Sink Hole Attack using RPL in IOT

Software Recommended: NetSim Standard v13.0 (32-bit/ 64-bit), Visual Studio 2017/2019

Project Download Link:
https://github.com/NetSim-TETCOS/SinkHole_attack_in_IoT_RPL_v13.0/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

Introduction:

In sinkhole Attack, a compromised node or malicious node advertises fake rank information to form the fake routes. After receiving the message packet, it drops 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.

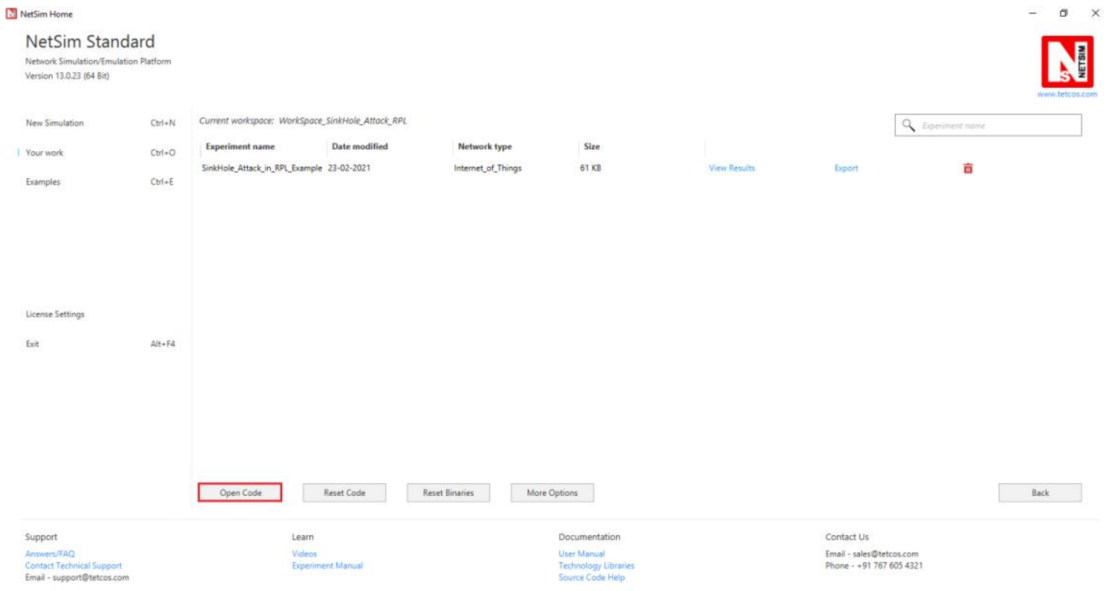
Sink Hole attack – The malicious node advertises the fake rank.
fn_NetSim_RPL_MaliciousRank( ) is the sink hole attack function.

Black Hole attack – The malicious node drops the packet.
rpl_drp_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. 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 13.0.23 (64 Bit)

www.tetcos.com

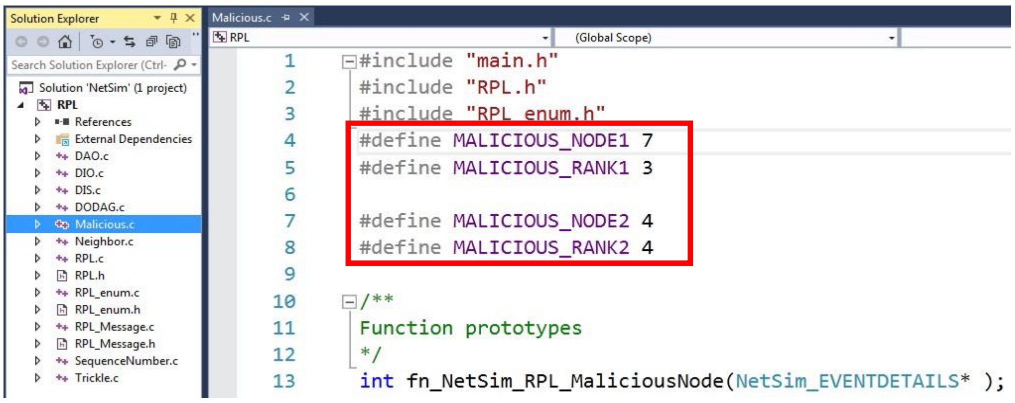
New Simulation	Ctrl+N
Your work	Ctrl+O
Examples	Ctrl+E
License Settings	
Exit	Alt+F4

