DIO Suppression Attack in RPL

Software: NetSim Standard v 14.1, Visual Studio 2022, MATLAB R2019 or higher

Project Download Link:

https://github.com/NetSim-TETCOS/DIO-Suppression-attack-in-IoT-RPL-v14.1/archive/refs/heads/main.zip

Follow the instructions specified in the following link to download and setup the project in NetSim:

https://support.tetcos.com/en/support/solutions/articles/14000128666-downloading-and-setting-up-netsim-file-exchange-projects

1 Introduction

In DIO Suppression Attack, a malicious node broadcast DIO messages to legitimate nodes. If malicious node transmits repeatedly a DIO message that is considered consistent by the receiving nodes. If the nodes receive enough consistent DIOs, they will suppress their own DIO transmission. Since DIO messages are exploited to discover neighbours and the network topology, their continuous suppression can cause some nodes to remain hidden and some routes to remain undiscovered. DIO Suppression attacks affect the performance of IoT networks protocols such as RPL protocol.

2 Implementation in RPL (for 1 sink)

In RPL the transmitter broadcasts the DIO during DODAG formation.

- The receiver on receiving the DIO from the transmitter updates its parent list, sibling list, rank and sends a DAO message with route information.
- Malicious node upon receiving the DIO message it transmits DIO message repeatedly to legitimate nodes.
- The legitimate nodes on listening to the malicious node DIO message they will suppress their own DIO transmission.
- The continuous suppression can cause some nodes to remain hidden and some routes to remain undiscovered.

The DIO.c file contains the following functions

1. **fn_NetSim_RPL_MaliciousNode();** //This function is used to identify whether a current device is malicious or not in-order to establish malicious behaviour.
2. **fn_NetSim_RPL_MaliciousNodeReplay();** //This function is used by the malicious node to transmit DIO message repeatedly to legitimate nodes.

You can set any device as malicious, and you can have more than one malicious node in a scenario. Device id's of malicious nodes can be set inside the fn_NetSim_RPL_MaliciousNode() function.

3 With-DIO Suppression Attack

1. The Workspace **DIO_Suppression_Attack_using_RPL_v14** comes with a sample network configuration that is already saved.
2. Go to Your Work option in NetSim Home Screen and open the saved example, **DIO_Suppression_Attack_Example**. The network scenario and the settings done is explained below:

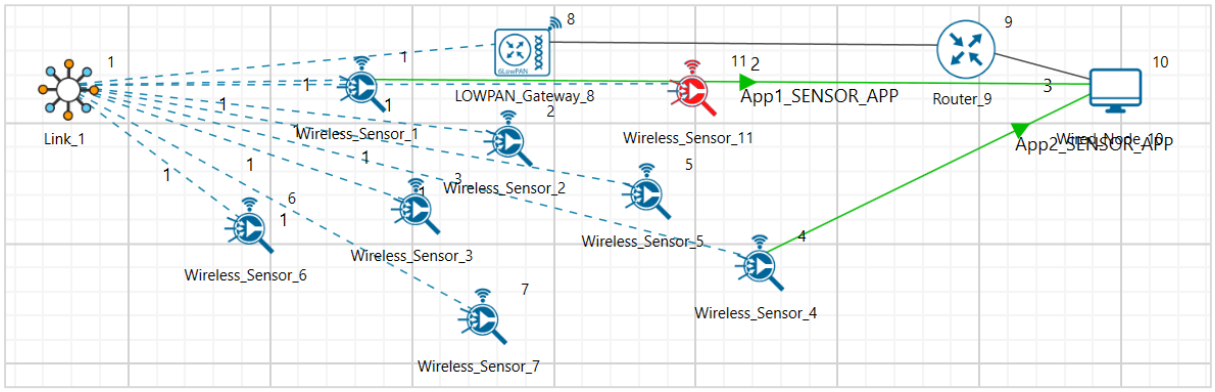

Figure 1: Network scenario showing a DIO Suppression attack in IoT RPL project. Includes 8 sensors (one malicious), a wired node, a router, and a LoWPAN gateway.

Note: In above screenshot Red color Wireless_Sensor_Node_11 is a malicious node.

