SinkHole Attack in LEACH

Software Recommended: NetSim standard v14.4, Visual Studio 2022

Project Download Link:

https://github.com/NetSim-TETCOS/Sinkhole-attack-in-leach-v14.4/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/support/solutions/articles/14000128666-downloading-and-setting-up-netsim-file-exchange-projects

Introduction :

Leach(Low – Energy Adaptive Clustering hierarchy) is a MAC protocol which is integrated with clustering and a simple routing protocol in wireless sensor networks (WSN). The goal of LEACH is to lower the energy consumption required to create and maintain clusters to improve the lifetime of a wireless sensor network.

This Cross-Layer Protocol is implemented in NetSim in the MAC layer which involves ZigBee Protocol and the Network layer which involves DSR protocol. The clustering of sensors happens in the Network Layer and the cluster head election involves interacting with the MAC layer to obtain the remaining power of the sensors.

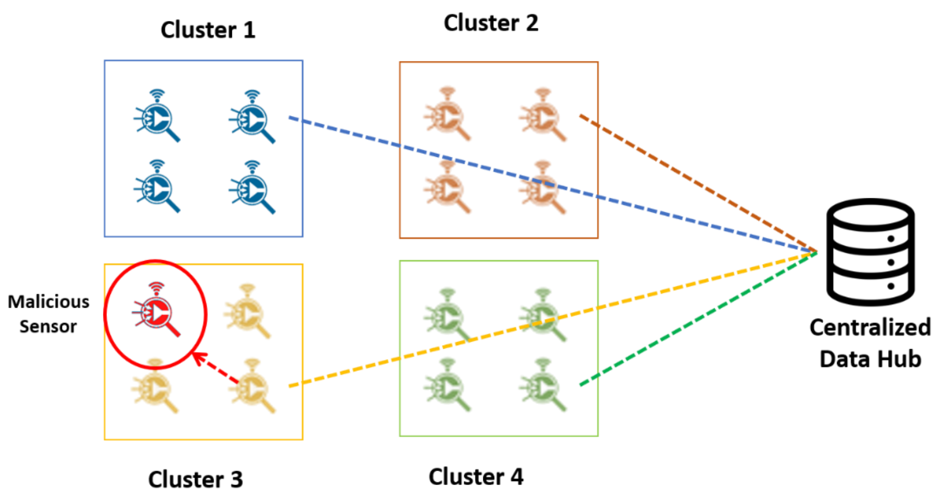


Figure 1: Network Scenario

Sinkhole Attack in Leach Overview:

1. In a smart city, sensor nodes are organized into four clusters, each with a cluster head. The cluster head collects data from nodes within the cluster and forwards it to a central sink node.

2. A malicious sensor node enters one of the clusters and advertises false battery information to become the cluster head instead of the legitimate node.

3. After becoming the cluster head, the malicious sensor node intercepts and redirects data packets from other nodes by responding to route requests, ensuring data passes through it.

4. Consequently, all data packets from the affected cluster are routed through the malicious node, which discards them, preventing any data from reaching the sink node.

Implementation:

A **LEACH.c** file is added to DSR Project.

1. For this implementation of LEACH, the number of Clusters is fixed as 4 and all the 4 clusters are equal. If the user wants to change it, then he/she must also change the static routing for the Cluster Heads and the Cluster Element array accordingly.

