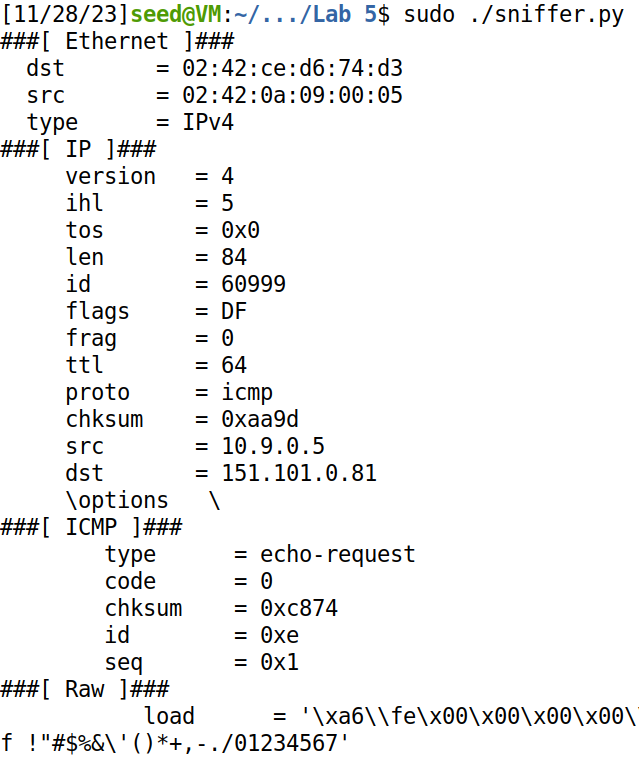
Sniffing/Spoofing Attacks (CS 915) Post-Lab Assignment Report

Hanzhi Zhang - 5525549

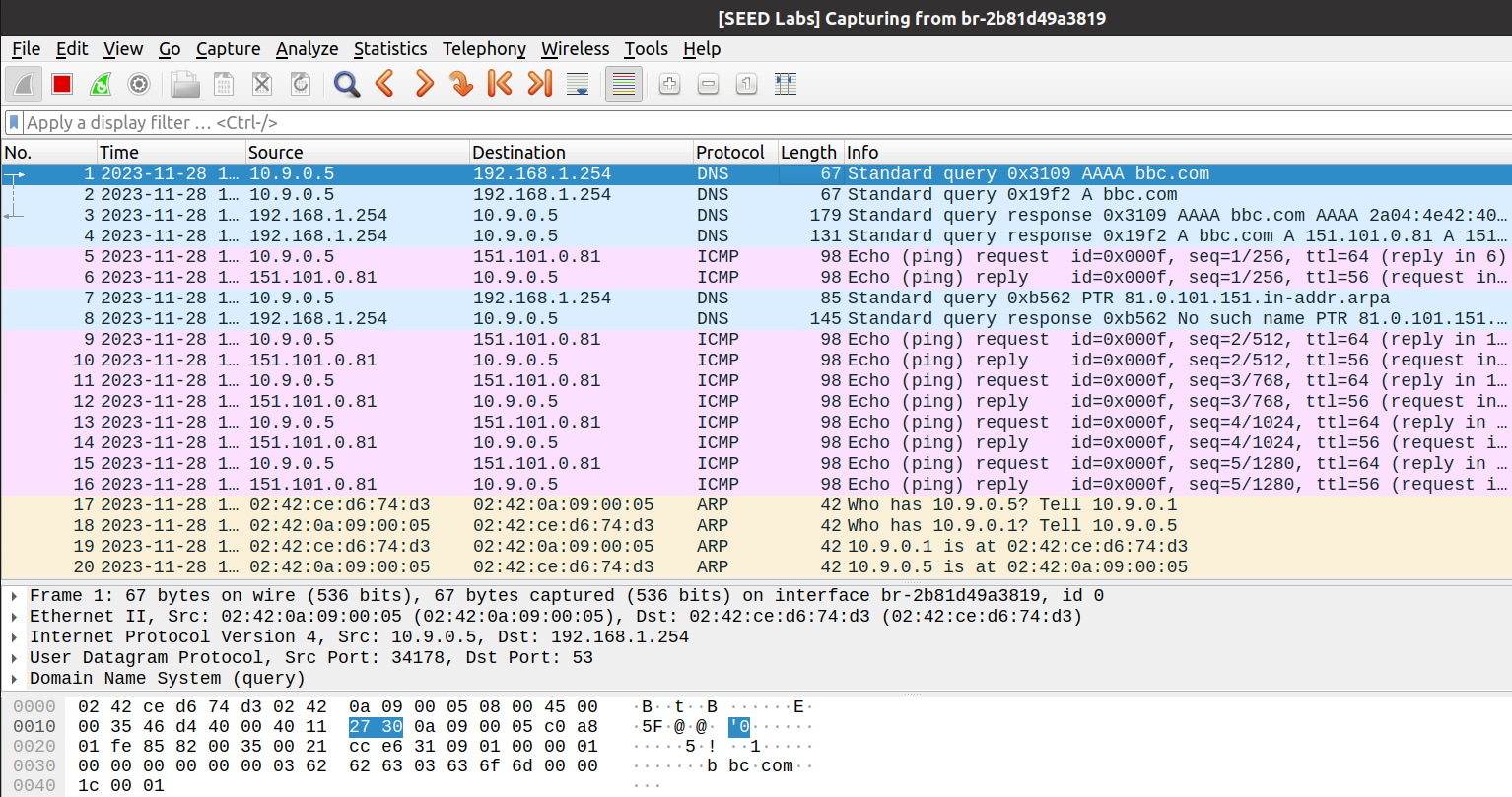
1. Task 1: Explain how you use Scapy, Wireshark and tcpdump to sniff packets.

