Sink Hole Attack with DODAG Visualization in IOT

Software Recommended: NetSim Standard v12.0 (32-bit/ 64-bit), Visual Studio 2017/2019, MATLAB (32/64 bit)

In sinkhole Attack, a compromised node or malicious node advertises fake rank information to form the fake routes. After receiving the message packet it drop the packet information. Sinkhole attacks affect the performance of IoT networks protocols such as RPL protocol.

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 does not update the rank instead it always advertises a fake rank.
- The other node on listening to the malicious node DIO message the update their rank according to the fake rank.
- After the formation of DODAG, if the node that is transmitting the packet has malicious node as the preferred parent, transmits the packet to it but the malicious node instead of transmitting the packet to its parent, it simply drops the packet resulting in zero throughput.

A file Malicious.c is added to the RPL project.
The 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_MaliciousRank( )`
   This function is used to give a fake rank to the malicious node.
3. `rpl_drop_msg( )`
   This function is used to drop the packet by the malicious node if it enters into its network layer.

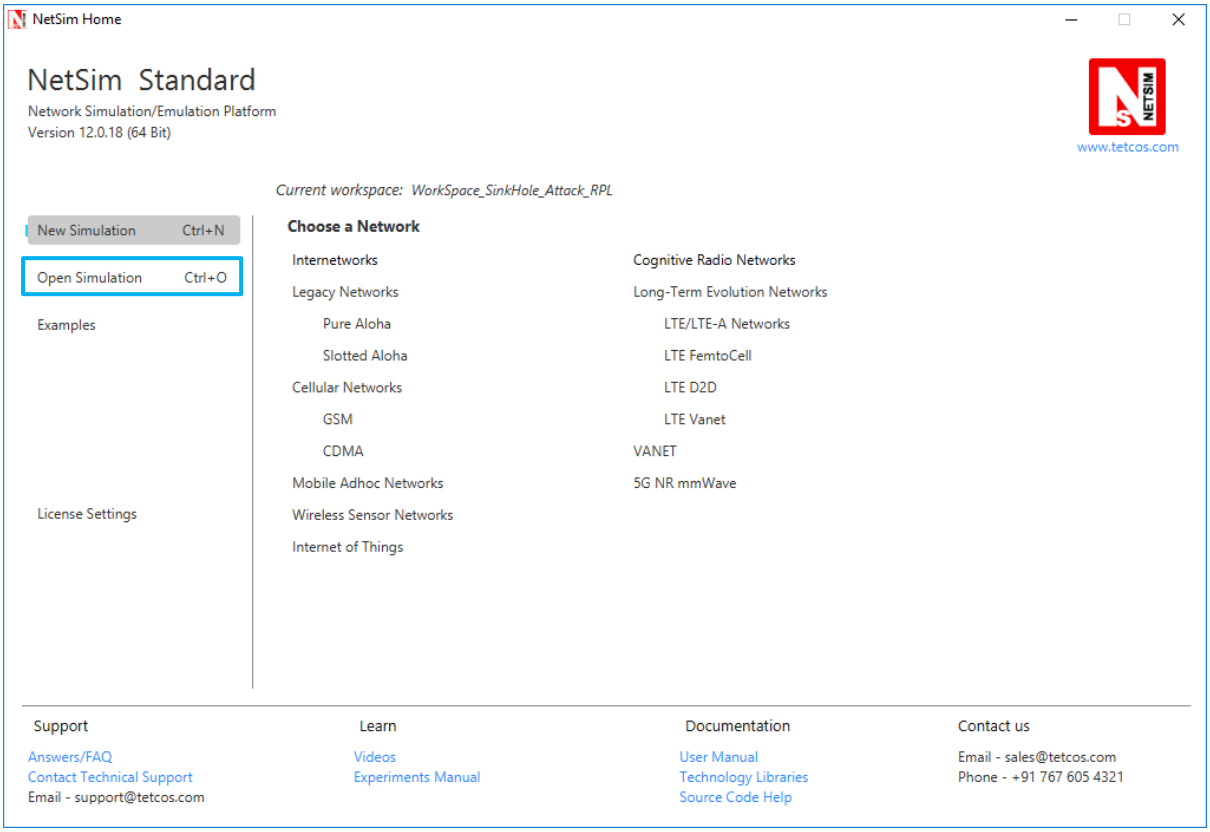
Sink Hole attack – The malicious node advertises the fake rank.
`fn_NetSim_RPL_MaliciousRank( )` is the sink hole attack function.

Black Hole attack – The malicious node drops the packet.
`rpl_drop_msg()` is the black hole attack function