Current workspace: WorkSpace_SinkHole_Attack_RPL

Experiment name	Date modified	Network type	Size		
SinkHole_Attack_in_RPL_Example	23-02-2021	Internet_of_Things	61 KB	View Results	Export

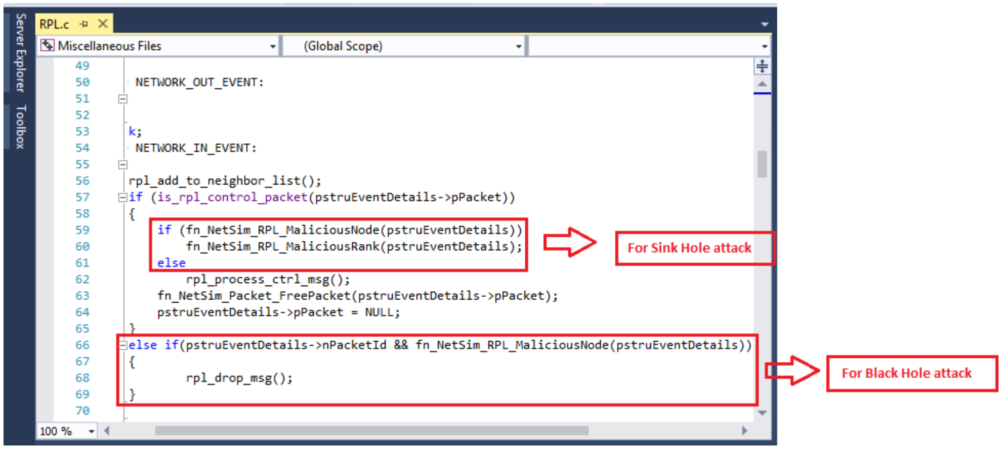
Support	Learn	Documentation	Contact Us
Answers/FAQ	Videos	User Manual	Email - sales@tetcos.com
Contact Technical Support	Experiment Manual	Technology Libraries	Phone - +91 767 605 4321
Email - support@tetcos.com		Source Code Help	

2. Set malicious node id and the fake Rank.



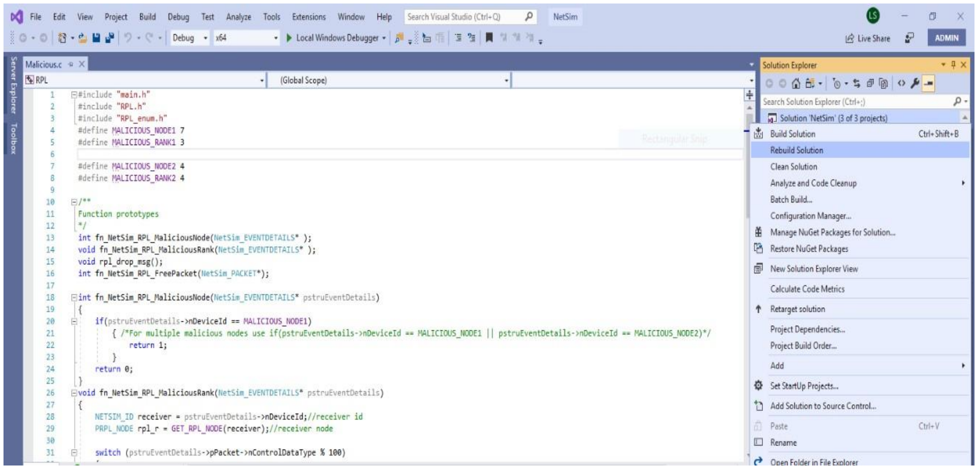
```c
#include "main.h"
#include "RPL.h"
#include "RPL_enum.h"
#define MALICIOUS_NODE1 7
#define MALICIOUS_RANK1 3

#define MALICIOUS_NODE2 4
#define MALICIOUS_RANK2 4

/**
Function prototypes
*/
int fn_NetSim_RPL_MaliciousNode(NetSim_EVENTDETAILS* );
```

