Secure AODV in MANET

Software: NetSim Standard v14.3, Microsoft Visual Studio 2022

Project Download Link:

https://github.com/NetSim-TETCOS/Secure-AODV-v14.3/archive/refs/heads/main.zip

Follow the instructions specified in the following link to download and set up the Project in NetSim:

https://support.tetcos.com/en/support/solutions/articles/14000128666-downloading-and-setting-up-netsim-file-exchange-projects

1 Introduction

SAODV is an extension of the AODV routing protocol that can be used to protect the route discovery mechanism by providing security features like integrity and authentication. The reason only route discovery is secured by AODV is that data messages can be protected using a point-to-point security protocol like IPSec. SAODV uses a key management system, and each node maintains public keys, encryption keys, and decryption keys.

The SAODV implementation introduces three core files: Secure AODV.c, RSA.c, and Malicious.c. RSA.c handles key generation, encryption, and decryption processes, adaptable for other encryption algorithms as needed. Malicious.c identifies unauthorized nodes in the network, strengthening security by limiting their participation.

This setup ensures unauthorized devices cannot access or interpret data, preserving communication integrity across devices.
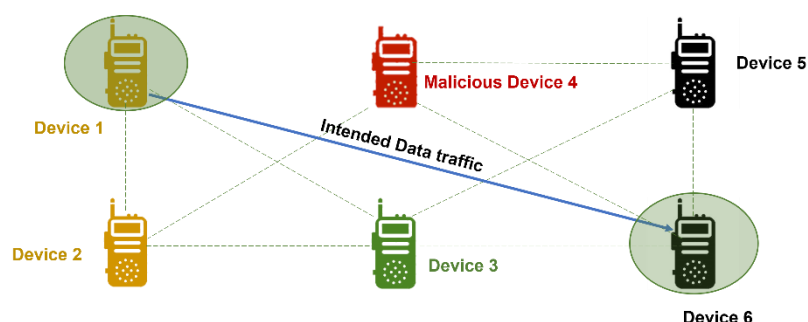

Figure 1:Secure AODV in Military communication using Manet

2 Secure AODV Overview

1. To discover a route to Device 6, Device 1 sends an RREQ encrypted with Device 6's public key.
2. When Device 6 receives the RREQ, it decrypts it with its private key.

3. Device 6 then sends an RREP back to Device 1, encrypted with Device-1's public key.

4. Device 1 decrypts the RREP with its private key.

If a malicious device tries to intercept the RREQ or RREP, it will not be able to decrypt it because it does not have the private keys of Device 1 or Device 6. Additionally, if a malicious device tries to modify the RREQ or RREP, the other nodes in the network will be able to detect the modification and discard the packet.

3 Case 1

Secure AODV implementation:

1. The **Secure-AODV-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 **Secure-AODV-Example** from the list of experiments.
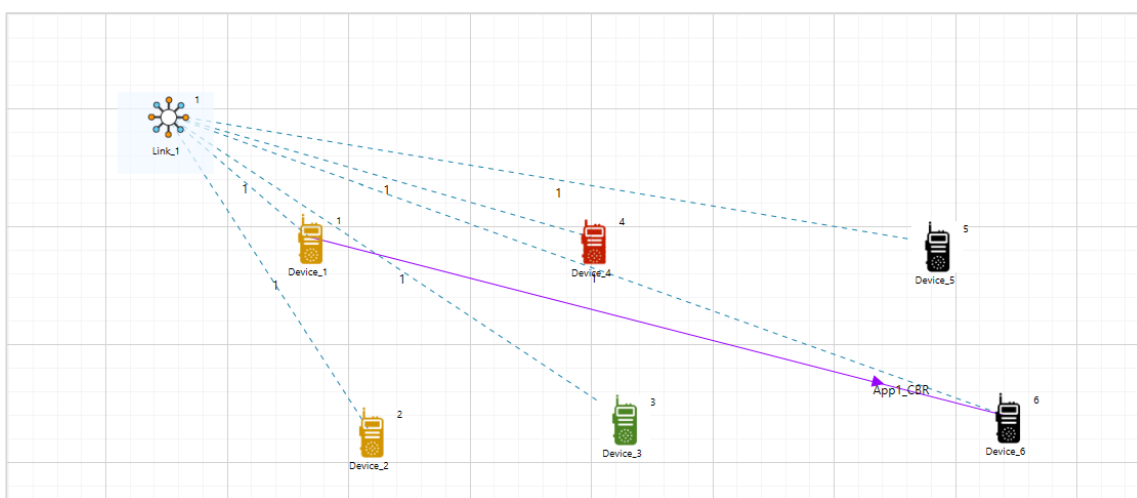

