**Dos Attack in Internet of Things**

**Software Used:** NetSim Standard v10.2, Microsoft Visual Studio 2015

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.

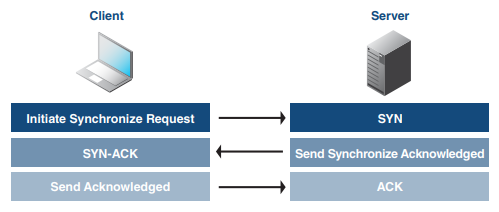
**Standard DDoS Attack types:**

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.

1. **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

1. **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 socket\_creation();

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

* static void send\_syn\_packet(PNETSIM\_SOCKET s);

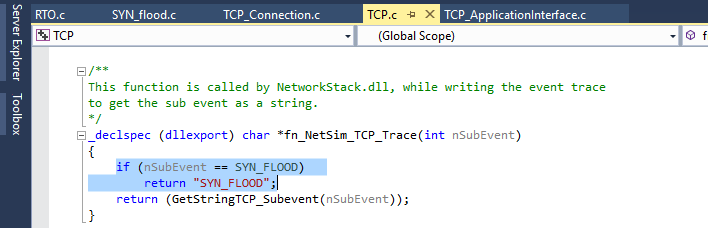
This function is used to create and send SYN packet to the network layer

* void syn\_flood();

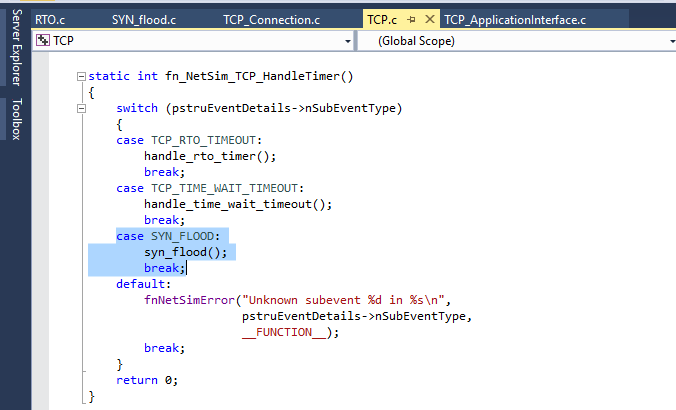
This function is used to check whether the socket is present or not and also adds a timer event called SYN\_FLOOD (triggers for every 1000µs)

**Code modifications done in NetSim:**

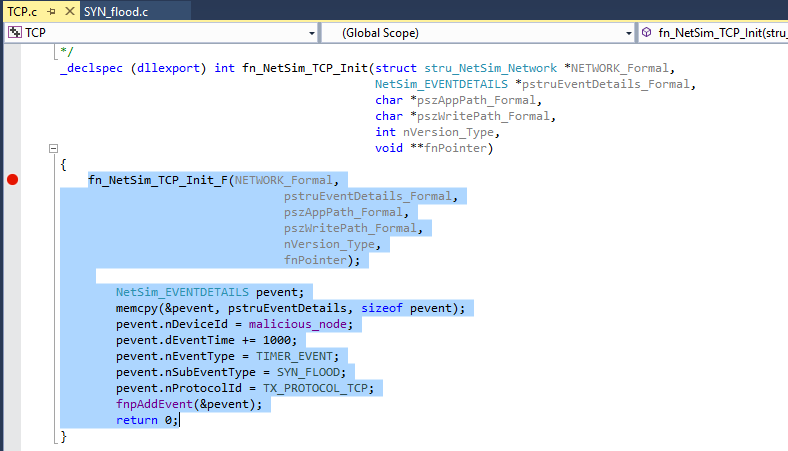
1. We have added the following lines of code in fn\_NetSim\_TCP\_Trace() function present in TCP.c file inside TCP project. This is used to add the SYN\_FLOOD sub-events in Event Trace file