Application 1	
Source	DEVICE ID 1
Destination	DEVICE ID 10
Packet Size	50 Bytes
Inter Arrival Time	1000000 µs
Packet Size	
Application 2	
Source	DEVICE ID 4
Destination	DEVICE ID 10
Packet Size	50 Bytes
Inter Arrival Time	1000000 µs
Link Properties (Link 1)	
Channel Characteristics	Pathloss Only
Pathloss Model	LOG DISTANCE
Pathloss Exponent	2.5

Table 1: Application and link properties.

3. Go to LOWPAN Gateway Properties >Network Layer > RPL >DIO Redundancy Constant > 6.

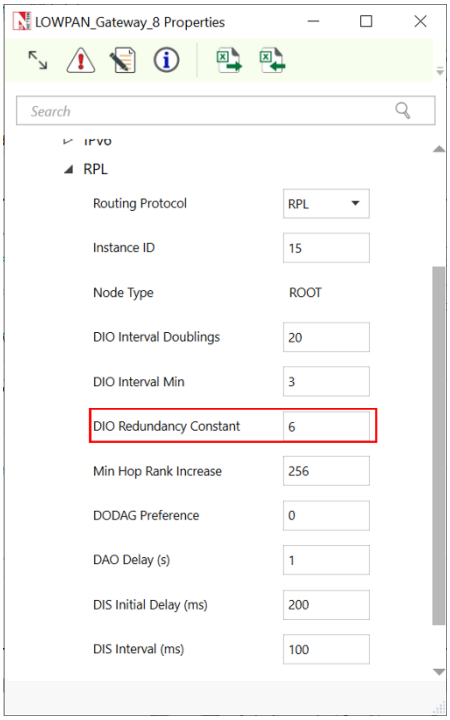

Figure 2: Setting the DIO Redundancy constant value to 6.

- The DIO suppression attack requires the adversary to transmit only DIO Redundancy Constant(k) messages at each Trickle period.
- DIO Redundancy Constant(k) acts as suppression threshold, as we set 6, the malicious node will replay the DIO message 6 times to the neighbouring nodes. After replaying the DIO message, the neighbouring nodes will suppress their own DIO transmission.

4. Run simulation and press any key to continue
5. It will open MatlabInterface.exe console window. You will observe that as the simulation starts in NetSim, MATLAB gets initialized and the DODAG plot associated with the IoT network is plotted during runtime.

4 Without-DIO Suppression Attack

- To run simulations without DIO Suppression attack, open the Source Code click on Your Work > Source Code > Open Code
- In RPL project, open RPL.h and set the value of the variable DIO_ATTACK_ENABLE to 0 instead of 1 as shown below

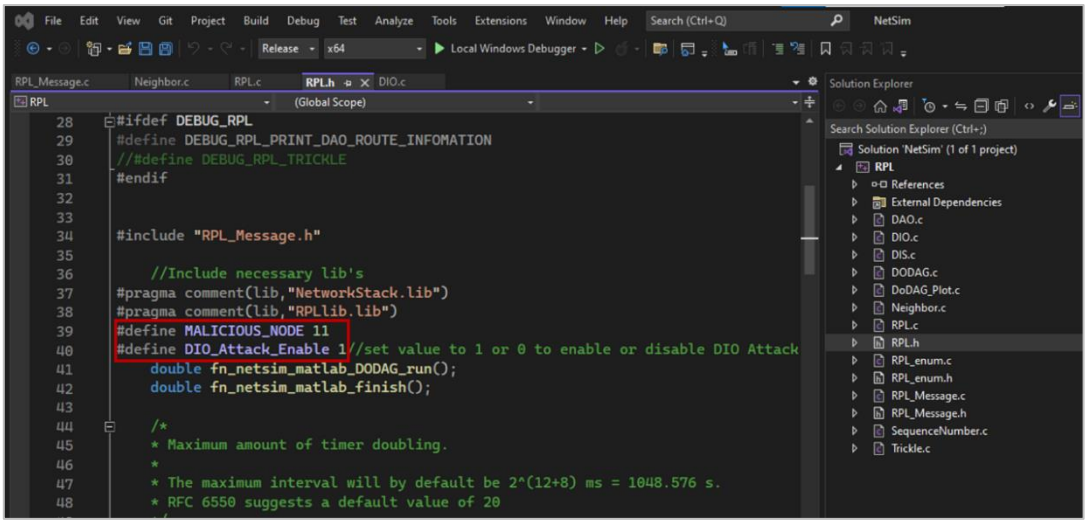

Figure 3: Defining 0 or 1 in the RPL.h file to set the attack status.