Figure 2: Network setup for Secure AODV in Manet

Key Functions in Secure_AODV.c :

A **Secure_AODV.c** file is added to the AODV project which contains the following important functions:

- **saodv_encrypt_packet();** //This function is used to encrypt the control packet data

- **saodv_decrypt_packet();** //This function is used to decrypt the control packet data

- **get_rrep_str_data();** //This function is used to get the route reply data from AODV_RREP control packet

- **get_rreq_str_data();** //This function is used to get the route request data from AODV_RREQ control packet

- **get_saodv_ctrl_packet_type();** //This function is used to change the control packet type from AODV (AODV_RREQ, AODV_RREP) to SAODV (SAODV_RREQ, SAODV_RREP)

- **get_saodv_ctrl_packet();** //This function is called whenever a new control packet is generated
- **get_aodv_ctrl_packet();** //This function is called while processing the control packets
2. Run the simulation for 100 seconds, After running the simulation, a Secure_AODV.log file gets created in the Result Dashboard Window

4 Results and discussion

After Running the simulation of the given Configuration file, open the packet trace in the Result Dashboard Window. In the packet trace, filter the control packets to **SAODV_RREQ** and **SAODV_RREP)**

Figure 5: Network Packet trace results for Secure AODV implementation

The SAODV logs certain details in **Secure_AODV.log**. The Log File can be observed in the Result Dashboard Window.
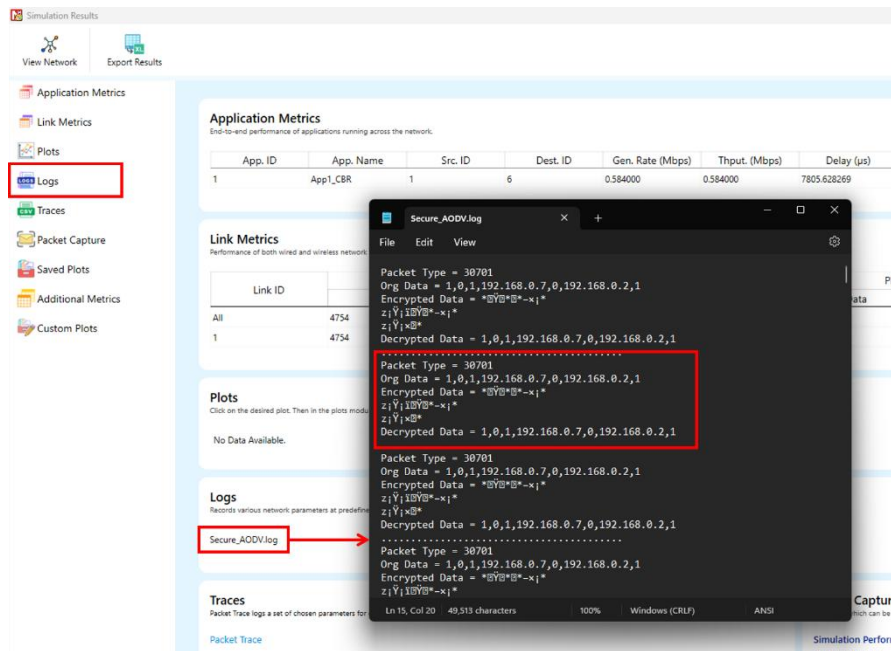
Figure 6: Secure AODV log file

The format of the log file is such that each control packet is logged. The first line represents the packet type and the numbering used in a NetSim internal numbering system where **30701 is RREQ and 30702 is RREP**. The second line is the message which is encrypted. The third line contains the encrypted message after running the RSA encryption algorithm. The fourth line is after decryption and if everything is OK, the 2nd and 4th lines must match.