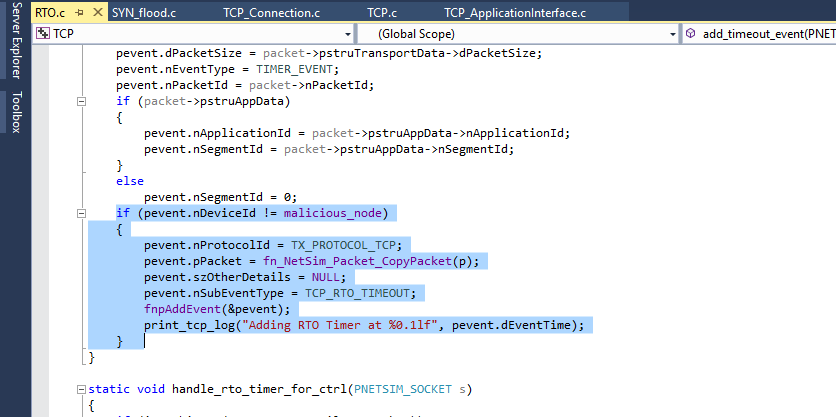
1. We have added the following lines of code in fn\_NetSim\_TCP\_HandleTimer() function present in TCP.c file inside TCP project. Used to add a TCP sub\_event called SYN\_FLOOD



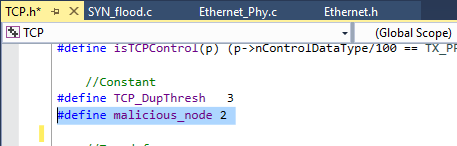
1. And modified the following lines of code in fn\_NetSim\_TCP\_Init() function resent in TCP.c inside TCP project



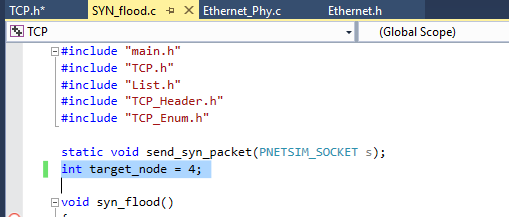
1. And modified the following lines of code in add\_timeout\_event() present in RTO.c file inside TCP project which avoids RTO timer for malicious nodes



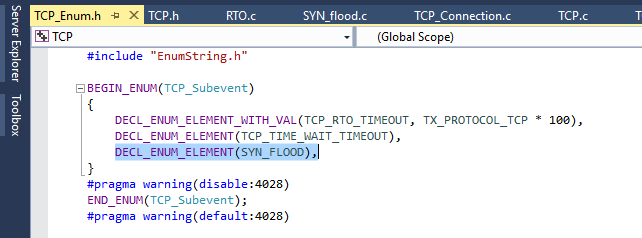
1. Users can give their own malicious ID in **TCP.h** file inside TCP project



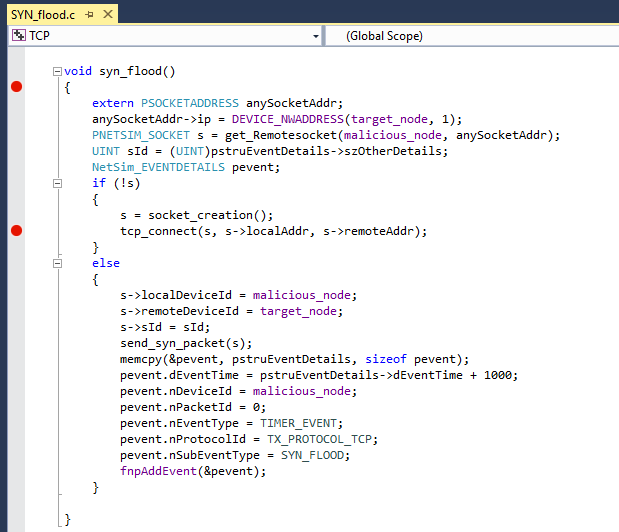
1. Users can give their own target ID in **SYN\_FLOOD.c** file inside TCP project

