

Laboratory: 2

Spoofing Exercise

Introduction

This report includes use of Nmap and Scapy tools for checking active hosts over the network and creating spoofed packets respectively to send over the network. Wireshark is used to capture and display the spoofed packets that are transmitted and to monitor if there are any other spoofing attempts. For this, we set up an environment as described below and generate traffic over the network from which we sniff for the packets.

Some similar tools like Nmap are: Angry IP scanner, Masscan, Dipiscan etc.

Alternates for Scapy: Python sql, Ironic, pandas python etc.

Setup and Process

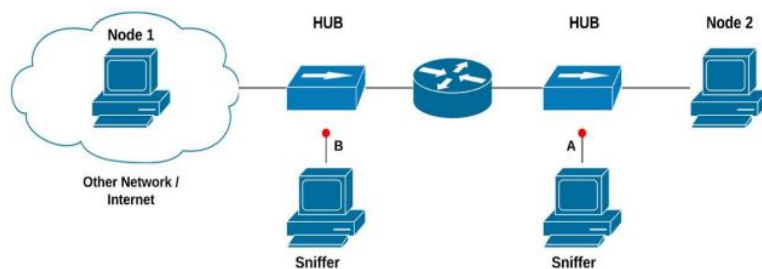


Figure1: Network set-up

Here, as shown above, node 1 (192.168.10.10) is machine from lab in clemson and node 2 (192.168.20.13) is machine from lab in charleston. All the machines are connected over the HUB and the red dots are the sniffing points (192.168.10.9). Snooping machine that I will be using for creating and sending packets (192.168.10.111) is connected to the same sub-net.

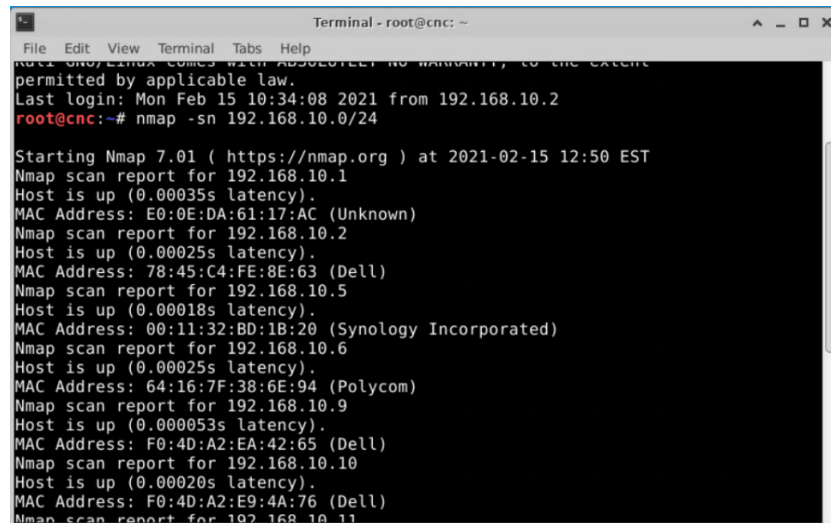
We login to sniffer machine using ssh and open wireshark and start the capture with enp2s0 interface. It immediately starts to capture packets over the wire. Now we ssh to spoofing machine simultaneously in a different terminal where we would check for active hosts in our subnet.

In order to make your spoofed packets harder to be detected, it is advised to use the IP address available over the network or in a range of what is accepted over the particular network.

In my case, I'm using Nmap to discover all the active hosts in my subnet. Using the following command:

```
$ nmap -sn 192.168.10.0/24
```

As shown in the figure: 2 below, all the active hosts MAC addresses and IP addresses are displayed.

