Sinkhole Attack in Manet using DSR

Software: NetSim Standard v14.2, Visual Studio 2022

Project Download Link:

https://github.com/NetSim-TETCOS/Sinkhole-Attack-in-DSR-v14.2/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

1 Introduction

Sinkhole attack is one of the most severe attack in wireless ad hoc networks. In a sinkhole attack, a compromised or malicious node advertises false routing information to appear as a specific node and attracts the entire network traffic. After receiving the traffic, the node can either modify the packet information or drop it, complicating the network. Sinkhole attacks affect the performance of ad hoc network protocols, such as the DSR protocol.

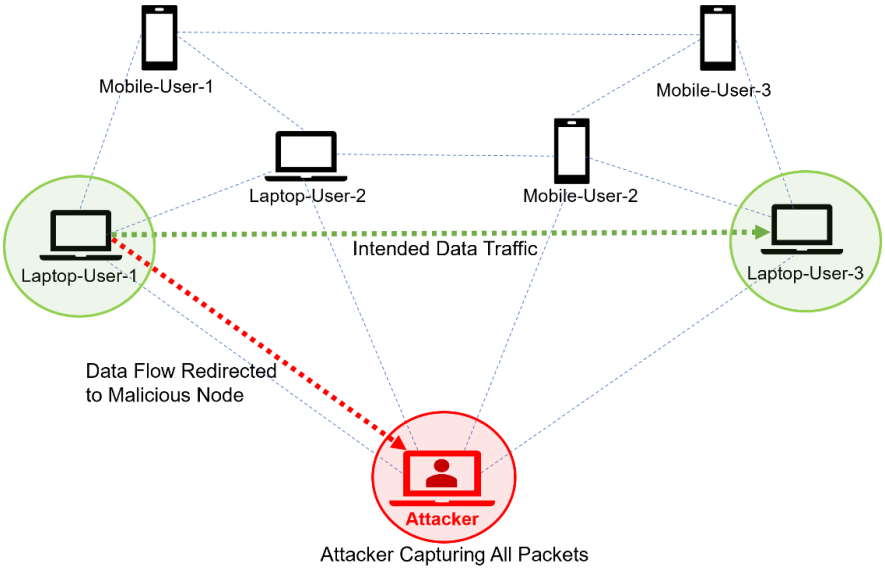


Figure 1: Sinkhole attack in MANET using DSR

2 Implementation in DSR

- In DSR, the source broadcasts the RREQ packet during route discovery.

- The destination, upon receiving the RREQ packet, replies with an RREP packet containing the route to reach the destination.
- However, intermediate nodes can also send RREP packets to the source if they have a route to the destination in their route cache.
- Taking advantage of this, the malicious node adds a fake route entry into its route cache, with the destination node as its next hop.
- Upon receiving the RREQ packet from the source, the malicious node sends a fake RREP packet with the fake route.
- The source node, upon receiving this fake RREP packet, perceives it as a better route to the destination.
- All network traffic is then attracted toward the sinkhole (malicious node), which can either modify the packet information or simply drop the packets.

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

- **fn_NetSim_DSR_MaliciousNode();** // This function is used to identify whether the current device is malicious in order to establish malicious behaviour.
- **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.

3 Steps to simulate

1. Open the source code in Visual studio by navigating to “Your work” on the home screen of NetSim. Then, select “Source code” and click on “Open Code”.
2. Expand the DSR project and open the Malicious.c file and set the malicious node id.

