

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

Project Download Link:

https://github.com/NetSim-TETCOS/DIO_Suppression_Attack_in_IoT_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 DIO Suppression Attack, a malicious node broadcast DIO message 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 neighbors 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.

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.

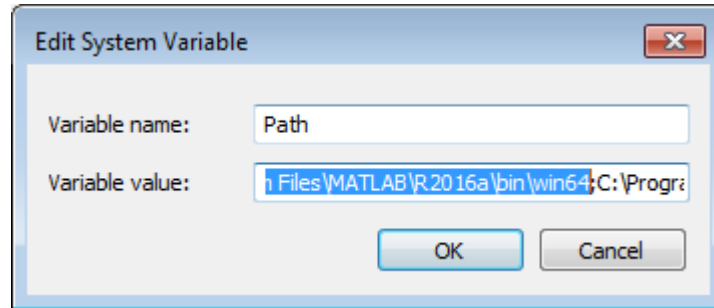
The DIO.c file contains the following functions

1. **fn_NetSim_RPL_MaliciousNode()**
This function is used to identify whether a current device is malicious or not in-order to establish malicious behavior.
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.

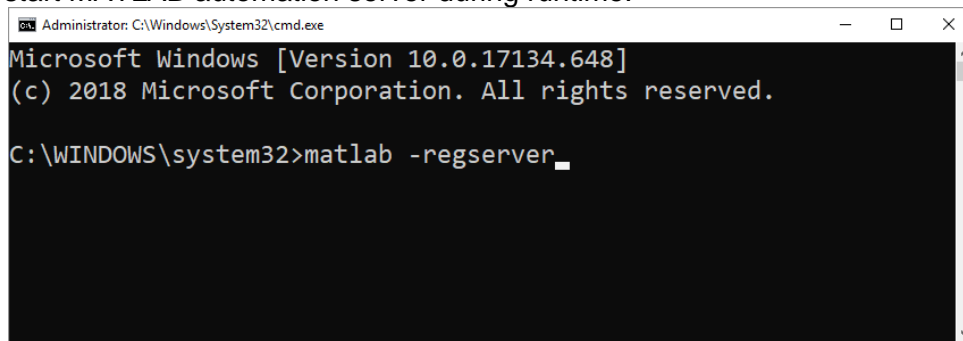
Steps:

1. Make sure that the following directory is in the PATH(Environment variable)
<Path where MATLAB is installed>\bin\win64

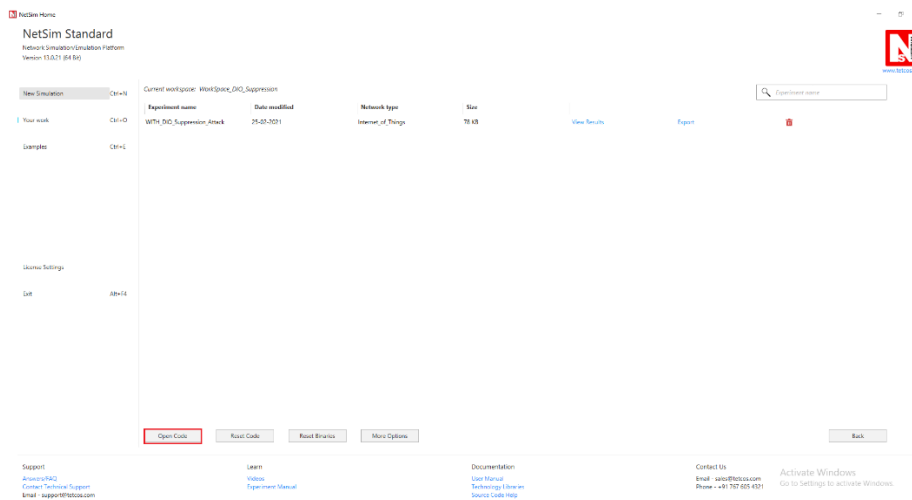


Note: If the machine has more than one MATLAB installed, the directory for the target platform must be ahead of any other MATLAB directory (for instance, when compiling a 64-bit application, the directory in the MATLAB 64-bit installation must be the first one on the PATH).

