False Data Injection Attack using RPL in IOT

Software: NetSim Standard v13.2, Visual Studio 2022 and Node-RED

Project Download Link:

https://github.com/NetSim-TETCOS/False-Data-Injection-Attack-in-IOT-with-Node-Red-Interfacing_v13.2/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.

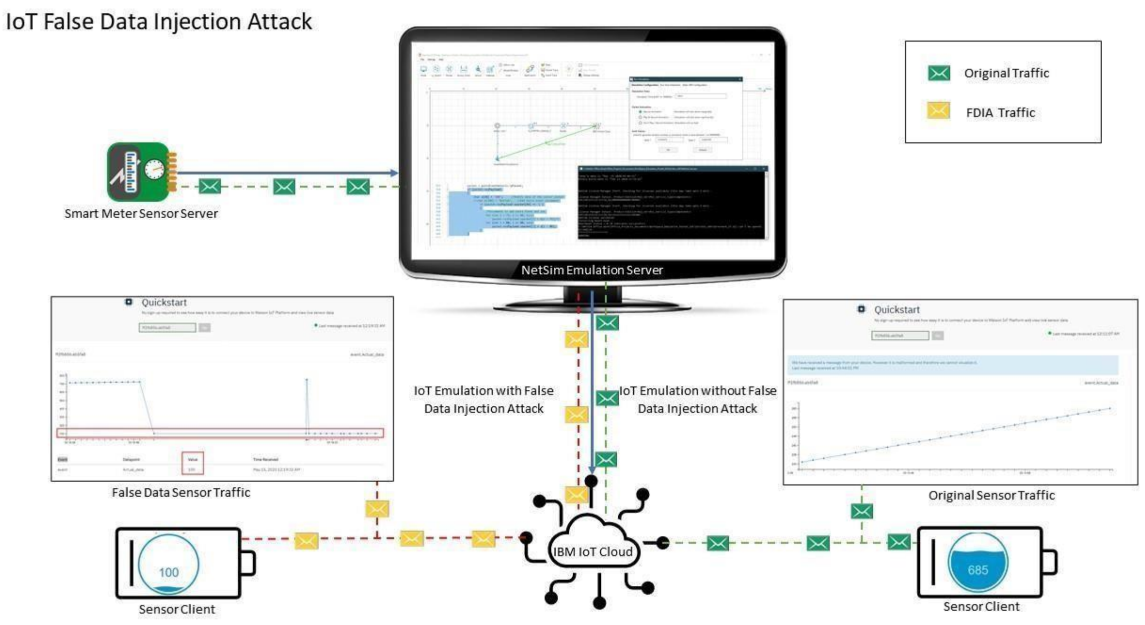

IoT False Data Injection Attack
Original Traffic
FDIA Traffic
Smart Meter Sensor Server
NetSim Emulation Server
Quickstart
False Data Sensor Traffic
IoT Emulation with False Data Injection Attack
IoT Emulation without False Data Injection Attack
Quickstart
Original Sensor Traffic
100
Sensor Client
IBM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

Figure 1: IOT False Data Injection Attack

How are the payloads 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.

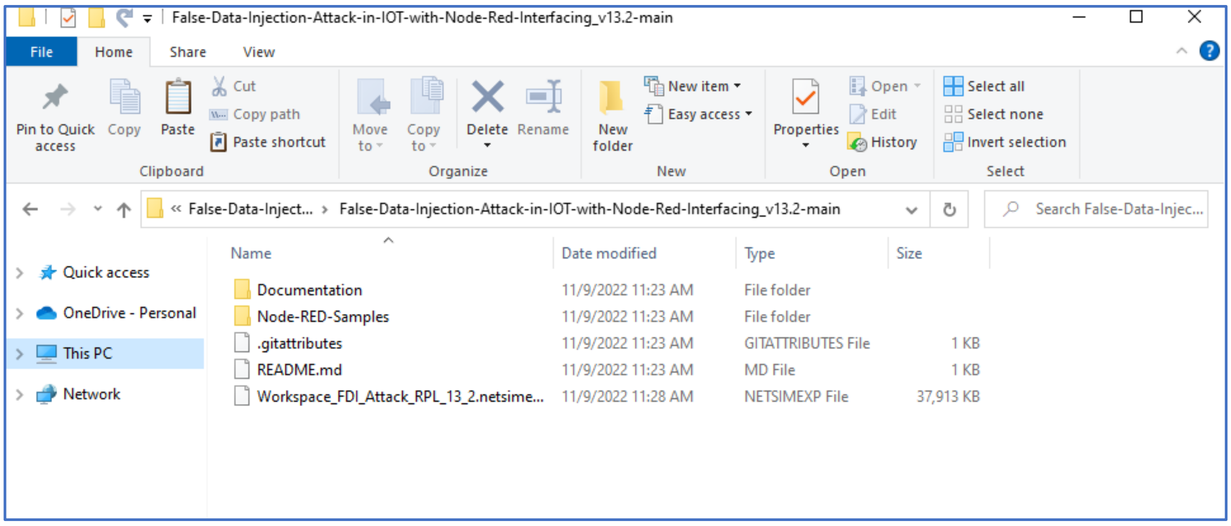
But to perform FDIA we are sending the sensor traffic through NetSim Emulator before it reaches to IBM cloud.

