

## False Data Injection Attack

**Two cases:** Simulation and Emulation

**Two Types:** Payload modification and Header modification

---

**Software:** NetSim Standard v13.3 (64 bit), Visual Studio 2022

**Project code download link:** [https://github.com/NetSim-TETCOS/False-Data-Injection-Attack-in-Internetworks\\_v13.3/archive/refs/heads/main.zip](https://github.com/NetSim-TETCOS/False-Data-Injection-Attack-in-Internetworks_v13.3/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.

FDI attacks can take various forms, such as modifying data in transit or at rest, manipulating data to cause system malfunctions, and altering system configurations. Attackers can use these methods to achieve different objectives, such as stealing sensitive data, causing financial losses, or disrupting critical infrastructure.

### Simulating an FDI Attack using NetSim

In this toy example, we launch an FDI attack on ICMP ping messages between a source and destination. We show two types of false data injection 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*

The first attack is carried out within NetSim Simulator while the second attack is executed using NetSim Emulator.

### 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.

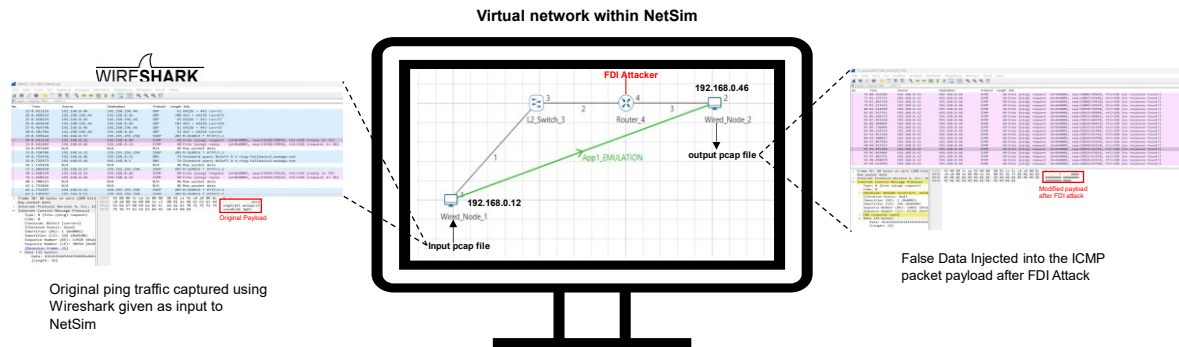


Fig 1: PING application between a real source and real destination is captured as a pcap file and given as an input to a virtual source inside NetSim. In this example, the source IP is set to 192.168.0.12 and the destination IP is set to 192.168.0.46. The external pcap file is available in the project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

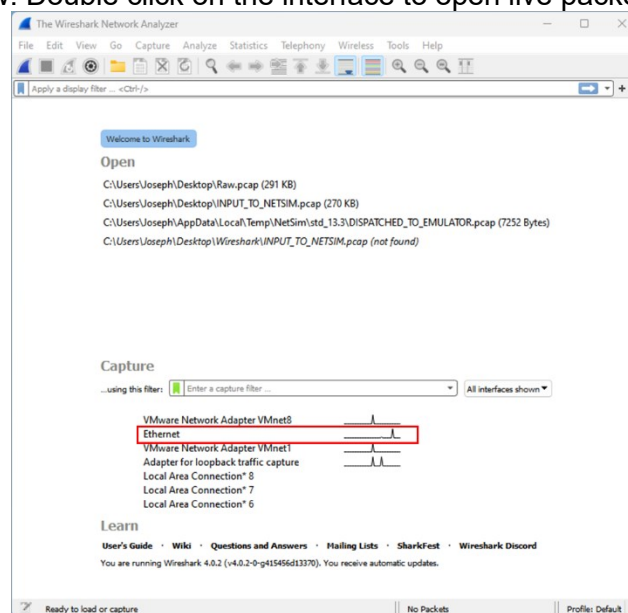
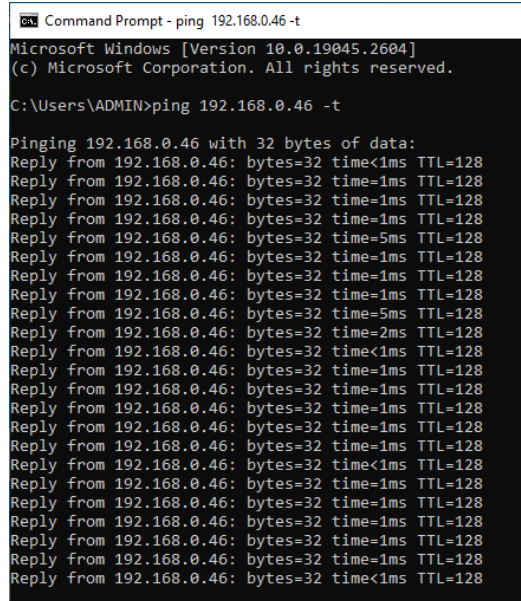