A terminal window titled "Terminal - root@cnc: ~" showing the output of an Nmap scan. The command entered is "nmap -sn 192.168.10.0/24". The output lists several active hosts with their IP addresses and MAC addresses. The hosts are 192.168.10.1, 192.168.10.2, 192.168.10.5, 192.168.10.6, 192.168.10.9, 192.168.10.10, and 192.168.10.11. Each host entry includes the IP address, a latency measurement, and the MAC address with its vendor name in parentheses. For example, 192.168.10.1 has MAC address E0:0E:DA:61:17:AC (Unknown). The terminal window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help".

```
root@cnc:~# nmap -sn 192.168.10.0/24

Starting Nmap 7.01 ( https://nmap.org ) at 2021-02-15 12:50 EST
Nmap scan report for 192.168.10.1
Host is up (0.00035s latency).
MAC Address: E0:0E:DA:61:17:AC (Unknown)
Nmap scan report for 192.168.10.2
Host is up (0.00025s latency).
MAC Address: 78:45:C4:FE:8E:63 (Dell)
Nmap scan report for 192.168.10.5
Host is up (0.00018s latency).
MAC Address: 00:11:32:BD:1B:20 (Synology Incorporated)
Nmap scan report for 192.168.10.6
Host is up (0.00025s latency).
MAC Address: 64:16:7F:38:6E:94 (Polycom)
Nmap scan report for 192.168.10.9
Host is up (0.000053s latency).
MAC Address: F0:4D:A2:EA:42:65 (Dell)
Nmap scan report for 192.168.10.10
Host is up (0.00020s latency).
MAC Address: F0:4D:A2:E9:4A:76 (Dell)
Nmap scan report for 192.168.10.11
```

Figure 2: Active hosts in Clemson lab

Now that the active hosts IP addresses are available, we need to create packet by defining the source, destination and payload. I'm using scapy for doing the same.

First, in the snooping machine enter the scapy interactive session.

Before sending the spoofed packets I'm sending a normal packet using scapy to Charleston machine (192.168.20.13). A normal packet with the message is sent over the wire as seen in the figure:3 below. The command I used is: `send(IP(dst='192.168.20.13')/'No spoofing')`. Here, you can see the source IP address is of our machine (192.168.10.111)

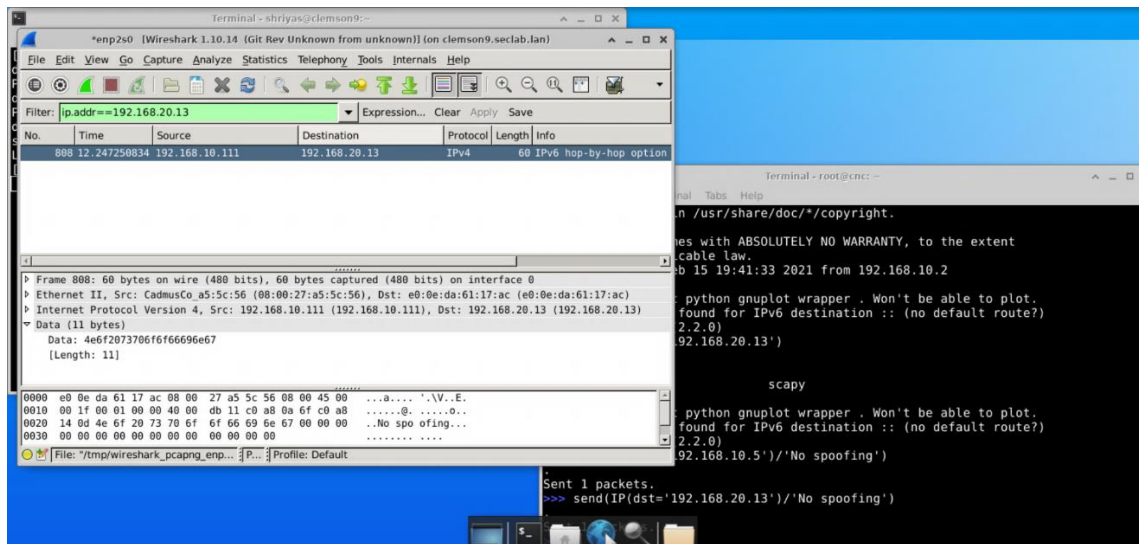


Figure 3: Sending normal packet over the wire

Now, for spoofing we can use the following command for creating the packet: `Send(IP(dst='<destination IP address>',src='<spoofing address>')/'message')`

We can also use the below command for the same. It creates a DNS query with a spoofed source IP address: `sendp(Ether(src='12:34:56:78:9a:ab')/IP(dst='<destination IP address>',src='<Spoofing address>')/'message')`

In my case, for my first spoofing attempt I've created a packet to send to Charleston machine (192.168.20.13) with the source IP address of machine from Clemson (192.168.10.10) with the message "Quick brown fox jumps over a lazy dog". As you can see in the figure:4 below. When you use the filter "ip.addr" with the destination IP, you see all the packet communications to the given IP. We can confirm this is the packet I spoofed by checking the data section. It contains the message "Quick brown fox jumps over a lazy dog". The source address of the packet capture is not of the machine I used to send the packet (192.168.10.111) but instead is the spoofed IP address (192.168.10.10). With this we can confirm that my attempt to send spoofed packets to a particular Charleston machine (192.168.20.13) is successful.

