False Data Injection Attack using RPL in IOT

Software Recommended: NetSim Standard v12.1/v12.2 (32-bit/ 64-bit), Visual Studio 2017/2019 and Node-RED

Follow the instructions specified in the following link to clone/download the project folder from GitHub using Visual Studio:
https://tetcos.freshdesk.com/support/solutions/articles/14000099351-how-to-clone-netsim-file- exchange-project-repositories-from-github-

Other tools such as GitHub Desktop, SVN Client, Sourcetree, Git from the command line, or any client you like to clone the Git repository.

Note: It is recommended not to download the project as an archive (compressed zip) to avoid incompatibility while importing workspaces into NetSim.

Secure URL for the GitHub repository:

v12.1:https://github.com/NetSim-TETCOS/False_Data_Injection_Attack_in_IoT_v12.1.git

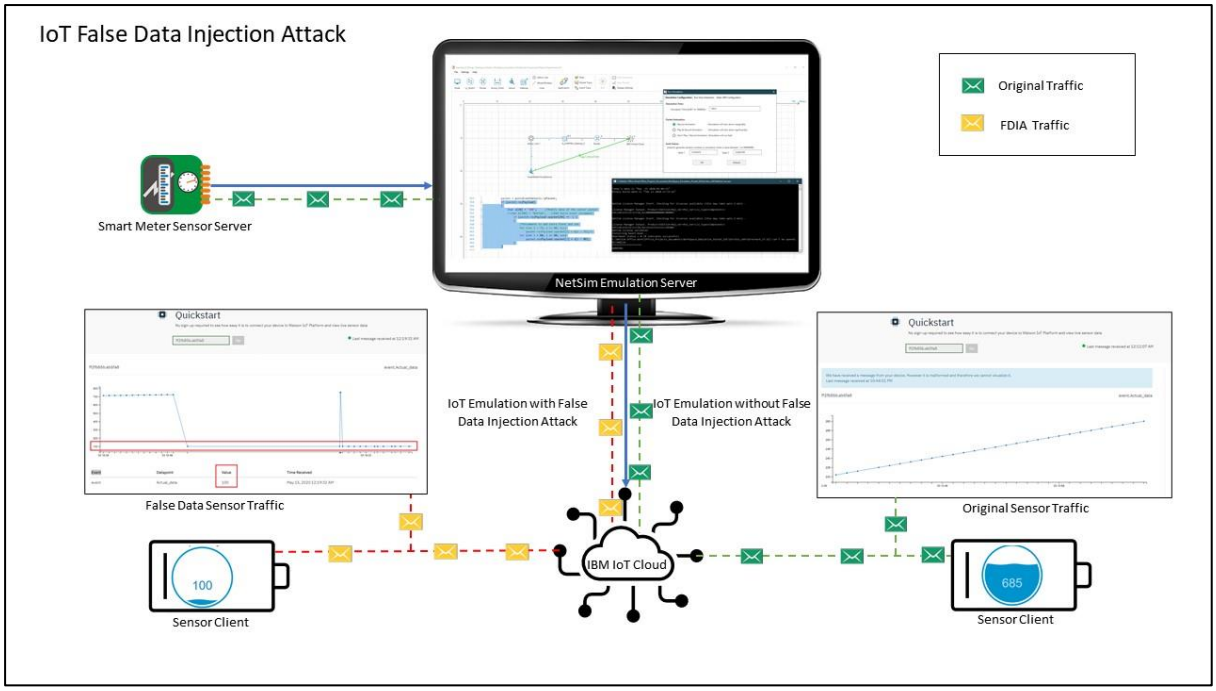
v12.2:https://github.com/NetSim-TETCOS/False-Data-Injection-Attack-in-IOT-with-Node-Red-Interfacing_v12.2.git

Note: The cloned project directory will contain the documentation specific to the NetSim version (v12.1/v12.2).

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
Original Traffic
FDIA Traffic
Smart Meter Sensor Server
NetSim Emulation Server
Quickstart
IoT Emulation with False Data Injection Attack
IoT Emulation without False Data Injection Attack
Quickstart
False Data Sensor Traffic
Original Sensor Traffic
IBM IoT Cloud
100
Sensor Client
685
Sensor Client

How the payloads are modified by interfacing with NetSim?

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

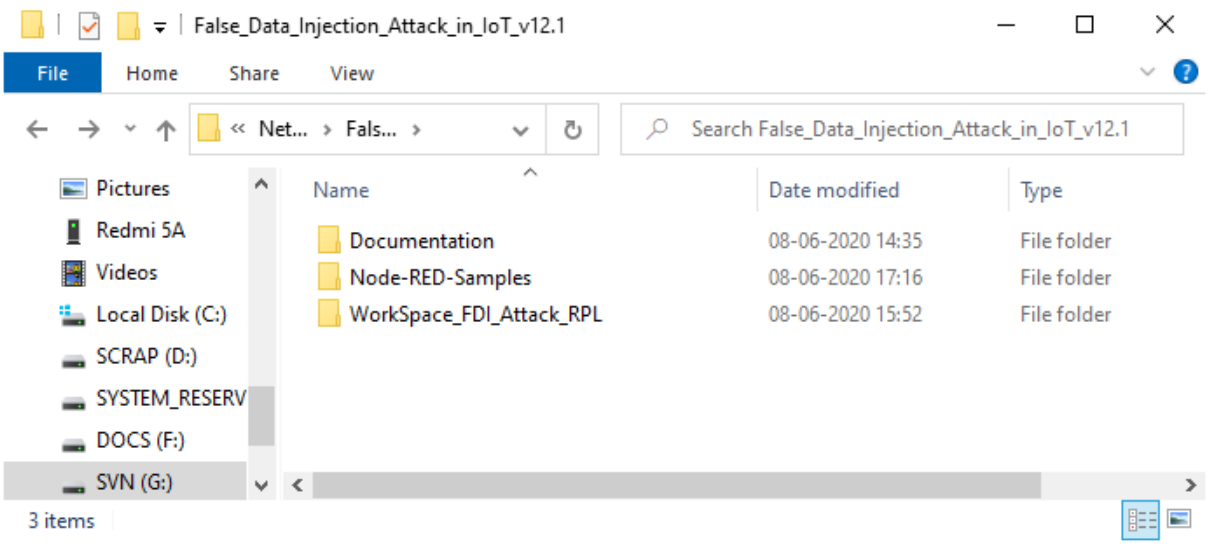
But to perform FDIA we are sending the sensor traffic through NetSim Emulator before it reaches to IBM cloud.