Fig 2: Select packet capture interface to capture packets at source.

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



```

Command Prompt - ping 192.168.0.46 -t
Microsoft Windows [Version 10.0.19045.2604]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ADMIN>ping 192.168.0.46 -t

Pinging 192.168.0.46 with 32 bytes of data:
Reply from 192.168.0.46: bytes=32 time<1ms TTL=128
Reply from 192.168.0.46: bytes=32 time=1ms TTL=128
Reply from 192.168.0.46: bytes=32 time=1ms TTL=128
Reply from 192.168.0.46: bytes=32 time=1ms TTL=128
Reply from 192.168.0.46: bytes=32 time=5ms TTL=128
Reply from 192.168.0.46: bytes=32 time=1ms TTL=128
Reply from 192.168.0.46: bytes=32 time=1ms TTL=128
Reply from 192.168.0.46: bytes=32 time=1ms TTL=128
Reply from 192.168.0.46: bytes=32 time=5ms TTL=128
Reply from 192.168.0.46: bytes=32 time=2ms TTL=128
Reply from 192.168.0.46: bytes=32 time<1ms TTL=128
Reply from 192.168.0.46: bytes=32 time=1ms TTL=128
Reply from 192.168.0.46: bytes=32 time=1ms TTL=128
Reply from 192.168.0.46: bytes=32 time=1ms TTL=128
Reply from 192.168.0.46: bytes=32 time=1ms TTL=128
Reply from 192.168.0.46: bytes=32 time<1ms TTL=128
Reply from 192.168.0.46: bytes=32 time=1ms TTL=128
Reply from 192.168.0.46: bytes=32 time=1ms TTL=128
Reply from 192.168.0.46: bytes=32 time=1ms TTL=128
Reply from 192.168.0.46: bytes=32 time=1ms TTL=128
Reply from 192.168.0.46: bytes=32 time=1ms TTL=128
Reply from 192.168.0.46: bytes=32 time=1ms TTL=128
Reply from 192.168.0.46: bytes=32 time=1ms TTL=128
Reply from 192.168.0.46: bytes=32 time<1ms TTL=128

```

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 the ping traffic capture is finished stop the Wireshark packet capture 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

