# **A SECURE INTRUSION DETECTION SYSTEM AGAINST DDoS ATTACK IN WIRELESS MOBILE AD-HOC NETWORK**

# 

# **A PROJECT REPORT**

*Submitted by*

JEROME S REG NO: 2020104054

HARISH KUMAR M REG NO: 2020104045

DHARUNPANDI M REG NO: 2020104032

*In partial fulfilment for the award of the degree of*

BACHELOR OF ENGINEERING

*in*

# **ELECTRONICS AND COMMUNICATION ENGINEERING**



SETHU INSTITUTE OF TECHNOLOGY, KARIAPATTI ANNA UNIVERSITY: CHENNAI 600 025

MARCH 2024

BONAFIDE CERTIFICATE

**ANNA UNIVERSITY: CHENNAI 600 025**

BONAFIDE CERTIFICATE

# Certified that this project report **“A SECURE INTRUSION DETECTION SYSTEM AGAINST DDOS ATTACK IN WIRELESS MOBILE AD-HOC NETWORK”** is the bonafide work of **“JEROME R (2020104054), HARISH KUMAR M (2020104045), DHARUNPANDI M (2020104032),”** who carried out the project work under my supervision.

|  |  |
| --- | --- |
| **SIGNATURE**  **Dr.M. PARISA BEHAM, M.E., Ph.D.,**  **Professor & Head of the Department,**  Department of ECE,  Sethu Institute of Technology, Pulloor,  Kariapatti, Virudhunagar-626115. | **SIGNATURE**  **R. ABDUL SIKKANDHAR, M.E.,(Ph.D), ASSISTANT PROFESSOR/ECE**  **SUPERVISOR**.  Department of ECE,  Sethu Institute of Technology, Pulloor, Kariapatti, Virudhunagar-626115. |

Submitted for the Project work (19UEC801) VIVA-VOICE Examination held

at Sethu Institute of Technology on .

INTERNAL EXAMINER EXTERNALEXAMINER

# **ACKNOWLEDGEMENT**

We would like to express my grateful thanks to Almighty God for giving the strength and the wisdom for completing this project.

We would first like to express my earnest gratefulness to our beloved Founder & Chairman Mr. S.Mohamed Jaleel, B.sc., B.L., for his encouragement extended to me to undertake this project work.

We wish to express my sincere thanks to our Chief Executive Officer Mr.S.M.Seeni Mohaideen , Joint Chief Executive Officer Mr.S.M.Seeni Mohammad Aliar Maraikkayar, Director Administration Ms.S.M.Nilofer Fathima and Director R & D Dr.S.M. Nazia fathima for providing support for the completion of the project.

We express my sincere thanks to our beloved Principal Dr. G.D. Siva Kumar, M.E., Ph.D, for providing all necessary support for the completion of the Project.

We wish to extend my gratitude and grateful thanks to our Dean & Head of the department Dr.M. Parisa Beham, M.E., Ph.D., Professor for her excellent guidance to complete my project proficiently. We wish to express my sincere thanks to our Professor Dr.R.TamilSelvi, M.Tech., Ph.D, Dean/BME/BT/Chemical for providing all necessary support for the completion of the Project.

We wish to express my sincere thanks to my Supervisor R. Abdul Sikkandhar, M.E., (Ph.D), Assistant Professor, for his admirable guidance and suggestions throughout this project work. Last but not the least, I thank and feel a deep sense of gratitude to my Parents, Faculty Members, Supporting Staff and Friends who had been all along encouraging me to endeavor this project.

|  |  |  |
| --- | --- | --- |
| S. No. | TITLE OF CONTENTs | PAGE No. |
|  | ABSTRACT | I |
|  | LIST OF FIGURES | II |
|  | LIST OF TABLES | III |
|  | LIST OF ABBREVIATION | IV |
| 1. | INTRODUCTION |  |
|  | 1.1. Synopsis | 01 |
|  | 1.2. About the Project | 02 |
| 2. | SYSTEM ANALYSIS |  |
|  | 2.1. Existing System | 03 |
|  | 2.1.1 Issues in Existing System | 03 |
|  | 2.2. Proposed System | 04 |
|  | 2.2.1. Advantages of Proposed System | 04 |
| 3. | SYSTEM ENVIRONMENT |  |
|  | 3.1. Hardware Requirements | 05 |
|  | 3.2. Software Requirements | 05 |
|  | 3.3. Software Specification | 06 |
| 4. | SYSTEM DESIGN |  |
|  | 4.1. System Flow Diagram | 11 |
|  | 4.2. Data Flow Diagram | 12 |
|  | 4.3. Overall Process Diagram | 17 |
|  | 4.4. Input Design and Output Design | 18 |
|  | 4.5. Database Design | 20 |
| 5. | SYSTEM DESCRIPTION |  |
|  | 5.1. Modules Description | 21 |
|  | 5.2. Introduction To SQL Server | 22 |
| 6. | IMPLEMENTATION | 23 |
| 7. | TESTING |  |
|  | 7.1 Unit Testing | 25 |
|  | 7.2 System Testing | 25 |
|  | 7.3 Integrated Testing | 26 |
| 8. | BACKUP AND MAINTENANCE | 27 |
| 9. | RESULT & DISCUSSION |  |
|  | Results | 28 |
|  | Sample Code | 37 |
| 10. | CONCLUSION | 57 |
| 11. | FUTURE ENHANCEMENT | 58 |
|  | REFERENCE | 59 |
|  | BIBLIOGRAPHY | 60 |

**ABSTRACT**

Wireless Mobile ad hoc network is an emerging technology and have great strength to be applied in critical situations like battlefields and commercial applications such as building, traffic surveillance, MANET is infrastructure less, with no any centralized controller exist and also, each node contains routing capability, each device in a MANET is independently free to move in any direction, and will therefore change its connections to other devices frequently So one of the major challenges wireless mobile ad hoc networks face today is security, because no central controller exists. MANETs are a kind of wireless ad hoc networks that usually has a routable networking environment on top of a link layer ad hoc network Ad hoc also contains wireless sensor network so the problems are facing by sensor network is also faced by MANET While developing the sensor nodes in unattended environment increases the chances of various attacks There are many security attacks in MANET and DDoS (Distributed denial of service) is one of them Our main aim is seeing the effect of DDoS in routing load, packet drop rate, end to end delay i.e., Maximizing due to attack on network and with these parameters and many more also, we build secure Intrusion Detection System to detect this kind of attack and block it Using Rate Limiting and Traffic Filtering Algorithm which Limits the rate of incoming traffic and filtering packets based on predefined criteria can help prevent DDoS attacks from overwhelming the network.

*Keywords: Wireless Mobile ad hoc network (MANET), Distribution denial of service (DDOS), Intrusion detection system (IDS).*

LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| No. | TITLE OF FIGUREs | PAGE No. |
| 1. | System Flow Diagram | 11 |
| 2. | Login Form Diagram | 12 |
| 3. | Set Node Status Diagram | 13 |
| 4. | Connect Receiver Diagram | 14 |
| 5. | Select File Diagram | 15 |
| 6. | Router Process Diagram | 16 |
| 7. | Over All Process Diagram | 17 |
| 8. | Login Form | 29 |
| 9. | View Router | 29 |
| 10. | Connect Receiver A | 30 |
| 11. | Node Status | 30 |
| 12. | Select File Name | 31 |
| 13. | Select Router | 31 |
| 14. | Select Destination Node | 32 |
| 15. | Enter IP Address | 32 |
| 16. | Successfully Send Your Data | 33 |
| 17. | Receiver A Receive Files | 33 |
| 18. | DDOS Attack Found | 34 |
| 20. | Location Discloser Attacker Found | 34 |
| 23. | IDRS Master | 35 |

LIST OF TABLES

|  |  |  |
| --- | --- | --- |
| No. | TITLE OF TABLES | PAGE No. |
| 1. | Default routing table | 20 |
| 2. | Receiver | 20 |
| 3. | Routing table | 20 |

LIST OF ABBREVIATION

|  |  |  |
| --- | --- | --- |
| No. | TITLE OF ABBREVIATION | MEANING |
| 1. | DDoS | Distributed denial of service |
| 2. | MANET | Wireless Mobile ad-hoc network |
| 3. | IDS | Intrusion Detection System |
| 4. | CTS | Common Type System |
| 5. | DNS | Domain Name Spacing |
| 6. | CLR | Common Language Runtime |
| 7. | CLS | Common Language Specification |
| 8. | AODV | Ad-Hoc On-Demand Distance Vector |
| 9. | SQL | Structured Query Language |
| 10. | XML | Extensible Markup Language |
| 11. | DBMS | Database Management System |

CHAPTER 1

1. INTRODUCTION

1.1. SYNOPSIS

Malware is pervasive in networks, and poses a critical threat to network security. However, we have very limited understanding of malware behavior in networks to date. In this paper, we investigate how malware propagate in networks from a global perspective. We formulate the problem, and establish a rigorous two-layer epidemic model for malware propagation from network to network. Based on the proposed model, our analysis indicates that the distribution of a given malware follows exponential distribution, power law distribution with a short exponential tail, and power law distribution at its early, late and final stages, respectively. Extensive experiments have been performed through two real-world global scale malware data sets, and the results confirm our theoretical findings.

1.2. ABOUT THE PROJECT

Malware are malicious software programs deployed by cyber attackers to compromise computer systems by exploiting their security vulnerabilities. Motivated by extraordinary financial or political rewards, malware owners are exhausting their energy to compromise as many networked computers as they can in order to achieve their malicious goals. A compromised computer is called a bot, and all bots compromised by a malware form a botnet. Botnets have become the attack engine of cyber attackers, and they pose critical challenges to cyber defenders. In order to fight against cyber criminals, it is important for defenders to understand malware behavior, such as propagation or membership recruitment patterns, the size of botnets, and distribution of bots. To date, we do not have a solid understanding about the size and distribution of malware or botnets. Researchers have employed various methods to measure the size of botnets, such as botnet infiltration, DNS redirection, and external information.