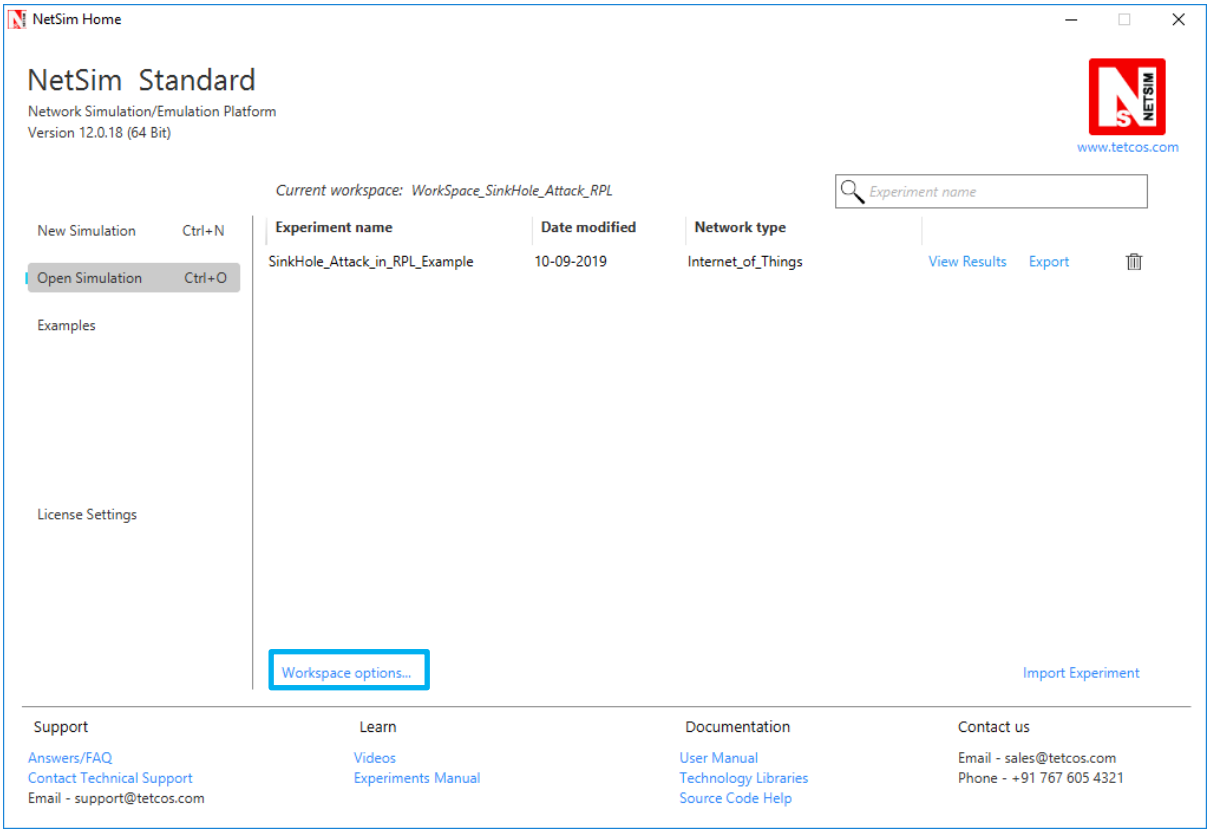
You can set any device as malicious and you can have more than one malicious node in a scenario.
Device id's of malicious nodes can be set inside the fn_NetSim_RPL_MaliciousNode() function.

Steps:

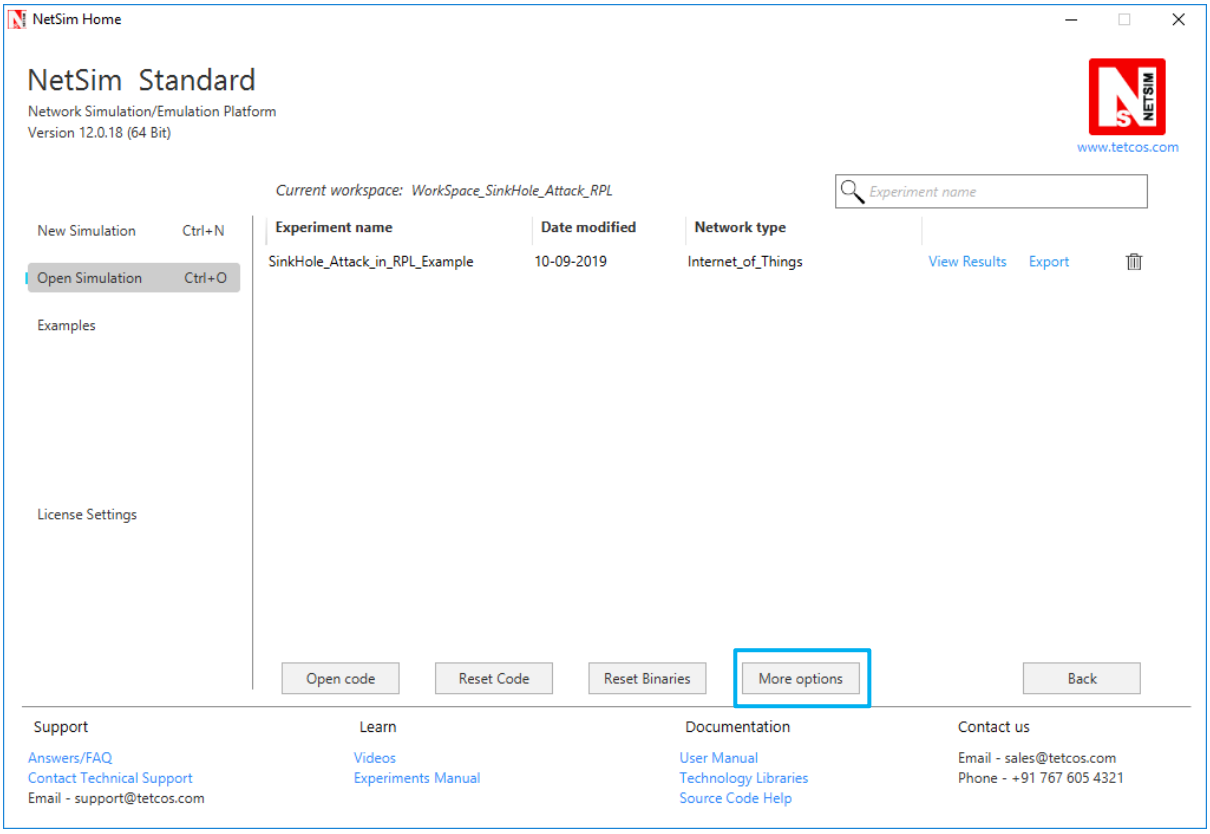
1. After you unzip the downloaded project folder, Open NetSim Home Page click on **Open Simulation** option,