....

Packet Type = 30701

Org Data = 1,0,1,192.168.0.7,0,192.168.0.2,1

Encrypted Data = š ššJÒÜšÀÐÜ Ü× šJÒÜšÀÐÜ ÜÒš

Decrypted Data = 1,0,1,192.168.0.7,0,192.168.0.2,1

....

5 Case 2

Malicious node implementation:

1. To enable Malicious node open the Source code in Visual Studio by going to Your work -> Source Code and Open code in the NetSim Home Screen window.

2. Expand the AODV project.

    Here users can enable the code to implement malicious node

3. Steps to Enable Malicious node

- Open the **'AODV.h'** file within the AODV project
- Find the line: **#define SAODV_ENABLE.**
- Comment this line by adding ' **//** ' at the beginning.
- Find the line: **//#define MALICIOUS_ENABLE.**
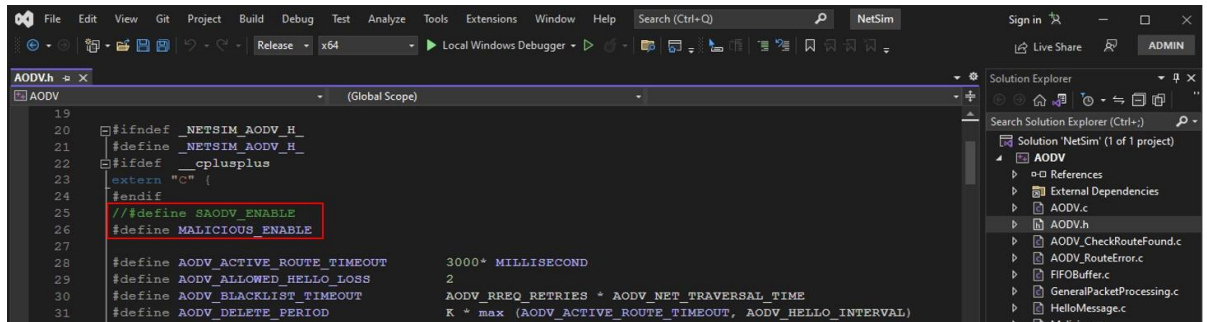- Uncomment this line by removing the ' **//** ' at the beginning.

Figure 3: Comment and Uncomment the code of SAODV and Malicious

4. Rebuild the AODV Project

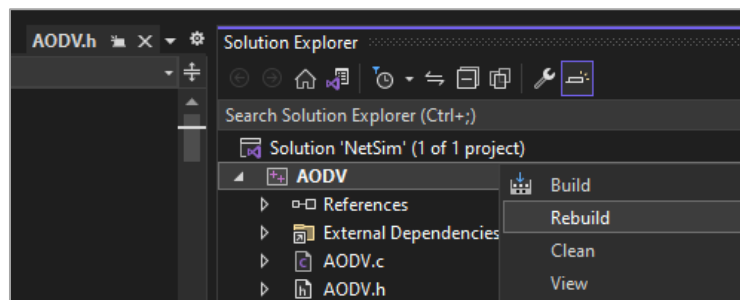- Right-click on the project in Solution Explorer.
- Select Rebuild

Figure 4: Screenshot of Solution Explorer of AODV project

A malicious node advertises wrong routing information to produce 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 them to make the network complicated.