- Rebuild the RPL Project and run Simulation.

5 Results and discussion

Case 1: With DIO Suppression Attack

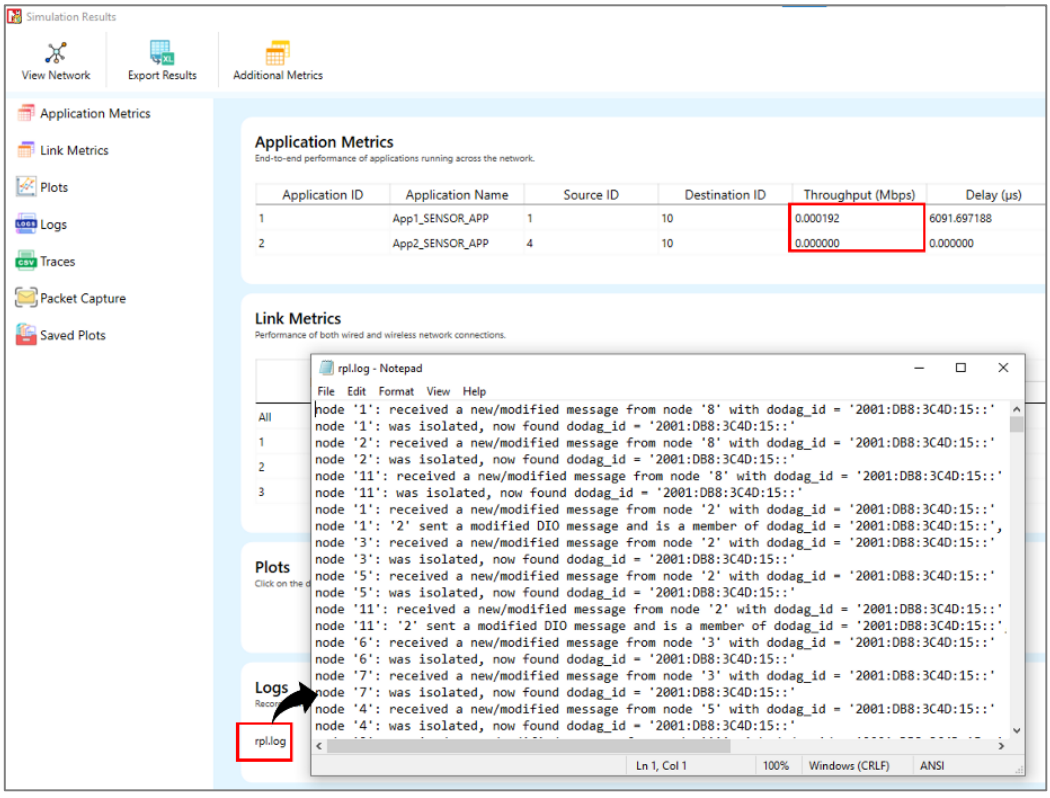

Figure 4: Sensor 4 suppressing its own DIO messages results in zero throughput for Application 2, due to a malicious node continuously sending DIO messages to Sensors 4. The RPL log shows information about the DODAG with the DIO suppression attack

The result dashboard provides access to the RPL log file, Where we can see the detailed information about the DODAG formation process.