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.
        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.



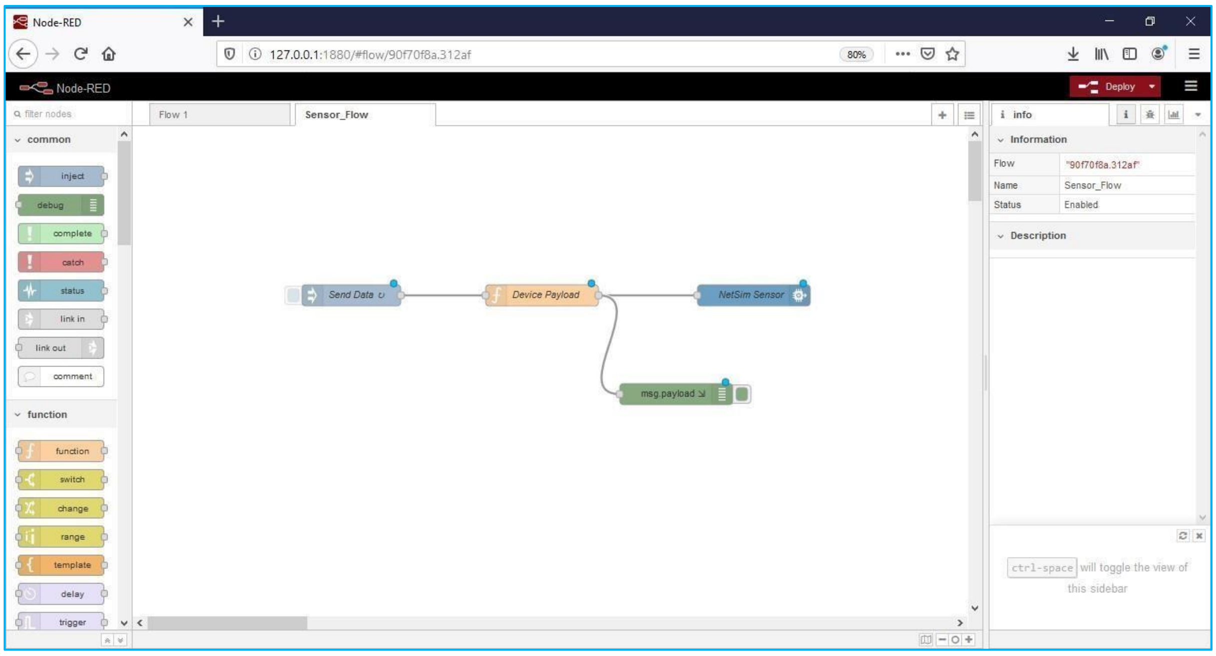
False-Data-Injection-Attack-in-IOT-with-Node-Red-Interfacing_v13.2-main

« False-Data-Inject... › False-Data-Injection-Attack-in-IOT-with-Node-Red-Interfacing_v13.2-main

Name	Date modified	Type	Size
Documentation	11/9/2022 11:23 AM	File folder	
Node-RED-Samples	11/9/2022 11:23 AM	File folder	
.gitattributes	11/9/2022 11:23 AM	GITATTRIBUTES File	1 KB
README.md	11/9/2022 11:23 AM	MD File	1 KB
Workspace_FDI_Attack_RPL_13_2.netsime...	11/9/2022 11:28 AM	NETSIMEXP File	37,913 KB

