21:03 Spoofing IP with IPIP

by Yannay Livneh

On the Internet, nobody knows you're a dog. Or so they said in 1993. IP, the most fundamental protocol of the Internet, does not enforce or verify the validity of the `source` field specified in the header of an IP packet. Anyone could just send packets spoofing whichever origin address as they liked. It was as easy as executing this Python code. (The `/` operator in the `scapy` package is used to stack the latter layer over the former.)

```
from scapy.all import *
packet = IP(src='13.37.13.37',
            dst='8.8.8.8')/"some data"
send(packet)
```

This made a lot of people very angry and been widely regarded as a bad move. So the elders of the Internet, the IETF, sat together in May 2000. They decided to drop packets they deemed fishy, and thus BCP 38 was born.[3] This fine document requires ISPs, the moderators of the Internet, to filter packets that originate from their customers with source IPs which were not assigned by the ISP.

Fast forward to 2020: many ISPs implemented this policy and cloud providers followed suit. Nowadays, the average Internet user can't really spoof IP packets. However, some machines in the Internet don't suffer from these policies. So if a user wants to spoof a packet, all they need to do is to ask one of these machines nicely to send a spoofed packet on the user's behalf. How does one ask a friendly machine to send a packet? Just send it over IP and the remote machine will do the rest. To illustrate it with Python code:

```
from scapy.all import *
packet = IP(src='13.37.13.37',
            dst='8.8.8.8')/"some data"
friendly_ip = '1.2.3.4'
send(IP(dst=friendly_ip, proto='ipip')/packet)
```

And this is it: all you need to do is find such a friendly machine and send it a spoofed packet to send using the somewhat forgotten "IP over IP" protocol (protocol number 4). This protocol was an early implementation for VPN. It's dead simple, just encapsulate another IP in an IP packet and send it. The receiver simply decapsulates the outer packet and sends the inner IP packet. No authentication, no filters, and no hassles. The Internet has evolved since those naïve days, but operating systems still implement this protocol. And sometimes, if you are lucky, some vendor opens it to the Internet for one reason or another. Surprisingly, this scenario happens quite more often than you might imagine. In fact, this is how I found it. I imagined the bug and then tried to scan the Internet to find such a machine.

This issue has more uses than simply spoofing, and some are worse than others (perhaps the subject for a future article). However, I find one of the uses rather amusing. Packet encapsulation is not limited and can be done multiple times in a recursive manner. The only limitation is the IP packet maximum length which is $2^{16}-1$. As every IP header size is at least twenty bytes, the limit for IPIP encapsulation is 3,276 layers. This is way more than the classic limitation of maximal network hops (TTL) a packet is allowed to in the IP protocol: 255. So using our new technique, we can craft the longest Pass-The-Parcel game in the history of the Internet. We can craft a single packet that would bounce around for a really long time, way more than you might have expected. I really like this idea.

Scanning code is attached to the PDF of this fine journal.[4] As for a proof that this technique works? Simply open `pocorgtfo21.pdf` in Wireshark![5]

[3] BCP38: Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing

[4] `unzip pocorgtfo21.pdf zmap-ipip.patch`

[5] `wireshark pocorgtfo21.pdf`