Sink Hole Attack in MANET using DSR

Software: NetSim Standard v14.0, Visual Studio 2022

Project Download Link:

https://github.com/NetSim-TETCOS/Sinkhole-Attack-in-DSR-v14.0/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:

Sinkhole attack is one of the most severe attacks in wireless Ad hoc networks. In sinkhole Attack, a compromised node or malicious node advertises wrong routing information to pretend itself as a specific node and receives whole network traffic. After receiving the whole network traffic, it can either modify the packet information or drop it to make the network complicated. Sinkhole attacks affect the performance of Ad hoc network protocols such as the DSR protocol.

Real-World Context:

In a real-world scenario, a small ad-hoc network which consist of a few laptops and mobile devices wirelessly connected to each other. When one node in this network is malicious, the impact can be significant.

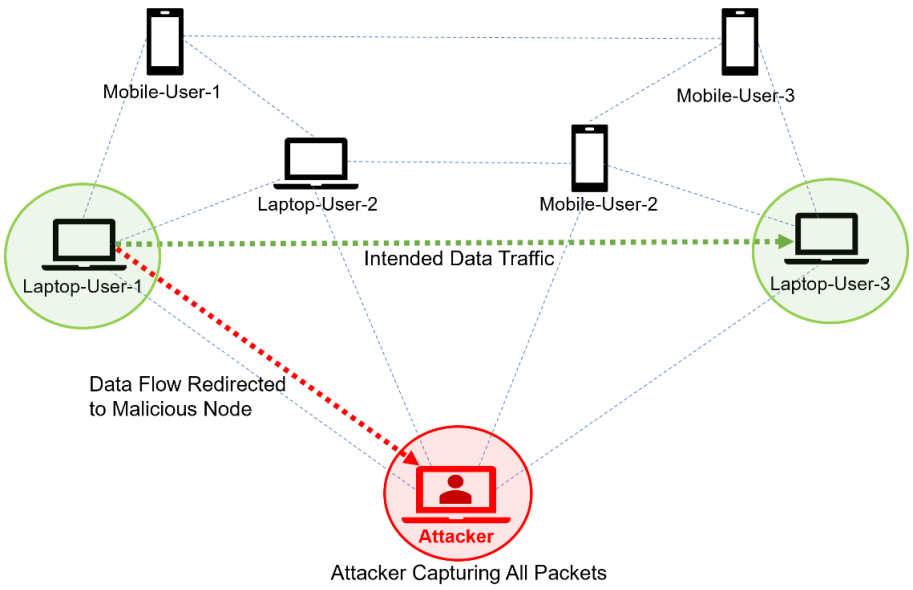


Figure 1: Real world scenario for Sink Hole Attack in MANET using DSR

Sinkhole Attack Overview:

- The Malicious nodes **(Attacker)** enters the network and starts intercepting and diverting network traffic.
- When **Laptop-User-1** attempts to communicate with **Laptop-User-3** the Malicious Node **(Attacker)**intercepts the communication by responding to route requests and diverting data packets.
- As a result, the data that **Laptop-User-1** intended for **Laptop-User-3** gets redirected to the Malicious Node **(Attacker)**. This could be sensitive information, files, or any data being exchanged.
- The Malicious Node **(Attacker)** intentionally discards all incoming data packets, thereby effectively preventing the data from reaching its intended destination.

NetSim's Role:

We use NetSim to simulate and analyze sinkhole attacks, aiding in the understanding of security vulnerabilities. NetSim offers us the means to replicate real-world scenarios within a controlled, virtual environment.

This document will provide a comprehensive overview of our project's objectives for studying the sinkhole attacks.

Implementation in DSR:

- In DSR the source broadcasts the RREQ packet during Route Discovery.
- The destination on receiving the RREQ packet replies with an RREP packet containing the route to reach the destination.