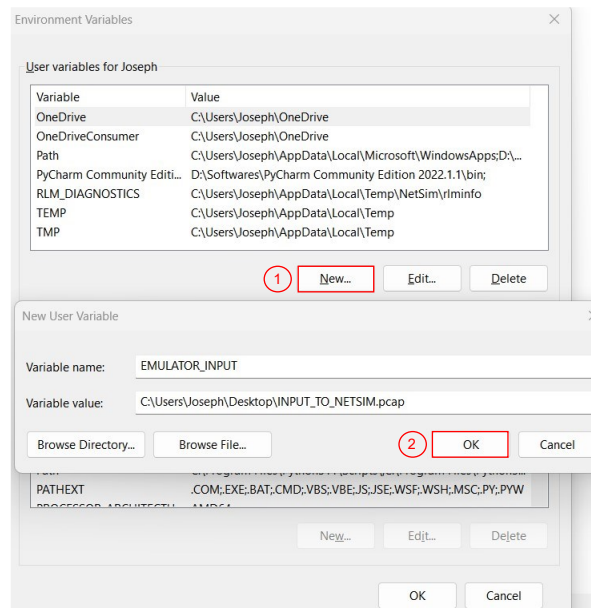


Fig 4 : Environment Variable Path

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

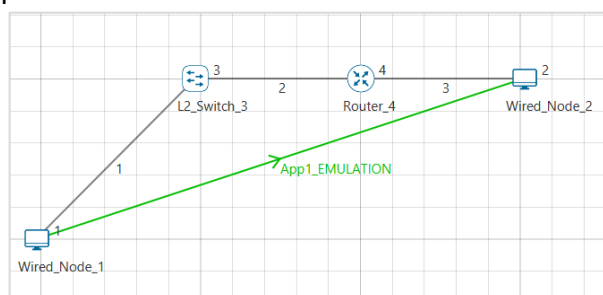


Fig 5: NetSim Emulation Scenario, Wired\_Node\_1 device mapped for Source IP 192.168.0.12 and Wired\_Node\_2 device mapped for Destination IP 192.168.0.46

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

### Results and Discussions

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

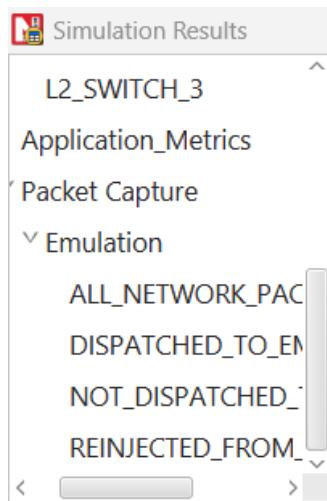


Fig 6: Emulation Packet Capture in Result Dashboard

We can observe original packets in the DISPATCHED\_TO\_EMUALTOR.pcap file.

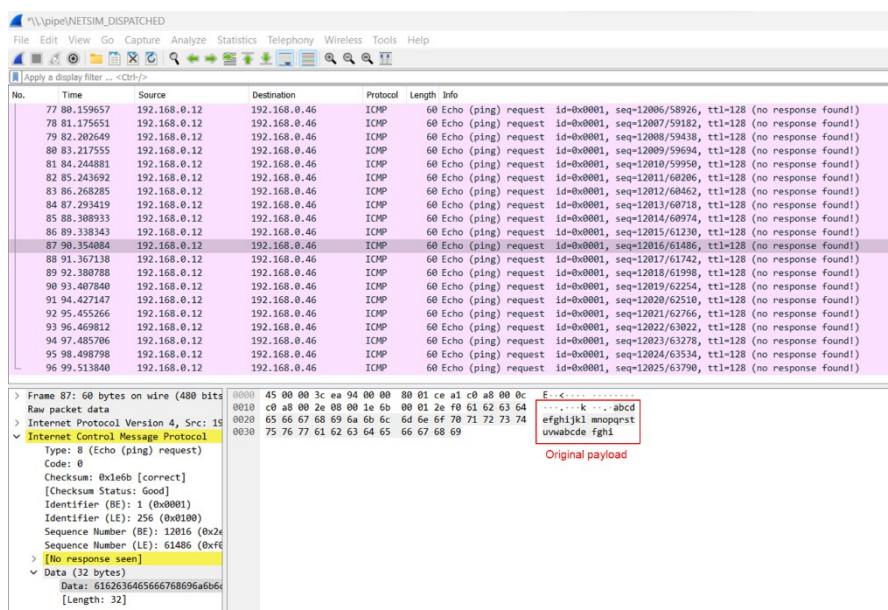


Fig 7: Original payload captured by NetSim emulator

We can see false data injected packets in the REINJECTED\_FROM\_EMUALTOR.pcap file