These efforts indicate that the size of botnets varies from millions to a few thousand. There are no dominant principles to explain these variations. As a result, researchers desperately desire effective models and explanations for the chaos. Dagon, Zou and Lee revealed that time zone has an obvious impact on the number of available bots. Mieghem et al indicated that network topology has an important impact on malware spreading through their rigorous mathematical analysis. Recently, the emergence of mobile malware, such as Cabir, Ikee, and Brador, further increases the difficulty level of our understanding on how they propagate. More details about mobile malware can be found at a recent survey paper. To the best of our knowledge, the best finding about malware distribution in large-scale networks comes from Chen and Ji: the distribution is non-uniform. All this indicates that the research in this field is in its early stage. The epidemic theory plays a leading role in malware propagation modeling. The current models for malware spread fall in two categories: the epidemiology model and the control theoretic model. The control system theory based models try to detect and contain the spread of malware. The epidemiology models are more focused on the number of compromised hosts and their distributions, and they have been explored extensively in the computer science community

CHAPTER 2

2. SYSTEM ANALYSIS

2.1. EXISTING SYSTEM

In existing system, Mobile ad-hoc networks devices or nodes or terminals with a capability of wireless communications and networking which makes them able to communicate with each other without the aid of any centralized system. This is an autonomous system in which nodes are connected by wireless links and send data to each other As we know that there is no any centralized system so routing is done by node itself. Due to its mobility and self routing capability nature, there are many weaknesses in its security. One of the serious attacks to be considered in ad hoc network is DDoS attack A DDoS attack is launched by sending huge amount of packets to the target node through the co-ordination of large amount of hosts which are distributed all over in the network. At the victim side this large traffic consumes the bandwidth and not allows any other important packet reached to the victim

2.1.1. ISSUES EXISTING SYSTEM

* Only little work has been done on the detection of App- DDoS attacks because there were few such attacks in the past.
* App-DDoS attacks are that the application- layer requests originating from the compromised hosts are indistinguishable from those generated by legitimate users.
* The implied premise of most current detection schemes is that the characteristics of DDoS attack traffic differ from normal traffic, which might fail because App-DDoS attacks may mimic the access behaviors of normal users.

2.2. PROPOSED SYSTEM

In proposed system, to solve the security issues we need an intrusion detection system. This can be categorized into two models:

1. Signature-based intrusion detection

Optimize and enhance key parameters of core devices and systems to improve resiliency in the case of a DDoS attack. This method works only in the event of a low-traffic DDoS attack, a large attack

(Either in bandwidth or packets per second) will quickly overwhelm these defenses.

2.Anomaly-based intrusion detection

The benefits of this IDS technique are that it can be able to detect attack without prior knowledge of attack. Intrusion attack is very easy in wireless network as compare to wired network. One of the serious attacks to be considered in ad hoc network is DDoS attack. Proposed system is beneficial since it turned into information system analyzing the traffic that will meet the organizations operating requirements. IN security, the file is transferred to the destination and the acknowledgement is given to the server. Bulk of data transfer is sent without traffic.

2.2.1. ADVANTAGES OF PROPOSED SYSTEM

* Security/encapsulation.
* Distributed databases.
* Faster problem solving.
* Security through redundancy.

CHAPTER 3

3. SYSTEM ENVIRONMENT

3.1. HARDWARE REQUIREMENTS

The Below Hardware Requirements were used in both Server and Client machines

when developing.

Processor : Intel(R) Core(TM) i3

Processor Speed : 3.06 GHz

RAM : 2 GB

Hard Disk Drive : 500 GB

CD-ROM Drive : Sony

Monitor : “17” inches

Keyboard : TVS Gold

Mouse : Logitech

3.2. SOFTWARE REQUIREMENTS

Operating System : Windows 7/10/11

Technology Used : Microsoft C#.NET

Database : SQL Server

Database Connectivity : ActiveX Data Object (ADO)

3.3. SOFTWARE SPECIFICATION

FEATURE OF C#.NET

Microsoft .NET is a set of Microsoft software technologies for rapidly building and integrating XML Web services, Microsoft Windows-based applications, and Web solutions. The .NET Framework is a language-neutral platform for writing programs that can easily and securely interoperate. There’s no language barrier with .NET: there are numerous languages available to the developer including Managed C++, C#, Visual Basic and Java Script. The .NET framework provides the foundation for components to interact seamlessly, whether locally or remotely on different platforms. It standardizes common data types and communications protocols so that components created in different languages can easily interoperate. “.NET” is also the collective name given to various software components built upon the .NET platform. These will be both products (Visual Studio.NET and Windows.NET Server, for instance) and services (like Passport, .NET My Services).

The .Net Framework

The .NET Framework has two main parts:

1. The Common Language Runtime (CLR).

2. A hierarchical set of class libraries.

The CLR is described as the “execution engine” of .NET. It provides the environment within which programs run.

The most important features are

* Conversion from a low-level assembler-style language, called Intermediate Language (IL), into code native to the platform being executed on.
* Memory management, notably including garbage collection.
* Checking and enforcing security restrictions on the running code.
* Loading and executing programs, with version control and other such features.
* The following features of the .NET framework are also worth description

Managed Code

The code that targets .NET, and which contains certain extra Information - “metadata” - to describe itself. Whilst both managed and unmanaged code can run in the runtime, only managed code contains the information that allows the CLR to guarantee, for instance, safe execution and interoperability.

Managed Data

With Managed Code comes Managed Data. CLR provides memory allocation and Deal location facilities, and garbage collection. Some .NET languages use Managed Data by default, such as C#, Visual Basic.NET and JScript.NET, whereas others, namely C++, do not. Targeting CLR can, depending on the language you’re using, impose certain constraints on the features available. As with managed and unmanaged code, one can have both managed and unmanaged data in .NET applications - data that doesn’t get garbage collected but instead is looked after by unmanaged code.

Common Type System

The CLR uses something called the Common Type System (CTS) to strictly enforce type-safety. This ensures that all classes are compatible with each other, by describing types in a common way. CTS define how types work within the runtime, which enables types in one language to interoperate with types in another language, including cross-language exception handling. As well as ensuring that types are only used in appropriate ways, the runtime also ensures that code doesn’t attempt to access memory that hasn’t been allocated to it.

Common Language Specification

The CLR provides built-in support for language interoperability. To ensure that you can develop managed code that can be fully used by developers using any programming language, a set of language features and rules for using them called the Common Language Specification (CLS) has been defined. Components that follow these rules and expose only CLS features are considered CLS-compliant.

Languages Supported By .Net

The multi-language capability of the .NET Framework and Visual Studio .NET enables developers to use their existing programming skills to build all types of applications and XML Web services. The .NET framework supports new versions of Microsoft’s old favorites Visual Basic and C++ (as VB.NET and Managed C++), but there are also a number of new additions to the family.

Visual Basic .NET has been updated to include many new and improved language features that make it a powerful object-oriented programming language. These features include inheritance, interfaces, and overloading, among others. Visual Basic also now supports structured exception handling, custom attributes and also supports multi-threading.

Visual Basic .NET is also CLS compliant, which means that any CLS-compliant language can use the classes, objects, and components you create in Visual Basic .NET.

C# is Microsoft’s new language. It’s a C-style language that is essentially “C++ for Rapid Application Development”. Unlike other languages, its specification is just the grammar of the language. It has no standard library of its own, and instead has been designed with the intention of using the .NET libraries as its own.

Microsoft Visual J# .NET provides the easiest transition for Java-language developers into the world of XML Web Services and dramatically improves the interoperability of Java-language programs with existing software written in a variety of other programming languages.

Active State has created Visual Perl and Visual Python, which enable .NET-aware applications to be built in either Perl or Python. Both products can be integrated into the Visual Studio .NET environment. Visual Perl includes support for Active State’s Perl Dev Kit.

Other languages for which .NET compilers are available include

* FORTRAN
* COBOL
* Eiffel

ASP.NET XML WEB SERVICES & Windows Forms

* Base Class Libraries
* Common Language Runtime
* Operating System

C#.NET is also compliant with CLS (Common Language Specification) and supports structured exception handling. CLS is set of rules and constructs that are supported by the CLR (Common Language Runtime). CLR is the runtime environment provided by the .NET Framework; it manages the execution of the code and also makes the development process easier by providing services.

C#.NET is a CLS-compliant language. Any objects, classes, or components that created in C#.NET can be used in any other CLS-compliant language. In addition, we can use objects, classes, and components created in other CLS-compliant languages in C#.NET. The use of CLS ensures complete interoperability among applications, regardless of the languages used to create the application.

Constructors and Destructors

Constructors are used to initialize objects, whereas destructors are used to destroy them. In other words, destructors are used to release the resources allocated to the object. In C#.NET the sub finalize procedure is available. The sub finalize procedure is used to complete the tasks that must be performed when an object is destroyed. The sub finalize procedure is called automatically when an object is destroyed. In addition, the sub finalize procedure can be called only from the class it belongs to or from derived classes.

Garbage Collection

Garbage Collection is another new feature in C#.NET. The .NET Framework monitors allocated resources, such as objects and variables. In addition, the .NET Framework automatically releases memory for reuse by destroying objects that are no longer in use.

In C#.NET, the garbage collector checks for the objects that are not currently in use by applications. When the garbage collector comes across an object that is marked for garbage collection, it releases the memory occupied by the object.

Overloading

Overloading is another feature in C#. Overloading enables us to define multiple procedures with the same name, where each procedure has a different set of arguments. Besides using overloading for procedures, we can use it for constructors and properties in a class.

Multithreading

C#.NET also supports multithreading. An application that supports multithreading can handle multiple tasks simultaneously, we can use multithreading to decrease the time taken by an application to respond to user interaction.

Structured Exception Handling

C#.NET supports structured handling, which enables us to detect and remove errors at runtime. In C#.NET, we need to use Try…Catch…Finally statements to create exception handlers. Using Try…Catch…Finally statements, we can create robust and effective exception handlers to improve the performance of our application.

The .Net Framework

The .NET Framework is a new computing platform that simplifies application development in the highly distributed environment of the Internet.

Objectives Of .Net Framework

1. To provide a consistent object-oriented programming environment whether object codes is stored and executed locally on Internet-distributed, or executed remotely.

2. To provide a code-execution environment to minimizes software deployment and guarantees safe execution of code.

3. Eliminates the performance problems.

There are different types of application, such as Windows-based applications and Web-based applications.

CHAPTER 4

4. SYSTEM DESIGN

4.1. SYSTEM FLOW DIAGRAM

Connect Receiver

False

True

View Router

Set Node Status

Select File

User Login

Login

Check Username & password

*Fig. 4.1.1 System Flow Diagram*

The Flow Diagram of System which is shown in the (Fig. 4.1.1). The interface displays options for network management. We can connect receivers, view router information, set node status, or select a file. Login requires username and password verification.

4.2. DATA FLOW DIAGRAM

Login Form

View Router

Connect Receiver

Set Node Status

Select File

Yes

No

*Fig. 4.2.1. Login Form*

This is a login screen for a network management tool. After successful login (username and password verification), you'll be granted access to various functionalities like viewing router information, connecting receivers, setting node status, and selecting files.

