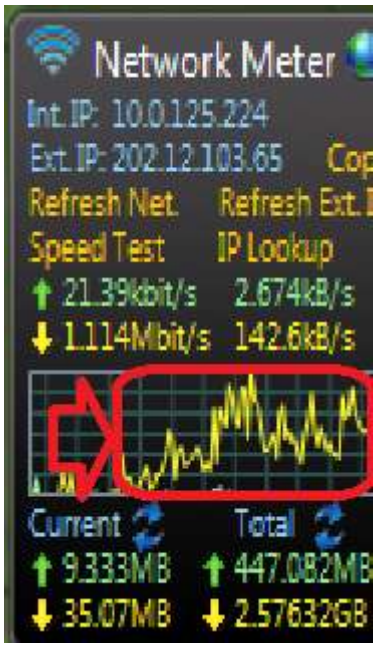


Fig. 15.1 : Increase in Network traffic When Attack was Done.

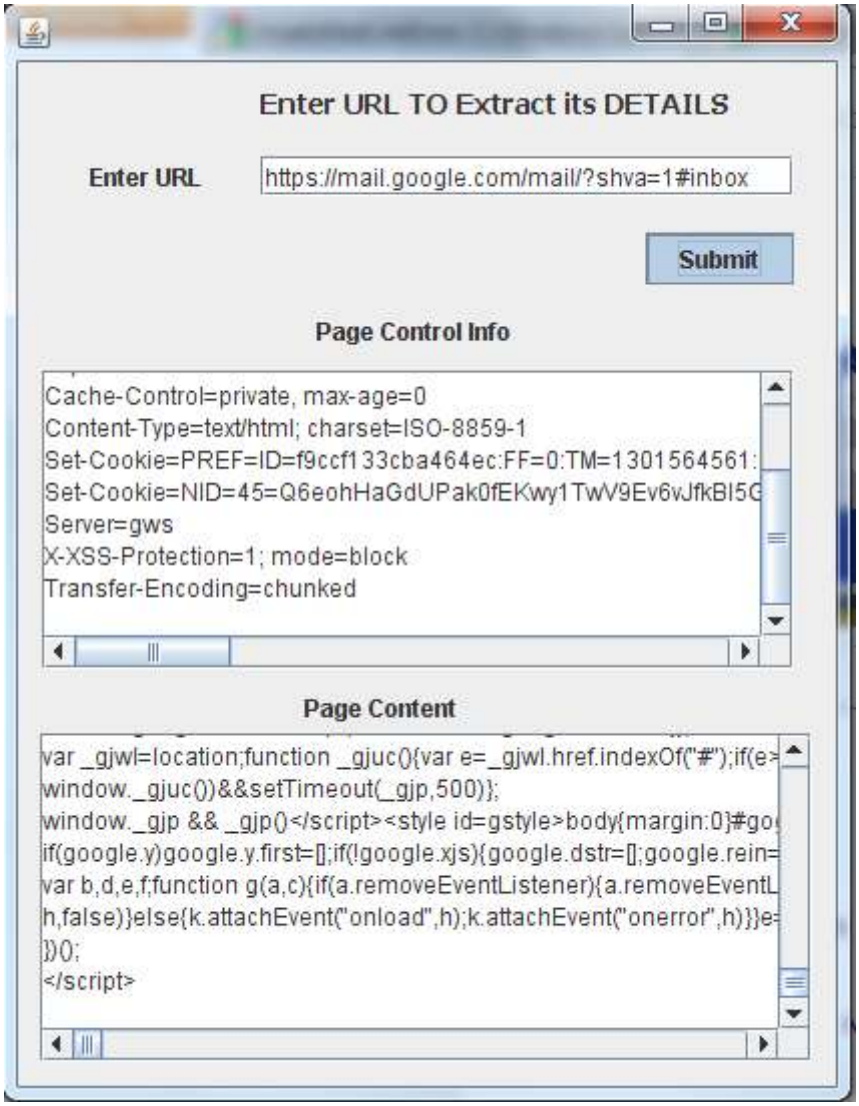


Fig. 16 : Working Web Page Scanner module

RESULTS & DISCUSSION

6.1 Results

I have successfully developed a Net Secure Application which can cater all the needs of its user related to their network monitoring and security issues. I have used the latest and most efficient technologies in developing this Net Secure Application. These technologies range from Java Network Programming and recent versions of Microsoft Access etc.

I have highly focussed myself during the complete development phase to follow the industry guidelines for the coding and nomenclature of my modules and components. I have taken every effort to minimise the redundant code. I have also minimised the number of lines of code & have used Multi-Threading as to optimize the speed of the Net Secure Application. The heavier (in terms of size) the Net Secure Application is the more time it takes to load so I minimised the connections to database and also retrieval of records, such that the performance of the application is improved much further.

When I compare this work with that of other existing Network monitoring applications or from the work done by other programmers in the same field I find dramatic difference (positive) in the functionality of the application I have developed.

I tried my best to deliver a full working Net Secure Application. Some of the features which are delivered by me are:

1.) User can scan and **get risk level** to which his/her is exposed to on internet and network.
2.) User can **check UDP & TCP Traffic** on his machine.
3.) User can **get a list of active services** on his machine which are using internet or network resources.
4.) User can **get a list of active port numbers** on your system.
5.) User can **get number of inbound/outbound gateways** open on your machine.
6.) User can also **get number of Active Protocols, Active Services and Active Application Programs** which uses network resources.
7.) User can **detect the presence of 200+ Trojans and Viruses.**
8.) User can **scan for the active windows private ports.**

9.) **Anti-Phishing** module of this application **compares two URLs to identify** whether **they point to same resource or not** and whether these resources are located on a same website or different.

10.) Anti-Phishing module also **tells the protocol active** on a particular URL ,the **port number** on which that resource is located ,and **exact address** of that resource.

11.) **Network-spy module** of this application is a platform developed to **implement simple network attacks** on a particular IP address and port number.

12.) With the help of Network-spy module user can scan a particular IP address to get a list of open connection ports on which attack is possible.

13.) Users are also provided with **advanced help feature** in Network-Spy module which **suggests user with ports which are vulnerable** to attack.

14.) The **Web Page scanner module** of my application scans a web page to **extract all its control information and the complete coding** behind any web page.

15.) Appropriate **Multi-Threaded** programs are used wherever possible.

16.) All these data is **represented with statistics** like progress bar, percentage displays and appropriate colour change.

FUTURE IMPLICATIONS

7.1 Future Implication:

This application is build keeping in mind the possibilities of future upgrades. This application is easily modifiable to support new features and upgrades.

Features to be added(being developed) :

- Multicast Packet Sniffer.
- Object Downloader and Scanner.
- Connection Blocking.

References

- Complete Java Reference by Herbert Schildt, TMH publication
- Network Programming by Harold, Oreilley Publication
- Advanced Java Networking by Prashant Sridharan.
- Java Security by Scott Oaks, Pearson Education.
- http://en.wikipedia.org/wiki/Microsoft_Access
- www.cs.usfca.edu
- *en.wikipedia.org/wiki/**Network_programming***
- www.silicontao.com/**Programming**Guide/other/beejnet.pdf
- *www.**packetattack**.com/tutorials.html*