PREVENTING DISTRIBUTED DENIAL-OF-SERVICE FLOODING ATTACKS WITH DYNAMIC PATH IDENTIFIERS

A MAJOR PROJECT REPORT

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

SUBMITTED BY

VENKAT SAI CHARAN BANDI 2210315763

UNDER THE GUIDANCE OF

Mr.S.Durga Prasad

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF TECHNOLOGY
GITAM (DEEMED TO BE UNIVERSITY)
HYDERABAD CAMPUS
April-2019

SCHOOL OF TECHNOLOGY
GITAM (DEEMED TO BE UNIVERSITY)
(Estd. u/s 3 of UGC Act 1956)
HYDERABAD CAMPUS

Rudraram Village, Patancheru Mandal,
Sangareddy Dist – 502329, TS, INDIA.
Ph: 08455-220556/57; Fax : 08455-20046
Website : www.gitam.edu

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the major project entitled **PREVENTING DISTRIBUTED DENIAL OF SERVICE FLOODING ATTACKS WITH DYNAMIC PATH IDENTIFIERS** was presented satisfactorily to Department of Computer Science and Engineering, SCHOOL OF TECHNOLOGY, GITAM (DEEMED TO BE UNIVERSITY), HYDERABAD by **VENKAT SAI CHARAN BANDI** bearing H.T-NO **2210315763** in partial fulfillment of requirement for their major project work carried out under my guidance and supervision.

[illegible]
Mr. S. Durga Prasad
Project Guide

[illegible] 2/4/19
B.Rajendra Prasad Babu
Project Coordinator

GITAM * (DEEMED TO BE UNIVERSITY) Hyderabad 502329

[illegible] 02/04/19
Dr.S.Phani kumar
Head of the Department
Dept of CSE

Name and Signature of the project panel members.

1. Sowmya [illegible]
2. G. Sridhar Reddy [illegible]
3. D. Koteswararao - [illegible]

DECLARATION

I, **Venkat Sai Charan Bandi** bearing roll number **2210315763** hereby declare that the major project report entitled **"PREVENTING DISTRIBUTED DENIAL OF SERVICE FLOODING ATTACKS WITH DYNAMIC PATH IDENTIFIERS"**, under the guidance of **Mr. S. Durga Prasad, Assistant Professor,** Department of Computer Science and Engineering, School of Technology, GITAM (Deemed to be University), Hyderabad, has been submitted major project evaluation. This is a record of Bonafide work carried out by us and the content embodied in this project have not been reproduced /copied from any source. The content embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

SIGNATURE:

Venkat Sai Charan Bandi **2210315763** [illegible]

ACKNOWLEDGMENT

The satisfaction and euphoria that accompany the successful completion of task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement always boosted the morale. We take a great pleasure in presenting a project, which is the result of a studied blend of both research and knowledge.

We first take the privilege to thank the Head of our Department, **Dr.S. Phani Kumar**, for permitting us in laying the first stone of success and providing the lab facilities, we would also like to thank the other staff in our department and lab assistants who directly or indirectly in successful completion of the project.

We feel great to thank, **Mr. S. Durga Prasad**, who is our project guide and who shared his valuable knowledge with us and made us understand the real essence of the topic and created interest in us to work.

CONTENTS

ABSTRACT

In recent years, there are increasing interests in using path identifiers (PIDs) as inter-domain routing objects. However, the PIDs used in existing approaches are static, which makes it easy for attackers to launch distributed denial of service (DDoS) flooding attacks. To address this issue we design, implement, and evaluate D- PID, a framework that uses PIDs negotiated between neighboring domains as inter- domain routing objects. In DPID, the PID of an inter-domain path connecting two domains is kept secret and changes dynamically.

We describe in detail how neighboring domains negotiate PIDs, how to maintain ongoing communications when PIDs change. We build a 42-node prototype comprised by six domains to verify D-PID's feasibility and conduct extensive simulations to evaluate its effectiveness and cost. The results from both simulations and experiments can be aided to show that D-PID can effectively prevent DDoS attacks.

LIST OF FIGURES

1. INTRODUCTION

Distributed Denial-of-Service (DDoS) flooding attacks are very harmful to the Internet. In a DDoS attack, the attacker uses widely distributed zombies to send a large amount of traffic to the target system, thus preventing legitimate users from accessing to network resources.

For example, a DDoS attack against BBC sites in Jan. 2016 reached 602 gigabits per second and "took them down for at least three hours" More recently, the hosting provider OVH suffered a large scale DDoS attack in Sep. 2016, launched by a botnet composed at least of 150,000 Internet-of-things (IoT) devices. This attack peaked at nearly one terabit per second (Tbps) and even forced Akamai to stop offering DDoS protection to OVH. Therefore, many approaches have been proposed in order to prevent DDoS flooding attacks, including network ingress filtering, IP trace back, capability-based designs, and shut-up messages.

1.1 MOTIVATION

We make a introduction to CoLoR because it lays the foundation of our PROJECT. We then describe why we should dynamically change PIDs, thus PROVIDING MOTIVATION

A. CoLoR

CoLoR is a receiver-driven information centric network architecture that assigns unique and persistent content names(or service identifiers, SIDs) to content chunks. CoLoR assigns intrinsic secure self-certifying node identifiers (NIDs) to network nodes and ASes so that authenticating a node/AS does not require an external authority such as ICANN, thus improving security and privacy.

In addition, two neighboring domains negotiate a PID for every inter-domain path between them and the PID is only known by them. The two domains then use the PIDs assigned to their inter domain paths to forward packets from one domain to the other.

B. Why Dynamically Changing PIDs

We explain why it is necessary to dynamically change PIDs in CoLoR. We explain two approaches to learning PIDs whey they are static.

1) Two approaches to learning PIDs: The first approach to learning PIDs is GET Luring, where an attacker uses an end host to register normal content names into the network, thus luring GET messages from content consumers. Since the corresponding PIDs are carried by the GET messages, the attacker then can learn a part of PIDs in the network. We call such a process as the PID learning stage.
2) Launching DDoS Attacks: Once the attacker has learned a part of PIDs in the network (through GET luring, botnet cooperation, or other possible approaches), it can freely send packets along the paths represented by the learned PIDs. We assume that the attacker can compromise a number of computers along the paths as zombies, by using similar methods with the ones in the current Internet (e.g., by using worms).

1.2 PROBLEM DEFINITION

The main problem Is the impact of DDoS Flooding attack ranges from the simple inconvenience to use a particular service to causing major failures at the targeted server. The existing methods used static path identifiers(S-PIDs) for communication between the routers or resource manager or systems. Due to the static nature it is easy to launch an attack by an attacker like DDoS flooding attacks. To overcome this issue, Dynamic path identifiers (D-PIDs) are one of the proposed methods to detect and defend the DDoS flooding attacks.

1.3 LIMITATIONS

- Needs more processing power as compared to traditional methods
- Requires a proper cryptographic and routing algorithms
- Complex Node generation methods have to utilized

1.4 OBJECTIVE

In D-PID, two adjacent domains periodically update the PIDs between them and used for packet forwarding. Even if the attacker tries to get the PIDs to its target and sends the malicious packets successfully, these PIDs will become invalid after a certain period and the subsequent attacking will be removed. Moreover, if the attacker tries to obtain the new PIDs to launch DDoS flooding attack it not only significantly increases the attacking cost but also makes it easy to detect the attacker. This DPID with data encryption provide more security to data throughout their network path. The basic functions of cryptography are encryption, decryption and cryptographic hashing. In order to encrypt and decrypt messages, the sender and recipient need to share a secret. Typically, this is a key, like a password, that is used by the cryptographic algorithm.

There is chance of attacking data through key assigned for data. but the proposed system, DPID with data encryption and decryption will also detect that type of attack also. For routing the data from source to destination in network through a secure path which means attack free. The breadth first search algorithm and detection of attack or checking the behaviour of each node is combined.

1.5 OUTCOMES

- Solving this problem, we would be able to quickly prevent DDOS attacks that may lead to unwanted complications
- Eliminate the non-relevant parts of the network connections and create appropriate dynamic features
- provide free or low-cost access to the method allowing small and start-up companies to reduce server loads and targeted attacks

2. LITERATURE SURVEY

2.1 RELATED WORK

Because of the complexity and difficulty in defending against DDoS flooding attacks, many approaches have been proposed in past two decades. For instance, filtering-based approaches aim at mitigating DDoS flooding attacks by deploying source address filtering at routers. Similarly, IP trace back-based methods trace attacks back through the network toward the attacking sources. In addition, the approaches proposed aim at mitigating DDoS attacks by sending shut-up messages to the attacking sources, assuming that they will cooperate and stop flooding. While there are too many literatures, we use a survey on existing approaches in defending again DDoS flooding attacks. Instead, we outline prior work closely related to this work and compare D-PID with them. A main reason that DDoS flooding attacks proliferate is a node can send any amount of data packets to any destination, regardless whether or not the destination wants the packets.

To address this issue, several approaches have been proposed. In the “off by default” approach, two hosts are not permitted to communicate by default. Instead, an end host explicitly signals, and routers exchange, the IP-prefixes that the end host wants to receive data packets from them by using an IP-level control protocol. The D-PID design is similar in spirit, since D-PID dynamically changes PIDs and a content provider can send data packets to a destination only when the destination explicitly sends out a GET message that is routed (by name) to the content provider. However, there are two important differences. First, the “off by default” approach works at the IP-prefix granularity, but D-PID is based on an information-centric network architecture and works at the content granularity. Second, the IP-prefixes that an end host wants to receive packets from are propagated throughout the Internet in the “off by default” approach, which may cause significant routing dynamics if the al owed IP-prefixes of end hosts change frequently.

On the other hand, the PIDs are kept secret and change dynamically in D-PID. While this incurs cost since destinations need to re-send GET messages, the results presented in Sec. V show that the cost is fairly small. The capability-based designs also share the same spirit with “off by default” and D-PID. In these approaches, a sender first obtains the permission from the destination in order to send data packets to it. The destination provides the capabilities to the sender if it wants to receive packets from the sender. The sender then embeds the obtained capabilities into packets. Routers along the path from the sender to the destination verify the capabilities in order to check whether or not the destination wants to receive the packets.

If not, the International Journal of Computer Sciences and Engineering Vol.6(7), Jul 2018, E-ISSN: 2347-2693 © 2018, IJCSE All Rights Reserved 1087
routers simply discard the packets. D-PID differentiates from the capability-based approaches in two aspects. One hand, communications are initiated by receivers in D-PID but by senders in capability-based approaches. On the other hand, the capability-based approaches are vulnerable to “denial-of capability” attacks, where compromised computers send plenty of capability requests to a victim, thus preventing normal users to obtain the capability from the victim. By contrast, D-PID effectively mitigates such attacks because of three reasons. First, the GET messages carry the PIDs along the paths from the compromised computers to the victim. Second, the PIDs are negotiated by neighbouring domains that can verify the authenticity of PIDs when they forward GET messages.

2.1.1 Identity

Authentication, Authorization Identity enables characterizing a user through the use of a login. Authentication is used to verify the user’s credentials. This is done in a secure, trustworthy and manageable manner. When authentication is complete, the Cloud authorization verifies the user’s rights. Guidance includes a centralized directory, identity management, privileged user and access management, role-based access control and separation of duties among main features. In addition, the service provider can frequently offer a free trial period. For example, in the summer of 2012, attackers (users for a free period) accessed Mat Hona’s data (writer for Wired Magazine) Apple, Gmail and Twitter accounts . They erased all his personal data in those accounts.