Node Information:

Set Node Status

Type Receiver Name

Set IP Address File Size Status

Not Inserted

Inserted

Node Table

Node Details Inserted

Node Table

*Fig 4.2.2 Set Node Status*

* Type Receiver Name: Enter the specific receiver you want to modify.
* Set IP Address/File Size/Status: It allows defining the IP address, file size, and potentially an operational status (active/inactive) for the receiver.
* Check Status: This button likely verifies if the changes were applied successfully.
* Not Inserted/Inserted: These options might indicate the outcome of the status update (failed or successful).
* Node Table/Node Details Inserted/Node Table: This suggests the system displays the updated information within a "Node Table" after successful insertion.

Connect Status of Receiver:

Connect Receiver

Not Connected

All Ready Connected

Connect to Receiver

*Fig 4.2.3. Connect Receiver*

* Check Receiver: This initiates a process to verify the receiver's connection status.
* Not Connected/All Ready Connected: These options display the outcome of the check - either the receiver is not currently connected or already established.
* Connect to Receiver: Assuming the check reveals a "Not Connected" status, this button triggers the actual connection process between the receiver and the network.

Types of File to be transmitted:

Select File

Successfully Send Data to Receiver

Not Found

Found

False

True

*Fig 4.2.4. Select File*

* Select File: Choose the data you want to transmit.
* Check Router Receiver & IP Address: Verify network connectivity and receiver location.
* Check Attack: Ensure secure transmission by checking for potential threats.
* Successfully Send Data to Receiver: Transmit the data securely to the designated recipient.

Router Process:

Sends Data

Router 1

Router 2

Router 3

*Fig 4.2.5. Router Process*

* Receive data packet.
* Check destination IP address.
* Consult routing table for best path.
* Forward packet towards its destination.

OVERALL PROCESS

File Transfer

File Transfer

Receiver

Default Routing Table

Routing Table

*Fig 4.3.1 Overall Process*

4.4. INPUT DESIGN AND OUTPUT DESIGN

INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

* What data should be given as input?
* How the data should be arranged or coded?
* The dialog to guide the operating personnel in providing input.
* Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant.

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
2. Select methods for presenting information.
3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

* Convey information about past activities, current status or projections of the future.
* Signal important events, opportunities, problems, or warnings.
* Trigger an action.
* Confirm an action.

4.5. DATABASE DESIGN

Table 4.5.1. Default routing table

|  |  |
| --- | --- |
| Field Name | Data Type |
| Snode | Text |
| Irouter | Text |
| Dest | Text |
| Attacker | Text |

Table 4.5.2. Receiver

|  |  |
| --- | --- |
| Field Name | Data Type |
| Nodename | Text |
| Ipaddress | Text |
| Port | Text |
| Status | Text |
| Fsize | Number |
| Ipstatus | text |

Table 4.5.3. Routing table

|  |  |
| --- | --- |
| Field Name | Data Type |
| Snode | Text |
| Irouter | Text |
| Dest | Text |
| Attacker | Text |

CHAPTER 5

5. SYSTEM DESCRIPTION

5.1. MODULES DESCRIPTION

Upload & Send files to users

In this module, server can upload the files in the database. After verify user digital signature file could be transfer to correct user via mobile ad-hoc network.

Attack on Ad-Hoc Network

In this module, to see what the attack on ad-hoc is network is Distributed Denial of Services (Malware). A Malware attack is a form of Malware attack but difference is that Malware attack is performed by only one node and Malware is performed b y the combination of many nodes. All nodes simultaneously attack on the victim node or network b y sending them huge packets, this will totally consume the victim bandwidth and this will not allow victim to receive the important data from the network.

Criteria for Attack detection

In this module, we use multiple nodes and simulate through different criteria are NORMAL, Malware and IDS (intrusion detection case)

Normal Case

We set number of sender and receiver nodes and transport layer mechanism as TCP and UDP with routing protocol as AODV (ad-hoc on demand distance vector) routing. After setting all parameter simulate the result through our simulator.

IDS Case

In IDS (Intrusion Detection System) we set one node as IDS node that node watch the all radio range mobile nodes if any abnormal behavior comes to our network, first check the symptoms of the attack and find out the attacker node , after finding attacker node, IDS block the attacker node and remove from the DMALWARE attack. In our simulation result we performed some analysis in terms of routing load , UDP analysis , TCP congestion window, Throughput Analysis and overall summery.

Simulation Results

In this module, we implement the random waypoint movement model for the simulation, in which a node starts at a random position, waits for the pause time, and then moves to another random position with a velocity

a. Throughput

b. Packet delivery fraction

c. End to End delay

d. Normalized routing load

5.2. INTRODUCTION TO SQL SERVER

The database component of Microsoft® SQL Server™ 2005 is a Structured Query Language (SQL)–based, scalable, relational database with integrated Extensible Markup Language (XML) support for Internet applications. Each of the following terms describes a fundamental part of the architecture of the SQL Server 2005 database component:

DATABASE

A database is similar to a data file in that it is a storage place for data. Like a data file, a database does not present information directly to a user; the user runs an application that accesses data from the database and presents it to the user in an understandable format.

Database systems are more powerful than data files in that data is more highly organized. In a well-designed database, there are no duplicate pieces of data that the user or application must update at the same time. Related pieces of data are grouped together in a single structure or record, and relationships can be defined between these structures and records.

A database typically has two main parts: first, the files holding the physical database and second, the database management system (DBMS) software that applications use to access data. The DBMS is responsible for enforcing the database structure, including:

* Maintaining relationships between data in the database.
* Ensuring that data is stored correctly, and that the rules defining data relationships

are not violated.

* Recovering all data to a point of known consistency in case of system failures.

CHAPTER 6

6. IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and it’s constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

Implementation is the stage in the project where the theoretical design is turned into a working system. The most critical stage is achieving a successful system and in giving confidence on the new system for the users, what it will work efficient and effectively.

The implementation process begins with preparing a plan for the implementation of the system. According to this plan, the activities are to be carried out in these plans; discussion has been made regarding the equipment, resources and how to test activities.

The coding step translates a detail design representation into a programming language realization. Programming languages are vehicles for communication between human and computers programming language characteristics and coding style can profoundly affect software quality and maintainability. The coding is done with the following characteristics in mind.

* Ease of design to code translation.
* Code efficiency.
* Memory efficiency.
* Maintainability.

CHAPTER 7

7. TESTING

Testing is an important stage in any system development life cycle. Testing is a process of executing a program with the intention of finding errors. The importance of software testing and its implications with respect to software quality cannot be overemphasized. Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. A good test case is one that has a high probability of finding a yet undiscovered error.

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

The important phase of software development is concerned with translating the design specification into the error-free source code. Testing is carried out to ensure that the system does not fail, that it meets the specification and it satisfies the user. The system testing was carried out in a systematic manner with a test data containing all possible combinations of data to check the features of the system. A test data was prepared for each module, which took care of all the modules of the program.

System Testing is an important stage where the system developed is tested with duplicate or original data. It is a process of executing a program with the intent of finding an error. It is a critical process that can consume fifty percent of the development time.

The following are the attributes of good test :

* A good test has a high probability of finding an error.
* A good test is not redundant.
* A good test should be "best of breed".
* A good test should be neither simple nor too complex.

7.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

In the unit testing the analyst tests the program making up a system. The software units in a system are the modules and routines that are assembled and integrated to perform a specific function. In a large system, many modules on different levels are needed.

Unit testing can be performed from the bottom up starting with the smallest and lowest level modules and proceeding one at a time. For each module in a bottom-up testing, a short program executes the module and provides the needed data.

7.2 SYSTEM TESTING

System testing can be defined in many ways, but a simple definition is that can be reasonably expected by the customer. After validation test has been conducted, one of two possible conditions exists.

* The functions or performance characteristics confirm to specification and are accepted.
* A deviation from specification is uncovered and a deficiency list is created.

Proposed system under consideration has been tested by using validation testing and found to be working satisfactorily.

For example, in this project validation testing is performed against module. This module is tested with the following valid and invalid inputs for the field id.

* 1. INTEGRATED TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Integration testing is a systematic technique for constructing the program structure while conducting test to uncover errors associate with interfacing. Objectives are used to take unit test modules and built program structure that has been directed by design. The integration testing is performed for this project when all the modules where to make it a complete system. After integration the project works successfully.

# **CHAPTER 8**

1. BACKUP AND MAINTENANCE

The term “Backup and maintenance” is used to describe the software engineering activities that occur following delivery of a software product to the customer. The maintenance phase of the software life cycle is the time period in which a software product performs useful work.

Maintenance activities involve making enhancements to software products, adapting products to new environments, and correcting problems. Software product enhancement may involve providing new functional capabilities, improving user displays and modes of interaction, upgrading external documents and internal documentation, or upgrading the performance characteristics of a system. Adaptation of software to a new environment may involve moving the software to different machine etc.

Problem correction involves modification and revalidation of software to correct errors. Some errors require immediate attention and some can be corrected on a scheduled, periodic basis, and other are known but never corrected.

Additional features were added to the admin part of the site after implementation. As the administrator needed some easier way of uploading for the prediction part of the site, that section was developed at this stage of the project. The daily prediction uploading section needed the date should be incremented automatically for each day of the following week. Those sections were done at this stage of the site development.

All of these tasks must be accomplished using a systematic, orderly approach to tracking and analysis of change requests and careful redesign, re-implementation, revalidation and re-documentation of the changes. Otherwise, the software product quickly degrades as a result of the maintenance process. It is not unusual for a well designed, properly implemented and adequately documented initial version of a software product to become unmentionable due to inadequate maintenance procedures. This can result in situation in which it becomes easier and less expensive to re implement a module or subsystem than to modify the existing version.

Software maintenance activities must not destroy the maintainability of the software. A small change in the source code often requires extensive changes to the test suite and the supporting documents. Failure to recognize the true cost of a “small change” in the source code is one of the most significant problems in software maintenance.

