**Dos Attack in Internet of Things**

**Software Used:** NetSim Standard v12.1 (32/64 bit), Visual Studio 2015/2017/2019

Follow the instructions specified in the following link to clone/download the project folder from GitHub using Visual Studio:

[https://tetcos.freshdesk.com/support/solutions/articles/14000099351-how-to-clone-netsim-file-](https://tetcos.freshdesk.com/support/solutions/articles/14000099351-how-to-clone-netsim-file-exchange-project-repositories-from-github-) [exchange-project-repositories-from-github-](https://tetcos.freshdesk.com/support/solutions/articles/14000099351-how-to-clone-netsim-file-exchange-project-repositories-from-github-)

Other tools such as GitHub Desktop, SVN Client, Sourcetree, Git from the command line, or any client you like to clone the Git repository.

**Note**: It is recommended not to download the project as an archive (compressed zip) to avoid incompatibility while importing workspaces into NetSim.

# Secure URL for the GitHub repository:

**https://github.com/NetSim-TETCOS/DOS\_Attack\_v12.0.git**

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.

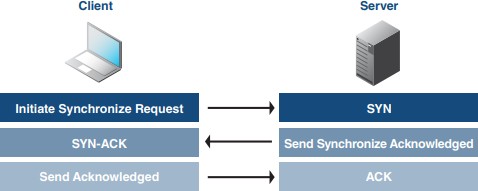
# Standard DDoS Attack types:

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:

# 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.

# 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

# 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);

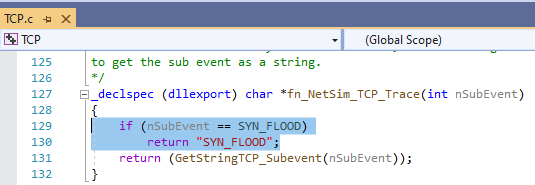
This function is used to create and send SYN packet to the network layer

* void syn\_flood();

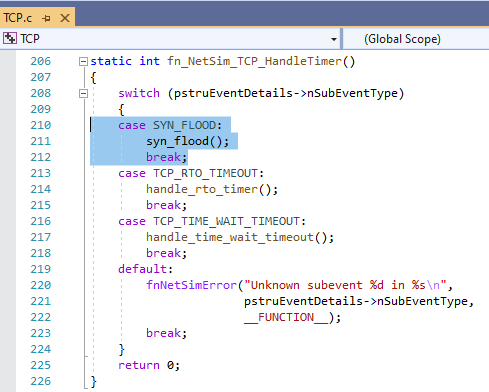
This function is used to check whether the socket is present or not and also adds a timer event called SYN\_FLOOD (triggers for every 1000µs)

# Code modifications done in NetSim:

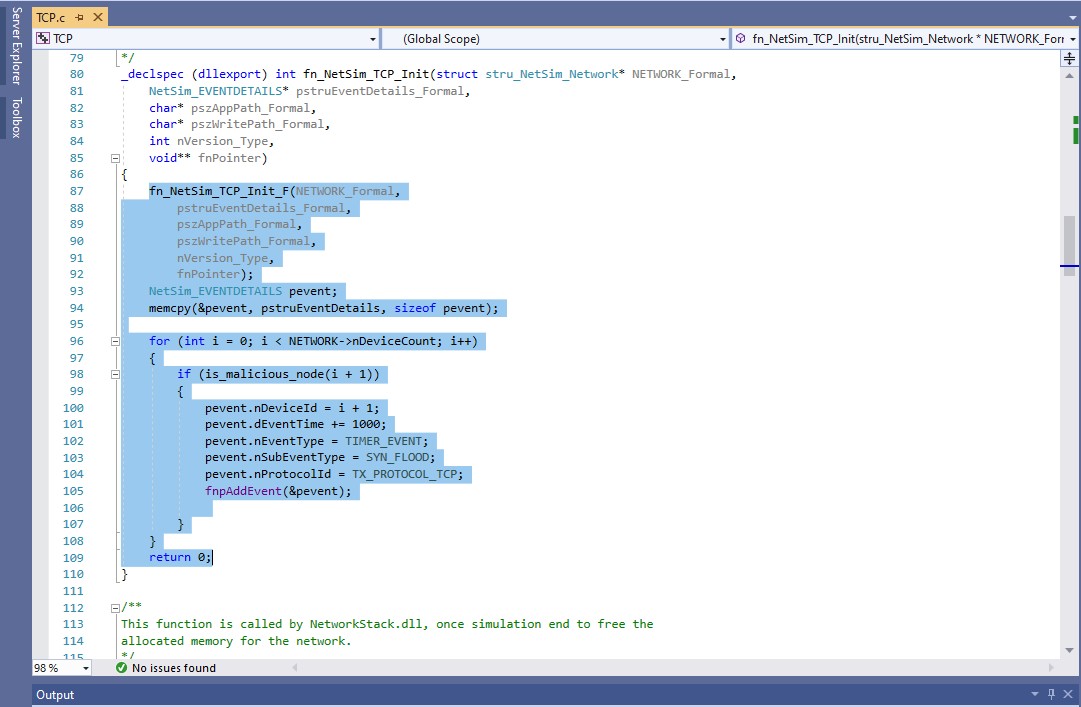
1. We have added the following lines of code in fn\_NetSim\_TCP\_Trace() function present in TCP.c file inside TCP project. This is used to add the SYN\_FLOOD sub-events in Event Trace file



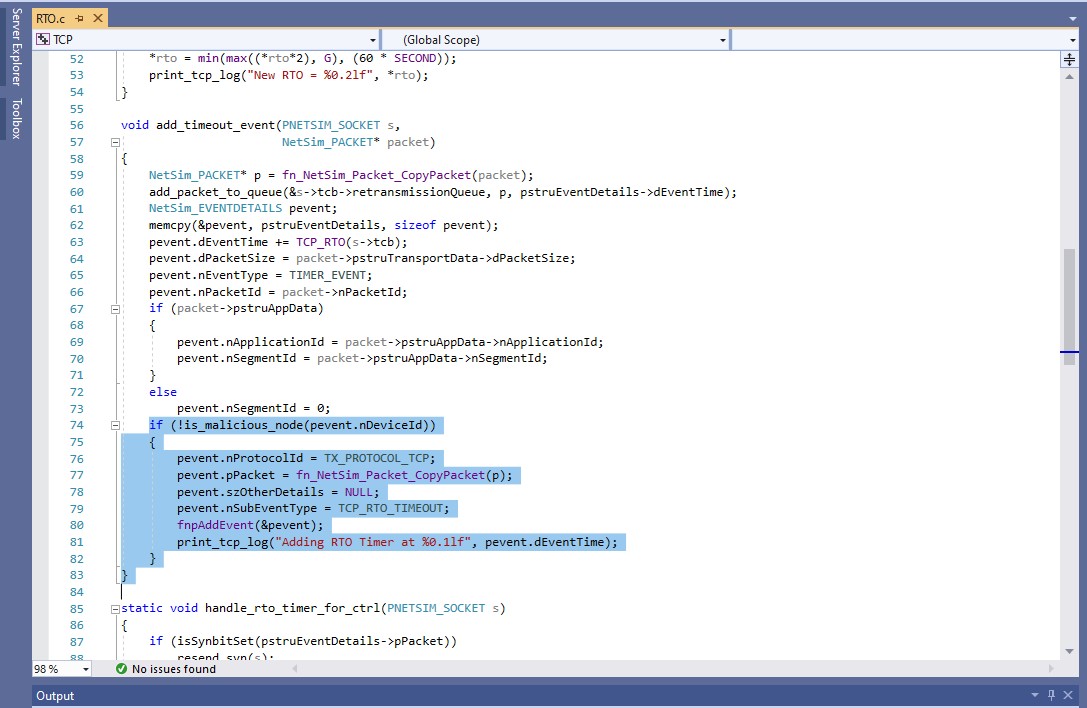
1. We have added the following lines of code in fn\_NetSim\_TCP\_HandleTimer() function present in TCP.c file inside TCP project. Used to add a TCP sub\_event called SYN\_FLOOD



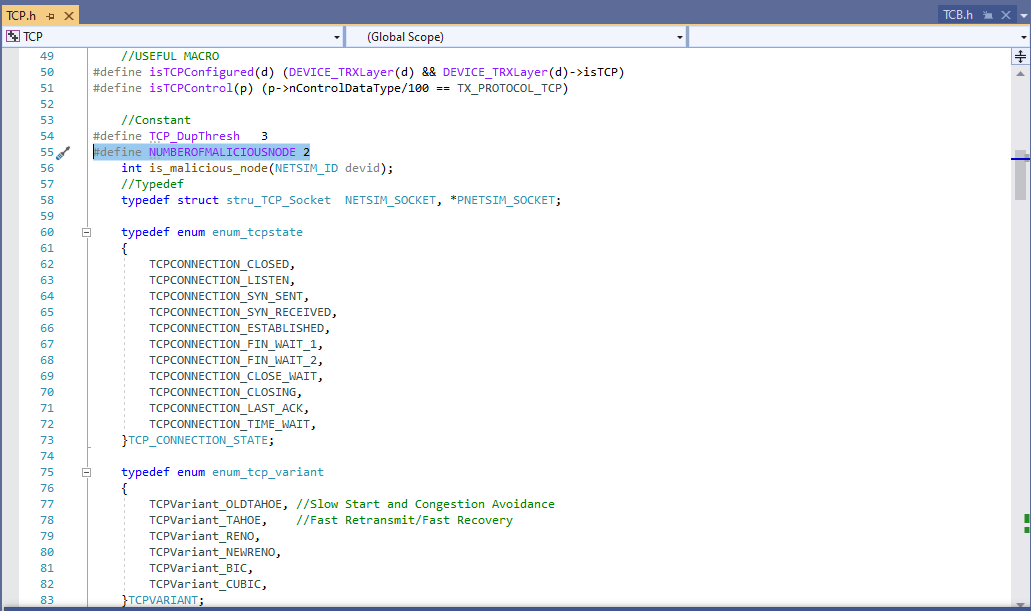
1. And modified the following lines of code in fn\_NetSim\_TCP\_Init() function resent in TCP.c inside TCP project



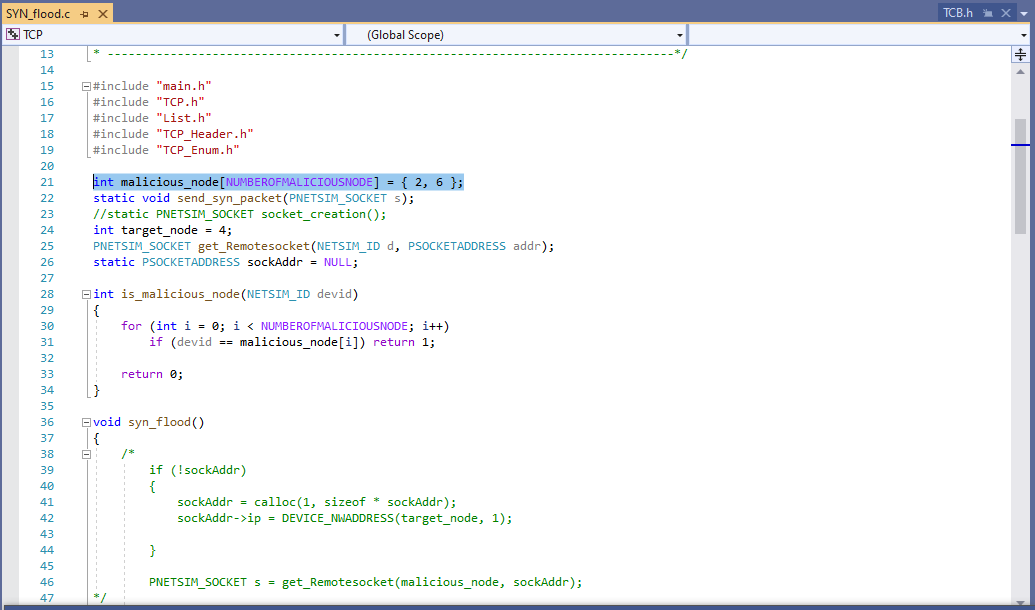
1. And modified the following lines of code in add\_timeout\_event() present in RTO.c file inside TCP project which avoids RTO timer for malicious nodes