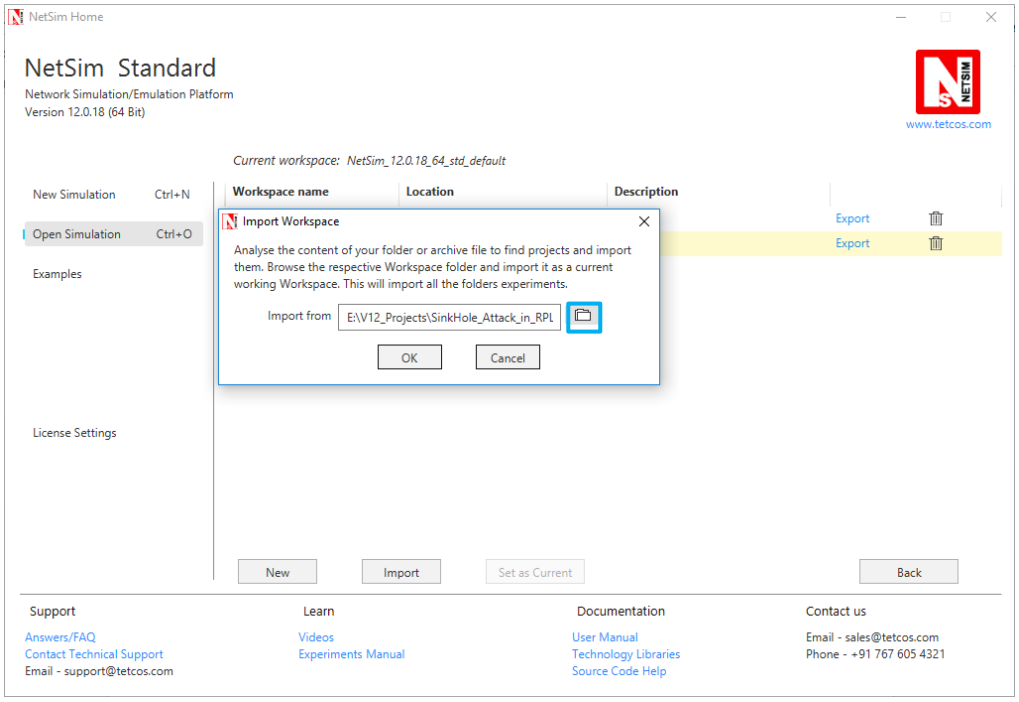
2. Click on **Workspace options**



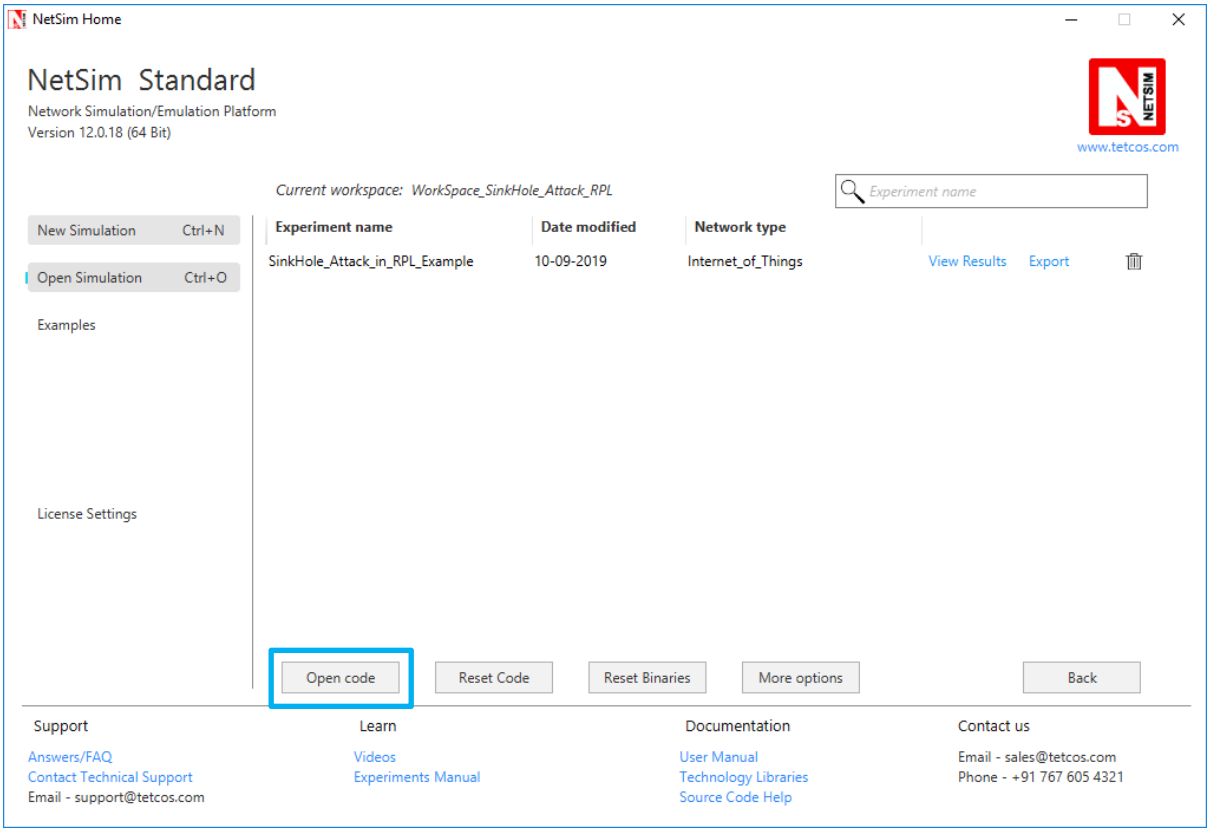
3. Click on **More Options,**



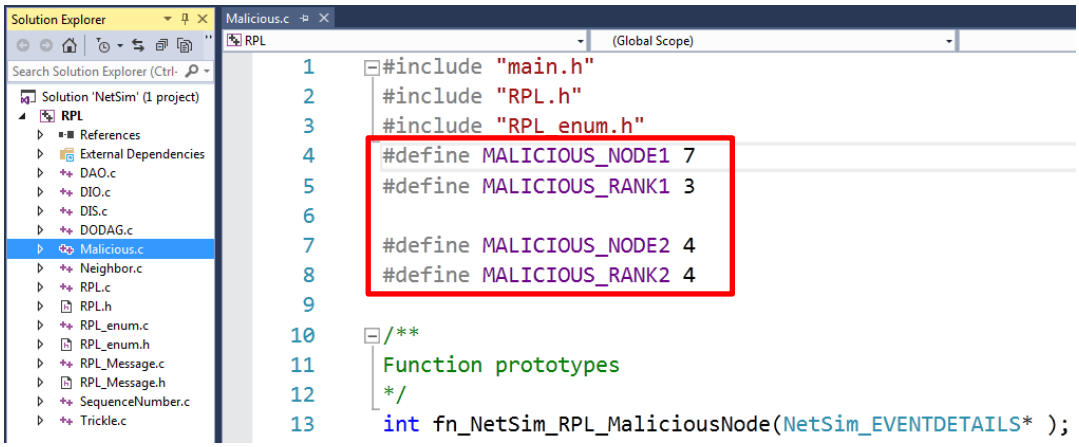
4. Click on **Import**, browse the extracted folder path and go into the WorkSpace_SinkHole_Attack_RPL