False Data Injection Attack
Two cases: Simulation and Emulation
Two Types: Payload modification and Header modification

Software: NetSim Standard v14.0 (64 bit), Visual Studio 2022
Project code download link: https://github.com/NetSim-TETCOS/False-Data-Injection-Attack-in-Internetworks_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

FDI (False Data Injection) attack is a type of cyber-attack where an attacker injects false data into a system or network with the intent of causing damage or disruption. FDI attacks can be launched against various types of systems, including industrial control systems, critical infrastructure, financial systems, and information systems.

In an FDI attack, the attacker may modify or manipulate data in transit or at rest to achieve their objectives. For example, an attacker may alter the data in a financial transaction to redirect funds to a different account, modify the configuration of an industrial control system to cause physical damage, or manipulate data in a way that causes a system to crash or malfunction.

Toy Example: FDI Attack on PING

In this example, we launch an FDI attack on ICMP ping messages between a source and destination. The destination receives the message and processes it as if it were legitimate.

Case 1: FDI implementation within NetSim simulator. Packet payload modification.

This case is a simpler method of simulating the FDI attack requiring only one machine. Case 2 (described later) involves using 3 machines.

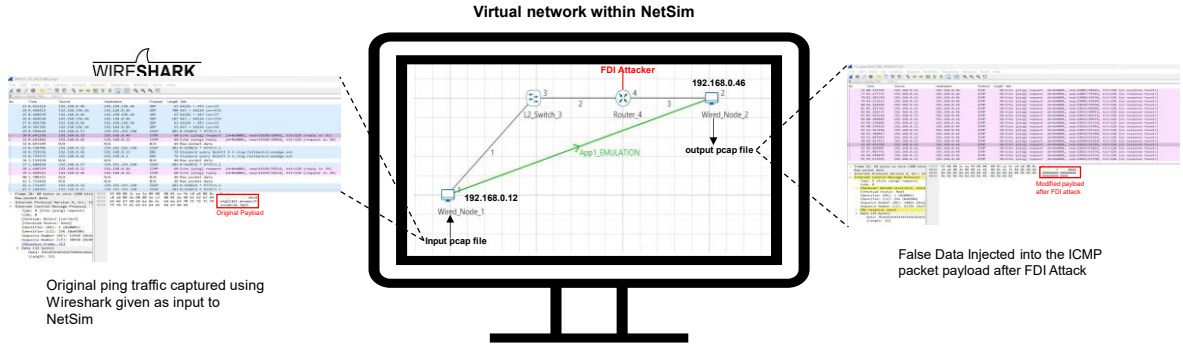


Fig 1: PING application between a real source and real destination is captured as a pcap file and given as an input to a virtual source inside NetSim. In this example, the source IP is set to 192.168.0.12 and the destination IP is set to 192.168.0.46. The external pcap file is available in the project download link.

Generating Packet capture for NetSim

We explain the steps used to capture PING data as a pcap file. This has been provided for those readers who may wish to capture their own pcap files and use implement the FDI attack on that.

1. Open Wireshark in the system where NetSim is installed.
2. Once the Wireshark is opened, please select the proper interface .(For Ex: Ethernet) as show below. Double click on the interface to open live packet capture window.

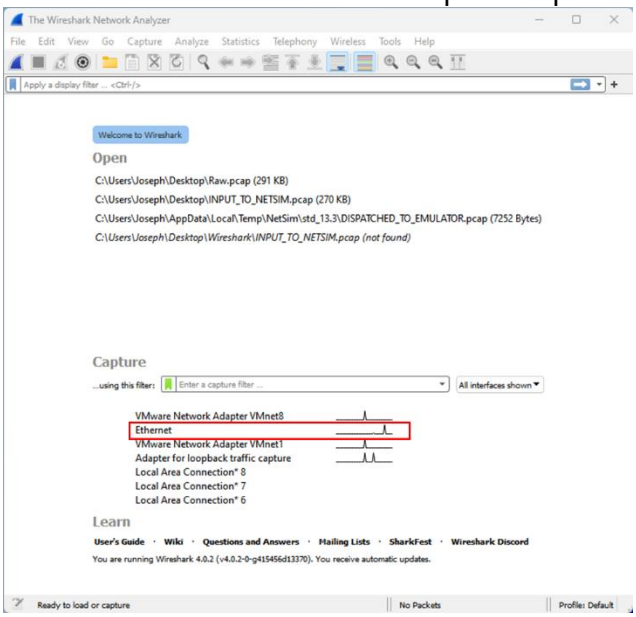


Fig 2: Select packet capture interface to capture packets at source.

