Dos Attack in Internetworks

Software Used: NetSim Standard v13.0 (32/64 bit), Visual Studio 2019

Project Download Link:

https://github.com/NetSim-TETCOS/DOS_Attack_in_Internetworks_v13.0/archive/refs/heads/main.zip

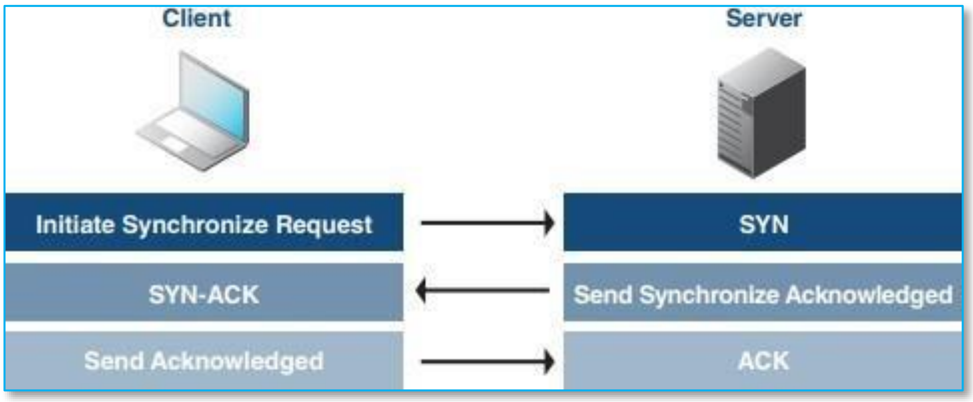
A Denial of Service (DoS) attack is an attempt to make a system unavailable to the intended user(s), such as preventing access to a website. A successful DoS attack consumes all available network or system resources, usually resulting in a slowdown or server crash. Whenever multiple sources are coordinating in the DoS attack, it becomes known as a DDoS (Distributed Denial of Service) attack. **Standard DDoS Attack types:**

1. SYN Flood
2. UDP Flood
3. SMBLoris
4. ICMP Flood
5. HTTP GET Flood

SYN Flood:

TCP SYN floods are DoS attacks that attempt to flood the DNS server with new TCP connection requests. Normally, a client initiates a TCP connection through a three-way handshake of messages:

- The client requests a connection by sending a SYN (synchronize) message to the server.
- The server acknowledges the request by sending SYN-ACK back to the client.
- The client answers with a responding ACK, establishing the connection.



This triple exchange is the foundation for every connection established using the Transmission Control Protocol (TCP). A SYN Flood is one of the most common forms of DDoS attacks. It occurs when an attacker sends a succession of TCP Synchronize (SYN) requests to the target in an attempt to consume enough resources to make the server unavailable for legitimate users. This works because a SYN request opens network communication between a prospective client and the target server. When the server receives a SYN request, it responds acknowledging the request and holds the communication open while it waits for the client to acknowledge the open connection. However, in a successful SYN Flood, the client acknowledgment never arrives, thus consuming the server's resources until the connection times out. A large number of incoming SYN requests to the target server exhausts all available server resources and results in a successful DoS attack. Before implementing this project in NetSim, users have to understand the steps given below:

1. TCP Log file

- User need to understand the TCP log file which will get created in the temp path of NetSim <Windows Temp Folder>/NetSim>
- The TCP Log file is usually a very large file and hence is disabled by default in NetSim.
- To enable logging, go to TCP.c inside the TCP project and change the function bool isTCPlog() to return true instead of false.

2. At malicious node:

Create a new timer event called SYN_FLOOD in TCP for sending TCP_SYN packets that should be triggered for every 1000 micro seconds. This will create and send the TCP_SYN packet for every 1000 micro seconds. SYN request opens network communication between a client and the target

3. At Target node:

When the target receives a SYN request, it responds acknowledging the request and holds the communication open while it waits for the client to acknowledge the open connection. If a SYN packet arrives at Receiver, it should reply with a SYN_ACK packet. For this SYN_ACK packet, add a processing time of 2000 micro seconds in Ethernet Physical Out. This delays the arrival of SYN_ACK at source node. During this delay, another SYN packet will get created at the malicious node. A large number of incoming SYN requests to the target exhausts all available server resources and results in a successful DoS attack **SYN_FLOOD in NetSim:**

To implement this project in NetSim, we have created SYN_FLOOD.c file inside TCP project. The file contains the following functions:

- int is_malicious_node();

  This function is used to check the node is malicious node or not

- int socket_creation();

  This function is used to create a new socket and update the socket parameters

- static void send_syn_packet(PNETSIM_SOCKET s);

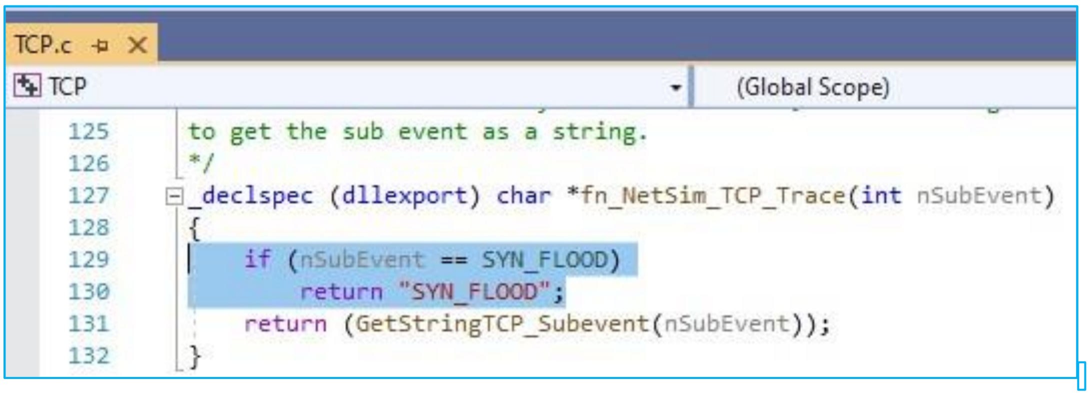
  This function is used to create and send SYN packet to the network layer

- void syn_flood();

This function is used to check whether the socket is present or not and also adds a timer event called SYN_FLOOD (triggers for every 1000µs)

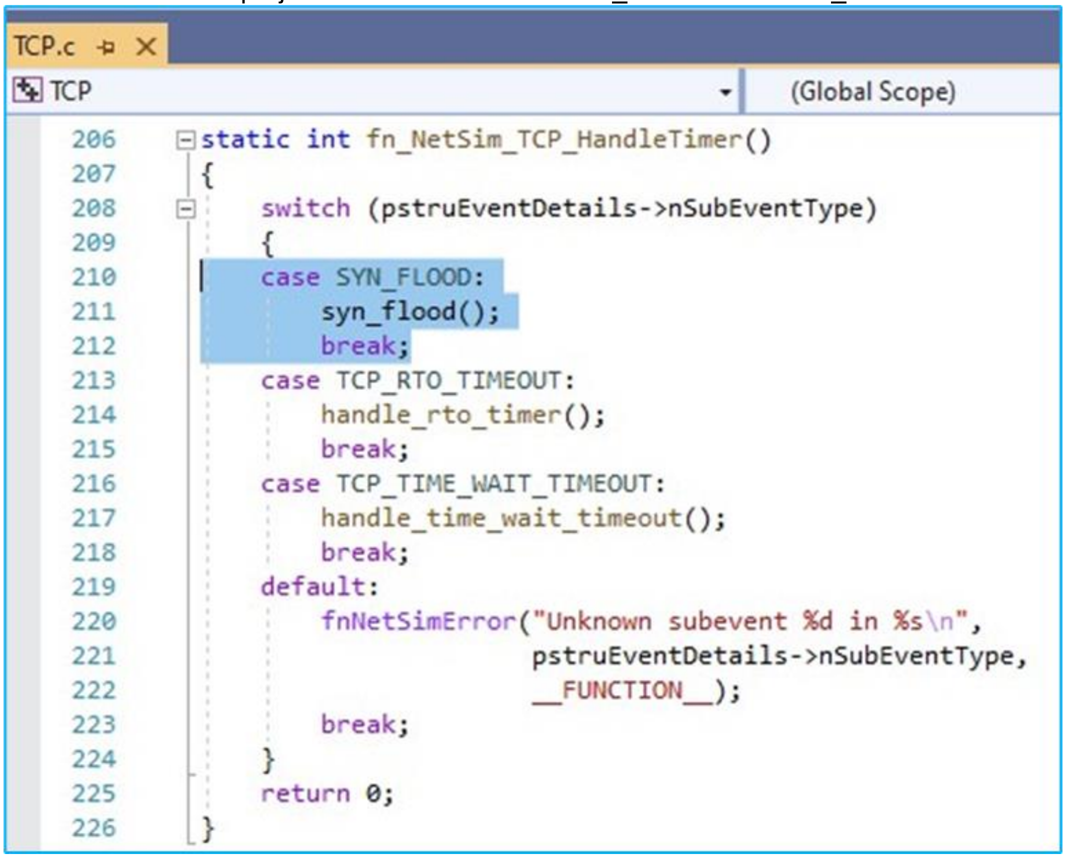
Code modifications done in NetSim:

1. We have added the following lines of code in fn_NetSim_TCP_Trace() function present in TCP.c file inside TCP project. This is used to add the SYN_FLOOD sub-events in Event Trace file



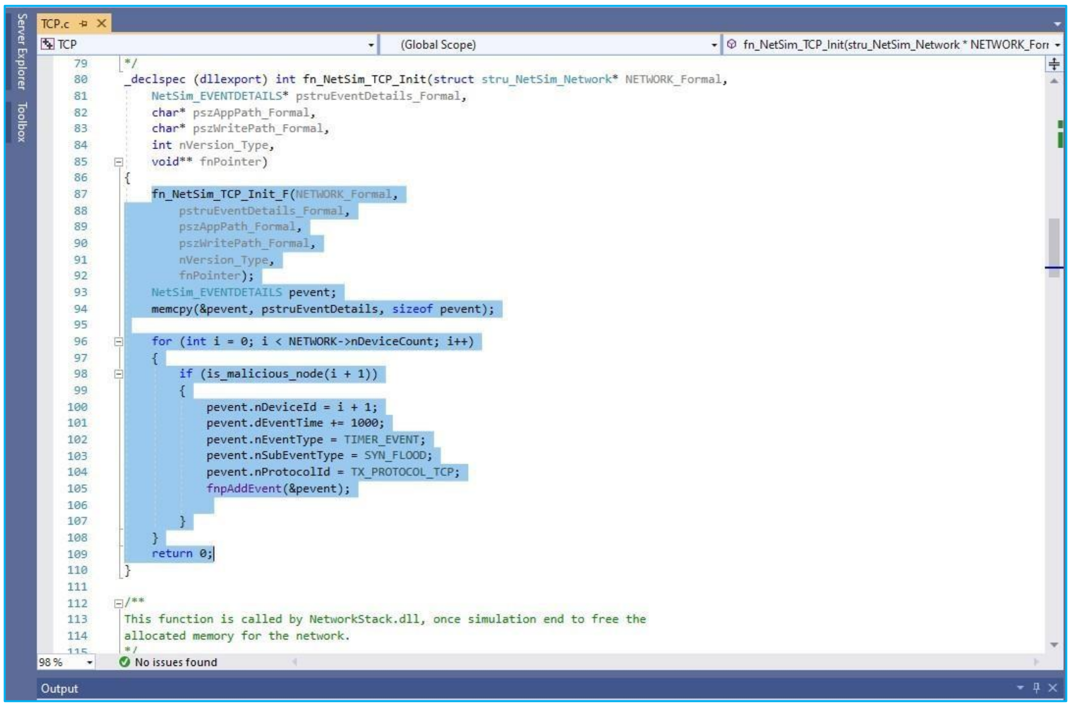
```
to get the sub event as a string.
*/
_declspec (dllexport) char *fn_NetSim_TCP_Trace(int nSubEvent)
{
    if (nSubEvent == SYN_FLOOD)
        return "SYN_FLOOD";
    return (GetStringTCP_Subevent(nSubEvent));
}
```

2. We have added the following lines of code in fn_NetSim_TCP_HandleTimer() function present in TCP.c file inside TCP project. Used to add a TCP sub_event called SYN_FLOOD



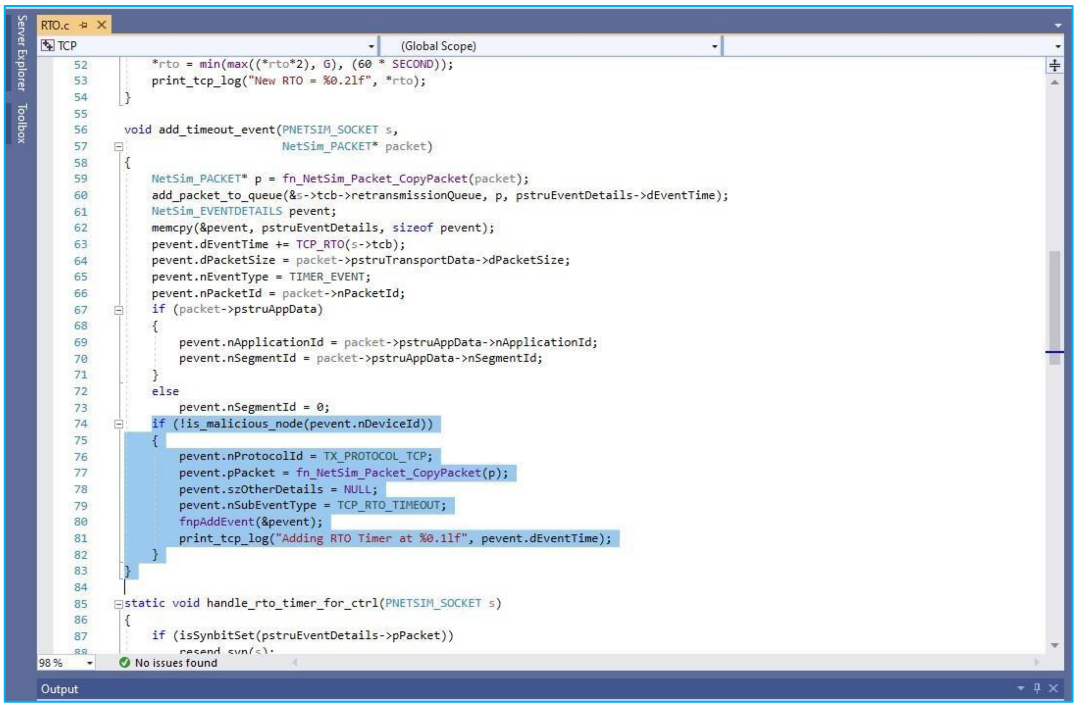
```
static int fn_NetSim_TCP_HandleTimer()
{
    switch (pstruEventDetails->nSubEventType)
    {
    case SYN_FLOOD:
        syn_flood();
        break;
    case TCP_RTO_TIMEOUT:
        handle_rto_timer();
        break;
    case TCP_TIME_WAIT_TIMEOUT:
        handle_time_wait_timeout();
        break;
    default:
        fnNetSimError("Unknown subevent %d in %s\n",
                      pstruEventDetails->nSubEventType,
                      __FUNCTION__);
        break;
    }
    return 0;
}
```