CHAPTER 9

RESULT & DISCUSSION

Login

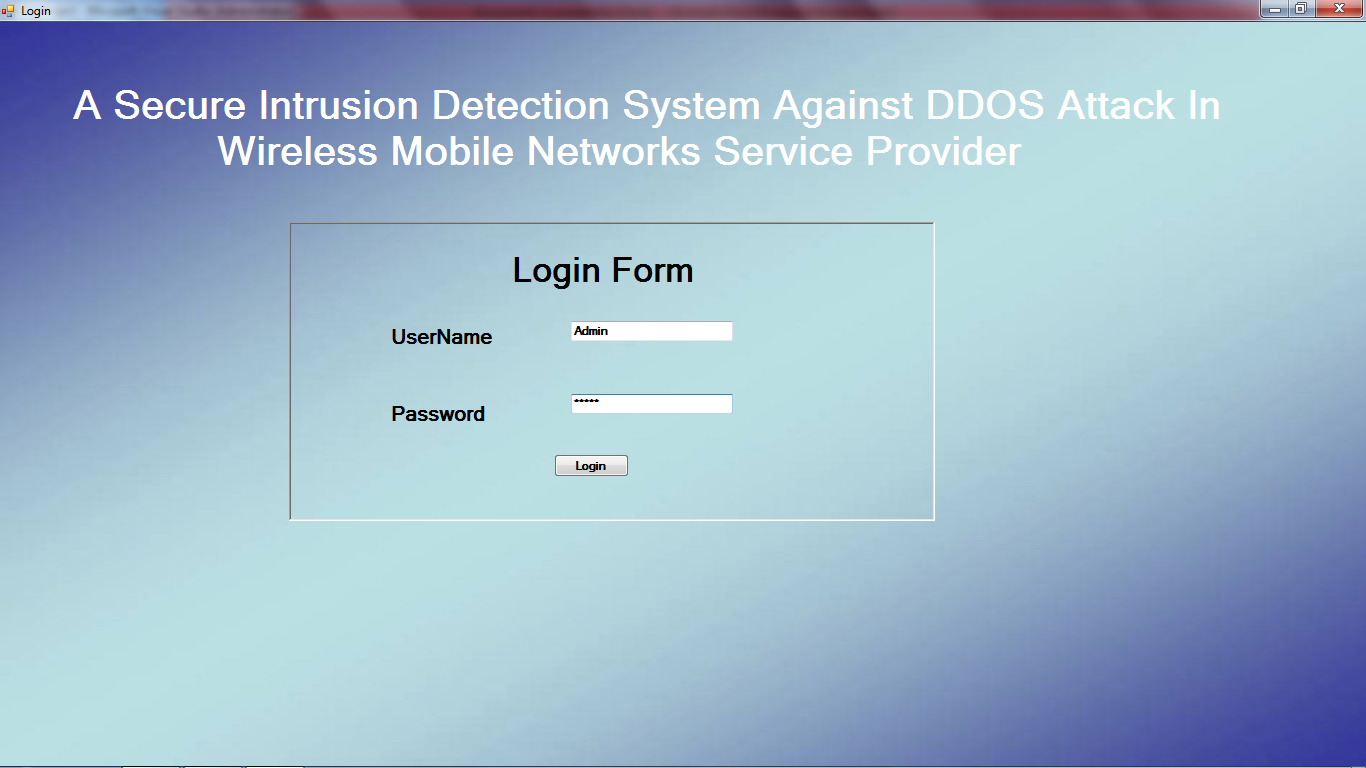


Fig 9.1: Loging into the Server.

View Router

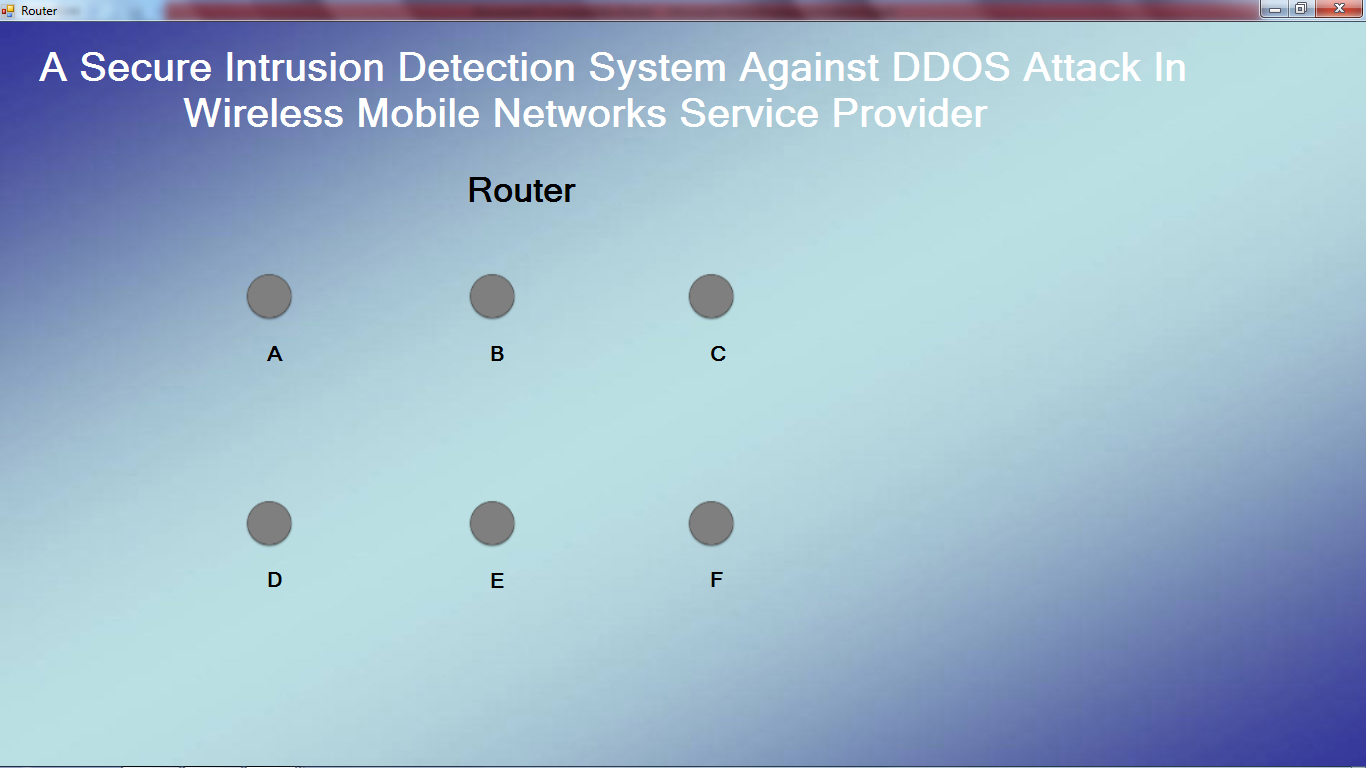


Fig 2: Different Router that are connect to the Server.

Connect Receiver A

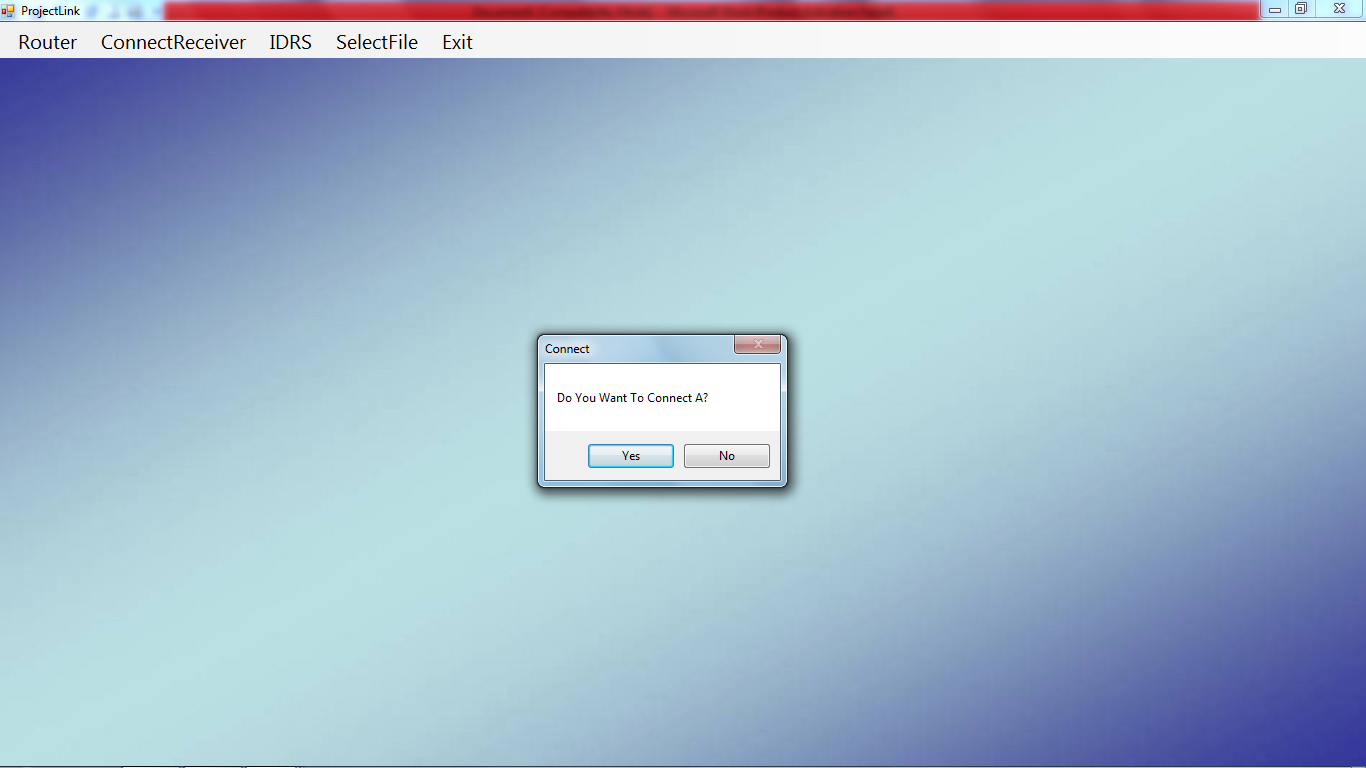


Fig 3: Creating a link between Two Router in our case it is `A`.

Node Status



Fig 4: Updating the Information about the Data.