Command I used for creating spoofed packet is: `Send(IP(dst='192.168.20.13',src='192.168.10.10')/'Quick brown fox jumps over a lazy dog')`

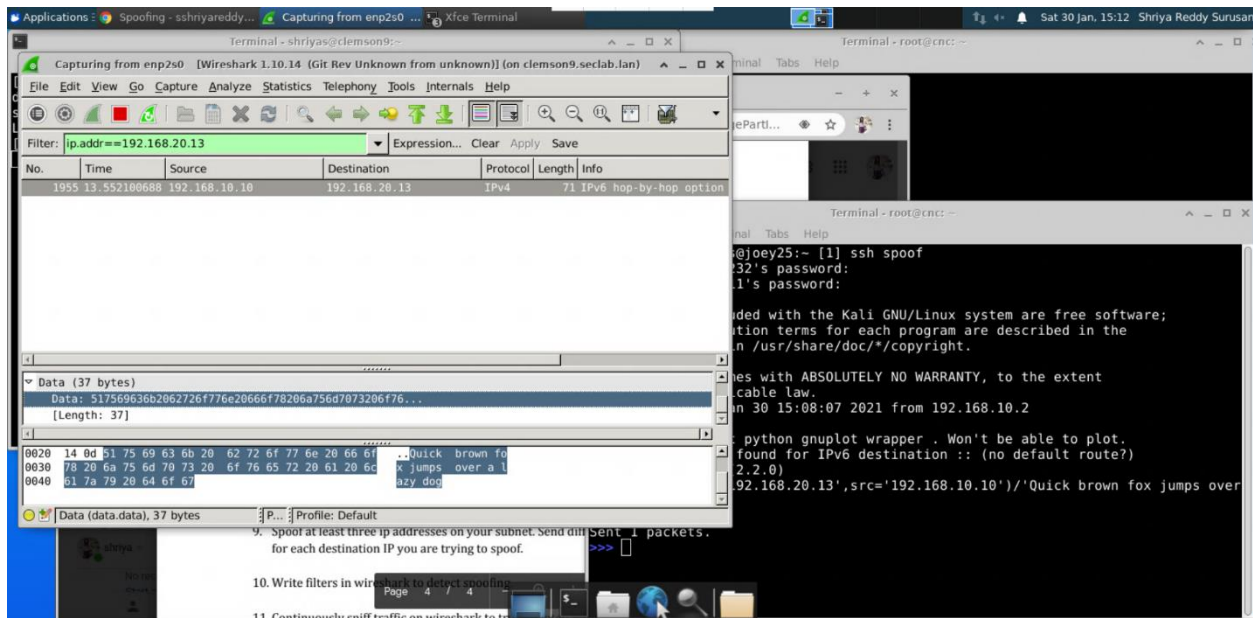


Figure 4: First Spoofed packet

My second spoofing attempt involved sending packet to Charleston machine (192.168.20.13) with spoofing another active Clemson machine (192.168.10.15) with the message “spoofing second messege “. You can see in the figure:5 below my packet is successfully sent with source IP address as “192.168.10.15”. Command I used is: `Send(IP(dst='192.168.20.13',src=' '192.168.10.15')/ spoofing second messege')`.