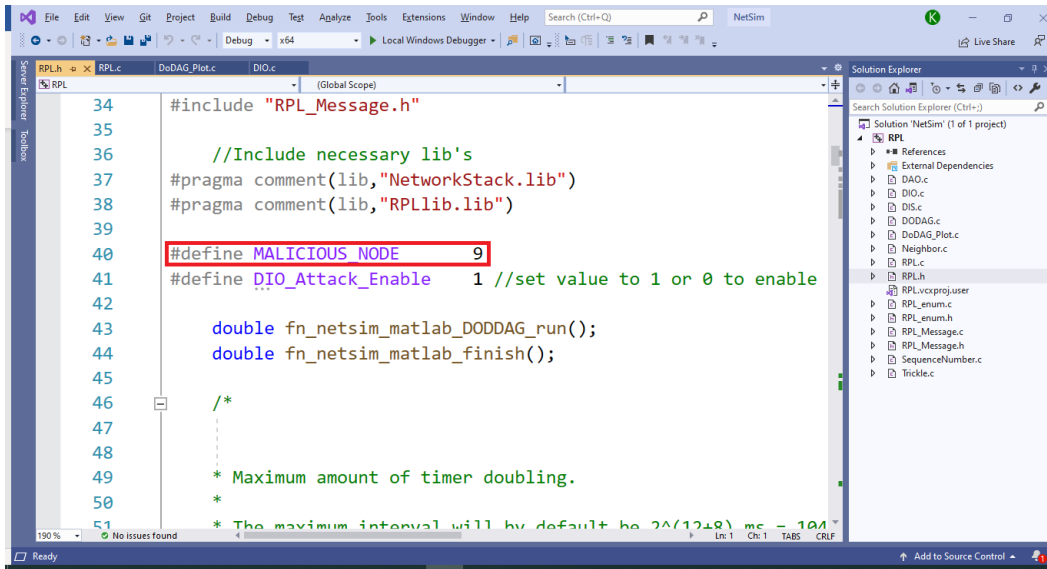
2. Open Command prompt as admin and execute the command “matlab -regserver”. This will register MATLAB as a COM automation server and is required for NetSim to start MATLAB automation server during runtime.