2.1.2 Confidentiality

A malicious attacker in a virtual machine can listen to another virtual machine . An attacker can very easily identify the data center of the Virtual Machine (VM) and can also obtain information about the IP address and the domain name of the data center. In addition, a VM can extract private cryptographic keys being used in other VMs on the same physical server, which subsequently implies the risk of data leakage .However, now, with Amazon Web Service (AWS), the client has the option to manage their own encryption keys .

2.1.3 Integrity

Phishing, fraud and exploitation of software vulnerabilities, traffic hijacking can eavesdrop activities and transactions, manipulate data, return falsified information and redirect clients to illegitimate sites. Similarly, weak interfaces and Application Program Interface APIs cannot protect users from accidental or malicious attempts . For example, Hewlett-Packard (Palo Alto, California, CA, USA) proposes an Integrity Virtual Machines Architecture .

2.1.4 Isolation

Cloud Computing must have a level of isolation among all the VM data and the hypervisor . In Infrastructure as a Service (IaaS), it means isolating VMs' storage, processing memory and access path networks. In Platform as a Service (PaaS): running services and API calls must be isolated. Moreover, in Software as a Service (SaaS): isolation amongst transactions must be achieved.

2.1.5 Availability

Illegitimate users consume much of the victim's processing power, memory, disk Alegitimate users from using the service. Consequently, the VM becomes unavailable, causing a Denial of Service (DoS) or Distributed Denial of Service (DDoS). For example, a DDoS attack with compromised Internet of Things devices happen on Dyn (DNS infrastructure) and paralysed some cloud computing-based sites such as GitHub and Airbnb.

3 PROBLEM ANALYSIS

3.1 PROBLEM DEFINITION

- Normally in general consumer server based connections the server load depends on the consumer's usage in processing power.
- Exploit can be used to take advantage over these network systems by forcing and utilizing more network capacity than needed.
- So we try and work around this problem by creating a dynamic network access.

3.2 BOUNDARIES OF THE PROBLEM

- This method allows us to prevent DDOS attacks by using general and overcoming traditional static node generation.
- Although this project enables more secure and functional network environment, we introduce much more complicated algorithms and comparable latency over static methods.

3.3 SOFTWARE REQUIREMENT SPECIFICATION

3.3.1 Functional Requirements

Admin:

Establish a node based network and provide a proper dynamic and greedy routing algorithm.

Users:

Connect to the network and specify the selected data to be sent to a particular part of the network.

3.3.2 Software Requirements

- Operating system: Windows XP or Windows 7, Windows 8.
- Coding Language: Java – AWT, Swings, Networking
- Data Base: My Sql / MS Access.
- Documentation: MS Office
- IDE: Eclipse Galileo
- Development Kit: JDK 1.6

3.3.3 Hardware Requirements

- System: Pentium Dual Core.
- Hard Disk: 120 GB.
- Output Devices: Monitor
- Input Devices: Keyboard, Mouse
- Ram: 1 GB

4 DESIGN

4.1 INTRODUCTION

Software design sites at the technical kernel of the software engineering process and is applied regardless of the development paradigm and the area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirements have been specified and analysed, system design is the first of the three technical activities – design, code and test that is required to build and verify software.

During design, progressive refinement of data structure, program structure and procedural details are developed, reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design, procedural design.

4.2 PLANNING

4.2.1 Input Design

Input Design plays a vital role in the life cycle of software development, it requires very careful attention of developers. The input design is to feed data to the application as accurate as possible. So inputs are supposed to be designed effectively so that the errors occurring while feeding are minimized. According to Software Engineering Concepts, the input forms or screens are designed to provide to have a validation control over the input limit, range and other related validations.

This system has input screens in almost all the modules. Error messages are developed to alert the user whenever he commits some mistakes and guides him in the right way so that invalid entries are not made. Let us see deeply about this under module design.

Input design is the process of converting the user created input into a computer-based format. The goal of the input design is to make the data entry logical and free from errors. The error is in the input are controlled by the input design. The application has been developed in user-friendly manner. The forms have been designed in such a way during the processing the cursor is placed in the position where must be entered. The user is also provided with in an option to select an appropriate input from various alternatives related to the field in certain cases.

Validations are required for each data entered. Whenever a user enters an erroneous data, error message is displayed and the user can move on to the subsequent pages after completing all the entries in the current page.

4.2.2 Output Design

The Output from the computer is required to mainly create an efficient method of communication within the company primarily among the project leader and his team members, in other words, the administrator and the clients. The output of VPN is the system which allows the project leader to manage his clients in terms of creating new clients and assigning new projects to them, maintaining a record of the project validity and providing folder level access to each client on the user side depending on the projects allotted to him. After completion of a project, a new project may be assigned to the client. User authentication procedures are maintained at the initial stages itself. A new user may be created by the administrator himself or a user can himself register as a new user but the task of assigning projects and validating a new user rests with the administrator only.

The application starts running when it is executed for the first time. The server has to be started and then the internet explorer in used as the browser. The project will run on the local area network so the server machine will serve as the administrator while the other connected systems can act as the clients. The developed system is highly user friendly and can be easily understood by anyone using it even for the first time.

4.3 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

4.3.1 Economical Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

4.3.2 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

4.3.3 Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

4.4 ARCHITECTURE DESIGN

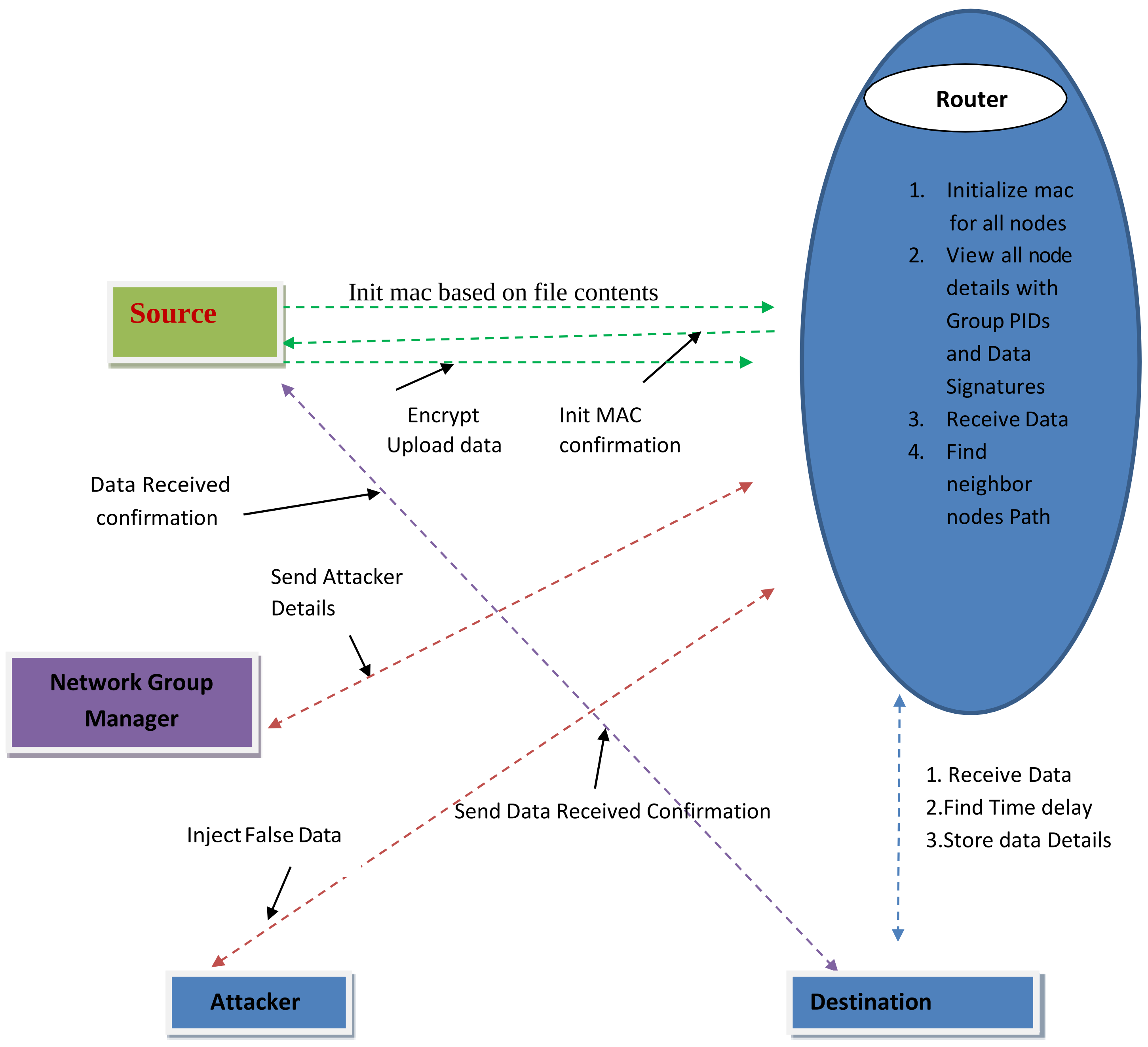

Router
1. Initialize mac for all nodes
2. View all node details with Group PIDs and Data Signatures
3. Receive Data
4. Find neighbor nodes Path
Source
Init mac based on file contents
Encrypt Upload data
Init MAC confirmation
Data Received confirmation
Send Attacker Details
Network Group Manager
Send Data Received Confirmation
Inject False Data
1. Receive Data
2.Find Time delay
3.Store data Details
Attacker
Destination