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.

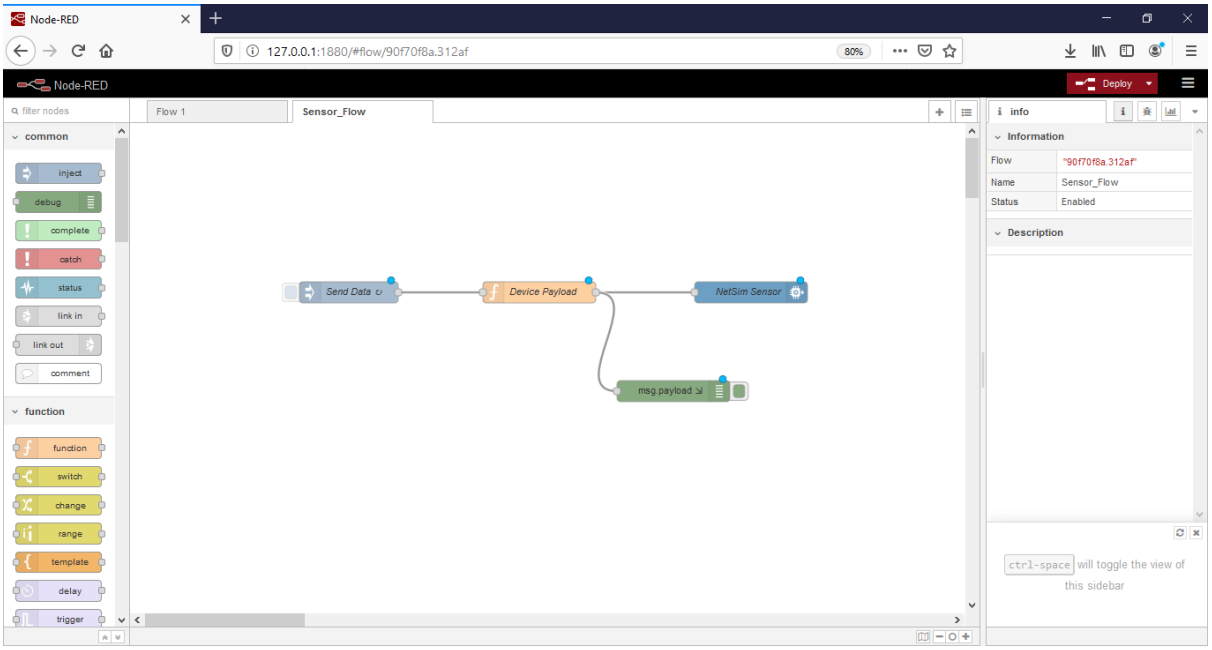
Node-Red Installation and its working

- 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
            1. npm install node-red-dashboard
            2. npm install node-red-contrib-scx-ibmiotapp
            3. npm install node-red-contrib-scx-ibmiotapp
        iii. Now run node-red by following command
            1. Node-red
    - Open browser http://127.0.0.1:1880/?#flow/

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

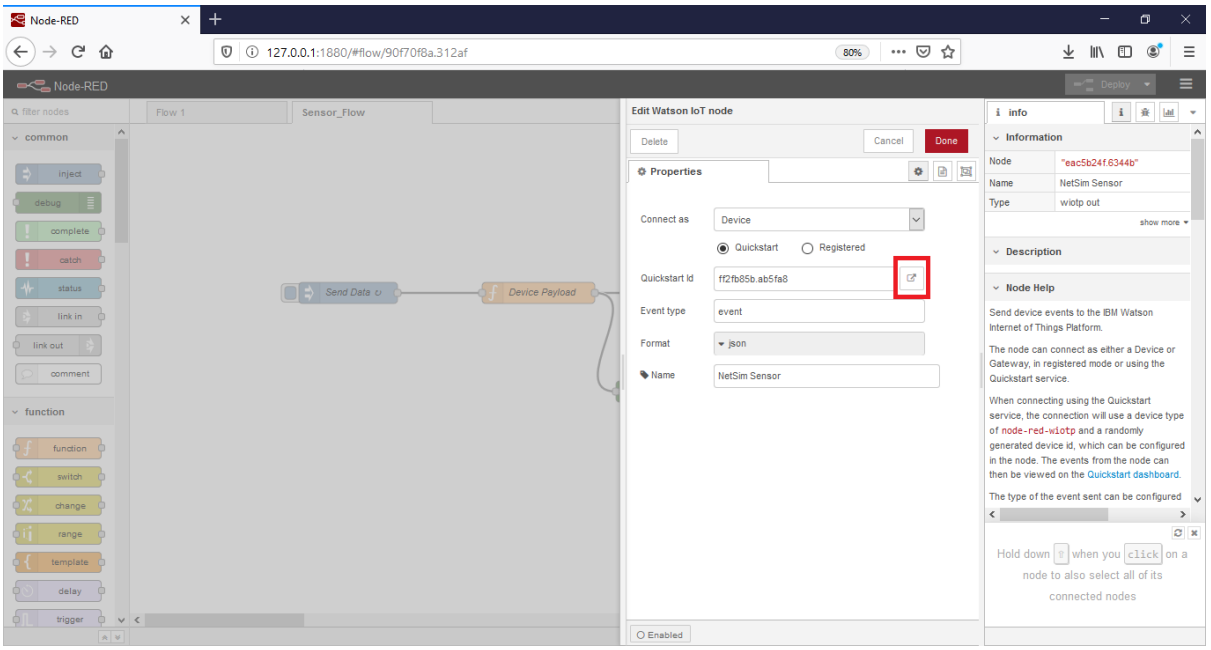


Upon importing the flow appears as shown below, in Node-RED:

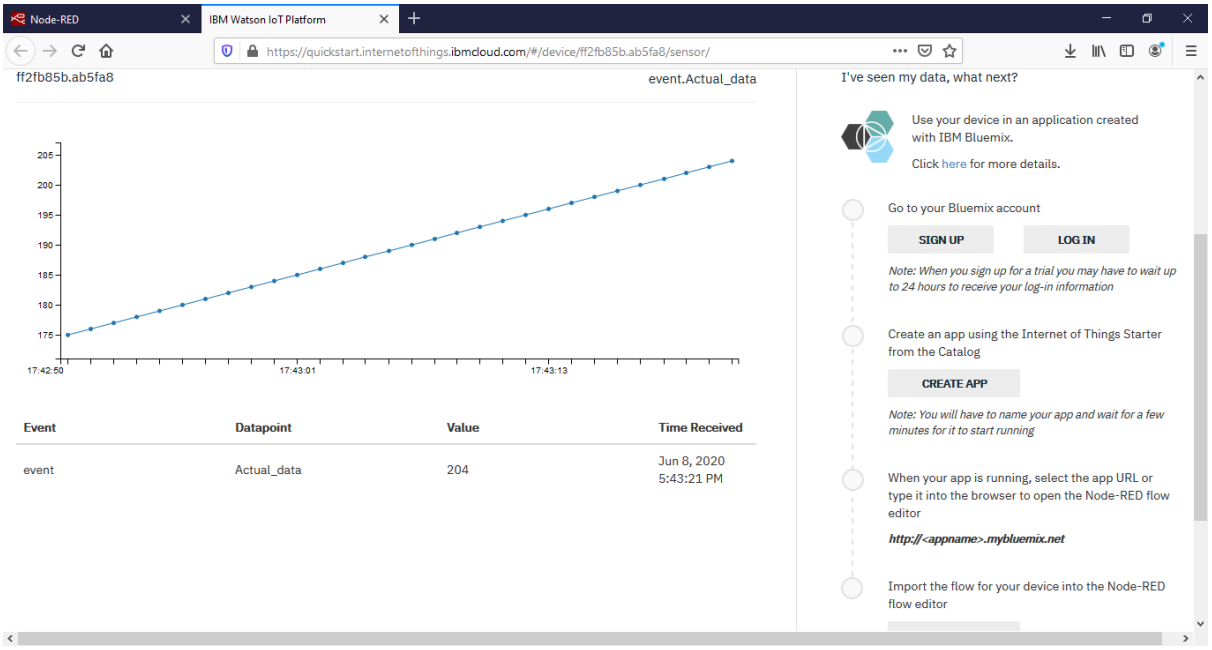


Click on the deploy button on the top left to start the flow.

Double click on the Blue NetSim Sensor icon and click on the link next to the Quickstart Id text box as shown below:



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



For further reference see: https://tetcos.com/pdf/v12.1/IOT-Application-IF-with-NodeRed-and-IBM-Watson.pdf

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

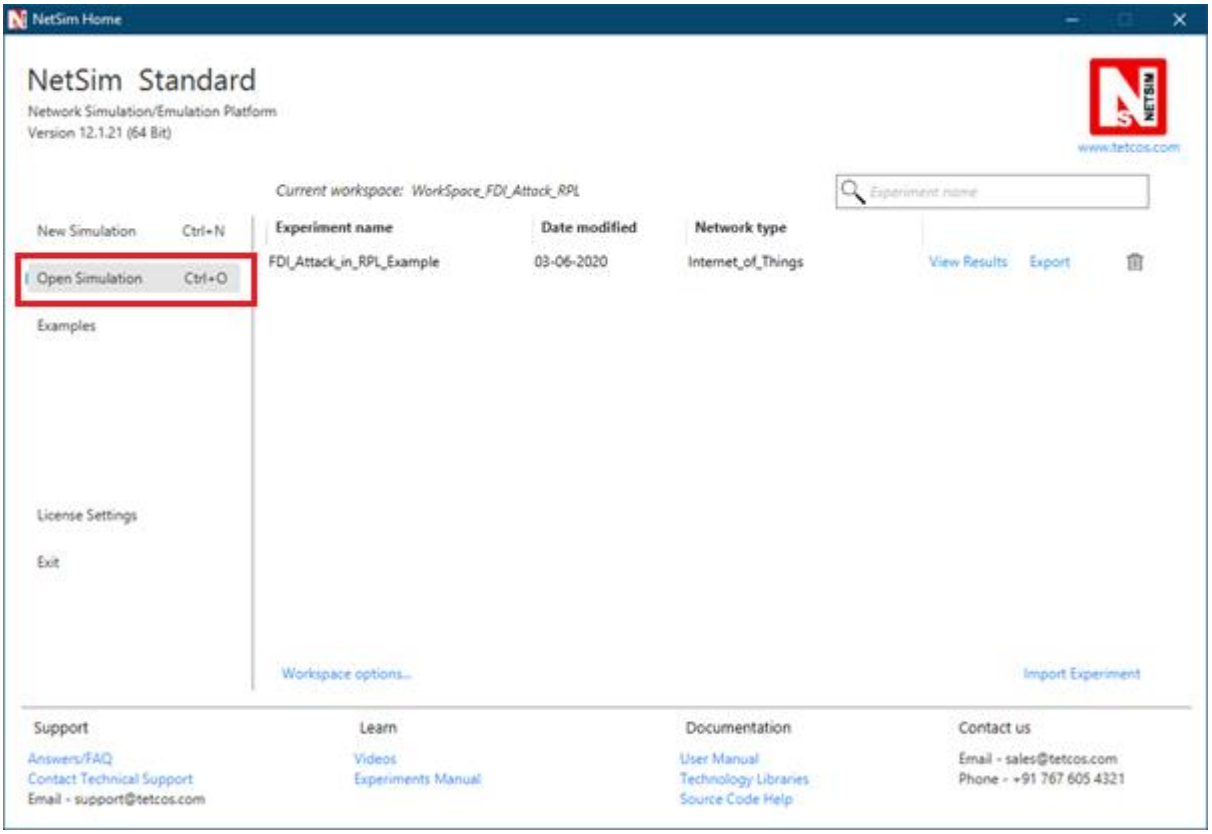
1. **fn_NetSim_RPL_MaliciousNode( )**
   This function is used to identify whether a current device is malicious or not in-order to establish malicious behaviour.
2. **fn_NetSim_RPL_MaliciousRank( )**
   This function is used to give a fake rank to the malicious node.
3. **rpl_false_data_emulation_injection ( )**
   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.
fn_NetSim_RPL_MaliciousRank () is the sink hole attack function.