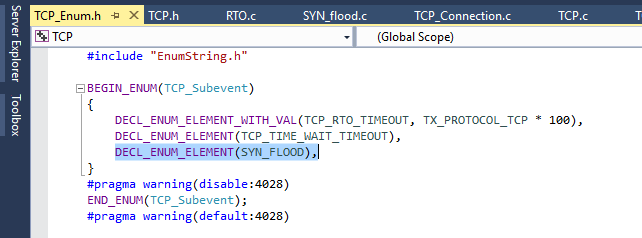
1. Users can give their own number of malicious node in **TCP.h** file inside TCP project



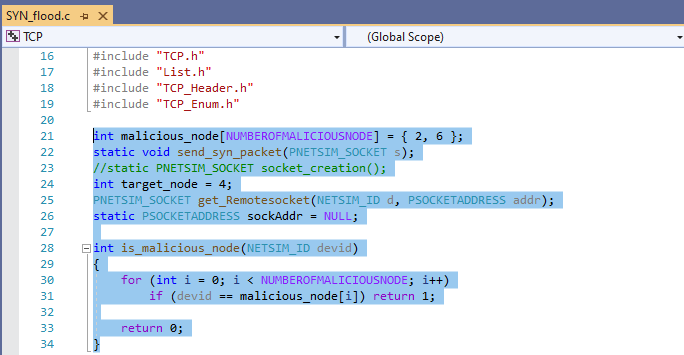
1. Users can give their own target ID and malicious ID in **SYN\_FLOOD.c** file inside TCP project

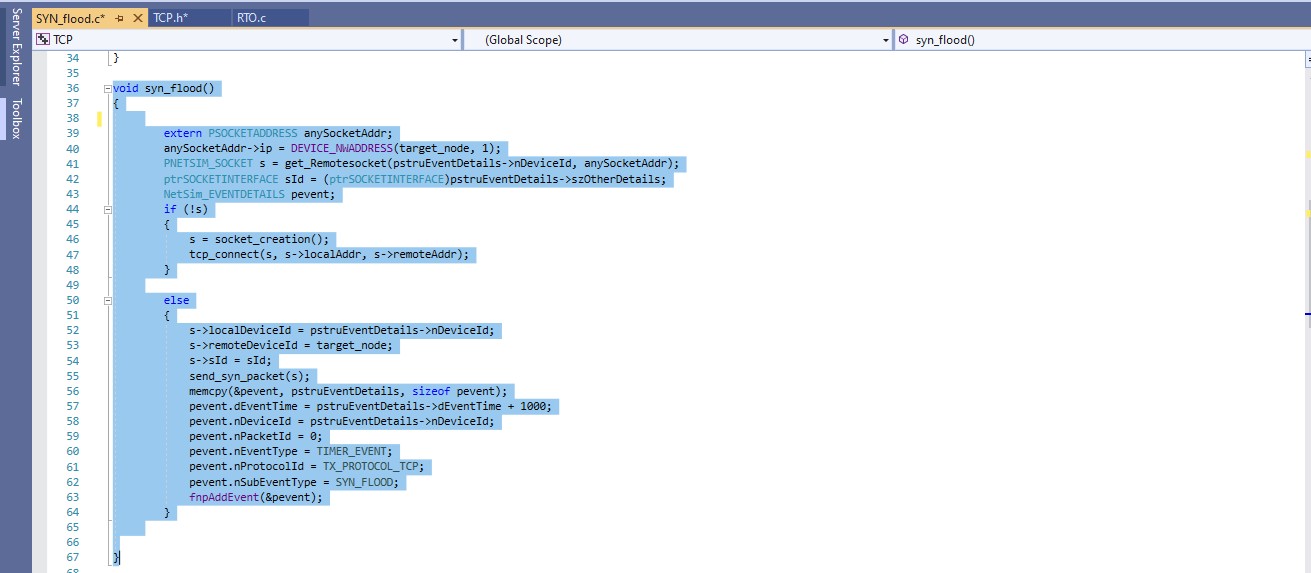


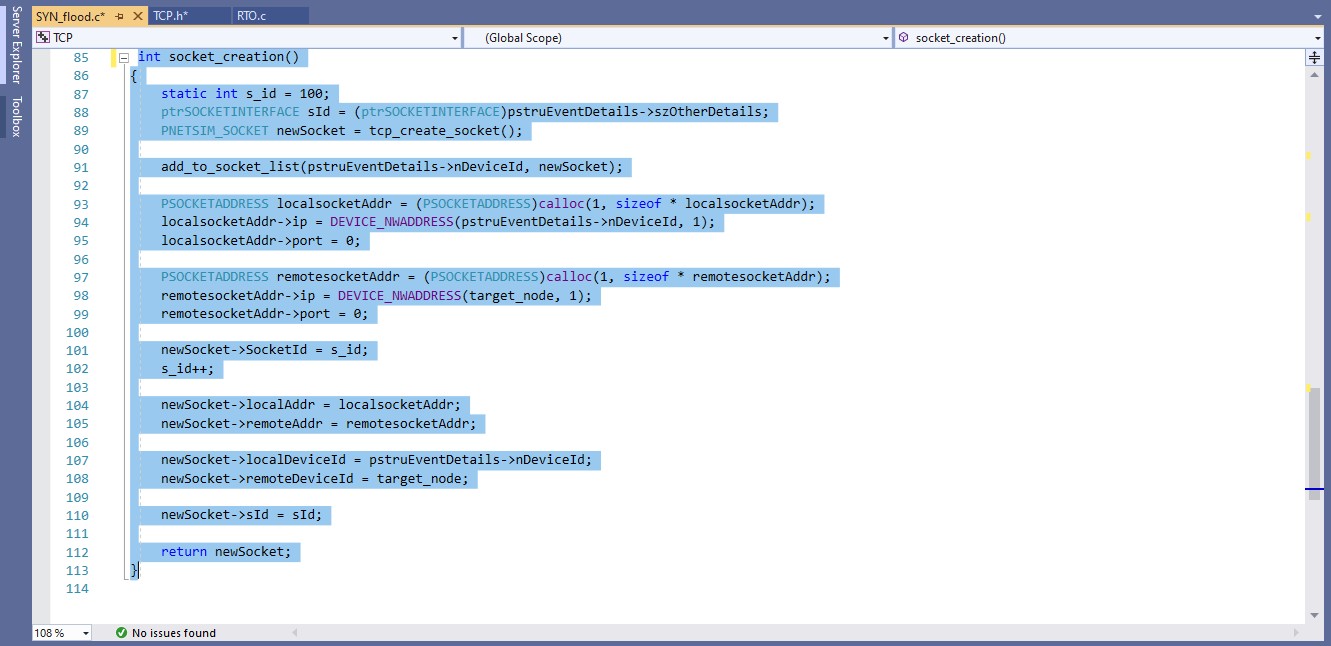
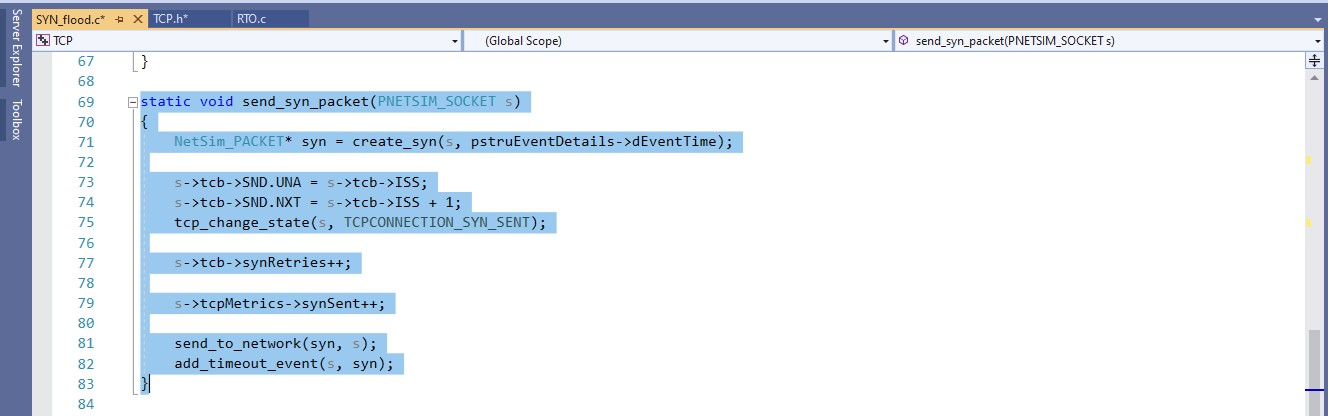
1. Added the following line in TCP\_Enum.h file inside TCP project to add a new TCP\_subevent called SYN\_FLOOD



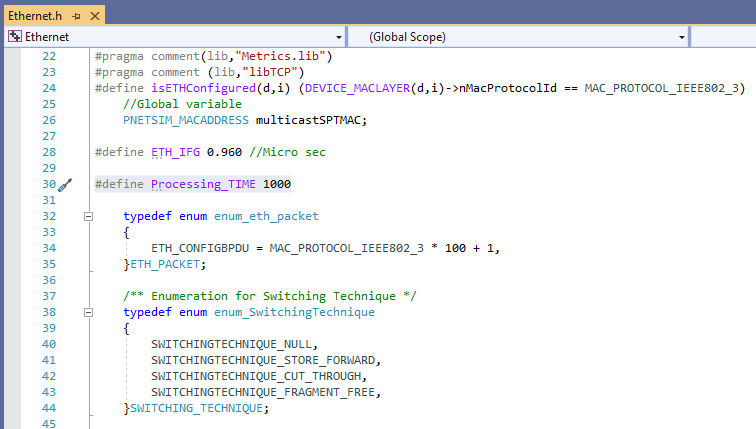
1. SYN\_FLOOD.c file contains the following functions