Fig 4.1 Architecture Design

4.5 FLOW CHART

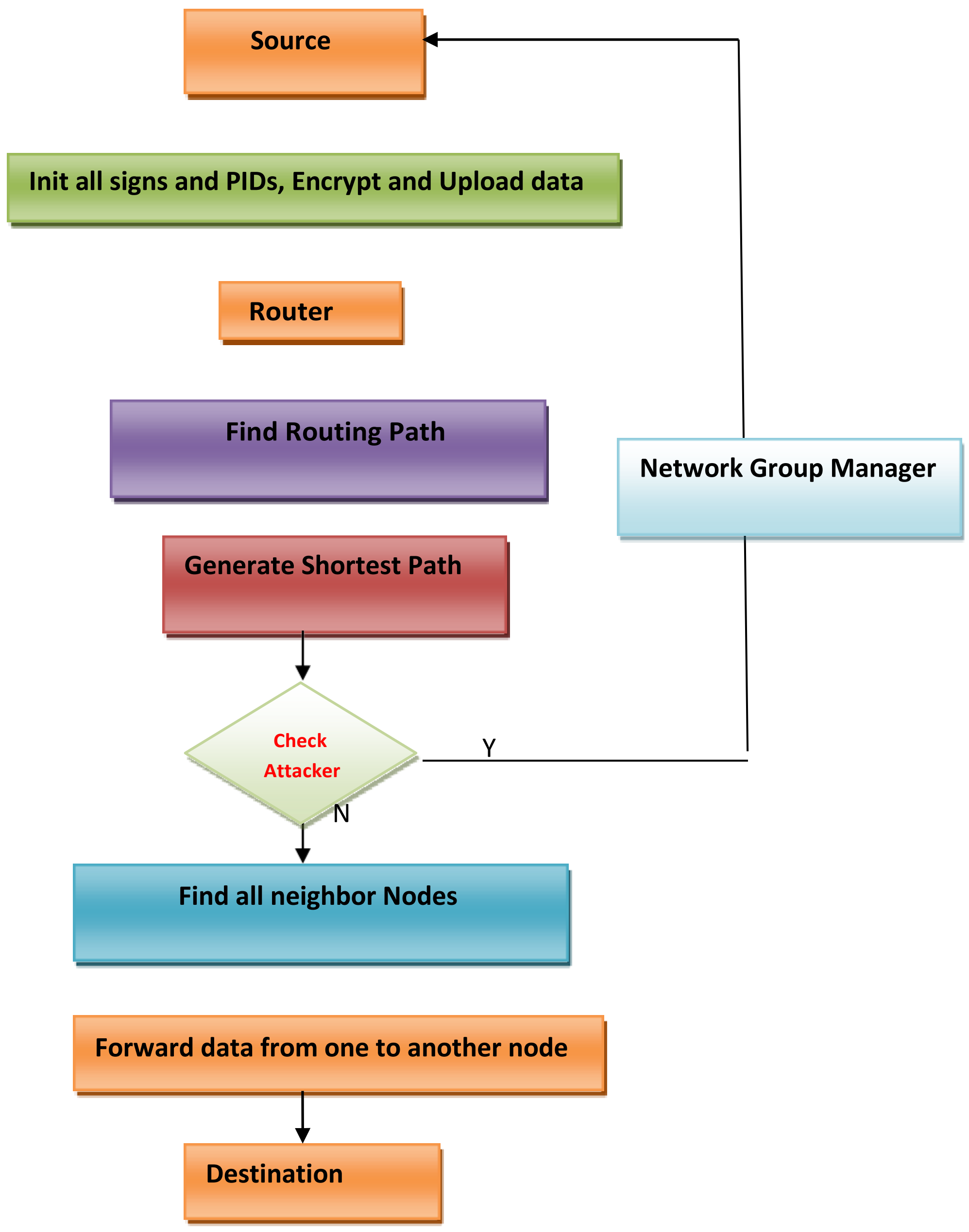


Fig 4.2 Flow Chart

5 IMPLEMENTAITON

5.1 MODULE DESIGN ANDORGANIZATION

The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.

5.2 MODULES

- **Source**
- **Router**
- **Group Manager**
- **Destination**
- **Attacker**

Modules Description

5.2.1 Source

In this module, the Source will browse an file, assign signature to all nodes, assign group PIDs to all groups (group1, group2 and group3) and then send to particular user (A, B, C, D, E and F). After receiving the file he will get response from the receiver. The Source can have capable of manipulating the data file and initializing keys / PIDs to all nodes before sending data to router.

5.2.2 Router

The Router manages a multiple Groups (Group1, Group2, Group3, and Group4) to provide data storage service. In Group n-number of nodes (n1, n2, n3, n4…) are present, and in a Router will check all PIDs and it will select the Neighbor node path. The router also will perform the following operations such as Initialize mac for all nodes, View all node details with Group PIDs and Data Signatures, Receive Data, Find neighbor nodes Path ,Find Type of attackers, Send Attackers to NW Group Manager, Find Routing path, Find time delay and Throughput.

5.2.3 Group Manager

In this module, the group manager can distribute key for each and every group (Group1, Group2 and Group3) and a group each node has a pair of group public/private keys issued by the group manager. Group signature scheme can provide authentications without disturbing the anonymity. Every member in a group may have a pair of group public and private keys issued by the group trust authority (Group Manager). Only the group trust authority (Group Manager) can trace the signer's identity and revoke the group keys. If any attacker will found in a node then the group manager will identify and then send to the particular users.

5.2.4 Destination

In this module, there are an n-numbers of receivers are present (A, B, C, D, E and F). All the receivers can receive the data file from the service provider. The service provider will send data file to router and router will connect to all groups and send to the particular receiver, without changing any file contents. The user can only access the data file. For the user level, all the privileges are given by the NGM authority and the Data users are controlled by the NGM Authority only. Users may try to access data files within the router.

5.2.5 Attacker

In this module, the attacker can attack the node in three ways Passive attack, DOS attack and Impression attack. Dos attack means he will inject fake Group to the particular node, Passive attack means he will change the IP address of the particular node and Impression attack means he will inject malicious data to the particular node.

5.3 ALGORITHM

One algorithm for finding the shortest path from a starting node to a target node in a weighted graph is Dijkstra's algorithm. The algorithm creates a tree of shortest paths from the starting vertex, the source, to all other points in the graph.

Dijkstra's algorithm, published in 1959 and named after its creator Dutch computer scientist Edsger Dijkstra, can be applied on a weighted graph. The graph can either be directed or undirected. One stipulation to using the algorithm is that the graph needs to have a nonnegative weight on every edge.

Dijkstra's algorithm finds a shortest path tree from a single source node, by building a set of nodes that have minimum distance from the source.

Graph

The graph has the following:

- Vertices, or nodes, denoted in the algorithm by or ;
- Weighted edges that connect two nodes: () denotes an edge, and denotes its weight. In the diagram on the right, the weight for each edge is written in gray.

This is done by initializing three values:

- An array of distances from the source node to each node in the graph, initialized the following way: () = 0; and for all other nodes , () = . This is done at the beginning because as the algorithm proceeds, the from the source to each node in the graph will be recalculated and finalized when the shortest distance to is found
- A queue of all nodes in the graph. At the end of the algorithm's progress, will be empty.
- An empty set, to indicate which nodes the algorithm has visited. At the end of the algorithm's run, will contain all the nodes of the graph.

The algorithm proceeds as follows:

1. While is not empty, pop the node , that is not already in , from with the smallest (). In the first run, source node will be chosen because () was initialized to 0. In the next run, the next node with the smallest value is chosen.
2. Add node to , to indicate that has been visited
3. Update values of adjacent nodes of the current node as follows: for each new adjacent node ,

- if () + < (), there is a new minimal distance found for , so update () to the new minimal distance value;
- Otherwise, no updates are made to ().

The algorithm has visited all nodes in the graph and found the smallest distance to each node.

```
begin
  $I_{ij}(0)$ = 0 if $i=j$
  $I_{ij}(0)$ = length(($v_i$,$v_j$)) if edge exists and $i \neq j$
  $I_{ij}(0)$ = $\infty$ otherwise
  for $k = 0$ to $N$-$1$
    for $i = 0$ to $N$-$1$
      for $j = 0$ to $N$-$1$
        $I_{ij}(k+1)$ = min($I_{ij}(k)$, $I_{ik}(k)+I_{kj}(k)$)
      endfor
    endfor
  endfor
  $S = I(N)$
end
```

5.4 SOURCE CODE

Router.java:

```
public class portlistener implements Runnable
        {
                int port;
                portlistener(int port)
                {
                        this.port=port;
                }
       public  void run()
                {
        if (this.port == 444) {
                                try {
ServerSocket server3 = new ServerSocket(444);
Socket con2;
while (true) {
                con2 = server3.accept();
                DataInputStream din = new DataInputStream(con2.getInputStream());
                String node = (String)din.readUTF();
```

```
                    String ip = (String)din.readUTF();
                    DBCon db = new DBCon();
                    Connection con = db.getConnection();
                    Statement stmt2 = con.createStatement();
                    String sql2 = "update Nodesinfo set
                    passive='PassiveAttacker',destip='"+ip+"' where Nname='"+node+"'";
                    stmt2.executeUpdate(sql2);
                    DataOutputStream dout = new
                    DataOutputStream(con2.getOutputStream())
                    dout.writeUTF("success");
                    }}
                    catch(Exception e1)
                    {e1.printStackTrace();}}
                        if (this.port == 4431) {
                            try {
ServerSocket server33 = new ServerSocket(4431);
Socket con22;
while (true) {
                    con22 = server33.accept();
                    DataInputStream din = new
                    DataInputStream(con22.getInputStream())
                    String node  = (String)din.readUTF();
                    String hash =  (String)din.readUTF();
                    String data = (String)din.readUTF();
                    DBCon db = new DBCon();
                    Connection con = db.getConnection();

                        Statement stmt2 = con.createStatement();
                        String sql2 = "update Nodesinfo set impression='Impression
Attacker',sig='"+hash+"',injdata='"+data+"' where Nname='"+node+"'";
                        stmt2.executeUpdate(sql2);
                        DataOutputStream dout = new
DataOutputStream(con22.getO utputStream());
                        dout.writeUTF("success");
                        }
                        }
                        catch(Exception e1)
                            {
                                e1.printStackTrace();
                            }}
                        if (this.port  == 442) {
                            try {

                                ServerSocket server3 = new ServerSocket(442);
                                Socket con21;
                                while (true) {
                                    con21 = server3.accept();
                                    DataInputStream din = new
DataInputStream(
                                    con21.getInputStream());
```

```
                                                String node = (String)din.readUTF();
                                                String hash = (String)din.readUTF();
                                                DBCon db = new DBCon();
                                                Connection con = db.getConnection();
                                                Statement stmt2 = con.createStatement();
                                                String sql2 = "update Nodesinfo set
dos='DOS Attacker',groupsign='"+hash+"' where Nname='"+node+"'";
                                                stmt2.executeUpdate(sql2);
                                                DataOutputStream dout = new
DataOutputStream(con21.getO utputStream());
                                                dout.writeUTF("success");
                        }}
                                catch(Exception  e1)
                                {
                                e1.printStackTrace();
                                }
                                }
                        if(this.port==206)
                        {
                                try
                                {
                                        DBCon db = new DBCon();
                                         Connection con = db.getConnection();
                                        ServerSocket sc = new ServerSocket(206);
                                        while(true)
                                        {
                                                Socket s = sc.accept();
                                                DataInputStream din = new
DataInputStream(s.getInputStream());
                                        String node = din.readUTF();
                                        String num=node.substring(4);
                                        Statement stmt = con.createStatement();
                                        String sql = "select * from Nodesinfo where
Nname='N'+'"+num+"'";
                                        ResultSet rs = stmt.executeQuery(sql);
                                        if(rs.next()==true)
                                        {
                                                String ipinfo= rs.getString(3);
                                                DataOutputStream dout = new
DataOutputStream(s.getOutputStream());
                                                dout.writeUTF(ipinfo);
                                        }

                                        }
                                }
                                catch(Exception e)
                                {
                                        e.printStackTrace();
                                }
                        }
```

```
			if(this.port==205)
			{
				try
				{
					DBCon db = new DBCon();
					 Connection con = db.getConnection();
					ServerSocket sc = new ServerSocket(205);
					while(true)
					{
						Socket s = sc.accept();
						DataInputStream din = new
DataInputStream(s.getInputStream());
					String node = din.readUTF();
					String moddest=din.readUTF();
					String  ip1=din.readUTF();
					String num=node.substring(4);
					Statement stmt = con.createStatement();
					String sql = "update Nodesinfo set
Destination='"+moddest+"',Attacker2='yes' where Nname='N'+'"+num+"'";
					stmt.executeUpdate(sql);
					Date d1 = new Date();
					String dt = d1.toString();
					Statement stmt1 = con.createStatement();
					String sql1 = "insert into Hacker
values('attacker','"+ip1+"','N'+'"+num+"','"+dt+"')";
					stmt1.executeUpdate(sql1);
					DataOutputStream dout = new
DataOutputStream(s.getO utputStream());
					dout.writeUTF("Updated Successfully");
					}
				}
				catch(Exception e)
				{
					e.printStackTrace();
				}	}
```

Networkmanager.java:

```
public static void main(String[] args) {
		try {
	UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.Windo
wsLookAndFeel");
		} catch (Exception e1) {
			e1.printStackTrace();
		}
		java.awt.EventQueue.invokeLater(new Runnable() {
```

```
                public void run() {
                      new NetworkManager();
                }
          });
    }
```