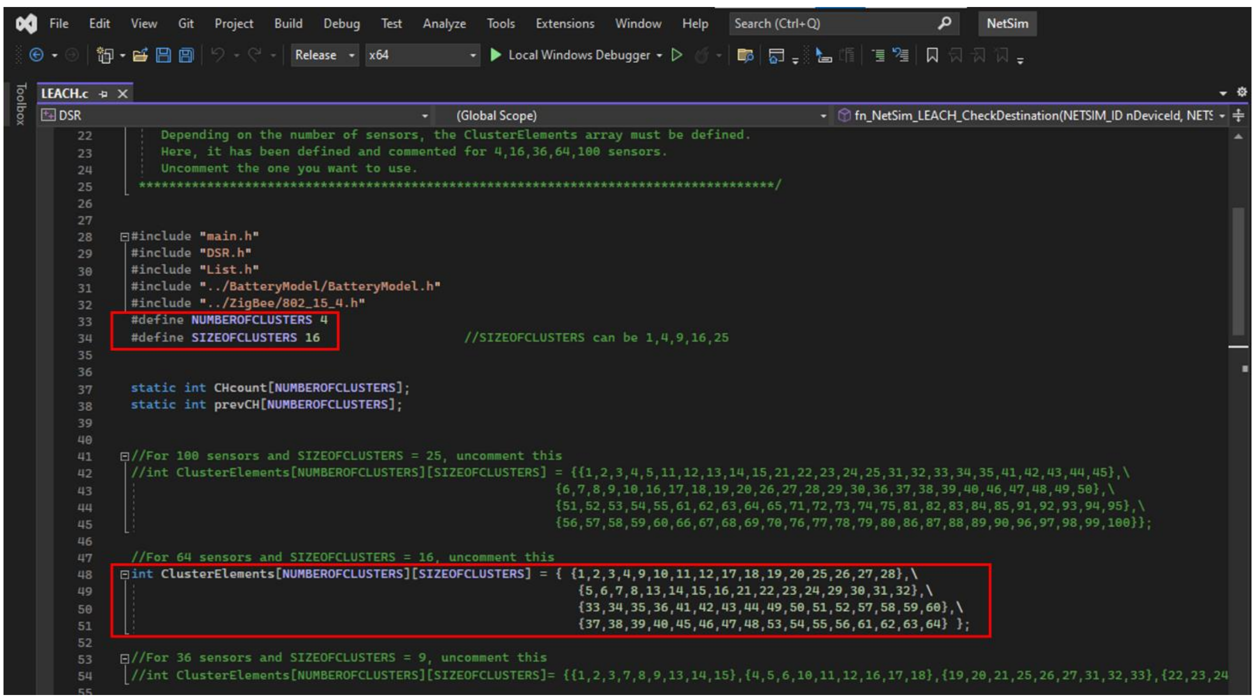


Figure 2:Leach.c file in source code

2. To make 4 equal clusters the number of sensors must be 4,16,36,64,100. Depending on the number of sensors, the Cluster Elements array must be defined. Here, it has been defined and commented on for 4,16,36,64,100 sensors. Uncomment the one you want to use.

The File contains the following functions:

fn_NetSim_LEACH_CheckDestination() // to check whether the current device is the destination or not.

fn_NetSim_LEACH_GetNextHop() // For getting the next hop device id.

fn_NetSim_LEACH_AssignClusterHead() // For electing the Cluster head based on Remaining energy.

fn_NetSim_LEACH_IdentifyCluster() // To determine the cluster to which a sensor belongs.

In this project, we are implementing a sinkhole attack on top of the LEACH project where a malicious node advertises false battery information to become a cluster head. Upon being elected as a cluster head, it attracts network traffic from all its cluster members and destroys the packets without forwarding them to the sink/base station.

A file **malicious.c** is added to the DSR project which contains the following functions:

- **fn_NetSim_DSR_MaliciousNode()** This function is used to identify whether a current device is malicious or not in order to establish malicious behavior.
- **fn_NetSim_DSR_MaliciousProcessSourceRouteOption()** This function is used to drop the received packets if the device is malicious, instead of forwarding the packet to the next hop. You can set any device as malicious, and you can have more than one malicious node in a

scenario. Device IDs of malicious nodes can be set inside the **fn_NetSim_DSR_MaliciousNode()** function.

Note: By default, Malicious Node is set to 22 and NUMBER OF CLUSTERS – 4, SIZE OF CLUSTERS – 16, If changed Rebuild the Solution as shown below:

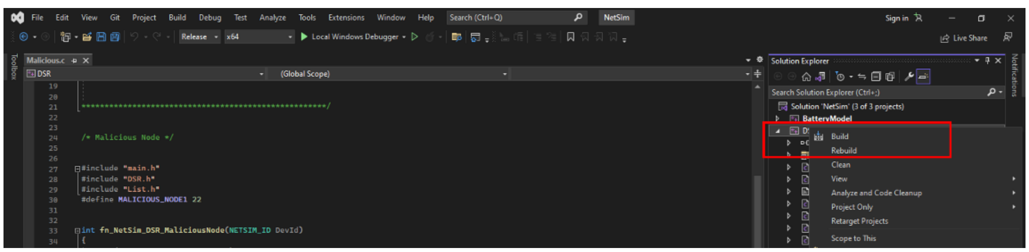


Figure 3:Solution Rebuild in source code

Right Click on the DSR project and Rebuild.

Example:

1. The **Sinkhole-in-LEACH-Workspace** comes with a sample network configuration that is already saved. To open this example, go to Your work in the Home screen of NetSim and click on the **Sinkhole-in-LEACH-Example**. from the list of experiments.
2. The network scenario consists of 64 sensors uniformly placed along with the SINKNODE as shown below.