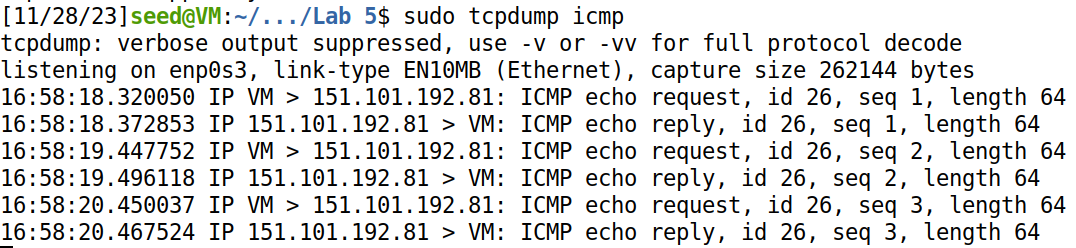


Scapy: set iface='br-2b81d49a3819' in sniffer.py and execute the code.

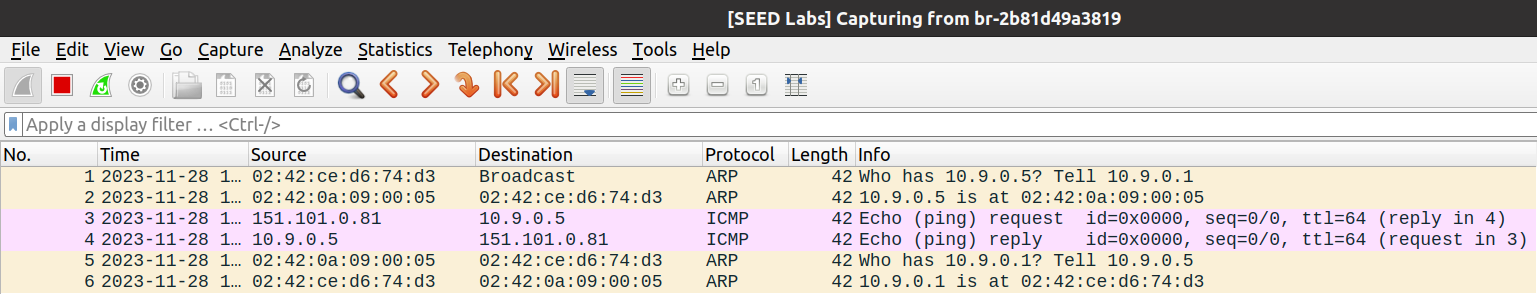
Wireshark: click 'br-2b81d49a3819', the pink entries are ICMP packets.

