**An Industry Oriented Mini Project**

on

**MALWARE PROPAGATION IN LARGE SCALE NETWORKS**

**Project report Submitted in partial fulfilment for the award of Degree of**

**BACHELOR OF TECHONOLGY**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

**Submitted**

**By**

**Aditya Goindi 14951A0501**

**Chigurapati Harshitha**  **14951A0512**

**Akhil Uday**  **14951A0505**

**Under the Supervision**

**of**

**Dr. K Suvarchala**

Professor, CSE



**COMPUTER SCIENCE AND ENGINEERING**

**INSTITUTE OF AERONAUTICAL ENGINEERING**

**(Autonomous)**

Dundigal, Hyderabad - 500043, Telangana

**2017-2018**

****

**INSTITUTE OF AERONAUTICAL ENGINEERING**

**(Autonomous)**

Dundigal, Hyderabad - 500043, Telangana

**COMPUTER SCIENCE AND ENGINEERING**

**CERTIFICATE**

This is to certify that work embodied in this dissertation entitled “**Malware Propagation In Large Scale Networks”** being submitted by

**Aditya Goindi 14951A0501**

**Chigurapati Harshitha 14951A0512**

**Akhil Uday 14951A0505**

For partial fulfilment of the requirement for the award of **Bachelor of Technology** in **Computer Science and Engineering** discipline to Institute of Aeronautical Engineering, Dundigal, Hyderabad during the academic year 2017-2018 as a record of bonafide work, undertaken by his/her supervision of the undersigned.

**Supervised by Head of the Department**

**Dr. K Suvarchala Dr. K Rajendra Prasad**

Associate professor, CSE Professor, CSE

**EXTERNAL EXAMINER**

**INSTITUTE OF AERONAUTICAL ENGINEERING**

**(Autonomous)**

Dundigal, Hyderabad - 500043, Telangana

**COMPUTER SCIENCE AND ENGINEERING**

**DECLARATION**

We the students of **Bachelor of Technology in Computer Science and Engineering**, **2017 - 2018**, Institute of Aeronautical Engineering, Dundigal, Hyderabad, hereby declare that the work presented in this project work entitled **“Malware Propagation In Large Scale Networks”** is the outcome of our own bonafide work and is correct to the best of our knowledge. This work follows engineering ethics and contains no material previously published or written by neither person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgement has been made in the text.

**Aditya Goindi 14951A0501**

**Chigurapati Harshitha 14951A0512**

**Akhil Uday 14951A0505**

**Date:**

**ACKNOWLEDGEMENT**

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whole constant guidance and encouragement crown all the efforts with success. We thank our college management and respected **Sri M. Rajashekar Reddy**, Chairman, IARE, Dundigal for providing us the necessary infrastructure to carry out the project work.

We express our sincere thanks to **Dr. L.V. Narasimha Prasad**, Professor, Principal, and who has been a great source of information for our work, and **Dr. K. Rajendra Prasad**, Professor and Head, Department of CSE, for extending his support to carry on this project work.

We are especially thankful to our internal supervisor **Dr. K Suvarchala**, Professor, Department of CSE, for his internal support and professionalism who helped us in shaping the project into successful one.

We take this opportunity to express our thanks to one and all who directly or indirectly helped us in bringing this effort to present form.

**Aditya Goindi**

**Chigurapati Harshitha**

**Akhil Uday**

**ABSTRACT**

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. We practice this principle in this study. In this project, we study the distribution of malware in terms of networks (e.g., autonomous systems, ISP domains, abstract networks of smart phones who share the same vulnerabilities) at large scales. In this kind of setting, we have a sufficient volume of data at a large enough scale to meet the requirements of the SI model. Different from the traditional epidemic models, we break our model into two layers. Our contributions are summarized as follows. We propose a two layer malware propagation model to describe the development of a given malware at the Internet level. Compared with the existing single layer epidemic models, the proposed model represents malware propagation better in large scale networks. We find the malware distribution in terms of networks varies from exponential to power law with a short exponential tail, and to power law distribution at its early, late, and final stage, respectively. These findings are firstly theoretically proved based on the proposed model, and then confirmed by the experiments through the two large-scale real-world data sets.

**TABLE OF CONTENTS**

1. Introduction 1

1.1 Existing System 1

1.1.2 Disadvantages of Existing System 1

1.2 Proposed System 2

1.2.1 Advantages of Proposed System 2

1.3 Objectives of Proposed system 2

2. Literature Survey 3

2.1 Introduction 3

2.2 Proximity Breeds Danger 3

2.3 Eigen Trust Algorithm 4

2.4 On Mobile Viruses 4

2.5 Optimal Malware Defense System 4

2.6 Delegation Forwarding 5

2.7 Performance Analysis 5

3. Analysis 6

3.1 Requirements 6

3.1.1 Functional Requirements 6

3.1.2 Non-Functional Requirements 6

3.2 Software Requirements 7

3.3 Hardware Requirements 7

4. Design 8

4.1 Introduction 8

4.2 System Architecture 8 4.3 Block Diagram 9

4.4 Modules 10

4.4.1 Service Provide 10

4.4.2 Router 10

4.4.3 Attacker 10

4.4.4 Epidemic model 10

Training Phase 10

Test Phase 10

4.4.5 End User 10

4.5 UML Diagrams 11 4.5.1 Use case Diagram 12 4.5.2 Class Diagram 13 4.5.3 Sequence Diagram 14

4.5.4 Activity Chart 15

4.5.5 Component Diagram 16 4.5.6 Deployment Diagram 17

5. Implementation and Coding 18

6. Test Cases and Output Screenshots 25

6.1 Selecting a Java file 26

6.2 Trusted Case 27

6.2.1 Test Case in Trusted Case 28

6.3 Traffic Case 29

6.3.1 Test Case in Traffic Case 30

6.4 Malware Spreading Case 31

6.4.1 Test Case in Malware Spreading Case 32

7. Conclusion and Feature Enhancement 34

References 35

**LIST OF FIGURES**

**FIGURE NO. TITLE PAGE NO**

Fig 4.2 System Architecture 8

Fig 4.3 Block Diagram 9

Fig 4.5.1 Use case Diagram 12

Fig 4.5.2 Class Diagram 13 Fig 4.5.3 Sequence Diagram 14

Fig 4.5.4 Activity Chart 15

Fig 4.5.5 Component Diagram 16

Fig 4.5.6 Deployment Diagram 17

Fig 6.1 Selecting a Java File 31

Fig 6.2 Trusted Cases 32

Fig 6.2.1 (i) Test case 33

Fig 6.3 Traffic case 34

Fig 6.3(a) Traffic case 35

Fig 6.3.1 (i) Test case 35

Fig 6.3.1(a) (i) Test case 36

Fig 6.4 Malware Spreading Case 37

Fig 6.4(a) Malware Spreading Case 37

Fig 6.4.1 (i) Test case 38

Fig 6.4.1(a) (i) Test case 39

1. **INTRODUCTION**

It is the computational process of discovering patterns in large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems. Data mining is the analysis step of the "knowledge discovery in databases" process, or KDD.

**1.1 Existing System**

* The epidemic theory plays a leading role in malware propagation modelling. 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.
* Zou et al. used a susceptible-infected (SI) model to predict the growth of Internet worms at the early stage.
* Gao and Liu recently employed a susceptible-infected-recovered (SIR) model to describe mobile virus propagation.

**1.1.1 Disadvantages of Existing System**

* One critical condition for the epidemic models is a large vulnerable population because their principle is based on differential equations.
* As pointed by Willinger et al. the findings, which we extract from a set of observed data, usually reflect parts of the studied objects. It is more reliable to extract the-oretical results from appropriate models with confirmation from sufficient real world data set experiments..

**1.2 Proposed System**

* In this paper, we study the distribution of malware in terms of networks (e.g., autonomous systems, ISP domains, and abstract networks of smart phones who share the same vulnerabilities) at large scales.
* In this kind of setting, we have a sufficient volume of data at a large enough scale to meet the requirements of the SI model. Different from the traditional epidemic models, we break our model into two layers.
* First of all, for a given time since the breakout of a malware, we calculate how many networks have been compromised based on the SI model.
* Secondly, for a compromised net-work, we calculate how many hosts have been compromised since the time that the network was compromised.

**1.2.1 Advantages of Proposed System**

* Our rigorous analysis, we find that the distribution of a given malware follows an exponential distribution at its early stage, and obeys a power law distribution with a short exponential tail at its late stage, and finally converges to a power law distribution.

**1.3 Objectives of Proposed System**

In our analysis, we will illustrate that the distribution of a given malware follows an exponential distribution at its early stage, and obeys a power law distribution with a short exponential tail at its late stage, and finally converges.

- If in any way IDS detects malware, then we alert you with the hash value of that file i.e. .Integrity verification.

**2. LITERATURE SURVEY**

**2.1 Introduction**

Data mining is the process of extracting patterns from data. As more data are gathered, with the amount of data doubling every three years, data mining is becoming an increasingly important tool to transform these data into information. It is commonly used in a wide range of profiling practices, such as marketing, surveillance, fraud detection and scientific discovery.