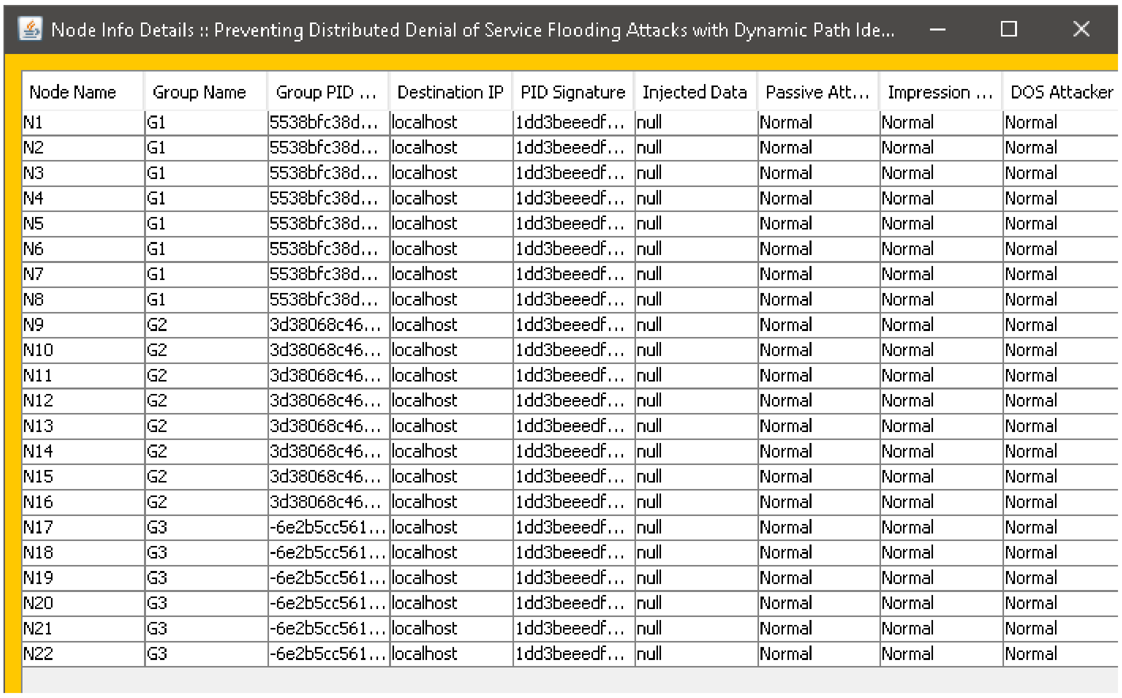
6 TESTING & VALIDATION

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.

Unit Testing is done by means of the developer itself in every degree of the project and best-tuning the horse and module predicated moreover achieved by way of the developer handiest here we are going to clear up all of the runtime errors.

Testing accomplished while application is in improvement level:

Checking for the node details:

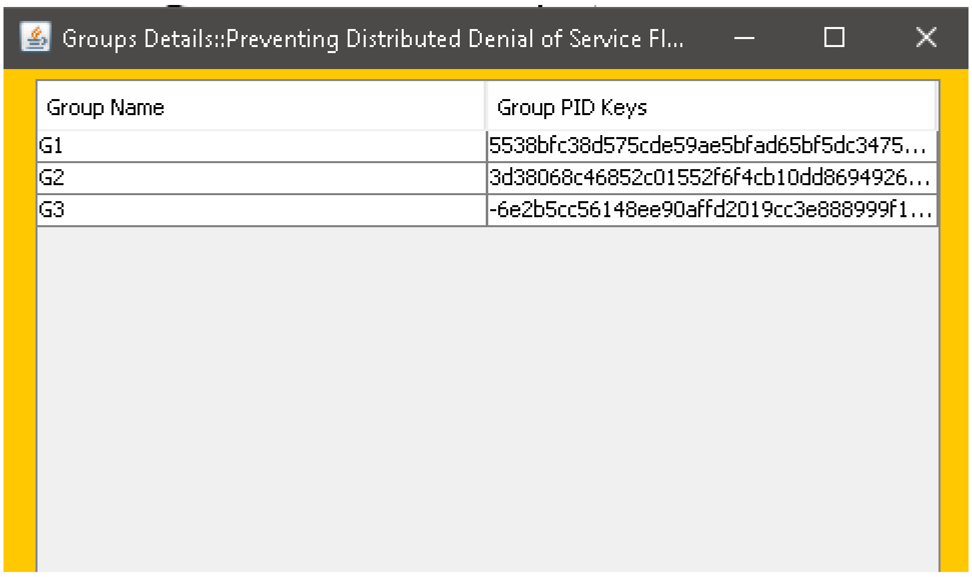


Node Info Details :: Preventing Distributed Denial of Service Flooding Attacks with Dynamic Path Ide...

Node Name	Group Name	Group PID ...	Destination IP	PID Signature	Injected Data	Passive Att...	Impression ...	DOS Attacker
N1	G1	5538bfc38d...	localhost	1dd3beeedf...	null	Normal	Normal	Normal
N2	G1	5538bfc38d...	localhost	1dd3beeedf...	null	Normal	Normal	Normal
N3	G1	5538bfc38d...	localhost	1dd3beeedf...	null	Normal	Normal	Normal
N4	G1	5538bfc38d...	localhost	1dd3beeedf...	null	Normal	Normal	Normal
N5	G1	5538bfc38d...	localhost	1dd3beeedf...	null	Normal	Normal	Normal
N6	G1	5538bfc38d...	localhost	1dd3beeedf...	null	Normal	Normal	Normal
N7	G1	5538bfc38d...	localhost	1dd3beeedf...	null	Normal	Normal	Normal
N8	G1	5538bfc38d...	localhost	1dd3beeedf...	null	Normal	Normal	Normal
N9	G2	3d38068c46...	localhost	1dd3beeedf...	null	Normal	Normal	Normal
N10	G2	3d38068c46...	localhost	1dd3beeedf...	null	Normal	Normal	Normal
N11	G2	3d38068c46...	localhost	1dd3beeedf...	null	Normal	Normal	Normal
N12	G2	3d38068c46...	localhost	1dd3beeedf...	null	Normal	Normal	Normal
N13	G2	3d38068c46...	localhost	1dd3beeedf...	null	Normal	Normal	Normal
N14	G2	3d38068c46...	localhost	1dd3beeedf...	null	Normal	Normal	Normal
N15	G2	3d38068c46...	localhost	1dd3beeedf...	null	Normal	Normal	Normal
N16	G2	3d38068c46...	localhost	1dd3beeedf...	null	Normal	Normal	Normal
N17	G3	-6e2b5cc561...	localhost	1dd3beeedf...	null	Normal	Normal	Normal
N18	G3	-6e2b5cc561...	localhost	1dd3beeedf...	null	Normal	Normal	Normal
N19	G3	-6e2b5cc561...	localhost	1dd3beeedf...	null	Normal	Normal	Normal
N20	G3	-6e2b5cc561...	localhost	1dd3beeedf...	null	Normal	Normal	Normal
N21	G3	-6e2b5cc561...	localhost	1dd3beeedf...	null	Normal	Normal	Normal
N22	G3	-6e2b5cc561...	localhost	1dd3beeedf...	null	Normal	Normal	Normal

Fig 6.1 Node Details

Group details



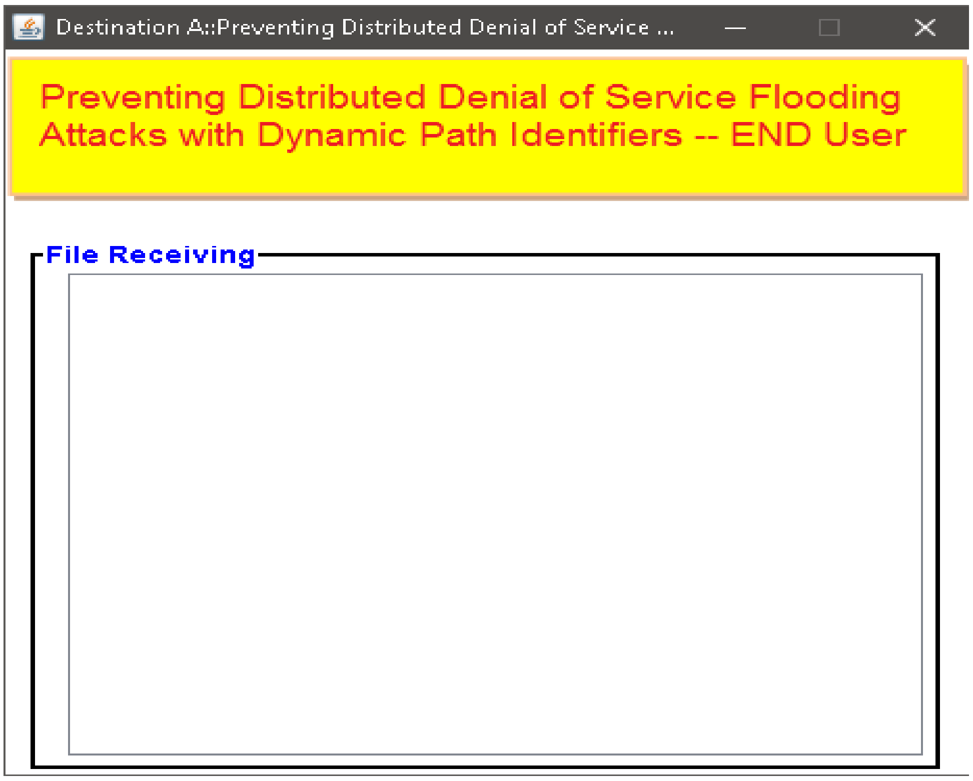
Groups Details::Preventing Distributed Denial of Service Fl...

Group Name	Group PID Keys
G1	5538bfc38d575cde59ae5bfad65bf5dc3475...
G2	3d38068c46852c01552f6f4cb10dd8694926...
G3	-6e2b5cc56148ee90affd2019cc3e888999f1...

Fig 6.2 Group Details

Manual testing and validation on project application:

File Receiving Medium



Destination A::Preventing Distributed Denial of Service ...