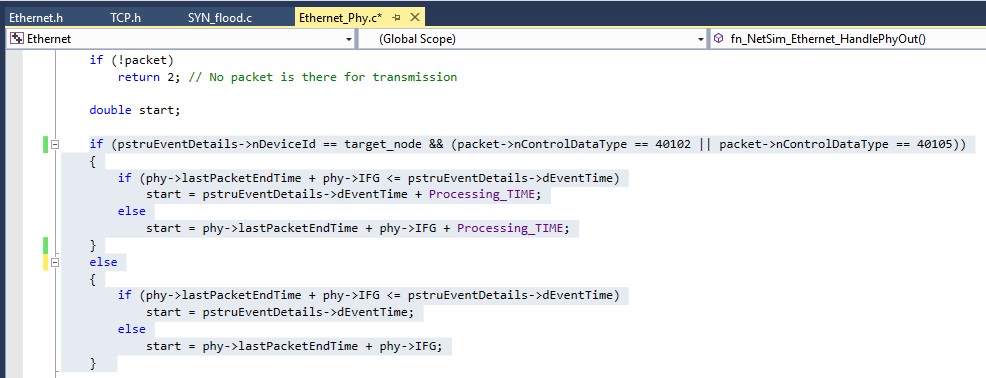




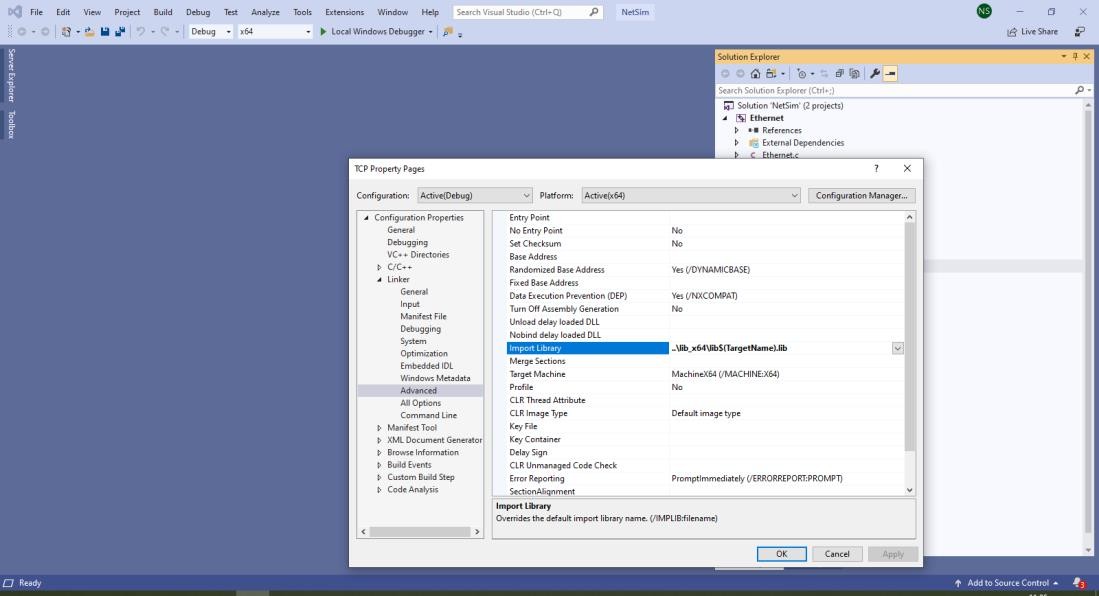


1. Added PROCESSING\_TIME macro in Ethernet.h file inside ETHERNET project



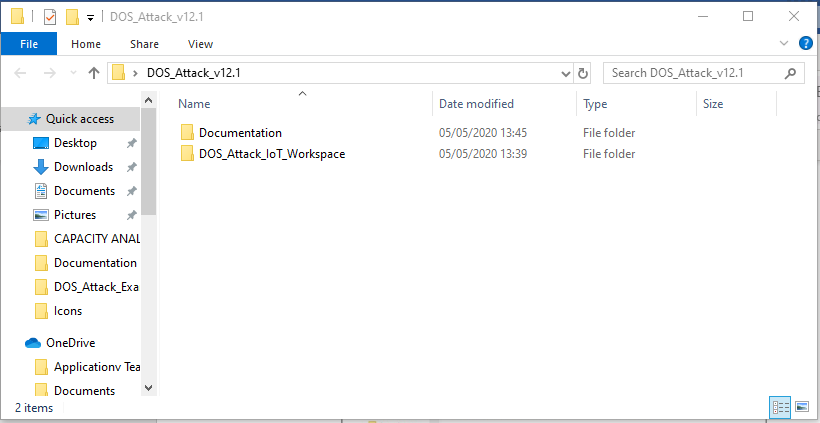
1. Modified the following lines of code in fn\_NetSim\_Ethernet\_HandlePhyOut() function present in Ethernet\_Phy.c file inside Ethernet project.
2. Right click on TCP project → Properties→Linker → Advanced →import library 32-bit and 64-bit

# ..\lib\lib$(TargetName).lib or ..\lib\_x64\lib$(TargetName).lib



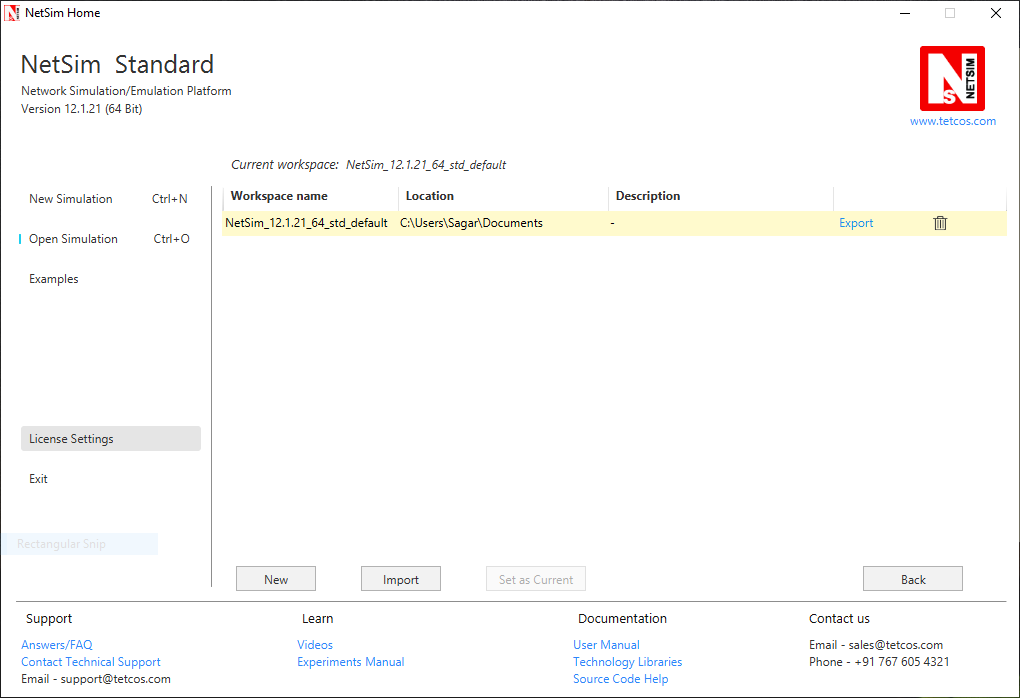
**Steps:**

* 1. The downloaded project folder contains the folders Documentation, and DOS\_Attack\_IoT\_Workspace directory as shown below:

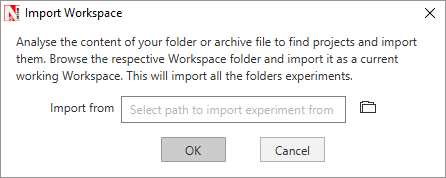


* 1. Import DOS\_Attack\_IoT\_Workspace by going to Open Simulation->Workspace Options-

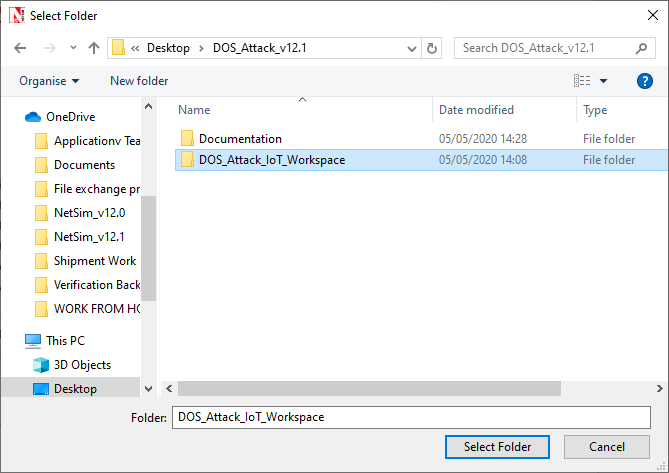
>More Options in NetSim Home window. Then select Import as shown below:



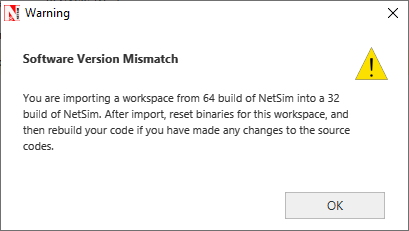
* 1. It displays a window where users need to give the path of the workspace folder and click on OK as shown below:



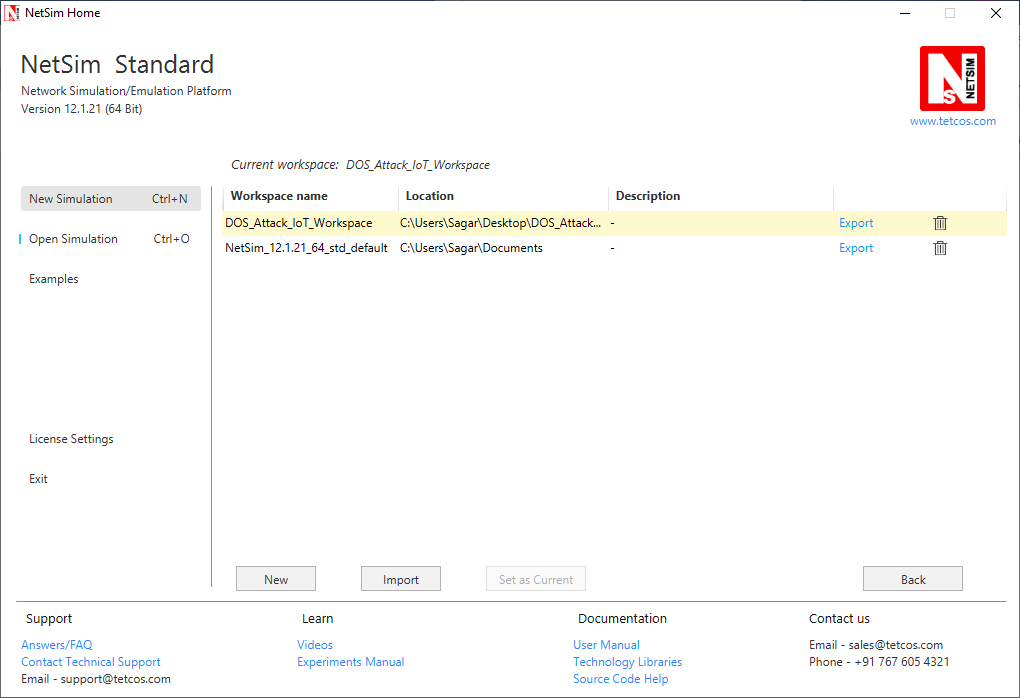
* 1. Browse to the DOS\_Attack\_IoT\_Workspace folder and click on select folder as shown below:



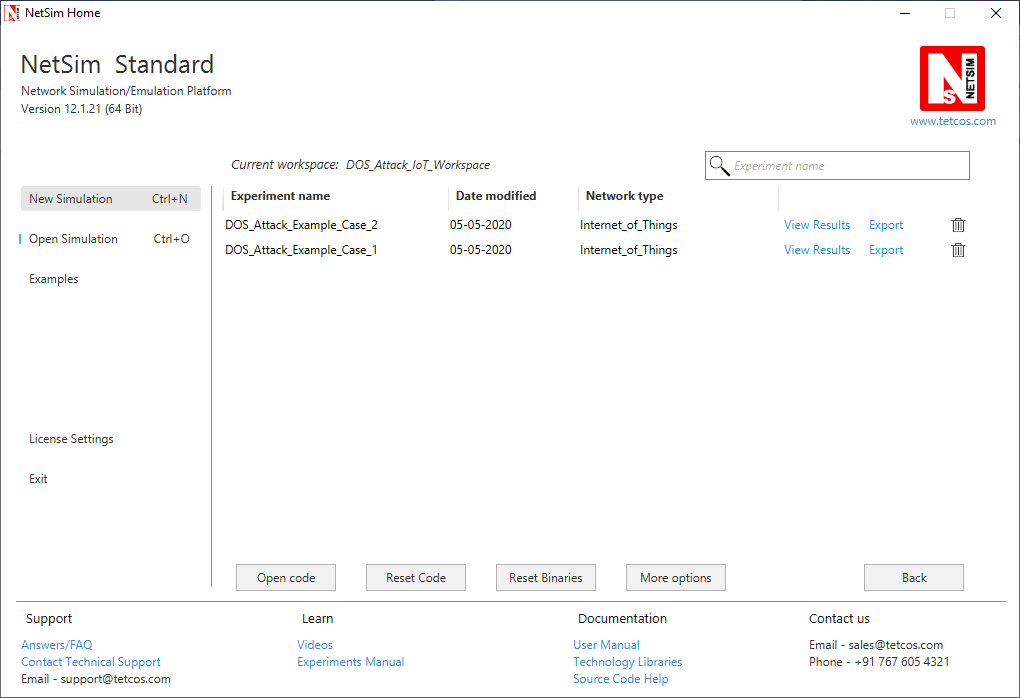
* 1. After this click on OK button in the Import Workspace window.
  2. While importing the workspace, if the following warning message indicating Software Version Mismatch is displayed, you can ignore it and proceed.



