DDoS Attack Simulation

&

Secure Network Simulation

Project of CPIT-375

By

Ahmed Abdulrhman	
Abdullah	
Maan	

Supervised by

Dr. Morshid Aldirbali

Table of Content

Chapter I, DDoS Attack Simulation

Introduction

Distributed Denial-of-Service (DDoS) attacks represent a significant threat to modern network infrastructures, where multiple compromised systems are used to target and overwhelm a particular service or server, often leading to service disruption. The objective of this simulation is to replicate the behavior and impact of a DDoS attack within a controlled network environment, focusing on the server's ability to handle a high volume of malicious traffic while maintaining its service to legitimate users.

Simulation

To simulate DDoS attack we used omnet++ as software to run and test the attack. Our project comprises ten files, each serving a distinct purpose in creating and managing the simulation components. Below is an overview of these files:

A. Omnet.ini

This file serves as the configuration file for the simulation.

B. ClientServerNetwork.ned

This file contains the network topology definition in the NED (Network Description) language. It specifies the structure of the network, including the arrangement of nodes, routers, and the server, as well as their interconnections. The bidirectional links between modules are defined here, ensuring accurate message flow during the simulation.

C. Node.h & Node.cc

These files define the Node module's logic. Nodes represent legitimate network clients that generate and send periodic messages to the server via the router. The implementation includes functions for message creation and transmission, ensuring the simulation of normal network activity.

D. Attacker.h & Attacker.cc

The Attacker module inherits from the Node module but introduces behavior specific to generating a high volume of traffic to simulate malicious activity. These files contain logic for launching the DDoS attack, including message flooding mechanisms aimed at overwhelming the server.

E. Router.h & Router.cc

The Router module manages message routing between nodes and the server. It dynamically forwards messages based on network conditions and destination addresses. The implementation ensures proper handling of both legitimate and attack traffic, mimicking real-world router behavior under stress.

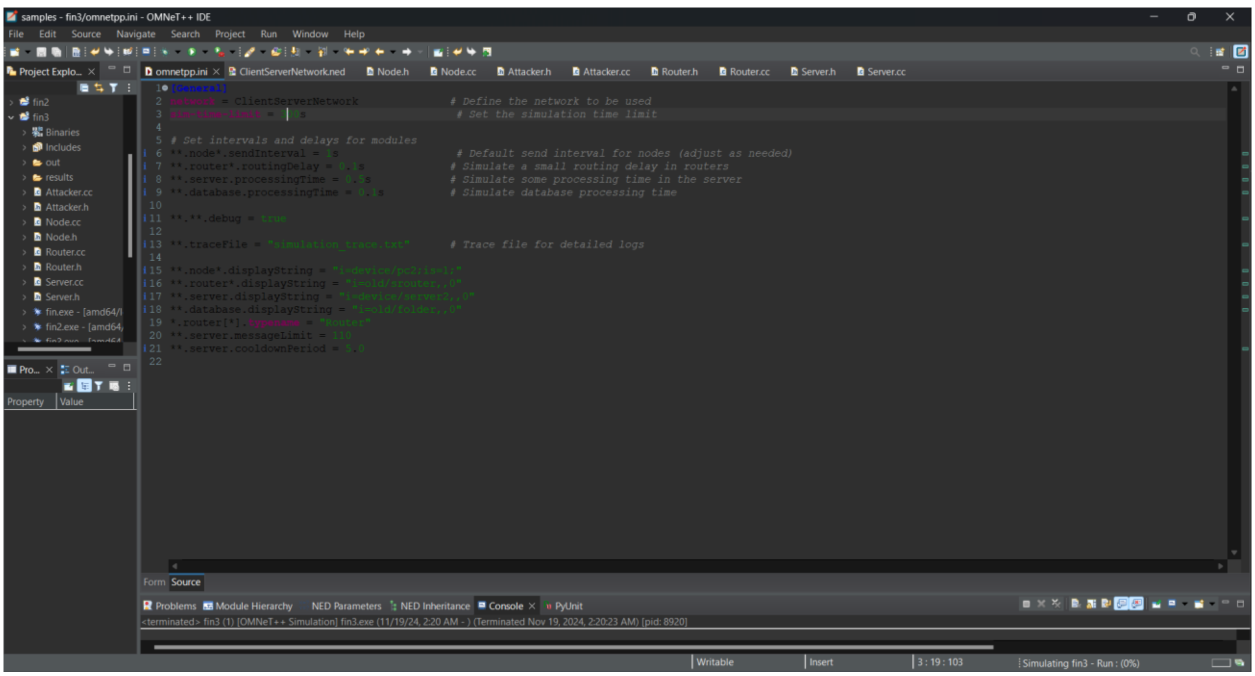
F. Server.h & Server.cc

The Server module represents the central target of the simulation. Its logic includes handling incoming messages, tracking message counts. The server also responds to legitimate nodes to simulate real-time interactions while managing overload conditions during the attack.