Key Functions in malicious.c:

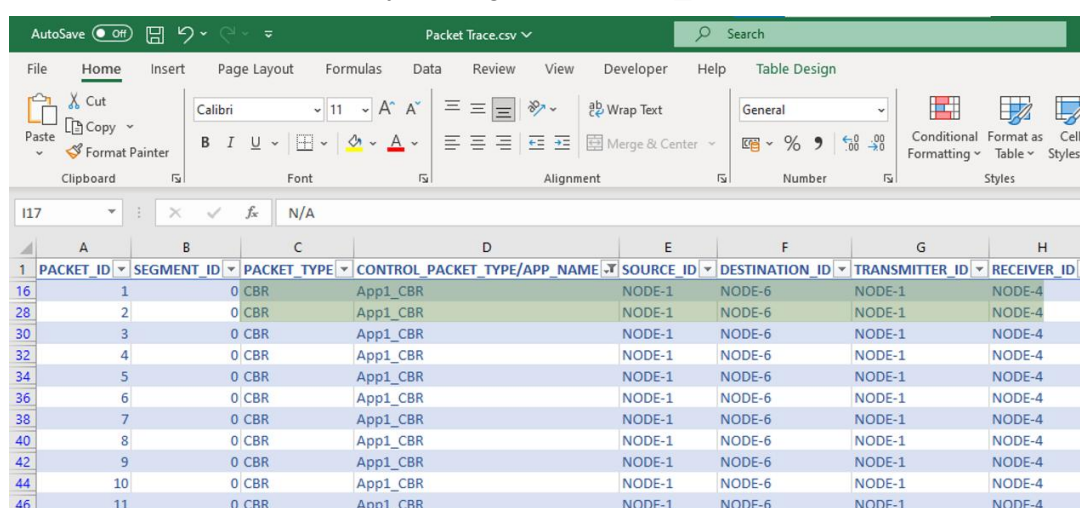A file **malicious.c** is added to the AODV project which contains the following functions:

- **IsMaliciousNode();** //This function is used to identify whether a current device is malicious or not in-order to establish malicious behaviour.

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

- **fn_NetSim_AODV_MaliciousProcessSourceRouteOption();** //This function is used to drop the received packets if the device is malicious, instead of forwarding the packet to the next hop

5. Run the simulation for 100 seconds in Netsim GUI.

6 Results and discussion

- You can set any device as a malicious node, and you can have more than one malicious node in a scenario.

- Device IDs of malicious nodes can be set using the malicious_node [ ] array present in malicious.c file.

- Rebuild the project and run the simulation.

- If we run the simulation without SAODV, we will get zero throughputs because the malicious node gets all the packets and drops without forwarding them to the destination. You can notice this in the NetSim packet trace by filtering the PACKET_TYPE to CBR.

Figure 8: NetSim Packet trace results for Malicious node implementation

7 Case 3

Both Secure AODV and Malicious node implementation:

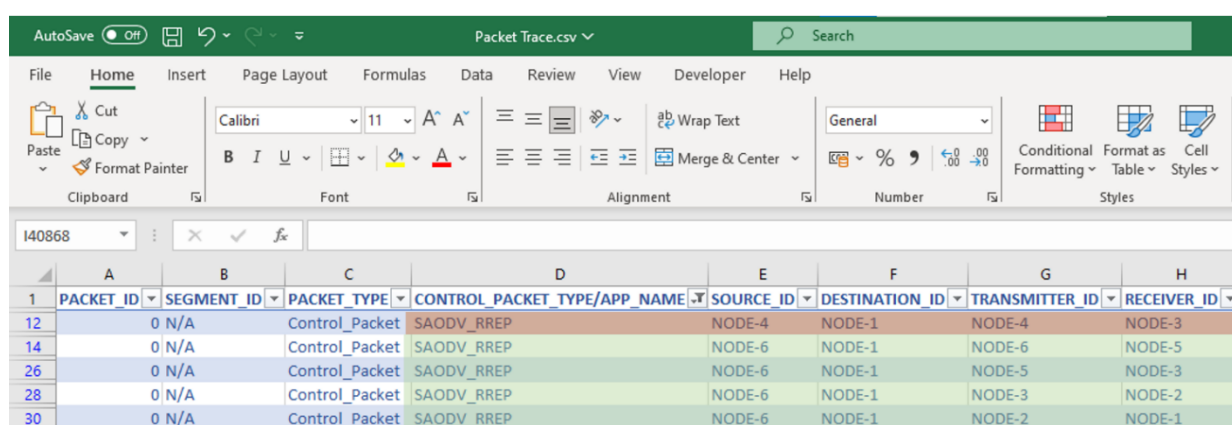Steps to Enable Both Secure AODV and Malicious Node