Preventing Distributed Denial of Service Flooding Attacks with Dynamic Path Identifiers -- END User

File Receiving

Fig 6.3 File Receiving Medium

Running the source file and browsing the file:

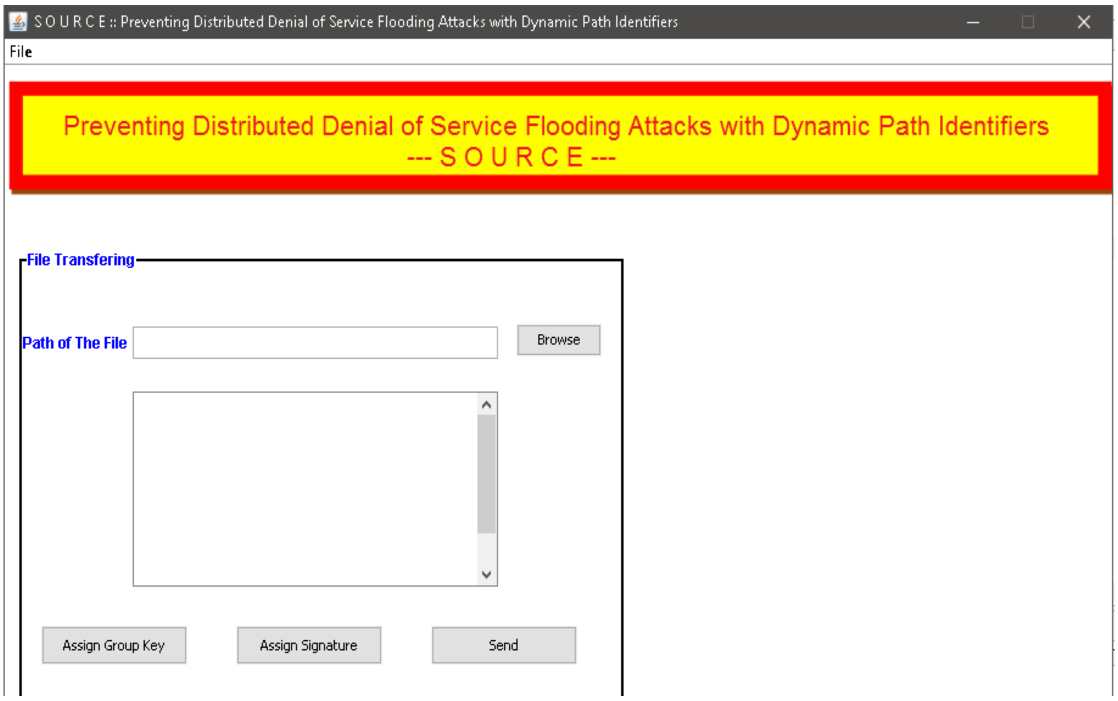
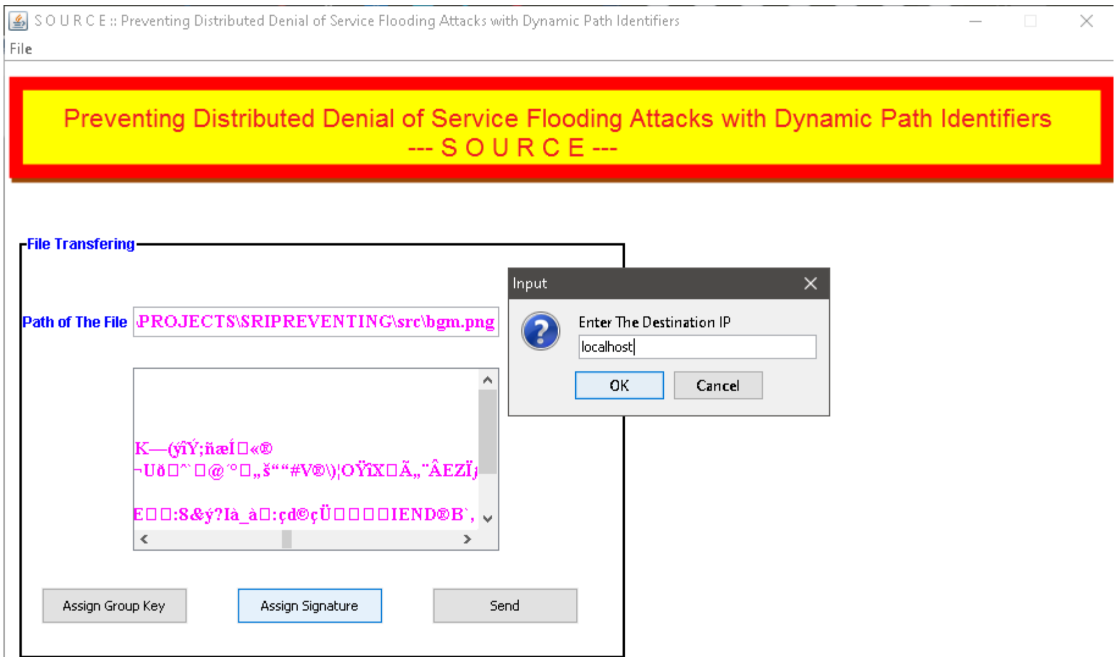

S O U R C E :: Preventing Distributed Denial of Service Flooding Attacks with Dynamic Path Identifiers
File
Preventing Distributed Denial of Service Flooding Attacks with Dynamic Path Identifiers
--- S O U R C E ---
File Transfering
Path of The File
Browse
Assign Group Key
Assign Signature
Send

Fig 6.4 Browsing the File

Entering the destination IP


S O U R C E :: Preventing Distributed Denial of Service Flooding Attacks with Dynamic Path Identifiers
File
Preventing Distributed Denial of Service Flooding Attacks with Dynamic Path Identifiers
--- S O U R C E ---
File Transfering
Path of The File
Input
Enter The Destination IP
OK
Cancel
Assign Group Key
Assign Signature
Send

PROJECTS\SRIPREVENTING\src\bgm.png

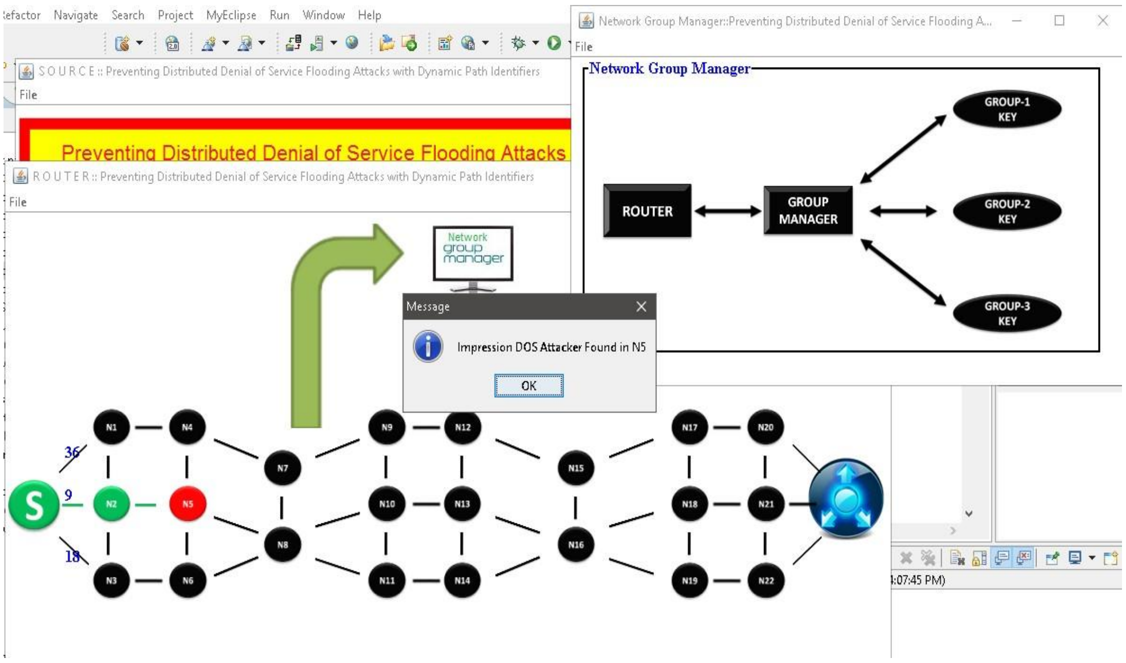
```
K—(ýîÝ;ñæÍ□«®
¬Uð□ˆ`□@´°□„š““#V®\)¦OŸîX□Ã„¨ÂEZÏ¡

E□□:8&ý?Ià_à□:çd©çÜ□□□□IEND®B`,
```

Enter The Destination IP: localhost

Fig 6.5 Destination IP

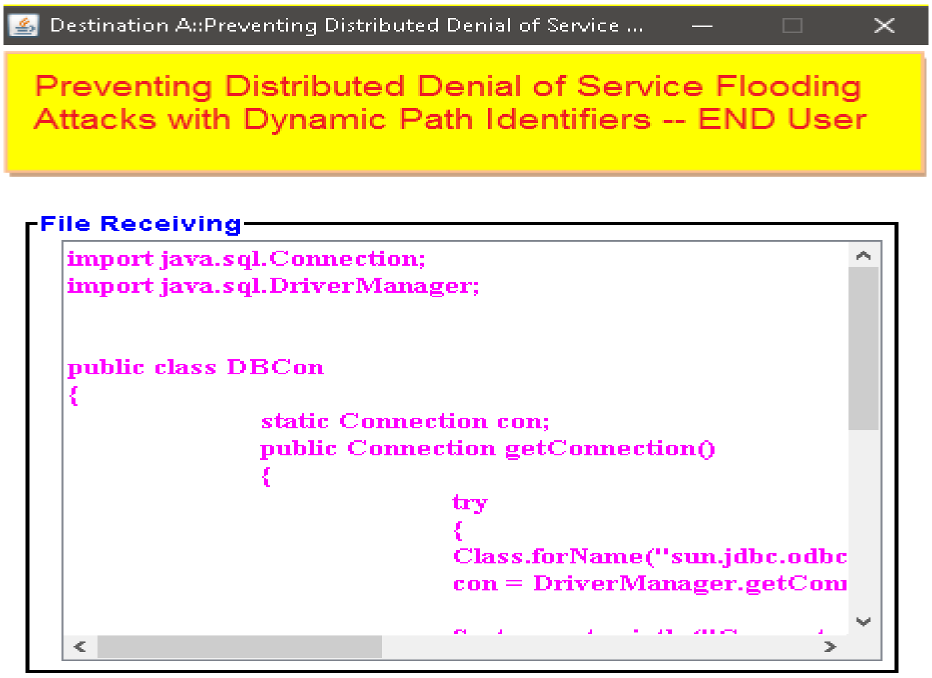
Test Node under attacked


Refactor Navigate Search Project MyEclipse Run Window Help
S O U R C E :: Preventing Distributed Denial of Service Flooding Attacks with Dynamic Path Identifiers
File
Preventing Distributed Denial of Service Flooding Attacks
R O U T E R :: Preventing Distributed Denial of Service Flooding Attacks with Dynamic Path Identifiers
File
Network group manager
Network Group Manager::Preventing Distributed Denial of Service Flooding A...
File
Network Group Manager
ROUTER
GROUP MANAGER
GROUP-1 KEY
GROUP-2 KEY
GROUP-3 KEY
Message
OK
S
N1
N2
N3
N4
N5
N6
N7
N8
N9
N10
N11
N12
N13
N14
N15
N16
N17
N18
N19
N20
N21
N22
36
9
18
:07:45 PM)

Impression DOS Attacker Found in N5

Fig 6.6 Test Node Under Attack

Resultant for end user


Destination A::Preventing Distributed Denial of Service ...
Preventing Distributed Denial of Service Flooding Attacks with Dynamic Path Identifiers -- END User
File Receiving

```
import java.sql.Connection;
import java.sql.DriverManager;


public class DBCon
{
                static Connection con;
                public Connection getConnection()
                {
                                try
                                {
                                Class.forName("sun.jdbc.odbc
                                con = DriverManager.getConn
```