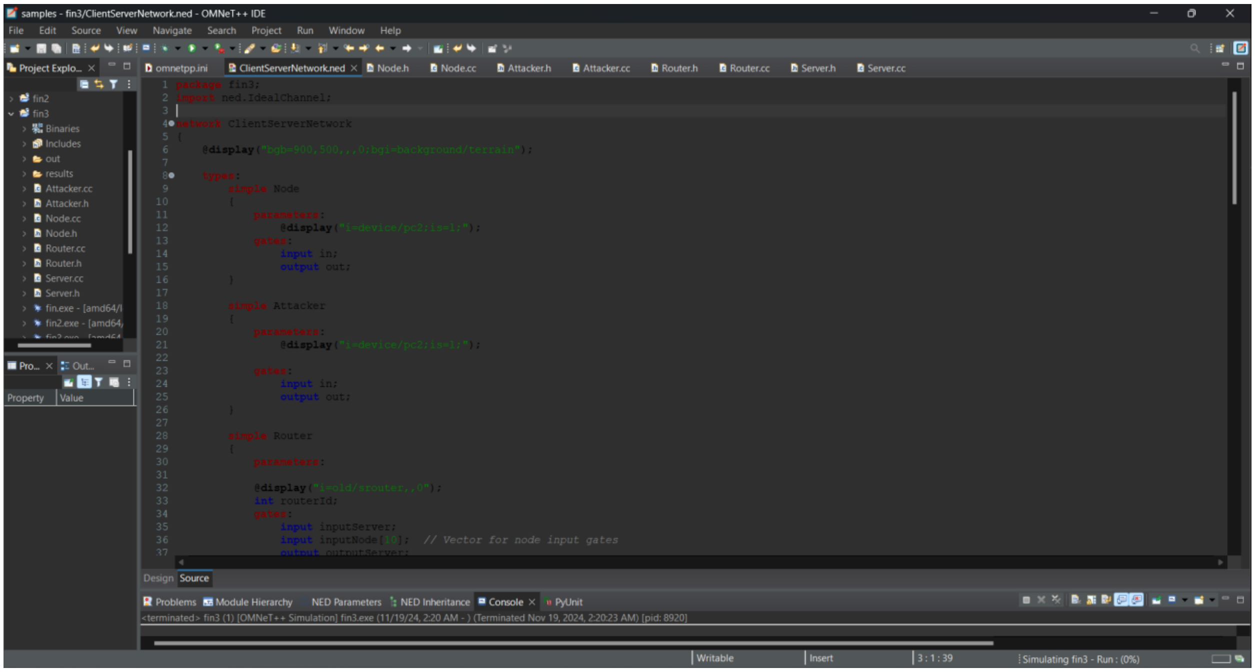
Files Logic

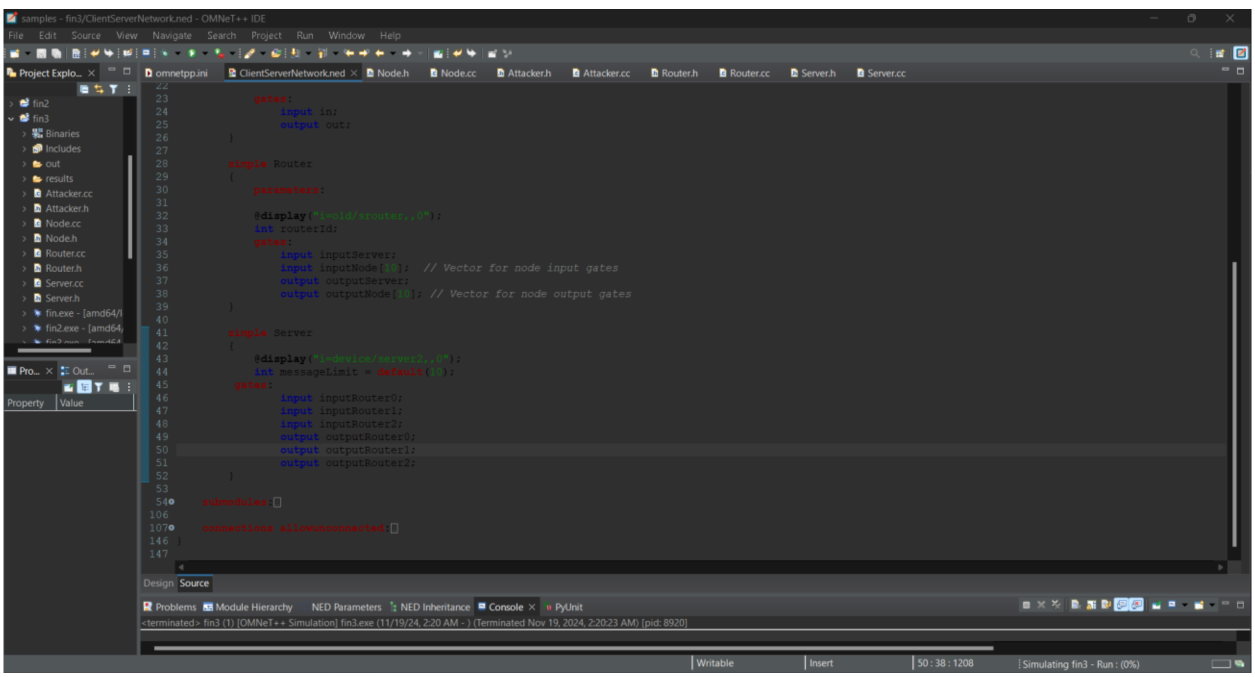
In this section, I will outline the logic of each file to provide an overview of the code used in the simulation. This explanation highlights the functionality and role of each component in the overall design.