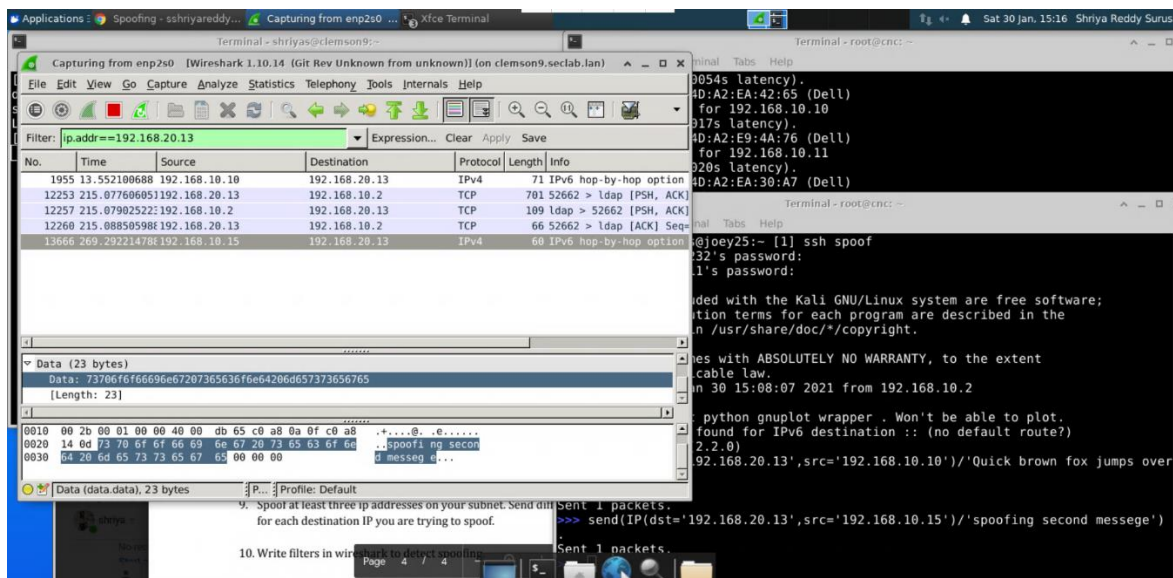


Figure 5: Second Spoofed packet

My third spoofing attempt involved sending packet to Charleston machine (192.168.20.13) with source address as “192.168.10.200” and payload as “messege three”. In all my spoofing attempts I’ve taken the

IP addresses of active Clemson machine to spoof to evade easy detection over the network. In the figure:6 below. My third spoofed packet is captured and you can see the source IP of it is displayed as “192.168.10.200” and not of my machine “192.168.10.111”. Hence, spoofing was successful.

The filter I used to capture my spoofed packets is “*ip.addr == 192.168.20.13*”. Since I used the same destination address to send my spoofed packets to, it displays all the communications to and from it(which can be seen below). Figure: 5 below shows all my spoofed packets.