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



```
 NETWORK_OUT_EVENT:

k;
 NETWORK_IN_EVENT:

rpl_add_to_neighbor_list();
if (is_rpl_control_packet(pstruEventDetails->pPacket))
{
    if (fn_NetSim_RPL_MaliciousNode(pstruEventDetails))
        fn_NetSim_RPL_MaliciousRank(pstruEventDetails);
    else
        rpl_process_ctrl_msg();
    fn_NetSim_Packet_FreePacket(pstruEventDetails->pPacket);
    pstruEventDetails->pPacket = NULL;
}
else if(pstruEventDetails->nPacketId && fn_NetSim_RPL_MaliciousNode(pstruEventDetails))
{
        rpl_drop_msg();
}
```

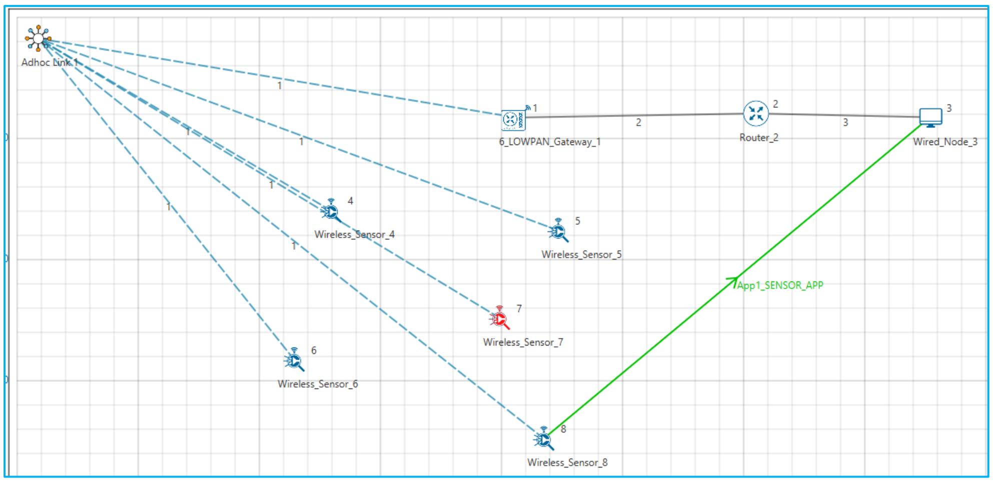
4. Now right click on Solution explorer and select Rebuild.



```
#include "main.h"
#include "RPL.h"
#include "RPL_enum.h"
#define MALICIOUS_NODE1 7
#define MALICIOUS_RANK1 3

#define MALICIOUS_NODE2 4
#define MALICIOUS_RANK2 4

/**
Function prototypes
*/
int fn_NetSim_RPL_MaliciousNode(NetSim_EVENTDETAILS* );
void fn_NetSim_RPL_MaliciousRank(NetSim_EVENTDETAILS* );
void rpl_drop_msg();
int fn_NetSim_RPL_FreePacket(NetSim_PACKET*);

int fn_NetSim_RPL_MaliciousNode(NetSim_EVENTDETAILS* pstruEventDetails)
{
    if(pstruEventDetails->nDeviceId == MALICIOUS_NODE1)
        { /*For multiple malicious nodes use if(pstruEventDetails->nDeviceId == MALICIOUS_NODE1 || pstruEventDetails->nDeviceId == MALICIOUS_NODE2)*/
            return 1;
        }
    return 0;
}
void fn_NetSim_RPL_MaliciousRank(NetSim_EVENTDETAILS* pstruEventDetails)
{
    NETSIM_ID receiver = pstruEventDetails->nDeviceId;//receiver id
    PRPL_NODE rpl_r = GET_RPL_NODE(receiver);//receiver node

    switch (pstruEventDetails->pPacket->nControlDataType % 100)
```