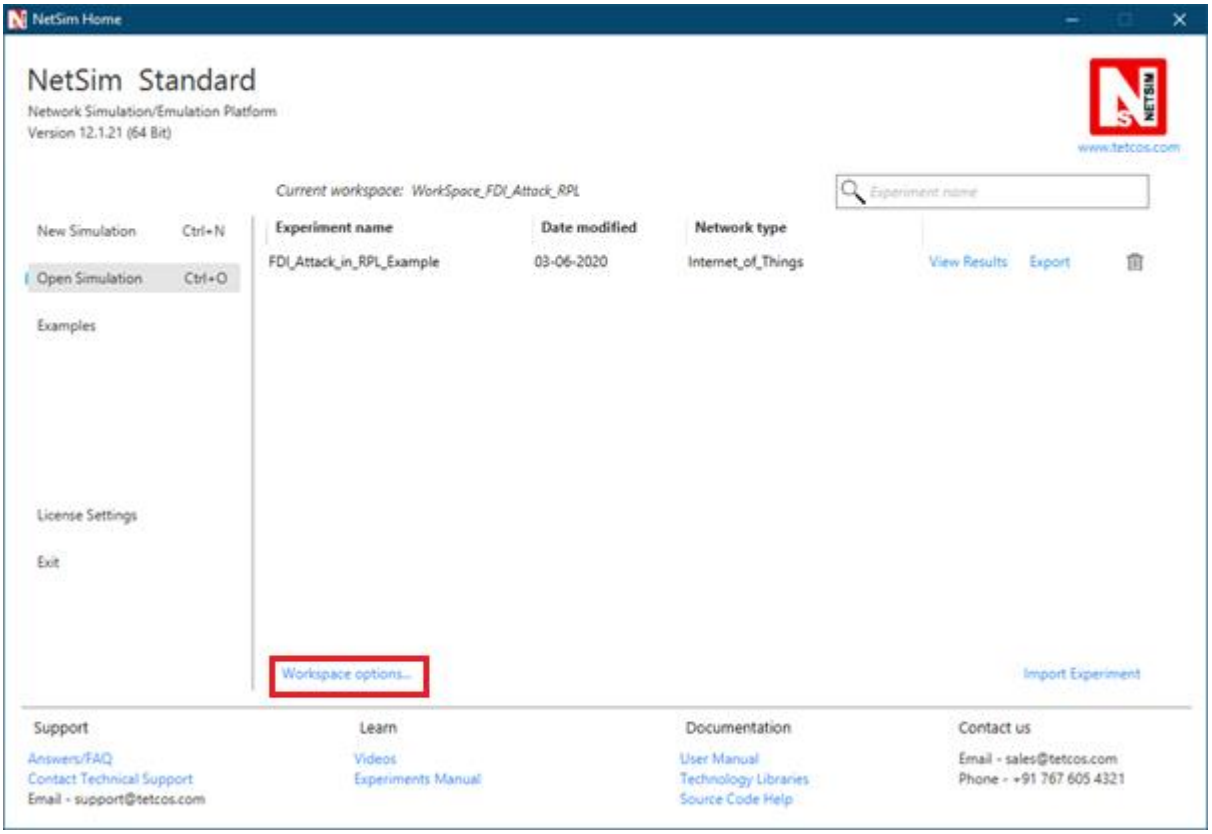
False data injection attack – The malicious node changes the payload of the packet.
rpl_false_data_emulation_injection ( ) is the False data injection
You can set any device as malicious and you can have more than one malicious node in a scenario.
Device id's of malicious nodes can be set inside the fn_NetSim_RPL_MaliciousNode() function.

Steps:

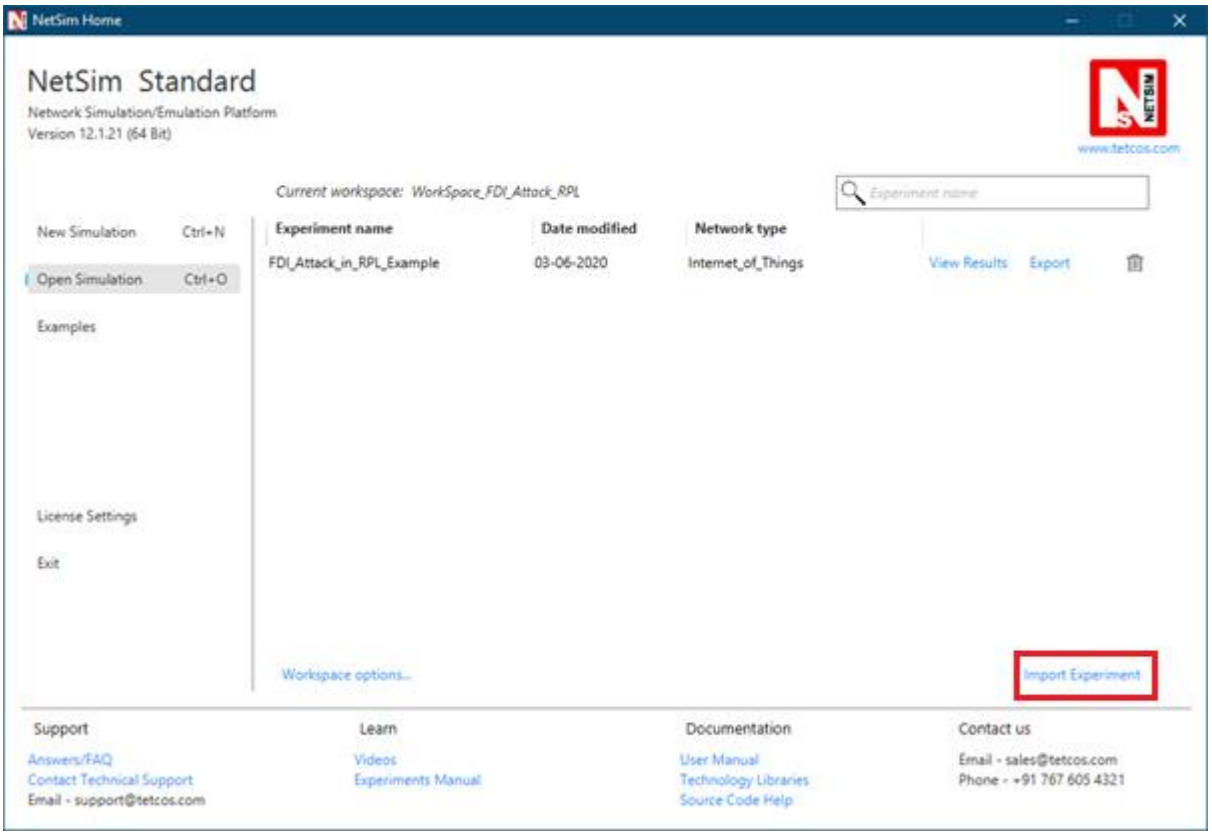
1. After you unzip the downloaded project folder, Open NetSim Home Page click on **Open Simulation** option,