3. And modified the following lines of code in fn_NetSim_TCP_Init() function resent in TCP.c inside TCP project



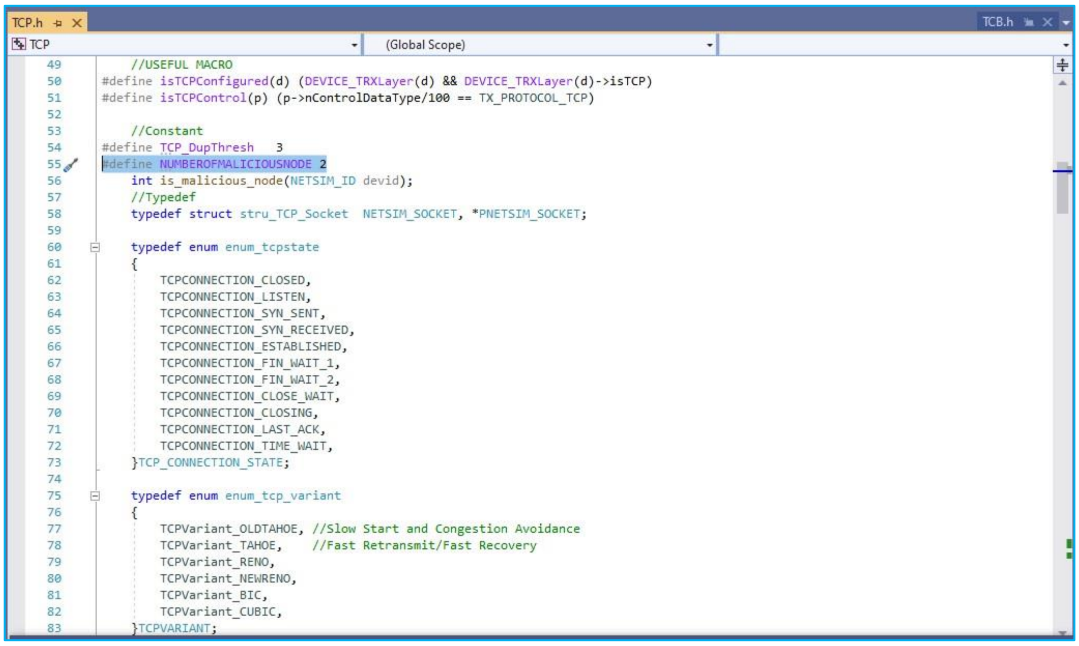
```c
*/
_declspec (dllexport) int fn_NetSim_TCP_Init(struct stru_NetSim_Network* NETWORK_Formal,
    NetSim_EVENTDETAILS* pstruEventDetails_Formal,
    char* pszAppPath_Formal,
    char* pszWritePath_Formal,
    int nVersion_Type,
    void** fnPointer)
{
    fn_NetSim_TCP_Init_F(NETWORK_Formal,
        pstruEventDetails_Formal,
        pszAppPath_Formal,
        pszWritePath_Formal,
        nVersion_Type,
        fnPointer);
    NetSim_EVENTDETAILS pevent;
    memcpy(&pevent, pstruEventDetails, sizeof pevent);

    for (int i = 0; i < NETWORK->nDeviceCount; i++)
    {
        if (is_malicious_node(i + 1))
        {
            pevent.nDeviceId = i + 1;
            pevent.dEventTime += 1000;
            pevent.nEventType = TIMER_EVENT;
            pevent.nSubEventType = SYN_FLOOD;
            pevent.nProtocolId = TX_PROTOCOL_TCP;
            fnpAddEvent(&pevent);

        }
    }
    return 0;
}

/**
This function is called by NetworkStack.dll, once simulation end to free the
allocated memory for the network.
*/
```

4. And modified the following lines of code in add_timeout_event() present in RTO.c file inside TCP project which avoids RTO timer for malicious nodes



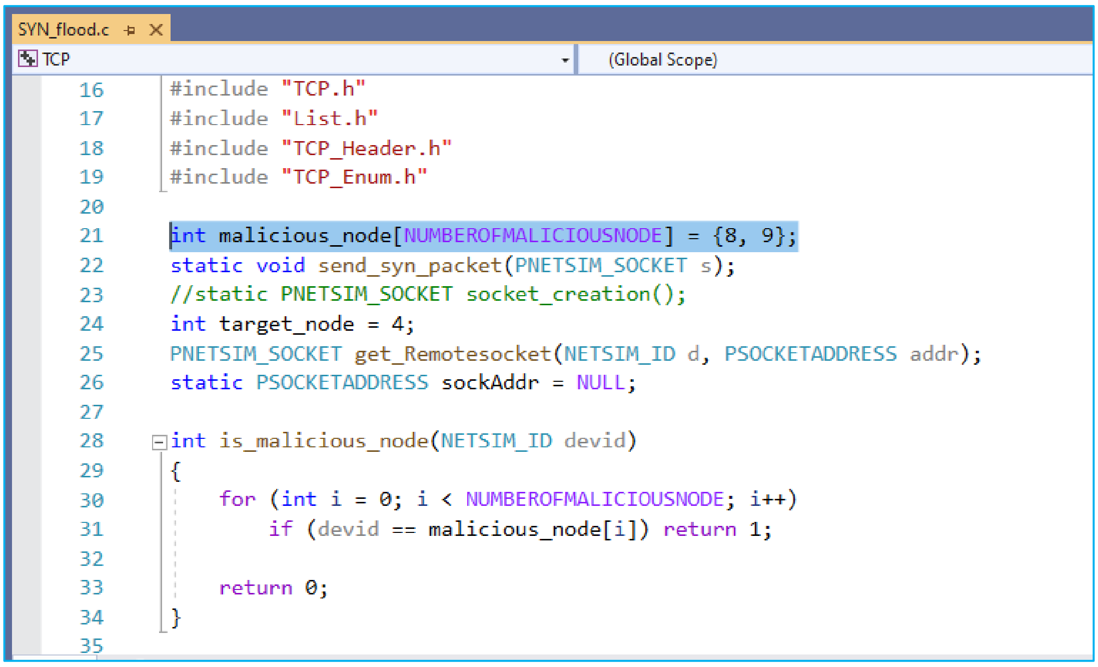
```c
    *rto = min(max((*rto*2), G), (60 * SECOND));
    print_tcp_log("New RTO = %0.2lf", *rto);
}

void add_timeout_event(PNETSIM_SOCKET s,
                       NetSim_PACKET* packet)
{
    NetSim_PACKET* p = fn_NetSim_Packet_CopyPacket(packet);
    add_packet_to_queue(&s->tcb->retransmissionQueue, p, pstruEventDetails->dEventTime);
    NetSim_EVENTDETAILS pevent;
    memcpy(&pevent, pstruEventDetails, sizeof pevent);
    pevent.dEventTime += TCP_RTO(s->tcb);
    pevent.dPacketSize = packet->pstruTransportData->dPacketSize;
    pevent.nEventType = TIMER_EVENT;
    pevent.nPacketId = packet->nPacketId;
    if (packet->pstruAppData)
    {
        pevent.nApplicationId = packet->pstruAppData->nApplicationId;
        pevent.nSegmentId = packet->pstruAppData->nSegmentId;
    }
    else
        pevent.nSegmentId = 0;
    if (!is_malicious_node(pevent.nDeviceId))
    {
        pevent.nProtocolId = TX_PROTOCOL_TCP;
        pevent.pPacket = fn_NetSim_Packet_CopyPacket(p);
        pevent.szOtherDetails = NULL;
        pevent.nSubEventType = TCP_RTO_TIMEOUT;
        fnpAddEvent(&pevent);
        print_tcp_log("Adding RTO Timer at %0.1lf", pevent.dEventTime);
    }
}

static void handle_rto_timer_for_ctrl(PNETSIM_SOCKET s)
{
    if (isSynbitSet(pstruEventDetails->pPacket))
        resend_syn(s);
```

5. Users can give their own number of malicious node in **TCP.h** file inside TCP project



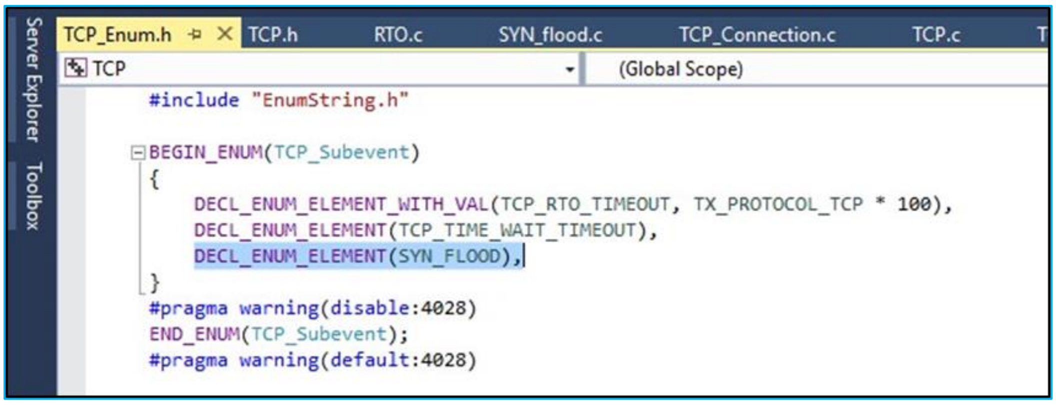
```c
    //USEFUL MACRO
#define isTCPConfigured(d) (DEVICE_TRXLayer(d) && DEVICE_TRXLayer(d)->isTCP)
#define isTCPControl(p) (p->nControlDataType/100 == TX_PROTOCOL_TCP)

    //Constant
#define TCP_DupThresh   3
#define NUMBEROFMALICIOUSNODE 2
    int is_malicious_node(NETSIM_ID devid);
    //Typedef
    typedef struct stru_TCP_Socket  NETSIM_SOCKET, *PNETSIM_SOCKET;

    typedef enum enum_tcpstate
    {
        TCPCONNECTION_CLOSED,
        TCPCONNECTION_LISTEN,
        TCPCONNECTION_SYN_SENT,
        TCPCONNECTION_SYN_RECEIVED,
        TCPCONNECTION_ESTABLISHED,
        TCPCONNECTION_FIN_WAIT_1,
        TCPCONNECTION_FIN_WAIT_2,
        TCPCONNECTION_CLOSE_WAIT,
        TCPCONNECTION_CLOSING,
        TCPCONNECTION_LAST_ACK,
        TCPCONNECTION_TIME_WAIT,
    }TCP_CONNECTION_STATE;

    typedef enum enum_tcp_variant
    {
        TCPVariant_OLDTAHOE, //Slow Start and Congestion Avoidance
        TCPVariant_TAHOE,    //Fast Retransmit/Fast Recovery
        TCPVariant_RENO,
        TCPVariant_NEWRENO,
        TCPVariant_BIC,
        TCPVariant_CUBIC,
    }TCPVARIANT;
```

6. Users can give their own target ID and malicious ID in **SYN_FLOOD.c** file inside TCP project



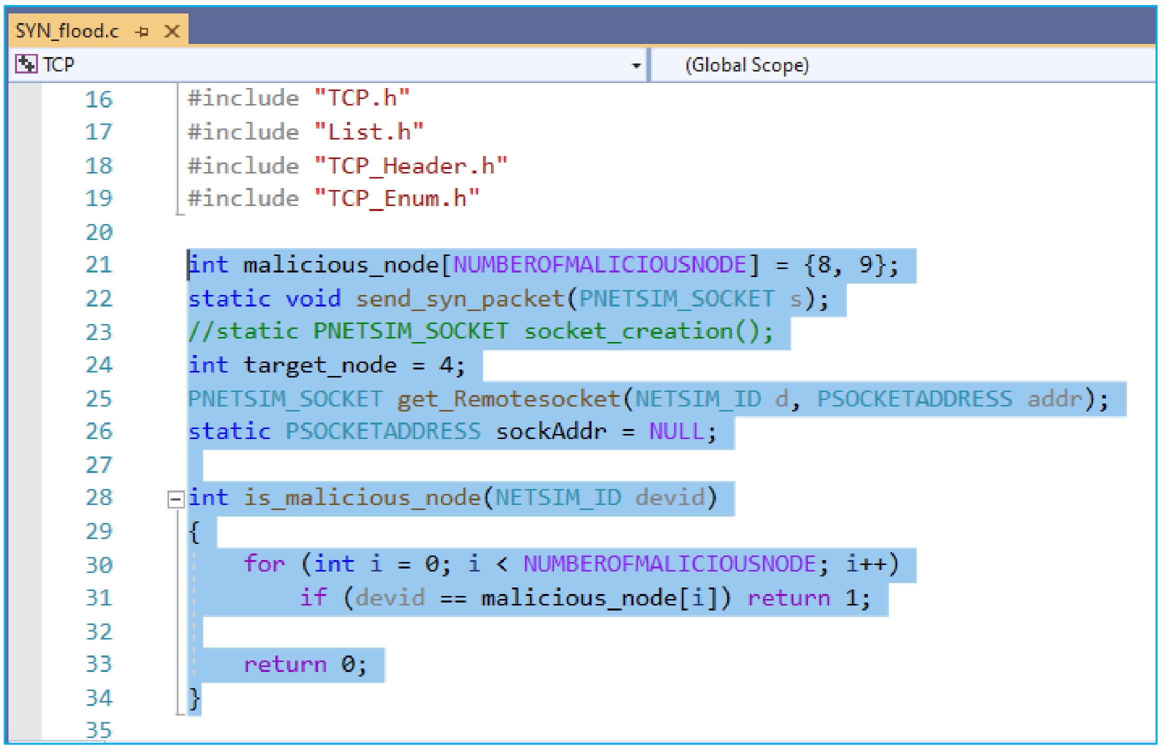
```c
#include "TCP.h"
#include "List.h"
#include "TCP_Header.h"
#include "TCP_Enum.h"

int malicious_node[NUMBEROFMALICIOUSNODE] = {8, 9};
static void send_syn_packet(PNETSIM_SOCKET s);
//static PNETSIM_SOCKET socket_creation();
int target_node = 4;
PNETSIM_SOCKET get_Remotesocket(NETSIM_ID d, PSOCKETADDRESS addr);
static PSOCKETADDRESS sockAddr = NULL;

int is_malicious_node(NETSIM_ID devid)
{
    for (int i = 0; i < NUMBEROFMALICIOUSNODE; i++)
        if (devid == malicious_node[i]) return 1;

    return 0;
}
```

7. Added the following line in TCP_Enum.h file inside TCP project to add a new TCP_subevent called SYN_FLOOD

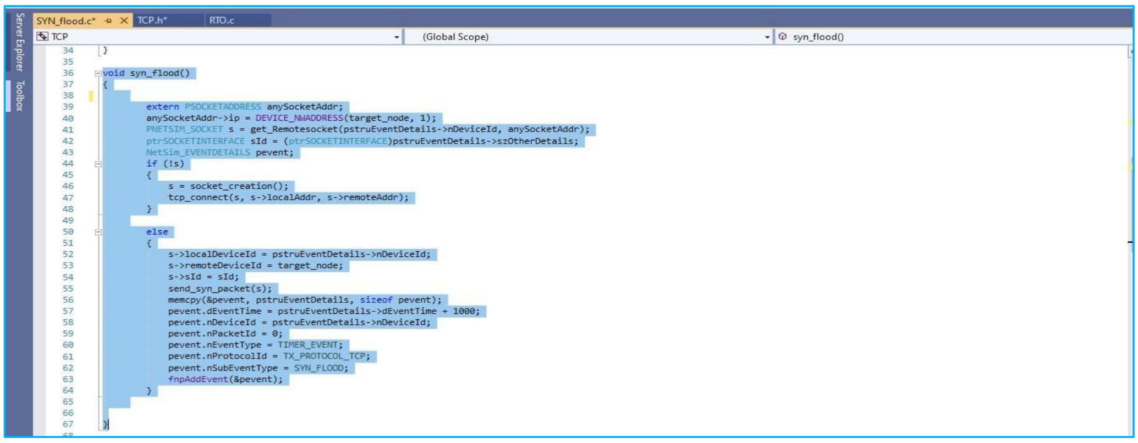


```c
#include "EnumString.h"

BEGIN_ENUM(TCP_Subevent)
{
    DECL_ENUM_ELEMENT_WITH_VAL(TCP_RTO_TIMEOUT, TX_PROTOCOL_TCP * 100),
    DECL_ENUM_ELEMENT(TCP_TIME_WAIT_TIMEOUT),
    DECL_ENUM_ELEMENT(SYN_FLOOD),
}
#pragma warning(disable:4028)
END_ENUM(TCP_Subevent);
#pragma warning(default:4028)
```