Select File Name

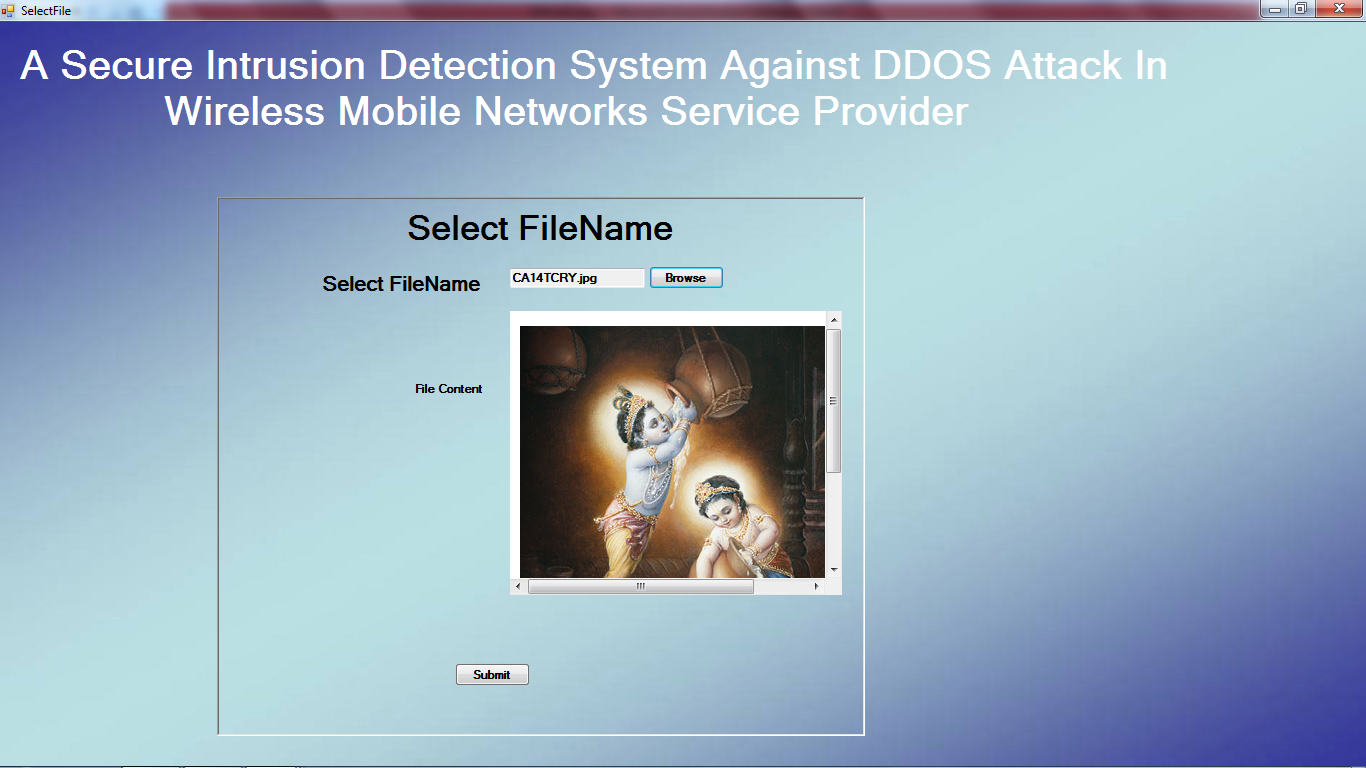


Fig 5: Selecting the File that is going to be transferred.

Select Router

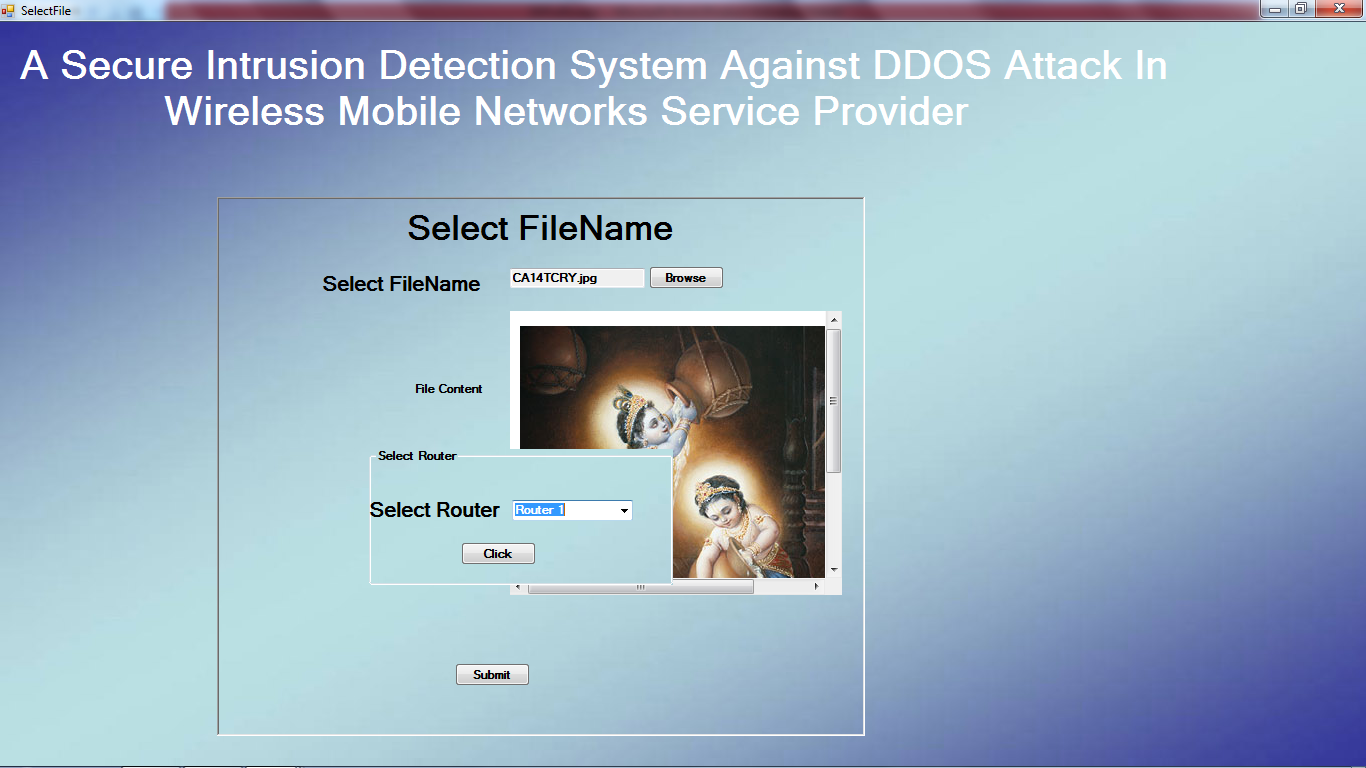


Fig 6: Selecting the Routing Path from the Given Path.

Select Destination Node



Fig 7: Selecting the Destination Node.

Enter IP Address

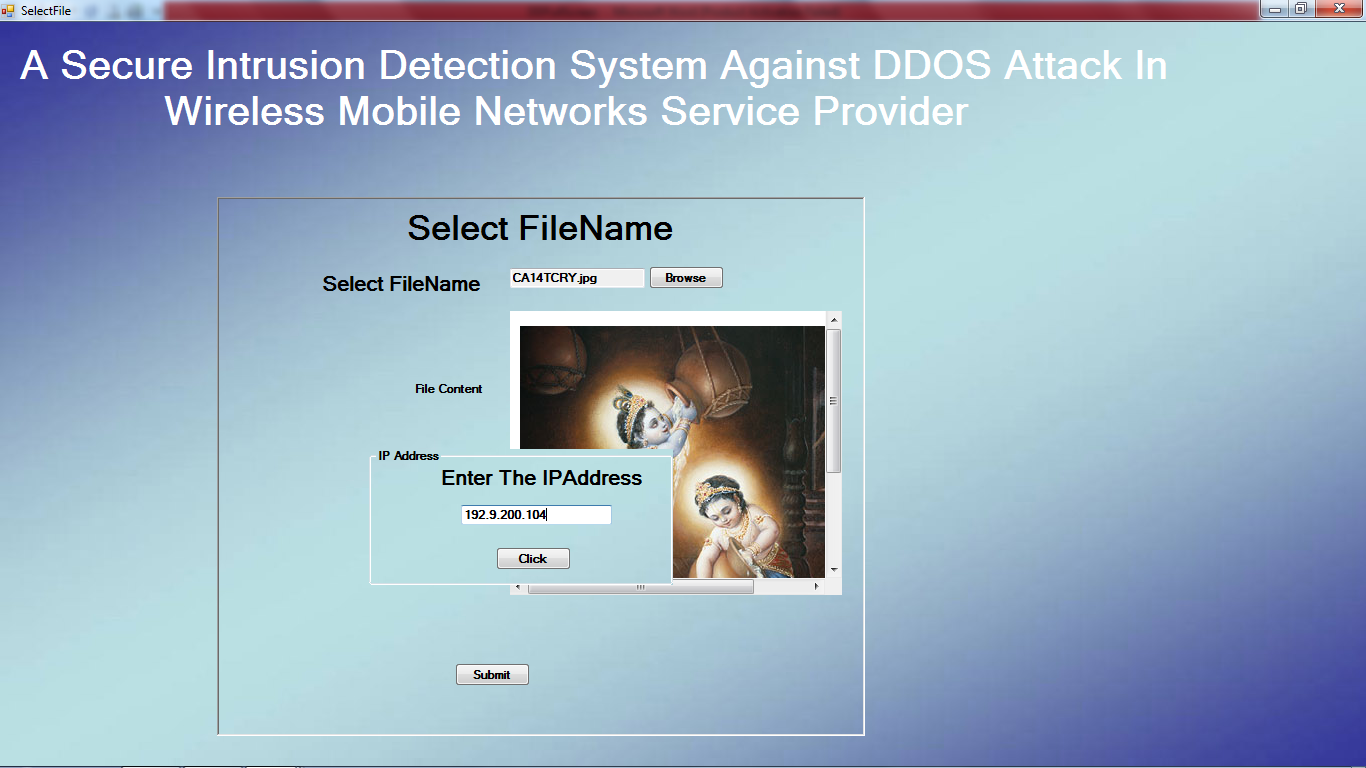


Fig 8: Enter the Destination IP Address.