Fig 6.7 Resultant for end User

7 RESULT ANALYSIS

Data Input Directory

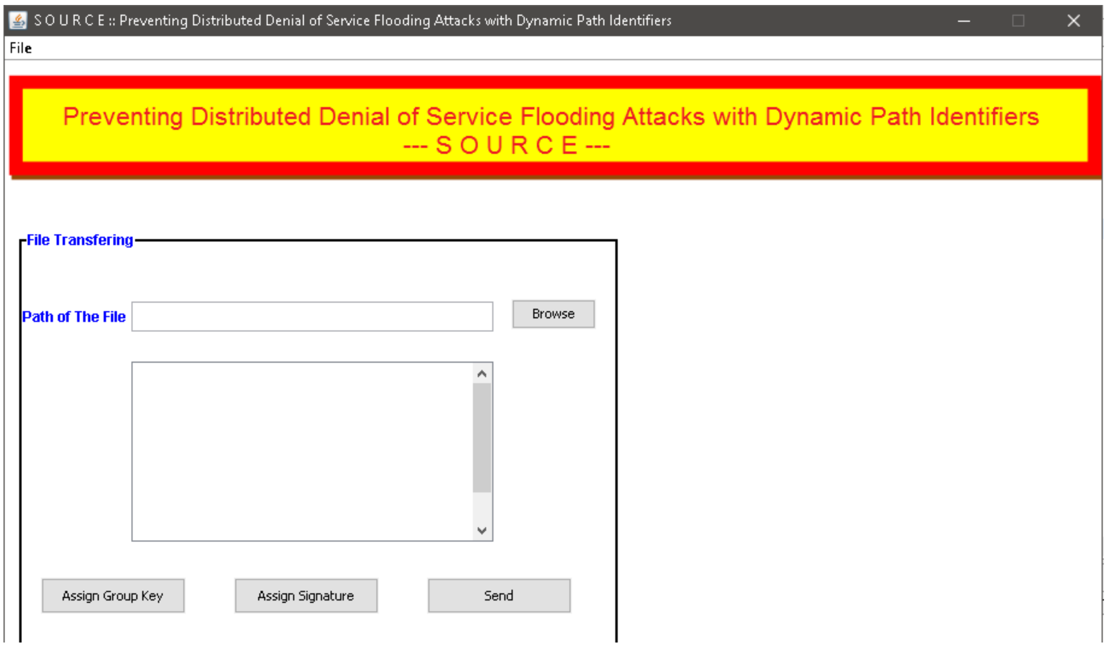
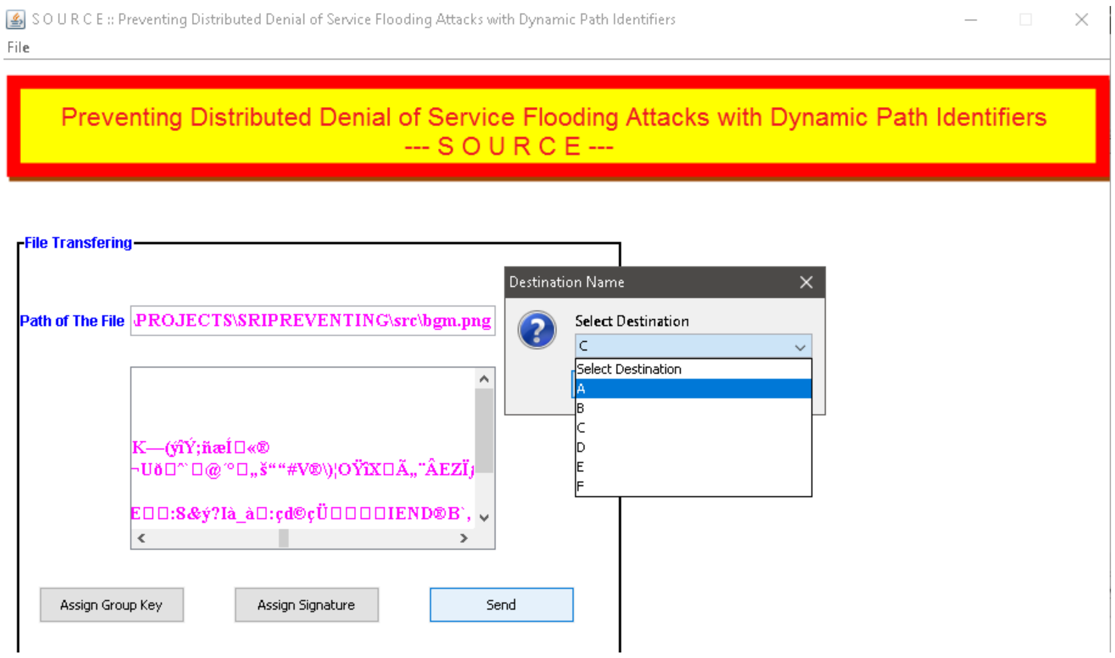

S O U R C E :: Preventing Distributed Denial of Service Flooding Attacks with Dynamic Path Identifiers
File
Preventing Distributed Denial of Service Flooding Attacks with Dynamic Path Identifiers
--- S O U R C E ---
File Transfering
Path of The File
Browse
Assign Group Key
Assign Signature
Send

Fig 7.1 Data Input Directory

Destination Specifying


S O U R C E :: Preventing Distributed Denial of Service Flooding Attacks with Dynamic Path Identifiers
File
Preventing Distributed Denial of Service Flooding Attacks with Dynamic Path Identifiers
--- S O U R C E ---
File Transfering
Path of The File
Destination Name
Select Destination
C
Select Destination
A
B
C
D
E
F
Assign Group Key
Assign Signature
Send

PROJECTS\SRIPREVENTING\src\bgm.png

```
K—(ýîÝ;ñæÍ□«®
¬Uð□^`□@´°□„š““#V®\)¦OŸîX□Ã„¨ÂEZÏ¡

E□□:8&ý?Ià_à□:çd©çÜ□□□□IEND®B`‚
```

Fig 7.2 Destination Specifying

Path Traversal

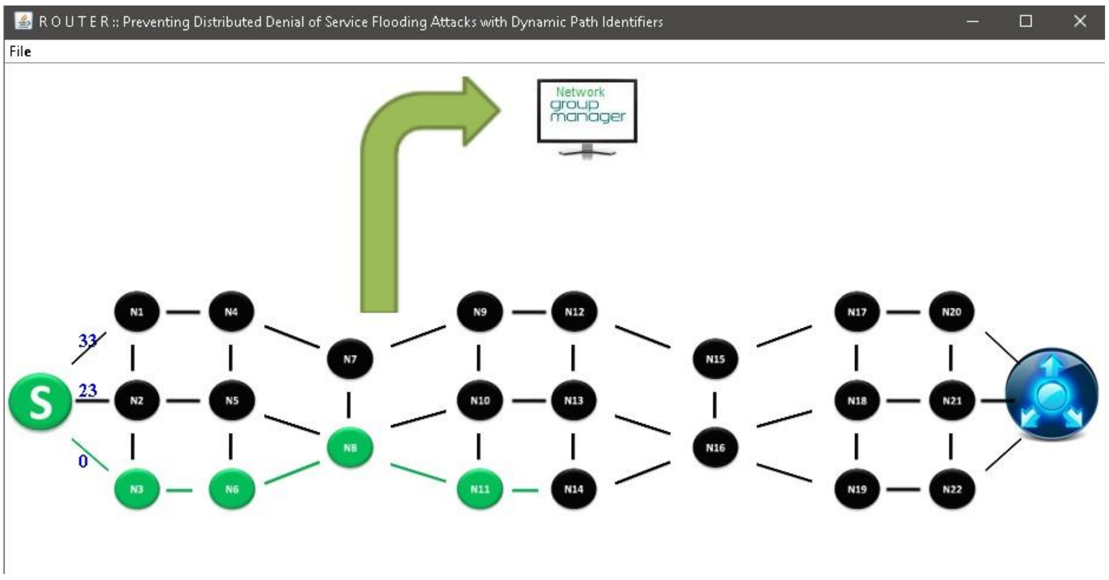

R O U T E R :: Preventing Distributed Denial of Service Flooding Attacks with Dynamic Path Identifiers
File
Network group manager
S
33
23
0
N1
N2
N3
N4
N5
N6
N7
N8
N9
N10
N11
N12
N13
N14
N15
N16
N17
N18
N19
N20
N21
N22

Fig 7.3 Path Traversal

Group Traversal

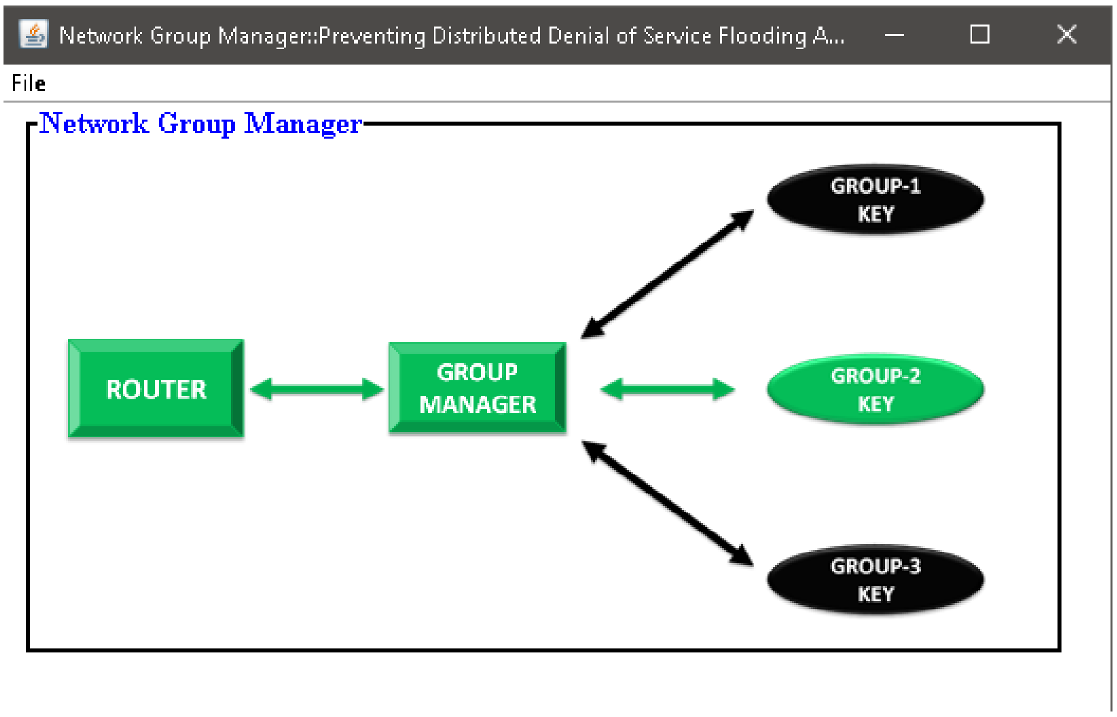
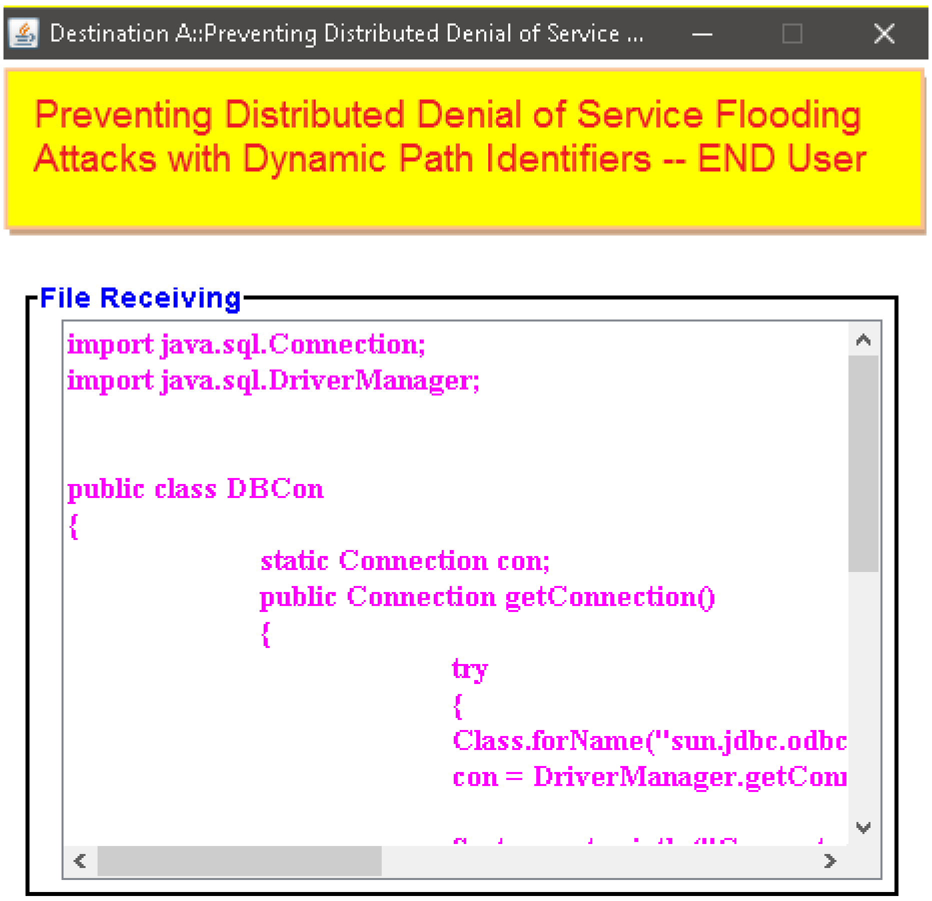