DODAG Formation Graph:

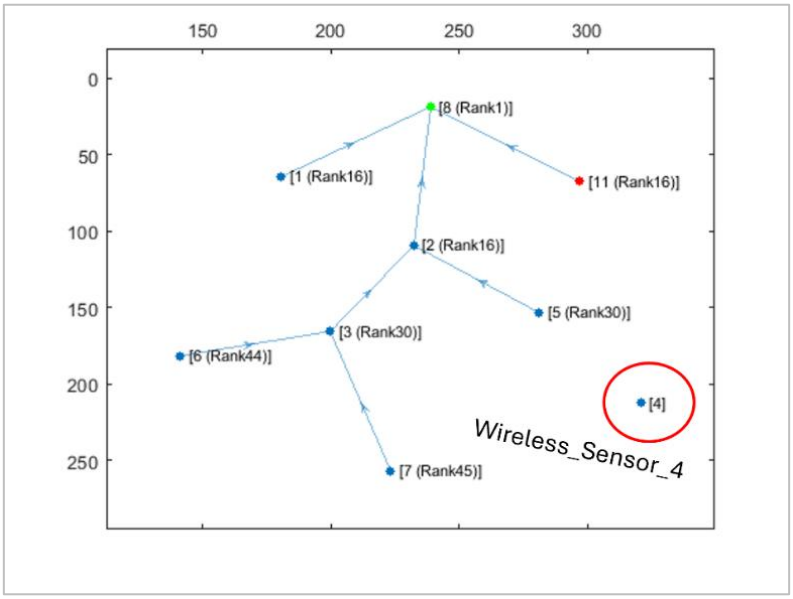

Figure 5: The DODAG shows that Sensor 4 is suppressed, so it has not joined the DODAG.

When root node (LowPan_Gateway) broadcast the DIO message all nodes that are present in the communication range will also broadcast their own DIO messages but when malicious node broadcasts the DIO message, it will repeatedly transmit the DIO message to the neighbour nodes such that it prevents the DIO messages from other neighbour nodes reaching them.

So, it degrades the routing information, and some nodes remain hidden in the network.

We can observe from the above graph that **Wireless_Sensor_4** is not part of DODAG formation as it is not discovered and remain hidden in the network.