- But Intermediate nodes can also send RREP packets to the source if they have a route to the destination in their route cache.
- Using this as an advantage the malicious node adds a fake route entry into its route cache with the destination node as its next hop.
- On receiving the RREQ packet from the source the malicious node sends a fake RREP packet with the fake route.
- The source node on receiving this fake RREP packet observes this as a better route to the destination.
- All the Network Traffic is attracted towards the Sinkhole (Malicious Node) and it can either modify the packet Information or simply drop the packet.

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_MaliciousRouteAddToCache();** //This function is used to add a fake route entry into the route cache of the malicious device with its next hop as the destination.
- **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 node, 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.

Steps to simulate:

1. Open the Source codes in Visual Studio by going to Your work in home screen of NetSim -> Source Code and click on the Open code.
2. Expand the DSR project and open the Malicious.c file and set the malicious node id.

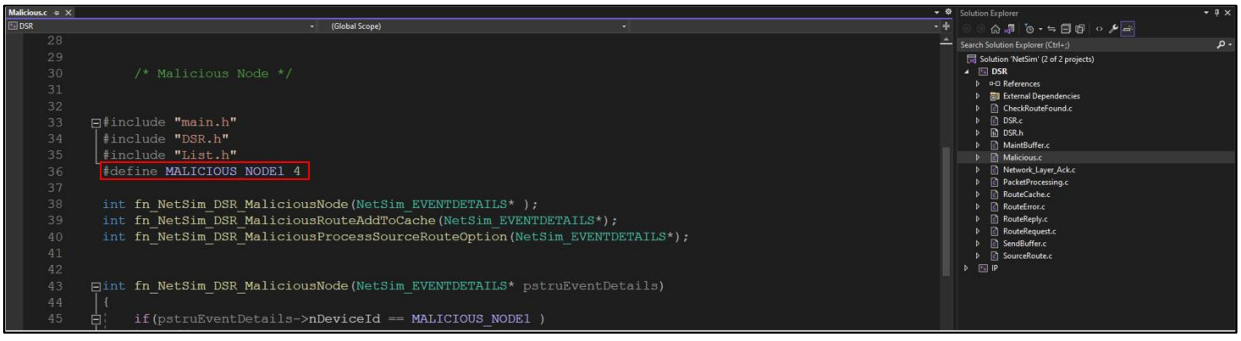


Figure 2: Set Malicious Node in malicious .c file

3. Now right-click on the DSR project and rebuild it.

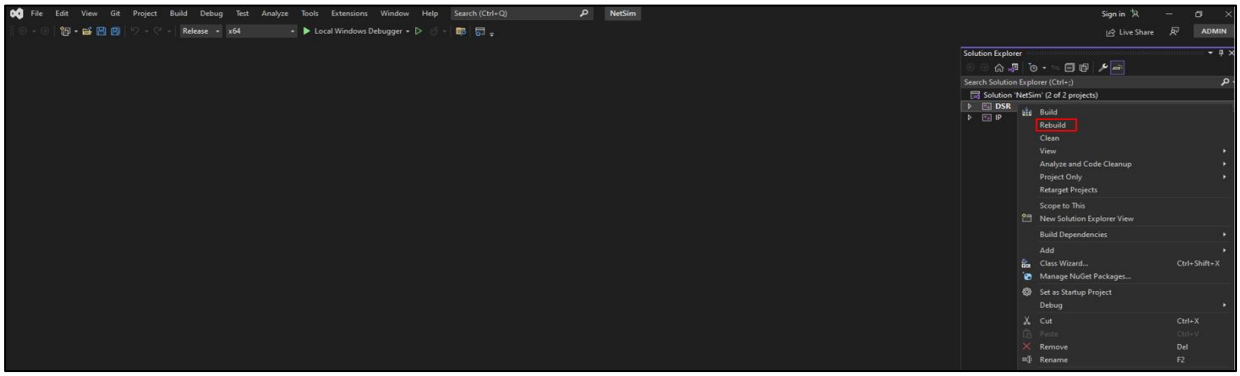


Figure 3: Screenshot of NetSim project source code in Visual Studio

4. Upon rebuilding, libDSR.dll will automatically get updated in the respective bin folder of the current workspace.

Example

1. The **SINK_HOLE_ATTACK_DSR_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 **SINK_HOLE_ATTACK_DSR_Example** from the list of experiments.
2. The network consists of 7 Wireless nodes with properties configured as shown below:

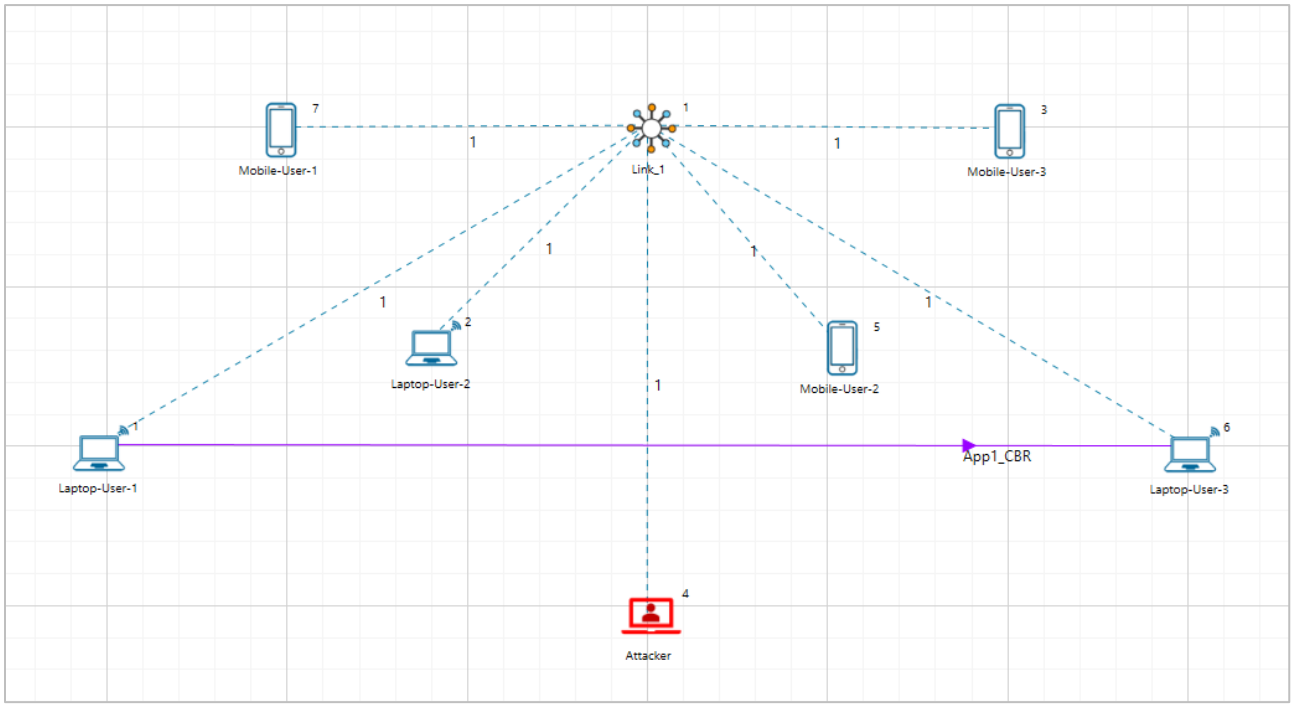


Figure 4: Network Topology for Sink Hole Attack in MANET using DSR

3. Set the Application properties

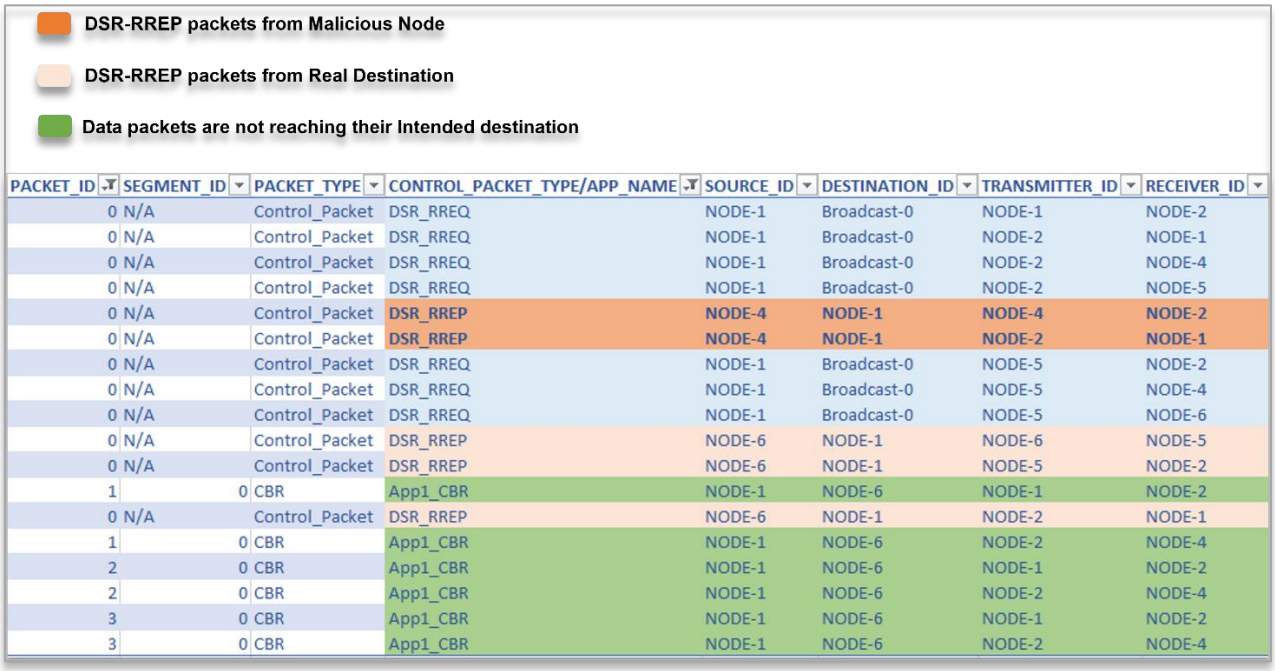
Application Properties	
Source ID	**1**
Destination ID	**6**

Table 1: Application Properties

4. In the Link Set the Channel Characteristics: **Pathloss only**, Path Loss Model: **Log Distance**, Path Loss Exponent: **3**
5. Run the Simulation for 100 seconds.

Results and discussion:

In the packet trace, we can see the impact of a malicious node within the MANET, showcasing how it disrupts the normal data transfer process



PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE/APP_NAME	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID
0	N/A	Control_Packet	DSR_RREQ	NODE-1	Broadcast-0	NODE-1	NODE-2
0	N/A	Control_Packet	DSR_RREQ	NODE-1	Broadcast-0	NODE-2	NODE-1