3. In this Example we have considered areal source with 192.168.0.12 and a real destination with IP 192.168.0.46. Open command line at source device and enter the command
   - ping 192.168.0.46 -t

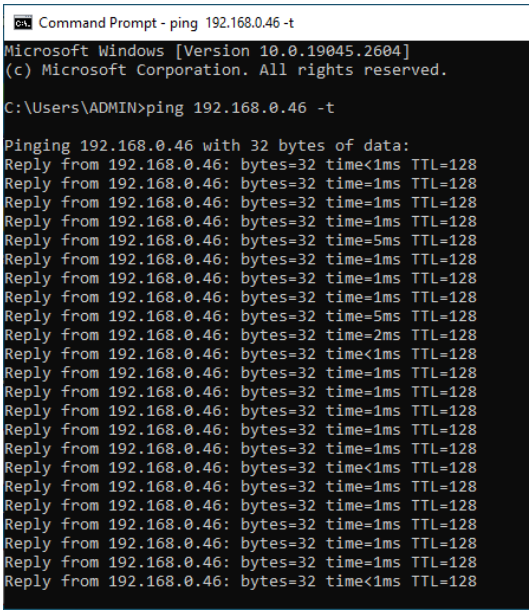
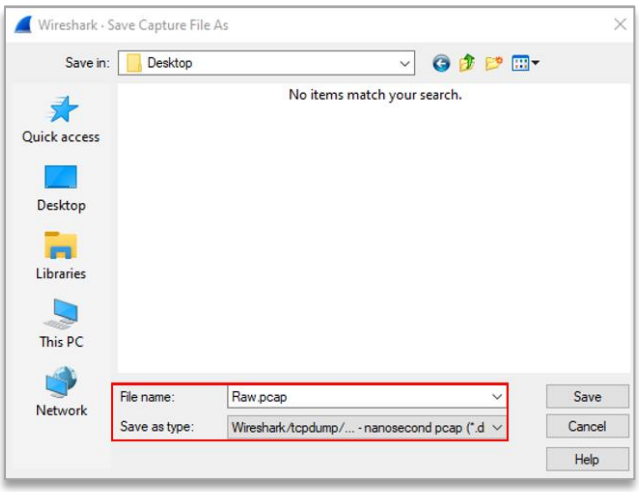


Fig 3 : Ping traffic between source IP 192.168.0.12 and destination IP 192.168.0.46

4. The pcap file will contain all incoming and outgoing packets from the system in which the capture is being done. Once you have captured the desired ping traffic stop the Wireshark packet capture using stop option and save the packet capture in a desired location with desired name (*.pcap) for E.g., Raw.pcap with Save as type as **Wireshark/tcpdump…. -pcap**.



5. This PCAP file needs to be edited before giving as input to NetSim. The editcap application in Wireshark Installation Directory can be used to edit the any pcap file to be provided as a input to NetSim
6. Go to Wireshark installation directory [C:\Program Files\Wireshark]
7. Open command prompt, and execute the following command:
   - **editcap -C 14 -L -T rawip -F pcap *"<File Location where the file is present>*\Raw.pcap" *"<File Location where the file needs to be saved>*\INPUT_TO_NETSIM.pcap"**

Steps to simulate by providing pcap packet capture file as input to NetSim

1. Go to start search Run → Enter the command “SystemPropertiesAdvanced” and then click on OK.
2. Click the Environment Variables → Add the following Environment PATH variable.
   <File-Path-where-INPUT_TO_NETSIM.pcap file is located>\INPUT_TO_NETSIM.pcap
   For eg: C:\Users\Joseph\Desktop\INPUT_TO_NETSIM.pcap

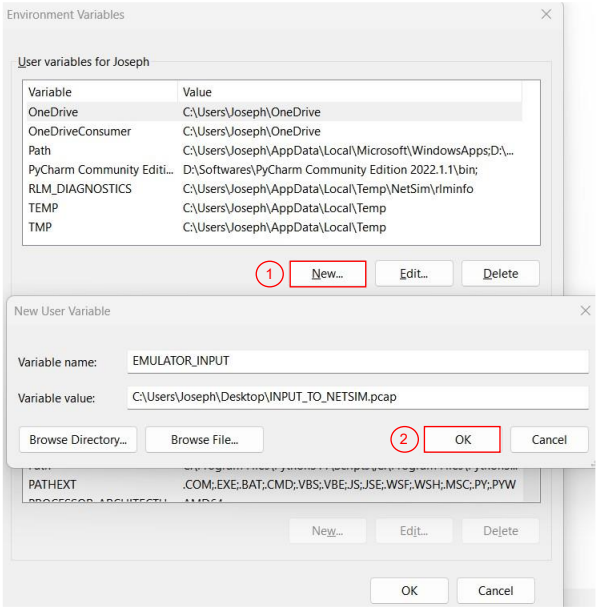


Fig 4 : Environment Variable Path

For more information how to provide pcap file as input refer our knowledge base article https://support.tetcos.com/support/solutions/articles/14000103748-how-can-i-provide-pcap-file-as-input-to-simulation-

Implementing the FDI attack

1. Run the NetSim in Administrator Mode (Right Click on NetSim Icon → Run as Administrator)
2. The **FDI_Attack_in_Internetworks_v13.3** comes with a sample network configuration that are already saved. To open this example, go to Your work in the home screen of NetSim and click on the **FDI_Sample_Internetwork** from the list of experiments.