directory. Click on Select folder button and then on **OK.**



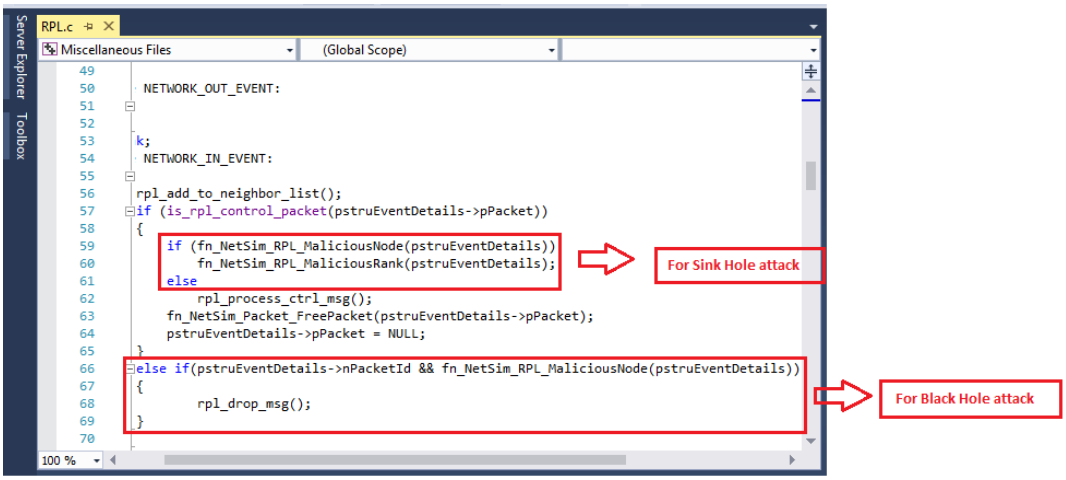
5. Go to home page, Click on **Open Simulation → Workspace options → Open code**



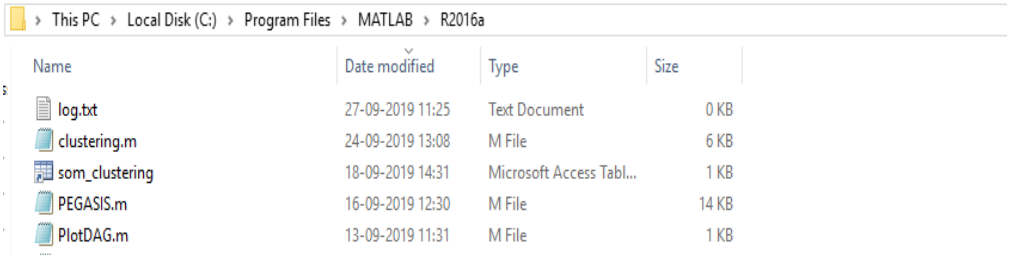
6. Set malicious node id and the fake Rank.