While data mining can be used to uncover patterns in data samples, it is important to be aware that the use of non-representative samples of data may produce results that are not indicative of the domain. Similarly, data mining will not find patterns that may be present in the domain, if those patterns are not present in the sample being "mined". There is a tendency for insufficiently knowledgeable "consumers" of the results to attribute "magical abilities" to data mining, treating the technique as a sort of all-seeing crystal ball. Like any other tool, it only functions in conjunction with the appropriate raw material: in this case, indicative and representative data that the user must first collect. Further, the discovery of a particular pattern in a particular set of data does not necessarily mean that pattern is representative of the whole population from which that data was drawn. Hence, an important part of the process is the verification and validation of patterns on other samples of data.

The term data mining has also been used in a related but negative sense, to mean the deliberate searching for apparent but not necessarily representative patterns in large numbers of data. To avoid confusion with the other sense, the terms data dredging and data snooping are often used

**2.2 Proximity Breeds Danger: Emerging Threats in Metro-area Wireless Networks**

The growing popularity of wireless networks and mobile devices is starting to attract unwanted attention especially as potential targets for malicious activities reach critical mass. In this study, we try to quantify the threat from large-scale distributed attacks on wireless networks, and, more specifically, wifi networks in densely populated metropolitan areas. We focus on three likely attack scenarios: “wildfire” worms that can spread contagiously over and across wireless LANs, coordinated citywide phishing campaigns based on wireless spoofing, and rogue systems for compromising location privacy in a coordinated fashion.

**2.3The Eigen Trust Algorithm for Reputation Management in P2P Networks**

Peer-to-peer file-sharing networks are currently receiving much attention as a means of sharing and distributing information. However, as recent experience shows, the anonymous, open nature of these networks offers an almost ideal environment for the spread of self-replicating inauthentic files. We describe an algorithm to decrease the number of downloads of inauthentic files in a peer-to-peer file-sharing network that assigns each peer a unique global trust value, based on the peer’s history of uploads.

**2.4 On Mobile Viruses Exploiting Messaging and Bluetooth Services**

The exponential growth of mobile messaging worldwide has made it an indispensable tool for social and business interactions. The interoperability between SMS (Short Messaging Service) and IM (Instant Messaging) networks has enabled mobile users to communicate over the Internet seamlessly.

**2.5 An Optimal Distributed Malware Defense System for Mobile Networks with Heterogeneous Devices**

As malware attacks become more frequent in mobile networks, deploying an efficient defense system to protect against infection and to help the infected nodes to recover is important to contain serious spreading and outbreaks. The technical challenges are that mobile devices are heterogeneous in terms of operating systems, and the malware can infect the targeted system in any opportunistic fashion via local and global connectivity, while the to-be-deployed defense system on the other hand would be usually resource limited. In this paper, we investigate the problem of optimal distribution of content-based signatures of malware to minimize the number of infected nodes, which can help to detect the corresponding malware and to disable further propagation.

**2.6 Delegation Forwarding**

Mobile opportunistic networks are characterized by unpredictable mobility, heterogeneity of contact rates and lack of global information. Successful delivery of messages at low costs and delays in such networks is thus challenging. Most forwarding algorithms avoid the cost associated with flooding the network by forwarding only to nodes that are likely to be good relays, using a quality metric associated with nodes. However it is non-trivial to decide whether an encountered node is a good relay at the moment of encounter. Thus the problem is in part one of online inference of the quality distribution of nodes from sequential samples, and has connections to optimal stopping theory. Based on these observations we develop a new strategy for forwarding, which we refer to as delegation forwarding. We analyse two variants of delegation forwarding and show that while naive forwarding to high contact rate nodes has cost linear in the population size, the cost of delegation forwarding is proportional to the square root of population size. We then study delegation forwarding with different metrics using real mobility traces and show that delegation forwarding performs as well as previously proposed algorithms at much lower cost. In particular we show that the delegation scheme based on destination contact rate does particularly well.

**2.7 Performance Analysis of the CONFIDANT Protocol**

Mobile ad-hoc networking works properly only if the par-ticipating nodes cooperate in routing and forwarding. How- ever, it may be advantageous for individual nodes not to cooperate. We propose a protocol, called CONFIDANT, for making misbehavior unattractive; it is based on selective altruism and utilitarianism. It aims at detecting and isolating misbehaving nodes, thus making it unattractive to deny cooperation. Trust relationships and routing decisions are based on experienced, observed, or reported routing and forwarding behavior of other nodes. The detailed implementation of CONFIDANT in this paper assumes that the net- work layer is based on the Dynamic Source Routing (DSR) protocol. We present a performance analysis of DSR formed by CONFIDANT and compare it to regular defenseless DSR. It shows that a network with CONFIDANT and up to 60% of misbehaving nodes behaves almost as well as a benign network, in sharp contrast to a defenseless network. All simulations have been implemented and performed in GloMoSim.

**3. ANALYSIS**

**3.1 Requirements**

Project requirements are conditions or tasks that must be completed to ensure the success or completion of the project. They provide a clear picture of the work that needs to be done. They're meant to align the project's resources with the objectives of the organization.

**3.1.1 Functional Requirements**

**Service Provider**

The Service Provider browses the required file, initializes nodes with digital signature and uploads to the end user (node a, node b, node c, node d, node e, node f) via Router.

**Router**

The Router is responsible for forwarding the data file in shortest distance to the destination; the Router consists of Group of nodes, the each and every node (n, n2,n3,n4,n5,n6,n7,n8,n8,n10,n11, n12, n13) consist of Bandwidth and Digital Signature. If router had found any malicious or traffic node in the router then it forwards to the IDS Manager. In Router we can assign the bandwidth for the nodes and can view the node details with their tags Node Name, Sender IP, Injected data, Digital Signature, Bandwidth and status.

**Attacker**

The malicious node or the traffic node details can be identified by a threshold-based classifier is employed in the Attack Detection module to distinguish DoS attacks from legitimate traffic. The Attacker can inject the fake message and generates the signature to a particular node in the router with the help of threshold-based classifier in testing phase and then adds to the attacker profile.

**3.1.2 Non-Functional Requirements**

Accessibility: The interface provided to the client makes it easy for basic operations .

Efficiency: The developer only needs to import necessary files for the interface to work .

Exploitability: The code and interface can be explored to any extent . There are no restrictions.

Extensibility: The project includes algorithms which can be extended without messing up the previous code.

Stability: The interface is very stable leading to much lesser errors occurring in the process. Alert generation automatically halts the program.

Scalability: The theory introduced in our project can be applied from any autonomous network to the largest network the internet.

Platform compatibility: The project works on any edition of Microsoft Windows operating system .

**4.1 Software Requirements**

* Operating system : Windows XP/7.
* Coding Language : JAVA/J2EE
* IDE : Net beans 7.4
* Database : MYSQL

**4.2 Hardware Requirements**

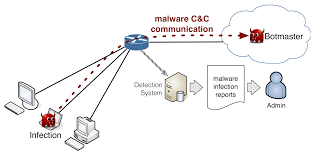
* System : Pentium IV 2.4 GHz.
* Floppy Drive : 1.44 Mb.
* Monitor : 15 VGA Colour.
* Ram : 512 Mb.

**5. SYSTEM DESIGN**

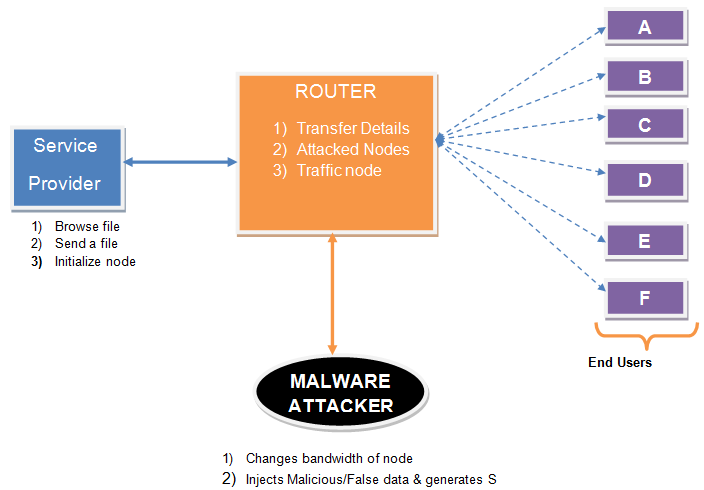
**5.1 Introduction**

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. ADFD is often used as a preliminary step to create an overview of the system, which can later be elaborated.

**5.2 System Architecture:**



**5.3 Block Diagram**

****

**4.4 Modules**

**4.4.1 Service Provider**

In this module, the Service Provider browses the required file, initializes nodes with digital signature and uploads to the end user (node a, node b, node c, node d, node e, node f) via Router.

**4.4.2 Router**

The Router is responsible for forwarding the data file in shortest distance to the destination; the Router consists of Group of nodes, the each and every node (n, n2,n3,n4,n5,n6,n7,n8,n8,n10,n11, n12, n13) consist of Bandwidth and Digital Signature. If router had found any malicious or traffic node in the router then it forwards to the IDS Manager. In Router we can assign the bandwidth for the nodes and can view the node details with their tags Node Name, Sender IP, Injected data, Digital Signature, Bandwidth and status.

**4.4.3 Attacker**

In this module, the malicious node or the traffic node details can be identified by a threshold-based classifier is employed in the Attack Detection module to distinguish DoS attacks from legitimate traffic. The Attacker can inject the fake message and generates the signature to a particular node in the router with the help of threshold-based classifier in testing phase and then adds to the attacker profile.

**4.4.4 Epidemic model For Detecting Malware spreading**

The IDS manager is nothing but Intrusion Detection System manager which is responsible to filter the malicious data and traffic data. The IDS manager decides the phases based on Router status and then decides on two phases i.e., the “Training Phase” and the “Test Phase”.

