

False Data Injection Attack using RPL in IOT

Software: NetSim Standard v13.1 (64 bit), Visual Studio 2019 and Node-RED

Project Download Link:

https://github.com/NetSim-TETCOS/False-Data-Injection-Attack-in-IOT-with-Node-Red-Interfacing_v13.1/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>

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

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, 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 modifies the packet payload and send it to its parent.

IoT False Data Injection Attack

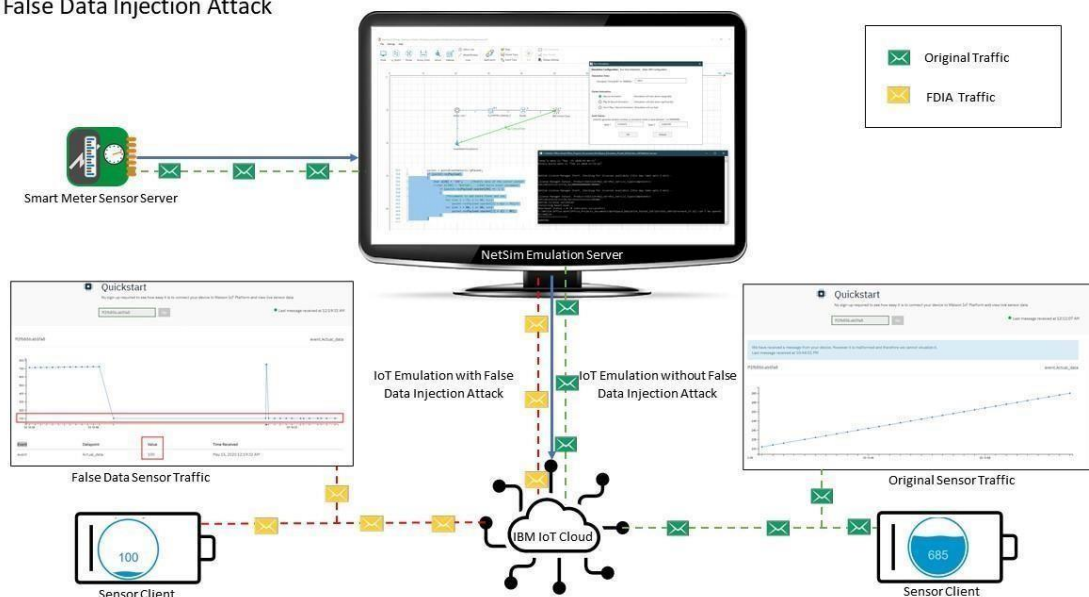


Figure 1: IOT False Data Injection Attack

How are the payloads modified by interfacing with NetSim?

Usually, the sensor traffic sent from sensor server is sent directly to IBM cloud and then the client connects the cloud to receive the sensor traffic.

But to perform FDIA we are sending the sensor traffic through NetSim Emulator before it reaches to IBM cloud.

NetSim emulator then access the payload and modify it and re-inject to the network. The cloud receives the data which is injected by Emulator.

Node-Red Installation and its working

- Installation of Node Red on Windows: Follow the steps on how to install Node-red in windows, at <https://nodered.org/docs/getting-started/windows>
- Running Node-Red and configuring Sensor server:
- Run Node-red through cmd window.
 - i. Open Node-red.js command prompt
 - ii. Install the following packages by following commands one by one.
 - a) npm install node-red-dashboard.
 - b) npm install node-red-contrib-scx-ibmiotapp
 - iii. Now run node-red by following command
 - a) Node-red
- Open browser <http://127.0.0.1:1880/?#flow/>

1. Import the Sensor_Flow.json sample into Node-RED from the Node-RED-Samples folder that is part of the Project directory.