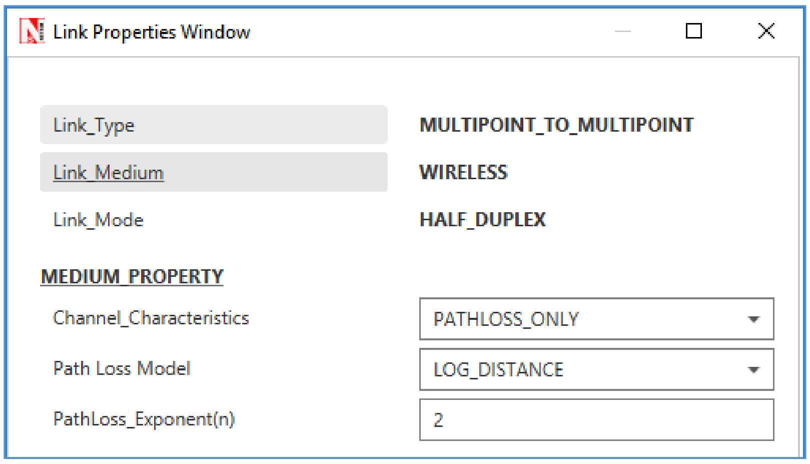
5. Upon rebuilding, modified binaries will automatically get updated in the respective bin folders of the current workspace

6. Go to NetSim home page, click on **Your work**, Click on SinkHole_Attack_in_RPL_Example.



Environment Properties:

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



Link_Type	MULTIPOINT_TO_MULTIPOINT
Link_Medium	WIRELESS
Link_Mode	HALF_DUPLEX

MEDIUM_PROPERTY

Channel_Characteristics	PATHLOSS_ONLY
Path Loss Model	LOG_DISTANCE
PathLoss_Exponent(n)	2

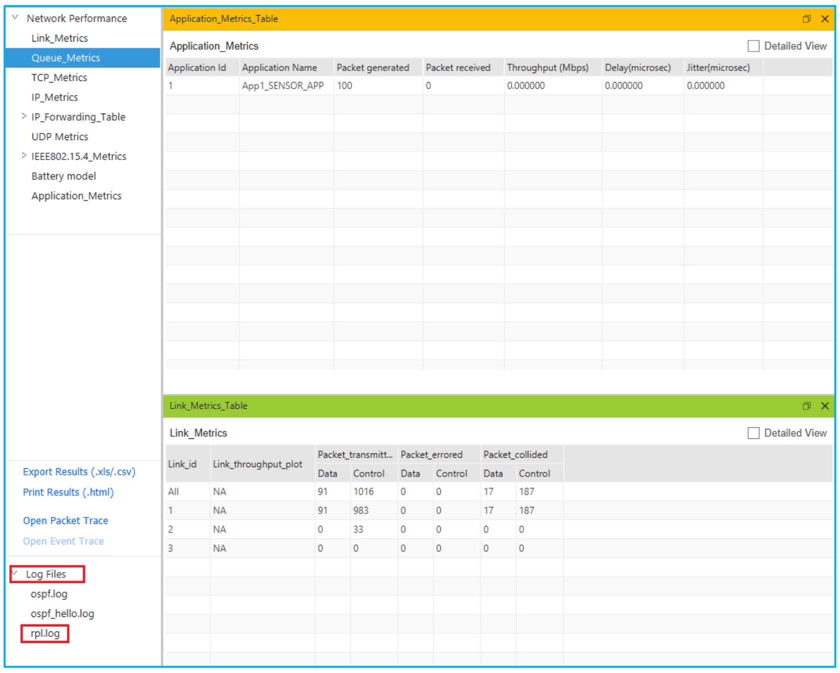
- Run Simulation for 100 Seconds.