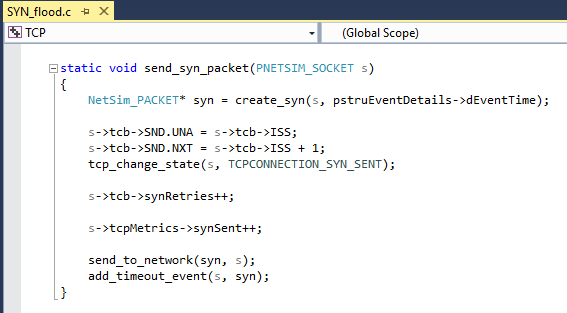


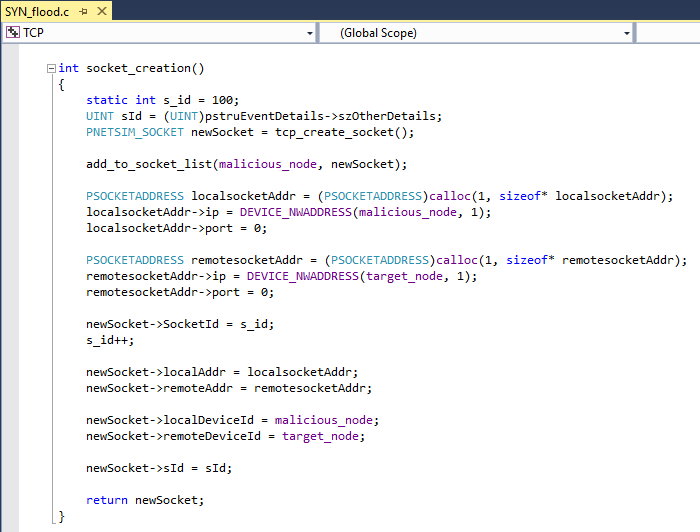
1. Added the following line in TCP\_Enum.h file inside TCP project to add a new TCP\_subevent called SYN\_FLOOD



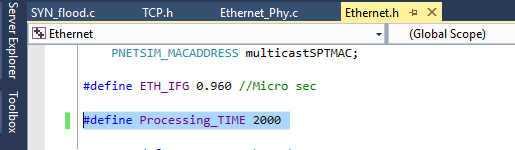
1. SYN\_FLOOD.c file contains the following functions



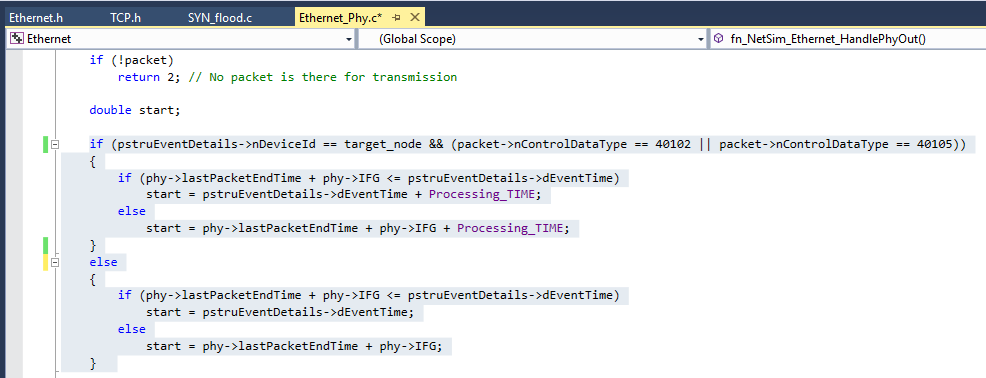




1. Added PROCESSING\_TIME macro in Ethernet.h file inside ETHERNET project

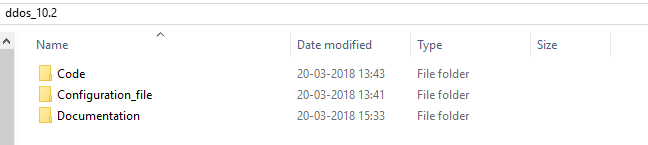


1. Modified the following lines of code in fn\_NetSim\_Ethernet\_HandlePhyOut() function present in Ethernet\_Phy.c file inside Ethernet project.