Case 2: Without DIO Suppression Attack

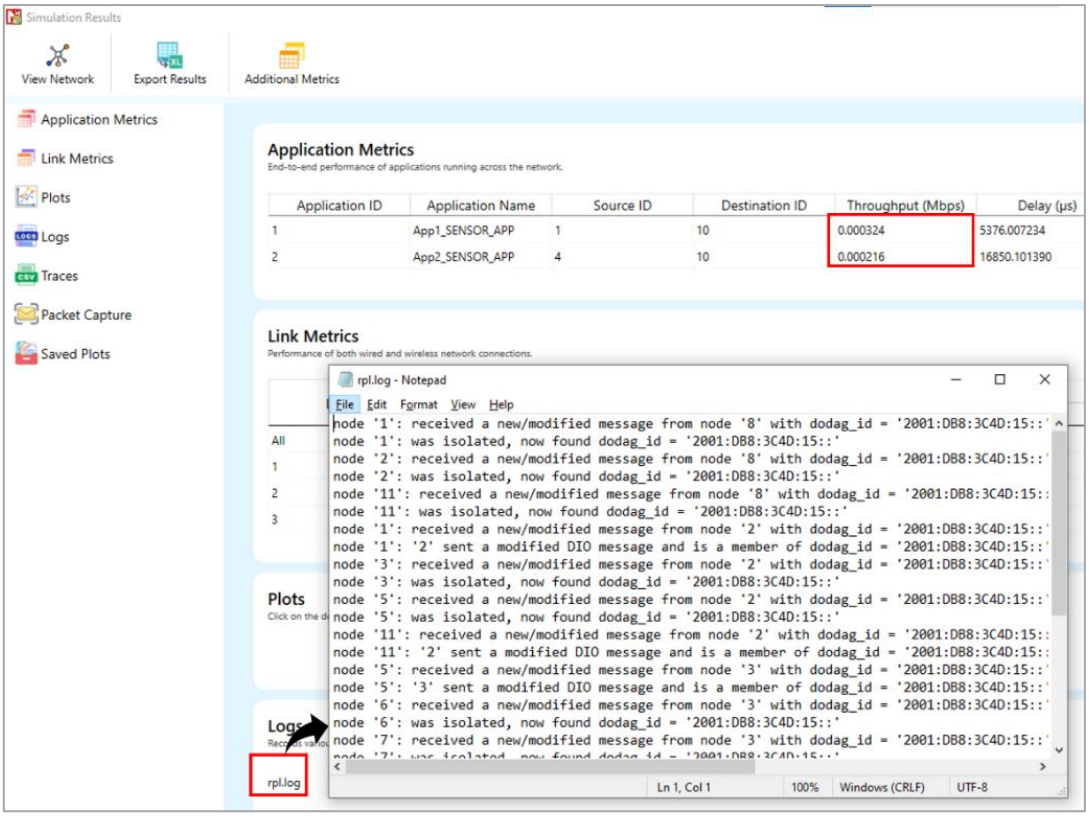

Figure 6: Applications that is having throughput and the RPL log that shows the DODAG information without DIO suppression attack.

DODAG Formation Graph:

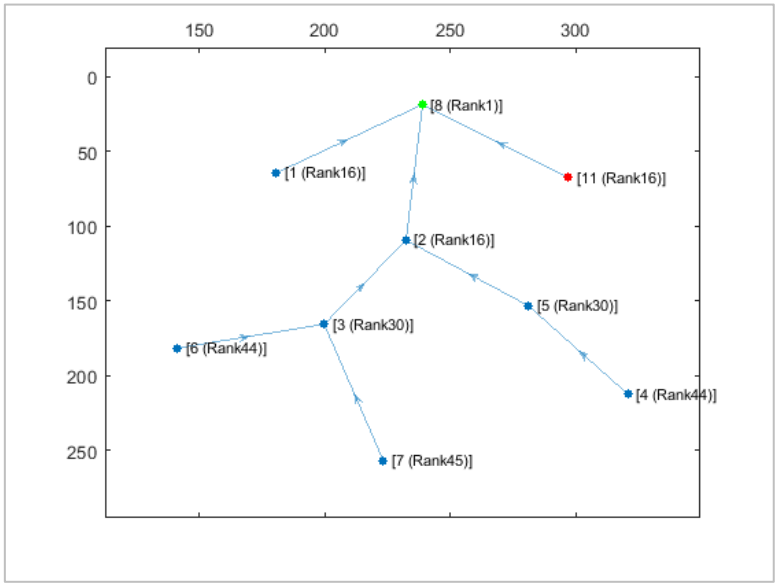

Figure 7: The DODAG shows the formation of all sensors without any attack.

We can observe from the graph that when the DIO Attack is disabled, The DODAG formation will happen with all the nodes being a part of it

With the DIO Suppression Attack disabled the performance of the network will increase in comparison with **Case 1** i.e., **DIO Attack Enabled**.

Case 3: With DIO Suppression Attack (DIO-Redundancy Constant = 7)