3. Go to home page, Click on **Your work>Workspace options** and click on the **Open code** button.

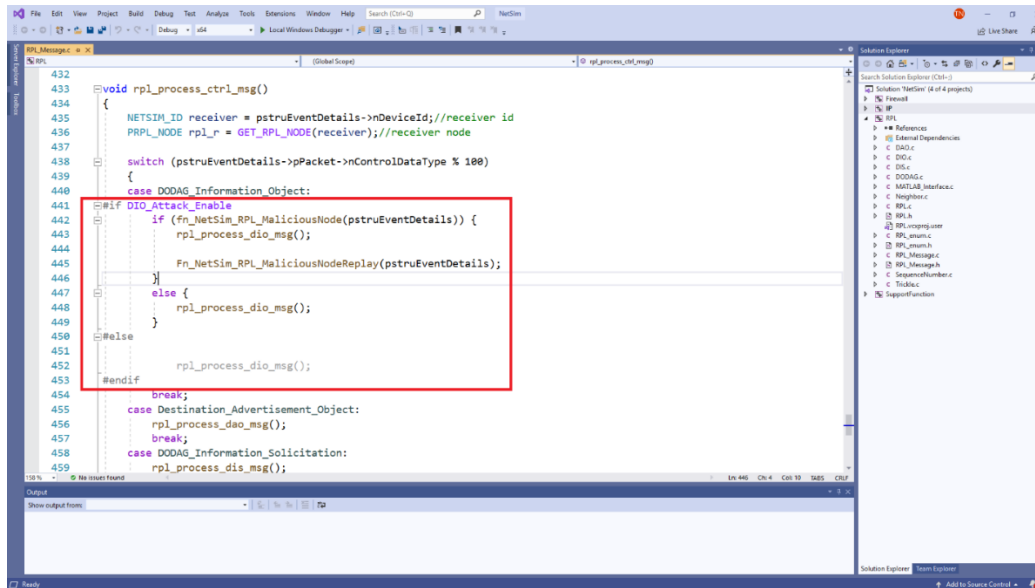


4. Set malicious node id in RPL.h file



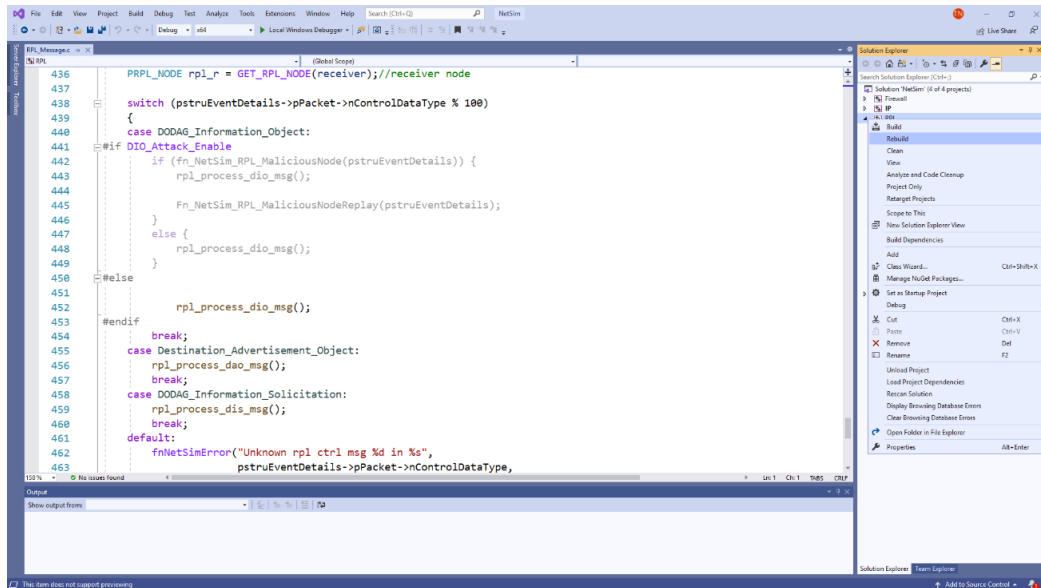
```
34 #include "RPL_Message.h"
35
36 //Include necessary lib's
37 #pragma comment(lib,"NetworkStack.lib")
38 #pragma comment(lib,"RPLlib.lib")
39
40 #define MALICIOUS_NODE 9
41 #define DIO_Attack_Enable 1 //set value to 1 or 0 to enable
42
43 double fn_netsim_matlab_DODDAG_run();
44 double fn_netsim_matlab_finish();
45
46 /*
47  *
48  * Maximum amount of timer doubling.
49  *
50  * The maximum interval will by default be 2^(12+8) ms = 1048576 ms
51  */
```