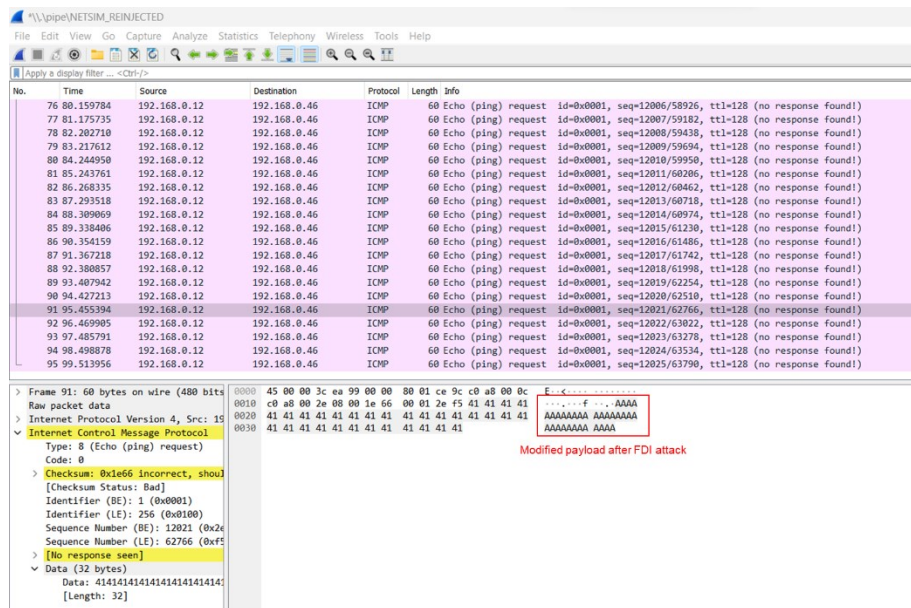


Fig 8: Traffic with false data injected. Notice the difference in payload.