Network Group Manager::Preventing Distributed Denial of Service Flooding A...
File
Network Group Manager
ROUTER
GROUP MANAGER
GROUP-1 KEY
GROUP-2 KEY
GROUP-3 KEY

Fig 7.4 Group Traversal

File Received Without any attack on Nodes


Destination A::Preventing Distributed Denial of Service ...

Preventing Distributed Denial of Service Flooding Attacks with Dynamic Path Identifiers -- END User

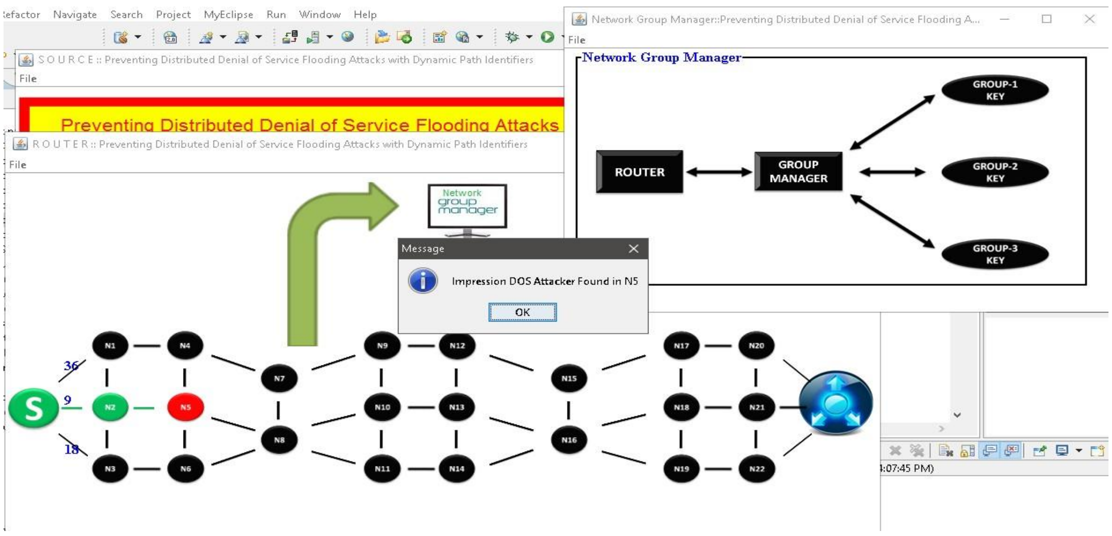
File Receiving

```
import java.sql.Connection;
import java.sql.DriverManager;


public class DBCon
{
                static Connection con;
                public Connection getConnection()
                {
                                try
                                {
                                Class.forName("sun.jdbc.odbc
                                con = DriverManager.getConn
```

Fig 7.5 File Received Without any Attack on Nodes

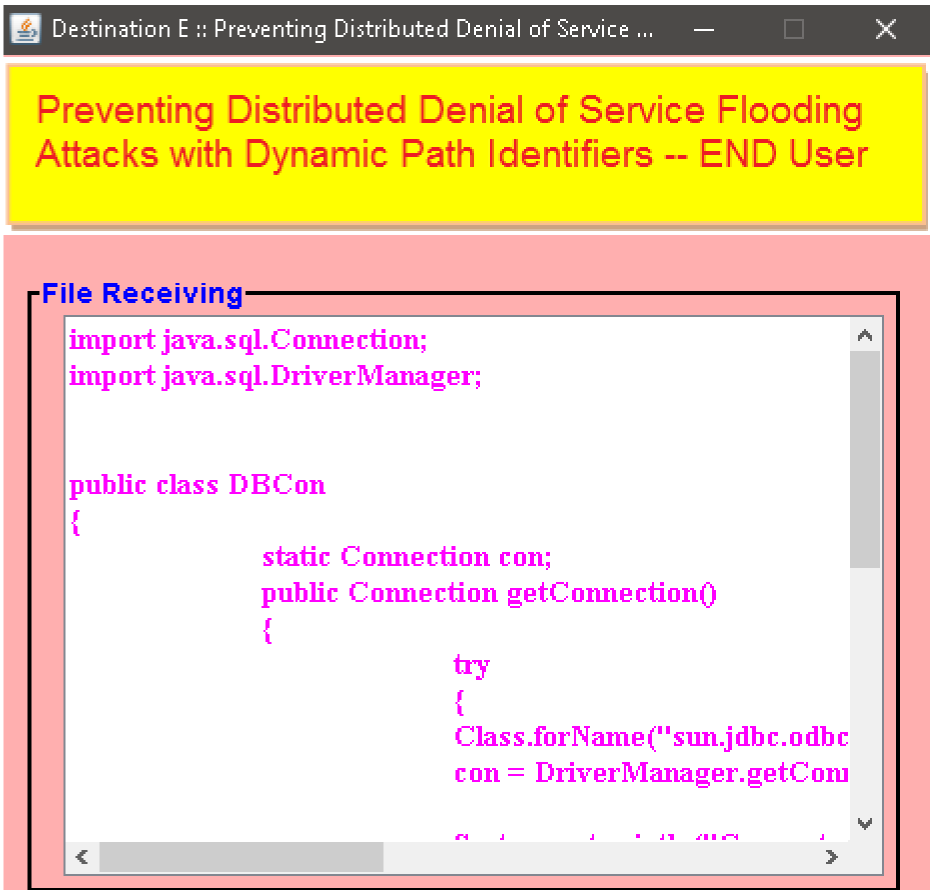
Nodes under Attack


Refactor Navigate Search Project MyEclipse Run Window Help
S O U R C E :: Preventing Distributed Denial of Service Flooding Attacks with Dynamic Path Identifiers
File
Preventing Distributed Denial of Service Flooding Attacks
R O U T E R :: Preventing Distributed Denial of Service Flooding Attacks with Dynamic Path Identifiers
File
Network group manager
Network Group Manager::Preventing Distributed Denial of Service Flooding A...
File
Network Group Manager
ROUTER
GROUP MANAGER
GROUP-1 KEY
GROUP-2 KEY
GROUP-3 KEY
Message
OK
S
36
9
18
N1
N2
N3
N4
N5
N6
N7
N8
N9
N10
N11
N12
N13
N14
N15
N16
N17
N18
N19
N20
N21
N22
:07:45 PM)

Impression DOS Attacker Found in N5

Fig 7.6 Nodes Under Attack

File Received With attack on Nodes


Destination E :: Preventing Distributed Denial of Service ...
File Receiving

Preventing Distributed Denial of Service Flooding Attacks with Dynamic Path Identifiers -- END User

```
import java.sql.Connection;
import java.sql.DriverManager;


public class DBCon
{
                static Connection con;
                public Connection getConnection()
                {
                                try
                                {
                                Class.forName("sun.jdbc.odbc
                                con = DriverManager.getConn
```

Fig 7.7 File Received With attack on Nodes

8 CONCLUSION

We have presented the design, implementation and evaluation of D-PID, a framework that dynamically changes path identifiers (PIDs) of inter-domain paths in order to prevent DDoS flooding attacks, when PIDs are used as inter-domain routing objects. We have described the design details of D-PID and implemented it in a 42-node prototype to verify its feasibility and effectiveness. We have presented numerical results from running experiments on the prototype.

The results show that the time spent in negotiating and distributing PIDs are quite small (in the order of latency) and D-PID is effective in preventing DDoS attacks. We have also conducted extensive simulations to evaluate the cost in launching DDoS attacks in D-PID and the overheads caused by D-PID. The results show that D-PID significantly increases the cost in launching DDoS attacks while incurs little overheads, since the extra number of GET messages is trivial (only 1.4% or 2.2%) when the retransmission period is 300 seconds, and the PID update rate is significantly less than the update rate of IP prefixes in the current Internet

REFERENCES

[1] J. Francois, I. Aib, and R. Boutaba, “Firecol: a Collaborative Protection Network for the Detection of Flooding ddos Attacks,” IEEE/ACM Trans. on Netw.,vol. 20, no. 6, Dec. 2012, pp. 1828-1841.

[2] OVH hosting suffers 1Tbps DDoS attack: largest Internet has ever seen. [Online] Available: https: //www.hackread.com/ovh-hostingsuffers- 1tbps- ddos-attack/.

[3] 602 Gbps! This May Have Been the Largest DDoS Attack in History. http://thehackernews.com/2016/01/biggest-ddos-attack.html.

[4] S. T. Zargar, J. Joshi, D. Tipper, “A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks,” IEEE Commun. Surv. &amp; Tut., vol. 15, no. 4, pp. 2046 - 2069, Nov. 2013.

[5] P. Ferguson and D. Senie, “Network Ingress Filtering: Defeating Denial of Service Attacks that Employ IP Source Address Spoofing,” IETF Internet RFC 2827, May 2000.

[6] K. Park and H. Lee, “On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets,” In Proc. SIGCOMM’01, Aug. 2001, San Diego, CA, USA.

[7] S. Savage, D. Wetheral , A. Karlin, and T. Anderson, “Practical Network Support for IP Traceback,” In Proc. SIGCOMM’00, Aug. 2000, Stockholm, Sweden.

[8] A. C. Snoeren, C. Partridge, L. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer, “Hash-Based IP Traceback,” In Proc. SIGCOMM’01, Aug. 2001, San Diego, CA, USA.