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

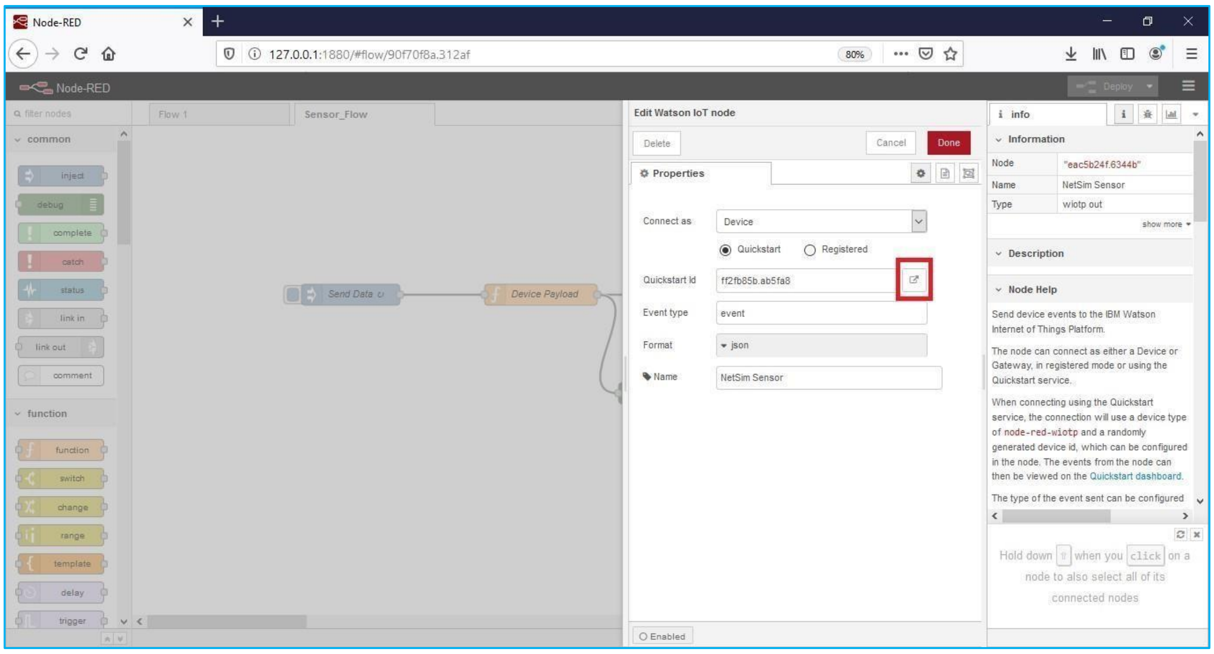
2. Upon importing the flow appears as shown below, in Node-RED:



Flow	"90f70f8a.312af"
Name	Sensor_Flow
Status	Enabled

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:



Connect as	Device
	[✓ Quickstart] Registered
Quickstart Id	ff2fb85b.ab5fa8
Event type	event
Format	json
Name	NetSim Sensor