3. The saved network scenario consists of
    - 2 Wired Node
    - 1 L2 Switch
    - 1 Router

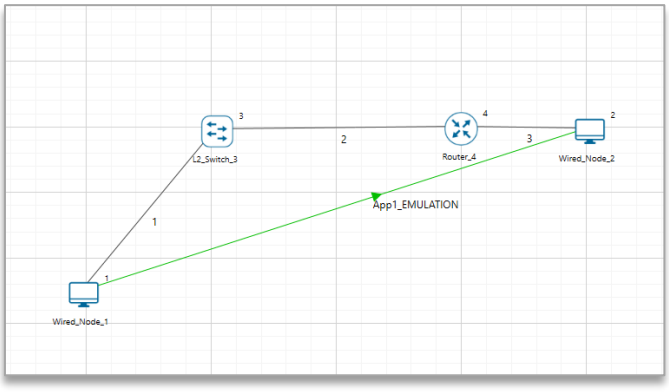


Fig 5: NetSim Emulation Scenario, Wired_Node_1 device mapped for Source IP 192.168.0.12 and Wired_Node_2 device mapped for Destination IP 192.168.0.46

4. Application Properties
    - Application Type - EMULATION
    - Source IP - 192.168.0.12

- Destination IP – 192.168.0.46

5. Run the Simulation for 100 sec.

Note: The source IP address refers to the IP address of the system from which you are initiating the ping command.
The destination IP address to the IP address of the device or system that you are pinging.

Observations

After the simulation is completed, you can observe the results using Wireshark captured files. In the Result Dashboard, On the left side, Packet Capture → Emulation and you can see all Emulated Packets captured.

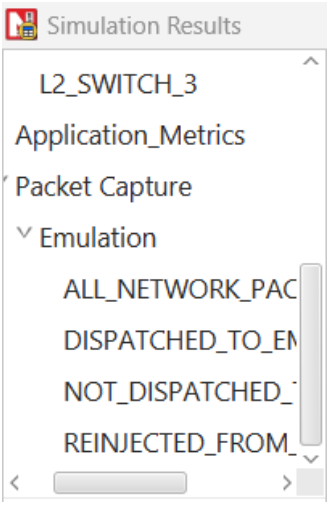


Fig 6: Emulation Packet Capture in Result Dashboard

We can observe original packets in the DISPATCHED_TO_EMUALTOR.pcap file.

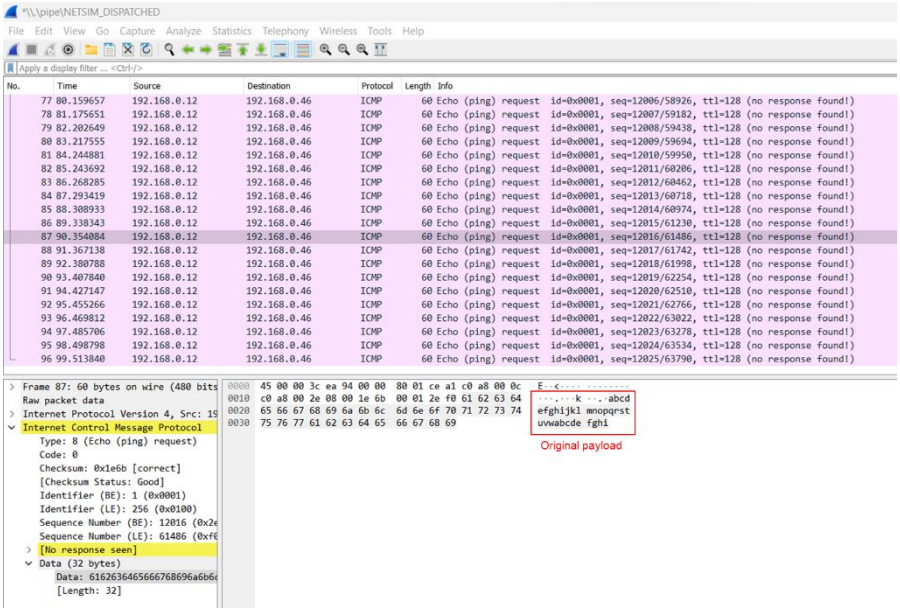


Fig 7: Original payload captured by NetSim emulator

We can observe false data injected packets in the REINJECTED_FROM_EMUALTOR.pcap file.