0	N/A	Control_Packet	DSR_RREQ	NODE-1	Broadcast-0	NODE-2	NODE-4
0	N/A	Control_Packet	DSR_RREQ	NODE-1	Broadcast-0	NODE-2	NODE-5
0	N/A	Control_Packet	DSR_RREP	NODE-4	NODE-1	NODE-4	NODE-2
0	N/A	Control_Packet	DSR_RREP	NODE-4	NODE-1	NODE-2	NODE-1
0	N/A	Control_Packet	DSR_RREQ	NODE-1	Broadcast-0	NODE-5	NODE-2
0	N/A	Control_Packet	DSR_RREQ	NODE-1	Broadcast-0	NODE-5	NODE-4
0	N/A	Control_Packet	DSR_RREQ	NODE-1	Broadcast-0	NODE-5	NODE-6
0	N/A	Control_Packet	DSR_RREP	NODE-6	NODE-1	NODE-6	NODE-5
0	N/A	Control_Packet	DSR_RREP	NODE-6	NODE-1	NODE-5	NODE-2
1	0	CBR	App1_CBR	NODE-1	NODE-6	NODE-1	NODE-2
0	N/A	Control_Packet	DSR_RREP	NODE-6	NODE-1	NODE-2	NODE-1
1	0	CBR	App1_CBR	NODE-1	NODE-6	NODE-2	NODE-4
2	0	CBR	App1_CBR	NODE-1	NODE-6	NODE-1	NODE-2
2	0	CBR	App1_CBR	NODE-1	NODE-6	NODE-2	NODE-4
3	0	CBR	App1_CBR	NODE-1	NODE-6	NODE-1	NODE-2
3	0	CBR	App1_CBR	NODE-1	NODE-6	NODE-2	NODE-4