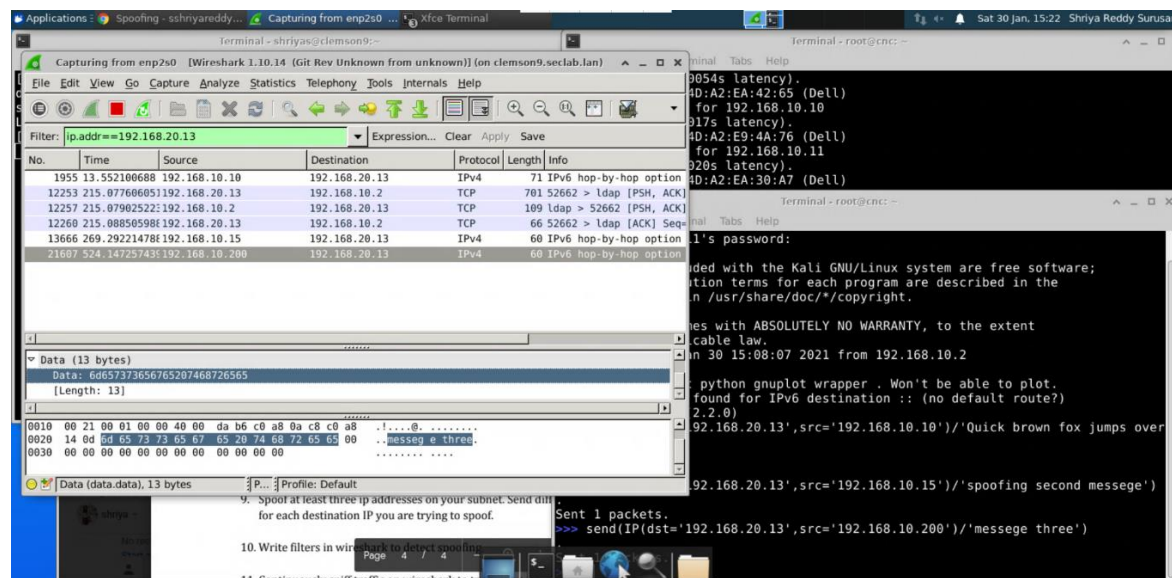


Figure 6: Second Spoofed packet

Now that we have successfully spoofed the packets. We will try intercepting the normal traffic to find if there are any other potential spoofing attempts. For this, Wireshark is started with enp2s0 interface and stopped after a few minutes such that we have captured enough traffic to analyze. We initially look into the traffic flow scanning all the communication between the IP addresses. For me, I’ve come across a source IP address of a packet (highlighted in the figure:7) ‘19.3.3.3’ which did not match any machine IP’s connected to the subnet nor it was even in the range of IP’s of the machines in the subnet. That was my first clue that it might be a spoofed packet. On further inspection of the packet, I could see the data size of it was 13 bytes which when compared to a normal packet is far less. This was my second clue.

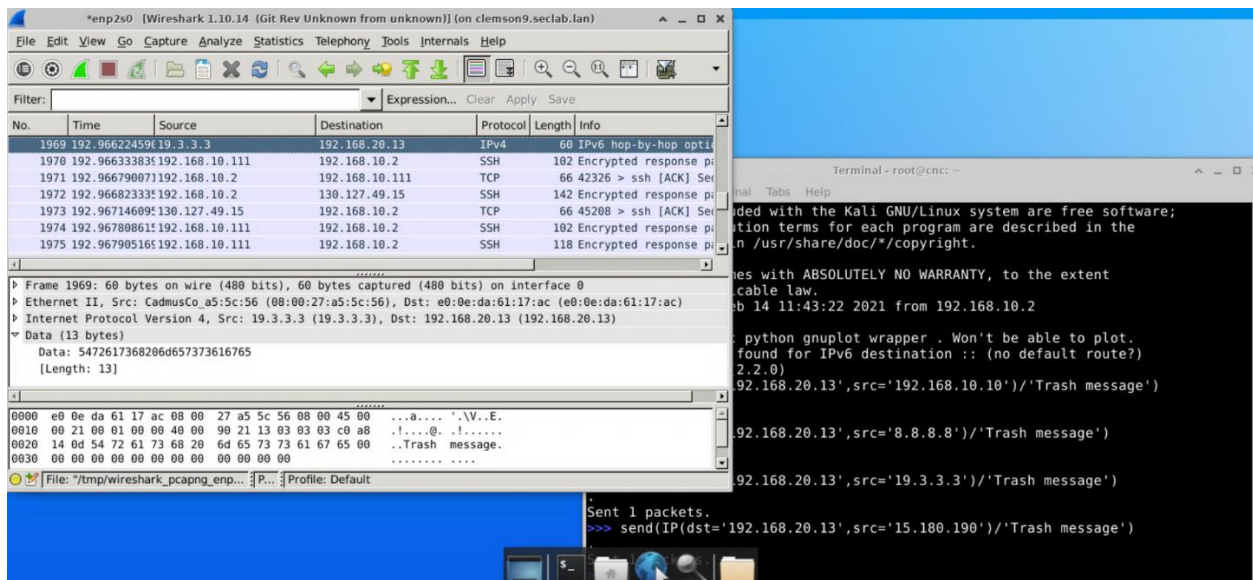


Figure 7: Potential spoofed packet

I could not see any other communications between this source IP address '19.3.3.3' (Figure:8) and destination "192.168.20.13". Though I cannot say for sure that getting no reply back to 19.3.3.3 is a factor for calling it a spoofed packet. But, combined with the other clues mention above I believe it is spoofed.