Node	"eac5b24f.6344b"
Name	NetSim Sensor
Type	wiotp out

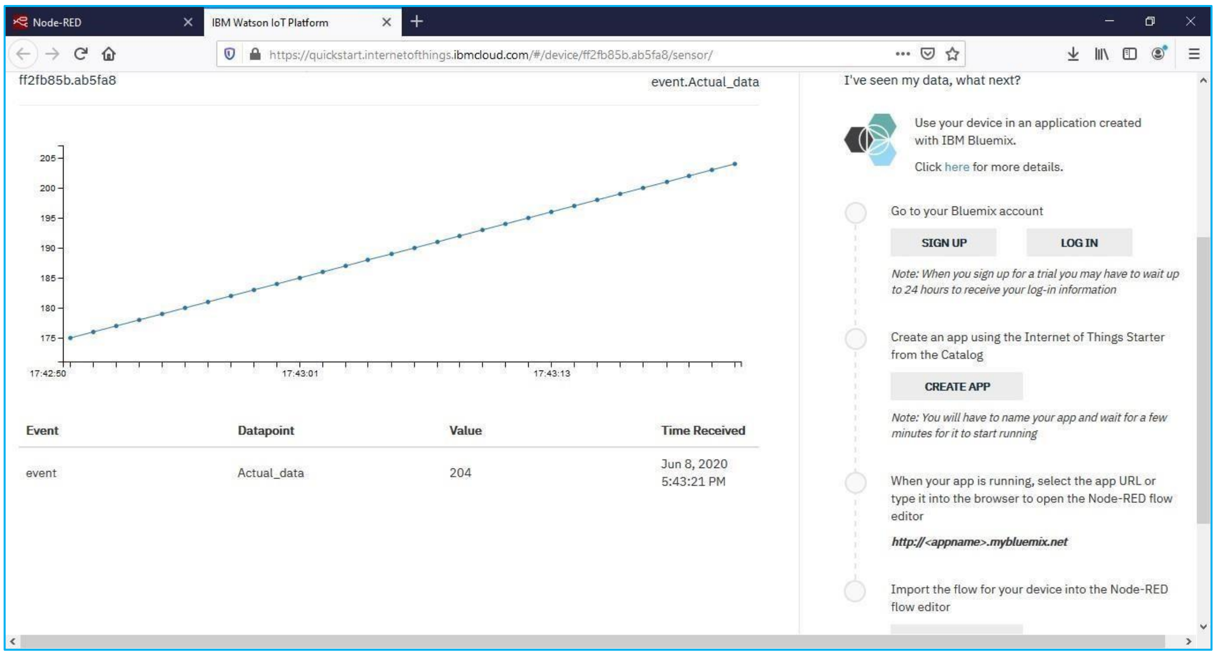
Send device events to the IBM Watson Internet of Things Platform.

The node can connect as either a Device or Gateway, in registered mode or using the Quickstart service.

When connecting using the Quickstart service, the connection will use a device type of node-red-wiotp and a randomly generated device id, which can be configured in the node. The events from the node can then be viewed on the Quickstart dashboard.

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:



Event	Datapoint	Value	Time Received
event	Actual_data	204	Jun 8, 2020 5:43:21 PM

I've seen my data, what next?

Use your device in an application created with IBM Bluemix.

Click here for more details.

Go to your Bluemix account

Note: When you sign up for a trial you may have to wait up to 24 hours to receive your log-in information

Create an app using the Internet of Things Starter from the Catalog

Note: You will have to name your app and wait for a few minutes for it to start running

When your app is running, select the app URL or type it into the browser to open the Node-RED flow editor

http://<appname>.mybluemix.net

Import the flow for your device into the Node-RED flow editor

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.

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:

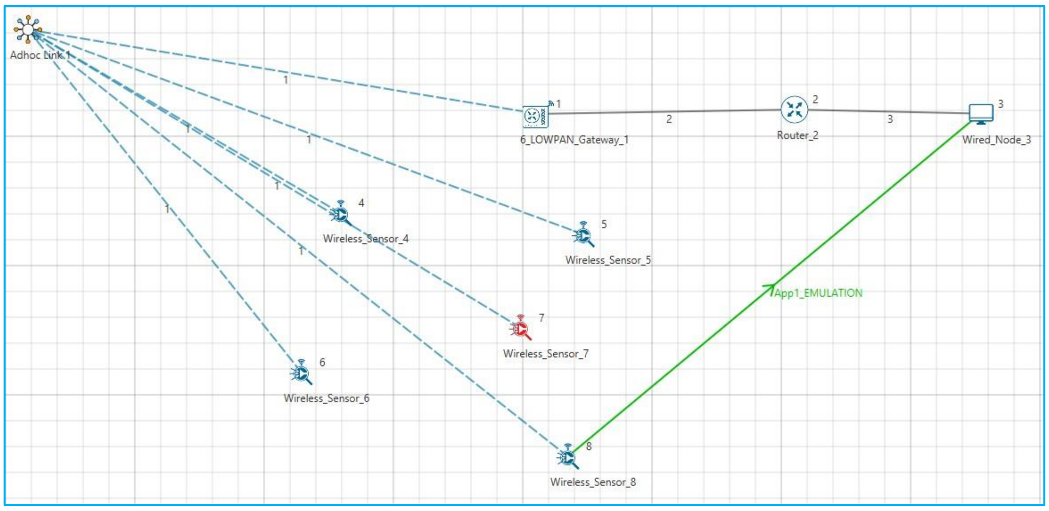


Figure 5: Sensor communicating with the server in NetSim

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

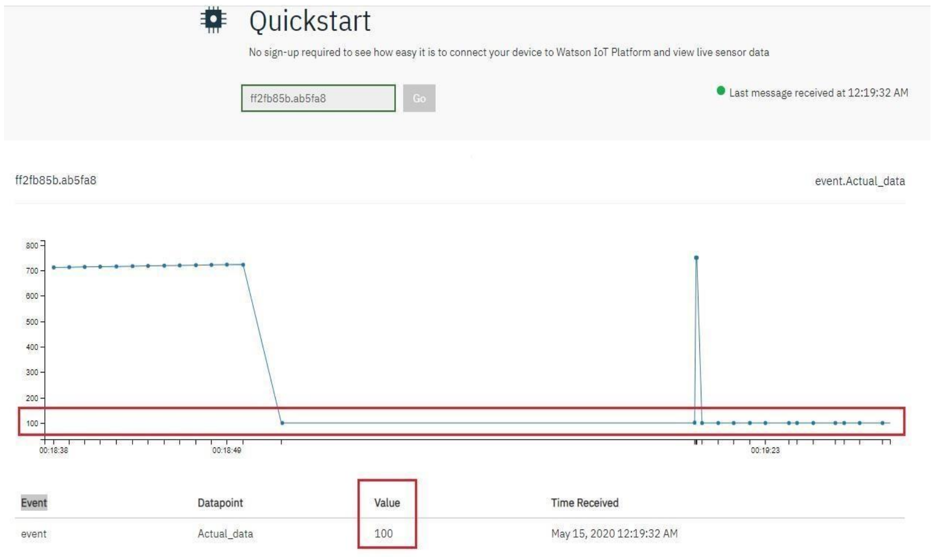
Results and discussion

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.

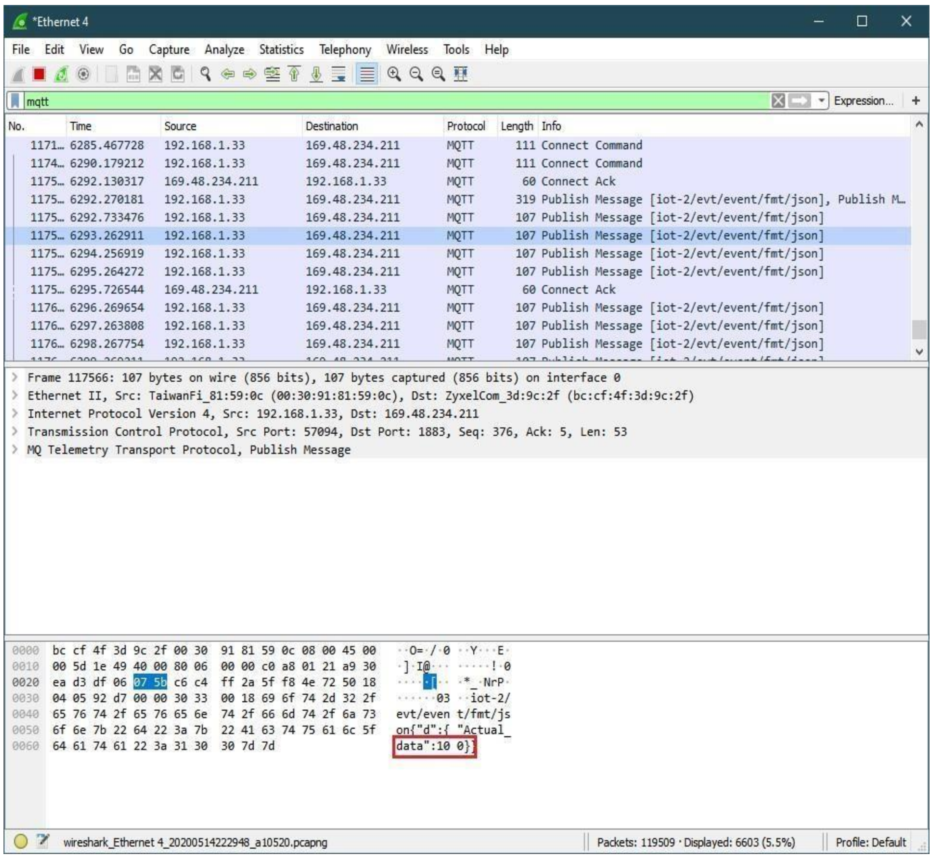
Root is 1 with rank = 1 (Since the Node Id_1 is 6LoWPAN Gateway)
Wireless_Sensor_Node_7(Malicious Node)

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.