Output

Open **rpllog.txt** file from simulation results window, 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)



Application_Metrics

Application Id	Application Name	Packet generated	Packet received	Throughput (Mbps)	Delay(microsec)	Jitter(microsec)
1	App1_SENSOR_APP	100	0	0.000000	0.000000	0.000000

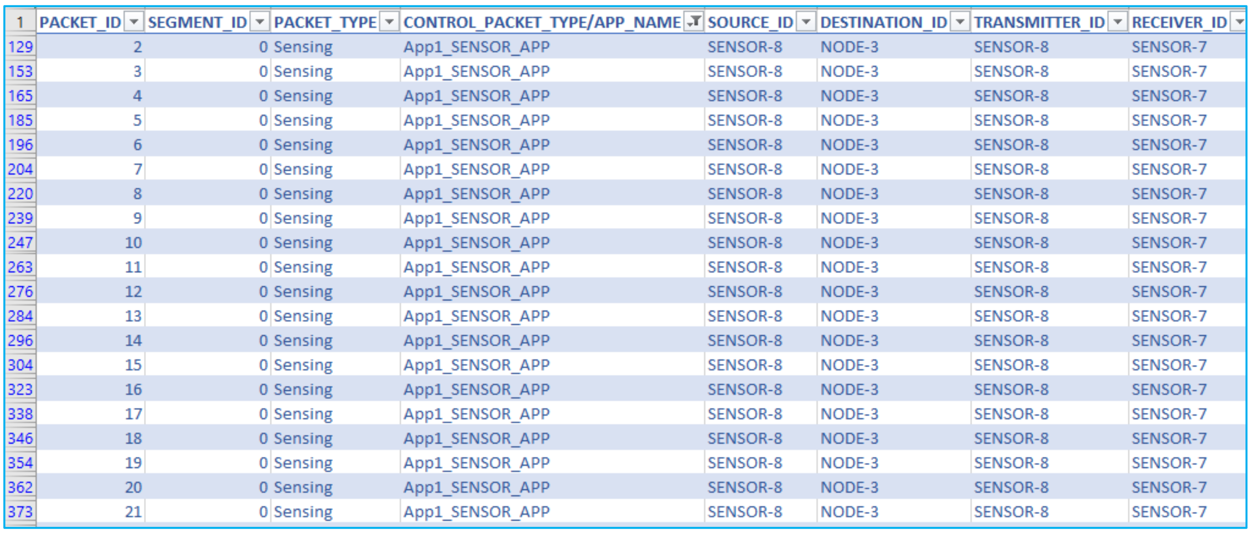
Link_Metrics

Link_id	Link_throughput_plot	Packet_transmitt...		Packet_errored		Packet_collided	
		Data	Control	Data	Control	Data	Control
All	NA	91	1016	0	0	17	187
1	NA	91	983	0	0	17	187
2	NA	0	33	0	0	0	0
3	NA	0	0	0	0	0	0

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 the **Control Packet Type/App Name** to **App1_ Sensor _App**.

Check the data packets flow, the **Transmitter_Id and receiver_Id** column. Since the node 7 is malicious node it drops the packet without forwarding it further.



1	PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID
129	2	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
153	3	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
165	4	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
185	5	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
196	6	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
204	7	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
220	8	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
239	9	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
247	10	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
263	11	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
276	12	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
284	13	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
296	14	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
304	15	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
323	16	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
338	17	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
346	18	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
354	19	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
362	20	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7
373	21	0	Sensing	App1_SENSOR_APP	SENSOR-8	NODE-3	SENSOR-8	SENSOR-7