5. The section of code that is highlighted is added to the RPL_Message.c file

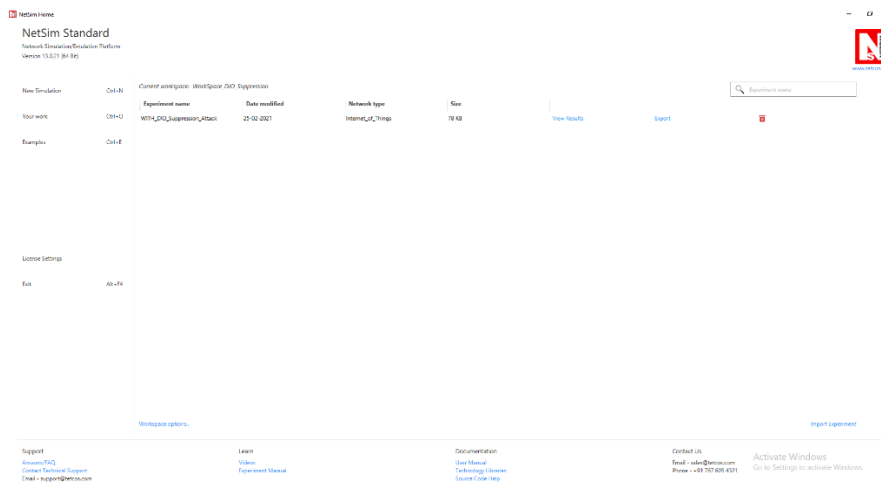


```
432 void rpl_process_ctrl_msg()
433 {
434     NETSIM_ID receiver = pstruEventDetails->nDeviceId;//receiver id
435     RPL_NODE rpl_n = GET_RPL_NODE(receiver);//receiver node
436     switch (pstruEventDetails->pPacket->nControlDataType % 100)
437     {
438     case DODAG_Information_Object:
439     #if DIO_Attack_Enable
440     if (fn_Netsim_RPL_MaliciousNode(pstruEventDetails)) {
441     rpl_process_dio_msg();
442     Fn_Netsim_RPL_MaliciousNodeReplay(pstruEventDetails);
443     }
444     else {
445     rpl_process_dio_msg();
446     }
447     #else
448     rpl_process_dio_msg();
449     #endif
450     #endif
451     break;
452     case Destination_Advertisement_Object:
453     rpl_process_dao_msg();
454     break;
455     case DODAG_Information_Solicitation:
456     rpl_process_dis_msg();
457     }
```

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

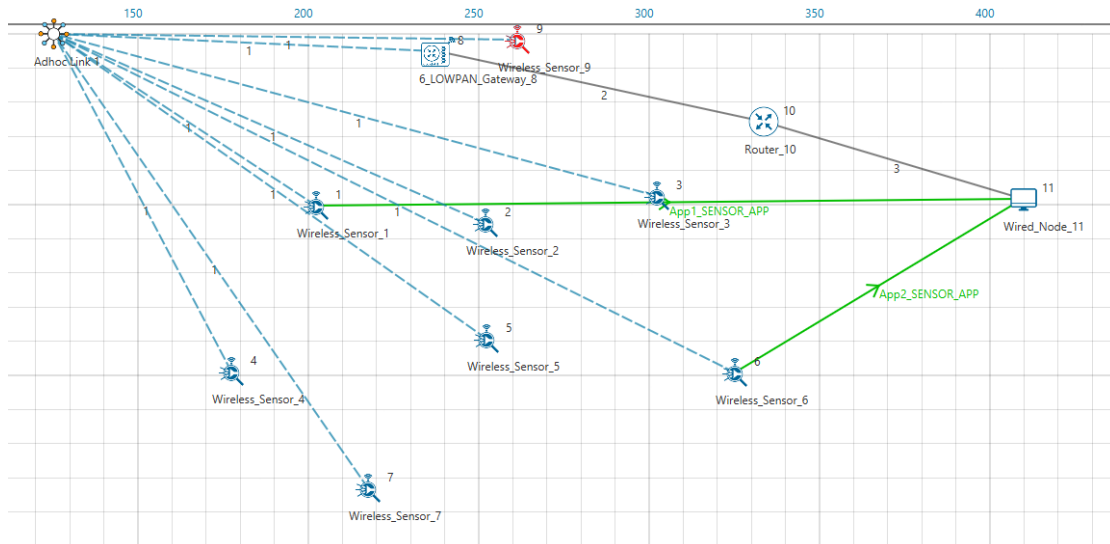


7. Upon rebuilding, **libRPL.dll** will automatically get replaced in the respective bin folders of the current workspace
8. Go to NetSim home page, click on **Your work**, Click on **WITH_DIO_Suppression_Attack**.