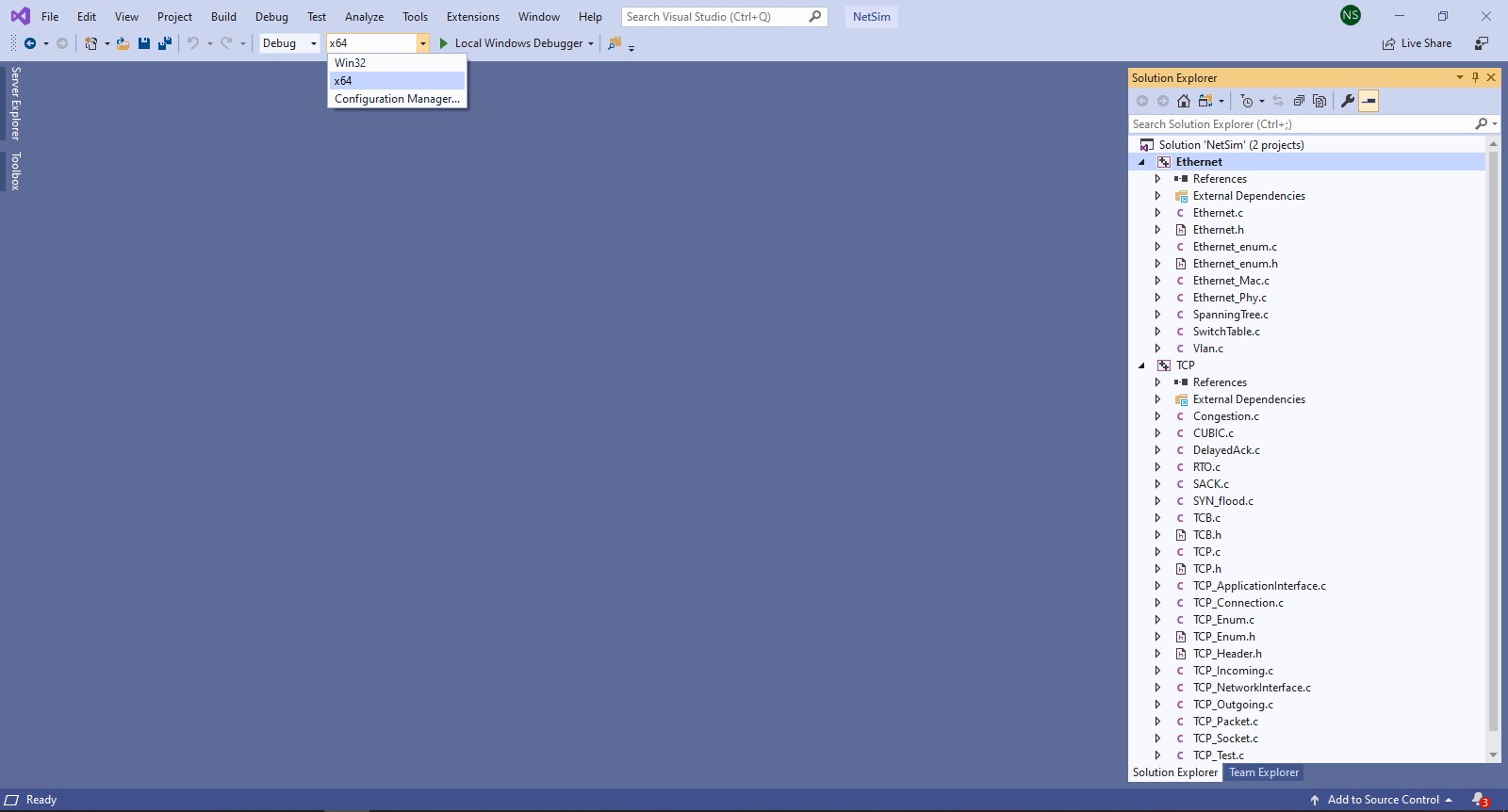
* 1. The Imported workspace will be set as the current workspace automatically. To see the imported workspace, click on Open Simulation->Workspace Options->More Options as shown below:



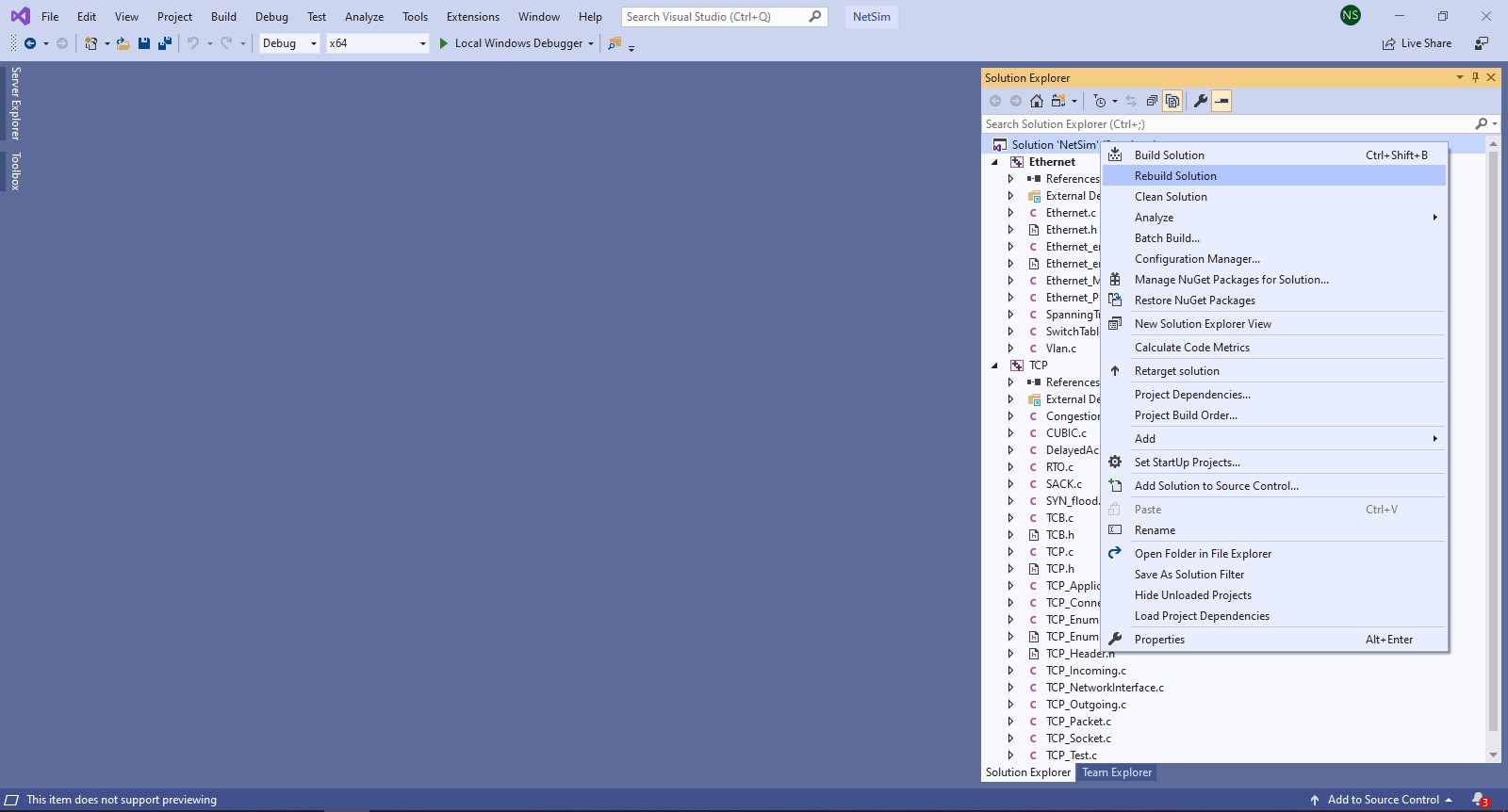
* 1. Open the Source codes in Visual Studio by going to Open Simulation-> Workspace Options and Clicking on Open code button as shown below:



* 1. Under the **TCP** project in the solution explorer you will be able to see that **SYN\_FLOOD.c** file.
  2. 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:



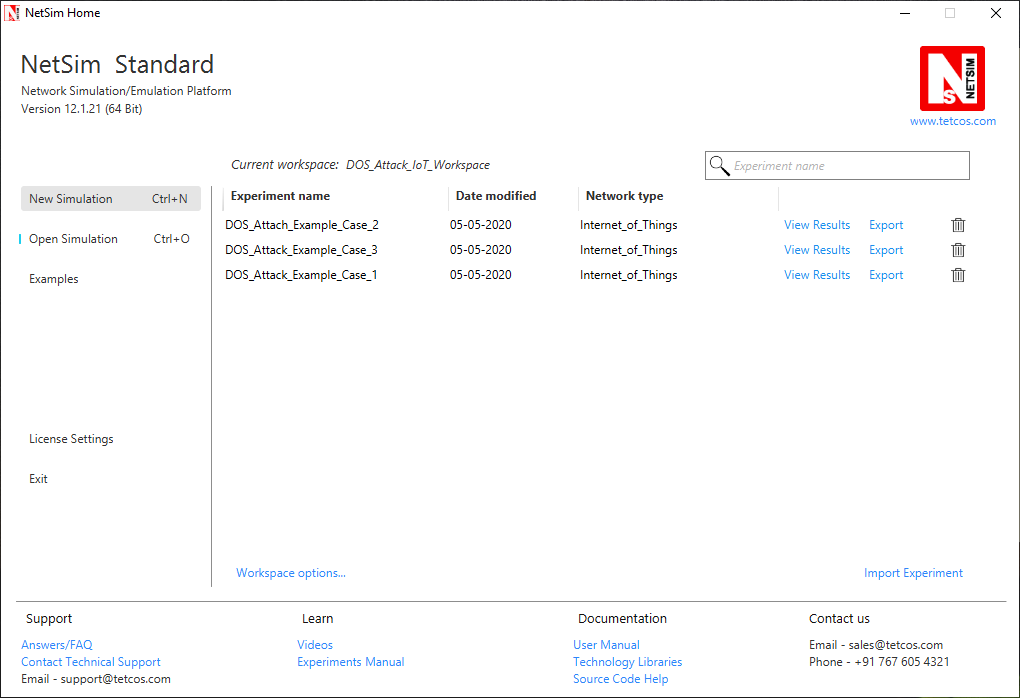
* 1. Right click on the solution in the solution explorer and select Rebuild.



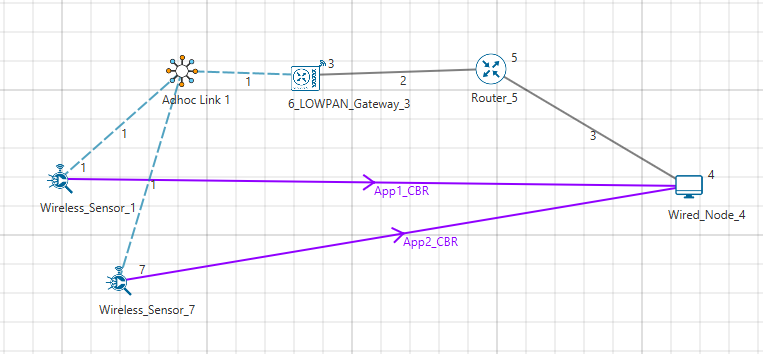
* 1. Upon successful build modified libTCP.dll and libEthernet.dll file gets automatically updated in the directory containing NetSim binaries. (Note: first rebuild the TCP project and then rebuild the Ethernet project)
  2. Run NetSim as Administrative mode.