Omnetpp.ini

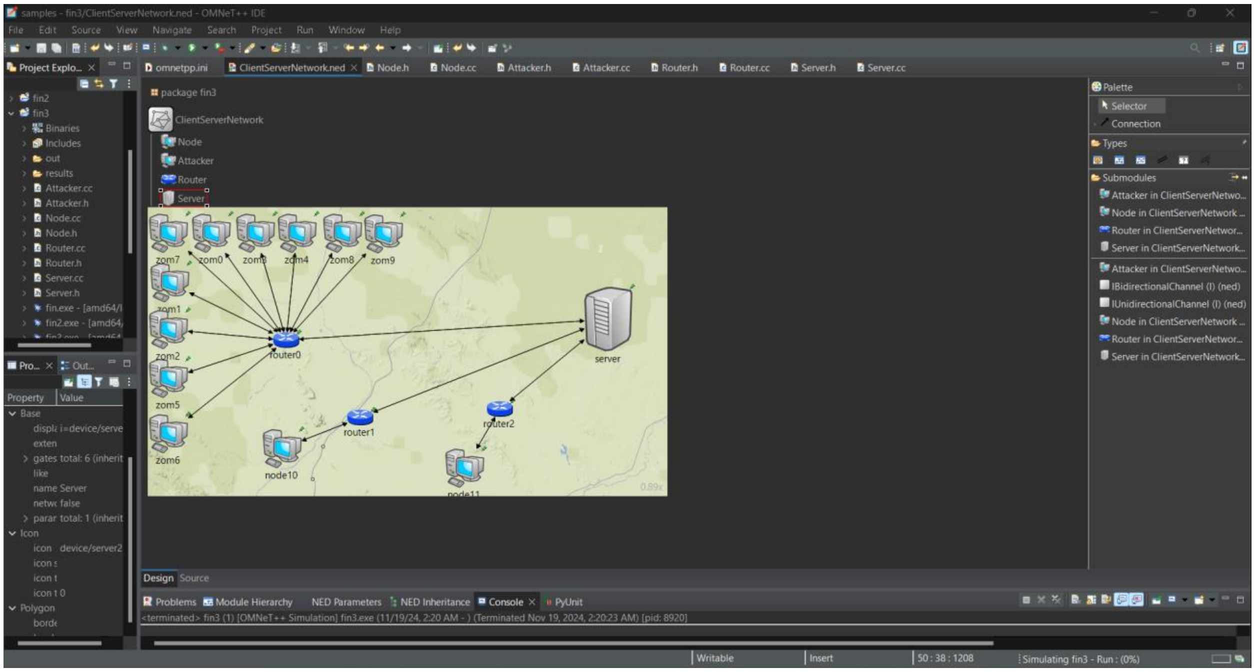


ClientServerNetwork.ned

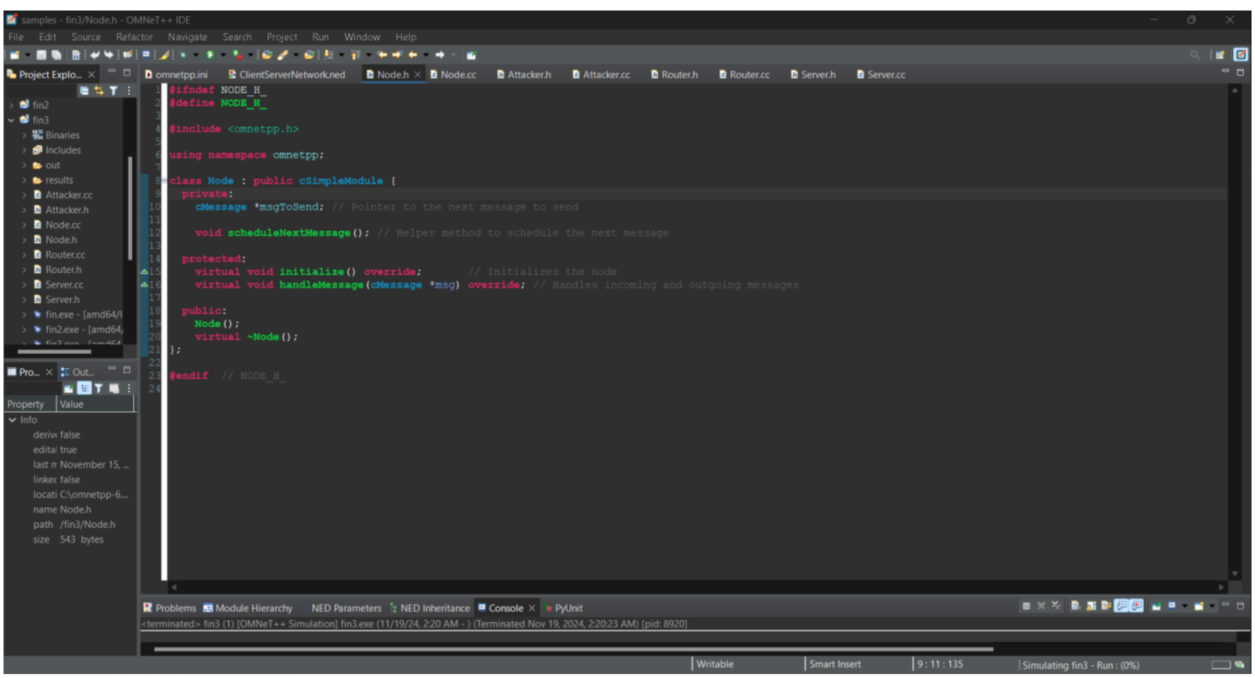




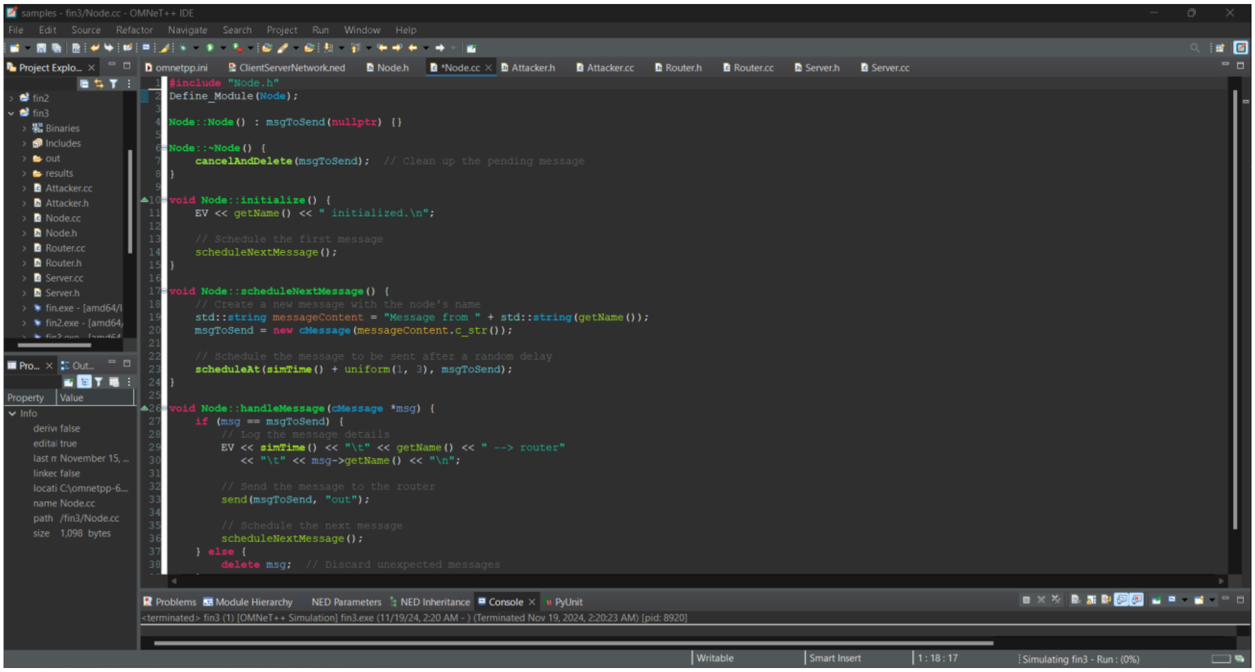
The topology



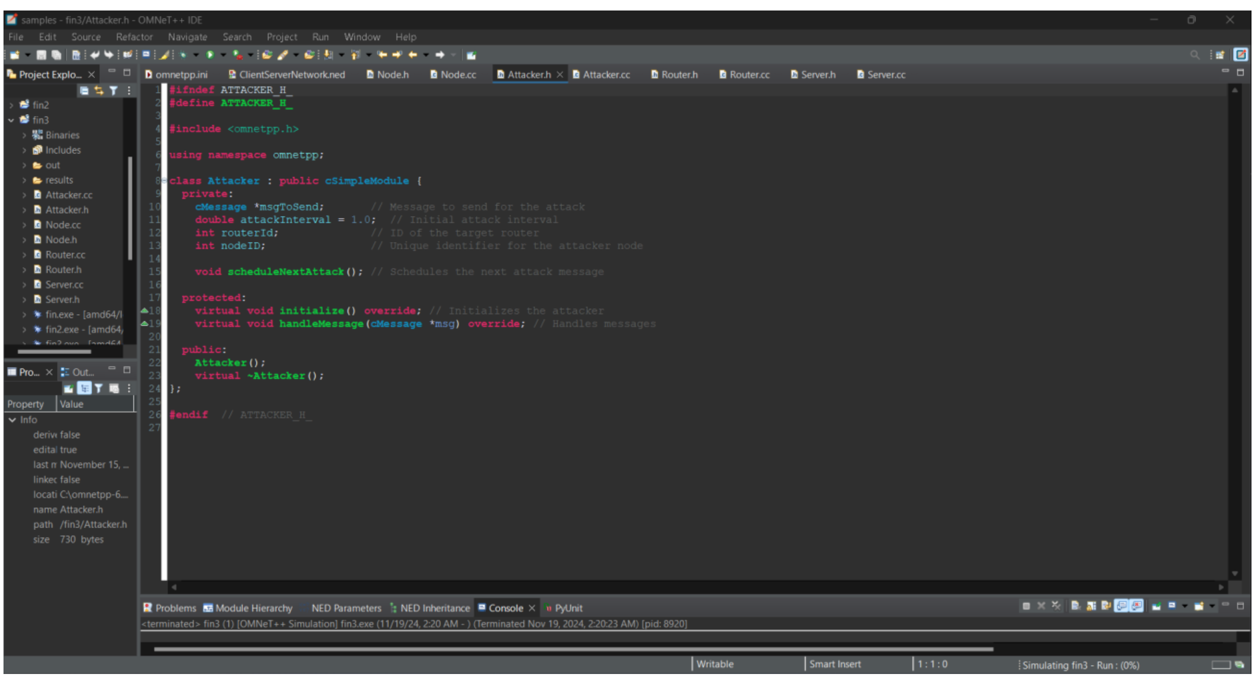
Node.h



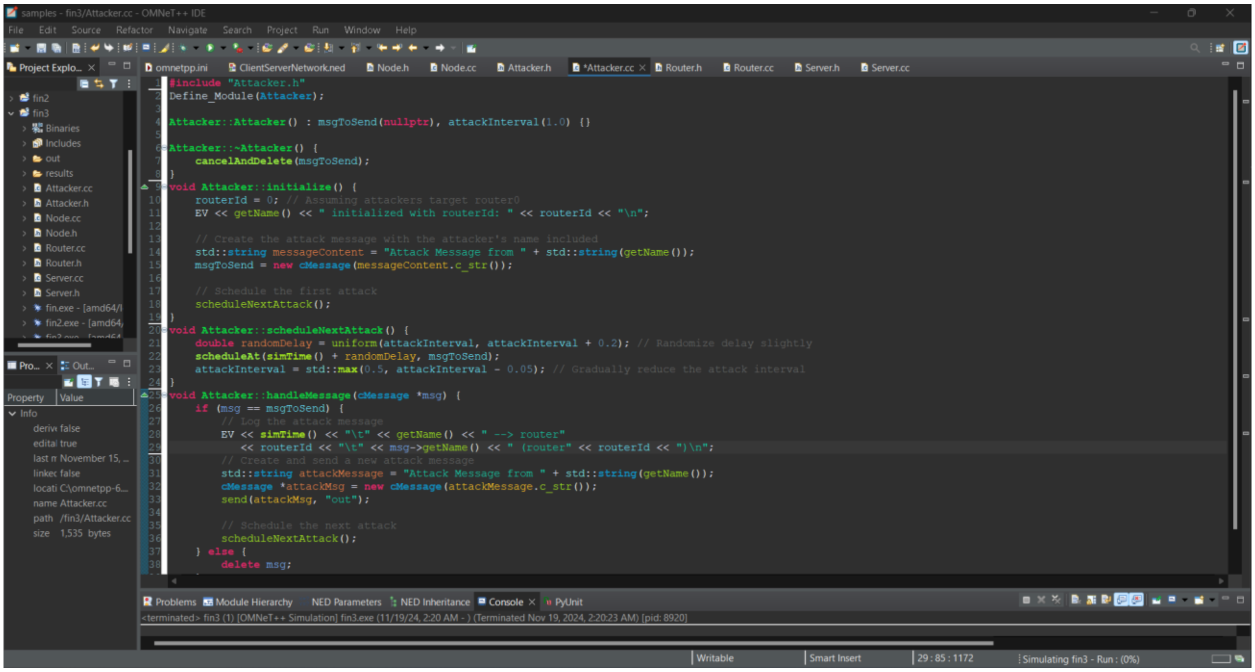
Node.cc



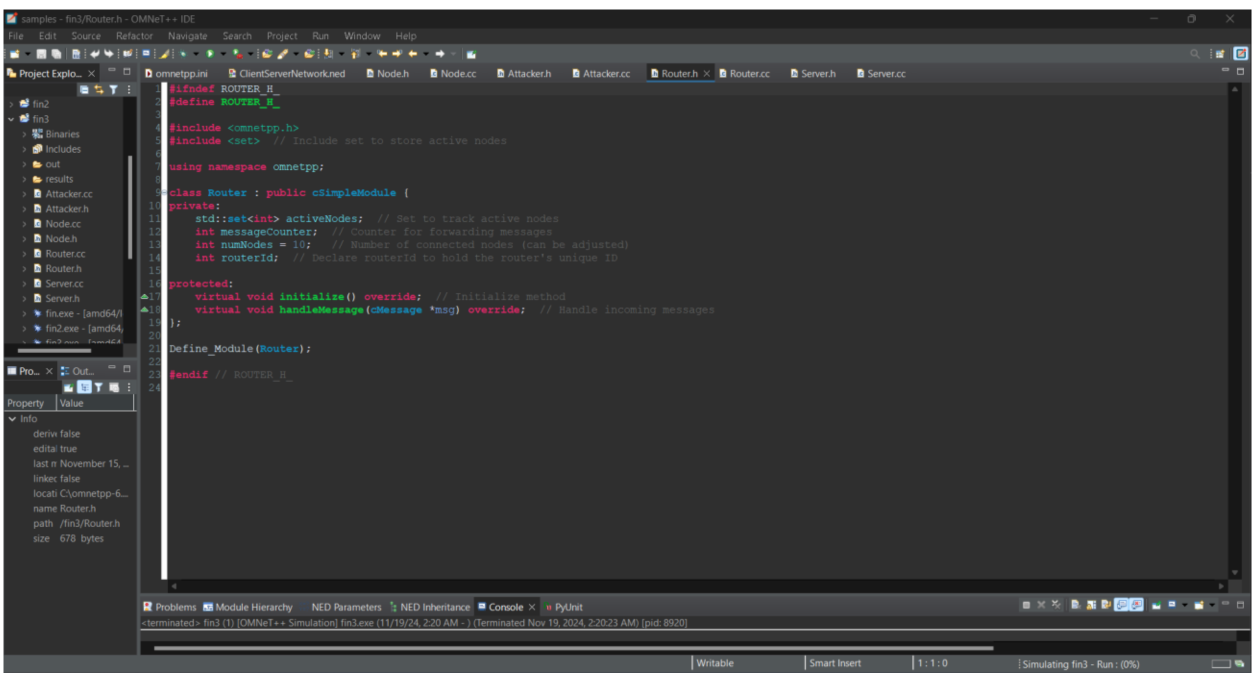
Attacker.h



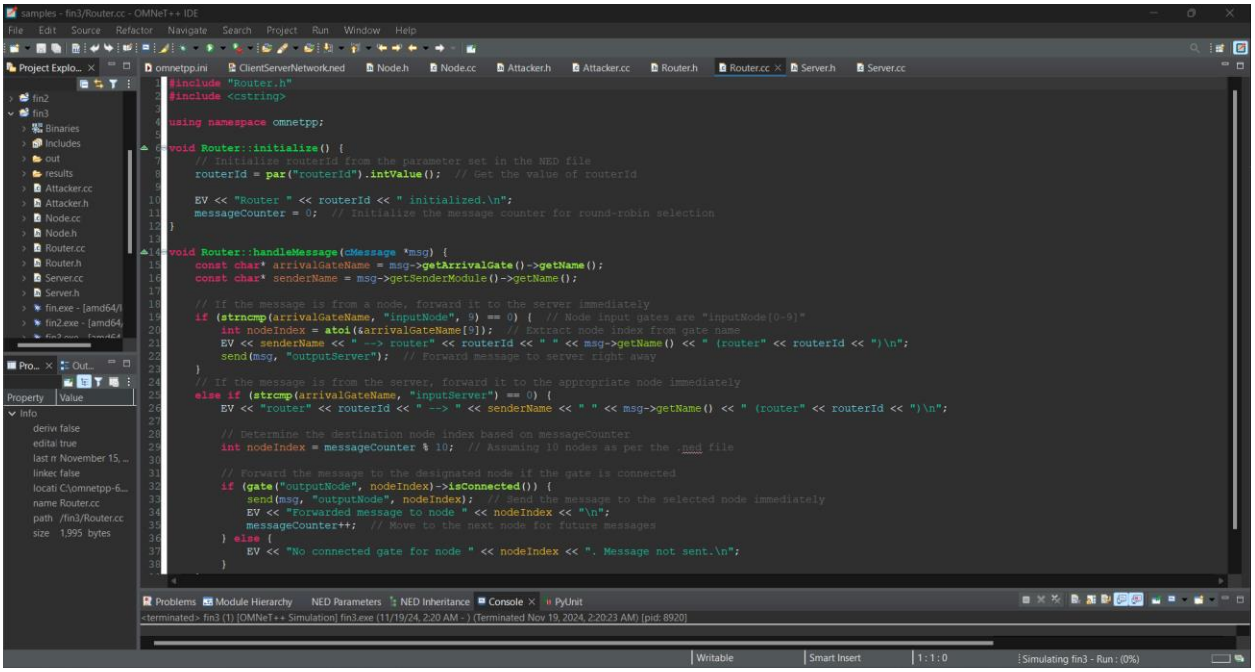
Attacker.cc



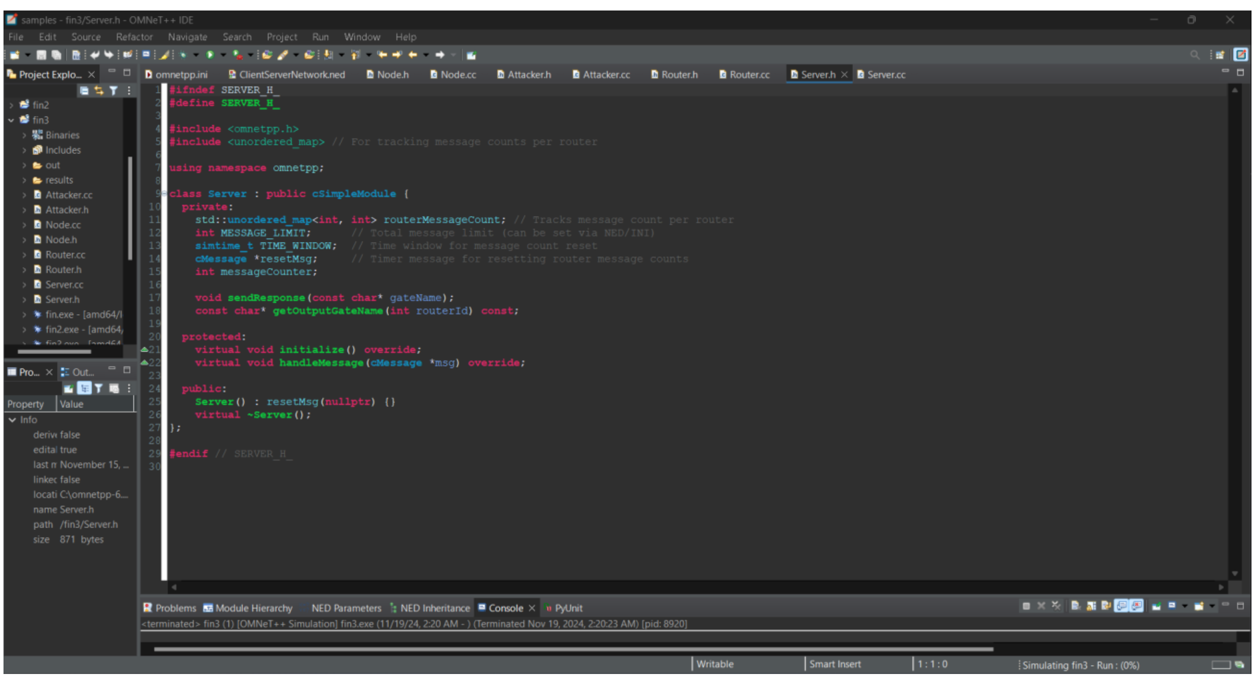
Router.h



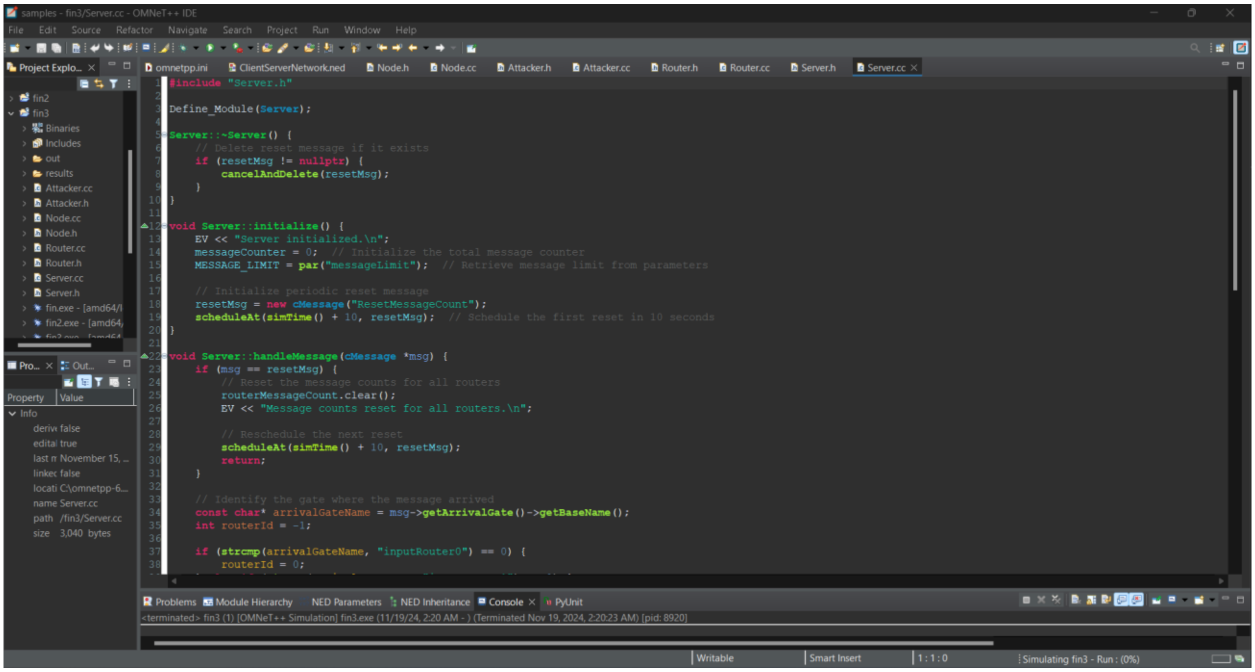
Routetr.cc

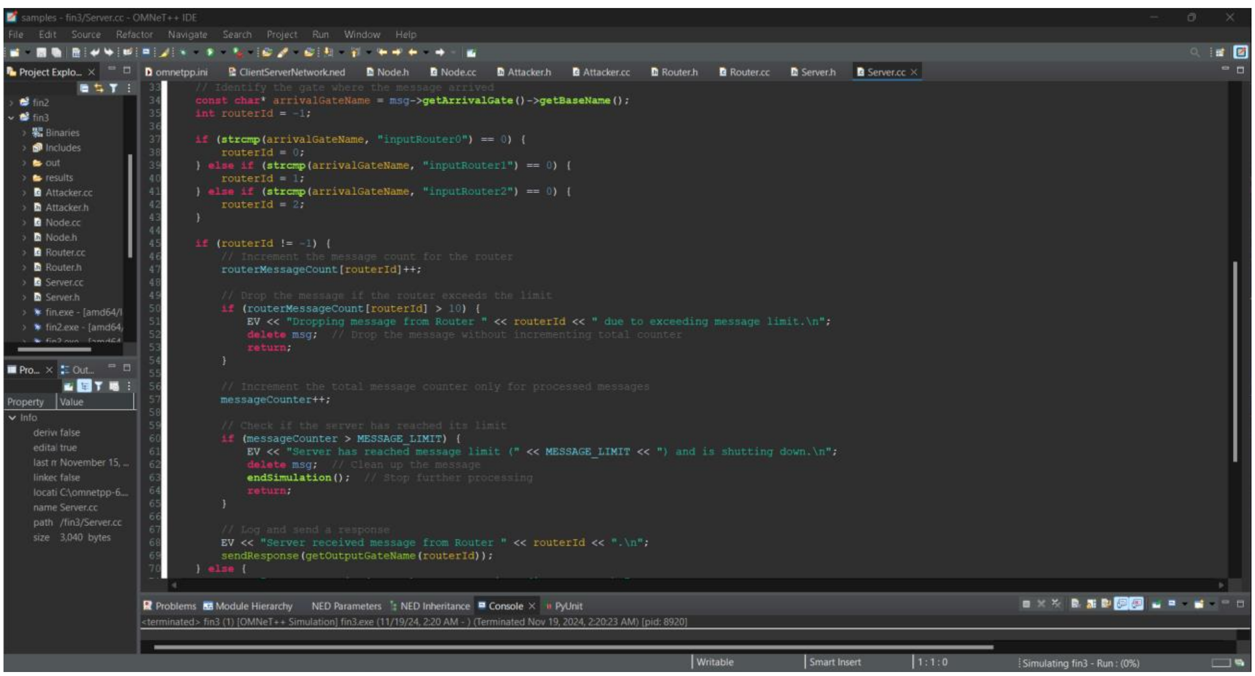


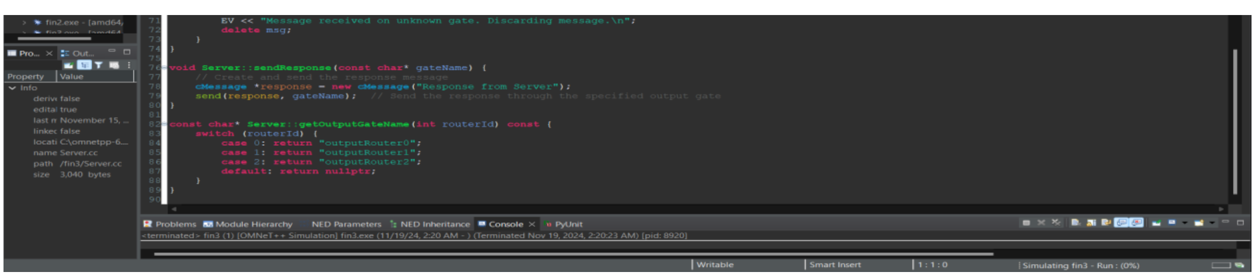
Server.h



Server.cc


samples - fin3/Server.cc - OMNeT++ IDE
#include "Server.h"
Define_Module(Server);
Server::~Server() {
    // Delete reset message if it exists
    if (resetMsg != nullptr) {
        cancelAndDelete(resetMsg);
    }
}
void Server::initialize() {
    EV << "Server initialized.\n";
    messageCounter = 0; // Initialize the total message counter
    MESSAGE_LIMIT = par("messageLimit"); // Retrieve message limit from parameters
    // Initialize periodic reset message