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

1. Create a network scenario in **IoT (Internet of Things)** with **UDP** running in the **Transport Layer** and **RPL** in **Network Layer**.
2. Go to **Your Work** option in NetSim Home Screen and open the saved example, **WITH_DIO_Suppression_Attack**. The network scenario and the settings done is explained below:



Note: In above screenshot Red color Wireless_Sensor_Node_9 is a malicious node.

For Application 1:

- Source – Device id 1
- Destination – Device id 11
- Packet Size – 50Bytes
- Inter Arrival Time – 1000000microsec

For Application 2:

- Source – Device id 6
- Destination – Device id 11
- Packet Size – 50Bytes
- Inter Arrival Time – 1000000microsec

Link Properties (Adhoc Link 1)

Channel Characteristics – Path Loss only
 Path Loss model – LOG DISTANCE
 Path Loss Exponent- 3

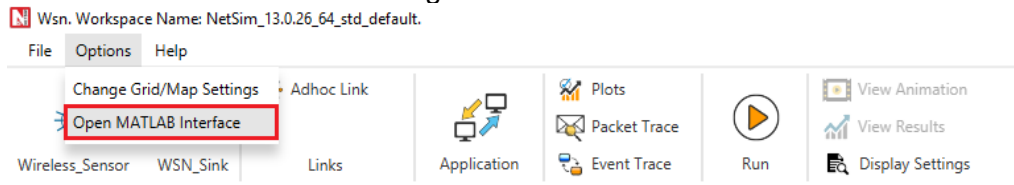
Device Properties: Go to 6LoWPAN Gateway Properties->Network_Layer->DIORedundancyConstant-> 6.

3. The DIO suppression attack requires the adversary to transmit only k (DIO Redundancy Constant) DIO messages at each Trickle period.
4. DIO Redundancy Constant(k) acts as suppression threshold, as we set 6, the malicious node will replay the DIO message 6 times to the neighboring nodes. After replaying the DIO message, the neighboring nodes will suppress their own DIO transmission.
5. Run simulation and press any key to continue. NetSim simulation console will show the following message in the console “Waiting for NetSim MATLAB Interface to connect...”

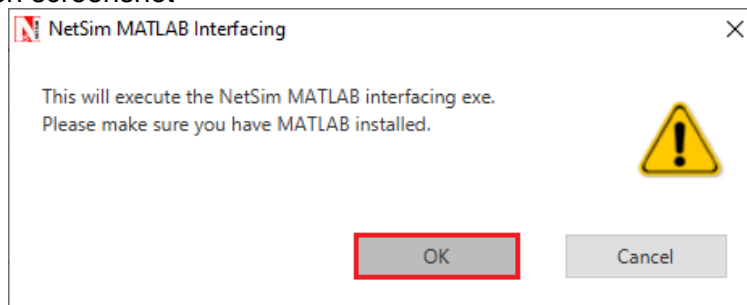
```
C:\Users\MAHAVEER\Documents\WorkSpace_DIO_Suppression\bin\bin_x64\NetSimCore.exe

***
NetSim start
Network Stack loaded
Error in creating C:\Users\MAHAVEER\AppData\Local\Temp\NetSim\std13.0.26_x64\log directory. Error number 17
Initializing simulation
Config file reading complete
License re-validation complete
Protocol binaries loaded
Stack variables initialized
Could Not Find C:\Users\MAHAVEER\AppData\Local\Temp\NetSim\std13.0.26_x64\Plot_*
Metrics variables initialized
Initialising Winsock for matlab interface...Initialized.
Waiting for NetSim Matlab Interface to connect...
```

6. Click on **Options** from NetSim design window and click on **Open MATLAB Interface** as shown in below given screenshot



7. It will open the popup related MATLAB Interfacing, Click on **OK**. As shown in below given screenshot



8. 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.
9. View the packet animation. You will find that malicious node (Device id 9) even after receiving DIO message from neighbor nodes it will start transmitting repeatedly DIO message to neighbor nodes.
10. This will cause some nodes to remain hidden and some route to remain undiscovered or in the worst case, a partition of the network.