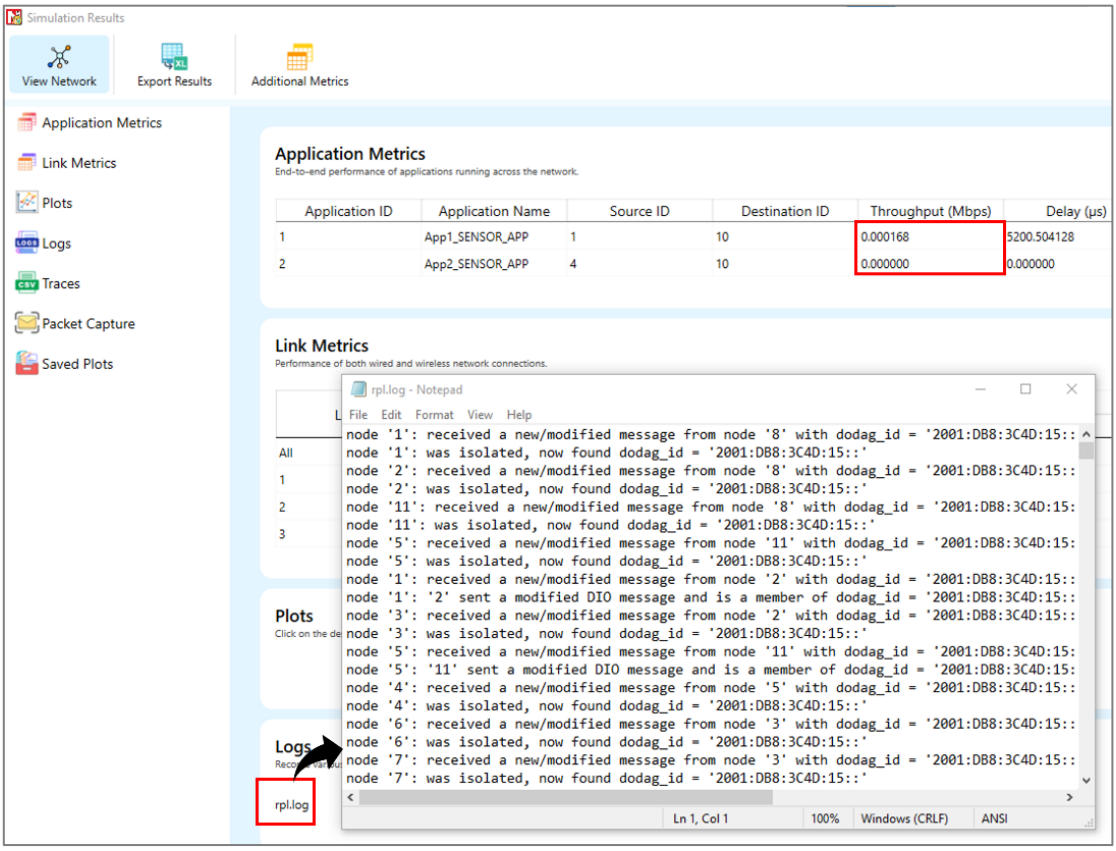

Figure 8: Sensor 4 and Sensor 5 suppressing their own DIO messages result in zero throughput for Application 2 with DIO-Redundancy Constant = 7, due to a malicious node continuously sending DIO messages to Sensors 4 and 5. The RPL log shows information about the DODAG with the DIO suppression attack.

DODAG Formation Graph:

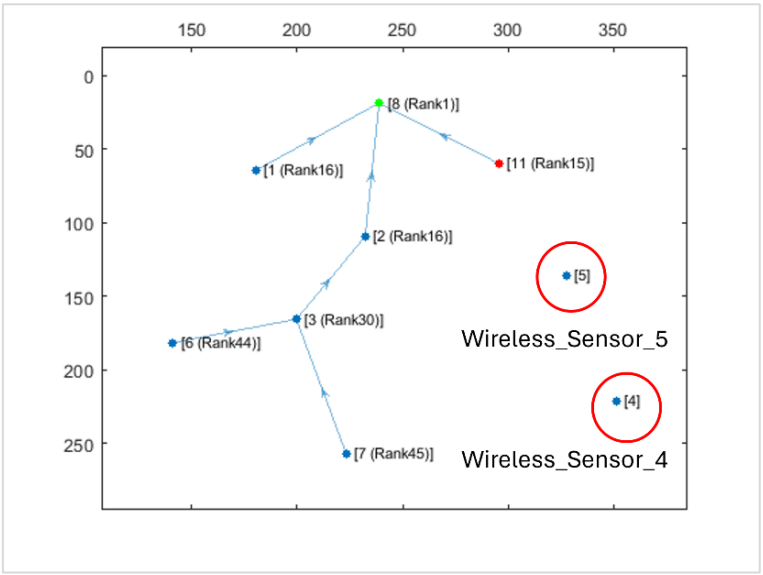

Figure 9: The DODAG shows that Sensor 4 and sensor 5 are suppressed, so it has not joined the DODAG.

Note : To set DIO-Redundancy Constant to 7 you can refer Figure 2.

We can observe from the graph that Wireless_Sensor_5 and Wireless_Sensor_Node_4 is not part of DODAG formation as they are not discovered and remain hidden in the network. And from the Simulation result dashboard that when we enable DIO Suppression attack in that situation some nodes are hidden due to which our throughput is getting decreased.

DIO Suppression severely degrade the performance of Low Power and Lossy Network (LLNs) because of the repeatedly transmitting the DIO message by the malicious node to neighbouring nodes.