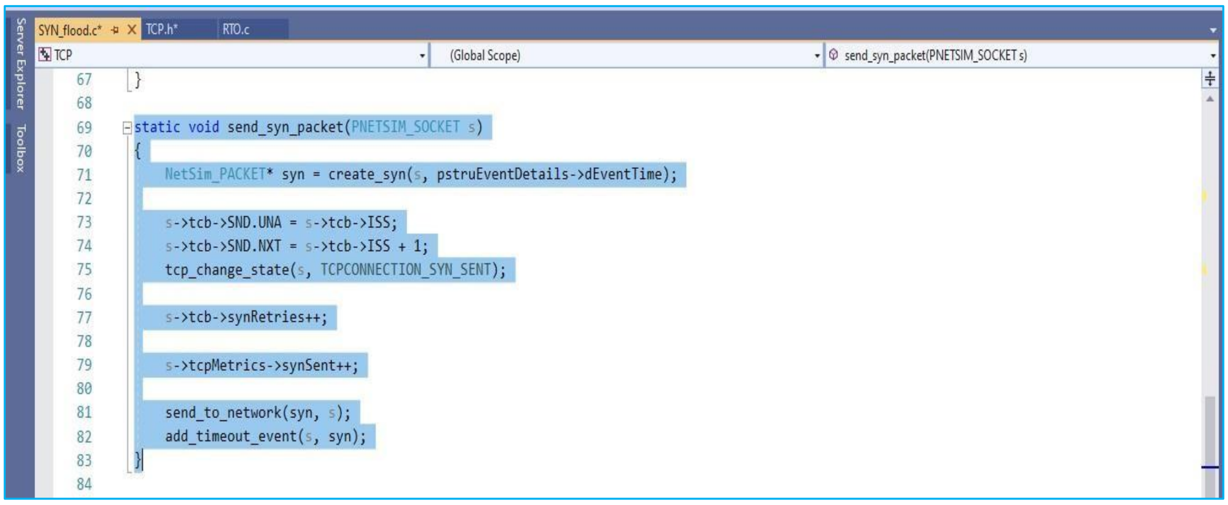
8. SYN_FLOOD.c file contains the following functions



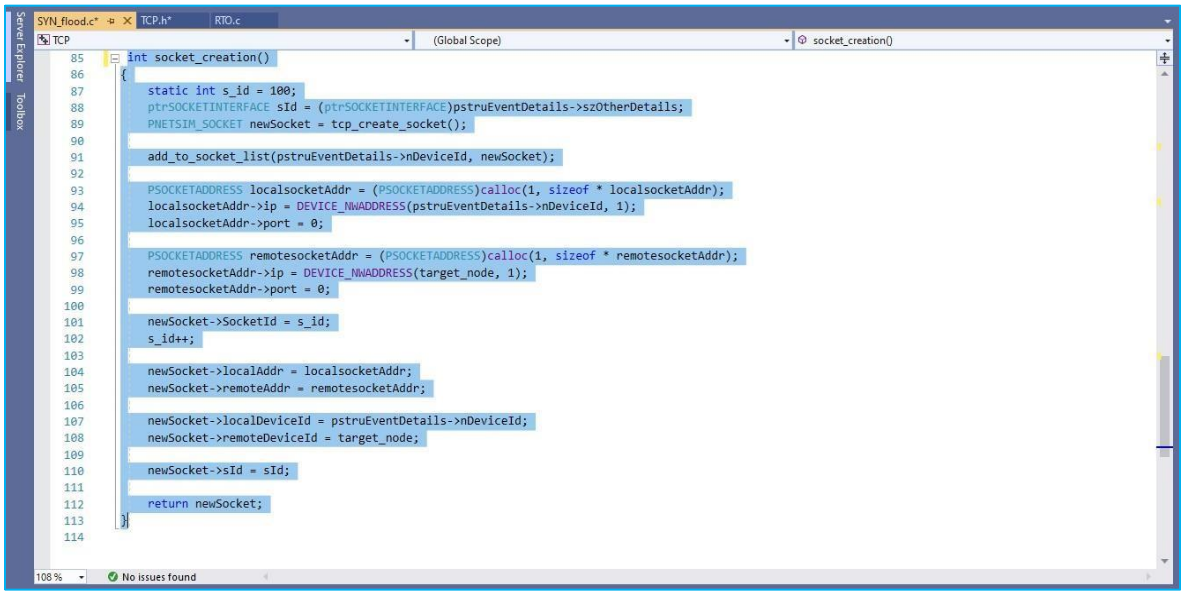
```c
#include "TCP.h"
#include "List.h"
#include "TCP_Header.h"
#include "TCP_Enum.h"

int malicious_node[NUMBEROFMALICIOUSNODE] = {8, 9};
static void send_syn_packet(PNETSIM_SOCKET s);
//static PNETSIM_SOCKET socket_creation();
int target_node = 4;
PNETSIM_SOCKET get_Remotesocket(NETSIM_ID d, PSOCKETADDRESS addr);
static PSOCKETADDRESS sockAddr = NULL;

int is_malicious_node(NETSIM_ID devid)
{
    for (int i = 0; i < NUMBEROFMALICIOUSNODE; i++)
        if (devid == malicious_node[i]) return 1;

    return 0;
}

```



```c
}

void syn_flood()
{

        extern PSOCKETADDRESS anySocketAddr;
        anySocketAddr->ip = DEVICE_NWADDRESS(target_node, 1);
        PNETSIM_SOCKET s = get_Remotesocket(pstruEventDetails->nDeviceId, anySocketAddr);
        ptrSOCKETINTERFACE sId = (ptrSOCKETINTERFACE)pstruEventDetails->szOtherDetails;
        NetSim_EVENTDETAILS pevent;
        if (!s)
        {
            s = socket_creation();
            tcp_connect(s, s->localAddr, s->remoteAddr);
        }

        else
        {
            s->localDeviceId = pstruEventDetails->nDeviceId;
            s->remoteDeviceId = target_node;
            s->sId = sId;
            send_syn_packet(s);
            memcpy(&pevent, pstruEventDetails, sizeof pevent);
            pevent.dEventTime = pstruEventDetails->dEventTime + 1000;
            pevent.nDeviceId = pstruEventDetails->nDeviceId;
            pevent.nPacketId = 0;
            pevent.nEventType = TIMER_EVENT;
            pevent.nProtocolId = TX_PROTOCOL_TCP;
            pevent.nSubEventType = SYN_FLOOD;
            fnpAddEvent(&pevent);
        }


}
```

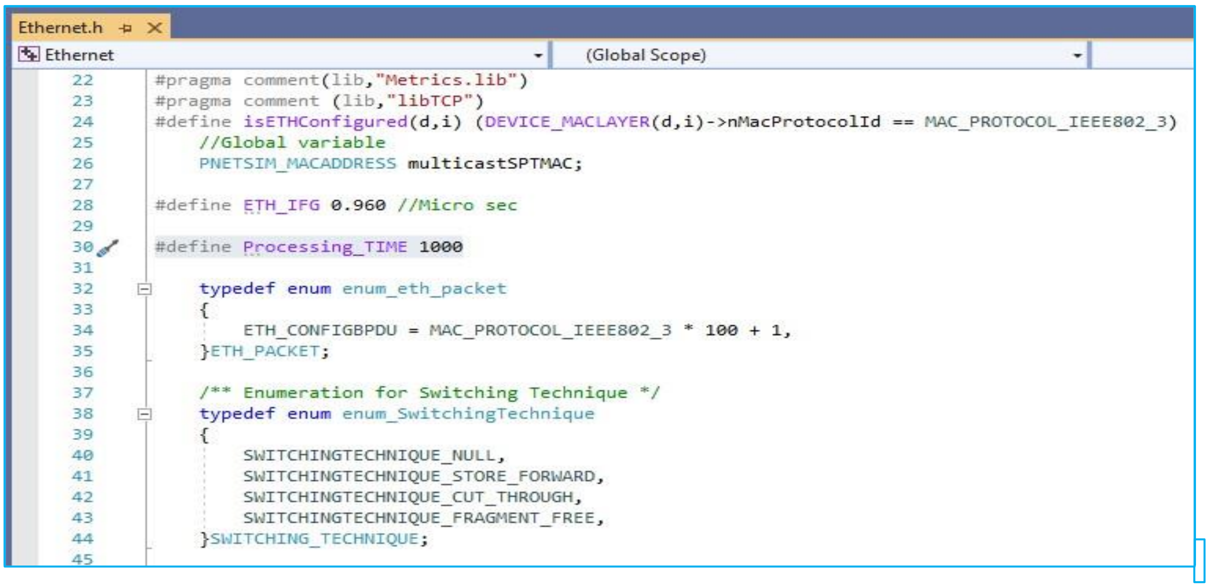


```c
}

static void send_syn_packet(PNETSIM_SOCKET s)
{
    NetSim_PACKET* syn = create_syn(s, pstruEventDetails->dEventTime);

    s->tcb->SND.UNA = s->tcb->ISS;
    s->tcb->SND.NXT = s->tcb->ISS + 1;
    tcp_change_state(s, TCPCONNECTION_SYN_SENT);

    s->tcb->synRetries++;

    s->tcpMetrics->synSent++;

    send_to_network(syn, s);
    add_timeout_event(s, syn);
}

```



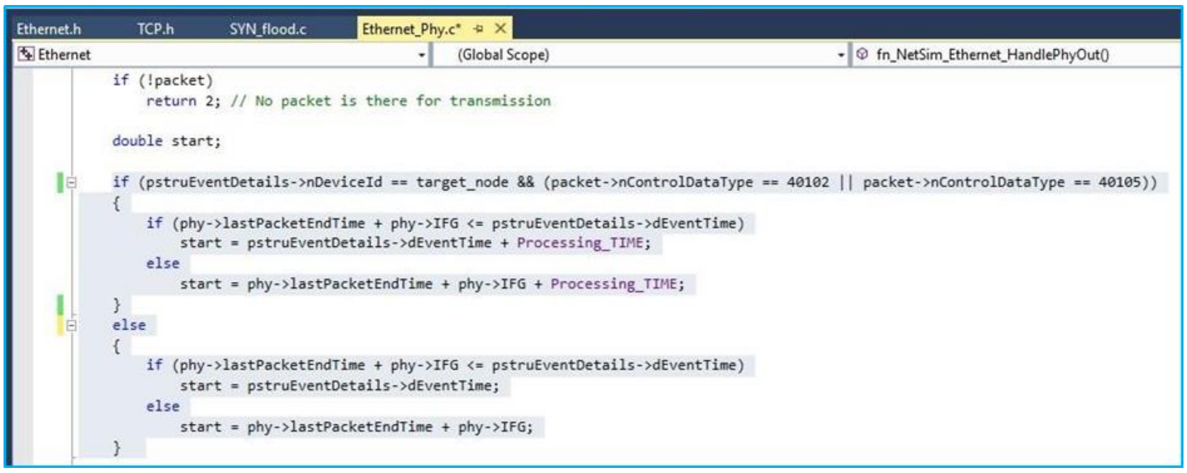
```c
int socket_creation()
{
    static int s_id = 100;
    ptrSOCKETINTERFACE sId = (ptrSOCKETINTERFACE)pstruEventDetails->szOtherDetails;
    PNETSIM_SOCKET newSocket = tcp_create_socket();

    add_to_socket_list(pstruEventDetails->nDeviceId, newSocket);

    PSOCKETADDRESS localsocketAddr = (PSOCKETADDRESS)calloc(1, sizeof * localsocketAddr);
    localsocketAddr->ip = DEVICE_NWADDRESS(pstruEventDetails->nDeviceId, 1);
    localsocketAddr->port = 0;

    PSOCKETADDRESS remotesocketAddr = (PSOCKETADDRESS)calloc(1, sizeof * remotesocketAddr);
    remotesocketAddr->ip = DEVICE_NWADDRESS(target_node, 1);
    remotesocketAddr->port = 0;

    newSocket->SocketId = s_id;
    s_id++;

    newSocket->localAddr = localsocketAddr;
    newSocket->remoteAddr = remotesocketAddr;

    newSocket->localDeviceId = pstruEventDetails->nDeviceId;
    newSocket->remoteDeviceId = target_node;

    newSocket->sId = sId;

    return newSocket;
}
```

9. Added PROCESSING_TIME macro in Ethernet.h file inside ETHERNET project



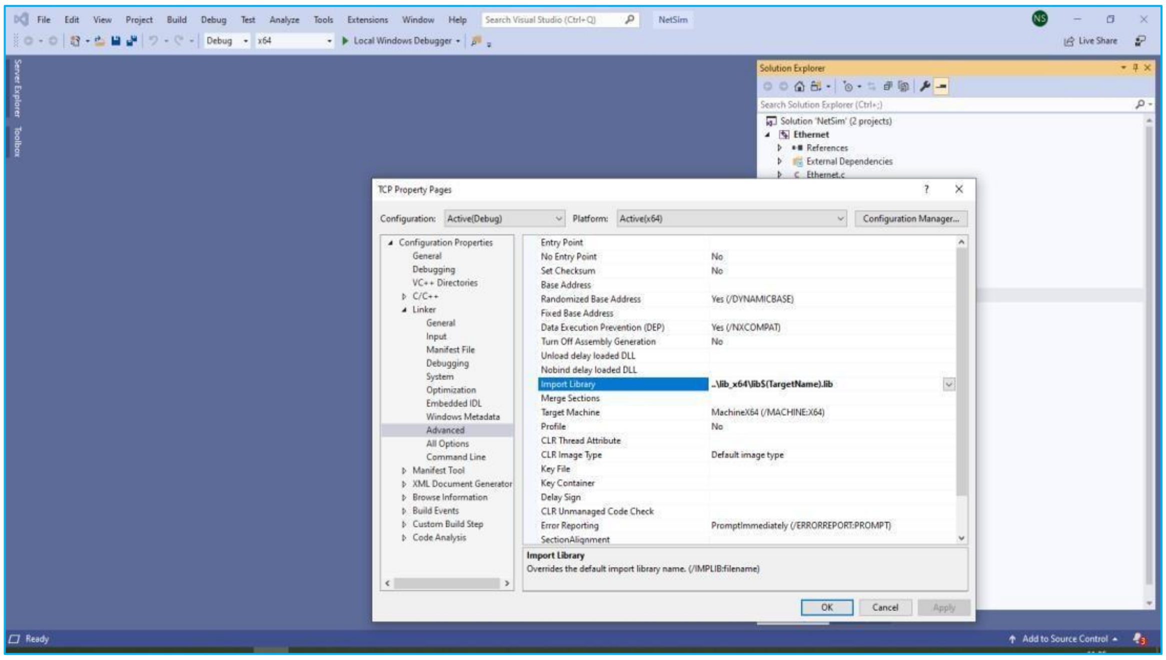
```c
#pragma comment(lib,"Metrics.lib")
#pragma comment (lib,"libTCP")
#define isETHConfigured(d,i) (DEVICE_MACLAYER(d,i)->nMacProtocolId == MAC_PROTOCOL_IEEE802_3)
    //Global variable
    PNETSIM_MACADDRESS multicastSPTMAC;

#define ETH_IFG 0.960 //Micro sec

#define Processing_TIME 1000

    typedef enum enum_eth_packet
    {
        ETH_CONFIGBPDU = MAC_PROTOCOL_IEEE802_3 * 100 + 1,
    }ETH_PACKET;

    /** Enumeration for Switching Technique */
    typedef enum enum_SwitchingTechnique
    {
        SWITCHINGTECHNIQUE_NULL,
        SWITCHINGTECHNIQUE_STORE_FORWARD,
        SWITCHINGTECHNIQUE_CUT_THROUGH,
        SWITCHINGTECHNIQUE_FRAGMENT_FREE,
    }SWITCHING_TECHNIQUE;
```

10. Modified the following lines of code in fn_NetSim_Ethernet_HandlePhyOut() function present in Ethernet_Phy.c file inside Ethernet project.



```c
if (!packet)
    return 2; // No packet is there for transmission

double start;

if (pstruEventDetails->nDeviceId == target_node && (packet->nControlDataType == 40102 || packet->nControlDataType == 40105))
{
    if (phy->lastPacketEndTime + phy->IFG <= pstruEventDetails->dEventTime)
        start = pstruEventDetails->dEventTime + Processing_TIME;
    else
        start = phy->lastPacketEndTime + phy->IFG + Processing_TIME;
}
else
{
    if (phy->lastPacketEndTime + phy->IFG <= pstruEventDetails->dEventTime)
        start = pstruEventDetails->dEventTime;
    else
        start = phy->lastPacketEndTime + phy->IFG;
}
```

11. Right click on TCP project ❼ Properties❼Linker ❼ Advanced ❼import library 32-bit and 64-bit **..\lib\lib$(TargetName).lib** or **..\lib_x64\lib$(TargetName).lib**