    resetMsg = new cMessage("ResetMessageCount");
    scheduleAt(simTime() + 10, resetMsg); // Schedule the first reset in 10 seconds
}
void Server::handleMessage(cMessage *msg) {
    if (msg == resetMsg) {
        // Reset the message counts for all routers
        routerMessageCount.clear();
        EV << "Message counts reset for all routers.\n";
        // Reschedule the next reset
        scheduleAt(simTime() + 10, resetMsg);
        return;
    }
    // Identify the gate where the message arrived
    const char* arrivalGateName = msg->getArrivalGate()->getBaseName();
    int routerId = -1;
    if (strcmp(arrivalGateName, "inputRouter0") == 0) {
        routerId = 0;


samples - fin3/Server.cc - OMNeT++ IDE
    // Identify the gate where the message arrived
    const char* arrivalGateName = msg->getArrivalGate()->getBaseName();
    int routerId = -1;
    if (strcmp(arrivalGateName, "inputRouter0") == 0) {
        routerId = 0;
    } else if (strcmp(arrivalGateName, "inputRouter1") == 0) {
        routerId = 1;
    } else if (strcmp(arrivalGateName, "inputRouter2") == 0) {
        routerId = 2;
    }
    if (routerId != -1) {
        // Increment the message count for the router
        routerMessageCount[routerId]++;
        // Drop the message if the router exceeds the limit
        if (routerMessageCount[routerId] > 10) {
            EV << "Dropping message from Router " << routerId << " due to exceeding message limit.\n";
            delete msg; // Drop the message without incrementing total counter
            return;
        }
        // Increment the total message counter only for processed messages
        messageCounter++;
        // Check if the server has reached its limit
        if (messageCounter > MESSAGE_LIMIT) {
            EV << "Server has reached message limit (" << MESSAGE_LIMIT << ") and is shutting down.\n";
            delete msg; // Clean up the message
            endSimulation(); // Stop further processing
            return;
        }
        // Log and send a response
        EV << "Server received message from Router " << routerId << ".\n";
        sendResponse(getOutputGateName(routerId));
    } else {