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

To run simulations without DIO Suppression attack set the value of the variable DIO_ATTACK_ENABLE to 0 instead of 1. Rebuild the RPL source codes and run Simulation.

```

34 #include "RPL_Message.h"
35
36 //Include necessary lib's
37 #pragma comment(lib,"NetworkStack.lib")
38 #pragma comment(lib,"RPLlib.lib")
39
40 #define MALICIOUS_NODE 9
41 #define DIO Attack Enable 1 //set value to 1 or 0 to enable
42
43 double fn_netsim_matlab_DODDAG_run();
44 double fn_netsim_matlab_finish();
45
46 /*
47  *
48  * Maximum amount of timer doubling.
49  *
50  * The maximum interval will by default be 2^(12+8) ms = 104

```

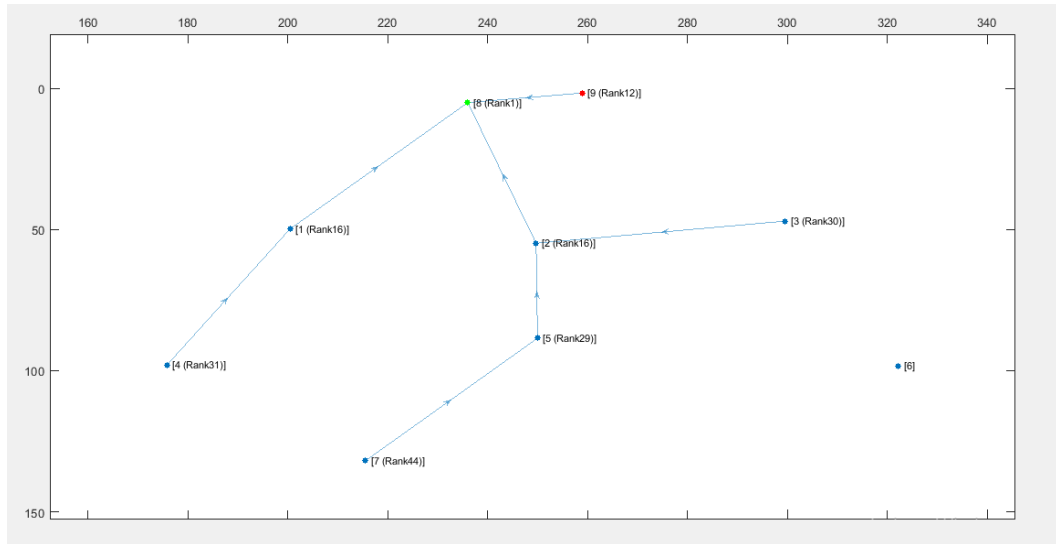
Output:

Case 1: With DIO Suppression Attack

Application	Application Name	Packet generated	Packet received	Throughput (B/s)	Delay (ms)	Jitter (ms)	Source	Destination	Segment Size	Segment Received	ACK Size	ACK Received	Duplicate ACK received
1	App1_SENDER_APP	100	0	0.00000	0.000000	0.000000	WIFIHEL_SENDER_0	ANY_DEVICE	0	0	0	0	0
2	App2_SENDER_APP	100	0	0.00000	0.000000	0.000000	WIFIHEL_SENDER_1	ANY_DEVICE	0	0	0	0	0
3	App3_SENDER_APP	100	0	0.00000	0.000000	0.000000	WIFIHEL_SENDER_2	ANY_DEVICE	0	0	0	0	0
4	App4_SENDER_APP	100	0	0.00000	0.000000	0.000000	WIFIHEL_SENDER_3	ANY_DEVICE	0	0	0	0	0
5	App5_SENDER_APP	100	0	0.00000	0.000000	0.000000	WIFIHEL_SENDER_4	ANY_DEVICE	0	0	0	0	0
6	App6_SENDER_APP	100	0	0.00000	0.000000	0.000000	WIFIHEL_SENDER_5	ANY_DEVICE	0	0	0	0	0
7	App7_SENDER_APP	100	0	0.00000	0.000000	0.000000	WIFIHEL_SENDER_6	ANY_DEVICE	0	0	0	0	0
8	App8_SENDER_APP	100	0	0.00000	0.000000	0.000000	WIFIHEL_SENDER_7	ANY_DEVICE	0	0	0	0	0
9	App9_SENDER_APP	100	0	0.00000	0.000000	0.000000	WIFIHEL_SENDER_8	ANY_DEVICE	0	0	0	0	0
10	App10_SENDER_APP	100	0	0.00000	0.000000	0.000000	WIFIHEL_SENDER_9	ANY_DEVICE	0	0	0	0	0