7. Add the code that is highlighted in RPL.c file

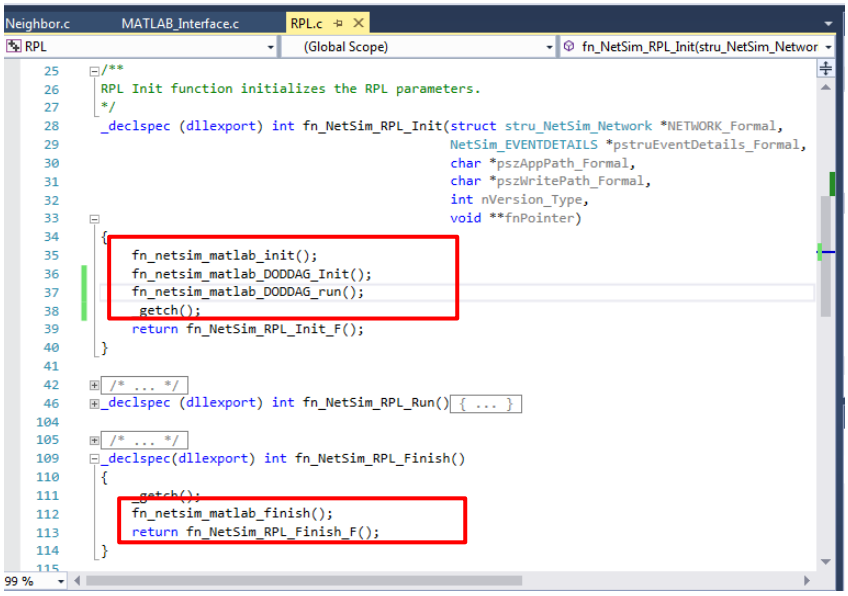


8. Place **PlotDAG.m** file inside the MATLAB root directory. For Eg: **"<MATLAB installed path>\MATLAB\R2019b", (Note: PlotDAG.m** is provided inside the MATLAB_Code directory)



9. Following modifications were done to the RPL project for this implementation:

a. Open RPL.c file and add **fn_netsim_matlab_init()**, **fn_netsim_matlab_DODDAG_run()** and **fn_netsim_matlab_DODDAG_Init()** inside **fn_NetSim_RPL_Init()** and **fn_netsim_matlab_finish()** inside **fn_NetSim_RPL_Finish ()**.



b. Add definitions of the following functions inside **RPL.h** file

```
a. double fn_netsim_matlab_init();
b. double fn_netsim_matlab_DODDAG_Init();
c. double fn_netsim_matlab_DODDAG_run();
```

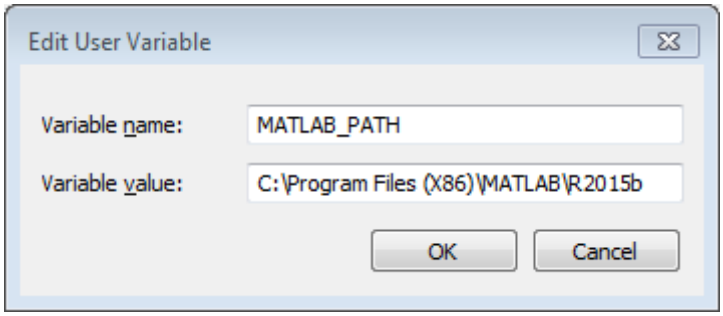
```
c. double fn_netsim_matlab_finish()
```

```
#define _NETSIM_RPL_H_
#ifdef __cplusplus
extern "C" {
#endif

    //Log settings
#define DEBUG_RPL
#ifdef DEBUG_RPL
//#define DEBUG_RPL_PRINT_IP_TABLE
#define DEBUG_RPL_PRINT_DAO_ROUTE_INFOMATION
//#define DEBUG_RPL_TRICKLE
#endif


#include "RPL_Message.h"

    //Include necessary lib's
#pragma comment(lib,"NetworkStack.lib")
#pragma comment(lib,"RPLlib.lib")
    double fn_netsim_matlab_init();
    double fn_netsim_matlab_DODDAG_Init();
    double fn_netsim_matlab_DODDAG_run();
    double fn_netsim_matlab_finish();

    /*
    * Maximum amount of timer doubling.
    *
```

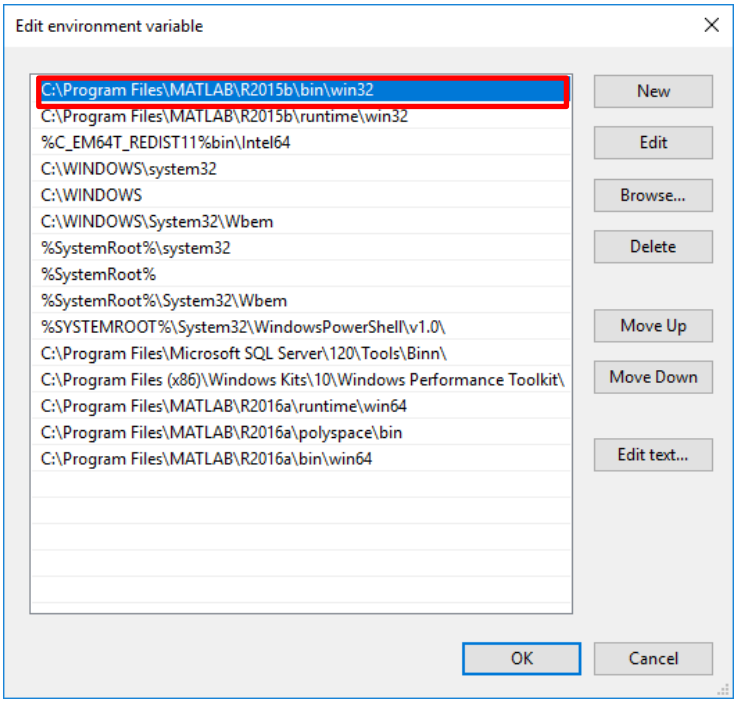
d. Go to the **Neighbor.c** file. Inside Function **void choose_parents_and_siblings(NETSIM_ID d)** add **fn_netsim_matlab_DODDAG_run()** below **rpl_add_route_to_parent()**

```
    dodag->pref_parent = rpl->neighbor_list[best_rank_index];
    if (dodag->pref_parent->nodeId != old_pref_parent)
    {
        NetSim_EVENTDETAILS pevent;
        memset(&pevent, 0, sizeof pevent);
        pevent.dEventTime = pstruEventDetails->dEventTime;
        pevent.nDeviceId = d;
        pevent.nDeviceType = DEVICE_TYPE(d);
        pevent.nEventType = TIMER_EVENT;
        pevent.nProtocolId = NW_PROTOCOL_RPL;
        pevent.nSubEventType = RPL_NEW_PREF_PARENT;
        fnpAddEvent(&pevent);
    }
    rpl_add_route_to_parent(d, dodag->pref_parent->nodeId);

    fn_netsim_matlab_DODDAG_run();

    for (i = 0; i < rpl->neighbor_count; i++)
    {
        PRPL_NEIGHBOR neighbor = rpl->neighbor_list[i];

        if (matching_ranks[i] >= INFINITE_RANK)
        {
            if (neighbor->lastDIOMSG != NULL)
            { /* forget messages from other DODAG iterations */
                rpl_dio_pdu_free(neighbor->lastDIOMSG);
                neighbor->lastDIOMSG = NULL;
```