        EV << "Message received on unknown gate. Discarding message.\n";
        delete msg;
    }
}
void Server::sendResponse(const char* gateName) {
    // Create and send the response message
    cMessage *response = new cMessage("Response from Server");
    send(response, gateName); // Send the response through the specified output gate
}
const char* Server::getOutputGateName(int routerId) const {
    switch (routerId) {
        case 0: return "outputRouter0";
        case 1: return "outputRouter1";
        case 2: return "outputRouter2";
        default: return nullptr;
    }
}

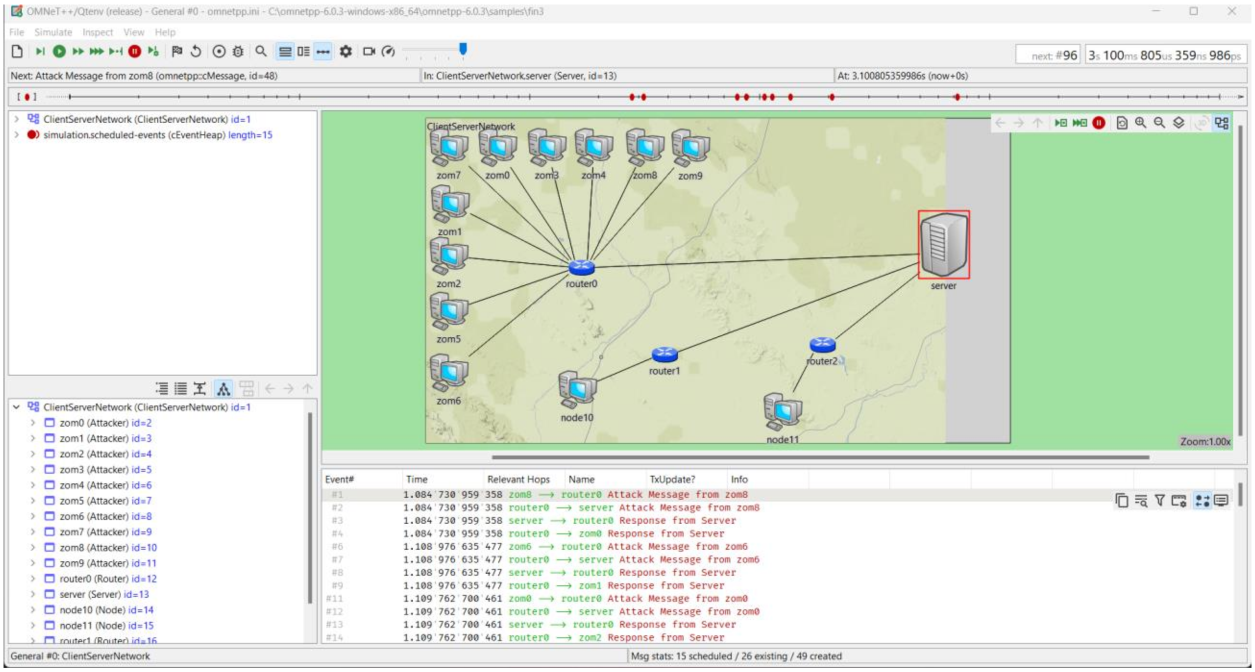
Mitigation Technique

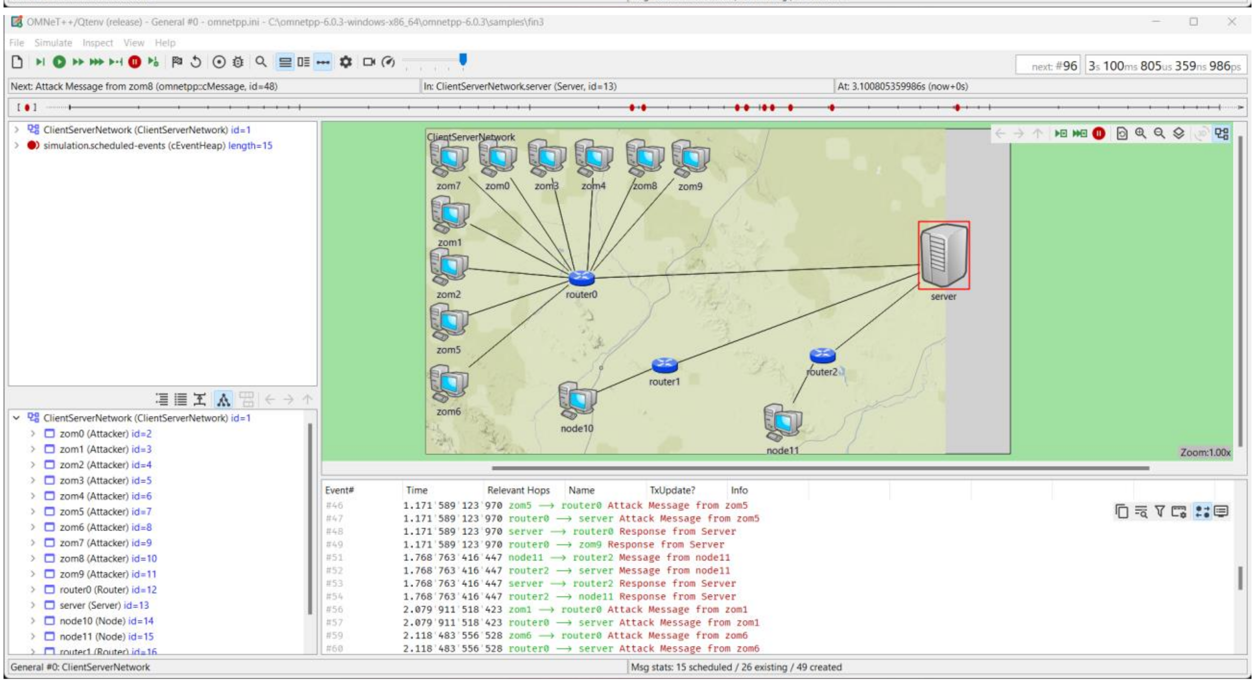
The Mitigation technique a combination of:

- **Rate-Based Filtering:** Dropping messages from sources (router0 in this case) that exceed a certain limit of messages per second.
- **Traffic Shaping:** Allocating bandwidth or server resources to legitimate traffic while denying resources to suspected attackers.
- **Blackhole Routing (Simplified):** While not permanently blackholing traffic, you are effectively discarding packets from router0 to safeguard resources.

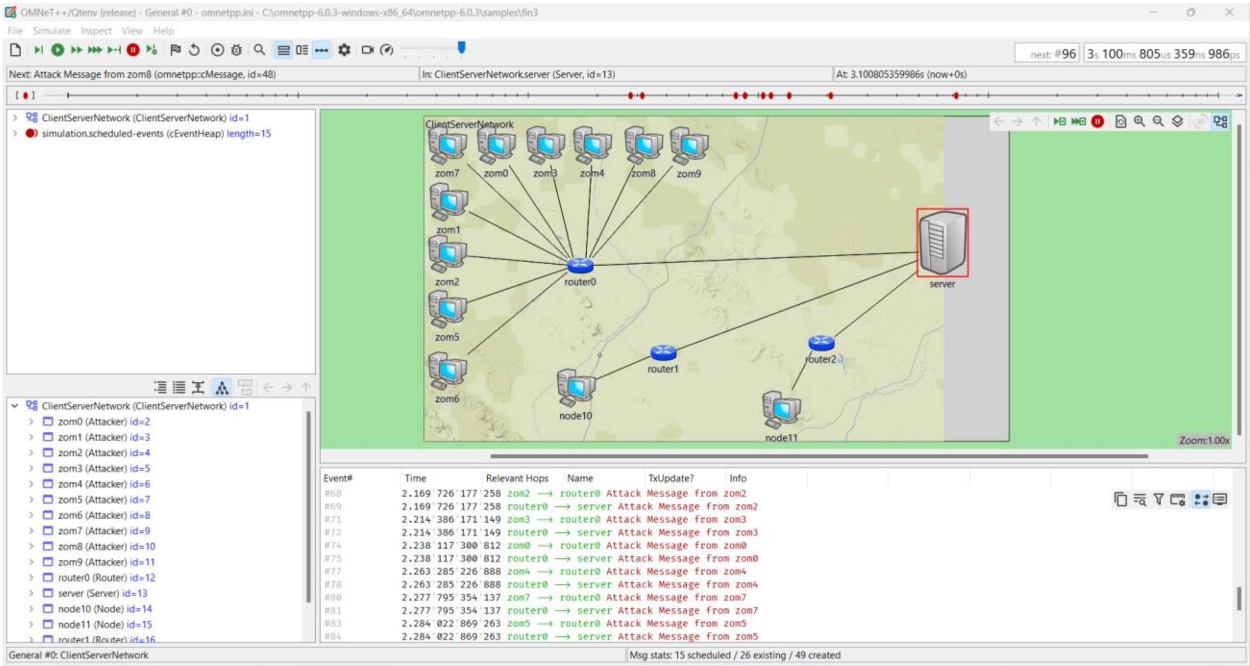
Result

First when we run the simulation, the netwrok should send work just fine, attacker (router0) send a lot of msgs using 10 nodes (zom0 – zom9) in the other hand the nodes (node10 – node11) work as a normal end points.