- Open the **'AODV.h'** file within the AODV project.
- Find the line: **//#define SAODV_ENABLE** and **//#define MALICIOUS_ENABLE.**
- Uncomment both lines by removing the ' **//** ' at the beginning.

Rebuild the Project and run the simulation for 100 sec in Netsim GUI.

8 Results and discussion:

Packets will be transmitted to the destination since SAODV helps in overcoming the Malicious Node problem. Route reply RREP from malicious node 4 will not be accepted by Node 1. It takes the Route reply from node 2 and forms the route. This can be observed in Packet Trace by filtering the CONTROL_PACKET_TYPE to SAODV_RREP. Malicious node 4 is transmitting the Route Reply (RREP) to node 3, but node 3 is not forwarding to any nodes .On the other hand, when it comes to node 6, it is sending the RREP to node 5, then from 5 to 3, from 3 to 2 and finally from 2 to 1.
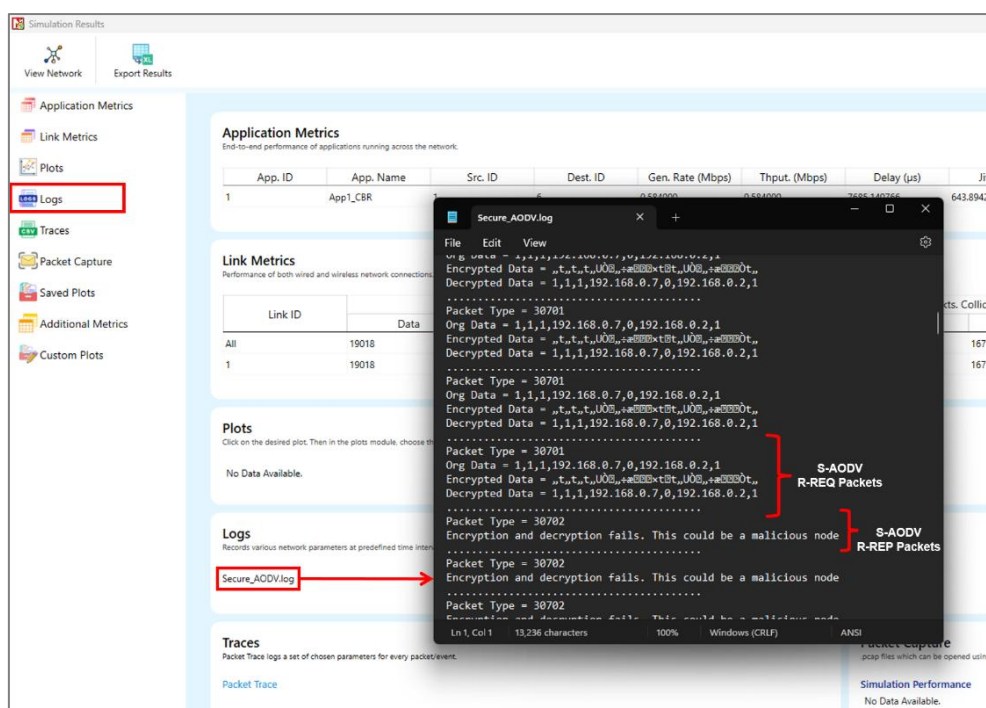

Figure 9: NetSim Packet trace results for Secure AODV and Malicious node implementation

The SAODV logs certain details in **Secure_AODV.log**. The Log File can be observed in the Result Dashboard Window.

Figure 10: Secure AODV log file

The Packet type 30701 = RREQ is the request packet and the Packet type 30702=RREP is the reply packet, when the malicious node tries to decrypt the message.