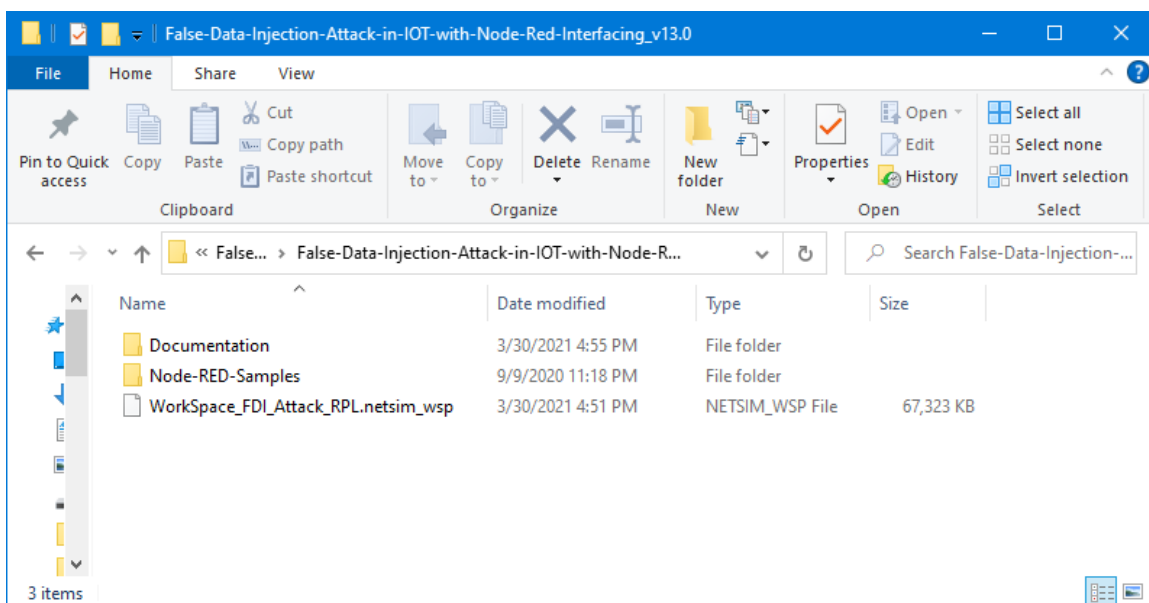


Figure 2: Project directory Contains documents, Samples and workspace etc

2. Upon importing the flow appears as shown below, in Node-RED:

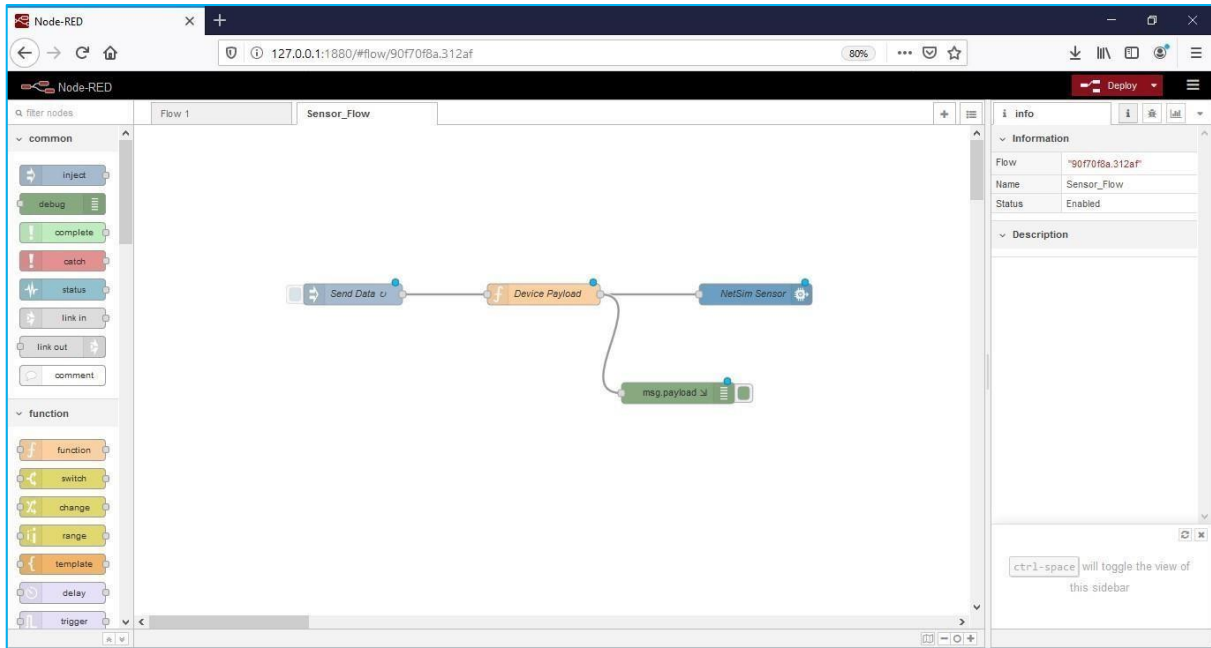


Figure 3: Node-RED flow