## 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

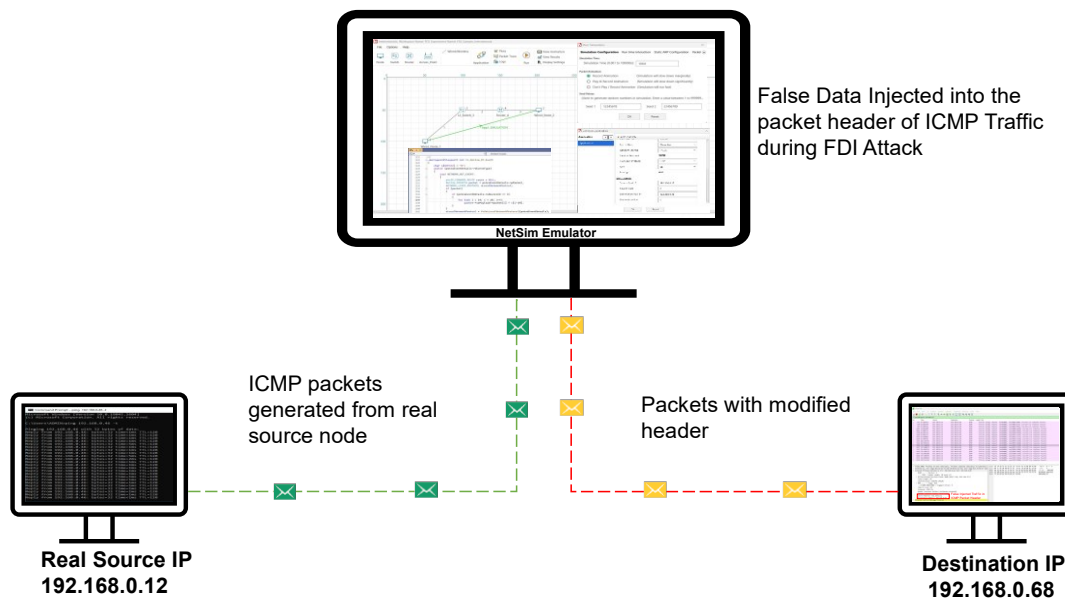


Fig 9 : PING application between source and destination. The source IP is set to 192.168.0.12 and the destination IP is set to 192.168.0.46. In NetSim We are implementing FDI Attack by modifying the destination IP address to 192.168.0.68 in ICMP packet header.

### 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

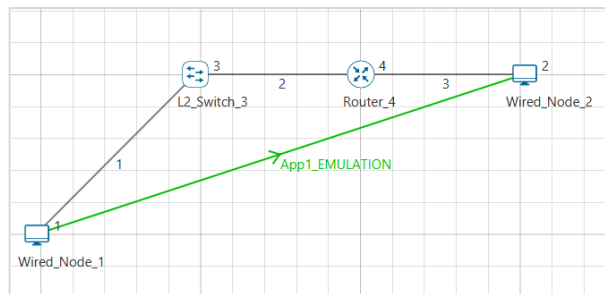


Fig 10: NetSim Emulation Scenario, Wired\_Node\_1 device mapped for Source IP 192.168.0.12 and Wired\_Node\_2 device mapped for Destination IP 192.168.0.46

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

### 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
  - a. **route add 192.168.0.46 mask 255.255.255.255 192.168.0.81 metric 1**
  - b. After execution, you will get “OK”.

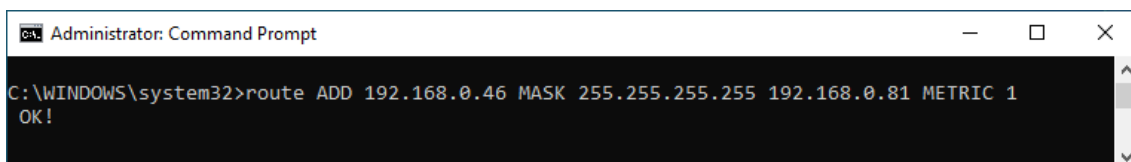


Fig 11: Adding the Static route from source to destination via gateway as NetSim emulation server-192.168.0.81

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

```

Administrator: Command Prompt
C:\WINDOWS\system32>netstat -r

=====
Interface List
7...94 de 80 59 86 99 .....Realtek PCIe GbE Family Controller
1.....Software Loopback Interface 1
=====

IPv4 Route Table
=====
Active Routes:
Network Destination        Netmask          Gateway          Interface        Metric
0.0.0.0                    0.0.0.0          192.168.0.1      192.168.0.12     291
127.0.0.0                  255.0.0.0        On-link          127.0.0.1        331
127.0.0.1                  255.255.255.255  On-link          127.0.0.1        331
127.255.255.255            255.255.255.255  On-link          127.0.0.1        331
192.168.0.12               255.255.255.255  On-link          192.168.0.12     291
192.168.0.46               255.255.255.255  192.168.0.81     192.168.0.12     36
224.0.0.0                  240.0.0.0        On-link          127.0.0.1        331
224.0.0.0                  240.0.0.0        On-link          192.168.0.12     291
255.255.255.255            255.255.255.255  On-link          127.0.0.1        331
255.255.255.255            255.255.255.255  On-link          192.168.0.12     291
=====

```

Fig 12 : Display of routing information at source node 192.168.0.12

We can observe that for the node 192.168.0.46, the gateway address assigned is 192.168.0.81, which is the IP Address of the system where NetSim Emulation server is running