Successfully Send Your Data

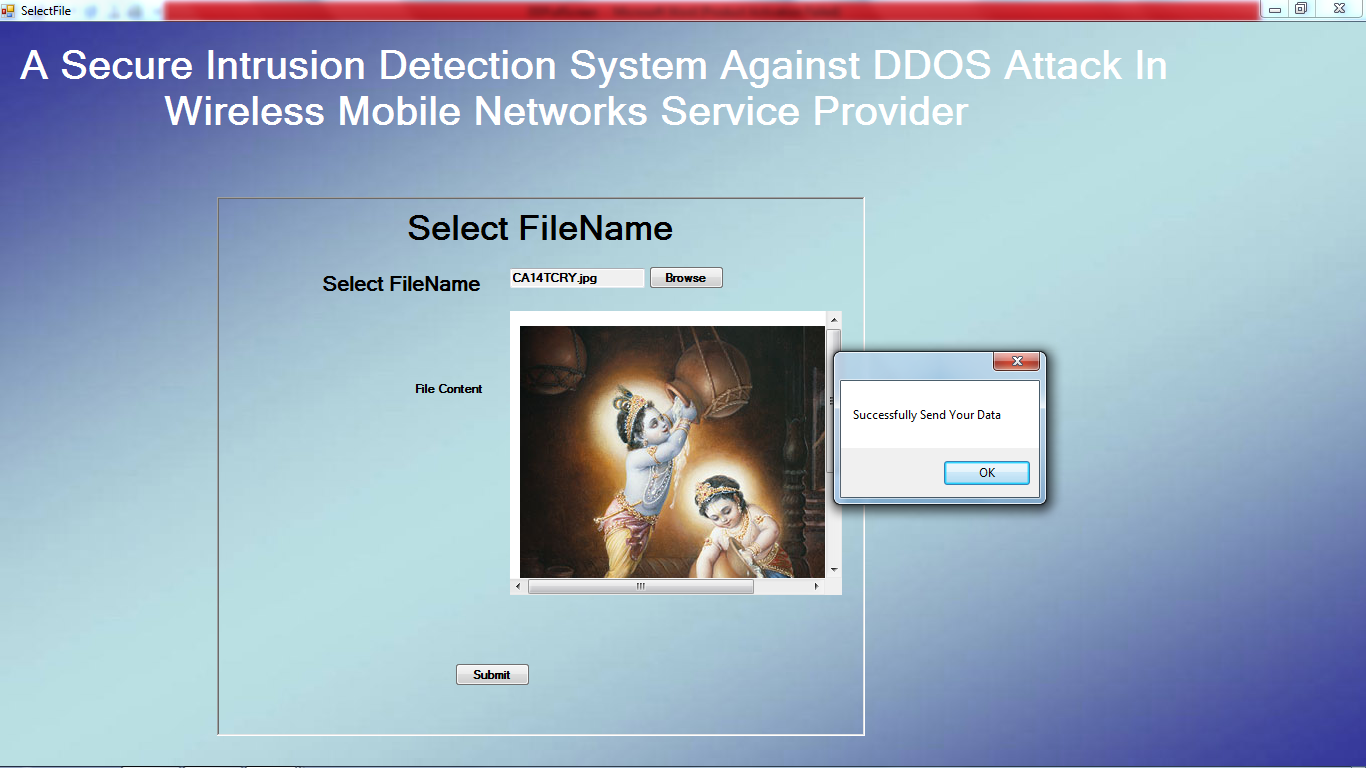


Fig 9: Getting a POP-UP saying that Data has been send Successfully.

Receiver A Receive Files

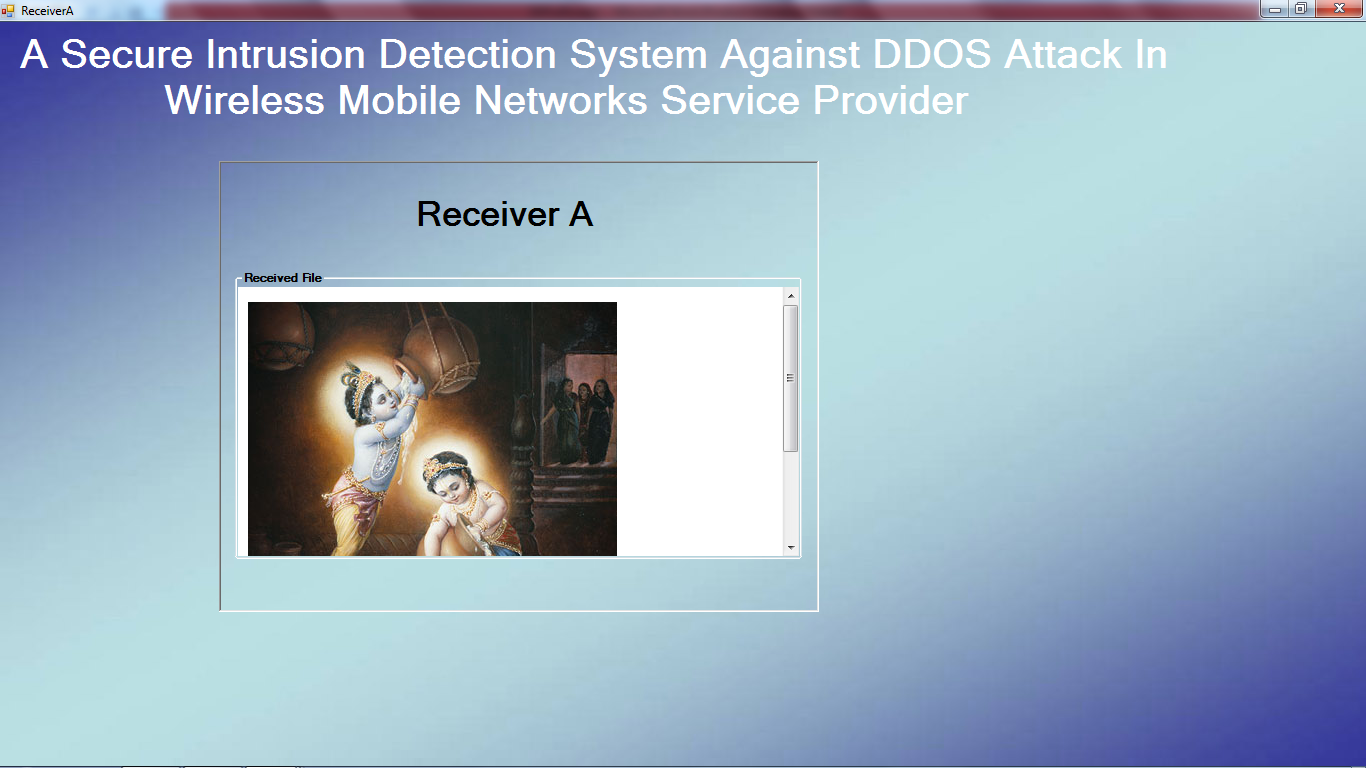


Fig 10: We can see the Receiver `A` Got the Data.

DDOS Attack Found

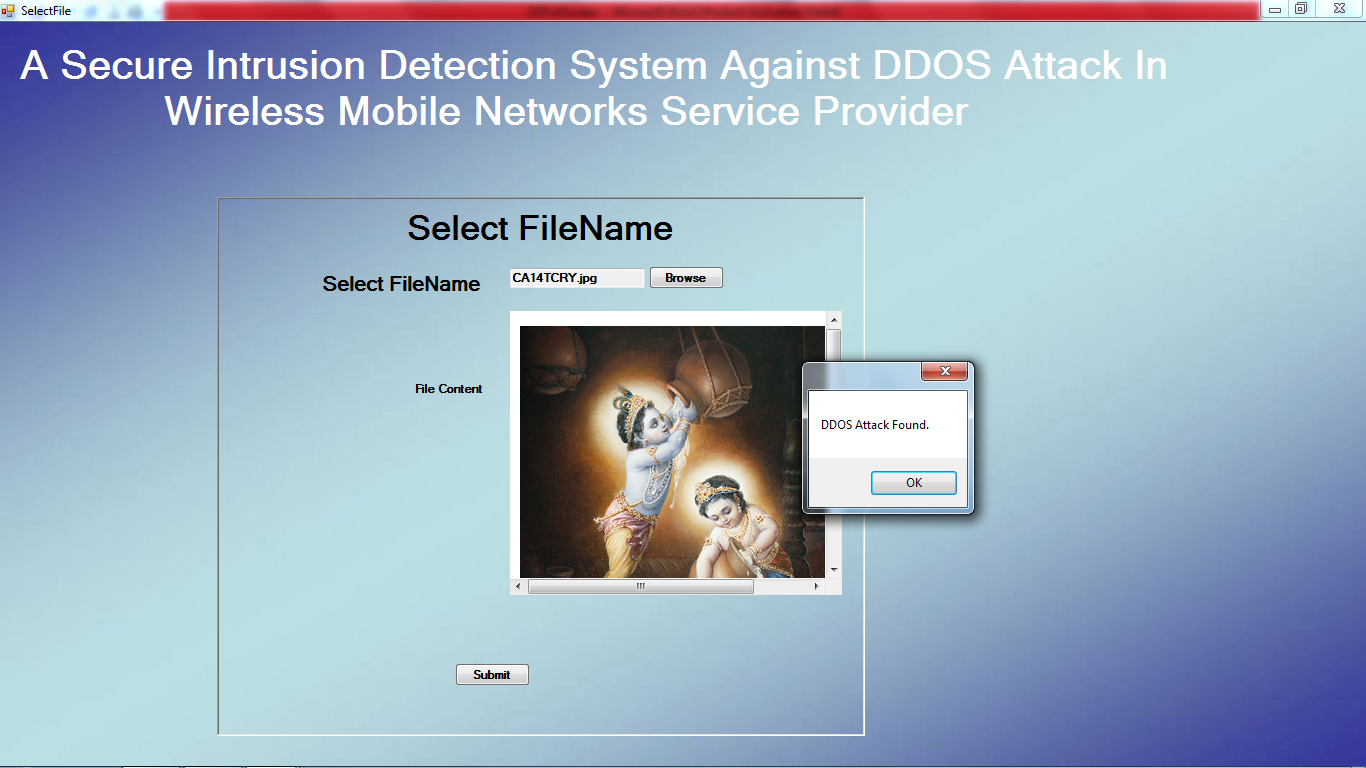


Fig 11: Suddenly we got an DDOS Attack. Which id indicated by a POP-UP.