3. Click on the deploy button on the top left to start the flow.
4. Double click on the Blue NetSim Sensor icon and click on the link next to the Quickstart Id text box as shown below:

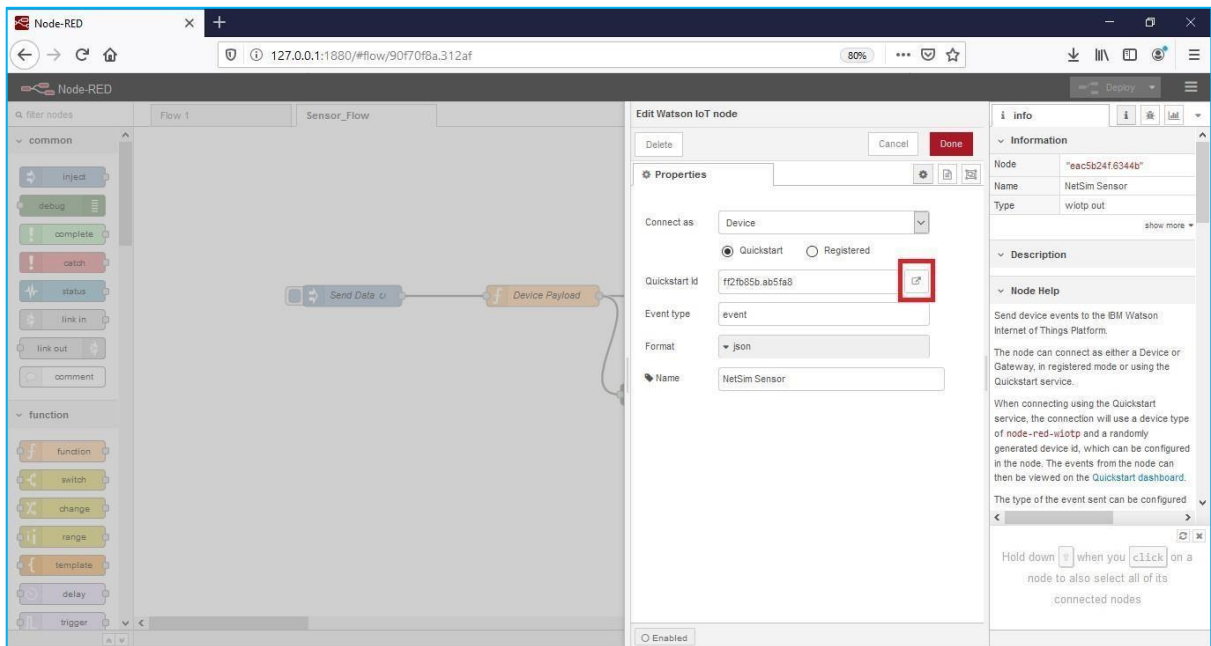
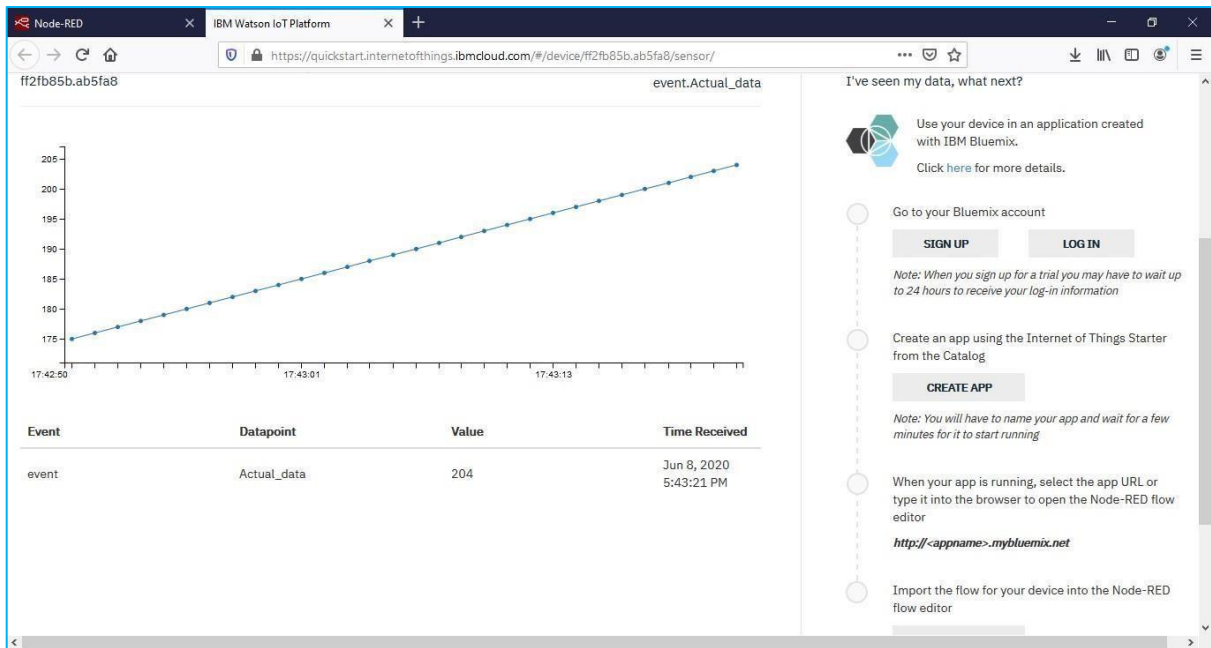