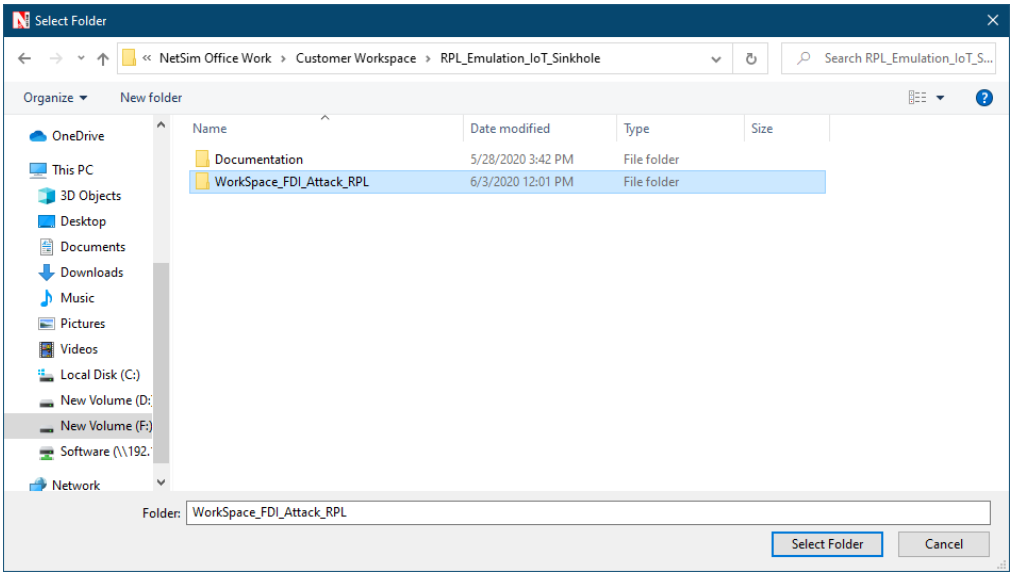
2. Click on **Workspace options**



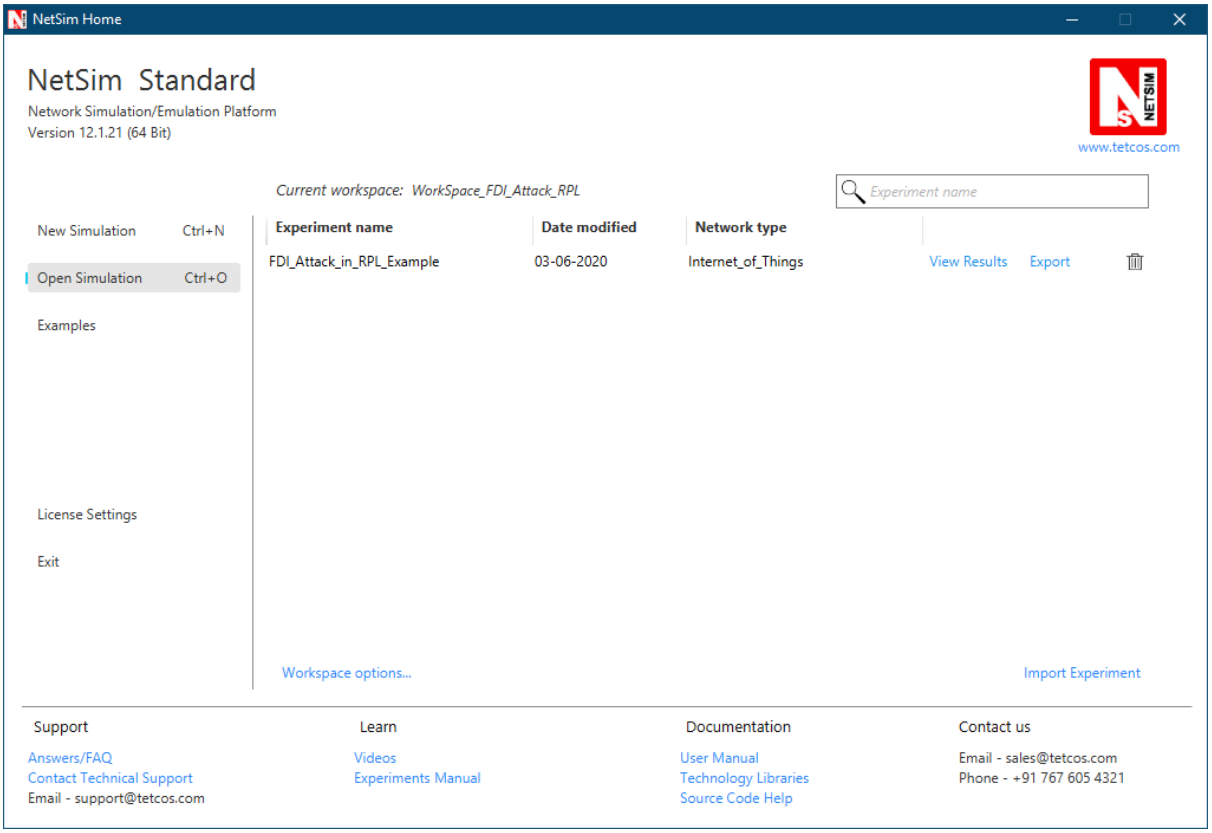
3. Click on **More Options,**



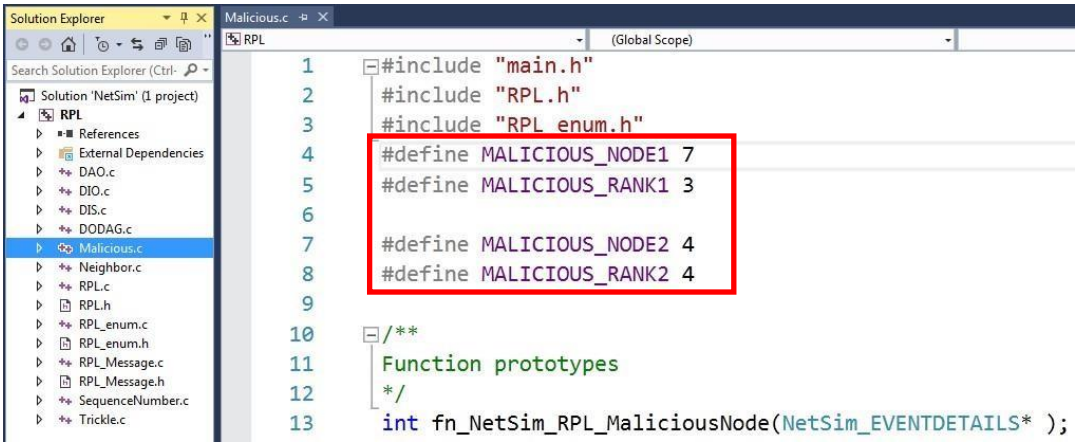
4. Click on **Import**, browse the extracted folder path and go into WorkSpace_FDI_Attack_RPL directory. Click on Select folder button and then on **OK.**