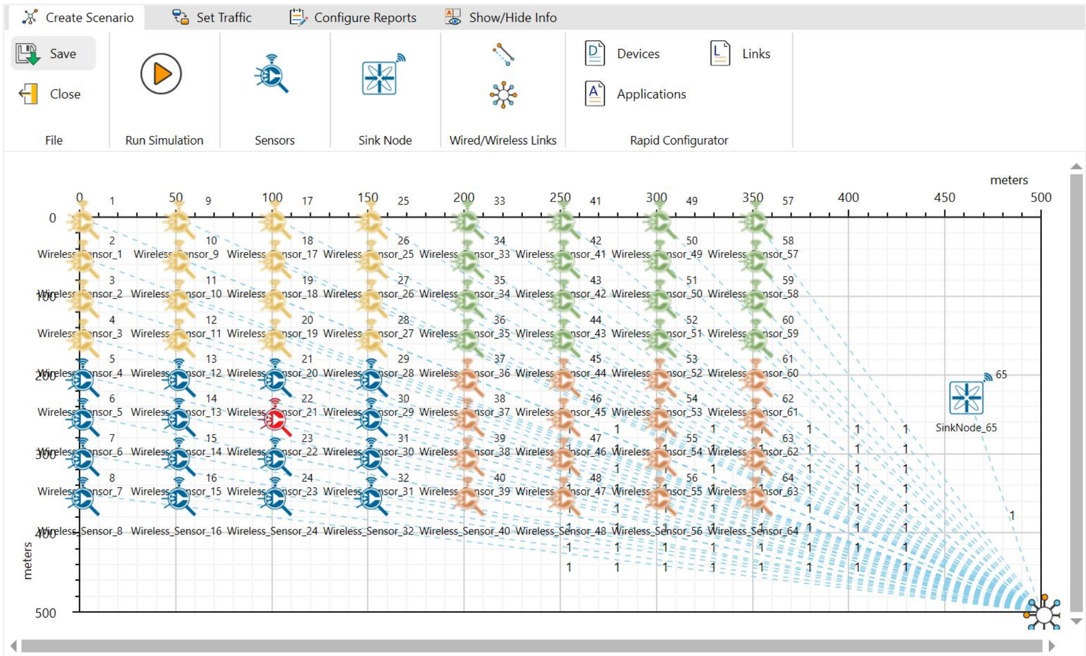


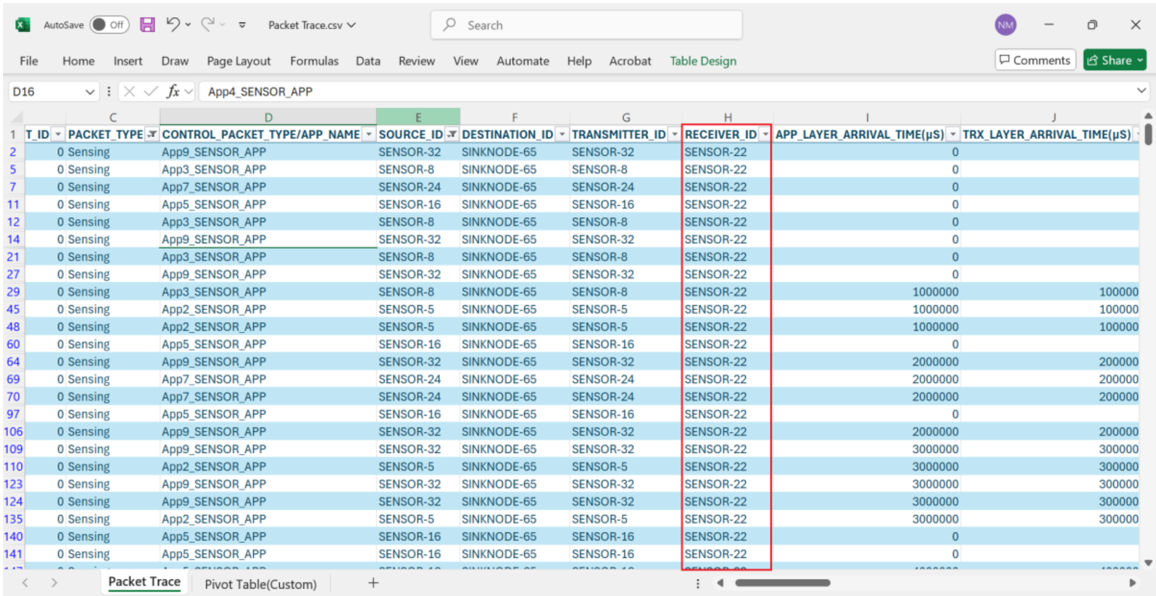
Figure 4:Network setup for Sinkhole attack in LEACH