TCP Property Pages

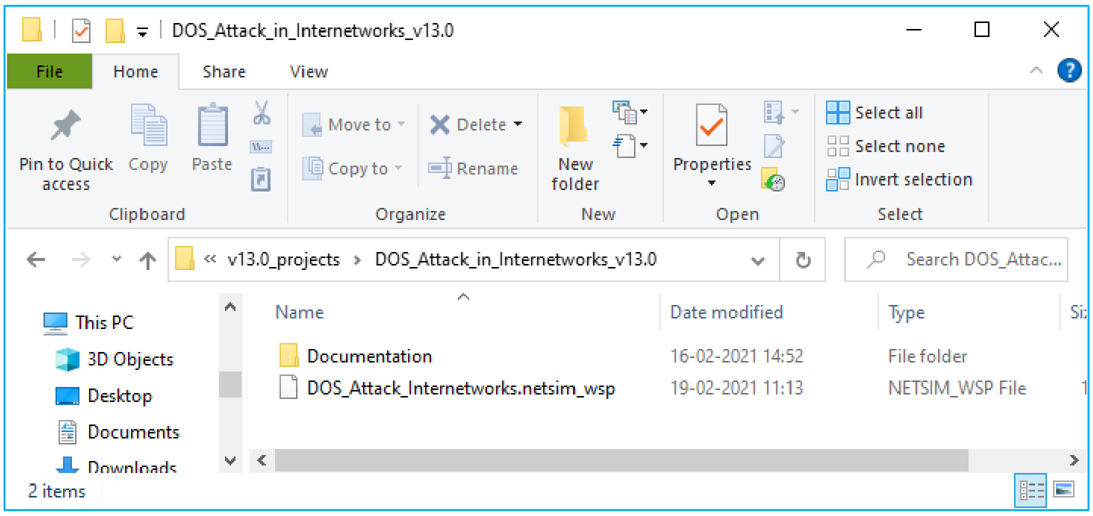
Configuration: Active(Debug) Platform: Active(x64)

Entry Point	
No Entry Point	No
Set Checksum	No
Base Address	
Randomized Base Address	Yes (/DYNAMICBASE)
Fixed Base Address	
Data Execution Prevention (DEP)	Yes (/NXCOMPAT)
Turn Off Assembly Generation	No
Unload delay loaded DLL	
Nobind delay loaded DLL	
Import Library	..\lib_x64\lib$(TargetName).lib
Merge Sections	
Target Machine	MachineX64 (/MACHINE:X64)
Profile	No
CLR Thread Attribute	
CLR Image Type	Default image type
Key File	
Key Container	
Delay Sign	
CLR Unmanaged Code Check	
Error Reporting	PromptImmediately (/ERRORREPORT:PROMPT)
SectionAlignment	

Import Library
Overrides the default import library name. (/IMPLIB:filename)

Steps:

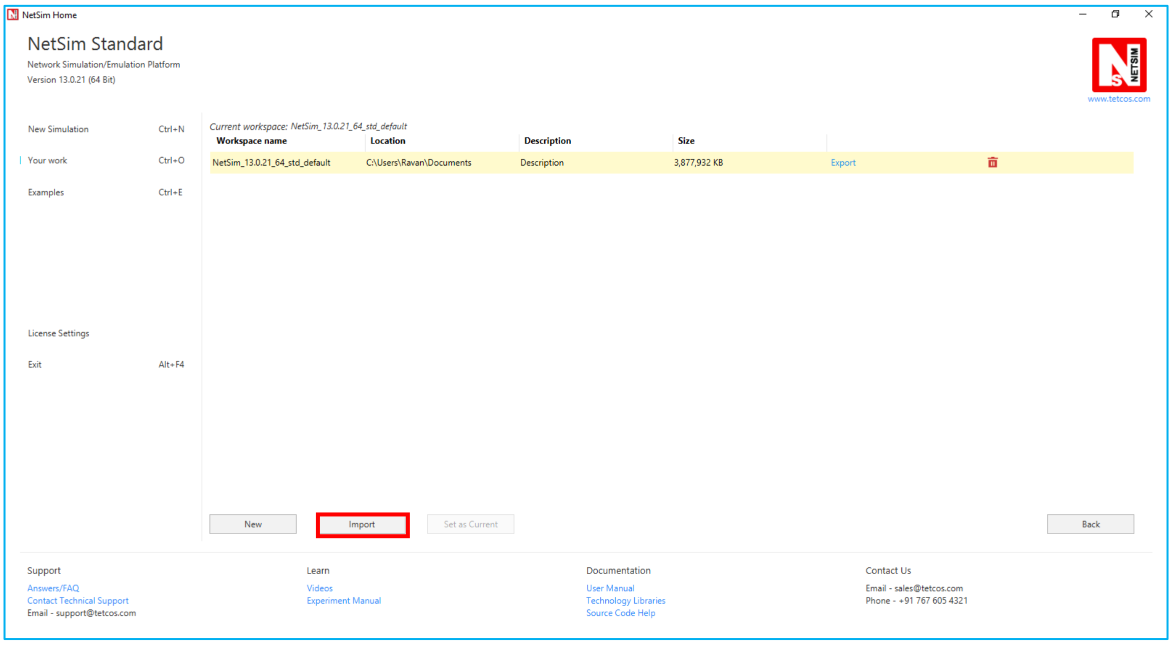
1. The downloaded project folder contains the folders Documentation, and DOS_Attack_Internetworks.netsim_wsp directory as shown below:



DOS_Attack_in_Internetworks_v13.0

Name	Date modified	Type
Documentation	16-02-2021 14:52	File folder
DOS_Attack_Internetworks.netsim_wsp	19-02-2021 11:13	NETSIM_WSP File

2. Import DOS_Attack_Internetworks.netsim_wsp by going to Your work->Workspace Options->More Options in NetSim Home window. Then select Import as shown below:


NetSim Home

NetSim Standard
Network Simulation/Emulation Platform
Version 13.0.21 (64 Bit)

www.tetcos.com

New Simulation	Ctrl+N
Your work	Ctrl+O
Examples	Ctrl+E
License Settings	
Exit	Alt+F4

Current workspace: NetSim_13.0.21_64_std_default

Workspace name	Location	Description	Size		
NetSim_13.0.21_64_std_default	C:\Users\Ravan\Documents	Description	3,877,932 KB	Export	🗑

Support
Answers/FAQ
Contact Technical Support
Email - support@tetcos.com

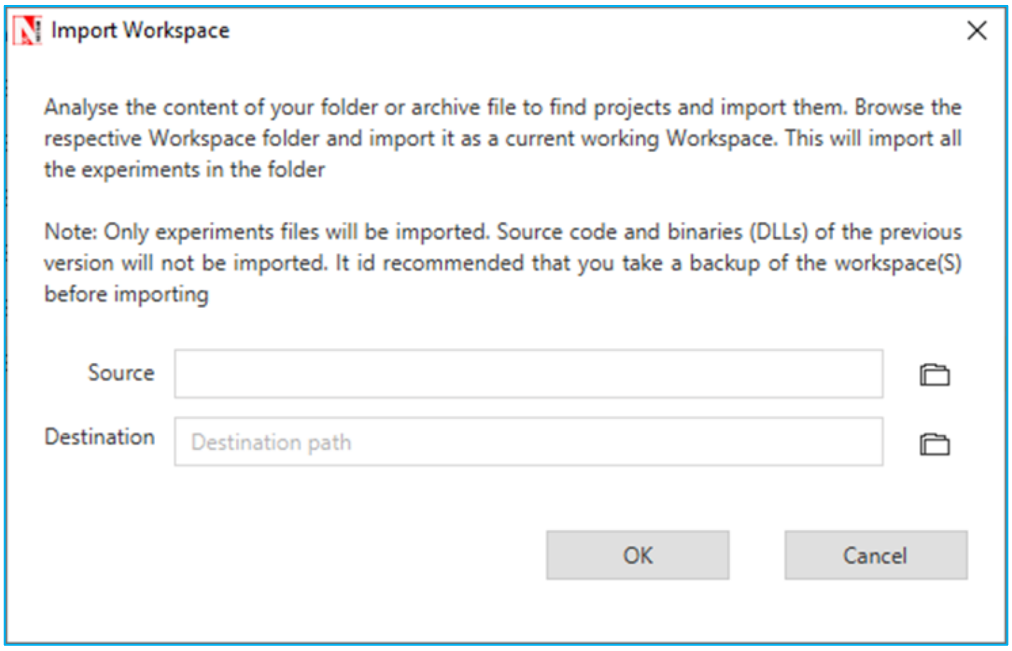
Learn
Videos
Experiment Manual

Documentation
User Manual
Technology Libraries
Source Code Help

Contact Us
Email - sales@tetcos.com
Phone - +91 767 605 4321

3. This will display a window were users need to give the source file (exported workspace file) and the Destination, the path where the workspace is to be imported to and then click on ok.

   Note: Only exported workspaces with “.netsim_wsp” extension can be imported.

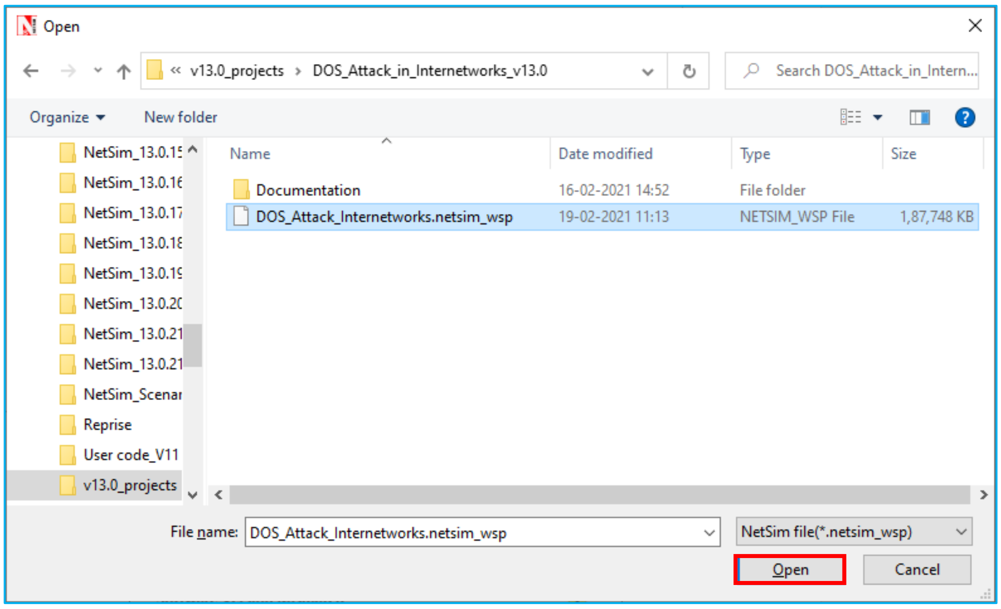

Import Workspace

Analyse the content of your folder or archive file to find projects and import them. Browse the respective Workspace folder and import it as a current working Workspace. This will import all the experiments in the folder

Note: Only experiments files will be imported. Source code and binaries (DLLs) of the previous version will not be imported. It id recommended that you take a backup of the workspace(S) before importing

Source:
Destination: Destination path

4. Browse to the DOS_Attack_Internetworks folder and click on select folder as shown below:


Open

« v13.0_projects › DOS_Attack_in_Internetworks_v13.0

Organize ▾ New folder

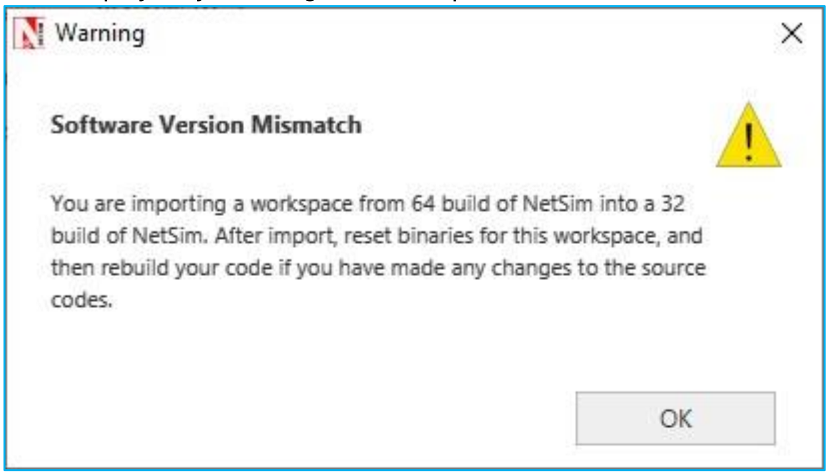
Name	Date modified	Type	Size
Documentation	16-02-2021 14:52	File folder	
DOS_Attack_Internetworks.netsim_wsp	19-02-2021 11:13	NETSIM_WSP File	1,87,748 KB

File name: DOS_Attack_Internetworks.netsim_wsp

NetSim file(*.netsim_wsp)

5. After this click on OK button in the Import Workspace window.

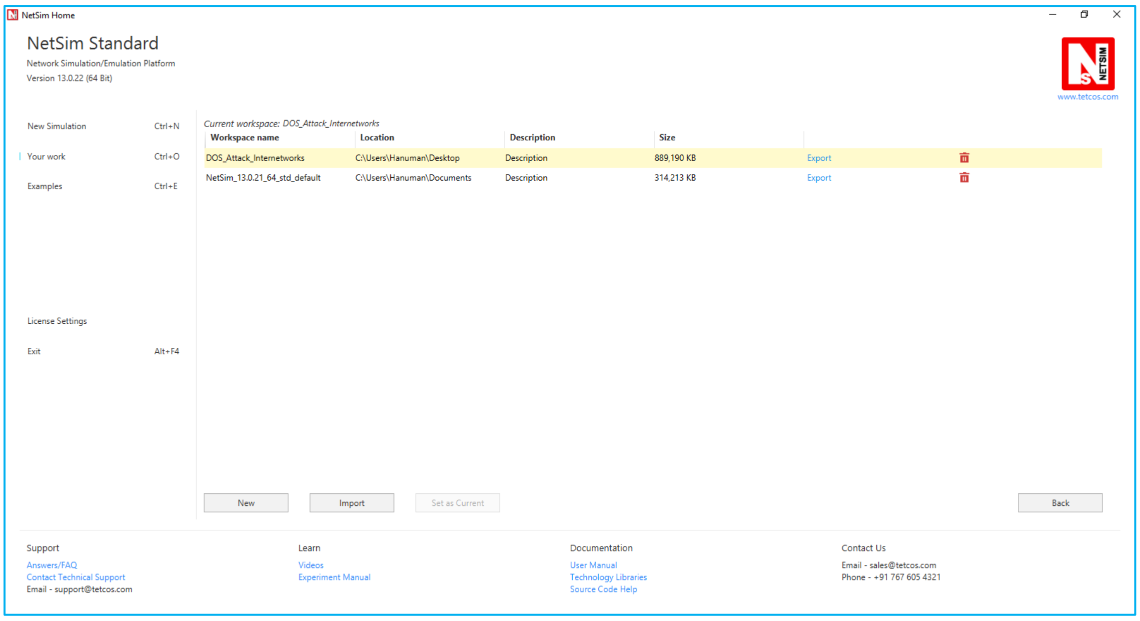
6. While importing the workspace, if the following warning message indicating Software Version Mismatch is displayed, you can ignore it and proceed.


Warning

Software Version Mismatch

You are importing a workspace from 64 build of NetSim into a 32 build of NetSim. After import, reset binaries for this workspace, and then rebuild your code if you have made any changes to the source codes.