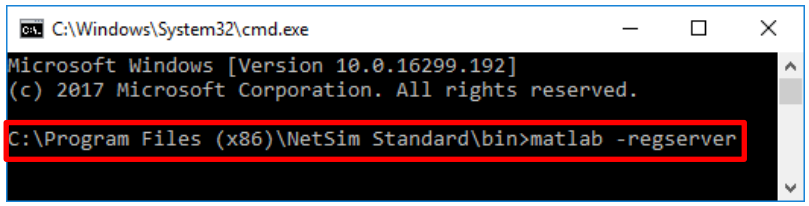
10. Create a user variable with the name of MATLAB_PATH and provide the path of the installation directory of user's respective MATLAB version.



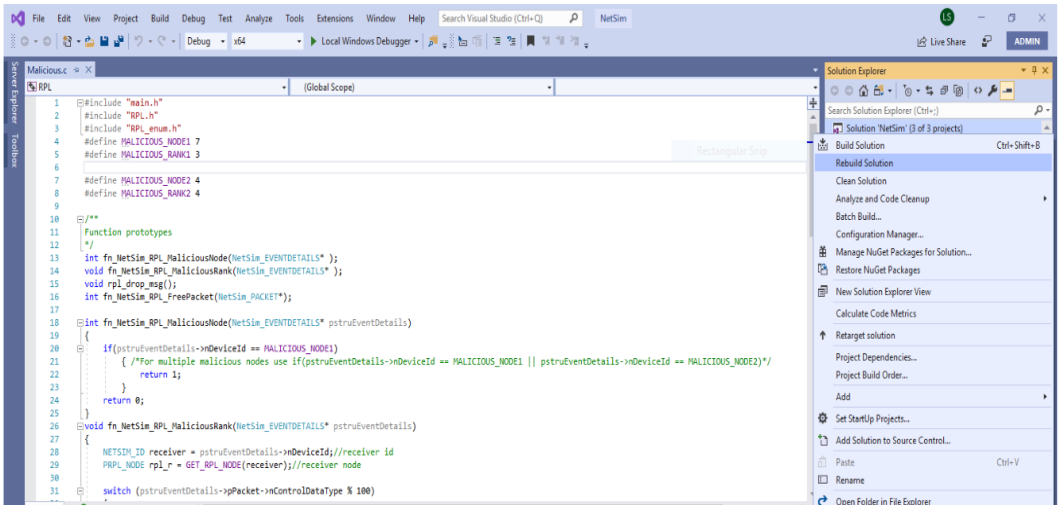
11. Make sure that the following directory is in the PATH(Environment variable)
<Path where MATLAB is installed>\bin\win32



(**Note:** To run this code 32- bit version of MATLAB must be installed in your system. If you are interfacing for the first time then open command window and go to the **<NetSim installed directory>\bin** and type **matlab -regserver**)



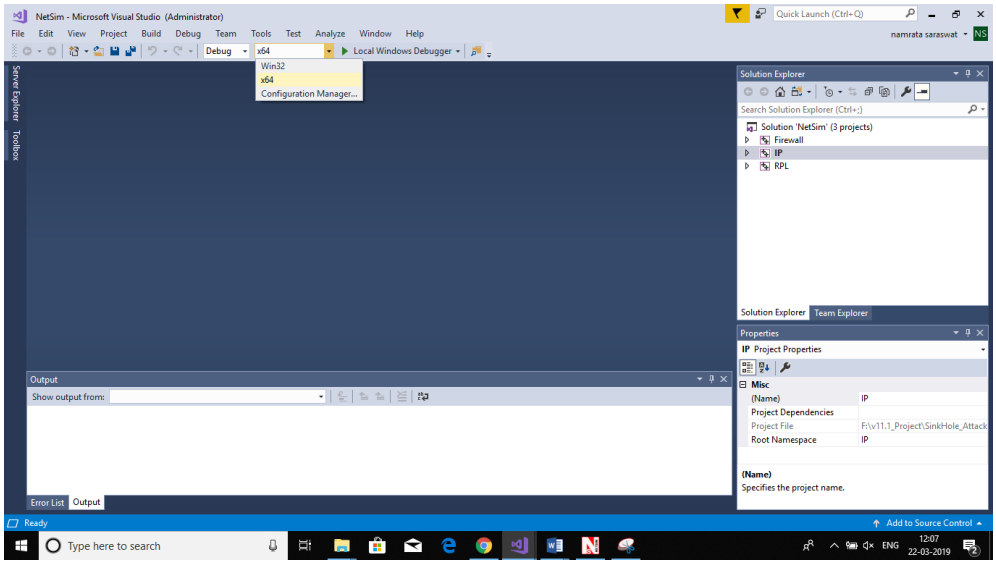
12. Now right click on Solution explorer and select Rebuild.