3. Run the simulation for 100 seconds.

Results and discussion:

- View the packet trace. You will note that the sensors directly start transmitting packets without route establishment since the routes are statically defined in LEACH. You will also note that the cluster heads keep changing dynamically in Clusters 1, 3, and 4. In cluster 2, the cluster members transmit packets to the malicious node (device id 22) since it advertises false battery information to become a cluster head.

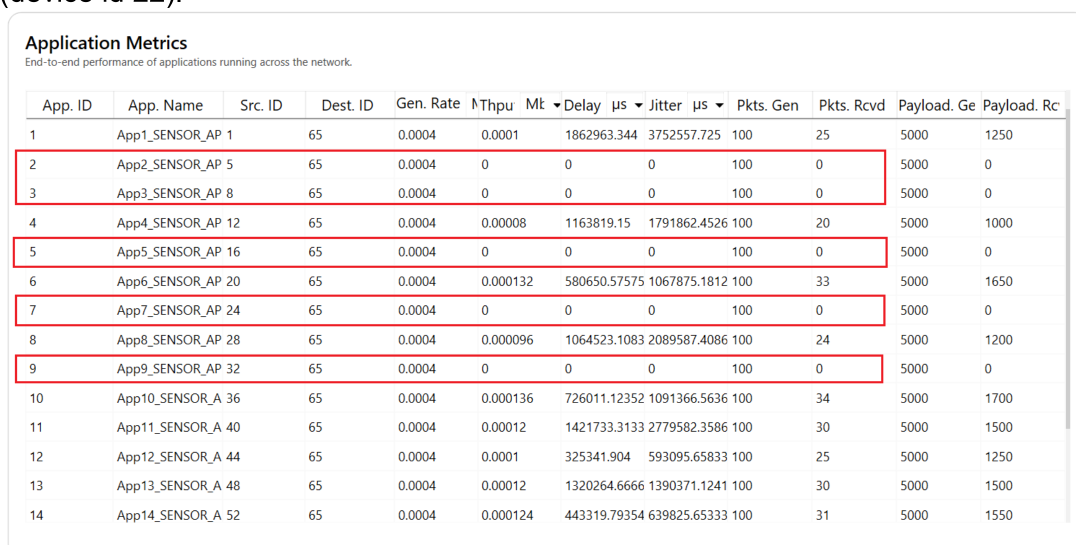
- This can be observed in the Packet trace by applying filters to the Source_ID column by selecting only Sensor-5, 8,16,24,32. You will be able to see that the receiver id is sensor-22 throughout the simulation. All the nodes in Cluster2 are sending data packets to the malicious node (Sensor-22) since it is the Cluster Head