7. The Imported workspace will be set as the current workspace automatically. To see the imported workspace, click on Your work->Workspace Options->More Options as shown below:


NetSim Home

NetSim Standard
Network Simulation/Emulation Platform
Version 13.0.22 (64 Bit)

Current workspace: DOS_Attack_Internetworks

Workspace name	Location	Description	Size		
DOS_Attack_Internetworks	C:\Users\Hanuman\Desktop	Description	889,190 KB	Export	🗑
NetSim_13.0.21_64_std_default	C:\Users\Hanuman\Documents	Description	314,213 KB	Export	🗑

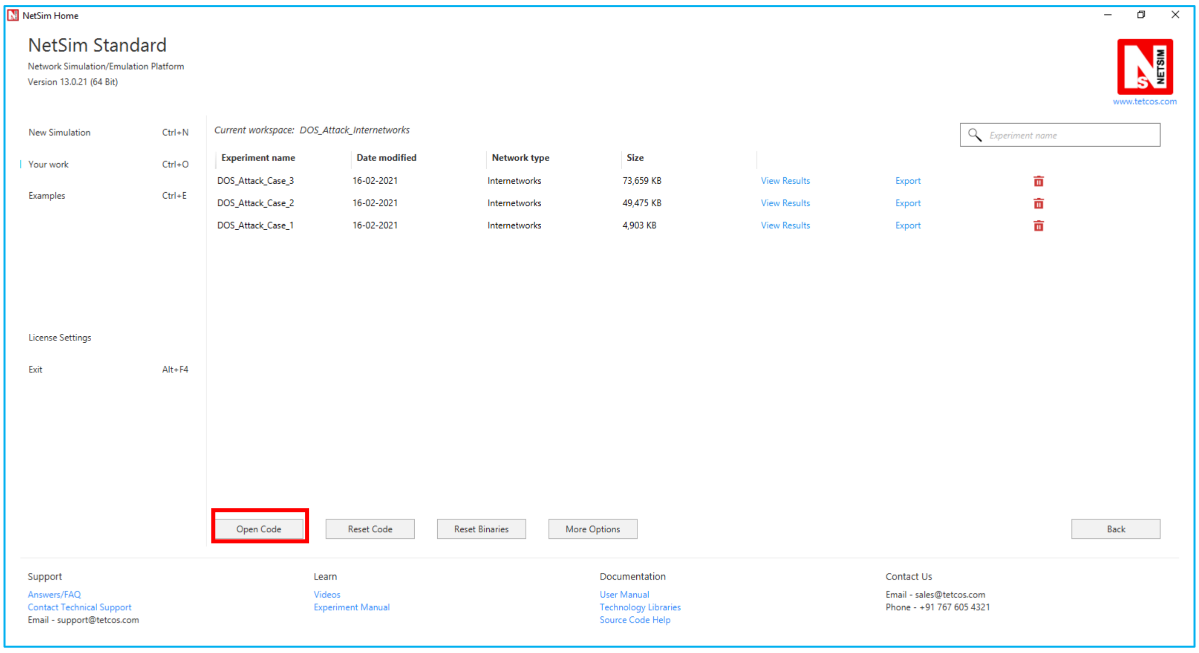
Support
Answers/FAQ
Contact Technical Support
Email - support@tetcos.com

Learn
Videos
Experiment Manual

Documentation
User Manual
Technology Libraries
Source Code Help

Contact Us
Email - sales@tetcos.com
Phone - +91 767 605 4321

8. Open the Source codes in Visual Studio by going to Your work-> Workspace Options and Clicking on Open code button as shown below:


NetSim Home

NetSim Standard
Network Simulation/Emulation Platform
Version 13.0.21 (64 Bit)

New Simulation	Ctrl+N
Your work	Ctrl+O
Examples	Ctrl+E
License Settings	
Exit	Alt+F4

Current workspace: DOS_Attack_Internetworks

Experiment name	Date modified	Network type	Size			
DOS_Attack_Case_3	16-02-2021	Internetworks	73,659 KB	View Results	Export	🗑
DOS_Attack_Case_2	16-02-2021	Internetworks	49,475 KB	View Results	Export	🗑
DOS_Attack_Case_1	16-02-2021	Internetworks	4,903 KB	View Results	Export	🗑

Support
Answers/FAQ
Contact Technical Support
Email - support@tetcos.com

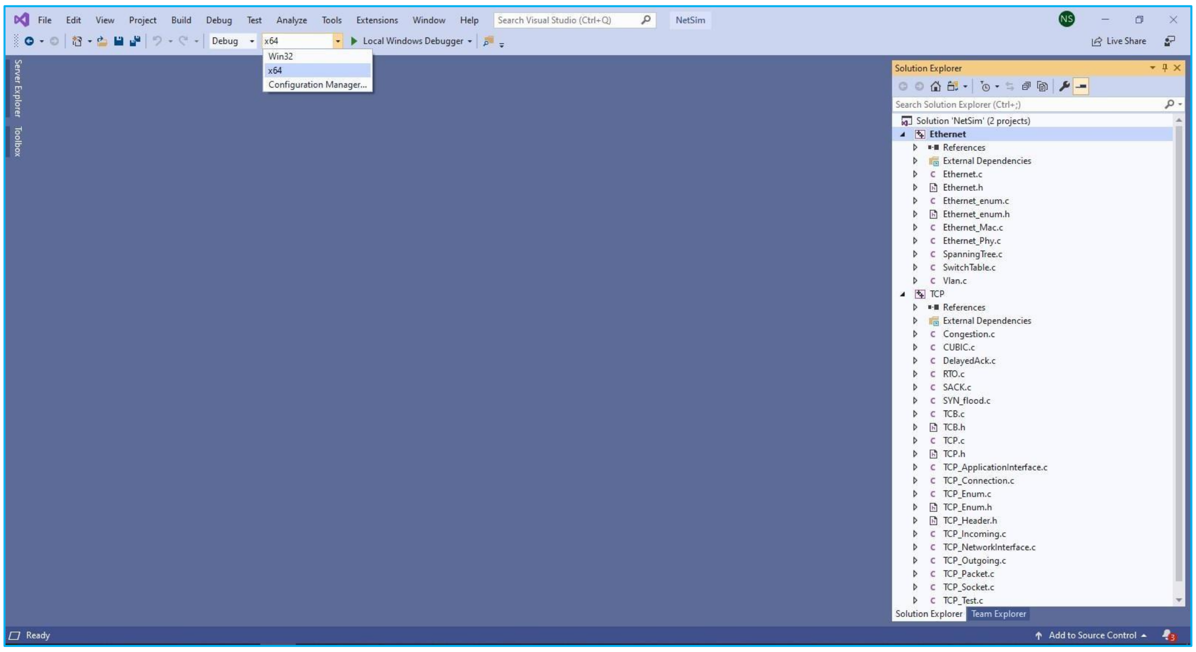
Learn
Videos
Experiment Manual

Documentation
User Manual
Technology Libraries
Source Code Help

Contact Us
Email - sales@tetcos.com
Phone - +91 767 605 4321

9. Under the **TCP** project in the solution explorer you will be able to see that **SYN_FLOOD.c** file.

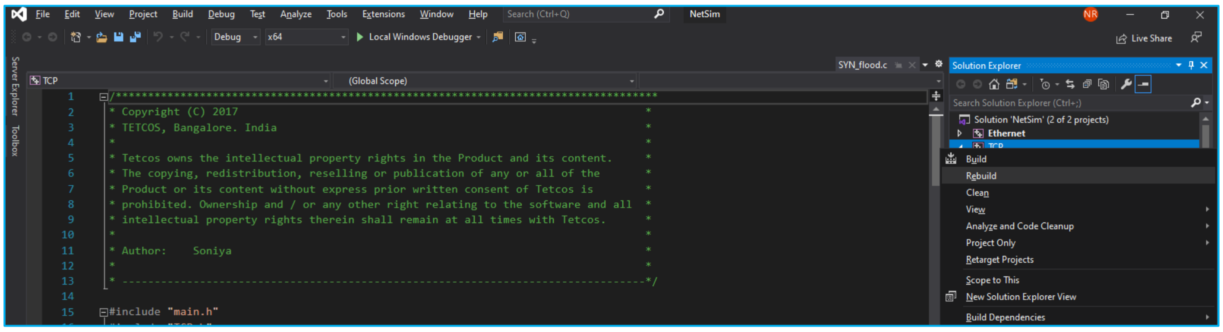
10. Based on whether you are using NetSim 32 bit or 64 bit setup you can configure Visual studio to build 32 bit or 64 bit Dll files respectively as shown below:



Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'NetSim' (2 projects)
- Ethernet
  - References
  - External Dependencies
  - Ethernet.c
  - Ethernet.h
  - Ethernet_enum.c
  - Ethernet_enum.h
  - Ethernet_Mac.c
  - Ethernet_Phy.c
  - SpanningTree.c
  - SwitchTable.c
  - Vlan.c
- TCP
  - References
  - External Dependencies
  - Congestion.c
  - CUBIC.c
  - DelayedAck.c
  - RTO.c
  - SACK.c
  - SYN_flood.c
  - TCB.c
  - TCB.h
  - TCP.c
  - TCP.h
  - TCP_ApplicationInterface.c
  - TCP_Connection.c
  - TCP_Enum.c
  - TCP_Enum.h
  - TCP_Header.h
  - TCP_Incoming.c
  - TCP_NetworkInterface.c
  - TCP_Outgoing.c
  - TCP_Packet.c
  - TCP_Socket.c
  - TCP_Test.c

Solution Explorer | Team Explorer

11. Right click on the solution in the solution explorer and select Rebuild. (Note: first rebuild the TCP project and then rebuild the Ethernet project)

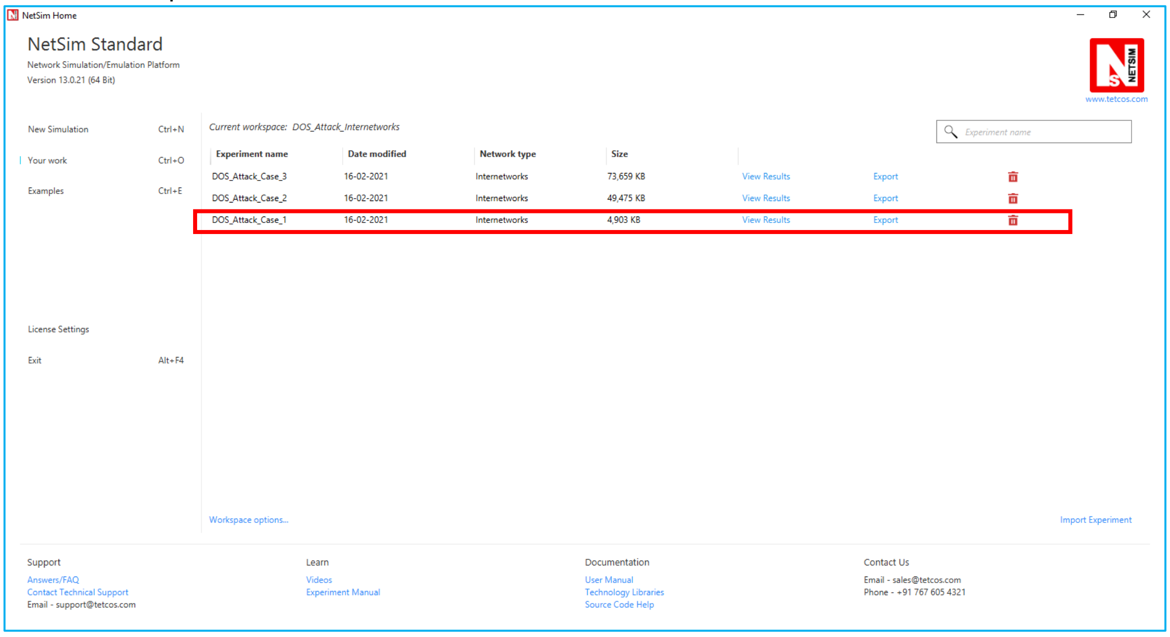


```
/*****************************************************************************
* Copyright (C) 2017                                                         *
* TETCOS, Bangalore. India                                                   *
*                                                                            *
* Tetcos owns the intellectual property rights in the Product and its content.  *
* The copying, redistribution, reselling or publication of any or all of the *
* Product or its content without express prior written consent of Tetcos is  *
* prohibited. Ownership and / or any other right relating to the software and all *
* intellectual property rights therein shall remain at all times with Tetcos.  *
*                                                                            *
* Author:    Soniya                                                          *
*                                                                            *
* ---------------------------------------------------------------------------*/

#include "main.h"
```

12. Upon successful build modified libTCP.dll and libEthernet.dll file gets automatically updated in the directory containing NetSim binaries.

Case-1: Without Malicious Node

1. Then DOS_Attack_Internetworks comes with a sample configuration that is already saved. To open this example, go to Your work and click on the DOS_Attack_Case_1 that is present under the list of experiments as shown below:



Current workspace: DOS_Attack_Internetworks

Experiment name	Date modified	Network type	Size		
DOS_Attack_Case_3	16-02-2021	Internetworks	73,659 KB	View Results	Export
DOS_Attack_Case_2	16-02-2021	Internetworks	49,475 KB	View Results	Export
DOS_Attack_Case_1	16-02-2021	Internetworks	4,903 KB	View Results	Export

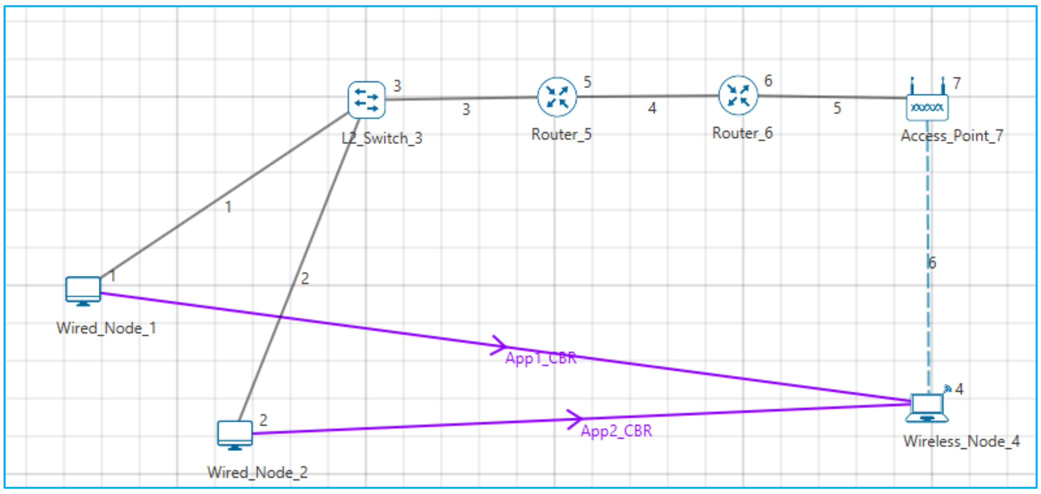
Support
Answers/FAQ
Contact Technical Support
Email - support@tetcos.com

Learn
Videos
Experiment Manual

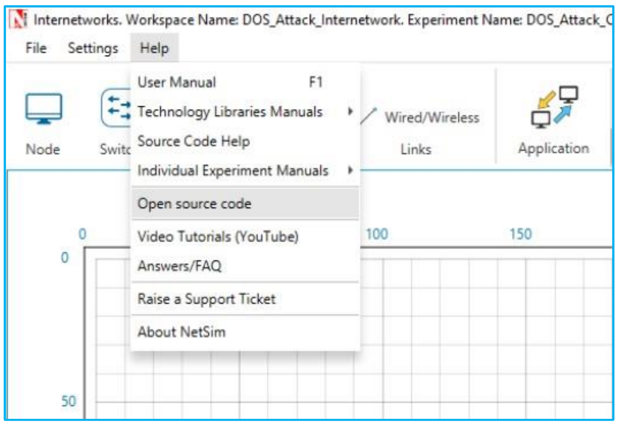
Documentation
User Manual
Technology Libraries
Source Code Help

Contact Us
Email - sales@tetcos.com
Phone - +91 767 605 4321

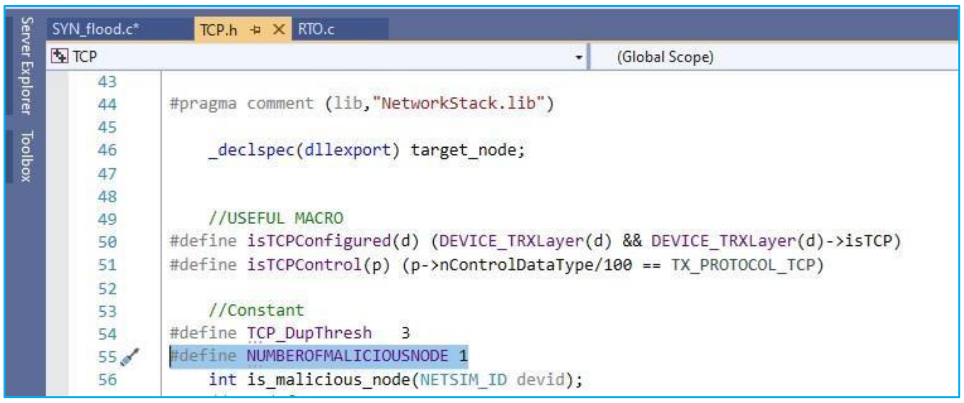
2. The saved network scenario consisting of 2 Wired Nodes, 1 L2 Switch, 2 router, 1 Access Point and 1 wireless node in the grid environment forming a internetworks Network. Traffic is configured from Wired node to the Wireless node.