Note: *You should select the any ICMP Packet to observe the changes.*

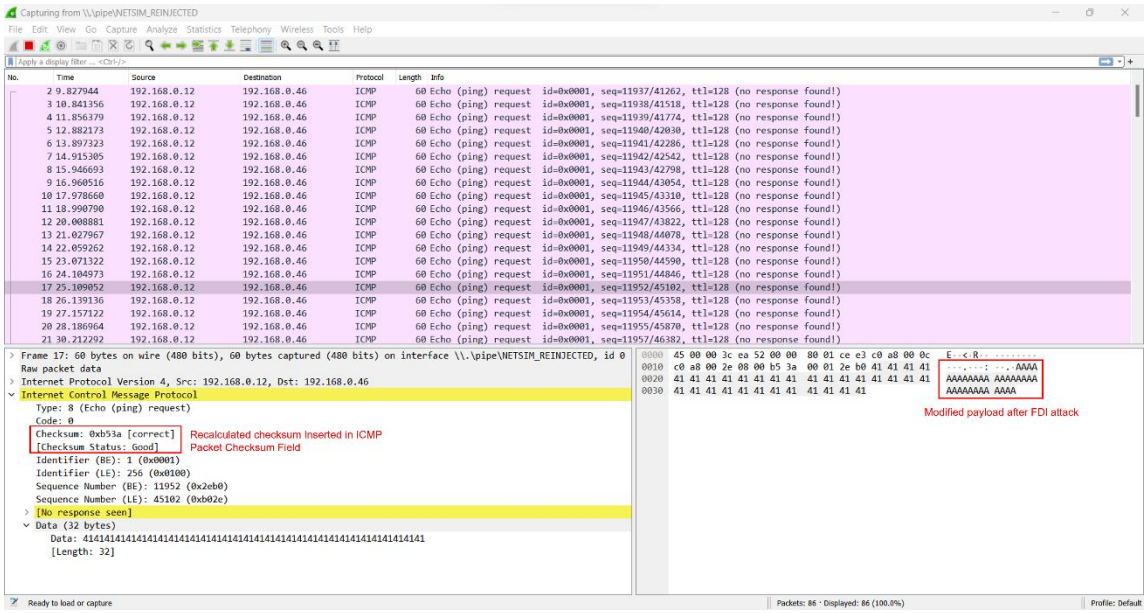


Fig 8: Traffic with false data injected. Observe the difference in payload and checksum is recalculated and inserted in ICMP packet checksum field.

Case 2: FDI implementation in NetSim emulator. Packet header modification.

We have 3 systems – Source, Destination, and Emulator. The PING packets from source to destination pass through the emulator.

FDI attack on real traffic using NetSim Emulator

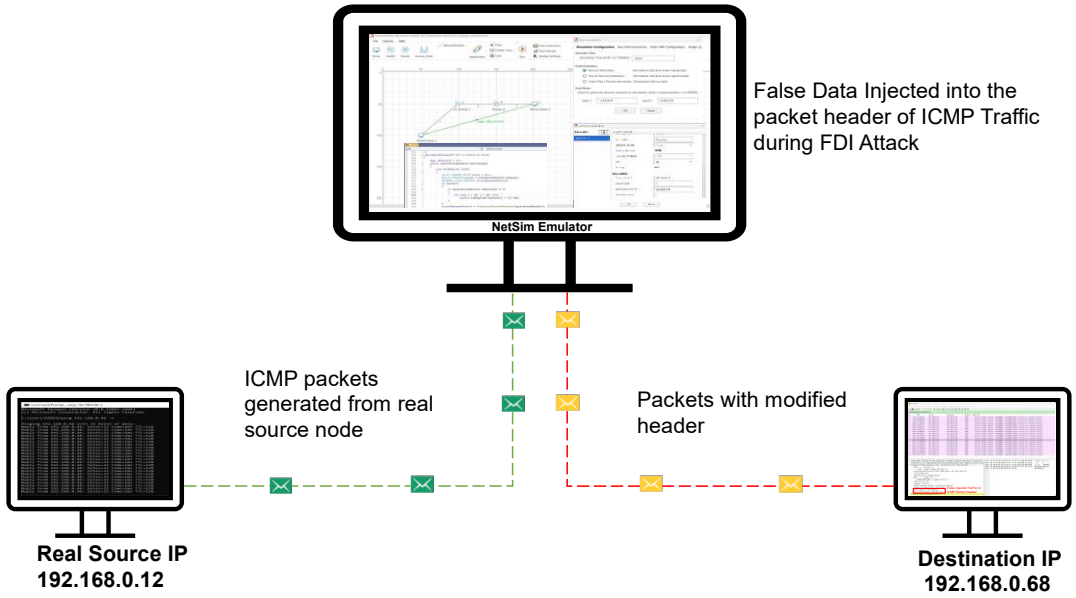


Fig 9 : PING application between source and destination. The source IP is set to 192.168.0.12 and the destination IP is set to 192.168.0.46. In NetSim We are implementing FDI Attack by modifying the destination IP address to 192.168.0.68 in ICMP packet header.

Steps to Simulate

The set-up to run emulation would be to have a minimum of three (3) PC's. One would be the real source, the second would run NetSim emulation server, and the third would be the real destination.