# Case-1: Without Malicious Node

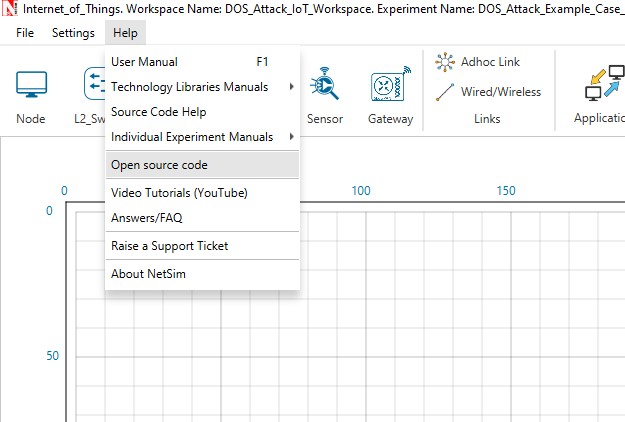
1. Then DOS\_Attack\_IoT\_Workspace comes with a sample configuration that is already saved. To open this example, go to Open Simulation and click on the DOS\_Attack\_Example\_Case\_1 that is present under the list of experiments as shown below:



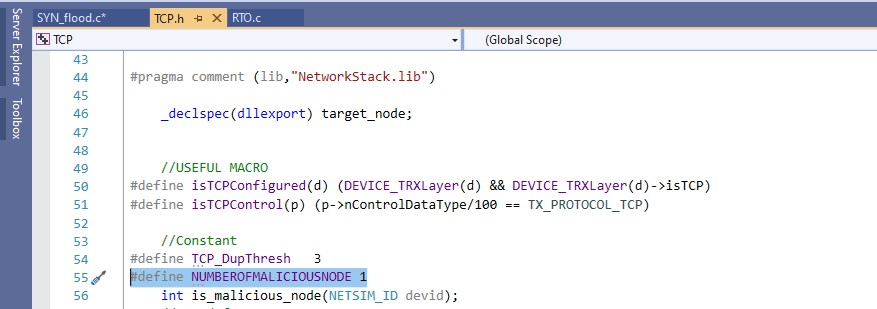
1. The saved network scenario consisting of 2 sensors, 1 6LOWPAN Gateway, 1 router, and 1 wired node in the grid environment forming a IoT Network. Traffic is configured from sensor node to the Wired Node.



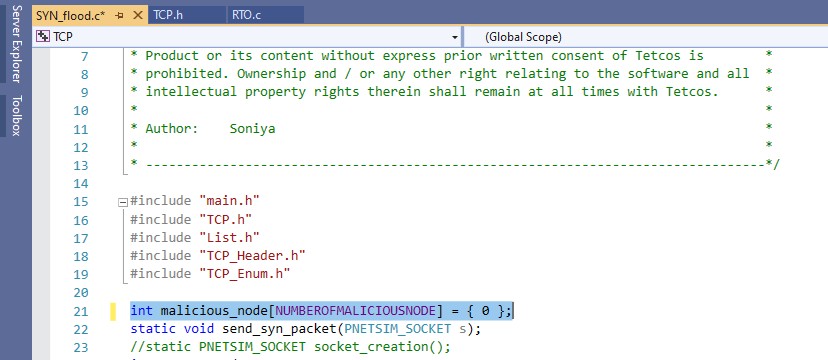
1. Help → Open Source code



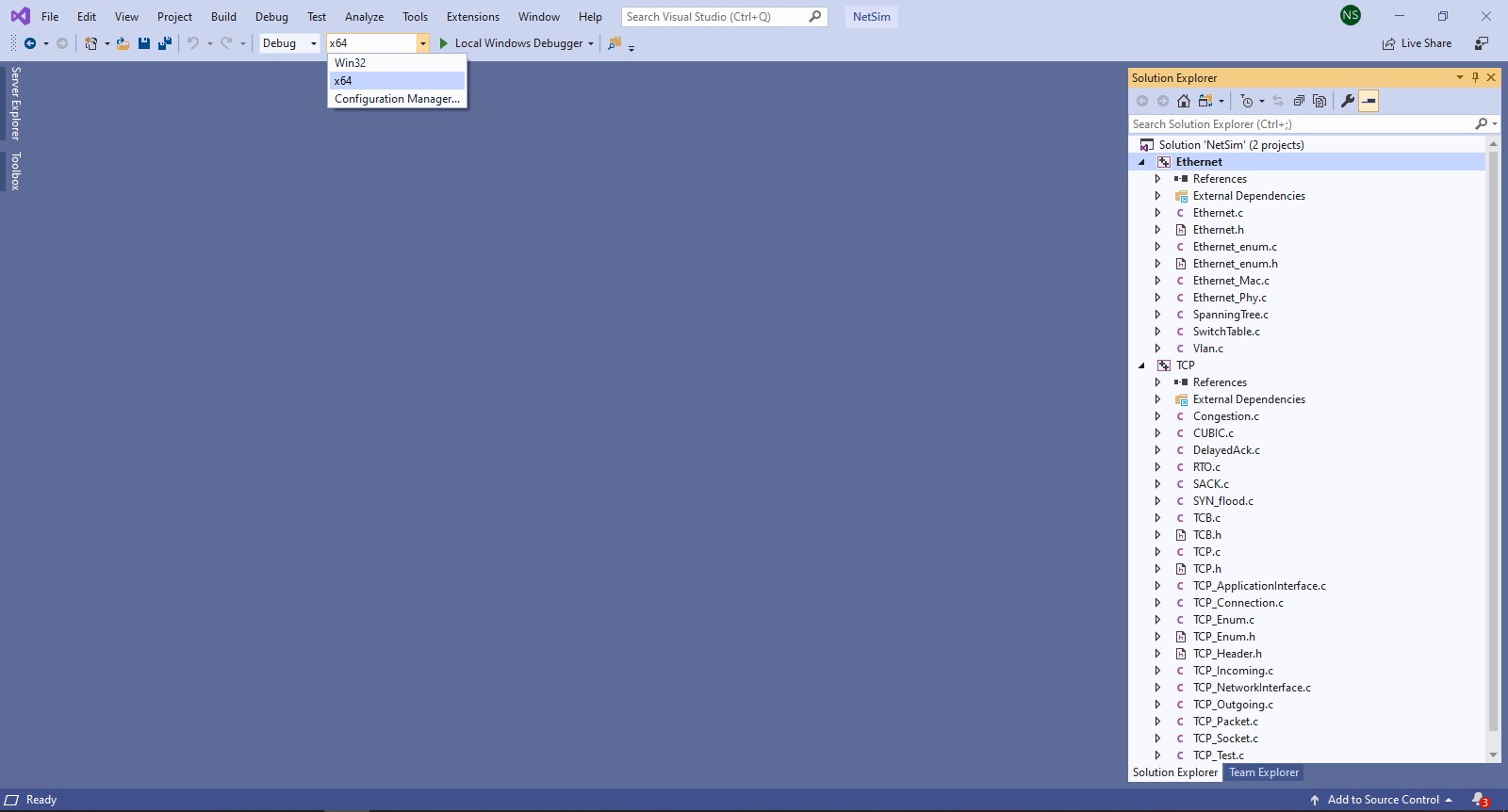
1. In TCP.h set **NUMBEROFMALICIOUSNODE** as 1.



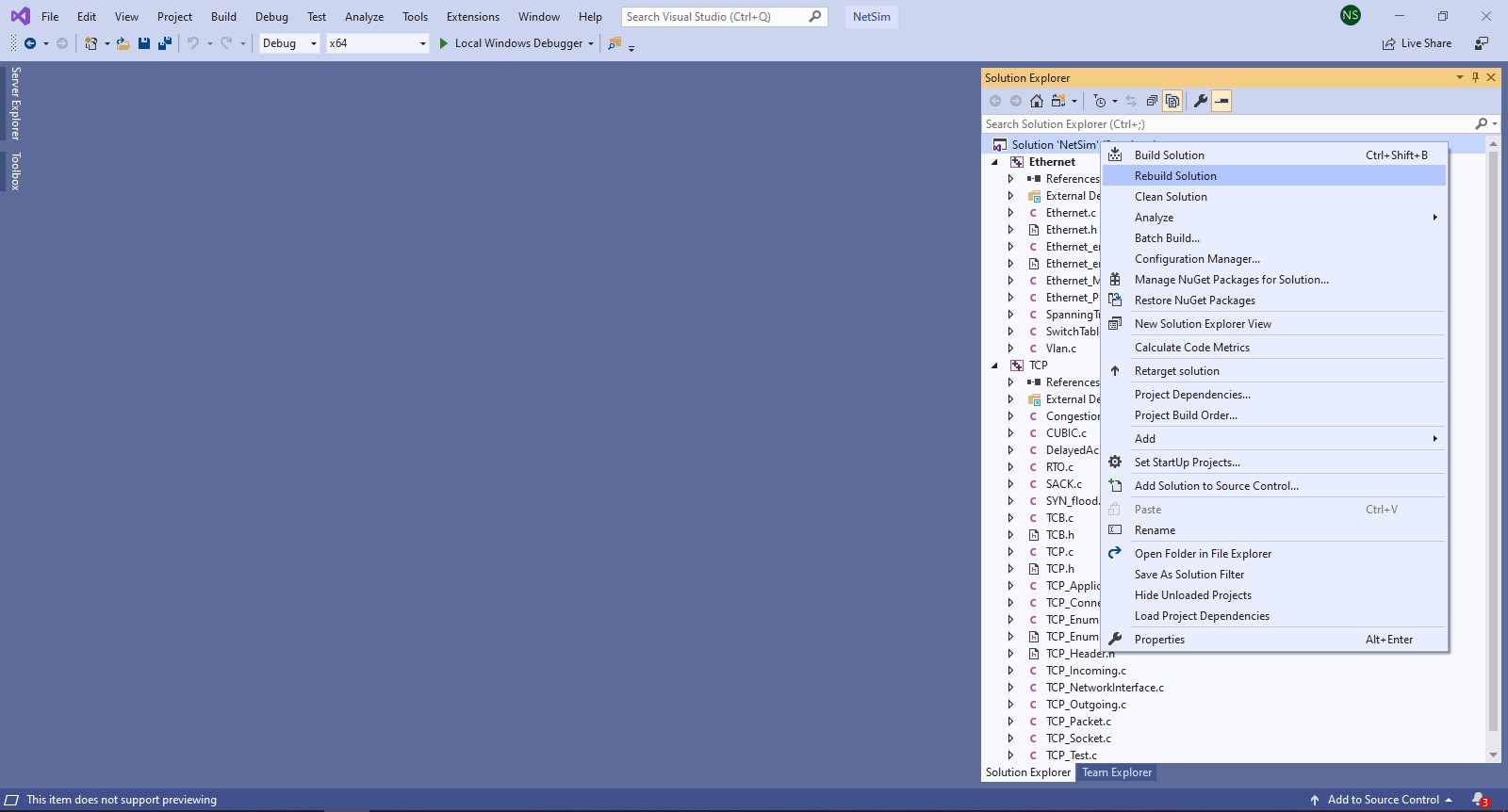
1. In SYN\_FLOOD.c set **malicious node** as 0**.**



1. 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:



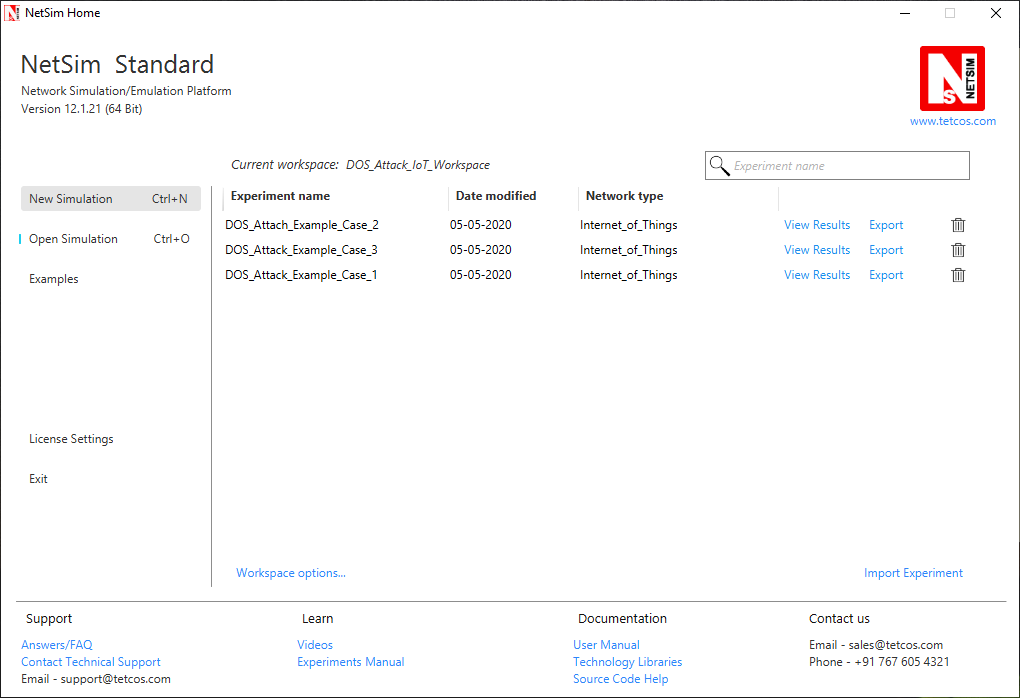
1. Right click on the solution in the solution explorer and select Rebuild.