Location Discloser Attacker Found

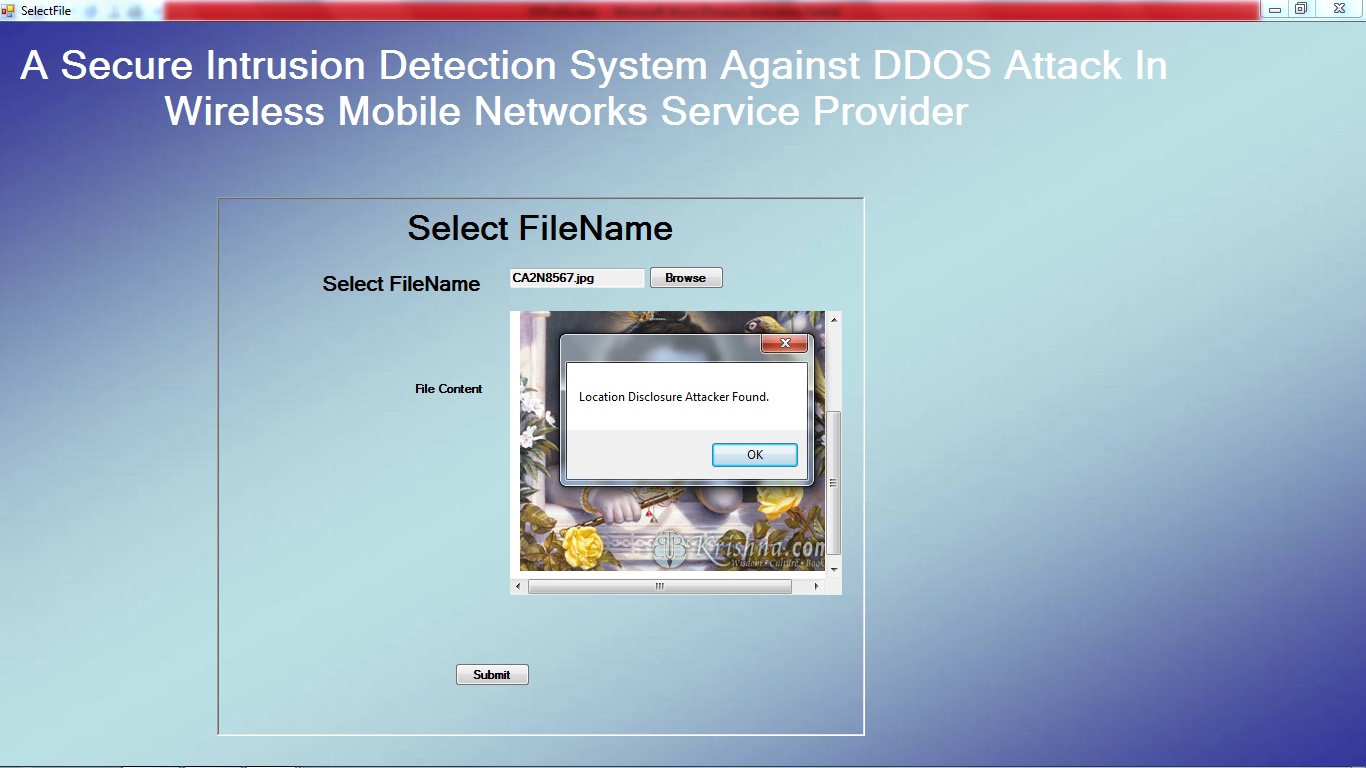


Fig 12: We got another type of Attack as well. Which is Generated by the System.

IDRS Master

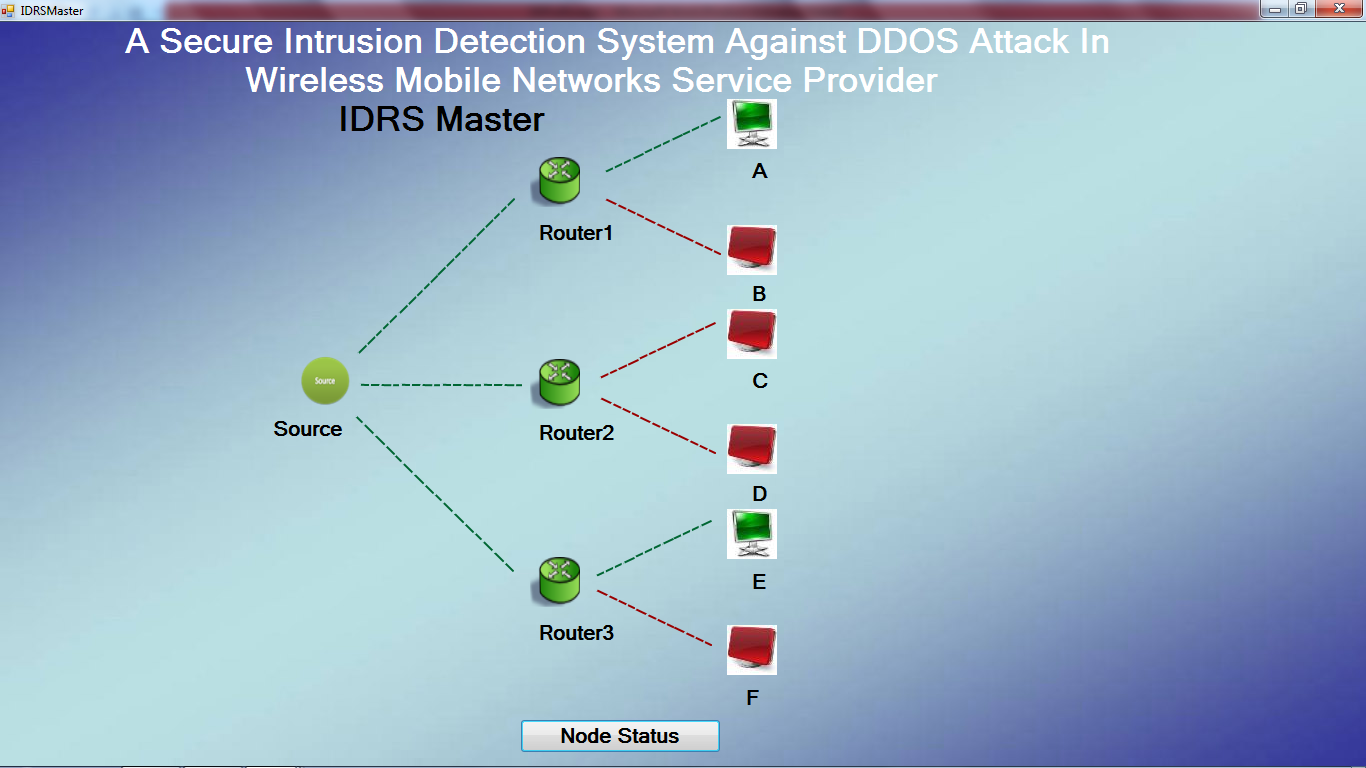


Fig 13: We can see which Router link are been attacked. With this we can make a secure. Network by selecting best and optimal routing path for transfer Data.

SAMPLE CODE:

IDRS Master:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace DDProject

{

public partial class IDRSMaster : Form

{

public IDRSMaster()

{

InitializeComponent();

}

SqlDataAdapter adp;

DataTable dt;

ConnectionClass c = new ConnectionClass();

private void IDRSMaster\_Load(object sender, EventArgs e)

{

}

/\*private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

comboBox2.Items.Clear();

comboBox2.Text = "";

if(comboBox1.SelectedIndex == 0)

{

comboBox2.Items.Add("A");

comboBox2.Items.Add("B");

}

else if(comboBox1.SelectedIndex == 1)

{

comboBox2.Items.Add("C");

comboBox2.Items.Add("D");

}

else if(comboBox1.SelectedIndex == 2)

{

comboBox2.Items.Add("E");

comboBox2.Items.Add("F");

}

}

private void button1\_Click(object sender, EventArgs e)

{

string path = Application.StartupPath;

path = path.Substring(0, path.LastIndexOf("bin"));

//MessageBox.Show(path);

path = path + "LineImage\\";

string status = "Block";

if(comboBox1.SelectedIndex == 0)

{

ti1.Visible = true;

ti2.Visible = true;

ti3.Visible = true;

ti4.Visible = true;

if(comboBox2.SelectedIndex == 0)

{

if(status.Equals("Block"))

{

r1t1.Image = Image.FromFile(path + "redt1.png");

r1t2.Image = Image.FromFile(path + "redt1.png");

r1t3.Image = Image.FromFile(path + "redt1.png");

i1.Image = Image.FromFile(path + "off.jpg");

}

else if(status.Equals("UnBlock"))

{

r1t1.Image = Image.FromFile(path + "st1.png");

r1t2.Image = Image.FromFile(path + "st1.png");

r1t3.Image = Image.FromFile(path + "st1.png");

i1.Image = Image.FromFile(path + "on.jpg");

}

r1t1.Visible = true;

r1t2.Visible = true;

r1t3.Visible = true;

}

else if(comboBox2.SelectedIndex == 1)

{

if(status.Equals("Block"))

{

r1b1.Image = Image.FromFile(path + "redr1.png");

r1b2.Image = Image.FromFile(path + "redr1.png");

r1b3.Image = Image.FromFile(path + "redr1.png");

i2.Image = Image.FromFile(path + "off.jpg");

}

else if(status.Equals("UnBlock"))

{

r1b1.Image = Image.FromFile(path + "sr1.png");

r1b2.Image = Image.FromFile(path + "sr1.png");

r1b3.Image = Image.FromFile(path + "sr1.png");

i2.Image = Image.FromFile(path + "on.jpg");

}

r1b1.Visible = true;

r1b2.Visible = true;

r1b3.Visible = true;

}

}

else if(comboBox1.SelectedIndex == 1)

{

si1.Visible = true;

si2.Visible = true;

si3.Visible = true;

si4.Visible = true;

if(comboBox2.SelectedIndex == 0)

{

if(status.Equals("Block"))

{

r2t1.Image = Image.FromFile(path + "redt1.png");

r2t2.Image = Image.FromFile(path + "redt1.png");

r2t3.Image = Image.FromFile(path + "redt1.png");

i3.Image = Image.FromFile(path + "off.jpg");

}

else if(status.Equals("UnBlock"))

{

r2t1.Image = Image.FromFile(path + "st1.png");

r2t2.Image = Image.FromFile(path + "st1.png");

r2t3.Image = Image.FromFile(path + "st1.png");

i3.Image = Image.FromFile(path + "on.jpg");

}

r2t1.Visible = true;

r2t2.Visible = true;

r2t3.Visible = true;

}

else if(comboBox2.SelectedIndex == 1)

{

if(status.Equals("Block"))

{

r2b1.Image = Image.FromFile(path + "redr1.png");

r2b2.Image = Image.FromFile(path + "redr1.png");

r2b3.Image = Image.FromFile(path + "redr1.png");

i4.Image = Image.FromFile(path + "off.jpg");

}

else if(status.Equals("UnBlock"))

{

r2b1.Image = Image.FromFile(path + "sr1.png");

r2b2.Image = Image.FromFile(path + "sr1.png");

r2b3.Image = Image.FromFile(path + "sr1.png");

i4.Image = Image.FromFile(path + "on.jpg");

}

r2b1.Visible = true;

r2b2.Visible = true;

r2b3.Visible = true;

}

}

else if(comboBox1.SelectedIndex == 2)

{

bi1.Visible = true;

bi2.Visible = true;

bi3.Visible = true;

bi4.Visible = true;

if(comboBox2.SelectedIndex == 0)

{

if(status.Equals("Block"))

{

r3t1.Image = Image.FromFile(path + "redt1.png");

r3t2.Image = Image.FromFile(path + "redt1.png");

r3t3.Image = Image.FromFile(path + "redt1.png");

i5.Image = Image.FromFile(path + "off.jpg");

}

else if(status.Equals("UnBlock"))

{

r3t1.Image = Image.FromFile(path + "st1.png");

r3t2.Image = Image.FromFile(path + "st1.png");

r3t3.Image = Image.FromFile(path + "st1.png");

i5.Image = Image.FromFile(path + "on.jpg");

}

r3t1.Visible = true;

r3t2.Visible = true;

r3t3.Visible = true;

}

else if(comboBox2.SelectedIndex == 1)

{

if(status.Equals("Block"))