5. Go to home page, Click on **Open Simulation** > **Workspace options** and click on the **Open code** button.



6. Set malicious node id and the fake Rank.

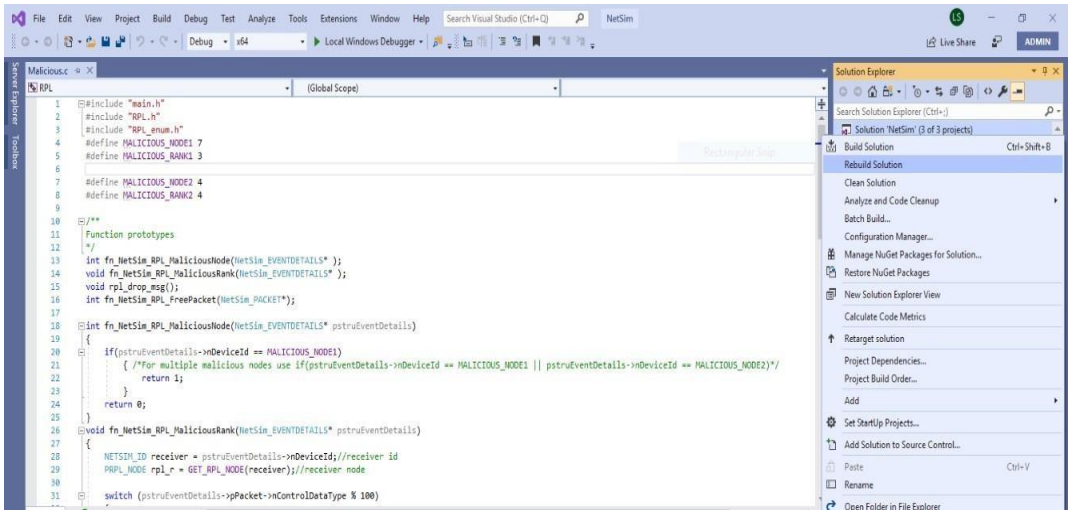


7. Add the code that is highlighted in RPL.c file

```
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();
    }
}
break;
```

False data injection

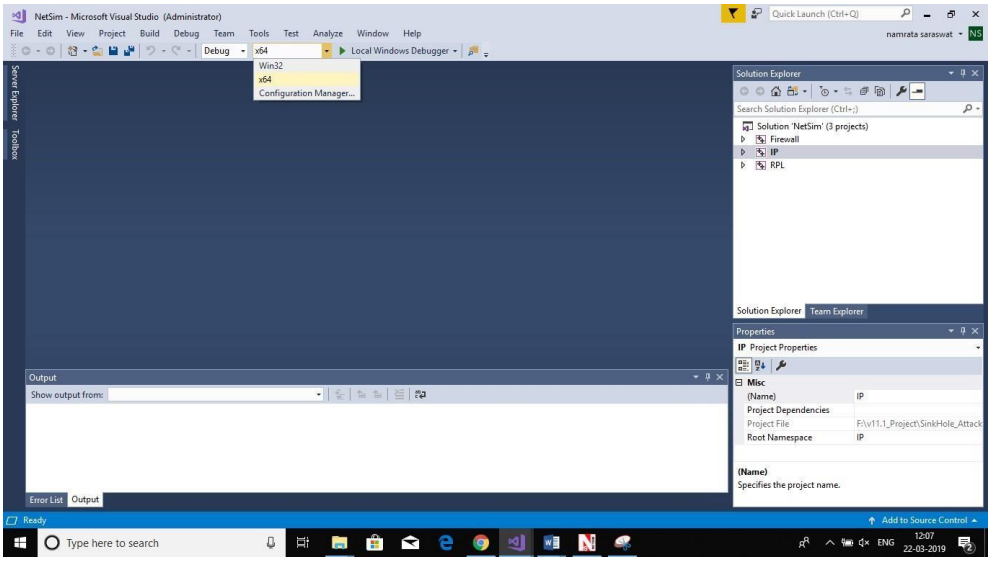
8. Now right click on Solution explorer and select Rebuild.



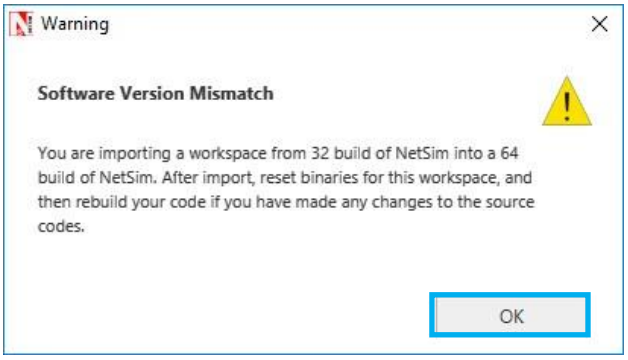
9. Upon rebuilding, **libRPL.dll, libIP.dll,SupportFunction.dll** and **Firewall.dll** will automatically get replaced in the respective bin folders of the current workspace

Note:

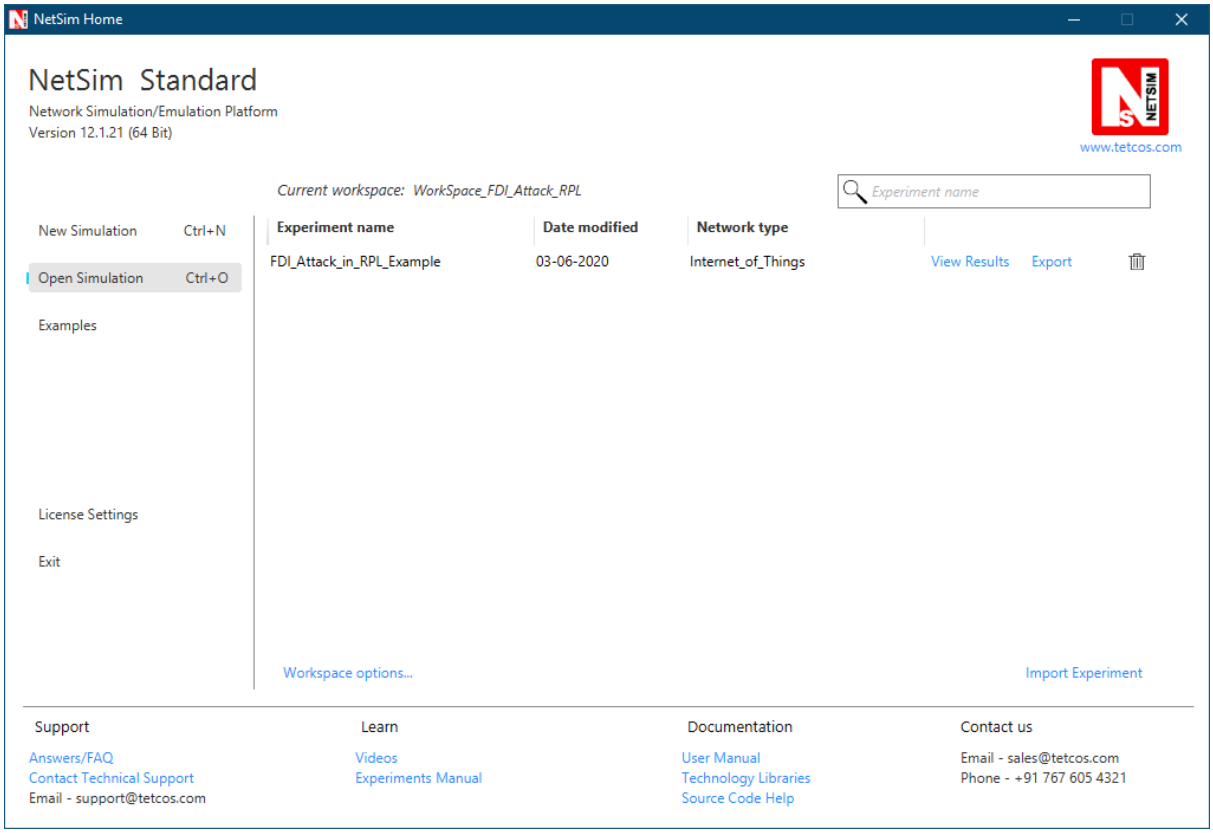
1. Based on whether you are using NetSim 32 bit or 64 bit setup you can configure Visual studio to build 32 bit or 64 bit Dll files respectively as shown below:



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



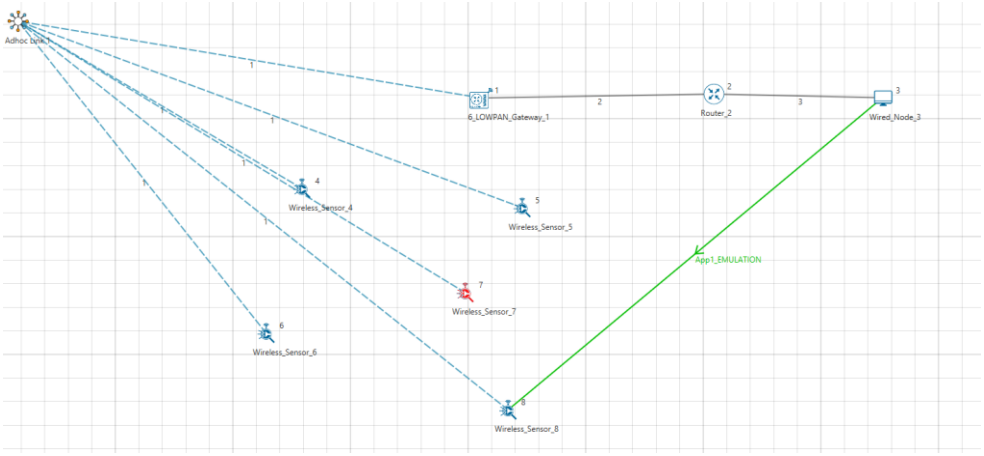
10. Go to NetSim home page, click on **Open Simulation**, Click on **FDI_Attack_in_RPL_Example**.



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

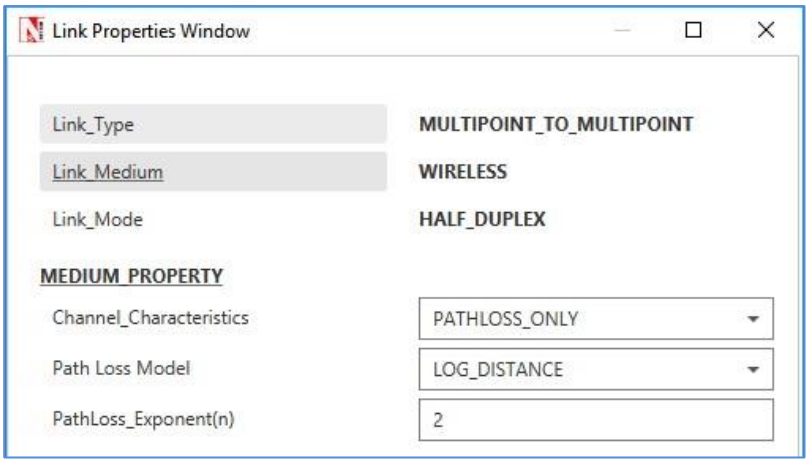
1. Create a network scenario in **IoT (Internet of Things) with UDP running in the Transport Layer and RPL in Network Layer.**

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

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 analyzed using wireshark capture files after emuatlion or during emulation on IBM platform.