**Training Phase**

The Normal Profile Generation module is operated in the Training Phase to generate profiles for various types of legitimate traffic records, and the generated normal profiles are stored in a database.

**Test Phase**

The Tested Profile Generation module is used in the Test Phase to build profiles for individual observed traffic records. Then, the tested profiles are handed over to the Attack Detection module, which compares the individual tested profiles with the respective stored normal profiles.

**4.4.5 End User**

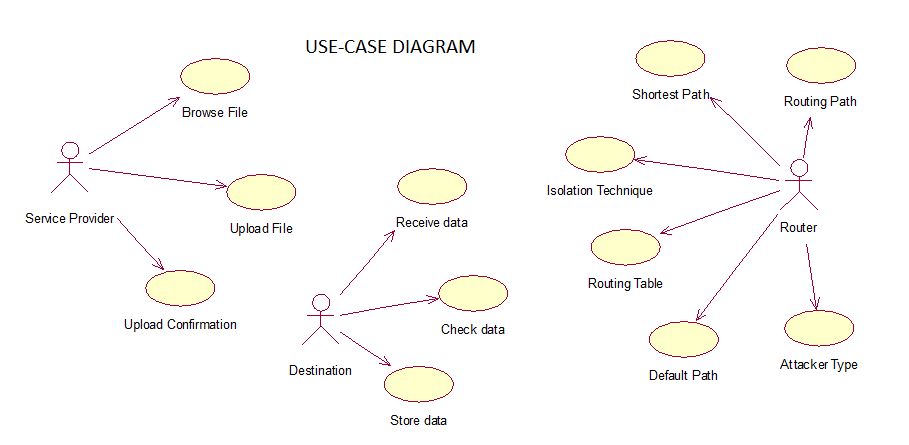
In this module, the End user can receive the data file from the Service Provider which is sent via Router, if malicious or traffic node is found in the router then it forwards to the IDS Manager to filter the content and adds to the attacker profile.

**4.5 UML Diagram**

UML is a method for describing the system architecture in detail using the blueprint. UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. UML is a important part of developing objects oriented software and software development process.UML uses mostly graphical notation to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software. UML is a general-purpose visual modeling language that is used to specify, visualize, construct, and document the artifacts of the software system. It will provide vocabulary and rules for communications and function on conceptual and physical representation. So it is modeling language. It means building models that are precise, unambiguous and complete. In particular the UML address the specification of all the important analysis, design and implementation decisions that must be made in developing and displaying a software intensive system. The UML includes both graphical and textual representation. It makes easy visualize the system and for better understanding. UML modes can be directly connected to a variety of programming languages and is sufficiently expressive and free from any ambiguity to permit the direct execution models.

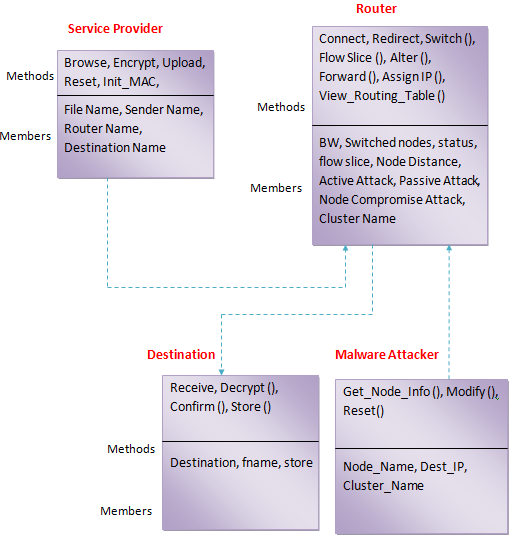
**4.5.1 Use case Diagram**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

****

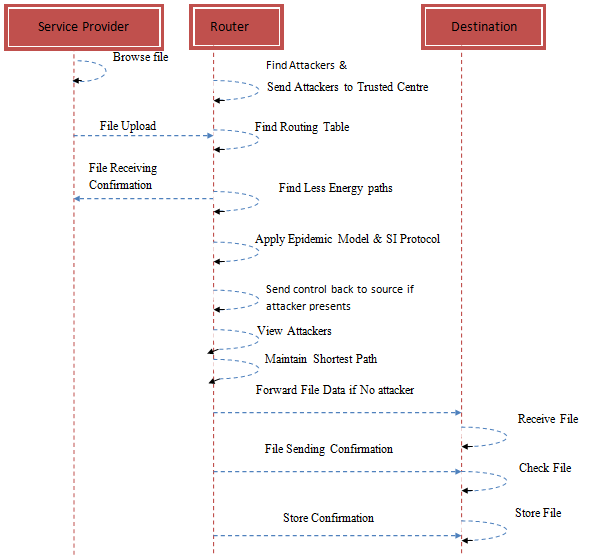
**4.5.2 Class Diagram**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

****

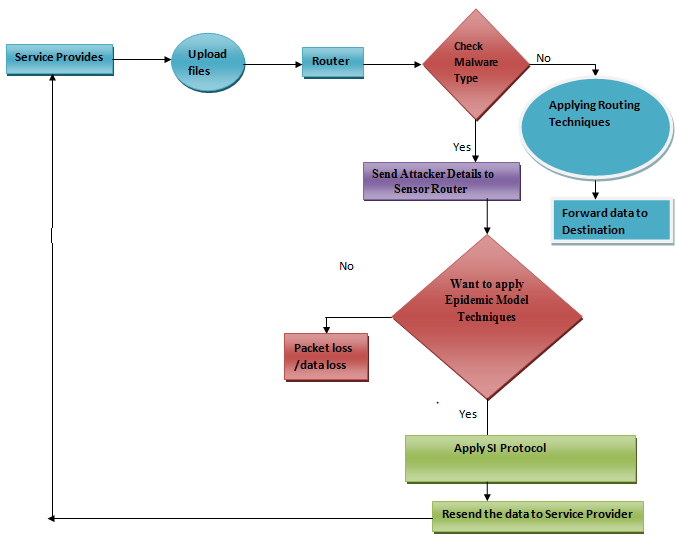
**4.5.3 Sequence Diagram**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

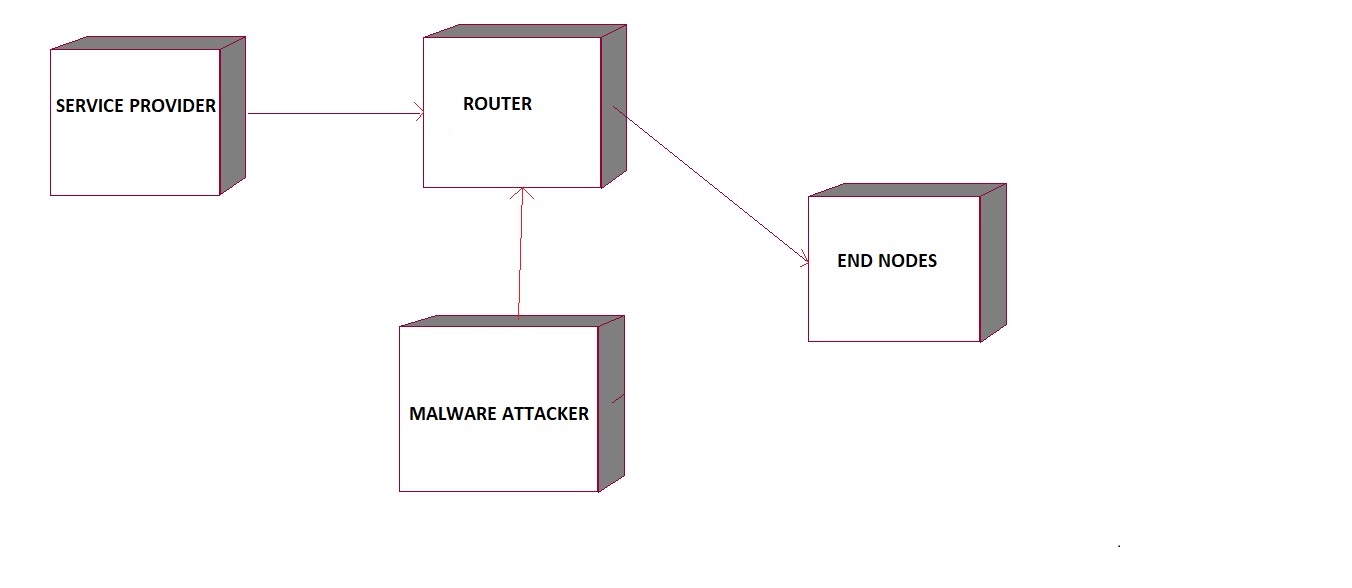
****

**4.5.4 Activity Chart**

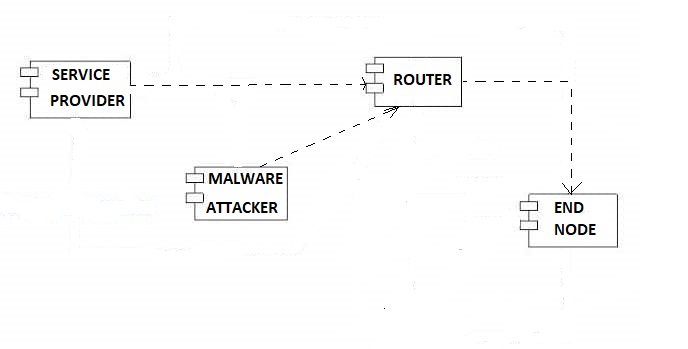
The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system



**4.5.5 Deployment Diagram**

****

**4.5.6 Component Diagram**

****

**5. IMPLEMENTATION AND CODING**

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.