In this Example, we have considered 3 systems as shown below.
Real Source IP: 192.168.0.12
NetSim Emulation Server IP: 192.168.0.81
Real Destination IP: 192.168.0.46

Setting up the NetSim Emulation Server

1. Run the NetSim in Administrator Mode (Right Click on NetSim Icon → Run as Administrator)
2. Open the Existing Sample **FDI_Sample_Internetwork** from the list of Experiments (In NetSim Home Screen → Your Work)
3. The saved network scenario consists of
    - 2 Wired Node
    - 1 L2 Switch
    - 1 Router

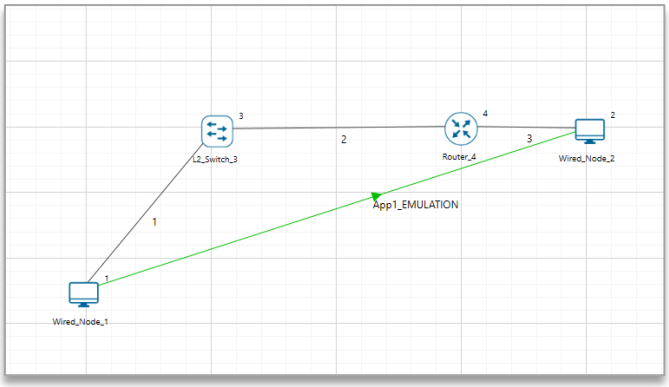


Fig 10: NetSim Emulation Scenario, Wired_Node_1 device mapped for Source IP 192.168.0.12 and Wired_Node_2 device mapped for Destination IP 192.168.0.46

4. Application Properties
    - Application Type - EMULATION
    - Source IP - 192.168.0.12
    - Destination IP – 192.168.0.46
5. Run the Simulation for 100 sec.

Note: The source IP address refers to the IP address of the system from which you are initiating the ping command.
The destination IP address to the IP address of the device or system that you are pinging.

Setting up the Real Source and Destination

The client systems which are sources of real traffic can be connected to NetSim emulator by resetting the gateway. Once the gateway for the client system is set as the NetSim Emulator PC then traffic from the clients will go via NetSim Emulator PC.

Configuring NetSim Emulator as a Gateway in NetSim in Windows clients

1. Open command prompt in Administrator Mode
2. Type the command.
   - route add <Network Destination> mask <Subnet Mask> <Gateway IP> metric 1
   - **route add 192.168.0.46 mask 255.255.255.0 192.168.0.81 metric 1**
   - After the Execution , you will get “OK”.

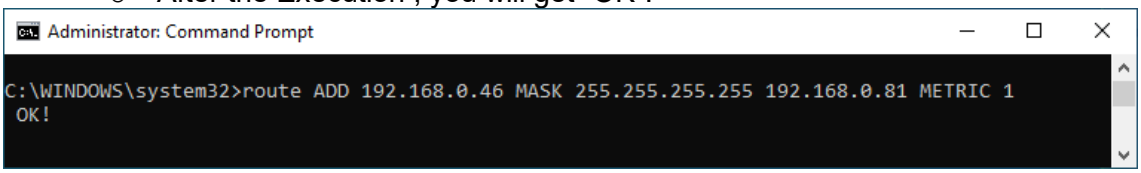


Fig 11: Adding the Static route from source to destination via gateway as NetSim emulation server-192.168.0.81

3. To check whether IP Configuration affected or not type the command as show below
   - **netstat -r**

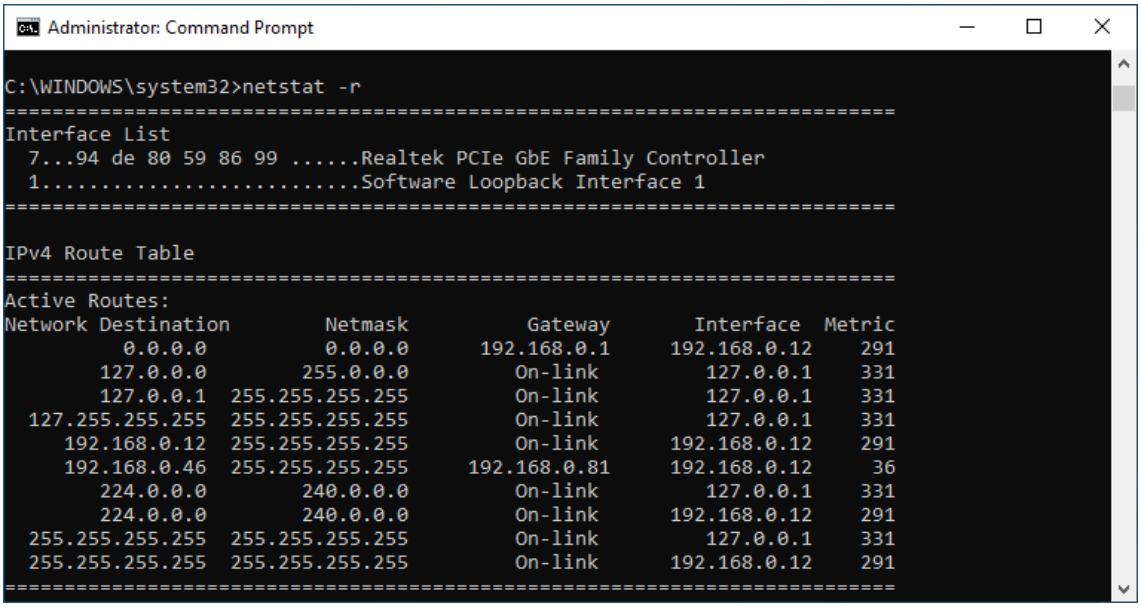


Fig 12 : Display of routing information at source node 192.168.0.12

You can observe that for the Destination node 192.168.0.46, the gateway address assigned is 192.168.0.81 (IP Address of the system where NetSim Emulation server is running)