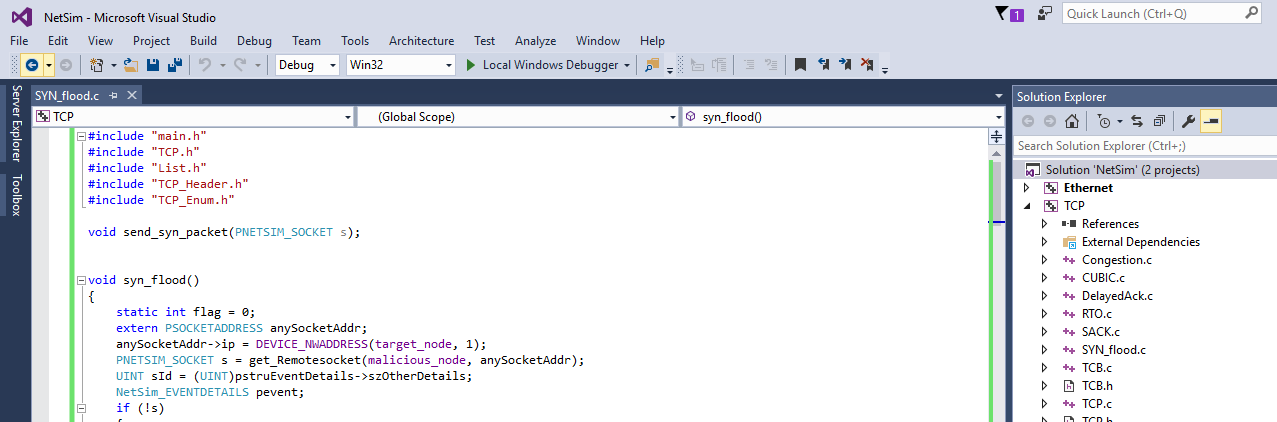


**Steps:**

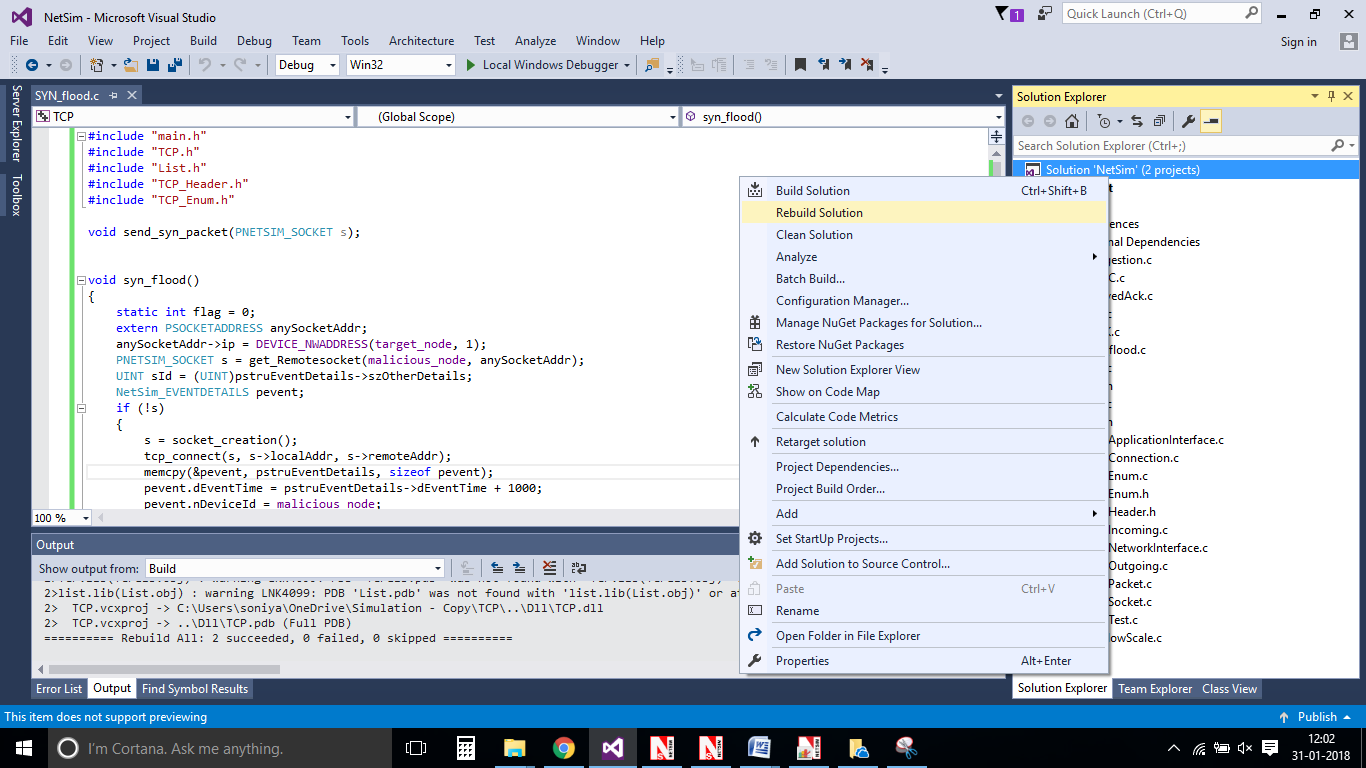
* After you unzip the file, the folder would look like