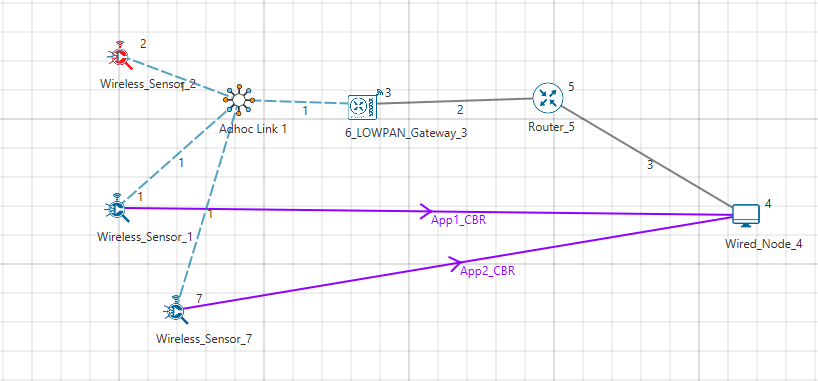
1. Upon successful build modified libTCP.dll and libEthernet.dll file gets automatically updated in the directory containing NetSim binaries. (Note: first rebuild the TCP project and then rebuild the Ethernet project)
2. Run the simulation for 100 seconds.

# Case-2: With one Malicious Node

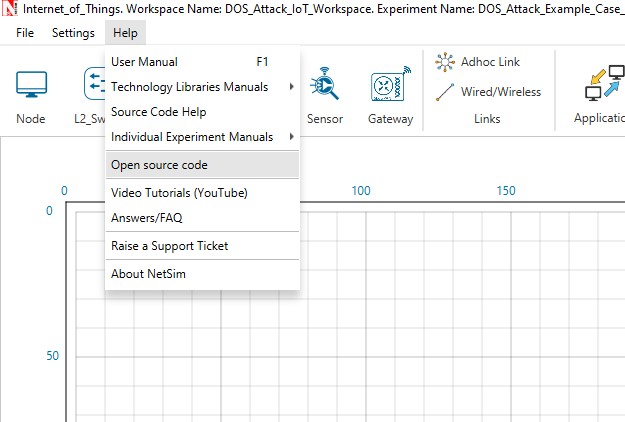
1. Then DOS\_Attack\_IoT\_Workspace comes with a sample configuration that is already saved. To open this example, go to Open Simulation and click on the DOS\_Attack\_Example\_Case\_2 that is present under the list of experiments as shown below:



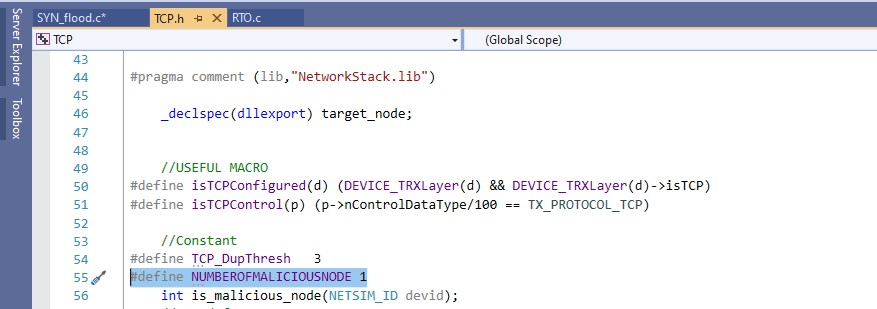
1. The saved network scenario consisting of 3 sensors, 1 6LOWPAN Gateway, 1 router, and 1 wired node in the grid environment forming a IoT Network. Traffic is configured from sensor node to the Wired Node.



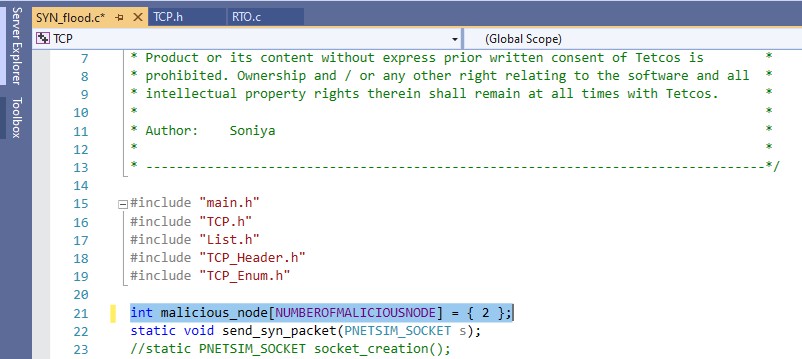
1. Help → Open Source code



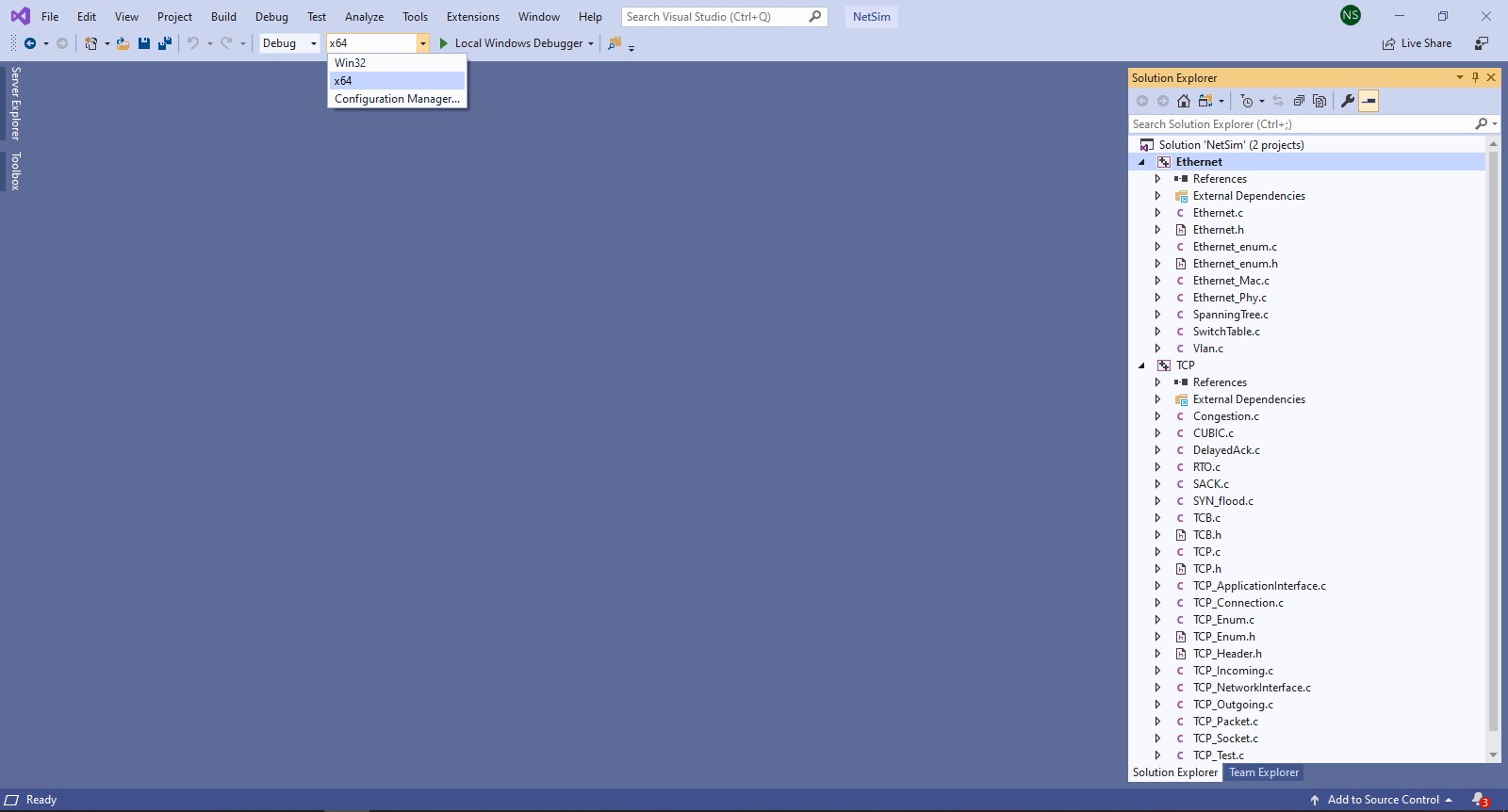
1. In TCP.h set **NUMBEROFMALICIOUSNODE** as 1.



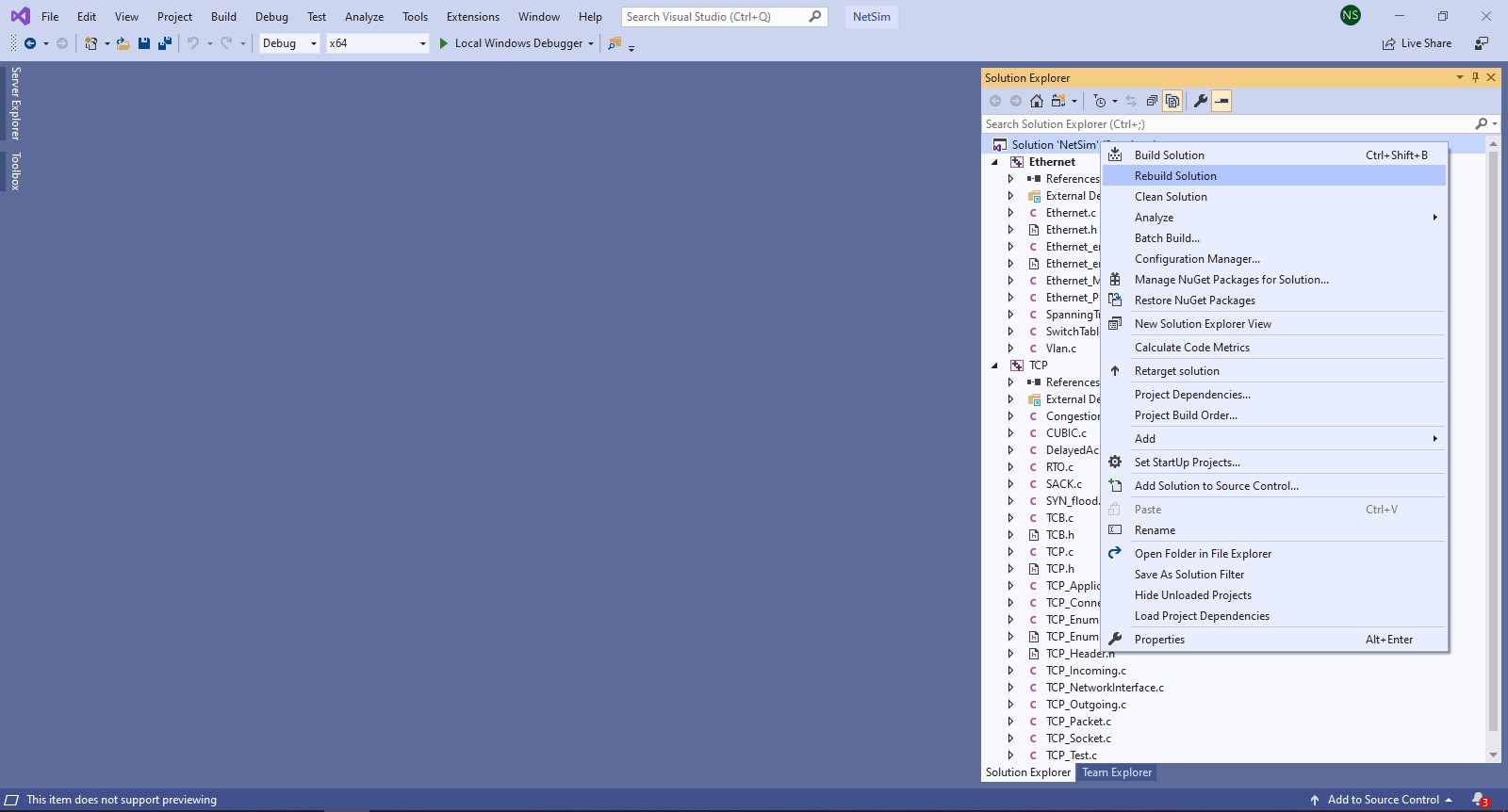
1. In SYN\_FLOOD.c set **malicious node** as 2**.**