Figure 5: Analysis of results using packet trace

- From the screenshot above You will find that the malicious node which is Node-4 gives **DSR_ RREP** on receiving **DSR_RREQ** and attracts packets towards it.
- While Node-4 (the malicious node) tries to send a **DSR_ RREP**, the legitimate destination, Node-6, also attempts to send **DSR_ RREP** packets.

- However, Node-4's **DSR_ RREP** is favored because it pretends to be the next node is the destination, even though Node-6 is the real destination.
- Because the malicious Node-4 tries to mislead the network by pretending to be a closer destination compared to route reply sent by the actual destination
- You will also find that whatever the Data packets generated from Node-1 are not forwarded by the Malicious Node-4

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 Application 1 registers as zero due to the presence of a malicious node (device ID 4) in the network. This node intentionally drops all data packets instead of transmitting them to their intended destination.

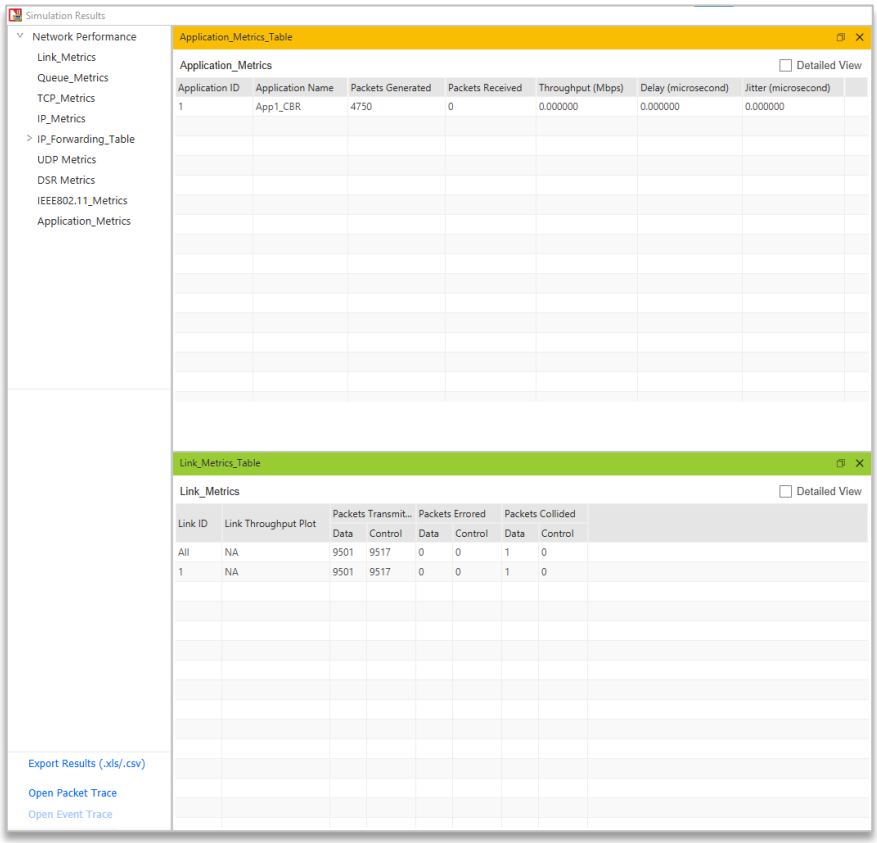


Figure 6: Results Dashboard