{

r3b1.Image = Image.FromFile(path + "redr1.png");

r3b2.Image = Image.FromFile(path + "redr1.png");

r3b3.Image = Image.FromFile(path + "redr1.png");

i6.Image = Image.FromFile(path + "off.jpg");

}

else if(status.Equals("UnBlock"))

{

r3b1.Image = Image.FromFile(path + "sr1.png");

r3b2.Image = Image.FromFile(path + "sr1.png");

r3b3.Image = Image.FromFile(path + "sr1.png");

i6.Image = Image.FromFile(path + "on.jpg");

}

r3b1.Visible = true;

r3b2.Visible = true;

r3b3.Visible = true;

}

}

}\*/

private void button2\_Click(object sender, EventArgs e)

{

ChangeNodeStatus cn = new ChangeNodeStatus();

cn.Show();

}

private void timer1\_Tick(object sender, EventArgs e)

{

try

{

adp = new SqlDataAdapter("select \* from ncstatus order by nodename", c.GetConnection ());

dt = new DataTable();

adp.Fill(dt);

if(dt.Rows.Count != 0)

{

string path = Application.StartupPath;

path = path.Substring(0, path.LastIndexOf("bin"));

//MessageBox.Show(path);

path = path + "LineImage\\";

string status = "";

foreach (DataRow dr in dt .Rows )

{

status = dr["status"].ToString();

if(dr["Router"].ToString ().Equals ("Router 1"))

{

ti1.Visible = true;

ti2.Visible = true;

ti3.Visible = true;

ti4.Visible = true;

if(dr["nodename"].ToString ().Equals ("A"))

{

if(status.Equals("Attack"))

{

r1t1.Image = Image.FromFile(path + "redt1.png");

r1t2.Image = Image.FromFile(path + "redt1.png");

r1t3.Image = Image.FromFile(path + "redt1.png");

i1.Image = Image.FromFile(path + "off.jpg");

}

else if(status.Equals("Success"))

{

r1t1.Image = Image.FromFile(path + "st1.png");

r1t2.Image = Image.FromFile(path + "st1.png");

r1t3.Image = Image.FromFile(path + "st1.png");

i1.Image = Image.FromFile(path + "on.jpg");

}

r1t1.Visible = true;

r1t2.Visible = true;

r1t3.Visible = true;

}

if(dr["nodename"].ToString().Equals("B"))

{

if(status.Equals("Attack"))

{

r1b1.Image = Image.FromFile(path + "redr1.png");

r1b2.Image = Image.FromFile(path + "redr1.png");

r1b3.Image = Image.FromFile(path + "redr1.png");

i2.Image = Image.FromFile(path + "off.jpg");

}

else if(status.Equals("Success"))

{

r1b1.Image = Image.FromFile(path + "sr1.png");

r1b2.Image = Image.FromFile(path + "sr1.png");

r1b3.Image = Image.FromFile(path + "sr1.png");

i2.Image = Image.FromFile(path + "on.jpg");

}

r1b1.Visible = true;

r1b2.Visible = true;

r1b3.Visible = true;

}

}

else if(dr["Router"].ToString().Equals("Router 2"))

{

si1.Visible = true;

si2.Visible = true;

si3.Visible = true;

si4.Visible = true;

if(dr["nodename"].ToString ().Equals ("C"))

{

if(status.Equals("Attack"))

{

r2t1.Image = Image.FromFile(path + "redt1.png");

r2t2.Image = Image.FromFile(path + "redt1.png");

r2t3.Image = Image.FromFile(path + "redt1.png");

i3.Image = Image.FromFile(path + "off.jpg");

}

else if(status.Equals("Success"))

{

r2t1.Image = Image.FromFile(path + "st1.png");

r2t2.Image = Image.FromFile(path + "st1.png");

r2t3.Image = Image.FromFile(path + "st1.png");

i3.Image = Image.FromFile(path + "on.jpg");

}

r2t1.Visible = true;

r2t2.Visible = true;

r2t3.Visible = true;

}

else if(dr["nodename"].ToString().Equals("D"))

{

if(status.Equals("Attack"))

{

r2b1.Image = Image.FromFile(path + "redr1.png");

r2b2.Image = Image.FromFile(path + "redr1.png");

r2b3.Image = Image.FromFile(path + "redr1.png");

i4.Image = Image.FromFile(path + "off.jpg");

}

else if(status.Equals("Success"))

{

r2b1.Image = Image.FromFile(path + "sr1.png");

r2b2.Image = Image.FromFile(path + "sr1.png");

r2b3.Image = Image.FromFile(path + "sr1.png");

i4.Image = Image.FromFile(path + "on.jpg");

}

r2b1.Visible = true;

r2b2.Visible = true;

r2b3.Visible = true;

}

}

else if(dr["Router"].ToString().Equals("Router 3"))

{

bi1.Visible = true;

bi2.Visible = true;

bi3.Visible = true;

bi4.Visible = true;

if(dr["nodename"].ToString().Equals("E"))

{

if(status.Equals("Attack"))

{

r3t1.Image = Image.FromFile(path + "redt1.png");

r3t2.Image = Image.FromFile(path + "redt1.png");

r3t3.Image = Image.FromFile(path + "redt1.png");

i5.Image = Image.FromFile(path + "off.jpg");

}

else if(status.Equals("Success"))

{

r3t1.Image = Image.FromFile(path + "st1.png");

r3t2.Image = Image.FromFile(path + "st1.png");

r3t3.Image = Image.FromFile(path + "st1.png");

i5.Image = Image.FromFile(path + "on.jpg");

}

r3t1.Visible = true;

r3t2.Visible = true;

r3t3.Visible = true;

}

if(dr["nodename"].ToString().Equals("F"))

{

if(status.Equals("Attack"))

{

r3b1.Image = Image.FromFile(path + "redr1.png");

r3b2.Image = Image.FromFile(path + "redr1.png");

r3b3.Image = Image.FromFile(path + "redr1.png");

i6.Image = Image.FromFile(path + "off.jpg");

}

else if(status.Equals("Success"))

{

r3b1.Image = Image.FromFile(path + "sr1.png");

r3b2.Image = Image.FromFile(path + "sr1.png");

r3b3.Image = Image.FromFile(path + "sr1.png");

i6.Image = Image.FromFile(path + "on.jpg");

}

r3b1.Visible = true;

r3b2.Visible = true;

r3b3.Visible = true;

}

}

}

}

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

}

CHAPTER 9

1. CONCLUSION

We thoroughly explore the problem of malware distribution at large-scale networks. The solution to this problem is desperately desired by cyber defenders as the network security community does not yet have solid answers. Different from previous modeling methods, we propose a two layer epidemic model: the upper layer focuses on networks of a large scale networks, for example, domains of the Internet; the lower layer focuses on the hosts of a given network. This two layer model improves the accuracy compared with the available single layer epidemic models in malware modeling. Moreover, the proposed two layer model offers us the distribution of malware in terms of the low layer networks. We perform a restricted analysis based on the proposed model, and obtain three conclusions: The distribution for a given malware in terms of networks follows exponential distribution, power law distribution with a short exponential tail, and power law distribution, at its early, late, and final stage, respectively. In order to examine our theoretical findings, we have conducted extensive experiments based on two real-world large-scale malware, and the results confirm our theoretical claims.

CHAPTER 10

1. FUTURE ENHANCEMENT

In regards to future work, we will firstly further investigate the dynamics of the late stage. More details of the findings are expected to be further studied, such as the length of the exponential tail of a power law distribution at the late stage. Secondly, defenders may care more about their own network, e.g., the distribution of a given malware at their ISP domains, where the conditions for the two layer model may not hold. We need to seek appropriate models to address this problem. Finally, we are interested in studying the distribution of multiple malware on large-scale networks as we only focus on one malware in this paper. We believe it is not a simple linear relationship in the multiple malware case compared to the single malware one.

REFERENCE

1. K. Bala, P N Renjith, M Latha, S.T. Santhanalakshmi, The Intrusion Detection System (IDS) is a systematic tool implemented to spot suspicious activity with network malfunctions and sends an alert to the security control center, (2023).
2. Mugdha Kirkire, Poonam Gupta” Intrusion Detection in Mobile Ad hoc Network ”International Journal of Computer Science and Mobile Computing, Vol.3 Issue.2, pp. 869-876, February- 2020.
3. U. Sharmila Begam, Dr. G. Murugaboopathi “A Recent Secure Intrusion Detection System For Manets” International Journal of Emerging Technology and Advanced Engineering Vol 3, Special Issue 1, January 2021.
4. Prajeet Sharma, Niresh Sharma, Rajdeep Singh , A Secure Intrusion detection system against DDOS attack in Wireless Mobile Ad-hoc Network, International Journal of Computer Applications (0975 – 8887) Volume 41– No.21, March 2020.
5. Hemant Sonawane, International Journal of Cryptography and Cyber Security (2013), Investigates the potential of blockchain technology to enhance DDoS resilience by decentralizing network resources and mitigating attacks.
6. Gagandeep, Aashima, Journal of Information Security (2012), Investigates the specific impact of DDoS attacks on the availability and performance of e-commerce websites. Identifying the vulnerabilities and risks DDoS attacks pose to e-commerce platforms.
7. Tiranuch Anantvalee, Jie Wu ,IEEE Transactions on Dependable and Secure Computing (2007),Presents an extensive survey of recent trends in DDoS attacks and the evolving landscape of countermeasures.Identifying the dynamic nature of DDoS attacks and the need for adaptive countermeasures.

BIBLIOGRAPHY

BOOKS

1. “Microsoft ASP.NET” by Mirdula Pariha, Jeff Webb, Tata Mac Graw Hill.
2. Microsoft SQL SERVER2000 by Ray Rankings, Paul Bertucci, Paul Jensen
3. Gerry O’Brien,” Microsoft IIS 5.0 Administration”
4. Sql Server - The complete Reference-Gayle Coffman
5. Teach yourself Sql Server-Richard Way Mire
6. Asp.Net-Stephen Walther (Second Edition)
7. Asp.Net-Daniel Maharry

WEBSITES

ASP.NET

1. www.asp.net/(S(pdfrohu0ajmwt445fanvj2r3))/learn/data-access/
2. www.w3schools.com/aspnet/default.asp
3. www.411asp.net/home/sites
4. www.cristiandarie.ro/asp-net-tutorial/
5. www.asp.net-tutorials.com/basics/first-website/

SQL SERVER

1. www.functionx.com/sqlserver/
2. www.technet.microsoft.com/en-us/library/ms169620.aspx
3. www.msdn.microsoft.com/en-us/library/ms169620(SQL.90).aspx
4. www.softwaretrainingtutorials.com/ms-sql-server-2005.php