Early stage- An early stage of the breakout of a malware means only a small percentage of vulnerable hosts have been compromised, and the propagation follows exponential distributions.

Final stage- The final stage of the propagation of a malware means that all vulnerable hosts of a given network have been compromised.

Late stage- A late stage means the time interval between the early stage and the final stage.

**Attacker.java**

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Container;

import java.awt.Desktop;

import java.awt.Font;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.BufferedInputStream;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.PrintStream;

import java.math.BigInteger;

import java.net.InetAddress;

import java.net.Socket;

import java.security.DigestInputStream;

import java.security.MessageDigest;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JScrollPane;

import javax.swing.JTextField;

import javax.swing.UIManager;

import java.awt.\*;

import javax.swing.\*;

public class Attacker extends JFrame implements ActionListener

{

Container c;

JButton jb,jb1,jb2,jb3;

JLabel al;

//JTextArea tf;

JTextField t1,t2;

String keyword = "ef50a0ef2c3e3a5fdf803ae9752c8c66";

public Font f1 = new Font("Times new roman", Font.BOLD, 18);

String hash;

String data;

Attacker()

{

JScrollPane pane = new JScrollPane();

setTitle("Malicious Attacker::Malware Propagation in Large-Scale Networks");

c = getContentPane();

c.setLayout(null);

c.setBackground(Color.yellow);

al=new JLabel("Enter the Data to Inject");

al.setForeground(Color.red);

jb = new JButton("Attack Dataset");

jb.setBounds(50,450,120,30);

al.setBounds(130,-10,150,100);

jb3 = new JButton("Generate Sign");

jb3.setBounds(200,150,120,30);

jb1 = new JButton("Malicious Attack Sensor");

jb1.setBounds(350,150,130,30);

jb2 = new JButton("Exit");

jb2.setBounds(500,150,100,30);

jb.addActionListener(this);

jb1.addActionListener(this);

jb2.addActionListener(this);

jb3.addActionListener(this);

//c.add(jb);

c.add(jb1);

c.add(jb2);

c.add(jb3);

c.add(al);

/\* tf = new JTextArea();

tf.setColumns(100);

tf.setForeground(Color.black);

tf.setFont(f1);

tf.setRows(200);

tf.setName("tf");

pane.setName("pane");

pane.setViewportView(tf);

pane.setBounds(250, 50, 350, 350);\*/

t1=new JTextField();

t1.setFont(f1);

t1.setForeground(Color.blue);

t1.setBounds(250, 90, 300, 30);

t2=new JTextField();

t2.setFont(f1);

t2.setForeground(Color.blue);

t2.setBounds(250, 30, 300, 30);

c.add(pane, BorderLayout.CENTER);

c.add(t1);

c.add(t2);

setSize(650,250);

setVisible(true);

}

public void actionPerformed(ActionEvent e)

{

if(e.getSource()==jb)

{

}

if(e.getSource()==jb3)

{

try

{

data=t2.getText();

PrintStream p = new PrintStream(new FileOutputStream("Attacker\\attack.txt"));

p.print(new String(data));

MessageDigest md = MessageDigest.getInstance("SHA1");

FileInputStream fstream1 = new FileInputStream("Attacker\\attack.txt");

DigestInputStream dis1 = new DigestInputStream(fstream1, md);

BufferedInputStream bis1 = new BufferedInputStream(dis1);

while (true) {

int b1 = bis1.read();

if (b1 == -1)

break;

}

BigInteger bi1 = new BigInteger(md.digest());

hash = bi1.toString(16);

t1.setText(hash);

}

catch(Exception e1)

{

e1.printStackTrace();

}

}

if(e.getSource()==jb1)

{

try

{

String[] dsname = { "Select Nodes", "N4", "N5", "N10", "N11" };

String dataname = (String) JOptionPane.showInputDialog(null,

"Select Nodes", "Node Name",

JOptionPane.QUESTION\_MESSAGE, null, dsname, dsname[0]);

String ip = JOptionPane.showInputDialog(null,

"Please Enter the IP Address Of Router");

InetAddress ia = InetAddress.getLocalHost();

String ip1= ia.getHostAddress();

Socket sc1 = new Socket(ip,442);

DataOutputStream dout1 = new DataOutputStream(sc1.getOutputStream());

dout1.writeUTF(dataname);

dout1.writeUTF(hash);

dout1.writeUTF(data);

dout1.writeUTF(ip1);

DataInputStream oin = new DataInputStream(sc1.getInputStream());

String msg1 = (String) oin.readUTF();

if(msg1.equalsIgnoreCase("success"))

{

JOptionPane.showMessageDialog(null, "Attack Completed");

}

else if(msg1.equalsIgnoreCase("failure"))

{

JOptionPane.showMessageDialog(null, "Attack Failure" );

}

}

catch(Exception e1)

{

e1.printStackTrace();

}

}

if(e.getSource()==jb2)

{

System.exit(0);

}

}

public static void main(String[] args) {

try {

UIManager

.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");

} catch (Exception e1) {

e1.printStackTrace();

}

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new Attacker();

}

});

}

**Dbcon.java**

import java.sql.Connection;

import java.sql.DriverManager;

public class DBCon

{

static Connection con;

public Connection getConnection()

{

try

{

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

con = DriverManager.getConnection("jdbc:odbc:Driver={Microsoft Access Driver (\*.mdb)};Dbq=src\\Database.mdb");

System.out.println("Connected");

}

catch(Exception e)

{

e.printStackTrace();

}

return con;

}

}

**6. TEST CASES AND OUTPUT SCREENSHOTS**

|  |  |
| --- | --- |
| **Figure Name** | **Figure Number** |
| 6.1.Selecting a Java File   1. Go to windows explorer. 2. Select required java file. 3. Select open. 4. File containing datasets is selected. | Fig 6.1 |
| 6.2.Trusted Case   1. Service provider sends a file from source to destination. | Fig 6.2 |
| (i)Test case  File received by the destination if legitimate. | Fig 6.2.1 |
| 6.3.Traffic case   1. Service provider sends a file from source to destination. | Fig 6.3, Fig 6.3(a) |
| (i)Test case  File received by the destination if legitimate.  Otherwise error blocks the file. | Fig 6.3.1, Fig 6.3.1(a) |
| 6.4.Malware Spreading Case   1. Service provider sends a file from source to destination. 2. Router blocks the file if hash does not match. | Fig 6.4, Fig 6.4(a) |
| (i)Test case  File received by the destination if legitimate.  Otherwise error blocks the file.  Signature is displayed. | Fig 6.4.1, Fig 6.4.1(a) |

**6.1 Selecting a Java file**

Select any one java file from database. After that IP address is given for the router. Then nodes will be initiated successfully. Then select a destination file and enter the IP address of destination file. There are three types of malware which occurs in a network