8. Open command line at Source node 192.168.0.12 and enter the command.
   - ping 192.168.0.46 -t

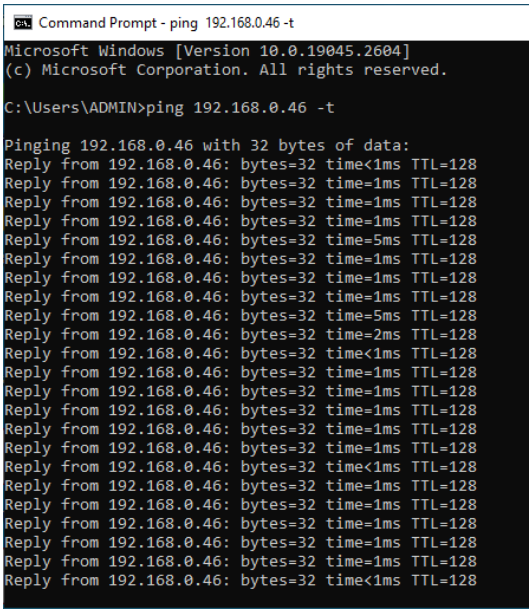


Fig 13 : Pinging to destination IP 192.168.0.46

Results and discussion

After the simulation is completed, you can observe the results using Wireshark captured files. In the Result Dashboard, On the left side, Packet Capture → Emulation and you can see all Emulated Packets captured.

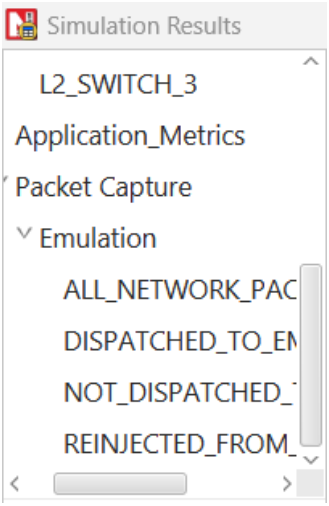


Fig 14: Emulation Packet Capture in Result Dashboard

We can observe original ping traffic generated at the source 192.168.0.12

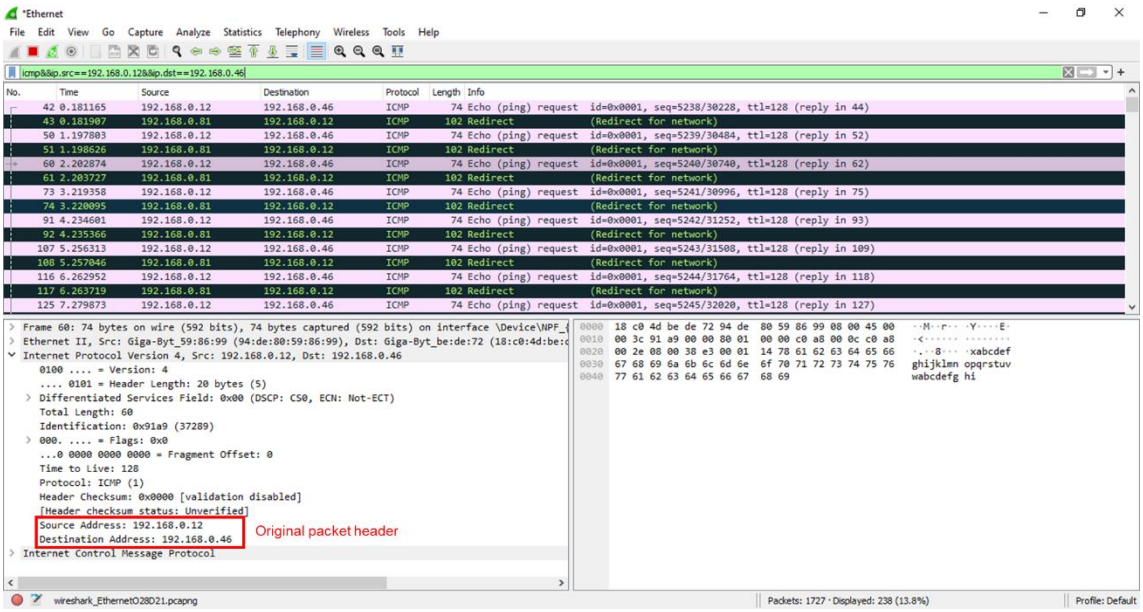


Fig 15: Original ICMP traffic generated from real source 192.168.0.12, captured using Wireshark.