Quickstart
No sign-up required to see how easy it is to connect your device to Watson IoT Platform and view live sensor data
ff2fb85b.ab5fa8
Go
Last message received at 12:19:32 AM
ff2fb85b.ab5fa8
event.Actual_data
800
700
600
500
400
300
200
100
00:18:38
00:18:49
00:19:23

Event	Datapoint	Value	Time Received
event	Actual_data	100	May 15, 2020 12:19:32 AM

Figure 6: Web console of IBM Platform


*Ethernet 4
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
mqtt
Expression...

```
No.      Time         Source          Destination     Protocol Length Info
  1171… 6285.467728  192.168.1.33    169.48.234.211  MQTT      111 Connect Command
  1174… 6290.179212  192.168.1.33    169.48.234.211  MQTT      111 Connect Command
  1175… 6292.130317  169.48.234.211  192.168.1.33    MQTT       60 Connect Ack
  1175… 6292.270181  192.168.1.33    169.48.234.211  MQTT      319 Publish Message [iot-2/evt/event/fmt/json], Publish M…
  1175… 6292.733476  192.168.1.33    169.48.234.211  MQTT      107 Publish Message [iot-2/evt/event/fmt/json]
  1175… 6293.262911  192.168.1.33    169.48.234.211  MQTT      107 Publish Message [iot-2/evt/event/fmt/json]
  1175… 6294.256919  192.168.1.33    169.48.234.211  MQTT      107 Publish Message [iot-2/evt/event/fmt/json]
  1175… 6295.264272  192.168.1.33    169.48.234.211  MQTT      107 Publish Message [iot-2/evt/event/fmt/json]
  1175… 6295.726544  169.48.234.211  192.168.1.33    MQTT       60 Connect Ack
  1176… 6296.269654  192.168.1.33    169.48.234.211  MQTT      107 Publish Message [iot-2/evt/event/fmt/json]
  1176… 6297.263808  192.168.1.33    169.48.234.211  MQTT      107 Publish Message [iot-2/evt/event/fmt/json]
  1176… 6298.267754  192.168.1.33    169.48.234.211  MQTT      107 Publish Message [iot-2/evt/event/fmt/json]
```

```
> Frame 117566: 107 bytes on wire (856 bits), 107 bytes captured (856 bits) on interface 0
> Ethernet II, Src: TaiwanFi_81:59:0c (00:30:91:81:59:0c), Dst: ZyxelCom_3d:9c:2f (bc:cf:4f:3d:9c:2f)
> Internet Protocol Version 4, Src: 192.168.1.33, Dst: 169.48.234.211
> Transmission Control Protocol, Src Port: 57094, Dst Port: 1883, Seq: 376, Ack: 5, Len: 53
> MQ Telemetry Transport Protocol, Publish Message
```

```
0000  bc cf 4f 3d 9c 2f 00 30  91 81 59 0c 08 00 45 00   ··O=·/·0 ··Y···E·
0010  00 5d 1e 49 40 00 80 06  00 00 c0 a8 01 21 a9 30   ·]·I@··· ·····!·0
0020  ea d3 df 06 07 5b c6 c4  ff 2a 5f f8 4e 72 50 18   ·····[·· ·*_·NrP·
0030  04 05 92 d7 00 00 30 33  00 18 69 6f 74 2d 32 2f   ······03 ··iot-2/
0040  65 76 74 2f 65 76 65 6e  74 2f 66 6d 74 2f 6a 73   evt/even t/fmt/js
0050  6f 6e 7b 22 64 22 3a 7b  22 41 63 74 75 61 6c 5f   on{"d":{ "Actual_
0060  64 61 74 61 22 3a 31 30  30 7d 7d                  data":10 0}}
```

wireshark_Ethernet 4_20200514222948_a10520.pcapng | Packets: 119509 · Displayed: 6603 (5.5%) | Profile: Default

Figure 7: 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();
    }
}
```