The DIO suppression attack, an attack that induces victim nodes to suppress the transmission of DIO messages. This causes a general degradation of the routes quality that can lead, eventually, to network partitions.

With the DIO Redundancy Constant set to 7 the Suppression is more than that of the DIO Redundancy constant 6.

Appendix: NetSim source code modifications and steps:

1. Add the following MATLAB install directory path in the Environment PATH variable <MATLAB_INSTALL_DIRECTORY>\bin\win64
   For eg: C:\Program Files\MATLAB\R2021b\bin\win64

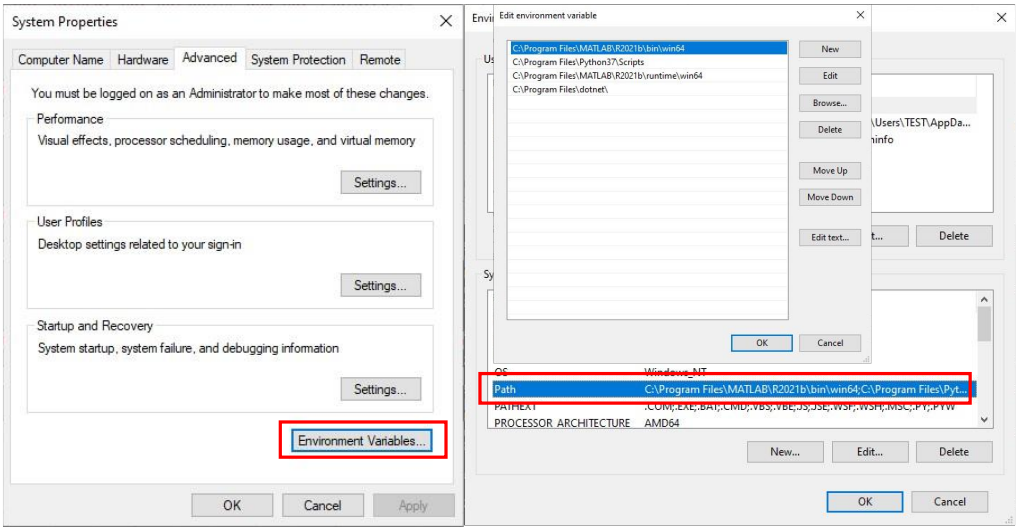

Figure 10: Setting the environment variable path to run the matlab.

Note: If the machine has more than one MATLAB installed, the directory for the target platform must be ahead of any other MATLAB directory (for instance, when compiling a 64-bit application, the directory in the MATLAB 64-bit installation must be the first one on the PATH).

2. Open Command prompt as admin and execute the command “matlab -regserver”. This will register MATLAB as a COM automation server and is required for NetSim to start MATLAB automation server during runtime.

3. Go to home page, Click on Your work>Source Code and click on the Open code button.
4. Set malicious node id in RPL.h file.

   #define MALICIOUS NODE 9

C Code that is highlighted in red colour is added to RPL_Message.c file

```
void rpl_process_ctrl_msg()
{
        switch (pstruEventDetails->pPacket->nControlDataType % 100)
        {
        case DODAG_Information_Object:
#if DIO_Attack_Enable
                if (fn_NetSim_RPL_MaliciousNode(pstruEventDetails)) {
                        rpl_process_dio_msg();
                        Fn_NetSim_RPL_MaliciousNodeReplay(pstruEventDetails);
                }
                else
                        rpl_process_dio_msg();
#else
                rpl_process_dio_msg();
#endif
                break;
        case Destination_Advertisement_Object:
                rpl_process_dao_msg();
                break;
        case DODAG_Information_Solicitation:
                rpl_process_dis_msg();
                break;
        default:
                fnNetSimError("Unknown rpl ctrl msg %d in %s",
                                        pstruEventDetails->pPacket->nControlDataType,
                                        __FUNCTION__);
                break;
        }
```

5. Now right click on Solution explorer and select Rebuild.
   a. Upon rebuilding, libRPL.dll will automatically get replaced in the respective bin folders of the current workspace.
6. Then run the Example scenario which came along with the Workspace.