Sink Hole Attack in MANET using DSR

Software Recommended: NetSim Standard v13.0, Visual Studio 2017/2019

Project Download Link:

https://github.com/NetSim-TETCOS/SINK_HOLE_ATTACK_DSR_v13_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 severe attacks in wireless Ad hoc network. In sinkhole Attack, a compromised node or malicious node advertises wrong routing information to produce itself as a specific node and receives whole network traffic. After receiving whole network traffic, it can either modify the packet information or drop them to make the network complicated. Sinkhole attacks affect the performance of Ad hoc networks protocols such as DSR protocol.

Implementation in DSR:

- In DSR the source broadcasts RREQ packet during Route Discovery.
- The destination on receiving the RREQ packet replies with a RREP packet containing the route to reach the destination.
- But Intermediate nodes can also send RREP packet 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 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.

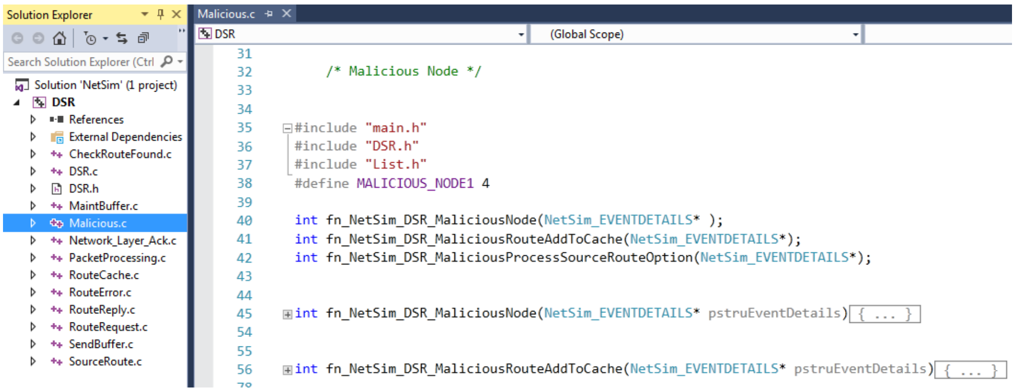
A file **malicious.c** is added to the DSR project which contains the following functions:

- 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, 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_DSR_MaliciousNode() function.

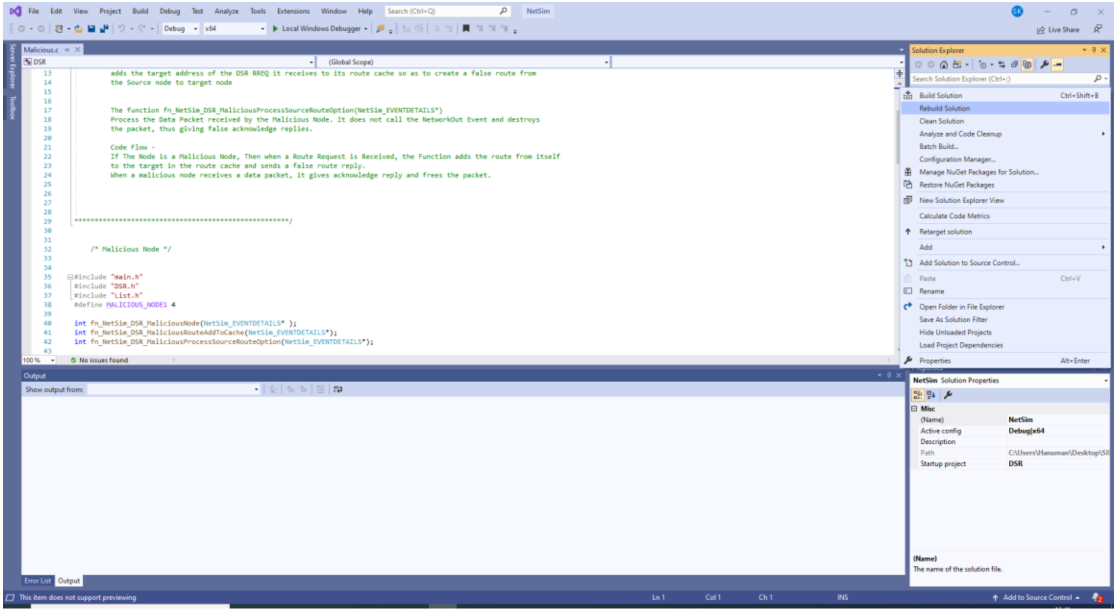
Steps:

1. Go to home page, Click on Your work → Workspace options → Open code
2. A malicious.c file is added to the DSR source code project in which the malicious node and its behaviour is defined.



```
        /* Malicious Node */

#include "main.h"
#include "DSR.h"
#include "List.h"
#define MALICIOUS_NODE1 4

int fn_NetSim_DSR_MaliciousNode(NetSim_EVENTDETAILS* );
int fn_NetSim_DSR_MaliciousRouteAddToCache(NetSim_EVENTDETAILS*);
int fn_NetSim_DSR_MaliciousProcessSourceRouteOption(NetSim_EVENTDETAILS*);


int fn_NetSim_DSR_MaliciousNode(NetSim_EVENTDETAILS* pstruEventDetails){ ... }


int fn_NetSim_DSR_MaliciousRouteAddToCache(NetSim_EVENTDETAILS* pstruEventDetails){ ... }
```

3. Now right click on Solution explorer in the solution explorer and select Rebuild



```
        adds the target address of the DSR RREQ it receives to its route cache so as to create a false route from
        the Source node to target node


        The function fn_NetSim_DSR_MaliciousProcessSourceRouteOption(NetSim_EVENTDETAILS*)
        Process the Data Packet received by the Malicious Node. It does not call the NetworkOut Event and destroys
        the packet, thus giving false acknowledge replies.

        Code Flow -
        If The Node is a Malicious Node, Then when a Route Request is Received, the Function adds the route from itself
        to the target in the route cache and sends a false route reply.
        When a malicious node receives a data packet, it gives acknowledge reply and frees the packet.




************************************************/


    /* Malicious Node */


#include "main.h"
#include "DSR.h"
#include "List.h"
#define MALICIOUS_NODE1 4

int fn_NetSim_DSR_MaliciousNode(NetSim_EVENTDETAILS* );
int fn_NetSim_DSR_MaliciousRouteAddToCache(NetSim_EVENTDETAILS*);
int fn_NetSim_DSR_MaliciousProcessSourceRouteOption(NetSim_EVENTDETAILS*);
```

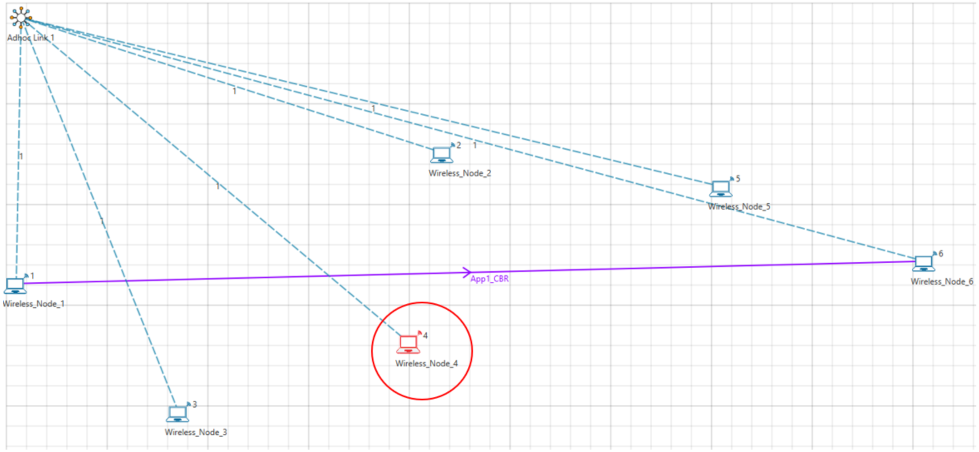
NetSim Solution Properties

Misc

(Name)	NetSim
Active config	Debug|x64
Description	
Path	C:\Users\Hanuman\Desktop\SI
Startup project	DSR

(Name)
The name of the solution file.