	T_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID	APP_LAYER_ARRIVAL_TIME(μS)	TRX_LAYER_ARRIVAL_TIME(μS)
2	0	Sensing	App9_SENSOR_APP	SENSOR-32	SINKNODE-65	SENSOR-32	SENSOR-22	0	
5	0	Sensing	App3_SENSOR_APP	SENSOR-8	SINKNODE-65	SENSOR-8	SENSOR-22	0	
7	0	Sensing	App7_SENSOR_APP	SENSOR-24	SINKNODE-65	SENSOR-24	SENSOR-22	0	
11	0	Sensing	App5_SENSOR_APP	SENSOR-16	SINKNODE-65	SENSOR-16	SENSOR-22	0	
12	0	Sensing	App3_SENSOR_APP	SENSOR-8	SINKNODE-65	SENSOR-8	SENSOR-22	0	
14	0	Sensing	App9_SENSOR_APP	SENSOR-32	SINKNODE-65	SENSOR-32	SENSOR-22	0	
21	0	Sensing	App3_SENSOR_APP	SENSOR-8	SINKNODE-65	SENSOR-8	SENSOR-22	0	
27	0	Sensing	App9_SENSOR_APP	SENSOR-32	SINKNODE-65	SENSOR-32	SENSOR-22	0	
29	0	Sensing	App3_SENSOR_APP	SENSOR-8	SINKNODE-65	SENSOR-8	SENSOR-22	1000000	100000
45	0	Sensing	App2_SENSOR_APP	SENSOR-5	SINKNODE-65	SENSOR-5	SENSOR-22	1000000	100000
48	0	Sensing	App2_SENSOR_APP	SENSOR-5	SINKNODE-65	SENSOR-5	SENSOR-22	1000000	100000
60	0	Sensing	App5_SENSOR_APP	SENSOR-16	SINKNODE-65	SENSOR-16	SENSOR-22	0	
64	0	Sensing	App9_SENSOR_APP	SENSOR-32	SINKNODE-65	SENSOR-32	SENSOR-22	2000000	200000
69	0	Sensing	App7_SENSOR_APP	SENSOR-24	SINKNODE-65	SENSOR-24	SENSOR-22	2000000	200000
70	0	Sensing	App7_SENSOR_APP	SENSOR-24	SINKNODE-65	SENSOR-24	SENSOR-22	2000000	200000
97	0	Sensing	App5_SENSOR_APP	SENSOR-16	SINKNODE-65	SENSOR-16	SENSOR-22	0	
106	0	Sensing	App9_SENSOR_APP	SENSOR-32	SINKNODE-65	SENSOR-32	SENSOR-22	2000000	200000
109	0	Sensing	App9_SENSOR_APP	SENSOR-32	SINKNODE-65	SENSOR-32	SENSOR-22	3000000	300000
110	0	Sensing	App2_SENSOR_APP	SENSOR-5	SINKNODE-65	SENSOR-5	SENSOR-22	3000000	300000
123	0	Sensing	App9_SENSOR_APP	SENSOR-32	SINKNODE-65	SENSOR-32	SENSOR-22	3000000	300000
124	0	Sensing	App9_SENSOR_APP	SENSOR-32	SINKNODE-65	SENSOR-32	SENSOR-22	3000000	300000
135	0	Sensing	App2_SENSOR_APP	SENSOR-5	SINKNODE-65	SENSOR-5	SENSOR-22	3000000	300000
140	0	Sensing	App5_SENSOR_APP	SENSOR-16	SINKNODE-65	SENSOR-16	SENSOR-22	0	
141	0	Sensing	App5_SENSOR_APP	SENSOR-16	SINKNODE-65	SENSOR-16	SENSOR-22	0	

Figure 5: Packet trace file referring to malicious node reception of transmitted packets

- This will have a direct impact on the Application Throughput which can be observed in the Application Metrics table present in the NetSim Simulation Results window. The throughput for applications 2, 3, 5, 7 and 9 is zero since the source ids belong to cluster2 having a malicious node (device id 22).



Application Metrics

End-to-end performance of applications running across the network.

App. ID	App. Name	Src. ID	Dest. ID	Gen. Rate	Thpu Mt	Delay μs	Jitter μs	Pkts. Gen	Pkts. Rcvd	Payload. Ge	Payload. Rc
1	App1_SENSOR_AP	1	65	0.0004	0.0001	1862963.344	3752557.725	100	25	5000	1250
2	App2_SENSOR_AP	5	65	0.0004	0	0	0	100	0	5000	0
3	App3_SENSOR_AP	8	65	0.0004	0	0	0	100	0	5000	0
4	App4_SENSOR_AP	12	65	0.0004	0.00008	1163819.15	1791862.4526	100	20	5000	1000
5	App5_SENSOR_AP	16	65	0.0004	0	0	0	100	0	5000	0
6	App6_SENSOR_AP	20	65	0.0004	0.000132	580650.57575	1067875.1812	100	33	5000	1650
7	App7_SENSOR_AP	24	65	0.0004	0	0	0	100	0	5000	0
8	App8_SENSOR_AP	28	65	0.0004	0.000096	1064523.1083	2089587.4086	100	24	5000	1200
9	App9_SENSOR_AP	32	65	0.0004	0	0	0	100	0	5000	0
10	App10_SENSOR_A	36	65	0.0004	0.000136	726011.12352	1091366.5636	100	34	5000	1700
11	App11_SENSOR_A	40	65	0.0004	0.00012	1421733.3133	2779582.3586	100	30	5000	1500
12	App12_SENSOR_A	44	65	0.0004	0.0001	325341.904	593095.65833	100	25	5000	1250
13	App13_SENSOR_A	48	65	0.0004	0.00012	1320264.6666	1390371.1241	100	30	5000	1500
14	App14_SENSOR_A	52	65	0.0004	0.000124	443319.79354	639825.65333	100	31	5000	1550

Figure 6: Simulation results window.