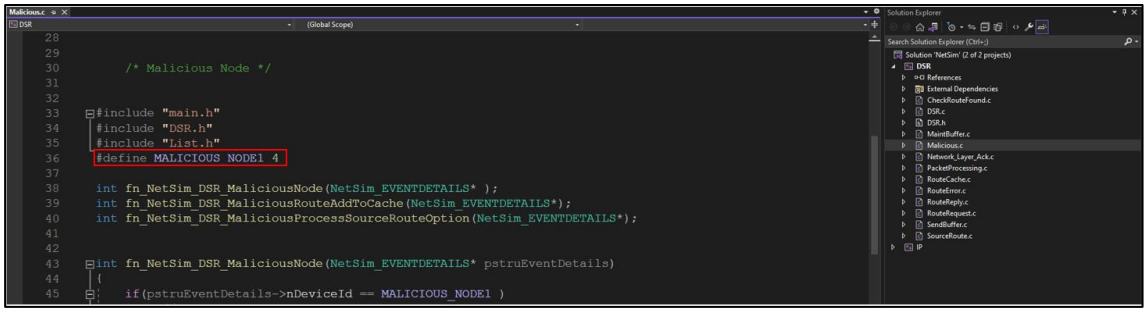


Figure 2: Set Malicious Node in malicious .c file

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

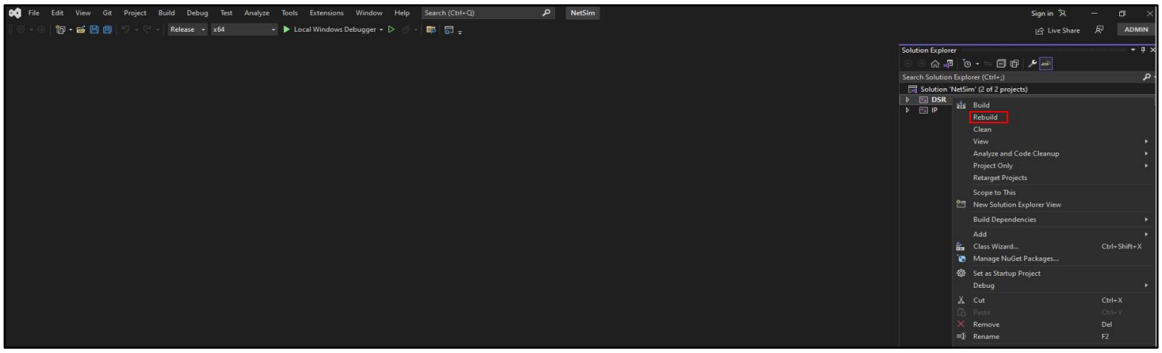


Figure 3: NetSim DSR project source code in Visual studio

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

4 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:

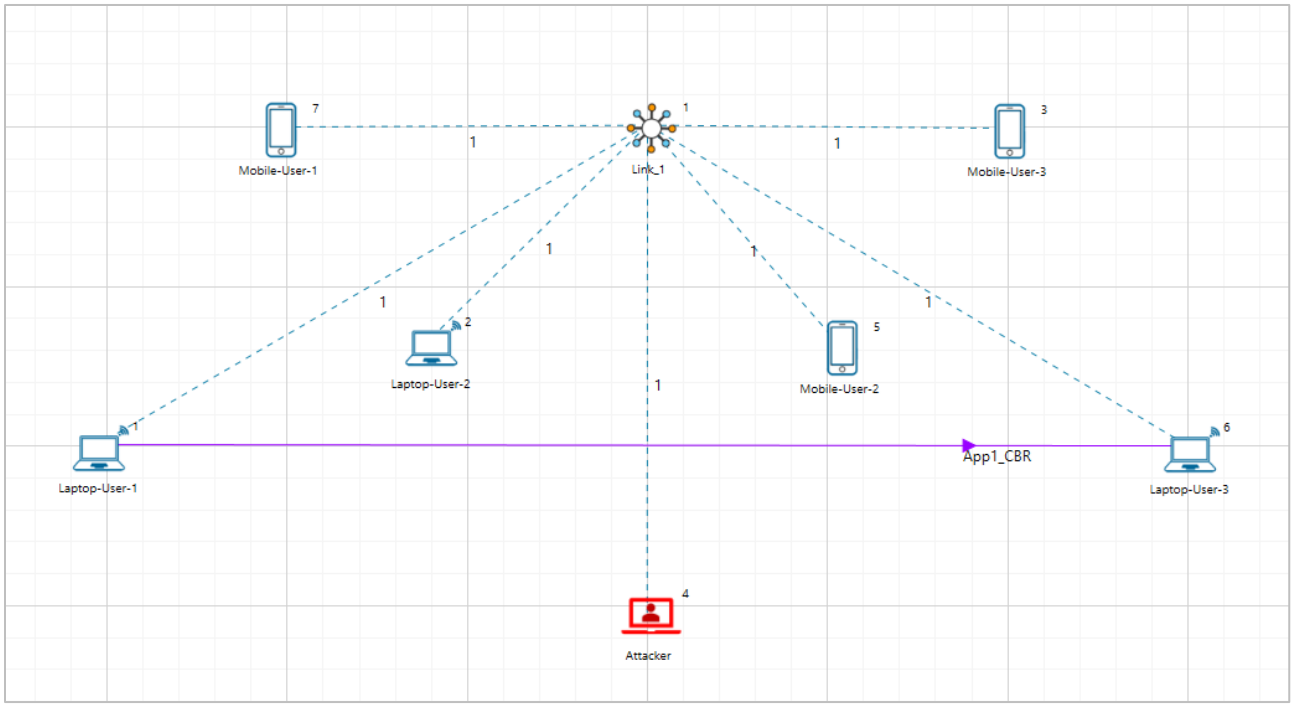


Figure 4: Network topology

3. Set the Application properties

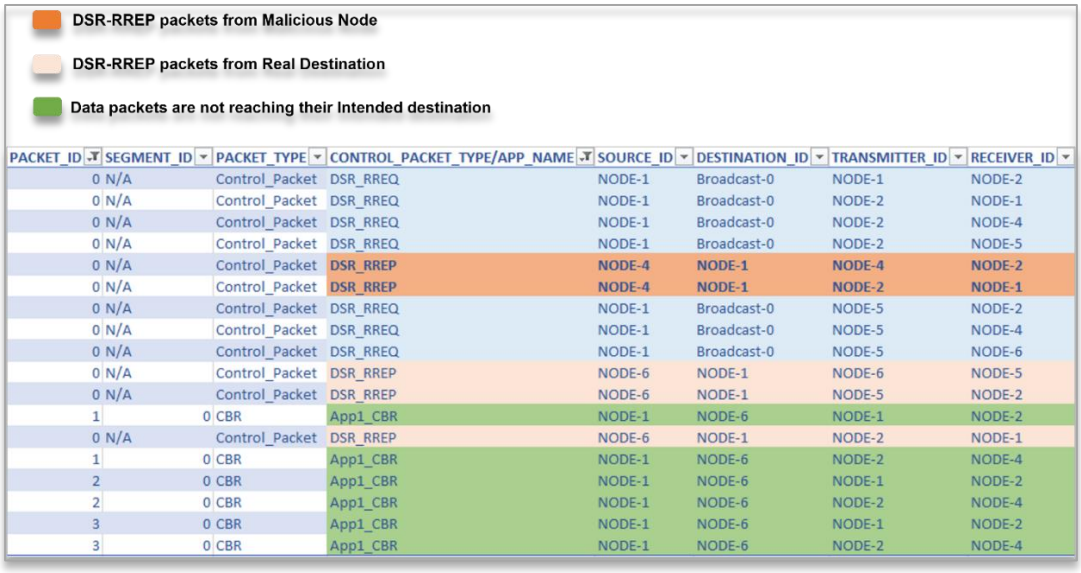
Application Properties	
Source ID	**1**
Destination ID	**6**

Table 1: Application Properties

4. In the Link set the Channel characteristics: **Pathloss only**, Pathloss model: **Log Distance**, Pathloss exponent: **3**
5. Run the Simulation for 100 seconds.

5 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



DSR-RREP packets from Malicious Node

DSR-RREP packets from Real Destination

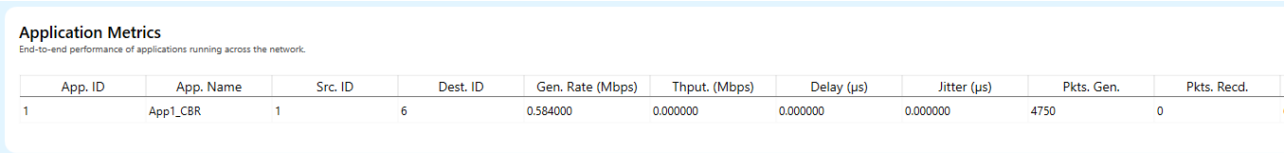
Data packets are not reaching their Intended destination

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

6. From the above screenshot, you will see that the malicious node, node-4, sends an **DSR_RREP** (Route Reply) upon receiving an **DSR_RREQ** (Route Request), attracting packets toward it.
    1. While node-4 (the malicious node) tries to send a Route Reply (**DSR_RREP**), the legitimate destination, Node-6, also attempts to reply with **DSR_RREP** packets.
7. However, node-4's **DSR_RREP** reply is favoured because it pretends to be the next node to the destination, even though node-6 is the actual destination.
    2. The malicious node-4 tries to mislead the network by pretending to be closer to the destination compared to the route reply sent by the actual destination.
    3. You will also find that the data packets generated by 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 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.



Application Metrics

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

App. ID	App. Name	Src. ID	Dest. ID	Gen. Rate (Mbps)	Thput. (Mbps)	Delay (µs)	Jitter (µs)	Pkts. Gen.	Pkts. Recd.
1	App1_CBR	1	6	0.584000	0.000000	0.000000	0.000000	4750	0

Figure 6: Results Dashboard