## Results and discussion

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

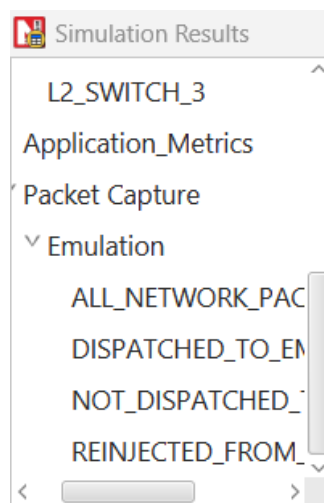


Fig 13: Emulation Packet Capture in Result Dashboard

We observe original ping traffic generated at the source 192.168.0.12



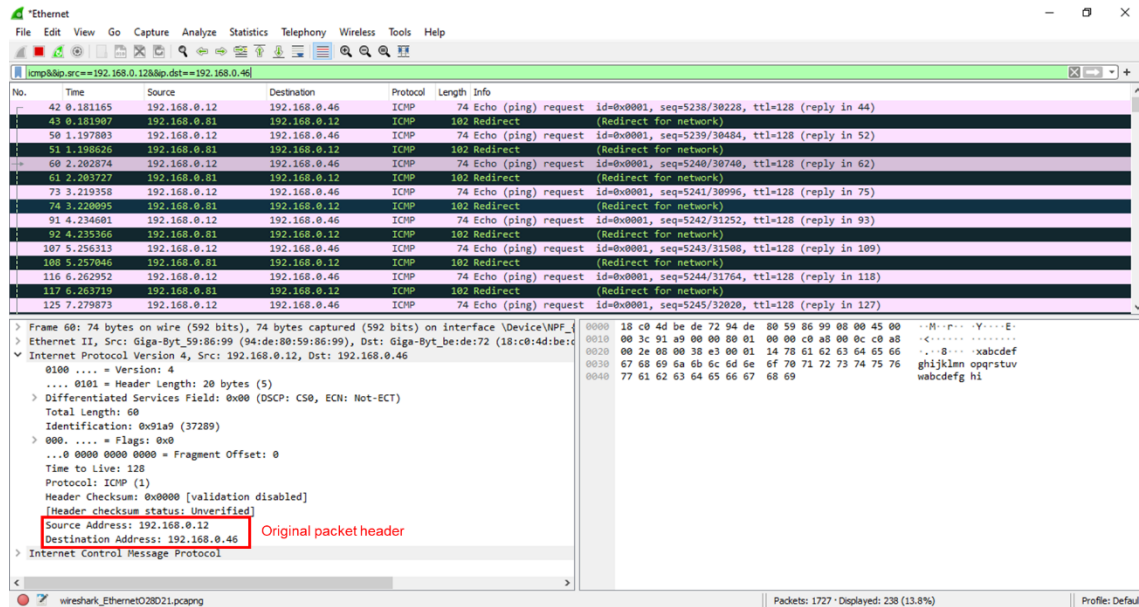


Fig 14: Original ping traffic generated from real source 192.168.0.12, captured using Wireshark.