1. 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:



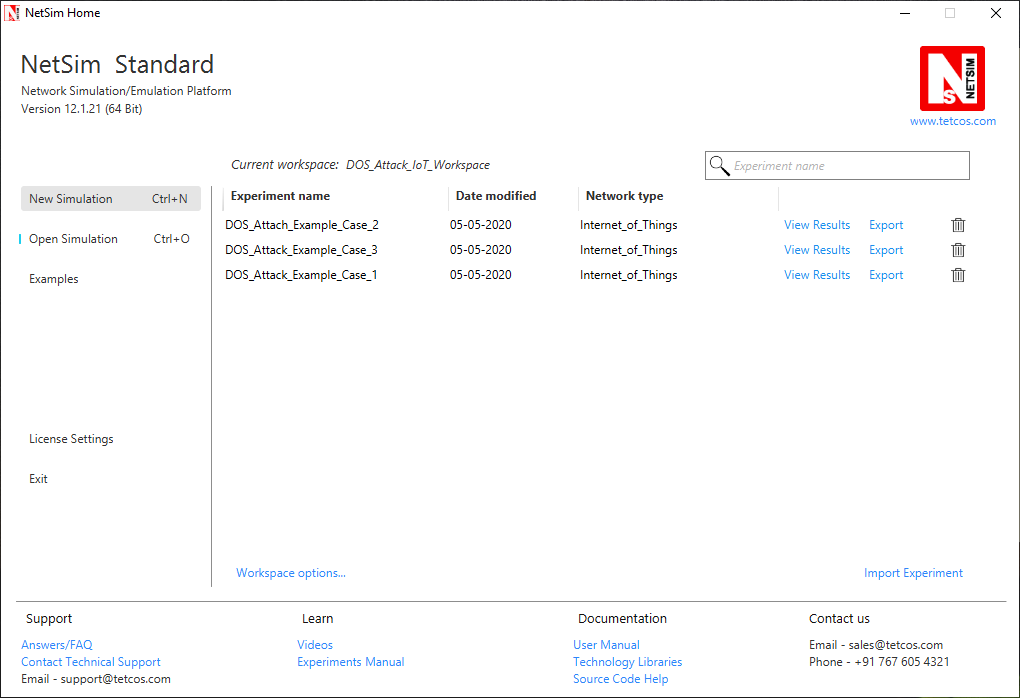
1. Right click on the solution in the solution explorer and select Rebuild.



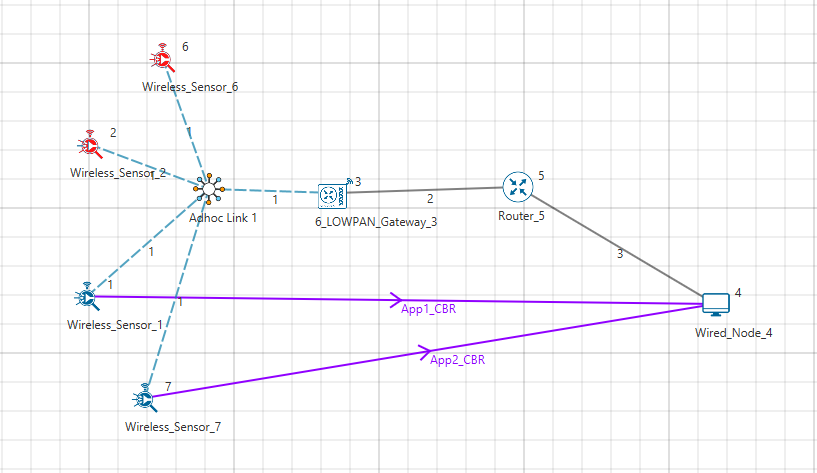
1. Upon successful build modified libTCP.dll and libEthernet.dll file gets automatically updated in the directory containing NetSim binaries. (Note: first rebuild the TCP project and then rebuild the Ethernet project)
2. Run the simulation for 100 seconds.

# Case-3: With two Malicious Node

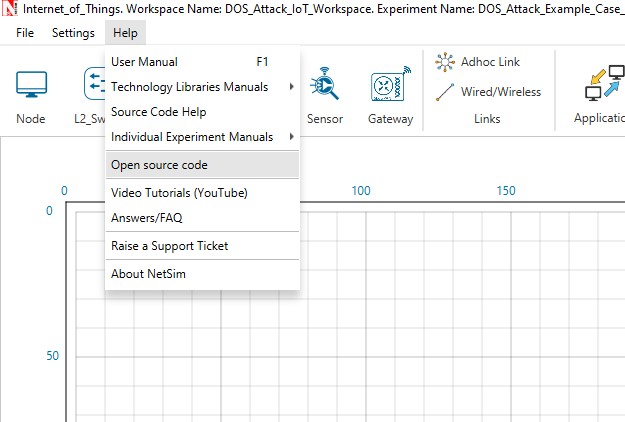
1. Then DOS\_Attack\_IoT\_Workspace comes with a sample configuration that is already saved. To open this example, go to Open Simulation and click on the DOS\_Attack\_Example\_Case\_3 that is present under the list of experiments as shown below:



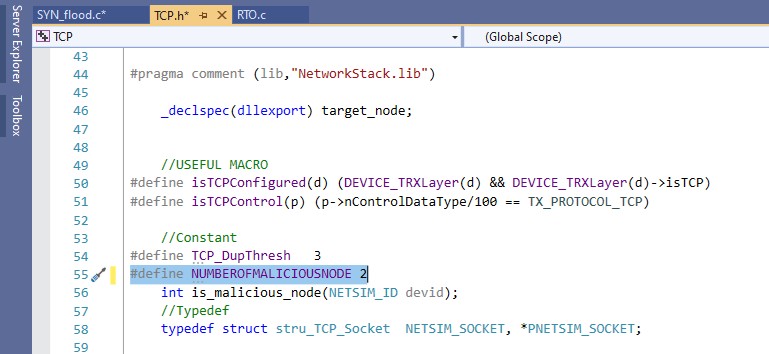
1. The saved network scenario consisting of 4 sensors, 1 6LOWPAN Gateway, 1 router, and 1 wired node in the grid environment forming a IoT Network. Traffic is configured from sensor node to the Wired Node.



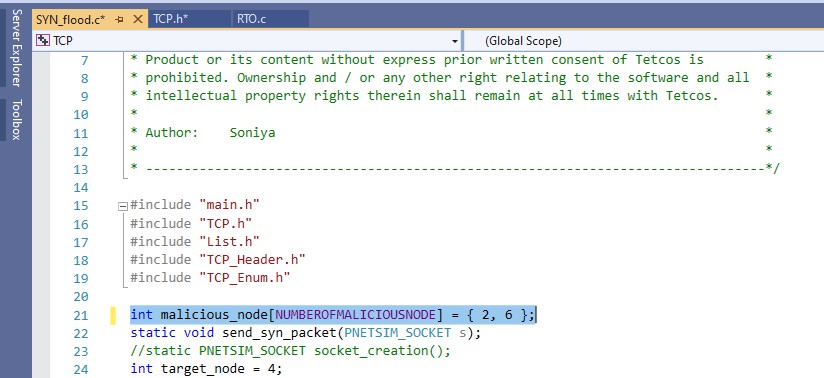
1. Help → Open Source code



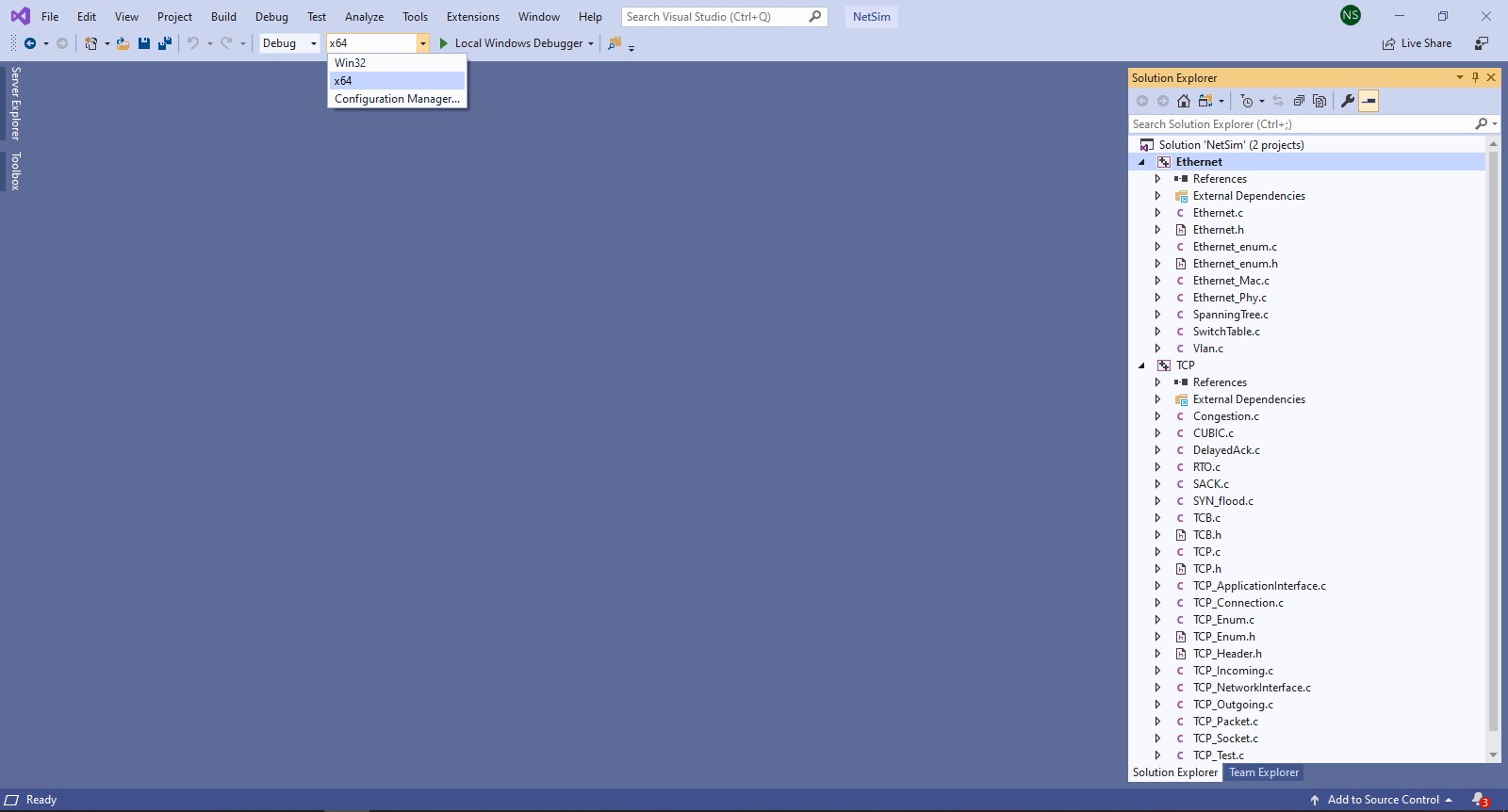
1. In TCP.h set **NUMBEROFMALICIOUSNODE** as 2.