We can observe false data injected packets in the false destination node 192.168.0.68

Note: *You should select the any ICMP Packet to observe the changes.*

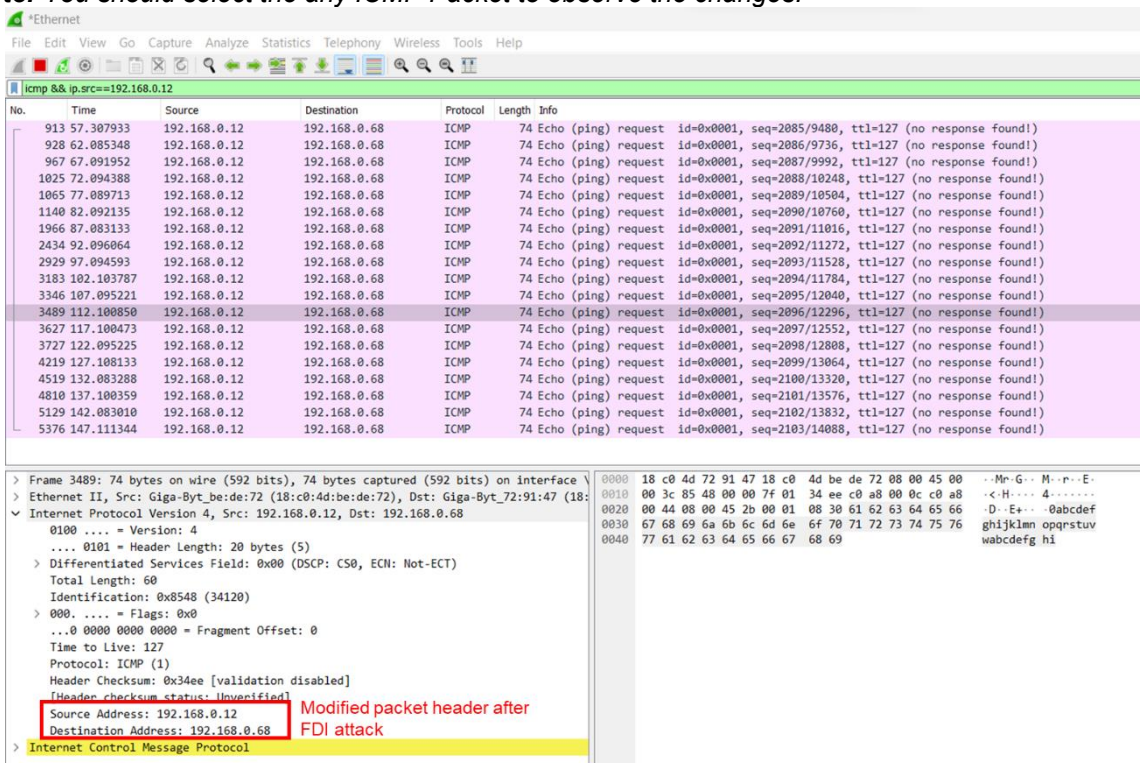


Fig 16: FDI Traffic captured by the destination 192.168.0.68, which is the false data Injected in the ICMP packet header by NetSim.

We can observe that the original ping traffic generated by the source 192.168.0.12 destined to 192.168.0.46 was passed via NetSim Emulation server 192.168.0.81. At the NetSim Emulation server we implemented the FDI attack. After the FDI attack in NetSim will reinject the modified packet to the actual network with Destination IP modified to 192.168.0.68. You can observe that the real destination will not receive any ICMP Packets from source 192.168.0.12, since

the destination address in different. If there is a machine with IP 192.168.0.68 in the network, then that machine will now receive the ICMP traffic from source 192.168.0.12.

Two Types of false data injection attacks: payload modification and header modification

In each of the two cases described earlier, we can model two kinds of attacks:

1. **Packet payload change**: The PING packet by default has its payload as *abcdefghijklmnopqrstuvwabcdefghi*, we modify this to *AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA*
2. **Packet header change**: The destination IP address of the ping is changed from *192.168.0.46* to *192.168.0.68*

Appendix: NetSim source code modifications

MS Visual Studio Development environment is required for editing and building NetSim source codes. Please see this link on setting up Visual Studio https://support.tetcos.com/support/solutions/articles/14000138721-what-components-of-visual-studio-community-2022-to-install-and-configure-to-work-with-netsim-source-c

To open our project source code section, in NetSim home screen to →your work → source code → open code.

NetSim comes with inbuilt low-level functions to capture packets. This code is not open for user modification. The code to access the payload/header and to modify the payload/header is open to users and can be modified. We show below the source code changes we have made in red. Users can alter these functions to implement their own FDI attacks. Once the code changes done rebuild the project by right click on IP project→ Rebuild, Once you rebuild is successful the project code modification will be affected in NetSim.