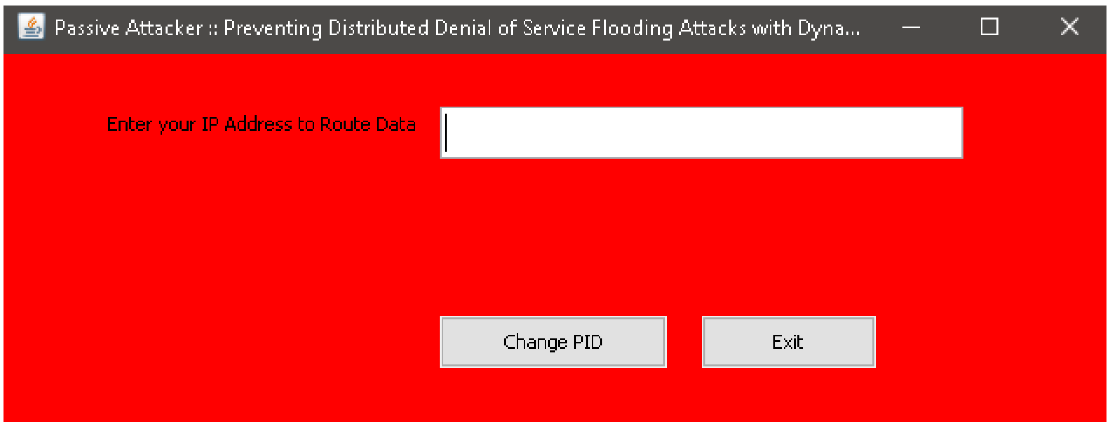
APPENDIX

Passive Attack.java:

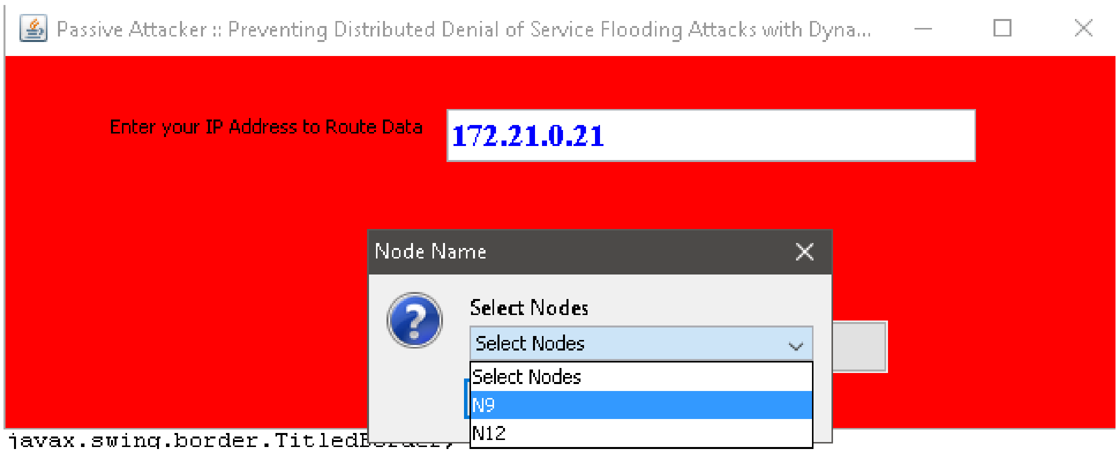
```
String[] dsname = { "Select Nodes", "N9", "N12" };

                                        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");
                                        Socket sc1 = new Socket(ip,444);
                                        DataOutputStream dout1 = new
DataOutputStream(sc1.getOutputStream());
                                        data=t2.getText();
                                        dout1.writeUTF(dataname);
                                        dout1.writeUTF(data);
                                        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" );
                                                                        }
```


Passive Attacker :: Preventing Distributed Denial of Service Flooding Attacks with Dyna...
Enter your IP Address to Route Data
Change PID
Exit

Fig(a) Passive Attacker


Passive Attacker :: Preventing Distributed Denial of Service Flooding Attacks with Dyna...

Enter your IP Address to Route Data: 172.21.0.21

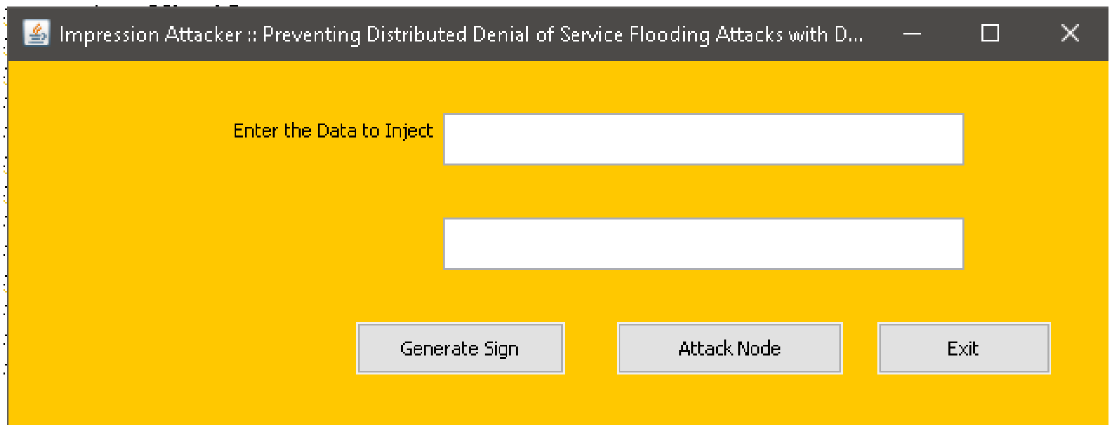
Node Name

Select Nodes

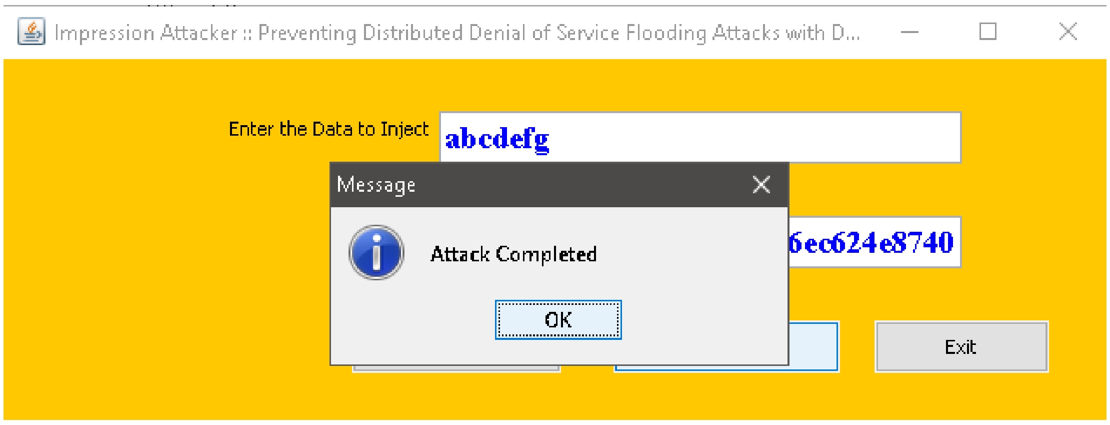
Fig(b) Attacking the Node

Impression Attack:

```
String[] dsname = { "Select Nodes", "N5", "N10", "N18" };
                                        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");
                                        Socket sc1 = new Socket(ip,4431);
                                        DataOutputStream dout1 = new
DataOutputStream(sc1.getOutputStream());
                                        dout1.writeUTF(dataname);
                                        dout1.writeUTF(hash);
                                        dout1.writeUTF(data);
                                        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" );
                                        }
```


Impression Attacker :: Preventing Distributed Denial of Service Flooding Attacks with D...
Enter the Data to Inject
Generate Sign
Attack Node
Exit

Fig(c) Impression Attack


Impression Attacker :: Preventing Distributed Denial of Service Flooding Attacks with D...
Enter the Data to Inject
abcdefg
6ec624e8740
Message
Exit

Attack Completed

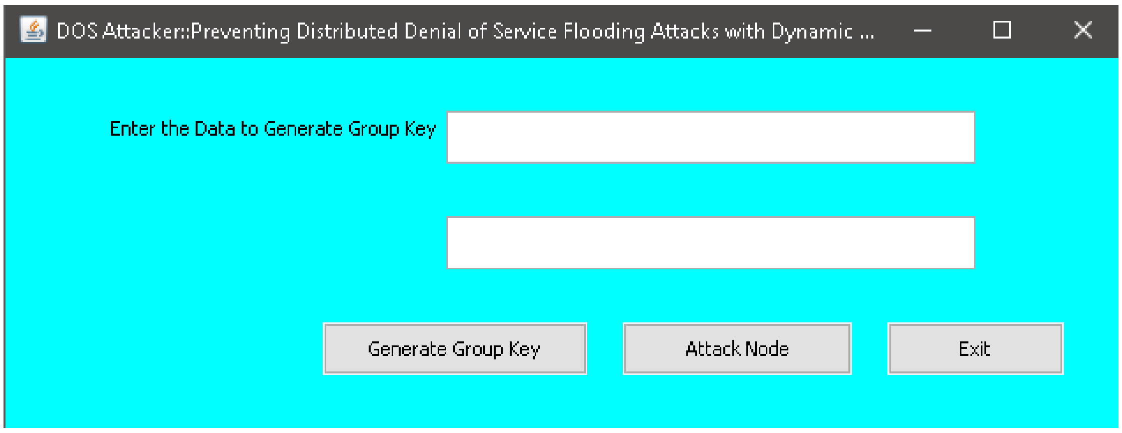
OK

Fig(d) Attack Completed

DDOS Attack

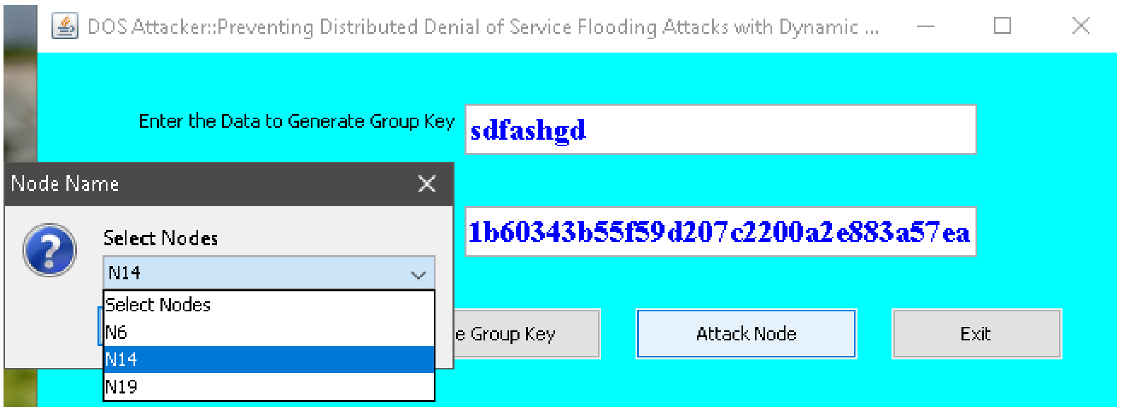
```
String[] dsname = { "Select Nodes", "N6", "N14",  "N19"  };
String dataname = (String)  JOptionPane.showInputDialog(null,
                                              "Select Nodes", "Node Name",
JOptionPane.QUESTION_MESSAGE, null, dsname, dsname[0]);
String ip = JOptionPane.showInputDialog(null,"Please Enter the IP Address Of Router");
Socket sc1 = new Socket(ip,442);
                              DataOutputStream dout1 = new
DataOutputStream(sc1.getOutputStream());
                       dout1.writeUTF(dataname);
                       dout1.writeUTF(hash);
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" );
                    }
```


DOS Attacker::Preventing Distributed Denial of Service Flooding Attacks with Dynamic ...

Enter the Data to Generate Group Key

Fig(e) DOS Attack


DOS Attacker::Preventing Distributed Denial of Service Flooding Attacks with Dynamic ...

Enter the Data to Generate Group Key: sdfashgd

1b60343b55f59d207c2200a2e883a57ea

Node Name

Select Nodes: N14

Fig(f) Attacking the Node