1. Trusted Case
2. Traffic Case
3. Malware Spreading

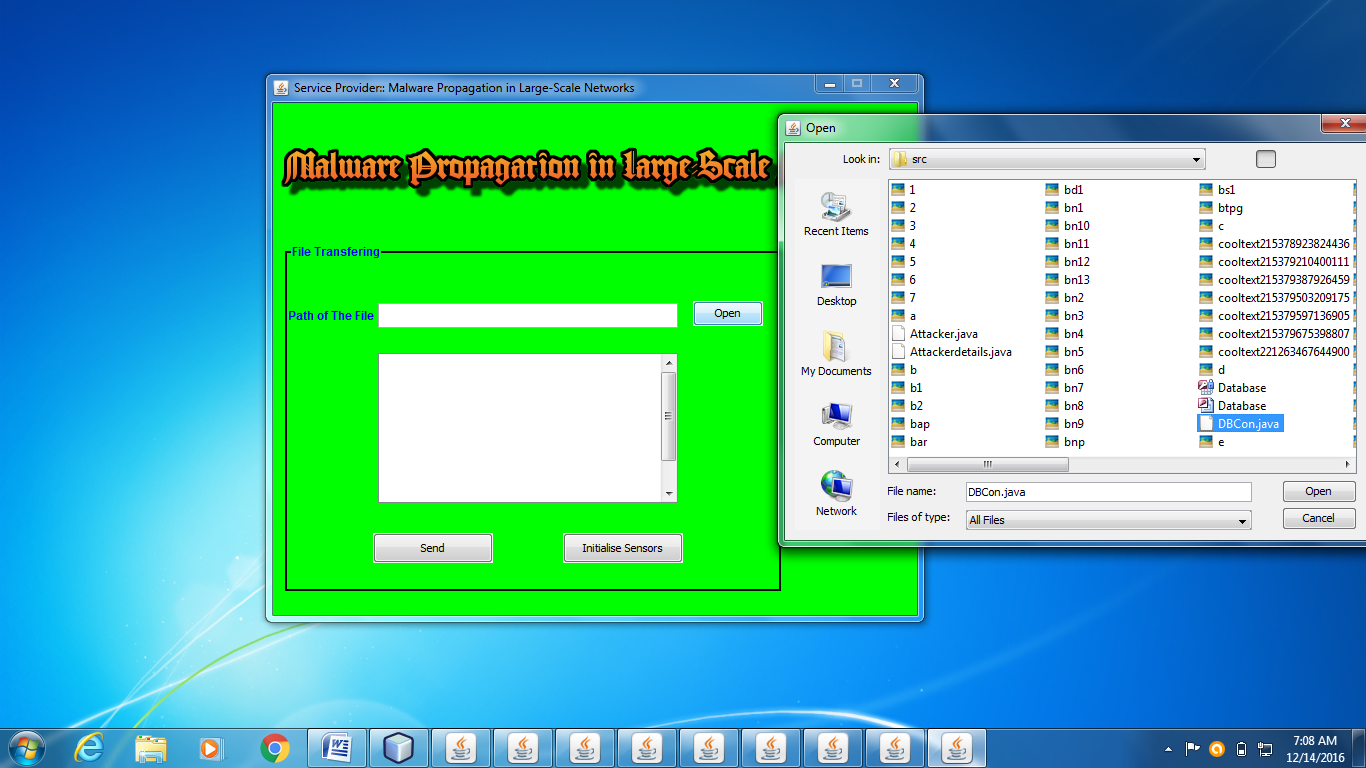


Fig 6.1

**6.2 Trusted Case**

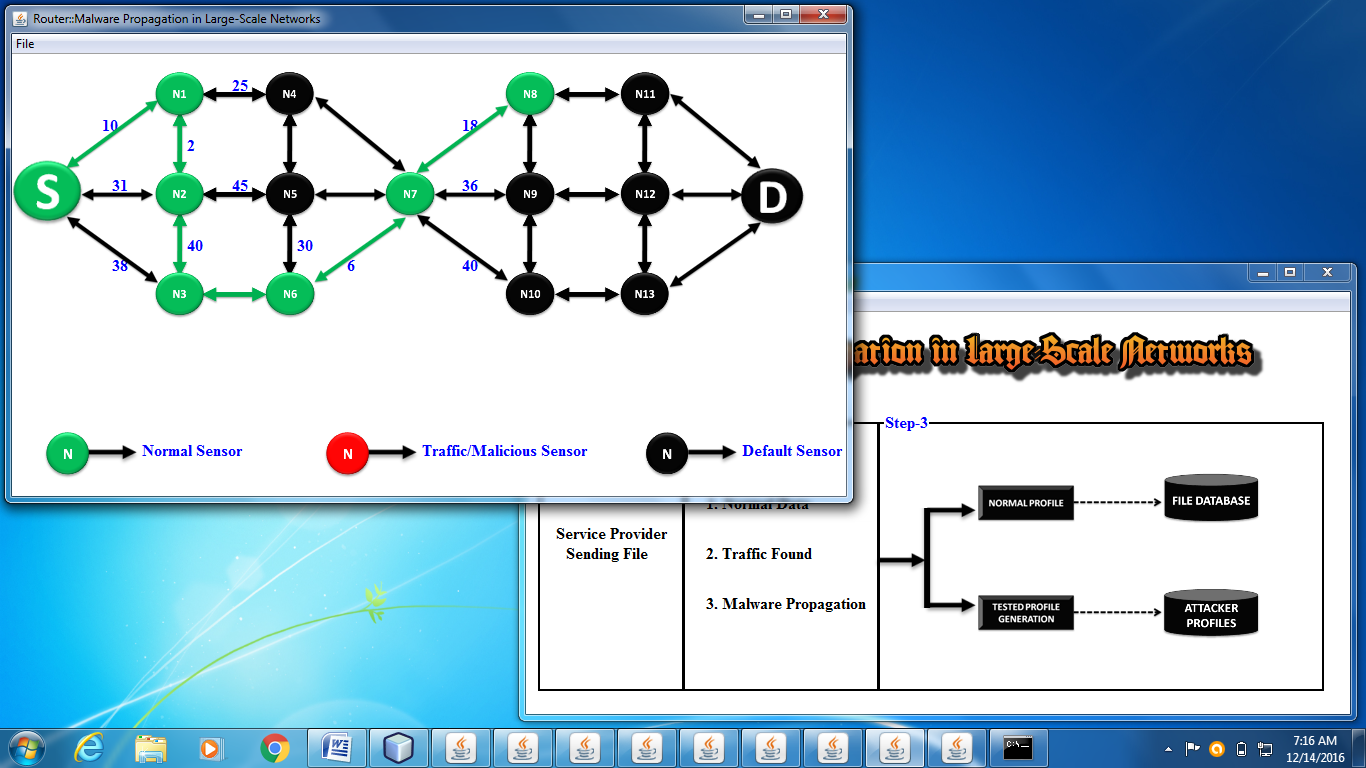


Fig 6.2

Whenever a service provider sends a file from source to destination in Trusted Case, the status of the router will flow as a normal data and in reading phase the data is shown as a flow from normal profile to file database and is shown as normal data found. If no error occurs then data received successfully by destination

**6.2.1 Test Case of Trusted Case**

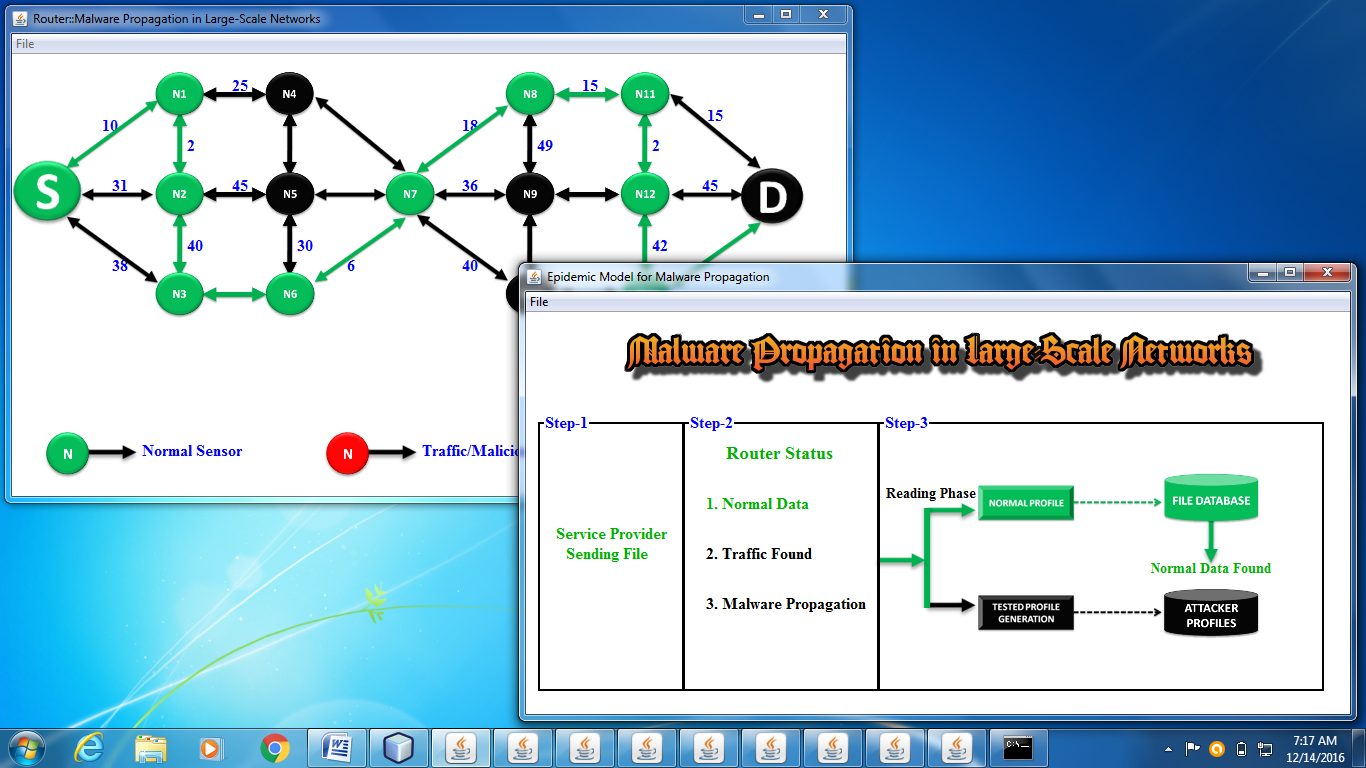


Fig 6.2.1

If any error occurs the flow of data in reading phase flows through tested profile generation to attacker profiles