....

Packet Type = 30702

Encryption and decryption fail. This could be a malicious node.

....

Appendix: NetSim source code modifications

We have added Secure_AODV.c, RSA.c and Malicious.c files, we have added the following macros code in AODV.h file within AODV project.

#define SAODV_ENABLE

#define MALICIOUS_ENABLE

Then we have added the following lines of code in enum_AODV_Ctrl_Packet in AODV.h file

---

//#ifdef SAODV_ENABLE

 SAODV_RREQ,

 SAODV_RREP,

 SAODV_RERR,

//#endif

We have added the following function prototypes in AODV.h file, within AODV project.

#ifdef SAODV_ENABLE

void get_saodv_ctrl_packet(NetSim_PACKET* packet);

void get_aodv_ctrl_packet(NetSim_PACKET* packet);

void saodv_copy_packet(NetSim_PACKET* dest, NetSim_PACKET* src);

void saodv_free_packet(NetSim_PACKET* packet);

void remove_from_mapper(void* ptr, bool isfree);

#endif // SAODV_ENABLE

bool IsMaliciousNode(NETSIM_ID devId);

We have added the following function prototypes in AODV.c file int fn_NetSim_AODV_MaliciousRouteAddToTable(NetSim_EVENTDETAILS*); int fn_NetSim_AODV_MaliciousProcessSourceRouteOption(NetSim_EVENTDETAILS*);

Changes to NETWORK_IN event in fn_NetSim_AODV_Run() function in AODV.c file, within AODV project

---

```
 #ifdef SAODV_ENABLE
switch (pstruEventDetails->pPacket->nControlDataType)
    {
            case SAODV_RREQ:
            case SAODV_RREP:
        case SAODV_RERR:
            get_aodv_ctrl_packet(pstruEventDetails->pPacket);
            break;
    }
    if (pstruEventDetails->pPacket == NULL)
    {
            return -1; //Decryption fail.
    }
#endif // SAODV_ENABLE
```

We have added the following lines of code in AODVctrlPacket_RREQ and default cases in NETWORK_IN event to check the current node is malicious or not.

---

if (IsMaliciousNode(pstruEventDetails->nDeviceId))

fn_NetSim_AODV_MaliciousRouteAddToTable(pstruEventDetails);

Changes code in fn_NetSim_AODV_CopyPacket () function, in AODV.c file, within AODV project

---

#ifdef SAODV_ENABLE switch(srcPacket->nControlDataType)

{

case SAODV_RERR: case

SAODV_RREQ: case

SAODV_RREP:

saodv_copy_packet(destPacket,srcPacket);

return 0; break;

default:

#endif

      return fn_NetSim_AODV_CopyPacket_F(destPacket,srcPacket);

#ifdef SAODV_ENABLE

      break;

      }

#endif

Changes code in int fn_NetSim_AODV_FreePacket () present in the AODV.c file, within AODV project

---

#ifdef SAODV_ENABLE

switch (packet->nControlDataType)

      {

```
 case SAODV_RERR:

case SAODV_RREQ:

case SAODV_RREP:

        saodv_free_packet(packet);

return 0;        break;

default:

                remove_from_mapper(packet->pstruNetworkData->Packet_RoutingProtocol, true);

                return 0;

break;

        }
#endif // SAODV_ENABLE
```

Changes code in fn_NetSim_AODV_GenerateRREQ (), fn_NetSim_AODV_RetryRREQ () and fn_NetSim_AODV_ForwardRREQ () functions present in RREQ.c file, within AODV project

```
#ifdef SAODV_ENABLE

        get_saodv_ctrl_packet(packet);

#endif
```

Changes code in fn_NetSim_AODV_GenerateRREP(), fn_NetSim_AODV_ForwardRREP () and fn_NetSim_AODV_GenerateRREPByIntermediate () functions present in RREP.c file, within AODV project

```
#ifdef SAODV_ENABLE

        get_saodv_ctrl_packet(packet);

#endif
```