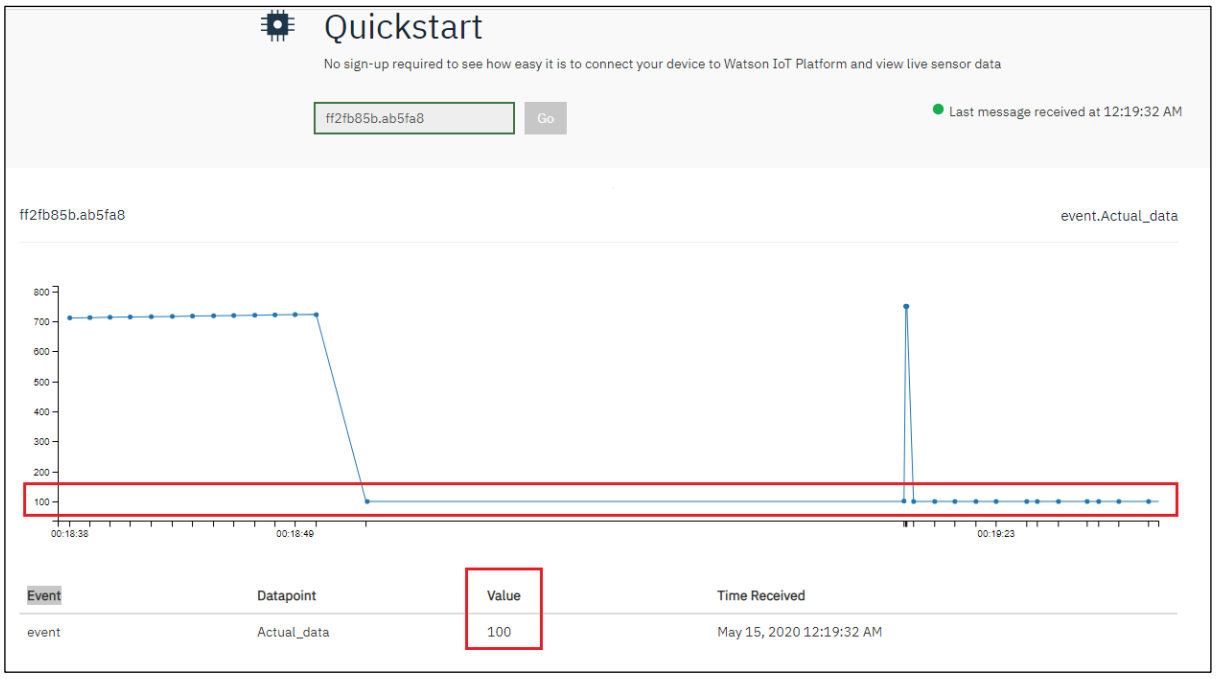


Figure Web console of IBM Platform

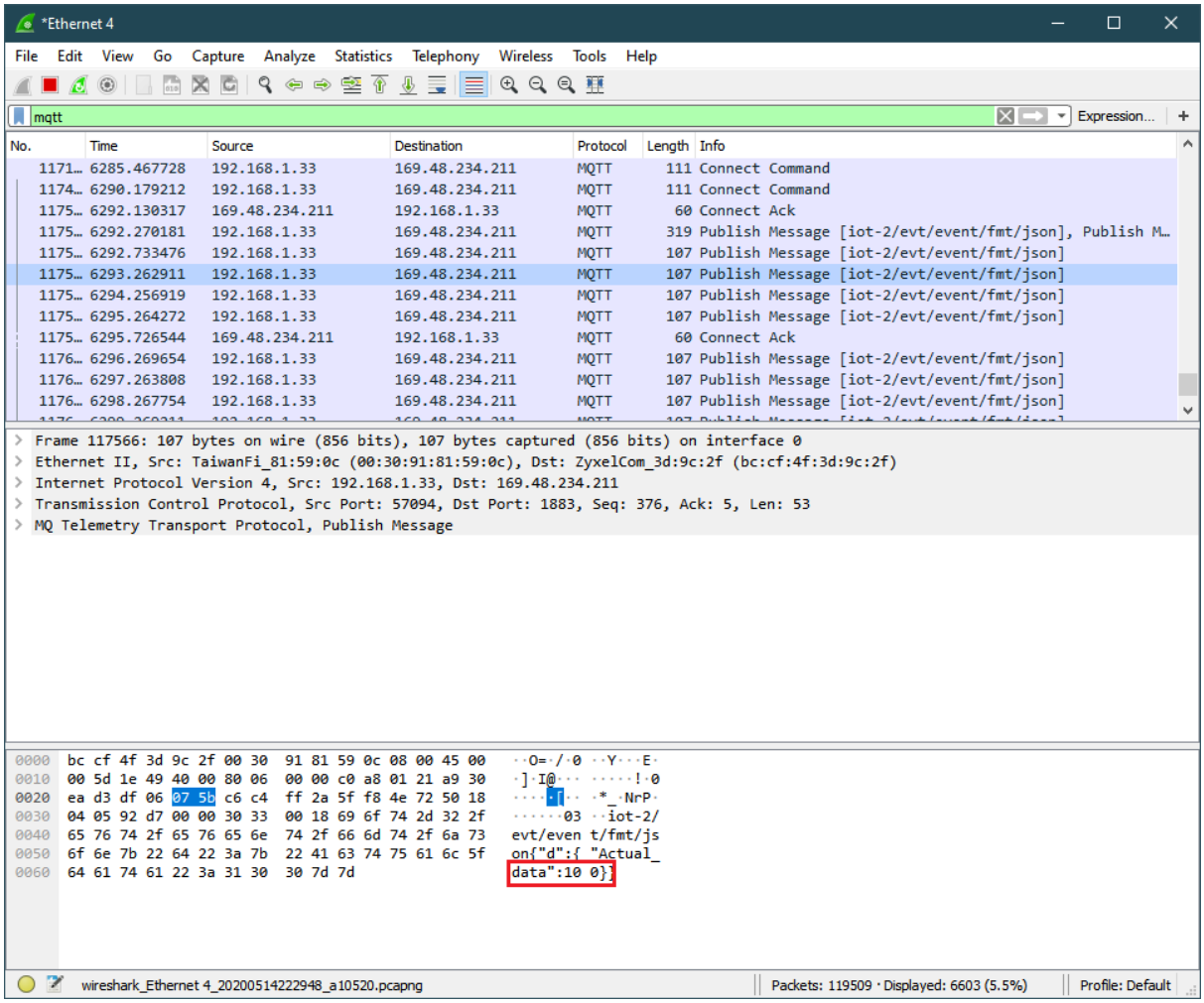


Figure Wireshark

Results and Analysis:

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.