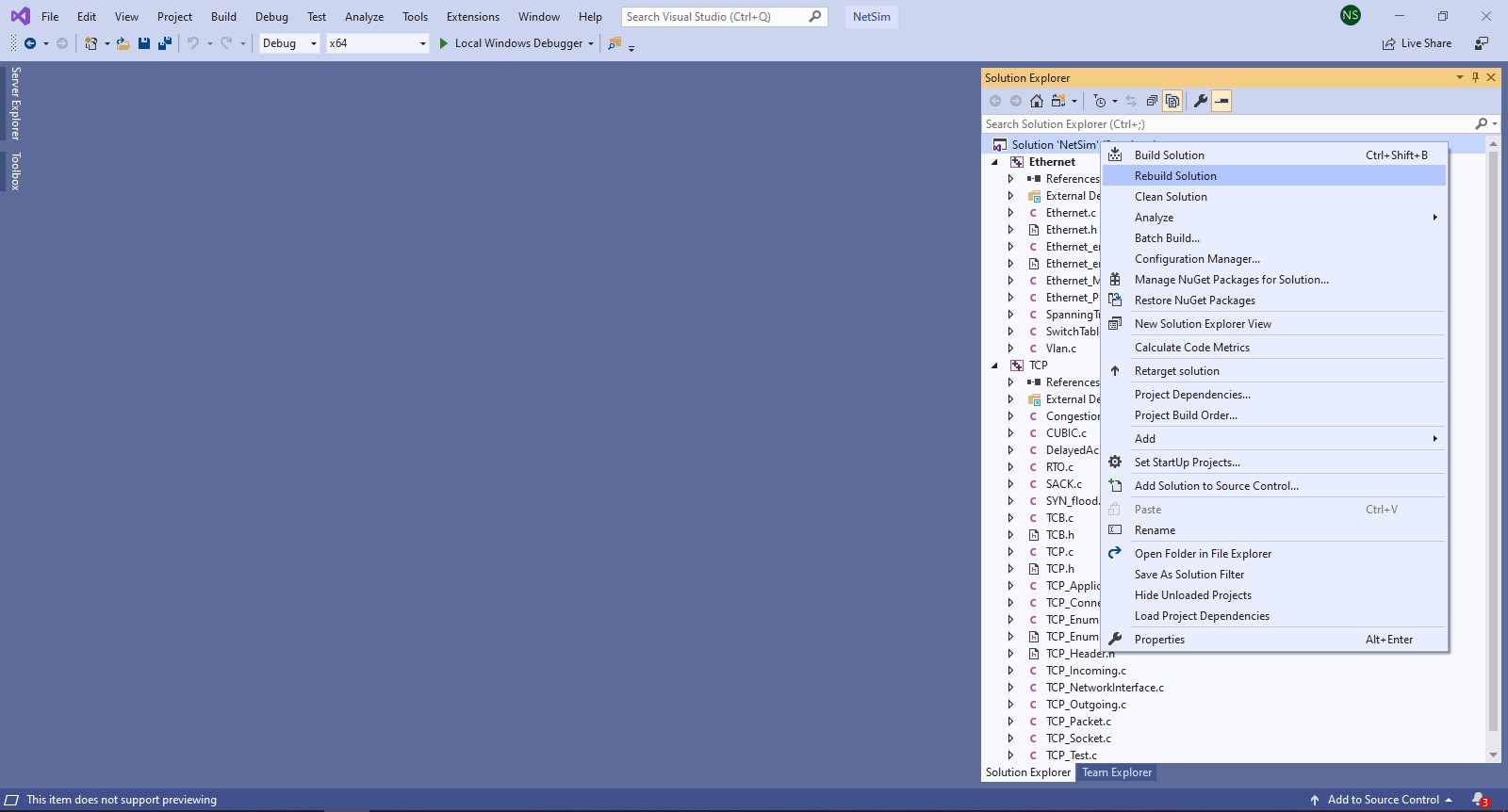
1. In SYN\_FLOOD.c set **malicious node** as 2, 6**.**



1. 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:



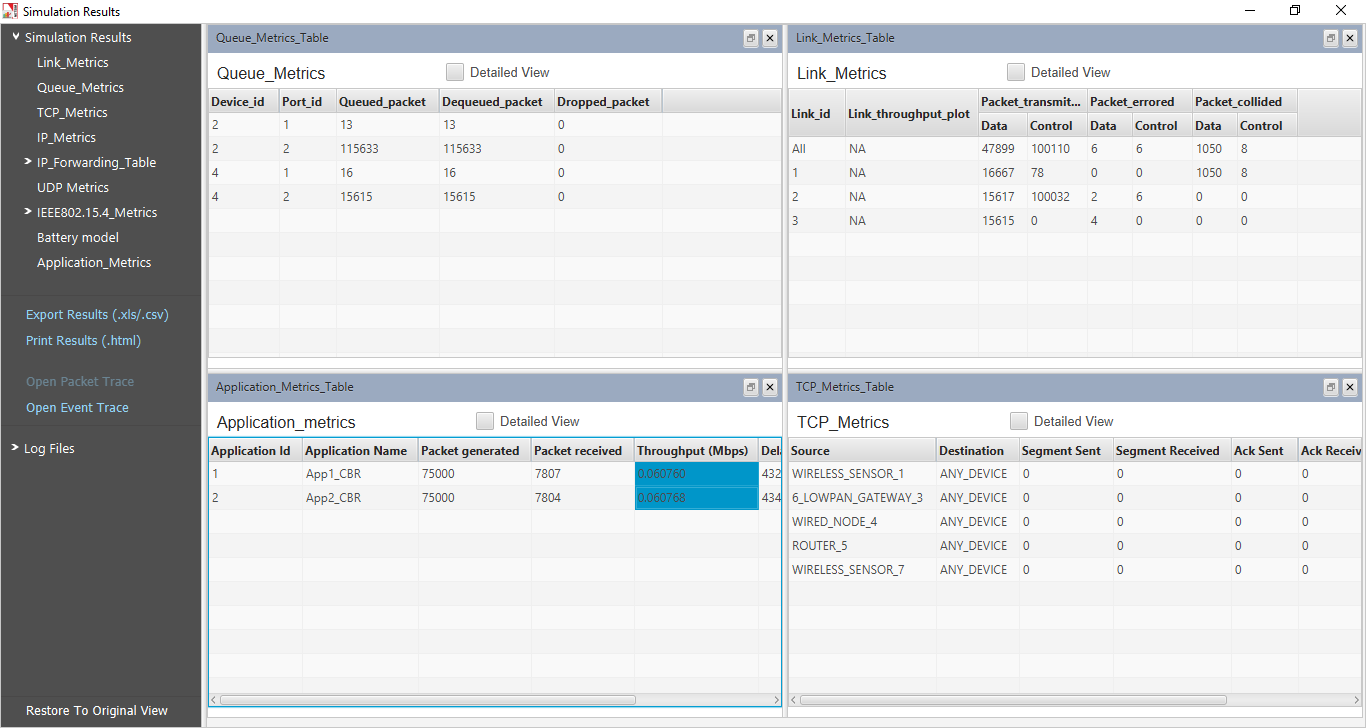
1. Right click on the solution in the solution explorer and select Rebuild.



1. Upon successful build modified libTCP.dll and libEthernet.dll file gets automatically updated in the directory containing NetSim binaries. (Note: first rebuild the TCP project and then rebuild the Ethernet project)
2. Run the simulation for 100 seconds.

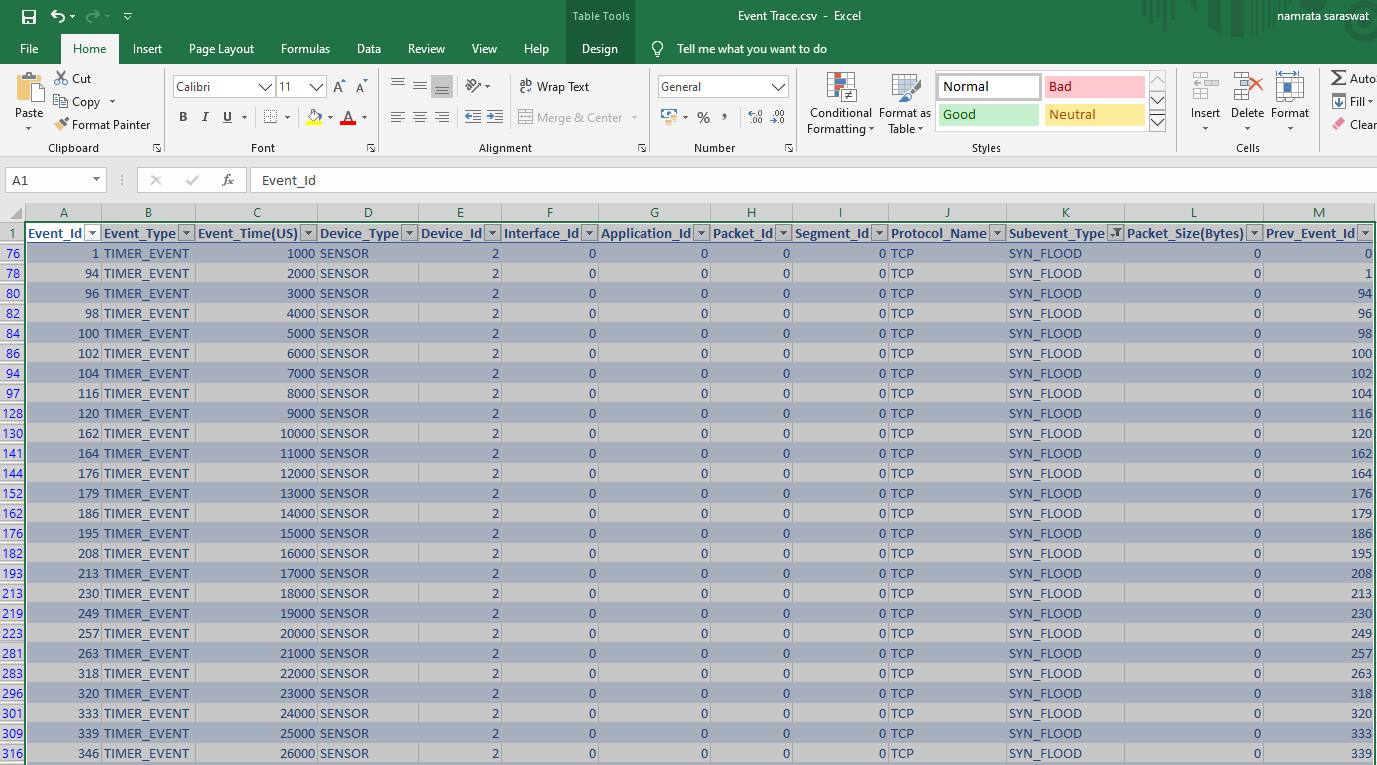
# Result:

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 sends from the malicious node, in case 1 there is no malicious node so there will be no SYN\_FLOOD packets.



|  |  |  |  |
| --- | --- | --- | --- |
|  | **Throughput\_APP1 (Mbps)** |  | **Throughput\_APP2 (Mbps)** |
| **Case-1: Malicious Node =0** | 0.060768 | | 0.060779 |
| **Case-2: Malicious Node =1** | 0.048536 | | 0.049747 |
| **Case-3: Malicious Node =2** | 0.042496 | | 0.041686 |

Go to the result window open Event trace, user can find out the SYN\_FLOOD packets via filtering subevent type as SYN\_FLOOD.



**Note:** Users can also create their own network scenarios in Internet of Things and run simulation.