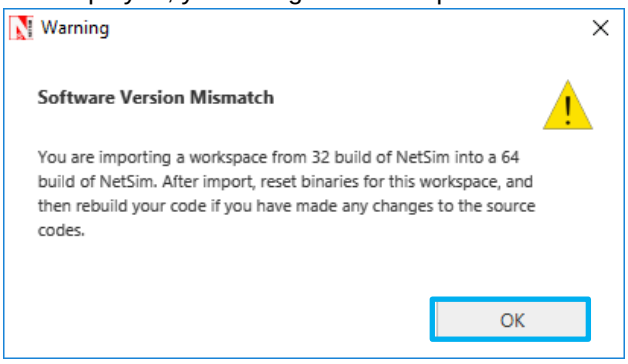
13. Upon rebuilding, **libRPL.dll, libIP.dll,** and **Firewall.dll** will automatically get replaced in the respective bin folders of the current workspace

Note:

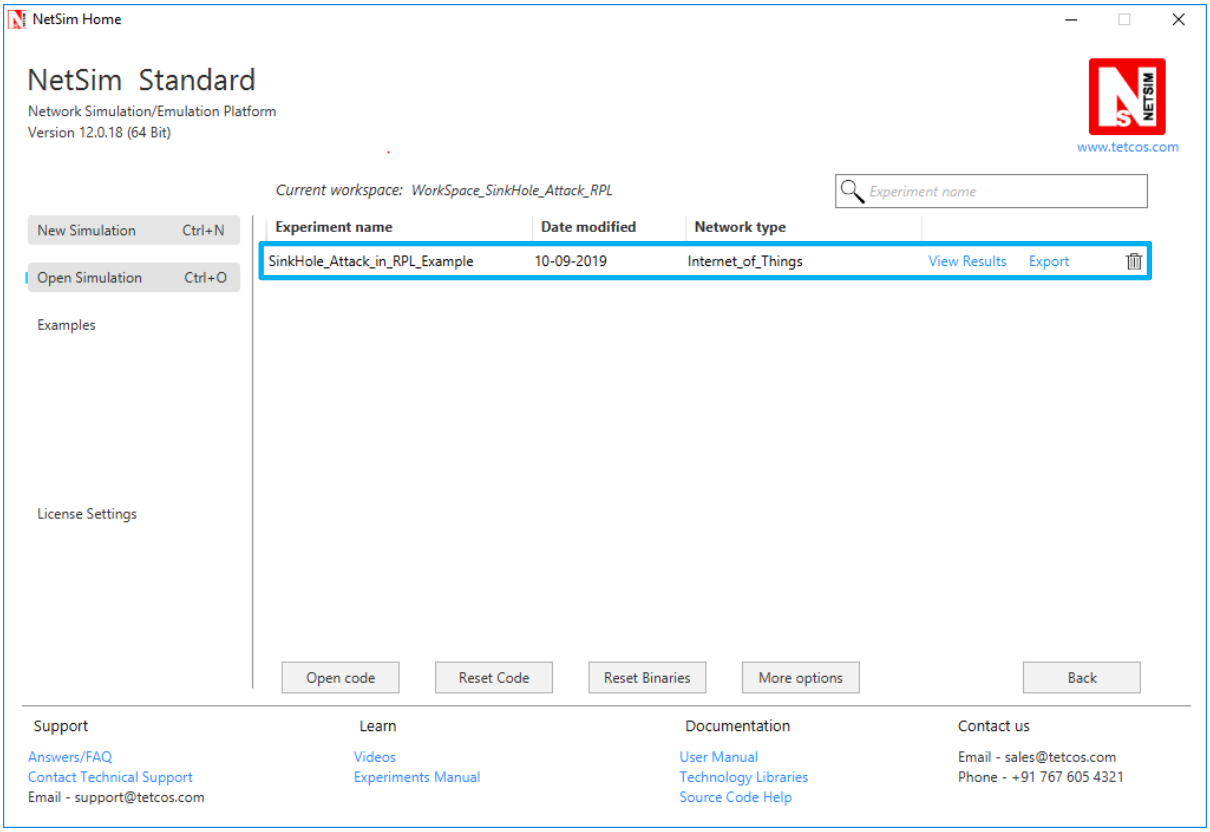
1. Based on whether you are using NetSim 32 bit or 64 bit setup you can configure Visual studio to build 32 bit or 64 bit Dll files respectively as shown below:



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



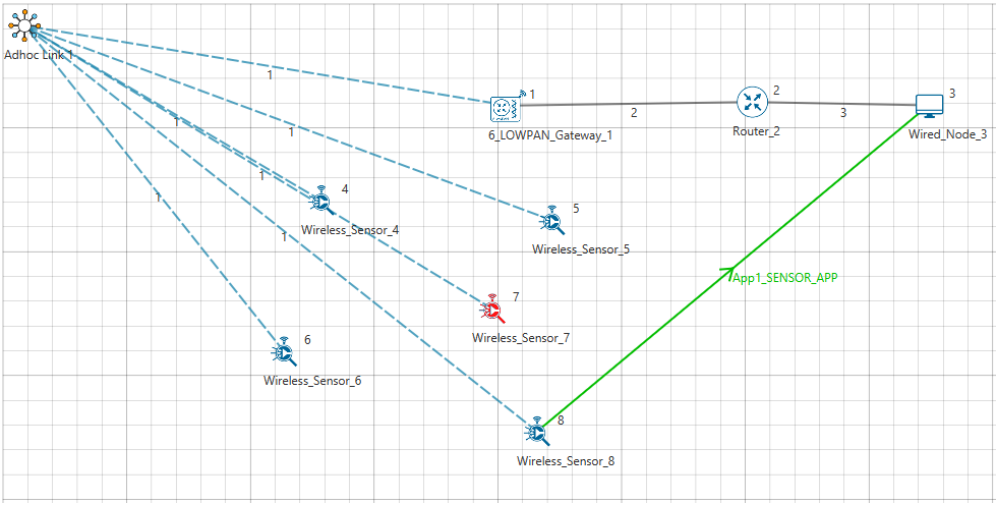
14. Go to NetSim home page, click on **Open Simulation**, Click on **SinkHole_Attack_in_RPL_Example**.