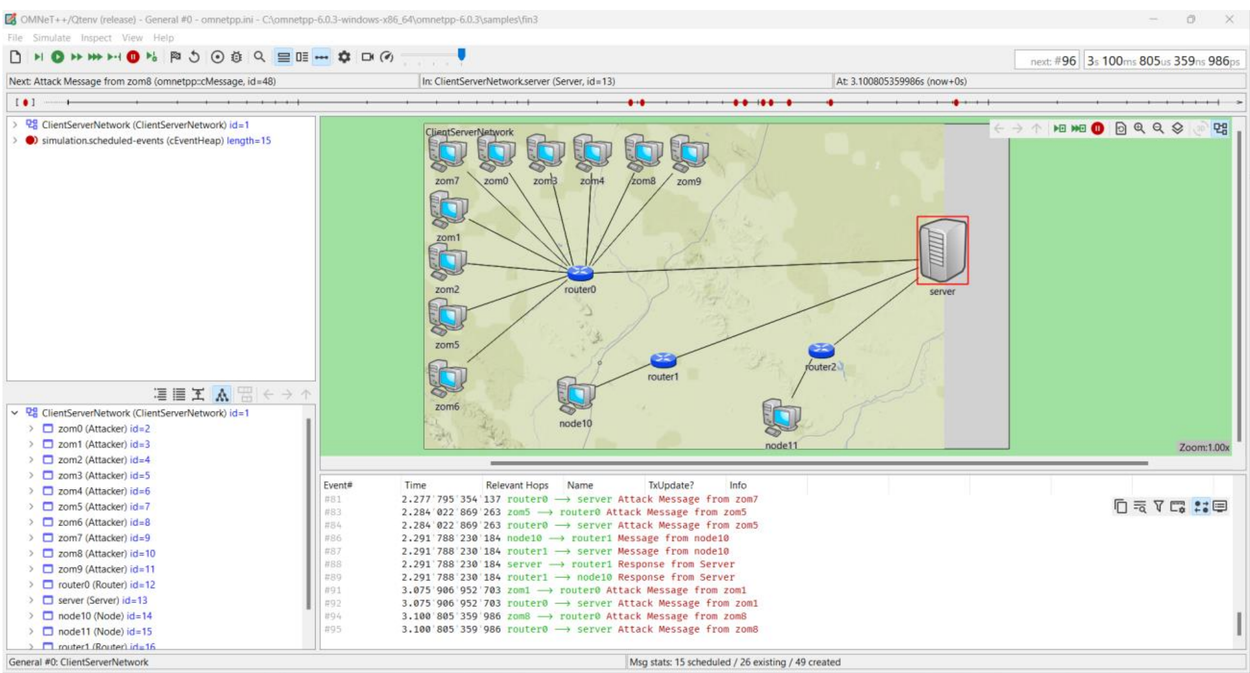




After hitting the limit now the server would not response the attacker



althought the server will response to the noraml nodes (node10 – node11)



Conclusion

The OMNeT++ DDoS simulation showed how a server handles a denial-of-service attack by dropping messages from attacker nodes in Network0 while still responding to normal nodes in Network1 and Network2. This helped demonstrate the impact of DDoS attacks and how servers can be designed to manage them effectively.

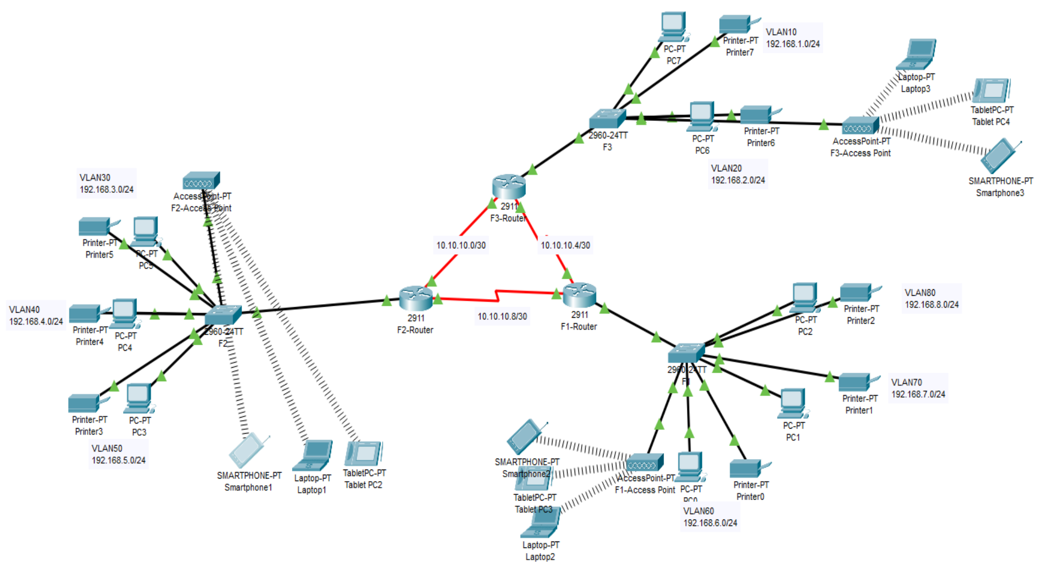
Chapter II, Secure Network Simulation

Introduction

The network topology is designed to efficiently manage devices across multiple VLANs with connectivity and routing ensured through routers and switches. The primary goal is to analyze the provided network diagram, detailing its components, structure, and functionality, while incorporating measures to enhance security and prevent potential attacks.

Network Overview

The diagram illustrates a hierarchical network topology comprising three routers , multiple switches , and access points to support wired and wireless connections. It is segmented into VLANs for logical isolation, security, and improved traffic management. The topology interconnects devices such as PCs, printers, laptops, tablets, and smartphones.



Key Components

1. Routers :

- Core routers (F1, F2, F3) are interconnected to provide redundancy and ensure failover using point-to-point connections.
- Configured for inter-VLAN routing and subnet allocation.

2. Switches :

- Layer 2 devices connecting end devices such as PCs and printers.
- VLAN tagging is implemented to segregate traffic.

3. Access Points:

- Provide wireless connectivity for mobile devices like laptops, smartphones, and tablets.
- WPA2-PSK encryption is recommended for secure wireless communication.

4. End Devices:

- Include PCs, printers, laptops, tablets, and smartphones spread across various VLANs.

VLAN Configuration

The network uses VLANs to logically segregate traffic, improving management and security:

- VLAN10 (192.168.1.0/24): Devices such as PC7 and Printer7.
- VLAN20 (192.168.2.0/24): Devices such as PC6 and Printer6.
- VLAN30 (192.168.3.0/24): Devices such as PC5 and Printer5.
- VLAN40 (192.168.4.0/24): Devices such as PC4 and Printer4.
- VLAN50 (192.168.5.0/24): Devices such as PC3 and Printer3.
- VLAN60 (192.168.6.0/24): Devices such as PC0 and Printer0.
- VLAN70 (192.168.7.0/24): Devices such as PC1 and Printer1.
- VLAN80 (192.168.8.0/24): Devices such as PC2 and Printer2.