**6.3 Traffic Case**

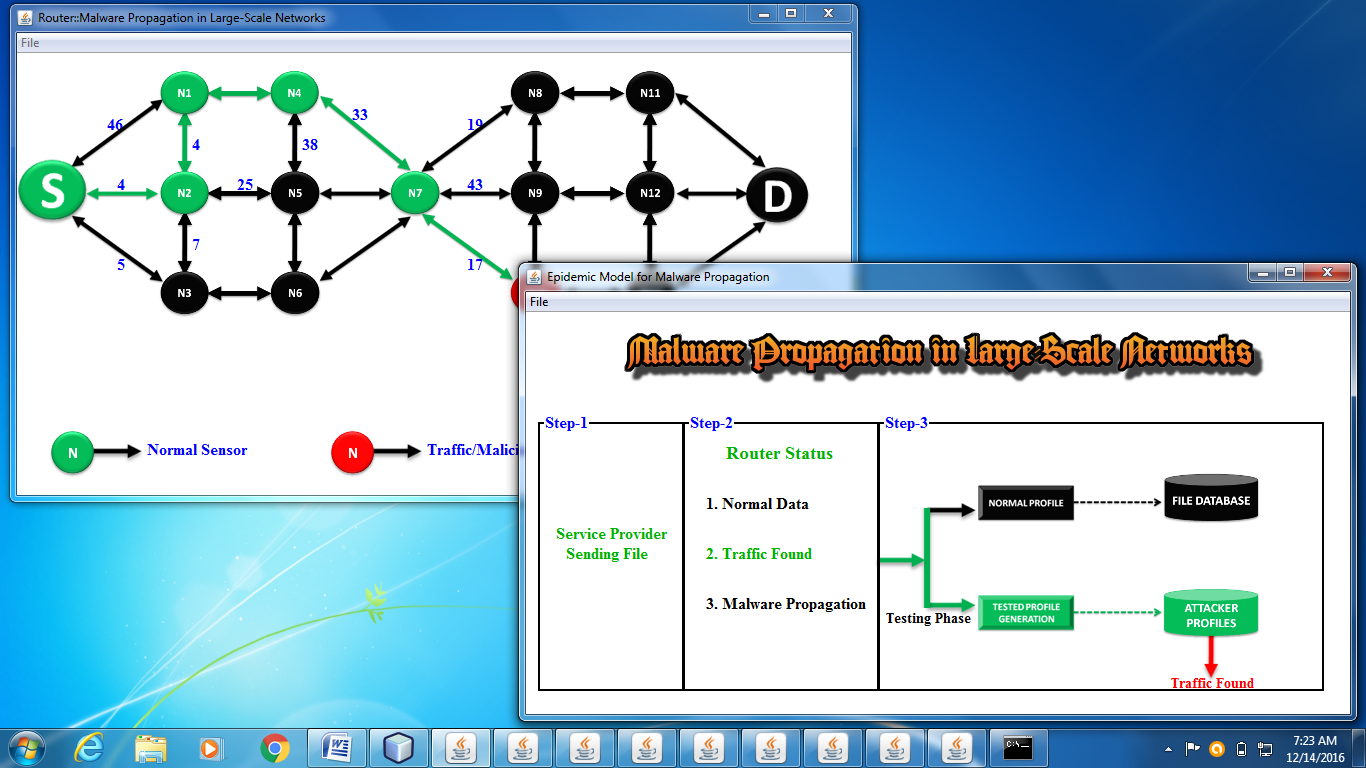
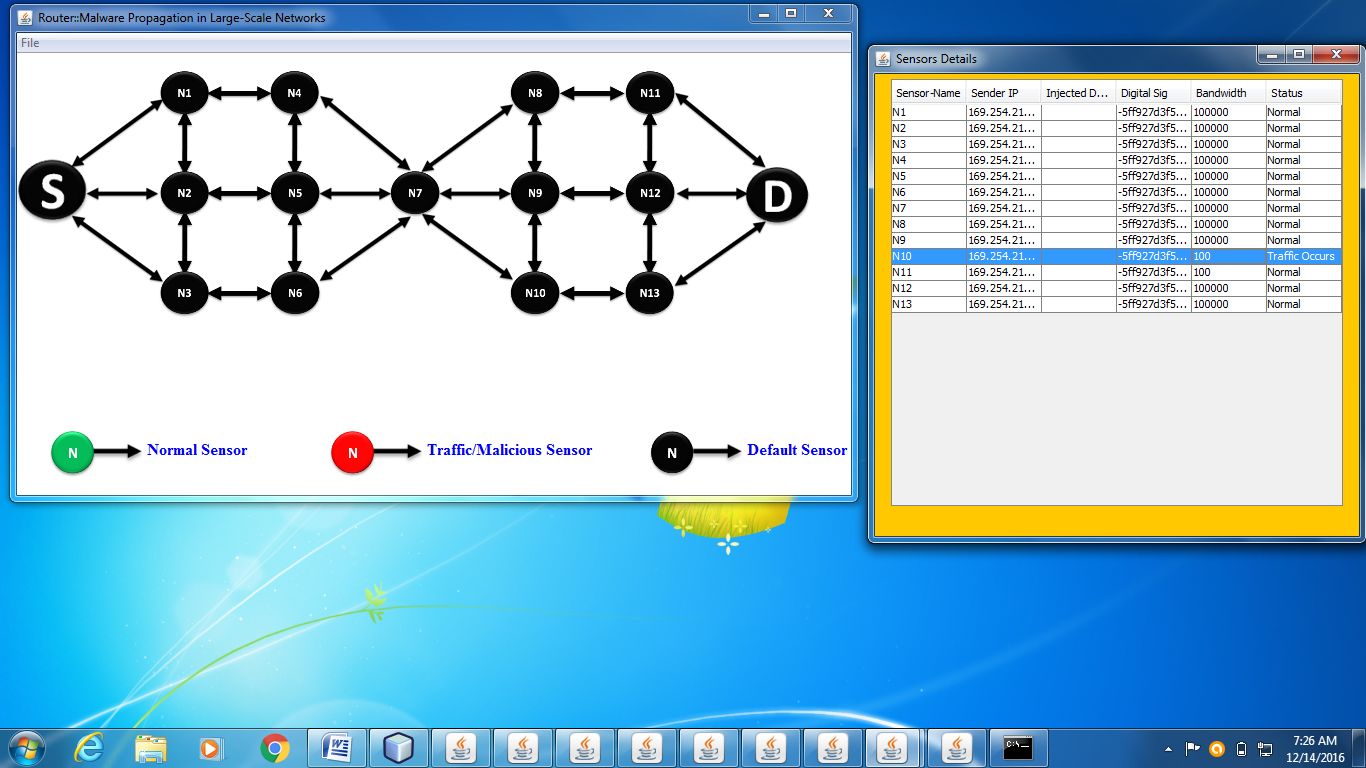
****

Fig 6.3

Whenever a service provider sends a file from source to destination in Traffic Case, the status of the router will flow as a traffic found. If error occurs then data is not received by destination. Then is travels from other node which has no traffic and file is sent from source to destination.

 Fig 6.3(a)