Link ID	Link ID (hex)	Link ID (dec)	Link ID (hex)	Link ID (dec)	Link ID (hex)	Link ID (dec)	Link ID (hex)	Link ID (dec)	Link ID (hex)	Link ID (dec)	Link ID (hex)	Link ID (dec)	Link ID (hex)	Link ID (dec)
1	0x00000001	1	0x00000002	2	0x00000003	3	0x00000004	4	0x00000005	5	0x00000006	6	0x00000007	7
2	0x00000008	8	0x00000009	9	0x0000000A	10	0x0000000B	11	0x0000000C	12	0x0000000D	13	0x0000000E	14
3	0x0000000F	15	0x00000010	16	0x00000011	17	0x00000012	18	0x00000013	19	0x00000014	20	0x00000015	21
4	0x00000016	22	0x00000017	23	0x00000018	24	0x00000019	25	0x0000001A	26	0x0000001B	27	0x0000001C	28
5	0x0000001D	29	0x0000001E	30	0x0000001F	31	0x00000020	32	0x00000021	33	0x00000022	34	0x00000023	35

DODAG Formation Graph:



When root node (Wireless_Sensor_Node_8) 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 neighbor nodes such that it prevents the DIO messages from other neighbor nodes reaching them.

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

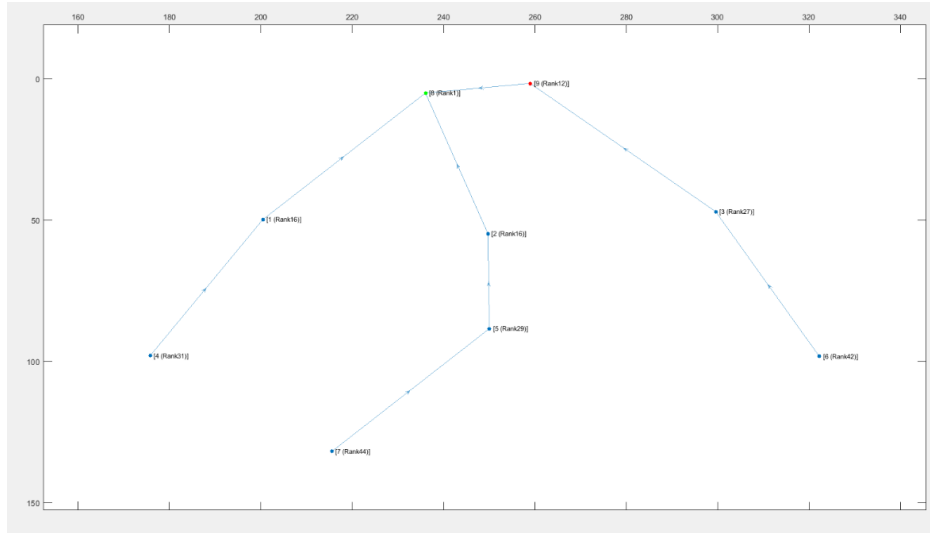
We can observe from the above graph that Wireless_Sensor_Node_6 is not part of DODAG formation as it is not discovered and remain hidden in the network.