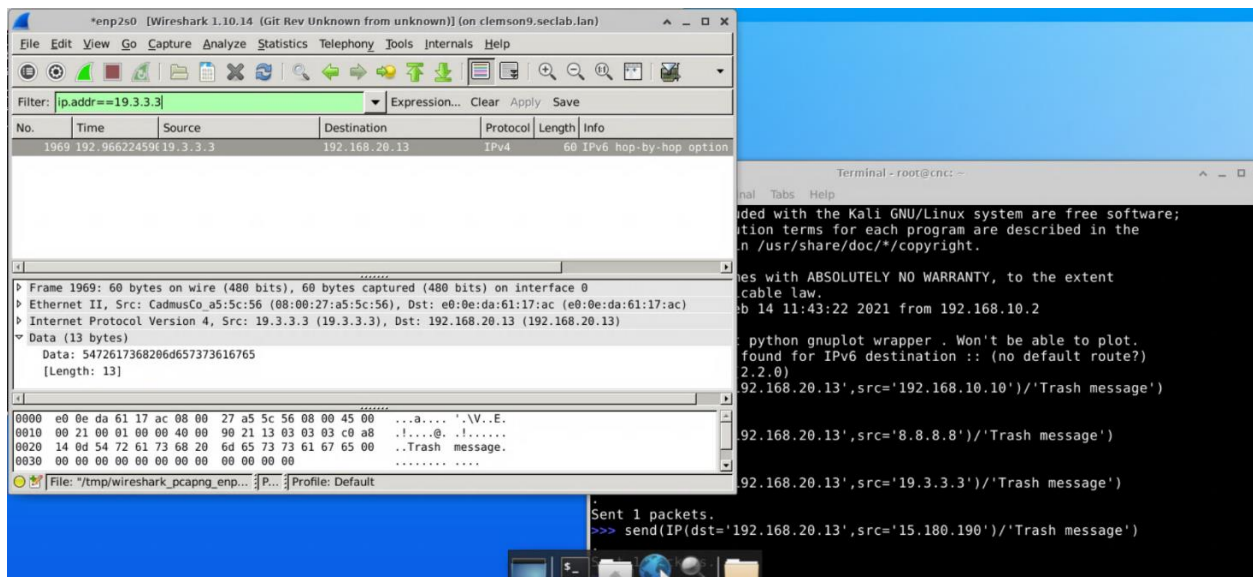


Figure 8: Checking all the communication from and to IP 19.3.3.3 using the filter

Questions

Question 1: How can you evade detection when spoofing others? Is it possible to ascertain the identity of the sender?

While sending the spoofed packets, one can avoid detection if the following are taken care of:

i) The source address used should be in the range of IP's allowed over the network or should be of any active IP's over the network to avoid being strange one out.

ii) The size of the data sent through the packets should not be too small. It draws attention of the researcher that it might be falsified information that's being sent.

Usually, deriving IP packet header data would give out the information of the sender such as geographic location of attacker's resources. But with IP address spoofed packets, this is not very reliable since the IP address and the geographic location of the traffic is masked. Thus, not knowing for sure who the actual sender is.

Question 2: Explain what ip address spoofing is, and what a host on the network must do to spoof its ip address

IP packet header consists of a lot of fields, of which consists of source and destination IP addresses. The process of creating an IP packet with a source address of somebody else to mask the real source identity is known as IP address spoofing. Here, the normal network connection is disrupted because when an IP address is spoofed, it causes the responses to be misdirected. This technique is very efficient for performing DDOS, since the responses from the pinged server to the spoofed address can overwhelm the victim (depending on how many packets were spoofed and sent to the server per second).

There are multiple tools available to change these IP packet header fields. What we need to do specifically in it to spoof the packets is to create it with spoofing IP address as the 'Source IP address' in header field before transferring it over the network such that the protocol used in the transport layer wraps the data with header information containing the spoofed address which in turn is wrapped again by IP protocol in internet layer with header information containing the spoofed address. One such tool available to do it is 'Scapy'. The following command can be used to do so:

```
send(IP(dst='<Destination IP>',src='<Spoofing address>')/'<Message to transfer>')
```

Question 3: Explain why an attacker cannot just grab any existing IP packet carrying UDP or TCP, change only the IP addresses in there, and expect the target host to accept the packet.

When the attacker grabs an existing IP packet. The data in IP packet in question is wrapped with 2 headers. First header being from the transport layer, where, UDP or TCP wraps the data with its own header containing the source address and destination address along with other fields and passes it down to

Internet layer. Here, the datagram descended from the upper layer is then fragmented into IP packets. Each fragmented packet is wrapped with IP protocol header containing source and destination addresses along with its other fields. Now, if the attacker just changes the source IP address from the header of IP packets. The receiving end point to assemble these packets back to datagram has 2 different source addresses residing in the IP header and TCP or UDP header. The header checksum of the IP packet is also changed. Thus, notifying the target host about the corruption during the transmission. Hence, making it a terrible way to spoof packets.

References

- [1]: <https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture16.pdf>
- [2]: <https://security.stackexchange.com/questions/31999/how-are-spoofed-packets-detected>
- [3]: <https://www.ionos.com/digitalguide/server/security/syn-flood/>