**6.3.1 Test Case in Traffic Case**

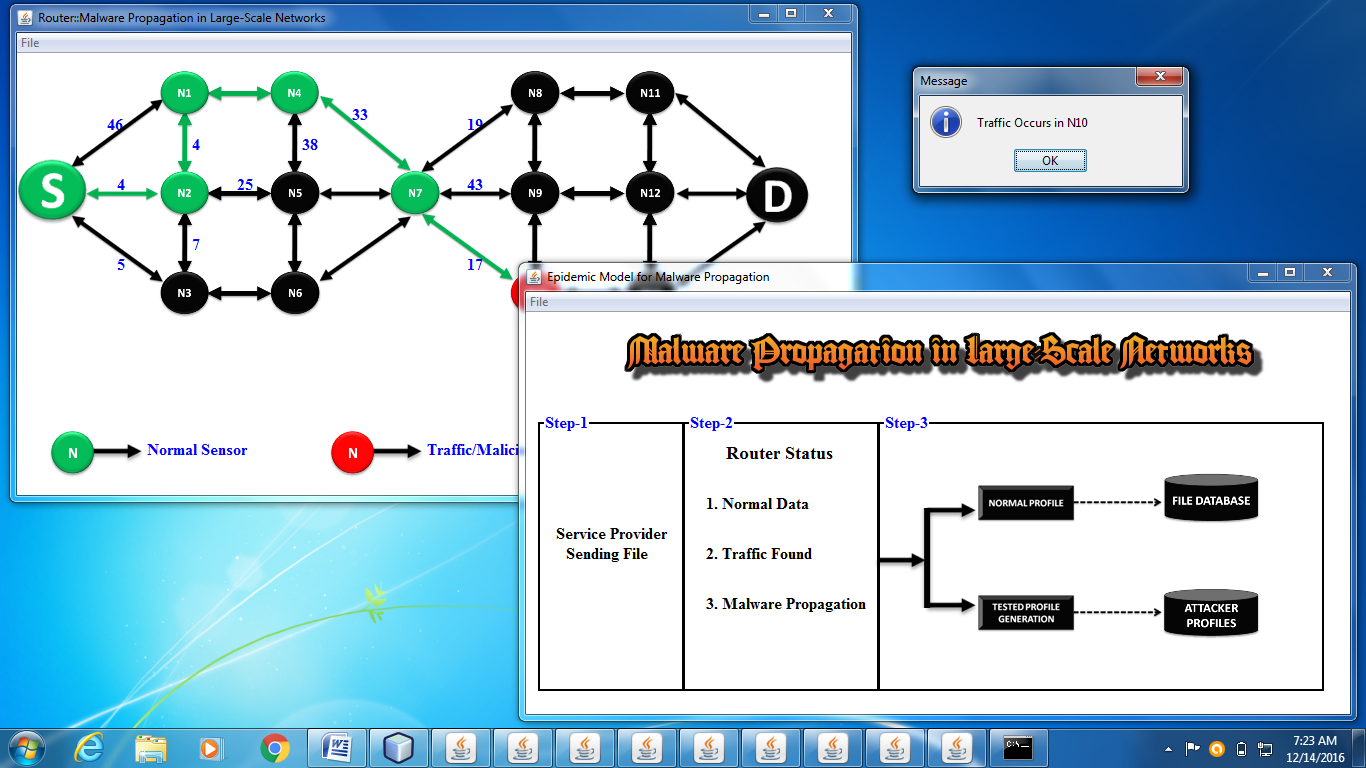


Fig 6.3.1

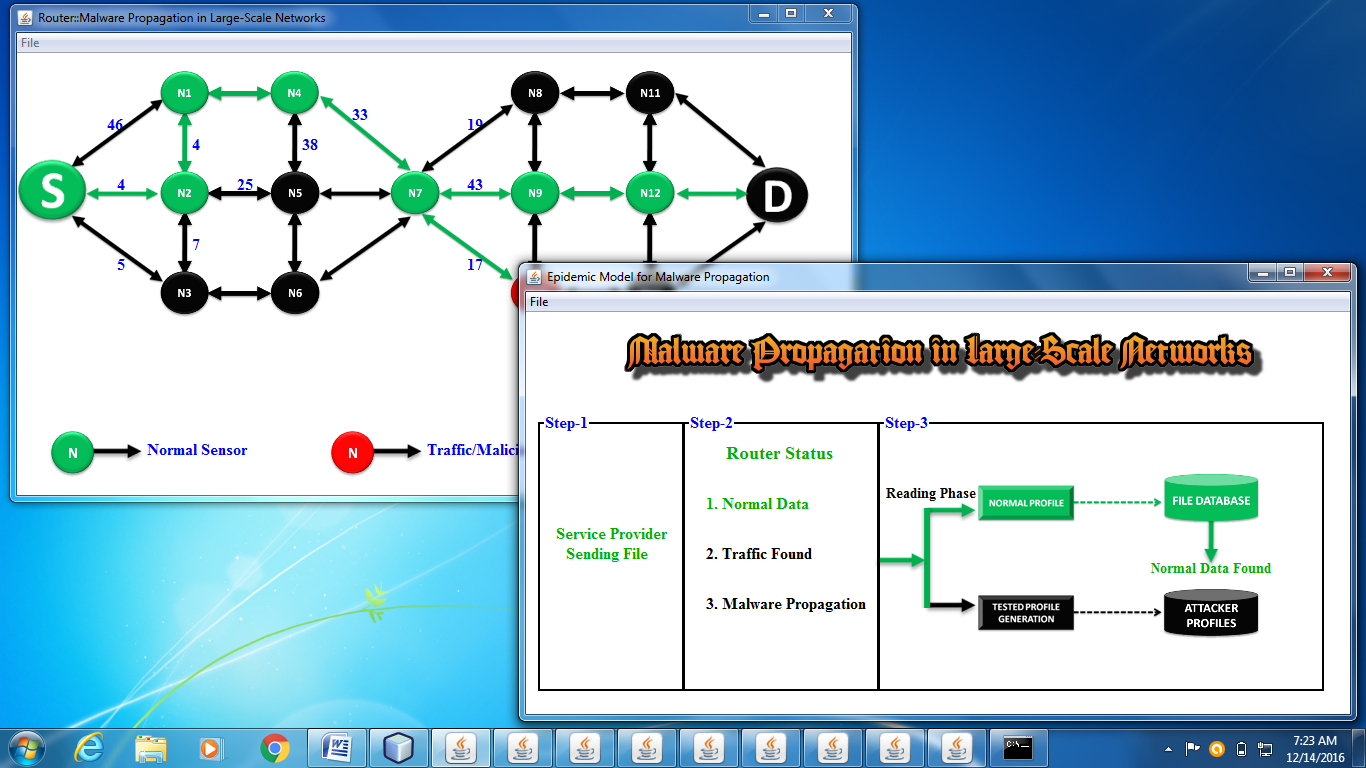


Fig 6.3.1(a)

If any error occurs the flow of data in reading phase flows through tested profile generation to attacker profiles. File is received by destination through another traversal.