3. Help ❼ Open Source code

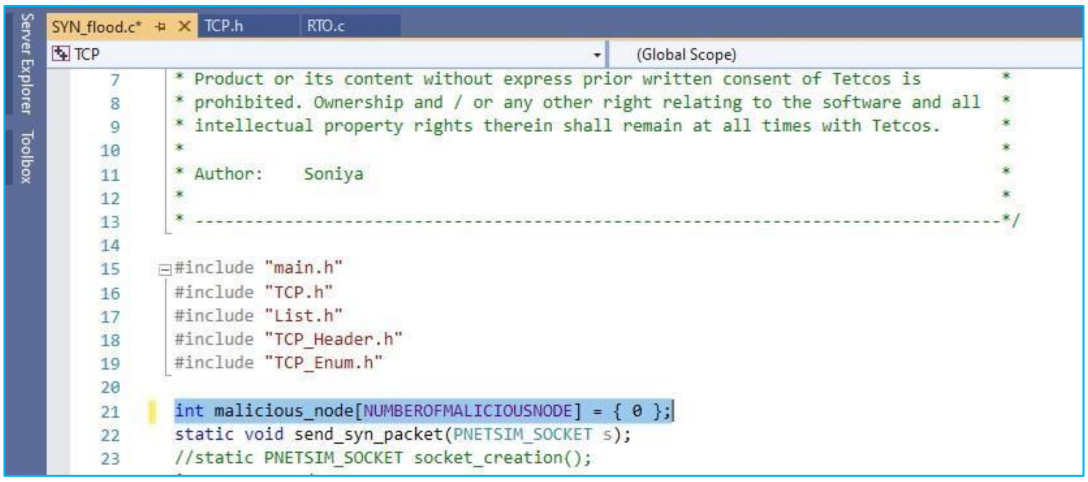

Internetworks. Workspace Name: DOS_Attack_Internetwork. Experiment Name: DOS_Attack_C
File
Settings
Help
User Manual F1
Technology Libraries Manuals
Source Code Help
Individual Experiment Manuals
Open source code
Video Tutorials (YouTube)
Answers/FAQ
Raise a Support Ticket
About NetSim
Node
Switc
Wired/Wireless Links
Application

4. In TCP.h set **NUMBEROFMALICIOUSNODE** as 1.



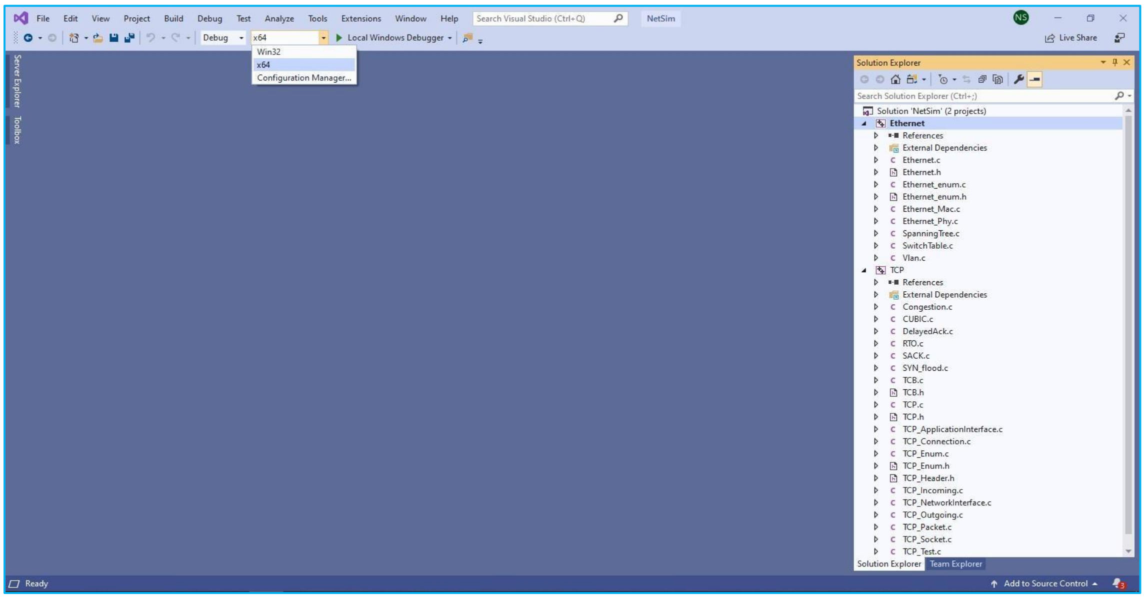
```
#pragma comment (lib,"NetworkStack.lib")

    _declspec(dllexport) target_node;


    //USEFUL MACRO
#define isTCPConfigured(d) (DEVICE_TRXLayer(d) && DEVICE_TRXLayer(d)->isTCP)
#define isTCPControl(p) (p->nControlDataType/100 == TX_PROTOCOL_TCP)

    //Constant
#define TCP_DupThresh   3
#define NUMBEROFMALICIOUSNODE 1
    int is_malicious_node(NETSIM_ID devid);
```

5. In SYN_FLOOD.c set **malicious node** as 0**.**

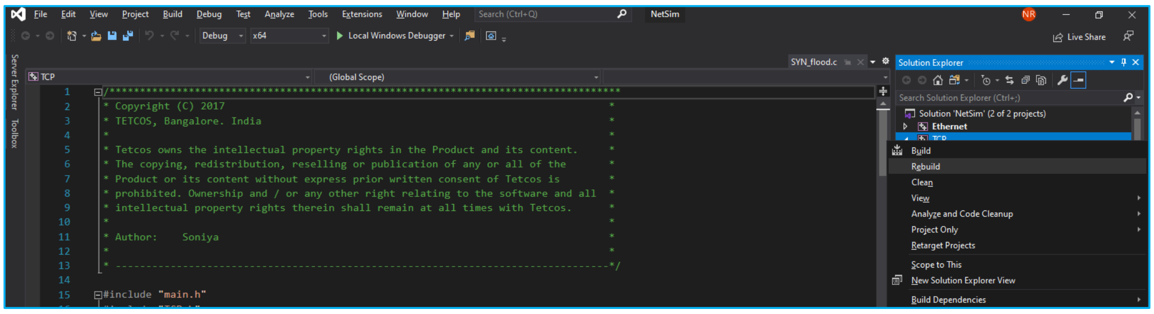


```
 * Product or its content without express prior written consent of Tetcos is   *
 * prohibited. Ownership and / or any other right relating to the software and all  *
 * intellectual property rights therein shall remain at all times with Tetcos.     *
 *                                                                                 *
 * Author:    Soniya                                                               *
 *                                                                                 *
 * ---------------------------------------------------------------------------------*/

#include "main.h"
#include "TCP.h"
#include "List.h"
#include "TCP_Header.h"
#include "TCP_Enum.h"

int malicious_node[NUMBEROFMALICIOUSNODE] = { 0 };
static void send_syn_packet(PNETSIM_SOCKET s);
//static PNETSIM_SOCKET socket_creation();
```

6. Based on whether you are using NetSim 32 bit or 64 bit setup you can configure Visual studio to build 32 bit or 64 bit Dll files respectively as shown below:



7. Right click on the solution in the solution explorer and select Rebuild. (Note: first rebuild the TCP project and then rebuild the Ethernet project)

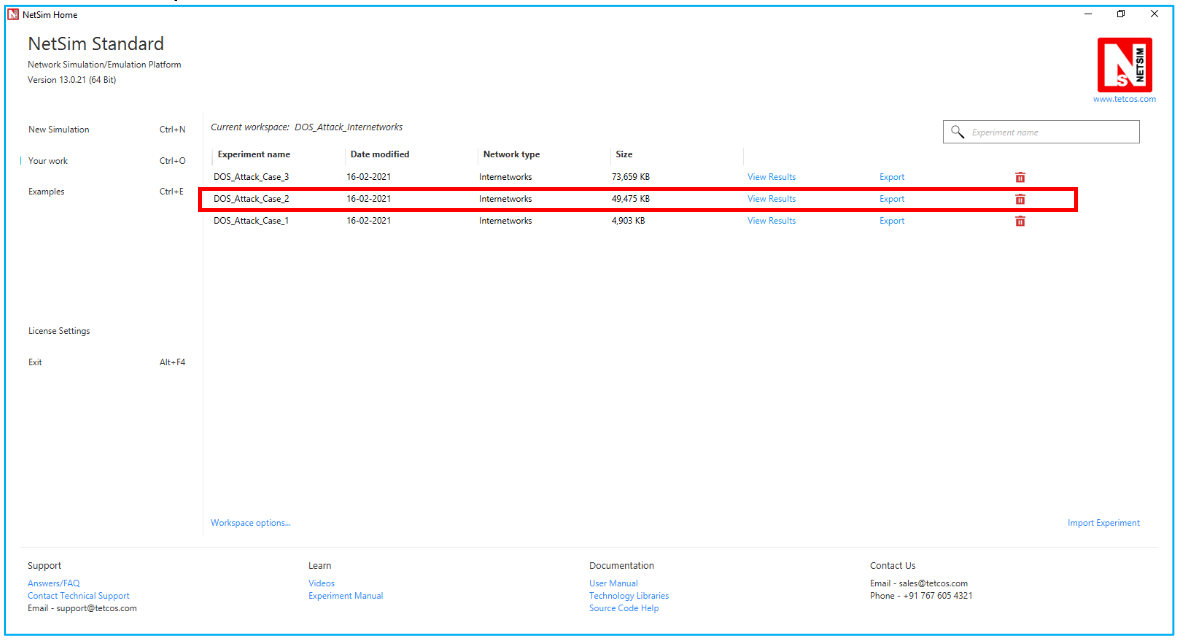


```
/********************************************************************************
* Copyright (C) 2017                                                            *
* TETCOS, Bangalore. India                                                      *
*                                                                               *
* Tetcos owns the intellectual property rights in the Product and its content.  *
* The copying, redistribution, reselling or publication of any or all of the    *
* Product or its content without express prior written consent of Tetcos is     *
* prohibited. Ownership and / or any other right relating to the software and all *
* intellectual property rights therein shall remain at all times with Tetcos.   *
*                                                                               *
* Author:    Soniya                                                             *
*                                                                               *
* ---------------------------------------------------------------------------------*/
#include "main.h"
```

8. Upon successful build modified libTCP.dll and libEthernet.dll file gets automatically updated in the directory containing NetSim binaries.
9. Run the simulation for 10 seconds.

Case-2: With one Malicious Node

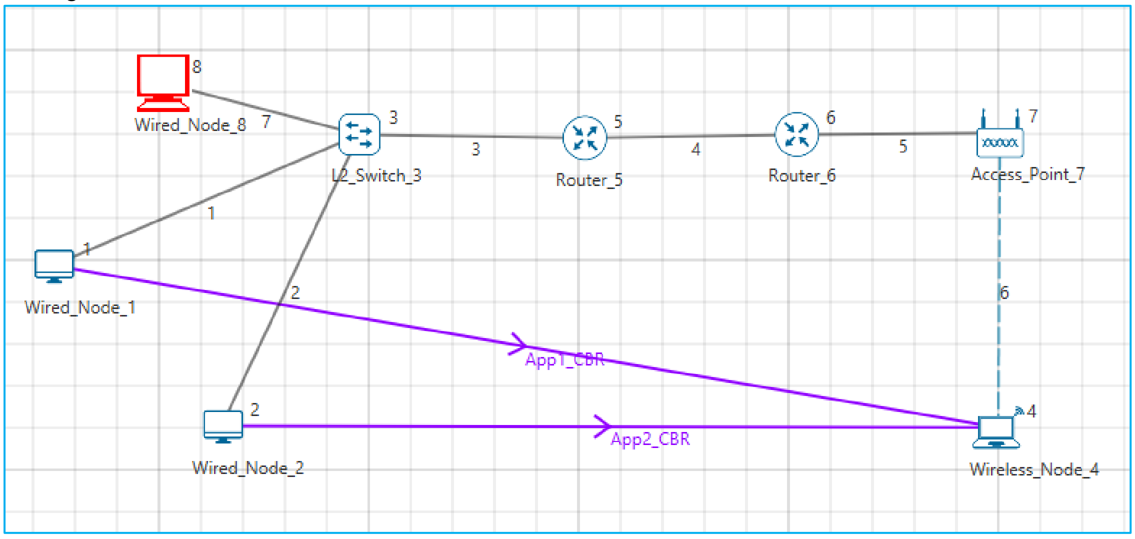
1. Then DOS_Attack_Internetworks comes with a sample configuration that is already saved. To open this example, go to Your work and click on the DOS_Attack_Case_2 that is present under the list of experiments as shown below:



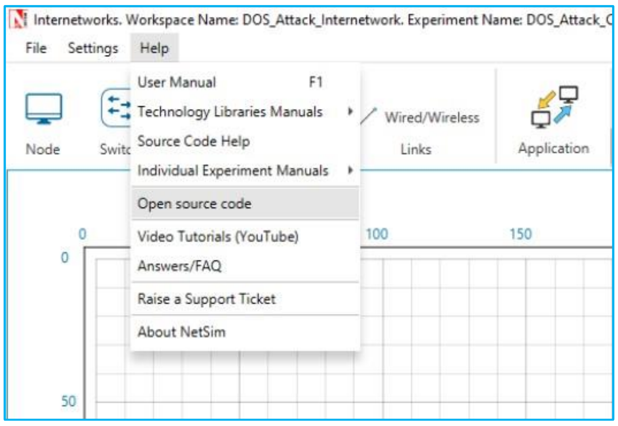
Current workspace: DOS_Attack_Internetworks

Experiment name	Date modified	Network type	Size		
DOS_Attack_Case_3	16-02-2021	Internetworks	73,659 KB	View Results	Export
DOS_Attack_Case_2	16-02-2021	Internetworks	49,475 KB	View Results	Export
DOS_Attack_Case_1	16-02-2021	Internetworks	4,903 KB	View Results	Export

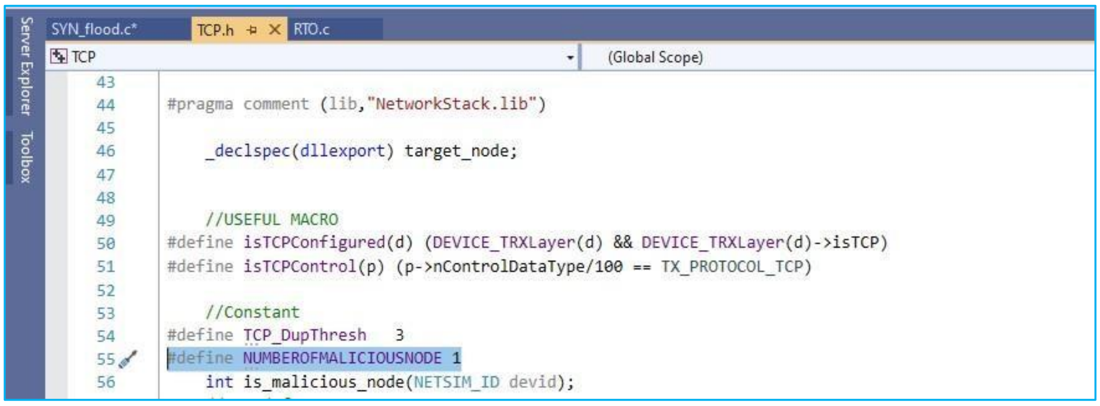
2. The saved network scenario consisting of 3 Wired Nodes, 1 L2 Switch, 2 router, 1 Access Point and 1 wireless node in the grid environment forming a internetworks Network. Traffic is configured from Wired node to the Wireless node.