4. Upon rebuilding, libDSR.dll will automatically get updated in the respective bin folder of the current workspace.
5. Go to Your Work option in NetSim home screen and click on the SINK_HOLE_ATTACK_DSR_Example to open the sample network scenario in NetSim.
6. The network consists of 6 Wired nodes with properties configured as shown below:

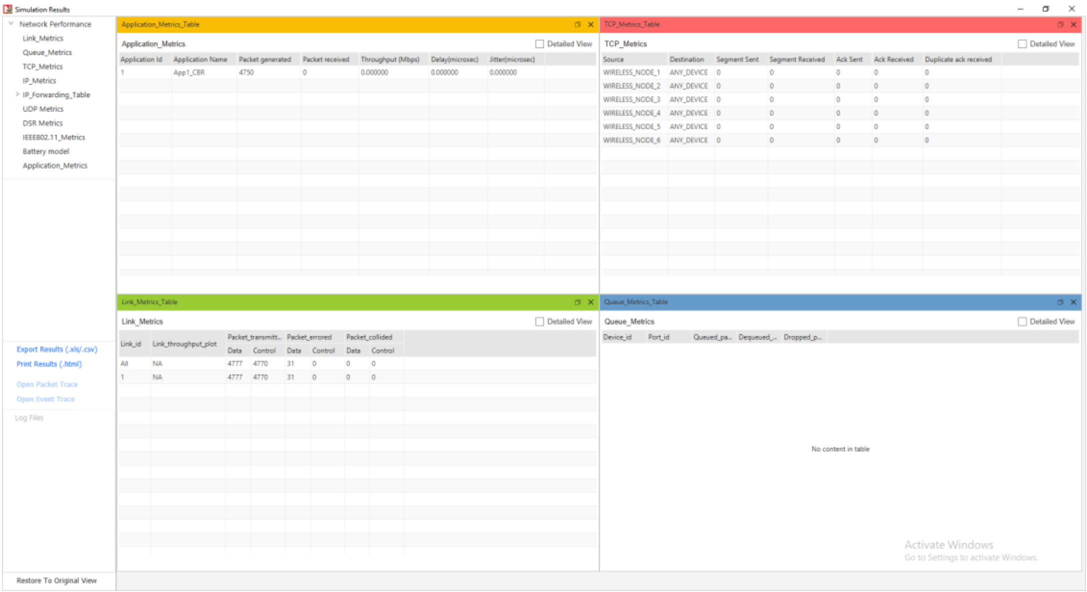


Source – Device id 1
Destination – Device id 6
Sinkhole (malicious node) – Device id 4

Link Properties (Adhoc link1)

Channel characteristics – Path Loss only
Path Loss model – LOG DISTANCE
Path Loss Exponent: 3

7. Run the Simulation for 100 seconds.
8. View the packet animation. You will find that the malicious node (Device id 4) gives Route Reply on receiving Route Request and attracts packets towards it. You will also find that the malicious node does not forward the packets that it receives.
9. This will have a direct impact on the Application Throughput which can be observed in the Application Metrics table present in NetSim Simulation Results window.



Application_Metrics

Application Id	Application Name	Packet generated	Packet received	Throughput (Mbps)	Delay(microsec)	Jitter(microsec)
1	App1_CBR	4750	0	0.000000	0.000000	0.000000

TCP_Metrics

Source	Destination	Segment Sent	Segment Received	Ack Sent	Ack Received	Duplicate ack received
WIRELESS_NODE_1	ANY_DEVICE	0	0	0	0	0
WIRELESS_NODE_2	ANY_DEVICE	0	0	0	0	0
WIRELESS_NODE_3	ANY_DEVICE	0	0	0	0	0
WIRELESS_NODE_4	ANY_DEVICE	0	0	0	0	0
WIRELESS_NODE_5	ANY_DEVICE	0	0	0	0	0
WIRELESS_NODE_6	ANY_DEVICE	0	0	0	0	0

Link_Metrics

Link_id	Link_throughput_plot	Packet_transmitt...		Packet_errored		Packet_collided	
		Data	Control	Data	Control	Data	Control
All	NA	4777	4770	31	0	0	0
1	NA	4777	4770	31	0	0	0

Queue_Metrics

Device_id	Port_id	Queued_pa...	Dequeued_...	Dropped_p...

No content in table