**6.4 Malware Spreading Case**

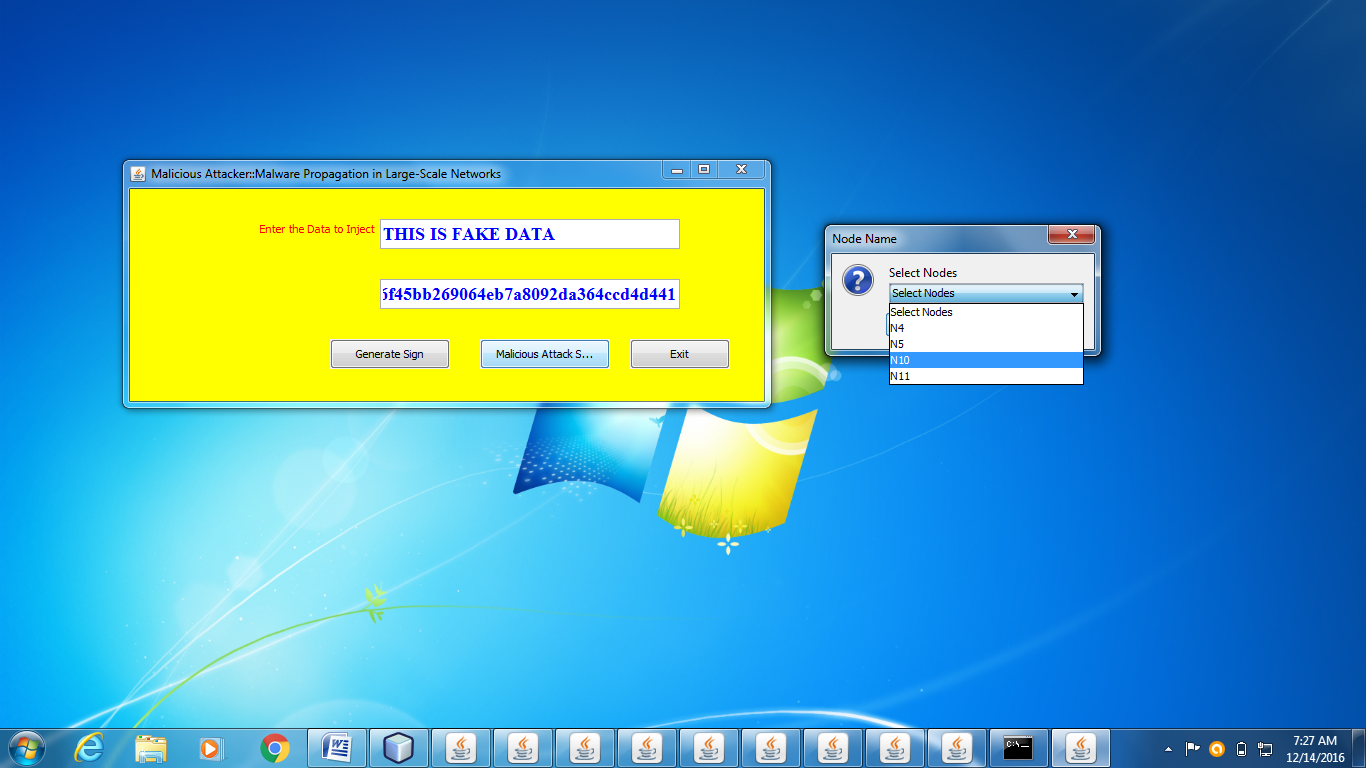


Fig 6.4

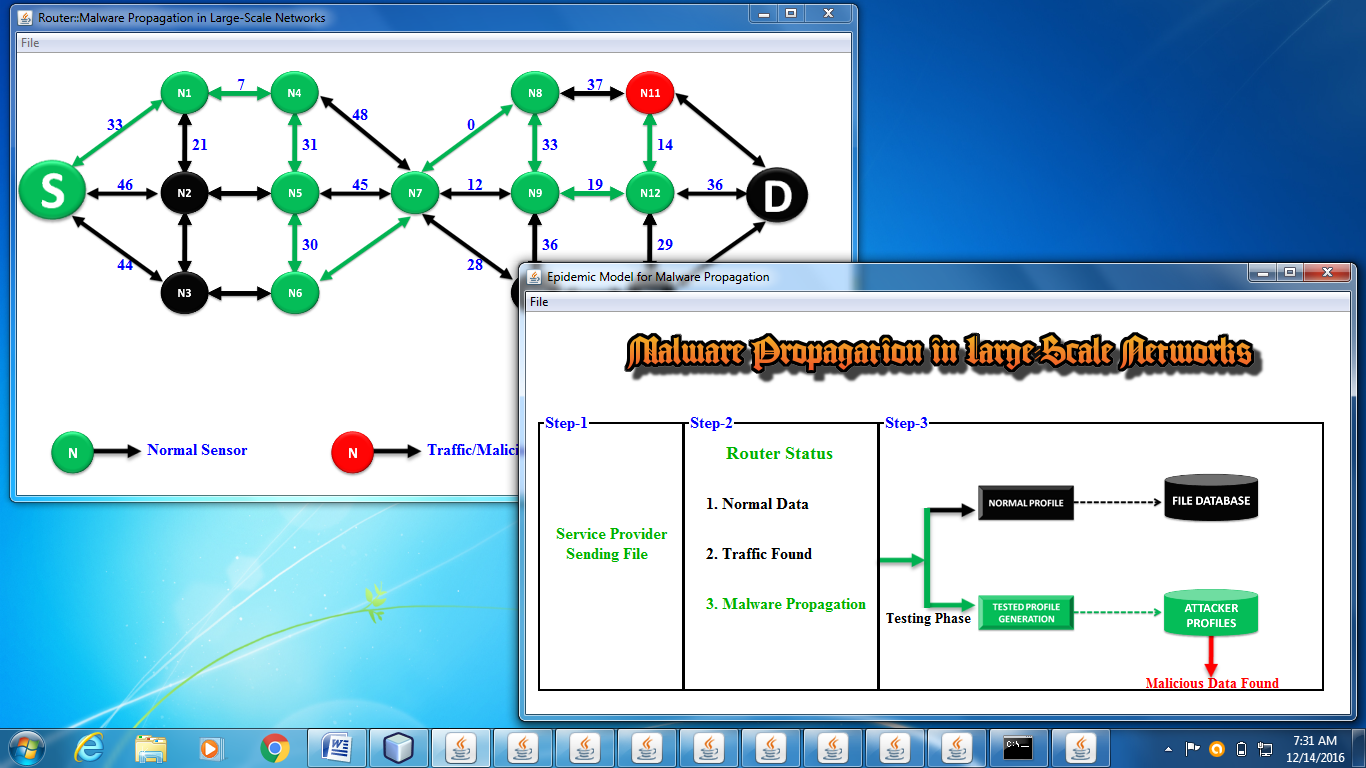
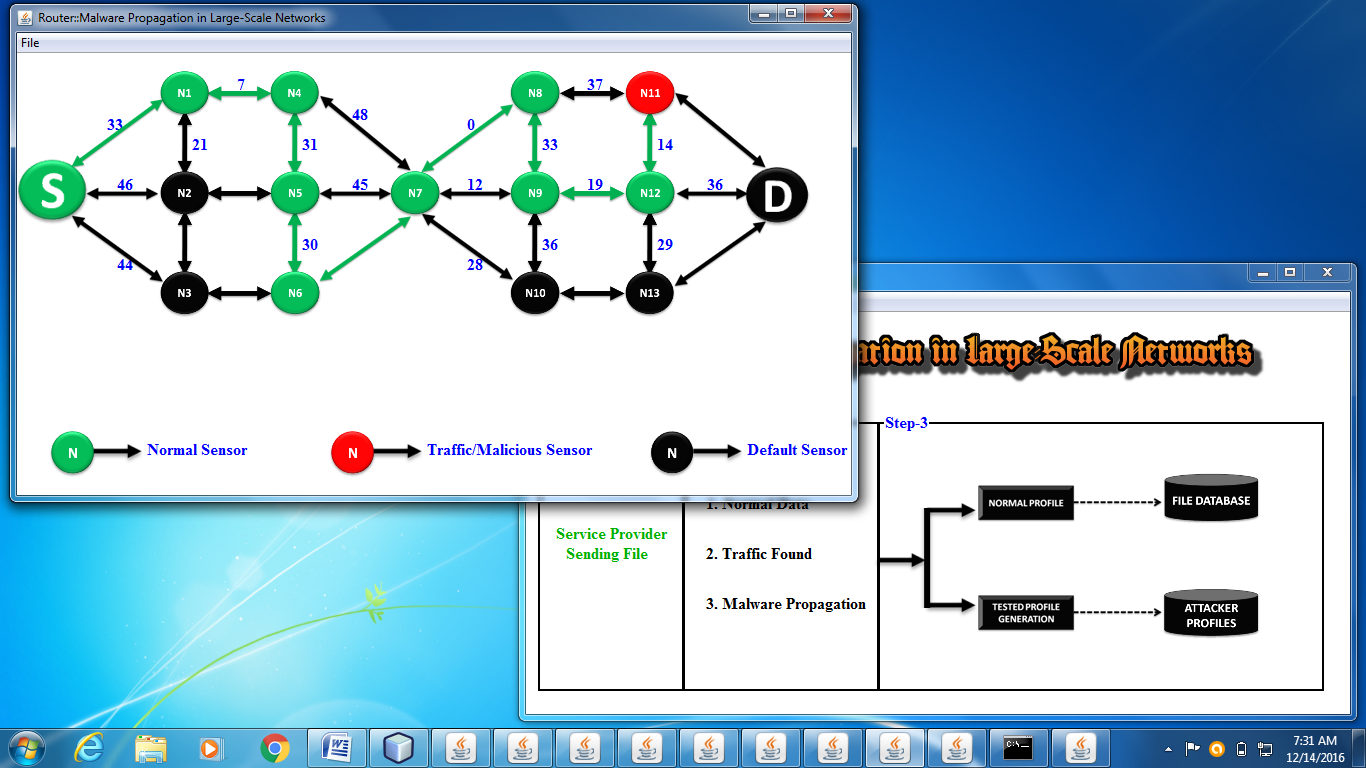


Fig 6.4(a)

Whenever a service provider sends a file from in malware spreading case, the status of router will flow as malware data found. In this case we need to inject a duplicate data

**6.4.1 Test Case in Malware Spreading Case**

**** Fig 6.4.1

Due to error the flow of data in reading phase flows through tested profile generation to attacker profiles. The red node indicates that it contains either traffic or malware.

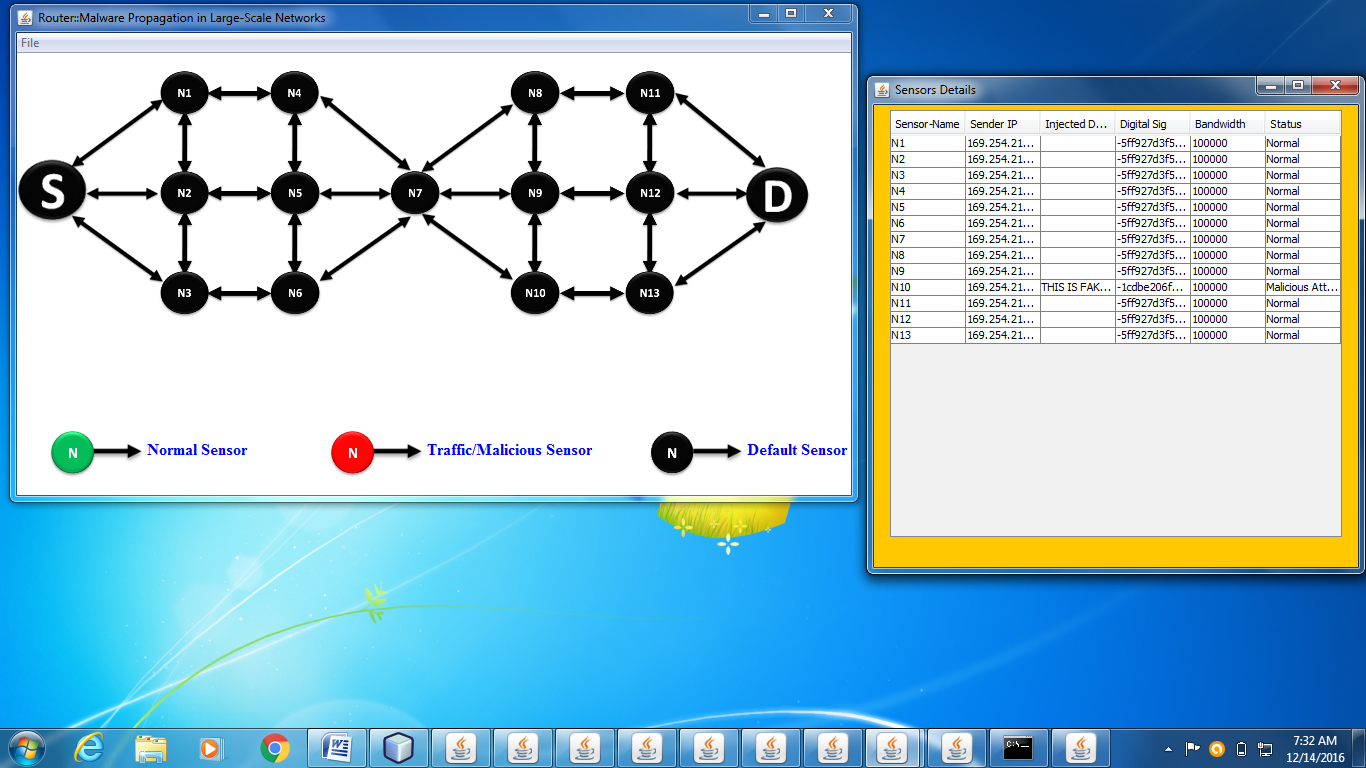


Fig 6.4.1(a)

It shows the malware propagation in nodes. Nodes are attacked by malware. It goes through a alternative way not to be attacked by malware. If malware is attacked it also spreads to the other nodes. Thus by applying proposed methods file is sent to destination.

**7. CONCLUSION AND FUTURE ENHANCEMENT**

In this paper, 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.

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.

.

**REFERENCES**

**1.** Phillip Lee, Student Member, IEEE, Andrew Clark, Member, IEEE, Basel Alomair, Member, IEEE, Linda Bushnell, Senior Member, IEEE, and Radha Poovendran, Fellow, IEEE, “Adaptive Mitigation of Multi-Virus Propagation: A Passivity-Based Approach”

**2.**  B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna, “Your botnet is my botnet: Analysis of a botnet takeover,” in CCS ’09: Proceedings of the 2009 ACM conference on computer communication security, 2009.

**3.**  D. Dagon, C. Zou, andW. Lee, “Modeling botnet propagation using time zones,” in Proceedings of the 13 th Network and Distributed System Security Symposium NDSS, 2006.

**4.** M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis, “My botnet is bigger than yours (maybe, better than yours): why size estimates remain challenging,” in Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets, 2007.

**5.** Java The Complete Reference , Book by Herbert Schilbt , 1996

**6.** SQL Quick Start Guide The Simplified Beginner Guide To SQL , Book by Clydebank

Technology , 11 March 2015.