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

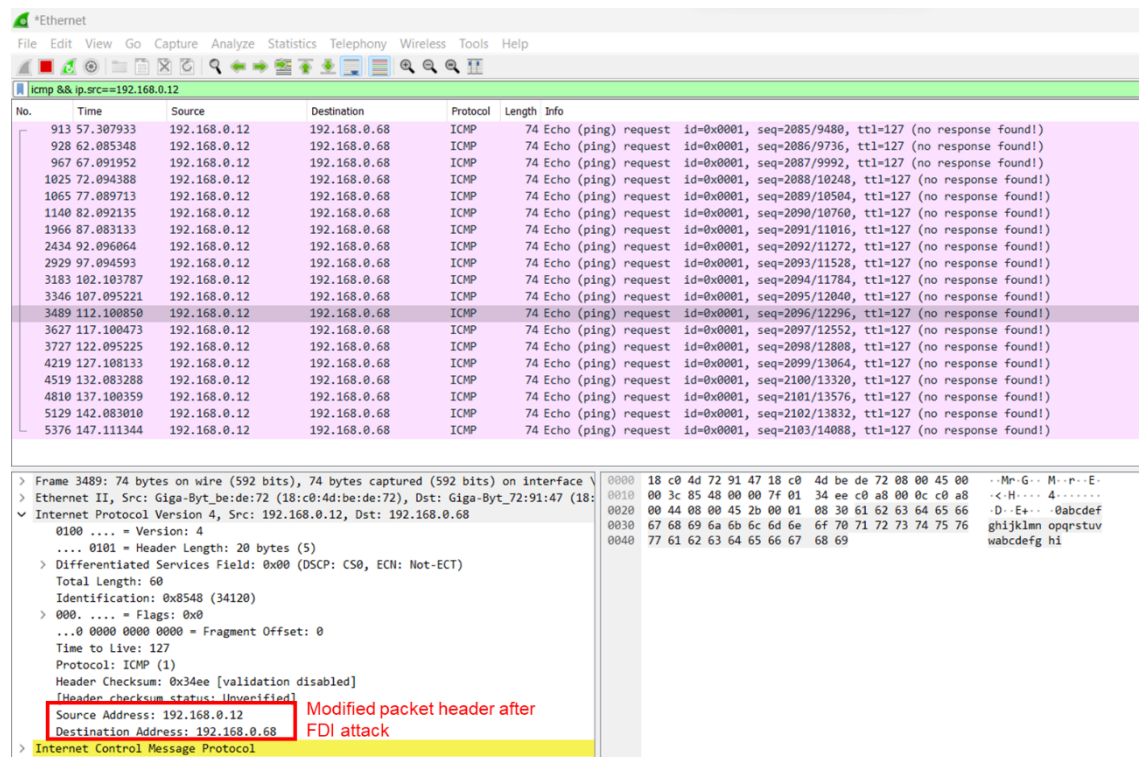


Fig 15: FDI Traffic captured by the destination 192.168.0.68, which is the false data injected in the ICMP packet header by NetSim.

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 server we implemented the FDI attack. After the FDI attack, NetSim will reinject the modified packet to the actual network with Destination IP modified to 192.168.0.68. We see that the real destination will not receive any ICMP Packets from source 192.168.0.12, since the destination address is now different in the packets. 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.

## 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.

---

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

---

#### Case 1: Payload modification

```
_declspec(dllexport) int fn_NetSim_IP_Run()
{
    //False Data
    char s[BUFSIZ] = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA";
    switch (pstruEventDetails->nEventType)
    {
        case NETWORK_OUT_EVENT:
        {
            ptrIP_FORWARD_ROUTE route = NULL;
            NetSim_PACKET* packet = pstruEventDetails->pPacket;
            NETWORK_LAYER_PROTOCOL nLocalNetworkProtocol;
            //False Data Injection in Network Layer into packet payload
            if (packet)
            {
                //Device ID of Attacker
                if (pstruEventDetails->nDeviceId == 4){
                    for (int i = 28; i < 60; i++)
                        packet->szPayload->packet[i] = s[i - 28];
                }
            }
            nLocalNetworkProtocol =
fnGetLocalNetworkProtocol(pstruEventDetails);
            if (nLocalNetworkProtocol)
            {
                fnCallProtocol(nLocalNetworkProtocol);
                return 0;
            }
            .....
            .....
        }
        .....
    }
}
```

**Case 2: Header modification**

```

_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 nLocalNetworkProtocol;
            // 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];
                }
            }
            nLocalNetworkProtocol =
fnGetLocalNetworkProtocol(pstruEventDetails);
            if (nLocalNetworkProtocol)
            {
                fnCallProtocol(nLocalNetworkProtocol);
                return 0;
            }
            .....
            .....
        }
        .....
        .....
    }
}

```