Case 1: Payload modification

Add a new function before **fn_NetSim_IP_Run()** and after **ip_handle_processing_delay() in IP.c file, in IP project.**

```
static void ip_handle_processing_delay()
{
        .....
}

// Function to calculate the Internet Checksum
uint16_t calculateChecksum(const uint8_t* data, size_t length) {
        uint32_t sum = 0;

        // Process each 16-bit chunk of data
        while (length > 1) {
                sum += ((uint16_t)data[0] << 8) + data[1];
                data += 2;
                length -= 2;
        }

        // If there's a remaining odd byte, add it to the sum
        if (length == 1) {
                sum += ((uint16_t)data[0] << 8);
```

```
	}

	// Fold the 32-bit sum to a 16-bit checksum
	while (sum >> 16) {
		sum = (sum & 0xFFFF) + (sum >> 16);
	}

	// Return the one's complement of the final sum
	return (uint16_t)(~sum);
}

//Seperate into 2 Bytes
static void separateBytes(uint16_t value, uint8_t* highByte, uint8_t* lowByte) {
	*highByte = (uint8_t)(value >> 8);  // Get the high byte
	*lowByte = (uint8_t)(value & 0xFF); // Get the low byte
}

/**
This function is called by NetworkStack.dll, whenever the event gets triggered
inside the NetworkStack.dll for IP.It includes NETWORK_OUT,NETWORK_IN and
TIMER_EVENT.
*/
_declspec(dllexport) int fn_NetSim_IP_Run()
{
	.....
}
```

Changes to **fn_NetSim_IP_Run() in IP.c file, in IP project**

```
_declspec(dllexport) int fn_NetSim_IP_Run()
{
	//False Data
	char s[BUFSIZ] = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA";
	uint8_t checkszero = 0x00;
	uint8_t packet_icmp[40];

	switch (pstruEventDetails->nEventType)
	{
		case NETWORK_OUT_EVENT:
		{
			ptrIP_FORWARD_ROUTE route = NULL;
			NetSim_PACKET* packet = pstruEventDetails->pPacket;
			NETWORK_LAYER_PROTOCOL nLocalNetworkProtcol;
			//False Data Injection in Network Layer into packet payload and
			regenerate the checksum field.
			if (packet)
			{
				//Device ID of Attacker
				if (pstruEventDetails->nDeviceId == 4)
				{
					for (int i = 28; i < 60; i++)
						packet->szPayload->packet[i] = s[i - 28];
						//Modifying the payload by inserting False Data
				}
```

```
                        //Checksum Recalculation
                        if (pstruEventDetails->nDeviceId == 4)
                        {
                                // Read the packet data into a C array.
                                unsigned char* packet_data = (unsigned char*)packet-
>szPayload->packet;

                                //Extract the ICMP Packet Payload
                                for (int k = 20; k < 60; k++) {
                                        packet_icmp[k - 20] = (uint8_t)packet_data[k];
                                }

                                //Set the Checksum Variable to 0 while calculating the
                                checksum
                                packet_icmp[2] = checkszero;
                                packet_icmp[3] = checkszero;

                                //Calculate the new checksum value for ICMP Packet
                                Payload
                                size_t length = sizeof(packet_icmp);
                                uint16_t checksum = calculateChecksum(packet_icmp,
length);

                                //Separate the 16-bit value to two 8-bit values
                                uint8_t highByte, lowByte;
                                separateBytes(checksum, &highByte, &lowByte);

                                //Update the checksum value in checksum field
                                packet->szPayload->packet[22] = highByte;
                                packet->szPayload->packet[23] = lowByte;
                        }
                }
                nLocalNetworkProtcol                                            =
fnGetLocalNetworkProtocol(pstruEventDetails);
                if (nLocalNetworkProtcol)
                {
                        fnCallProtocol(nLocalNetworkProtcol);
                        return 0;
                }

        ……
        ……
        }
    …….
    …….
    }
}
```

Case 2: Header modification

Changes to **fn_NetSim_IP_Run() in IP.c file, in IP project**

```
_declspec(dllexport) int fn_NetSim_IP_Run()
```

```
{
	//False Data
	char s[BUFSIZ] = "D";  //hexadecimal value for D is 68
	switch (pstruEventDetails->nEventType)
	{
		case NETWORK_OUT_EVENT:
		{
			ptrIP_FORWARD_ROUTE route = NULL;
			NetSim_PACKET* packet = pstruEventDetails->pPacket;
			NETWORK_LAYER_PROTOCOL nLocalNetworkProtcol;
			// False Data Injection in Network Layer into packet header
			if (packet)
			{
				//Device ID of Attacker
				if (pstruEventDetails->nDeviceId == 1){
					for (int i = 19; i < 20; i++)
						packet->szPayload->packet[i] = s[i - 19];
				}
			}
			nLocalNetworkProtcol =
fnGetLocalNetworkProtocol(pstruEventDetails);
			if (nLocalNetworkProtcol)
			{
				fnCallProtocol(nLocalNetworkProtcol);
				return 0;
			}
		.....
		.....
		}
	.....
	.....
	}
}
```