Security Measures

To ensure the network is protected against potential attacks:

1. **Enabling Port Security**

    - Only authorized devices can connect to specific ports.
    - Prevents unauthorized users from accessing sensitive resources.
    - Automatically shuts down the port in case of a violation, stopping potential threats.

```
Switch#enable
Switch#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Switch(config)#interface FastEthernet0/2
Switch(config-if)#switchport port-security maximum 1
Switch(config-if)#switchport port-security violation shutdown
Switch(config-if)#end
Switch#
%SYS-5-CONFIG_I: Configured from console by console

Switch#show port-security
Secure Port MaxSecureAddr CurrentAddr SecurityViolation Security Action
               (Count)       (Count)        (Count)
--------------------------------------------------------------------
        Fa0/2        1           1                 0         Shutdown
----------------------------------------------------------------------
Switch#
```

2. **Enable DHCP Snooping**

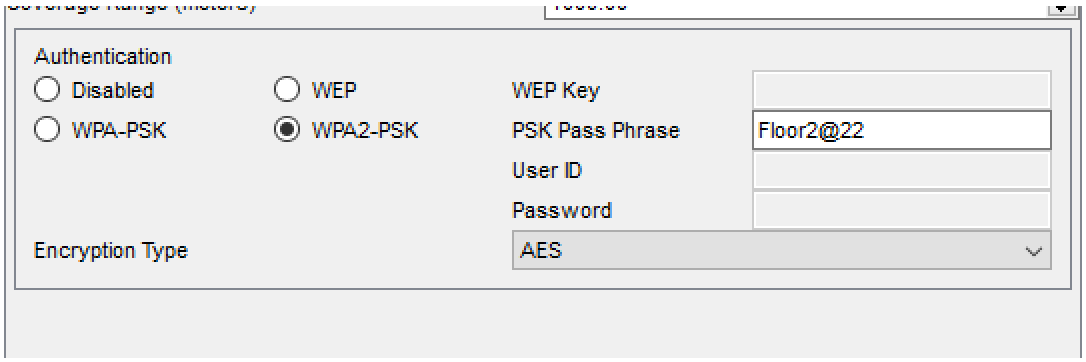
    - Ensures devices get IP addresses only from trusted DHCP servers.
    - Protects users from man-in-the-middle (MITM) and denial-of-service (DoS) attacks.
    - Maintains accurate DHCP bindings, helping secure network communications.

```
Switch(config)#ip dhcp snooping vlan 80
Switch(config)#ip dhcp snooping vlan 70
Switch(config)#ip dhcp snooping vlan 60
Switch(config)#interface FastEthernet0/1
```

```
Switch(config)#interface FastEthernet0/1
Switch(config-if)#ip dhcp snooping trust
Switch(config-if)#exit
Switch(config)#exit
Switch#show ip dhcp snooping
Switch DHCP snooping is enabled
DHCP snooping is configured on following VLANs:
60,70,80
Insertion of option 82 is enabled
Option 82 on untrusted port is not allowed
Verification of hwaddr field is enabled
Interface                  Trusted    Rate limit (pps)
-----------------------    -------    ----------------
FastEthernet0/1            yes        unlimited
Switch#
%SYS-5-CONFIG_I: Configured from console by console
```

3. Enabling WPA2-PSK for Wireless Networks

- Data is encrypted between devices and access points, protecting against eavesdropping and packet sniffing.
- Access is restricted to devices with the correct passphrase, preventing unauthorized access.
- Secures the wireless network against brute-force attacks (strong passphrases recommended).



4. Configuring a Switch for SSH

- Encrypts remote management sessions, protecting against eavesdropping.
- Ensures only authorized personnel can manage network devices.
- Provides secure authentication with user accounts and password encryption.

```
F2-Router(config)#crypto key generate rsa
*Mar 1 0:58:11.983: RSA key size needs to be at least 768 bits for ssh version 2
*Mar 1 0:58:11.983: %SSH-5-ENABLED: SSH 1.5 has been enabled
% You already have RSA keys defined named F2-Router.AK .
% Do you really want to replace them? [yes/no]: (Choose a key size of 1024 bits or
higher)
% Please answer 'yes' or 'no'.
% Do you really want to replace them? [yes/no]: yes
The name for the keys will be: F2-Router.AK
Choose the size of the key modulus in the range of 360 to 4096 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]: 1024
% Generating 1024 bit RSA keys, keys will be non-exportable...[OK]

F2-Router(config)#username admin privilege 15 password SecureP@ssword
*Mar 1 0:58:43.494: %SSH-5-ENABLED: SSH 1.99 has been enabled
F2-Router(config)#line vty 0 4
F2-Router(config-line)#transport input ssh
F2-Router(config-line)#login local
F2-Router(config-line)#exit
F2-Router(config)#service password-encryption
F2-Router(config)#
```

```
F2-Router#show ip ssh
SSH Enabled - version 1.99
Authentication timeout: 120 secs; Authentication retries: 3
F2-Router#show run
Building configuration...

Current configuration : 1857 bytes
!
version 15.1
no service timestamps log datetime msec
no service timestamps debug datetime msec
service password-encryption
!
hostname F2-Router
!
!
!
!
!
ip dhcp pool finance
 network 192.168.5.0 255.255.255.0
 default-router 192.168.5.1
 dns-server 192.168.5.1
ip dhcp pool HR
 network 192.168.4.0 255.255.255.0
 default-router 192.168.4.1
 --More--
```

Copy

Routing

- Inter-VLAN Routing: Routers provide communication between VLANs with ACLs to ensure only authorized traffic is allowed.
- Subnet Allocation: Use /30 subnets for point-to-point links, minimizing IP address waste and reducing attack surfaces.

Redundancy and Scalability

- The triangular interconnection of routers ensures redundancy and improves failover capabilities in case of link or device failure.
- VLAN segmentation allows easy scaling of the network as new devices and subnets can be added without significant reconfiguration.

Benefits of the Secure Network Design

1. Enhanced Security: Measures such as ACLs, encryption, and firewalls safeguard against unauthorized access and attacks.
2. Network Segmentation: VLANs isolate traffic, improving security and performance.
3. Scalability and Reliability: Redundancy ensures continuous availability, while VLANs allow seamless expansion.
4. Proactive Monitoring: IDPS and logging provide early detection of threats, reducing the risk of compromise.

Conclusion

The network topology is designed with a focus on segmentation, redundancy, and security. By implementing the recommended security measures, the network can effectively protect against potential threats such as unauthorized access, DDoS attacks, and VLAN hopping, ensuring a robust and reliable infrastructure.

References

- F. F., "Building Mitigation Request Protocol," *RFC 9132*, Internet Engineering Task Force, Dec. 2021. [Online]. Available: https://www.rfc-editor.org/rfc/rfc9132.html#name-building-mitigation-request
- Imperva, "What is Rate Limiting?" [Online]. Available: https://www.imperva.com/learn/application-security/rate-limiting/
- Ambassador Labs, "How to Configure Rate Limits to Prevent DDoS: Best Practices," 2021. [Online]. Available: https://www.getambassador.io/blog/configure-rate-limits-prevent-ddos-best-practices
- Internet Engineering Task Force, "Internet Denial-of-Service Considerations," *RFC 4732*. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc4732
- Imperva, "Rate Limiting as a DDoS Mitigation Strategy." [Online]. Available: https://www.imperva.com/learn/application-security/rate-limiting/#:~:text=Rate%20limiting%20mitigates%20DDoS%20threats,sometimes%20millions%20of%20IP%20addresses
- DDoS-Guard, "Traffic Filtering Methods." [Online]. Available: https://ddos-guard.net/blog/traffic-filtering-methods
- Cisco, "VLAN Configuration and Management," *Cisco Official Documentation*. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/switches/datacenter/sw/5_x/nx-os/layer2/configuration/guide/Cisco_Nexus_7000_Series_NX-OS_Layer_2_Switching_Configuration_Guide_Release_5-x_chapter4.html
- Cisco, "Access Control Lists (ACLs) and Best Practices," *Cisco Security Documentation*. [Online]. Available: https://www.cisco.com/c/en/us/support/docs/security/ios-firewall/23602-confaccesslists.html
- Catchpoint, "Inter-VLAN Routing and Subnetting," *Network Admin Guide*. [Online]. Available: https://www.catchpoint.com/network-admin-guide/inter-vlan-routing