Figure 4: Quickstart Id set in Blue NetSim Sensor icon

5. This opens IBM Watson Quickstart interface where the live readings received from the Node-RED flow is plotted as shown below:



Malicious.c file contains the following functions added to the RPL project.

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_false_data_emulation_injection();` //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.

1. `fn_NetSim_RPL_MaliciousRank ();` //is the sink hole attack function.

False data injection attack – The malicious node changes the payload of the packet.

2. `rpl_false_data_emulation_injection ();` // is the False data injection.

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 to simulate

1. Open the Source codes in Visual Studio by going to Your work->Source Code and Clicking on Open code Option in NetSim Home Screen window.
2. Now right click on Solution explorer and select Rebuild.

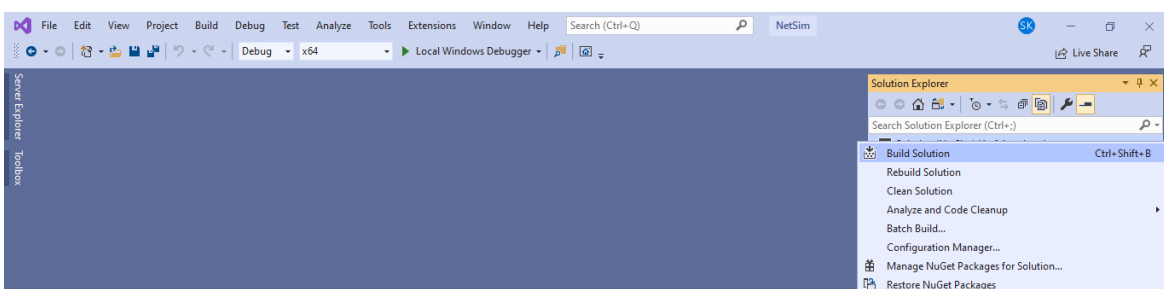


Figure 5: Screen shot of NetSim project source code in Visual Studio

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

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:

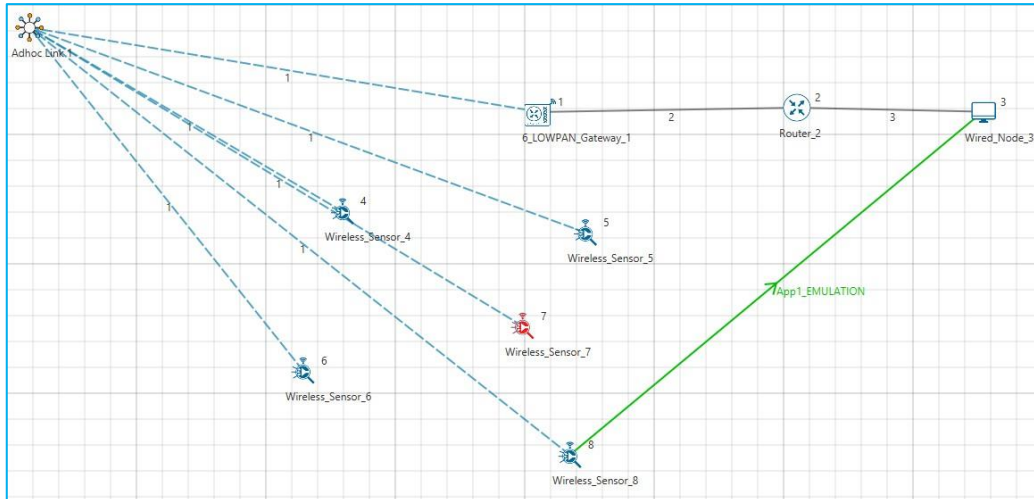


Figure 6: Sensor communicating with the server in NetSim

3. Right click on the Adhoc link icon and select Properties.
 - Channel Characteristics – PATHLOSS_ONLY.
 - Pathloss Model – LOG_DISTANCE.
 - Pathloss Exponent – 2.

Results and discussion

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)

Packet is transmitted by node 8(Sensor_8) is received by node 7(Sensor_7) since the node 7 is malicious node changes the payload of the packet and forwards the packet to the destination which can be analysed using Wireshark capture files after emulation or during emulation on IBM platform.