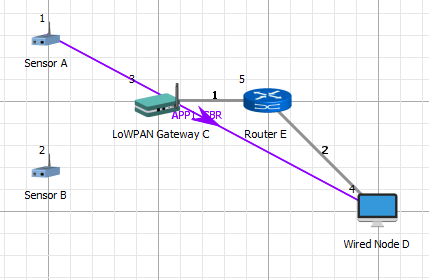
* Open Code folder and double click on the NetSim.sln file present to open the project in Visual Studio 2015.



* Right click on Solution in Solution Explorer and select rebuild solution.



* Upon rebuilding, **libTCP.dll** and **libEthernet.dll** will get created in the DLL folder
* Now copy the **libTCP.dll**, **libEthernet.dll** and **libTCP.pdb** from this DLL folder and paste it in NetSim bin folder present in the NetSim installation directory. The NetSim install directory would look something like < C:\Program Files (x86)\NetSim Standard\bin>
* Note that there exists **libTCP.dll** and **libEthernet.dll** in this bin folder. This is the default file being shipped with NetSim. The user is replacing this file with the newly built file.
* Therefore, take care to rename the original **libTCP.dll** and **libEthernet.dll** files, so that it isn’t lost. For example, you may rename it as **libTCP\_default.dll**, **libEthernet\_default.dll**.
* Run NetSim and open **Configuration.netsim** file present inside the Configuration\_File folder and run the simulation for 100 seconds.



Malicious Node

Delay = 10ms

* After simulation, open metrics window and observe the Application\_Throughput and SYN\_Sent packets under TCP\_Metrics. In this case the Application\_Throughput would be less and number of SYN\_Sent packets would be more since the malicious node would send the SYN requests continuously (which opens a network connection between client and server). During this time no other packets will be sent by the malicious node until TCP 3–way handshake process is completed
* Open Packet Trace and filter Control\_Packet\_Type to TCP\_SYN, Source\_ID to Node-2 and Receiver\_ID to Router-1. Then observe TCP\_SYN packets get created for every 1000µs in Node-2 (malicious node)
* Similarly open Event Trace and filter Subevent\_Type to SYN\_FLOOD, observe the event gets triggered for every 1000µs in node-2 under TCP protocol
* Simulate the same scenario with default dlls and compare the Application Throughput (high) and SYN\_Sent packets (less)

|  |  |  |
| --- | --- | --- |
| **Simulation time = 100s** | | |
| **PROCESSING\_TIME (µs)** | **Packets received** | **Throughput (Mbps)** |
| 2000 | 394 | 0.003069 |
| 3000 | 258 | 0.002010 |
| 4000 | 121 | 0.000942 |
| **Default** | | |
| 0 | 8896 | 0.069270 |

From the above table, users can observe that as processing time increases, the no. of packets received at the target node decreases since the processing time of TCP\_SYN\_ACK and TCP\_ACK increases

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