Case 2: Without DIO Suppression Attack

Application Metrics Table							TCP Metrics Table						
Application Metrics	Application Name	Packet generated	Packet received	Throughput (Mbps)	Download (KB/s)	Upload (KB/s)	Source	Destination	Segment Count	Segment Received	All Sent	All Received	Duplicate sent/received
1	App1 (SENSOR_APP)	100	80	0.000250	3330.59081	400.021540	WIRELESS_SENSOR_1	ANY_SERVICE	0	0	0	0	0
2	App2 (SENSOR_APP)	100	81	0.000216	3950.30110	3716.101760	WIRELESS_SENSOR_2	ANY_SERVICE	0	0	0	0	0
							WIRELESS_SENSOR_3	ANY_SERVICE	0	0	0	0	0
							WIRELESS_SENSOR_4	ANY_SERVICE	0	0	0	0	0
							WIRELESS_SENSOR_5	ANY_SERVICE	0	0	0	0	0
							WIRELESS_SENSOR_6	ANY_SERVICE	0	0	0	0	0
							WIRELESS_SENSOR_7	ANY_SERVICE	0	0	0	0	0
							WIRELESS_SENSOR_8	ANY_SERVICE	0	0	0	0	0
							MALICIOUS_SENSOR_9	ANY_SERVICE	0	0	0	0	0
							ROUTER_10	ANY_SERVICE	0	0	0	0	0
							WIRELESS_SENSOR_11	ANY_SERVICE	0	0	0	0	0

IP Metrics Table							OSPF Metrics Table						
IP Metrics	Interface Name	Packet Sent	Packet Received	Bytes Sent	Bytes Received	Link Cost	OSPF Metric	OSPF Metric	OSPF Metric	OSPF Metric	OSPF Metric	OSPF Metric	OSPF Metric
1	WLAN0	100	80	3330	4000	1	1	1	1	1	1	1	1
2	WLAN0	100	81	3950	3716	1	1	1	1	1	1	1	1
3	WLAN0	100	80	3330	4000	1	1	1	1	1	1	1	1

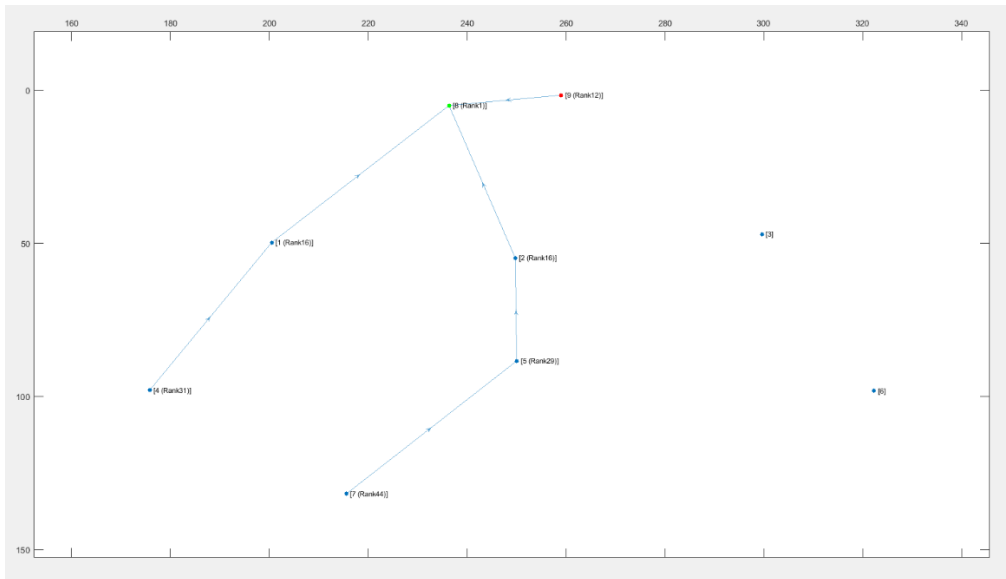
DODAG Formation Graph:



Case 3: With DIO Suppression Attack

When DIORedundancyConstant is set to 7

DODAG Formation Graph:



We can observe from the graph that Wireless_Sensor_Node_3 and Wireless_Sensor_Node_6 is not part of DODAG formation as it is not discovered and remain hidden in the network.

We can observe 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 neighboring 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.