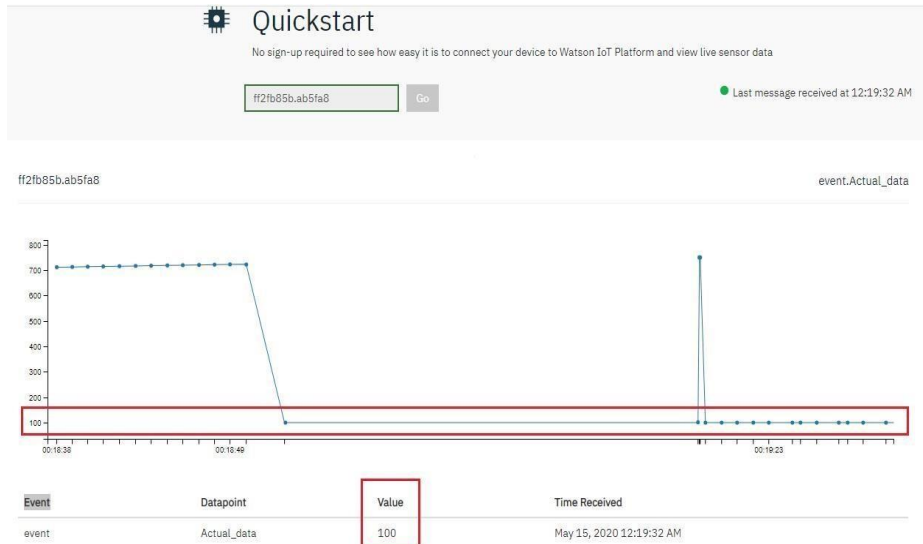


Figure 7: Web console of IBM Platform

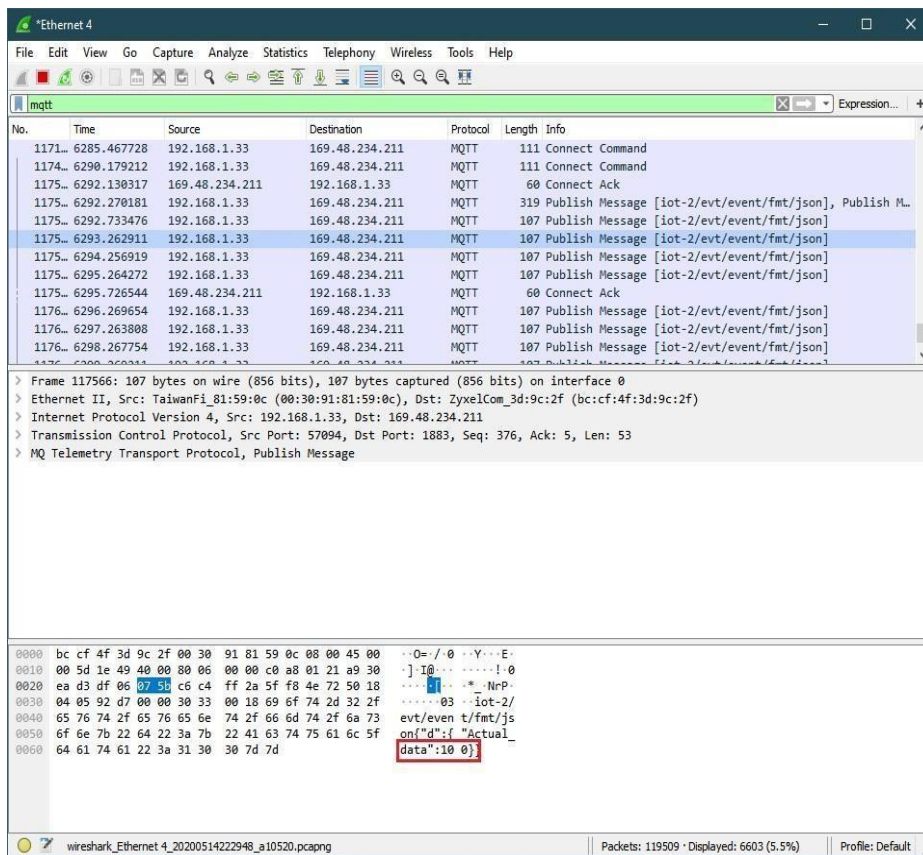


Figure 8: Wireshark

We see that the malicious node attracts network traffic by advertising false rank information. Subsequently it injects false data into the payload of the packet and the impact can be seen on the plots shown in IBM Watson user interface.

Appendix: NetSim source code modifications

Set malicious node id and the fake Rank in **Malicious.c** file.

```
#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_false_data_emulation_injection();
int fn_NetSim_RPL_FreePacket(NetSim_PACKET*);
```

Changes to **fn_NetSim_RPL_Run()** in **RPL.c** file, within **RPL** project

```
_declspec (dllexport) int fn_NetSim_RPL_Run()
{
switch (pstruEventDetails->nEventType)
{
case NETWORK_OUT_EVENT:
{
}
}
break;
case 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_false_data_emulation_injection();
}
}
}
```