Settings that were done to create the network scenario for SinkHole Attack:

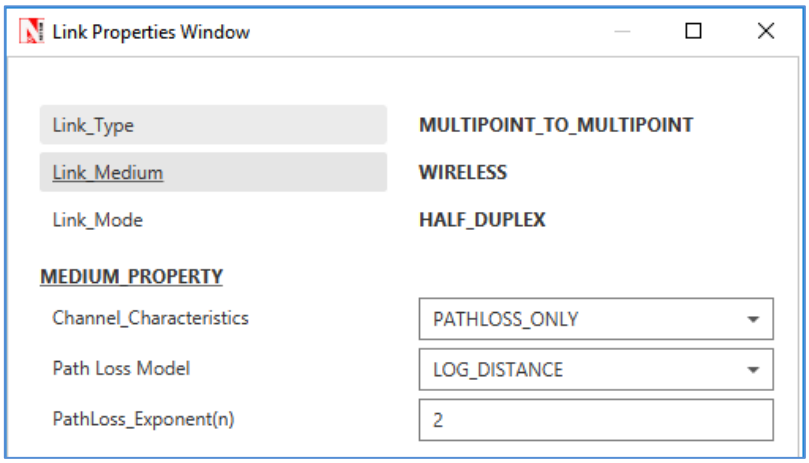
1. Create a network scenario in **IoT (Internet of Things) with UDP running in the Transport Layer and RPL in Network Layer.**

2. For example, you can create a scenario as shown in the following screenshot:



Environment Properties:

- Right click on the Adhoc link icon and select Properties.
- Select the Channel Characteristics and set the parameters accordingly.



Output

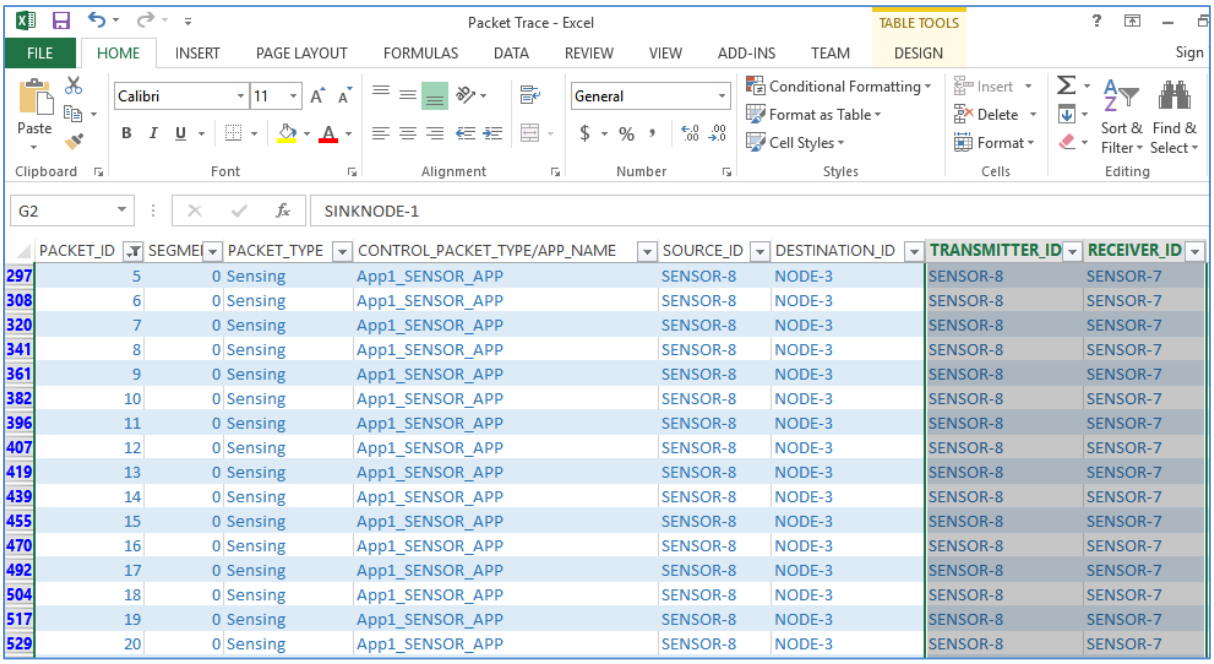
- Press + R and type %temp%, Temp folder will be opened.
- In Temp folder you will find a folder named NetSim.
- In NetSim, you will find a txt file named rpllog.txt

Open **rpllog.txt** then you will find the information about DODAG formation. For every DODAG, 6LoWPAN Gateway is the root of the DODAG

- Root is 1 with rank = 1 (Since the Node Id_1 is 6LoWPAN Gateway)
- Wireless_Sensor_Node_7(Malicious Node)

Packet is transmitted by node 8(Sensor_8) is received by node 7(Sensor_7) since the node 7 is malicious node it drops the packet. So the Throughput in this scenario is 0.

Open **Packet trace** file from simulation results window and filter only the data packets now check the **Transmitter_Id and receiver_Id** column. Since the node 7 is malicious node it drops the packet without forwarding it further.



	PACKET_ID	SEGME	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID
297	5	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
308	6	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
320	7	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
341	8	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
361	9	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
382	10	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
396	11	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
407	12	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
419	13	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
439	14	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
455	15	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
470	16	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
492	17	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
504	18	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
517	19	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
529	20	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7

A plot will open, showing the DODAG when the simulation is started and the first route is formed between sink node and the sensor. And the DODAG will be dynamically updated.