3. Help ❼ Open Source code

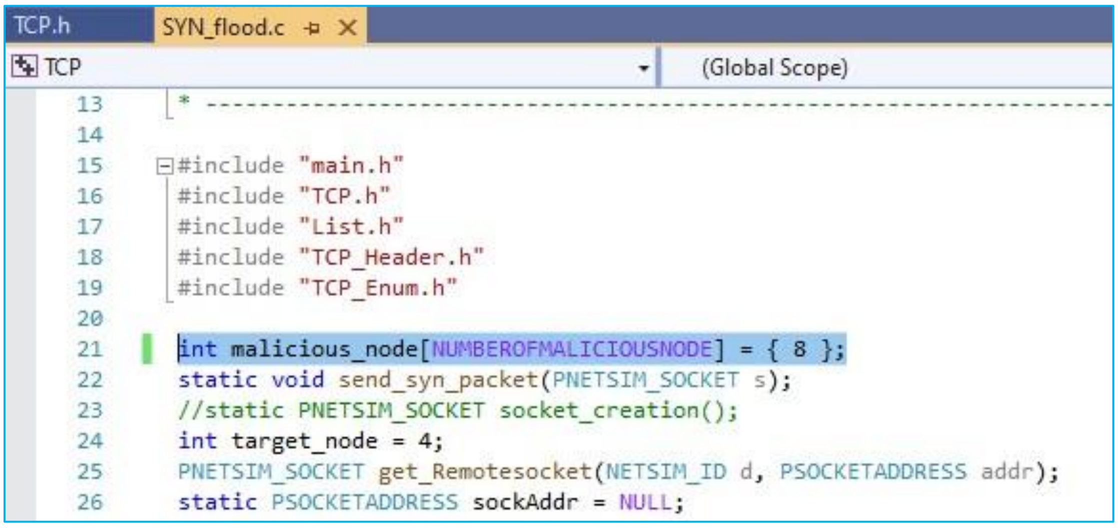

Internetworks. Workspace Name: DOS_Attack_Internetwork. Experiment Name: DOS_Attack_C
File
Settings
Help
User Manual F1
Technology Libraries Manuals
Source Code Help
Individual Experiment Manuals
Open source code
Video Tutorials (YouTube)
Answers/FAQ
Raise a Support Ticket
About NetSim
Node
Switc
Wired/Wireless
Links
Application

4. In TCP.h set **NUMBEROFMALICIOUSNODE** as 1.



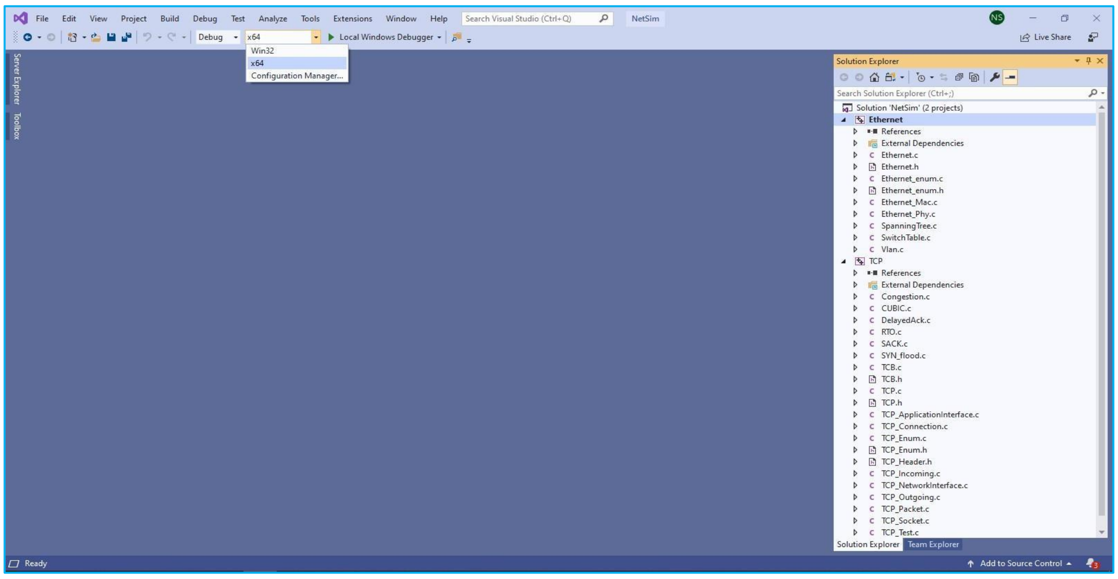
```

#pragma comment (lib,"NetworkStack.lib")

    _declspec(dllexport) target_node;


    //USEFUL MACRO
#define isTCPConfigured(d) (DEVICE_TRXLayer(d) && DEVICE_TRXLayer(d)->isTCP)
#define isTCPControl(p) (p->nControlDataType/100 == TX_PROTOCOL_TCP)

    //Constant
#define TCP_DupThresh    3
#define NUMBEROFMALICIOUSNODE 1
    int is_malicious_node(NETSIM_ID devid);
```

5. In SYN_FLOOD.c set **malicious node** as 8**.**

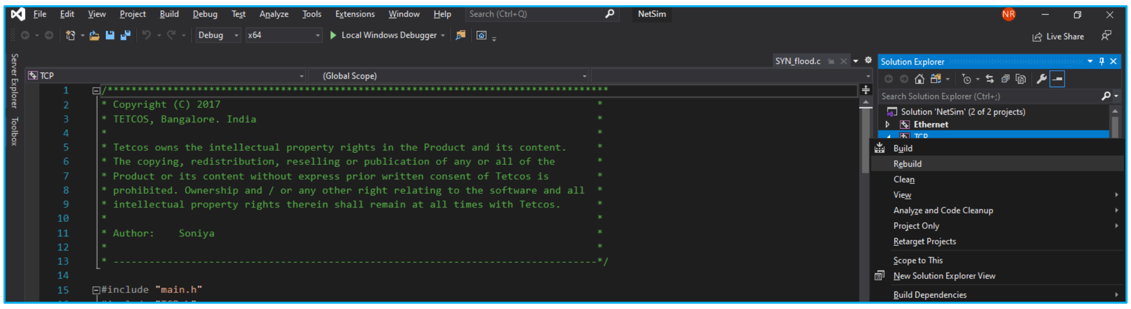


```c
 * ---------------------------------------------------------------------------

#include "main.h"
#include "TCP.h"
#include "List.h"
#include "TCP_Header.h"
#include "TCP_Enum.h"

int malicious_node[NUMBEROFMALICIOUSNODE] = { 8 };
static void send_syn_packet(PNETSIM_SOCKET s);
//static PNETSIM_SOCKET socket_creation();
int target_node = 4;
PNETSIM_SOCKET get_Remotesocket(NETSIM_ID d, PSOCKETADDRESS addr);
static PSOCKETADDRESS sockAddr = NULL;
```

6. Based on whether you are using NetSim 32 bit or 64 bit setup you can configure Visual studio to build 32 bit or 64 bit Dll files respectively as shown below:



7. Right click on the solution in the solution explorer and select Rebuild. (Note: first rebuild the TCP project and then rebuild the Ethernet project)



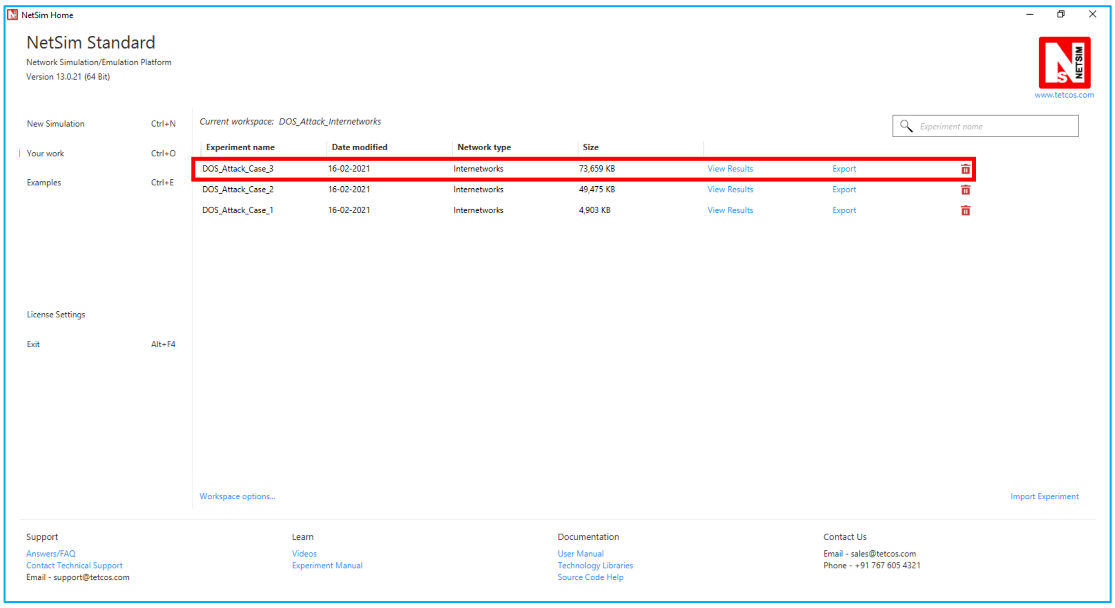
```c
/***************************************************************************
* Copyright (C) 2017                                                        *
* TETCOS, Bangalore. India                                                  *
*                                                                           *
* Tetcos owns the intellectual property rights in the Product and its content.    *
* The copying, redistribution, reselling or publication of any or all of the      *
* Product or its content without express prior written consent of Tetcos is       *
* prohibited. Ownership and / or any other right relating to the software and all *
* intellectual property rights therein shall remain at all times with Tetcos.     *
*                                                                           *
* Author:    Soniya                                                         *
*                                                                           *
* ---------------------------------------------------------------------------*/

#include "main.h"
```

8. Upon successful build modified libTCP.dll and libEthernet.dll file gets automatically updated in the directory containing NetSim binaries.

9. Run the simulation for 10 seconds.

Case-3: With two Malicious Node

1. Then DOS_Attack_Internetworks comes with a sample configuration that is already saved. To open this example, go to your work and click on the DOS_Attack_Case_3 that is present under the list of experiments as shown below:



NetSim Home

NetSim Standard
Network Simulation/Emulation Platform
Version 13.0.21 (64 Bit)

www.tetcos.com

New Simulation	Ctrl+N
Your work	Ctrl+O
Examples	Ctrl+E
License Settings	
Exit	Alt+F4

Current workspace: DOS_Attack_Internetworks

Experiment name	Date modified	Network type	Size		
DOS_Attack_Case_3	16-02-2021	Internetworks	73,659 KB	View Results	Export
DOS_Attack_Case_2	16-02-2021	Internetworks	49,475 KB	View Results	Export
DOS_Attack_Case_1	16-02-2021	Internetworks	4,903 KB	View Results	Export

Workspace options... Import Experiment

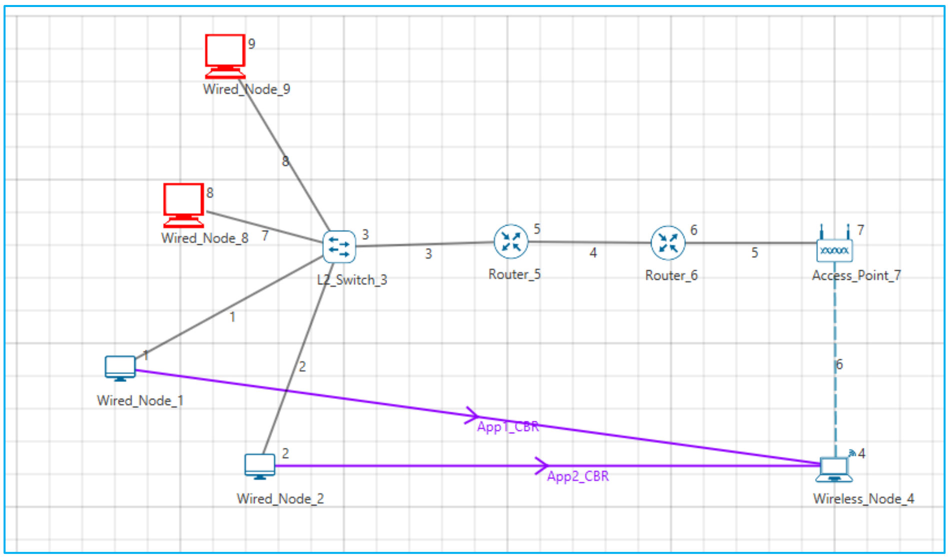
Support
Answers/FAQ
Contact Technical Support
Email - support@tetcos.com

Learn
Videos
Experiment Manual

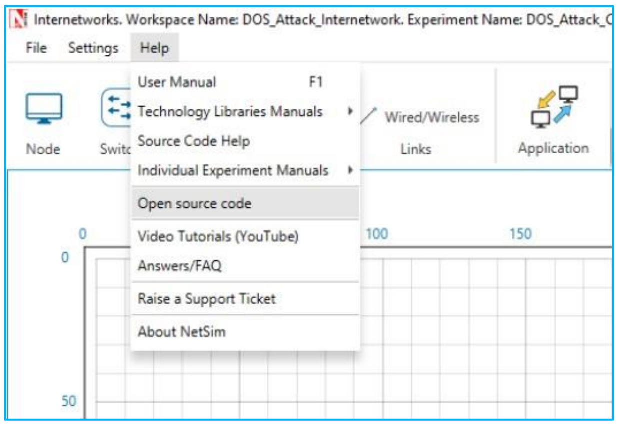
Documentation
User Manual
Technology Libraries
Source Code Help

Contact Us
Email - sales@tetcos.com
Phone - +91 767 605 4321

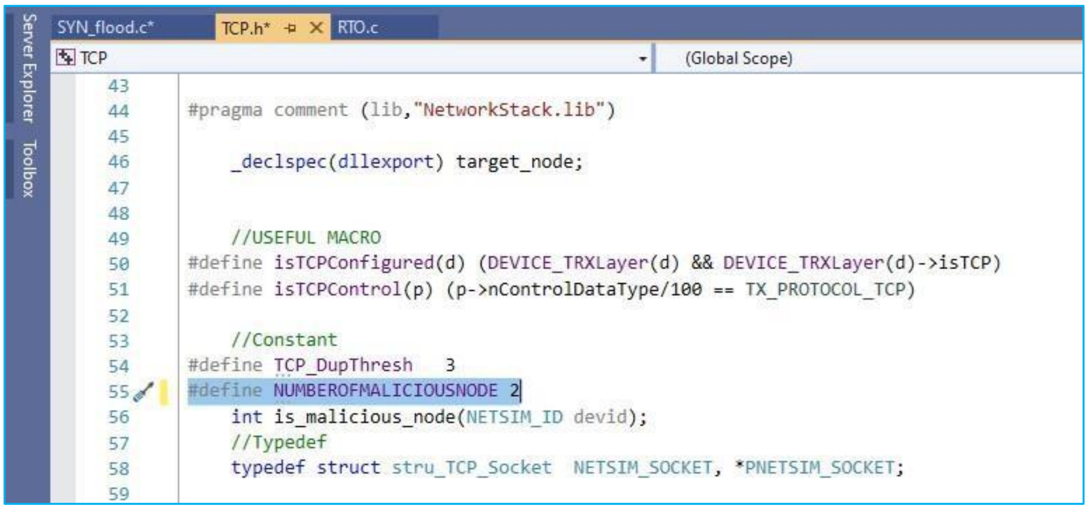
2. The saved network scenario consisting of 4 Wired Nodes, 1 L2 Switch, 2 router, 1 Access Point and 1 wireless node in the grid environment forming a internetworks Network. Traffic is configured from Wired node to the Wireless node.



3. Help ❼ Open Source code

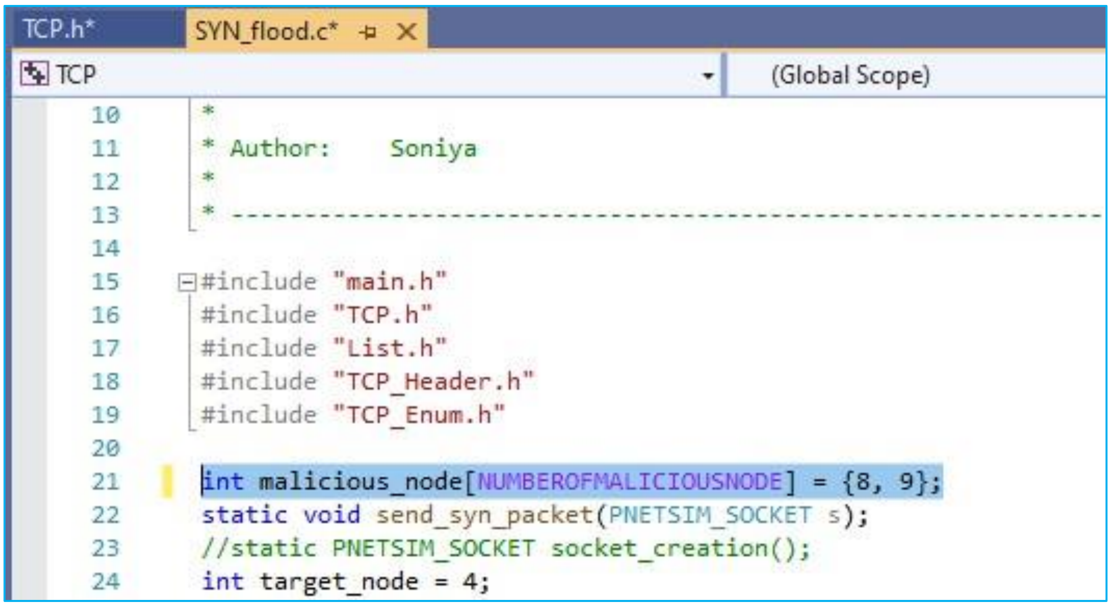

Internetworks. Workspace Name: DOS_Attack_Internetwork. Experiment Name: DOS_Attack_C
File
Settings
Help
User Manual F1
Technology Libraries Manuals
Source Code Help
Individual Experiment Manuals
Open source code
Video Tutorials (YouTube)
Answers/FAQ
Raise a Support Ticket
About NetSim
Node
Wired/Wireless Links
Application

4. In TCP.h set **NUMBEROFMALICIOUSNODE** as 2.



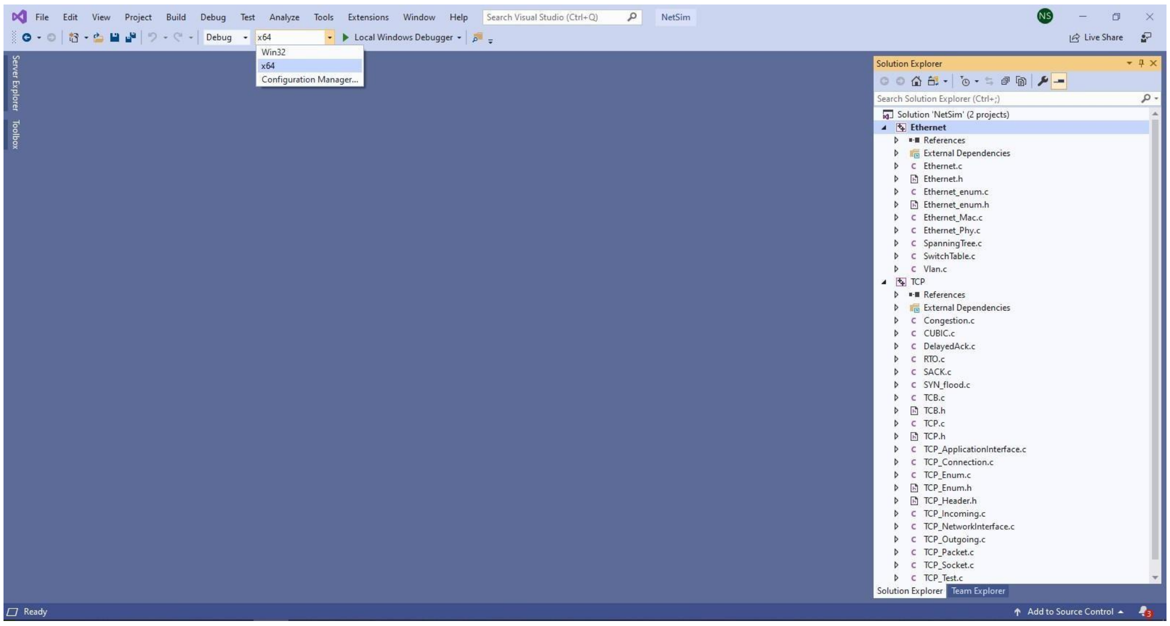
```
#pragma comment (lib,"NetworkStack.lib")

    _declspec(dllexport) target_node;


    //USEFUL MACRO
#define isTCPConfigured(d) (DEVICE_TRXLayer(d) && DEVICE_TRXLayer(d)->isTCP)
#define isTCPControl(p) (p->nControlDataType/100 == TX_PROTOCOL_TCP)

    //Constant
#define TCP_DupThresh    3
#define NUMBEROFMALICIOUSNODE 2
    int is_malicious_node(NETSIM_ID devid);
    //Typedef
    typedef struct stru_TCP_Socket  NETSIM_SOCKET, *PNETSIM_SOCKET;
```

5. In SYN_FLOOD.c set **malicious node** as 8, 9**.**

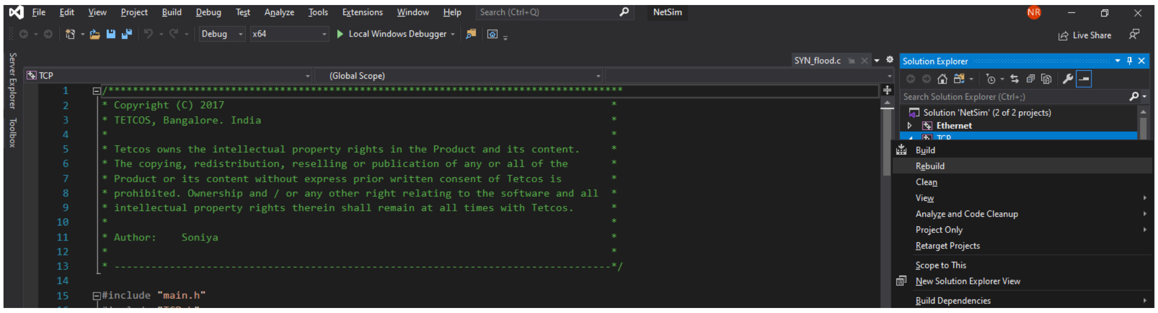


```
 *
 * Author:    Soniya
 *
 * ---------------------------------------------------------------

#include "main.h"
#include "TCP.h"
#include "List.h"
#include "TCP_Header.h"
#include "TCP_Enum.h"

int malicious_node[NUMBEROFMALICIOUSNODE] = {8, 9};
static void send_syn_packet(PNETSIM_SOCKET s);
//static PNETSIM_SOCKET socket_creation();
int target_node = 4;
```

6. Based on whether you are using NetSim 32 bit or 64 bit setup you can configure Visual studio to build 32 bit or 64 bit Dll files respectively as shown below:



7. Right click on the solution in the solution explorer and select Rebuild. (Note: first rebuild the TCP project and then rebuild the Ethernet project)



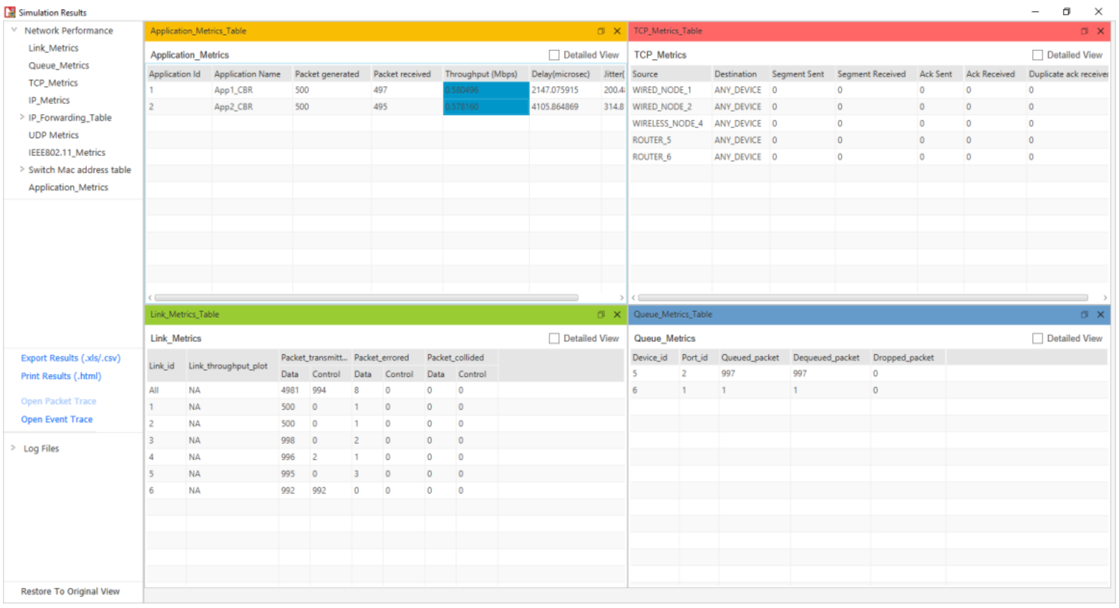
```
/************************************************************************************
* Copyright (C) 2017                                                               *
* TETCOS, Bangalore. India                                                         *
*                                                                                  *
* Tetcos owns the intellectual property rights in the Product and its content.     *
* The copying, redistribution, reselling or publication of any or all of the       *
* Product or its content without express prior written consent of Tetcos is       *
* prohibited. Ownership and / or any other right relating to the software and all  *
* intellectual property rights therein shall remain at all times with Tetcos.      *
*                                                                                  *
* Author:    Soniya                                                                *
*                                                                                  *
* ---------------------------------------------------------------------------------*/

#include "main.h"
```

8. Upon successful build modified libTCP.dll and libEthernet.dll file gets automatically updated in the directory containing NetSim binaries.

9. Run the simulation for 10 seconds.

Result:

After simulation, open metrics window and observe the Application_Throughput is decreasing for both applications as we increase the malicious node because of the SYN flood sent from the malicious node. In case 1 there is no malicious node so there will be no SYN_FLOOD packets.



Application_Metrics_Table

Application_Metrics

Application Id	Application Name	Packet generated	Packet received	Throughput (Mbps)	Delay(microsec)	Jitter(
1	App1_CBR	500	497	0.580496	2147.075915	200.4
2	App2_CBR	500	495	0.578160	4105.864869	314.8

TCP_Metrics_Table

TCP_Metrics

Source	Destination	Segment Sent	Segment Received	Ack Sent	Ack Received	Duplicate ack receive
WIRED_NODE_1	ANY_DEVICE	0	0	0	0	0
WIRED_NODE_2	ANY_DEVICE	0	0	0	0	0
WIRELESS_NODE_4	ANY_DEVICE	0	0	0	0	0
ROUTER_5	ANY_DEVICE	0	0	0	0	0
ROUTER_6	ANY_DEVICE	0	0	0	0	0

Link_Metrics_Table

Link_Metrics

Link_id	Link_throughput_plot	Packet_transmitt...		Packet_errored		Packet_collided	
		Data	Control	Data	Control	Data	Control
All	NA	4981	994	8	0	0	0
1	NA	500	0	1	0	0	0
2	NA	500	0	1	0	0	0
3	NA	998	0	2	0	0	0
4	NA	996	2	1	0	0	0
5	NA	995	0	3	0	0	0
6	NA	992	992	0	0	0	0

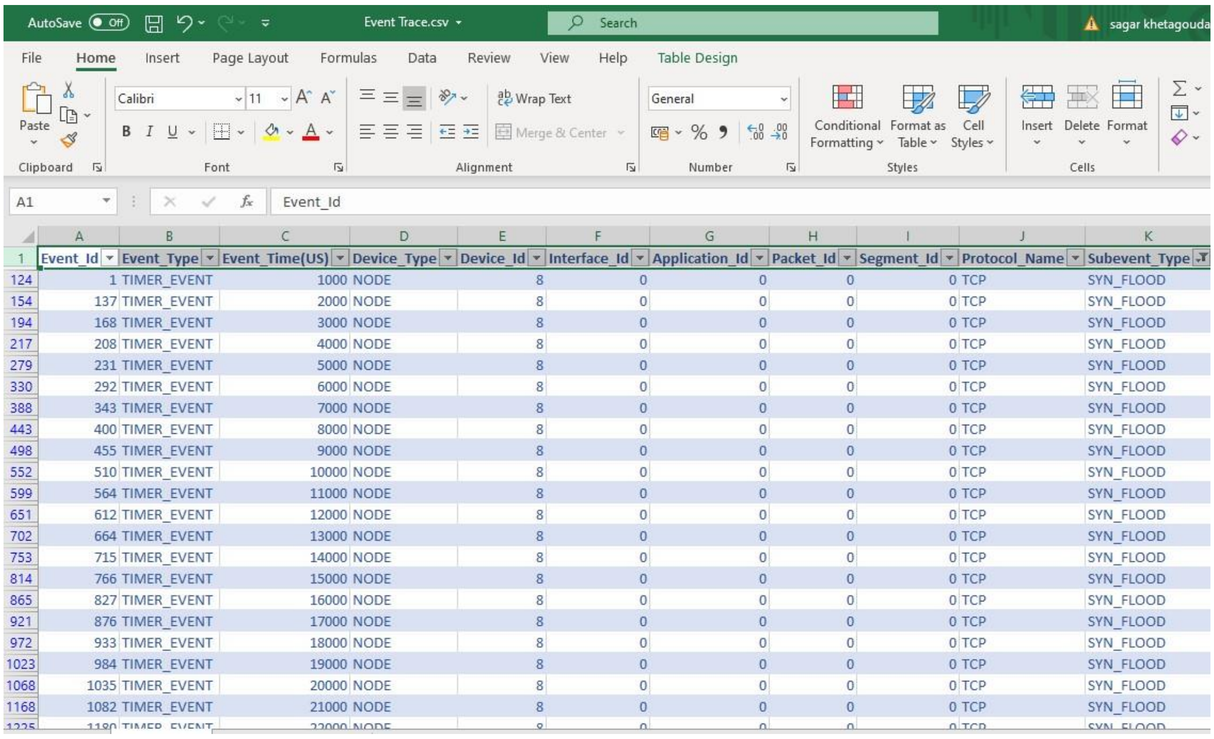
Queue_Metrics_Table

Queue_Metrics

Device_id	Port_id	Queued_packet	Dequeued_packet	Dropped_packet
5	2	997	997	0
6	1	1	1	0

	Throughput_APP1 (Mbps)	Throughput_APP2 (Mbps)
Case-1: Malicious Node =0	0.5805	0.5782
Case-2: Malicious Node =1	0.5233	0.5186
Case-3: Malicious Node =2	0.2873	0.2862

Go to the result window open Event trace, user can find out the SYN_FLOOD packets via filting subevent type as SYN_FLOOD.



	Event_Id	Event_Type	Event_Time(US)	Device_Type	Device_Id	Interface_Id	Application_Id	Packet_Id	Segment_Id	Protocol_Name	Subevent_Type
124	1	TIMER_EVENT	1000	NODE	8	0	0	0	0	TCP	SYN_FLOOD
154	137	TIMER_EVENT	2000	NODE	8	0	0	0	0	TCP	SYN_FLOOD
194	168	TIMER_EVENT	3000	NODE	8	0	0	0	0	TCP	SYN_FLOOD
217	208	TIMER_EVENT	4000	NODE	8	0	0	0	0	TCP	SYN_FLOOD
279	231	TIMER_EVENT	5000	NODE	8	0	0	0	0	TCP	SYN_FLOOD
330	292	TIMER_EVENT	6000	NODE	8	0	0	0	0	TCP	SYN_FLOOD
388	343	TIMER_EVENT	7000	NODE	8	0	0	0	0	TCP	SYN_FLOOD
443	400	TIMER_EVENT	8000	NODE	8	0	0	0	0	TCP	SYN_FLOOD
498	455	TIMER_EVENT	9000	NODE	8	0	0	0	0	TCP	SYN_FLOOD
552	510	TIMER_EVENT	10000	NODE	8	0	0	0	0	TCP	SYN_FLOOD
599	564	TIMER_EVENT	11000	NODE	8	0	0	0	0	TCP	SYN_FLOOD
651	612	TIMER_EVENT	12000	NODE	8	0	0	0	0	TCP	SYN_FLOOD
702	664	TIMER_EVENT	13000	NODE	8	0	0	0	0	TCP	SYN_FLOOD
753	715	TIMER_EVENT	14000	NODE	8	0	0	0	0	TCP	SYN_FLOOD
814	766	TIMER_EVENT	15000	NODE	8	0	0	0	0	TCP	SYN_FLOOD
865	827	TIMER_EVENT	16000	NODE	8	0	0	0	0	TCP	SYN_FLOOD
921	876	TIMER_EVENT	17000	NODE	8	0	0	0	0	TCP	SYN_FLOOD
972	933	TIMER_EVENT	18000	NODE	8	0	0	0	0	TCP	SYN_FLOOD
1023	984	TIMER_EVENT	19000	NODE	8	0	0	0	0	TCP	SYN_FLOOD
1068	1035	TIMER_EVENT	20000	NODE	8	0	0	0	0	TCP	SYN_FLOOD
1168	1082	TIMER_EVENT	21000	NODE	8	0	0	0	0	TCP	SYN_FLOOD
1225	1180	TIMER_EVENT	22000	NODE	8	0	0	0	0	TCP	SYN_FLOOD

Note: Users can also create their own network scenarios in Internetworks and run simulation.