tcpdump: use 'icmp' in the command to filter. In all the cases we get pairs of icmp ping request and replies, the number of them consistent with how many packets victim machine sends and which ones are received.





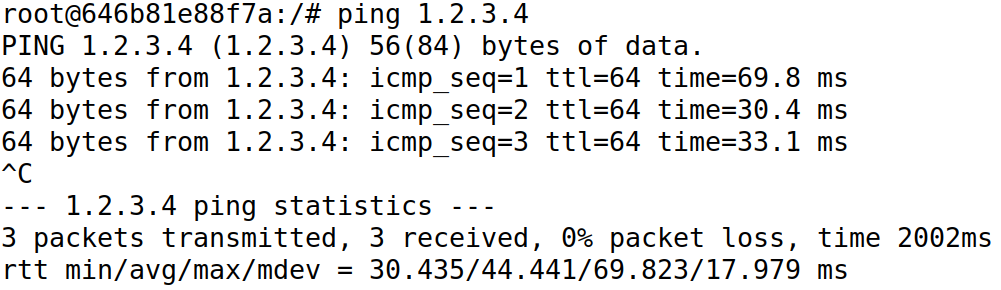
1. Task 2: Explain how you modify the provided code to spoof ICMP packets and the rationale for the modification. Present spoofing attack results with screenshots.①

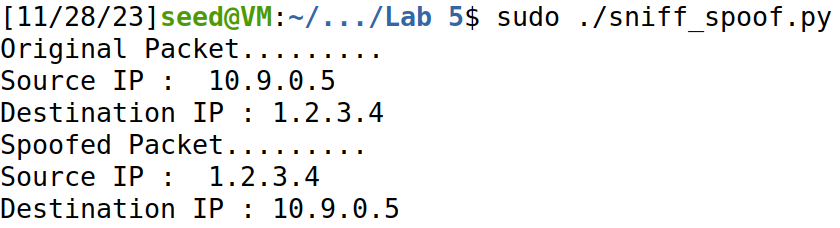


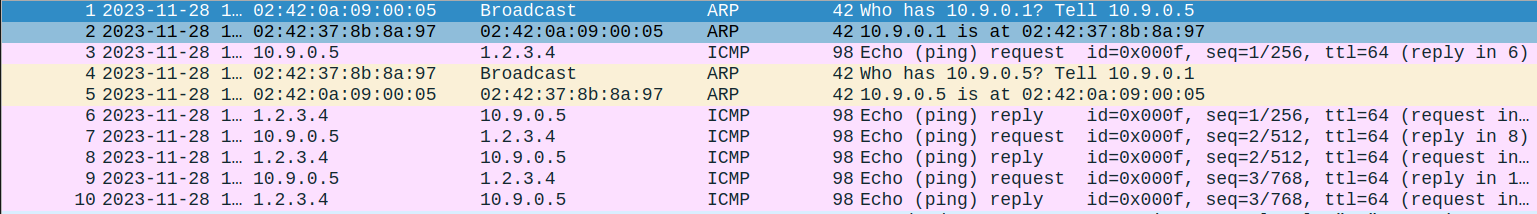
We add a.src = '151.101.0.81' to the provided code (between Line ① and Line ②), so a forged ping request from BBC.com is sent to the victim, the victim replied to it.

1. Task 3: Explain how you do the sniff-then-spoof attack in Task 3.

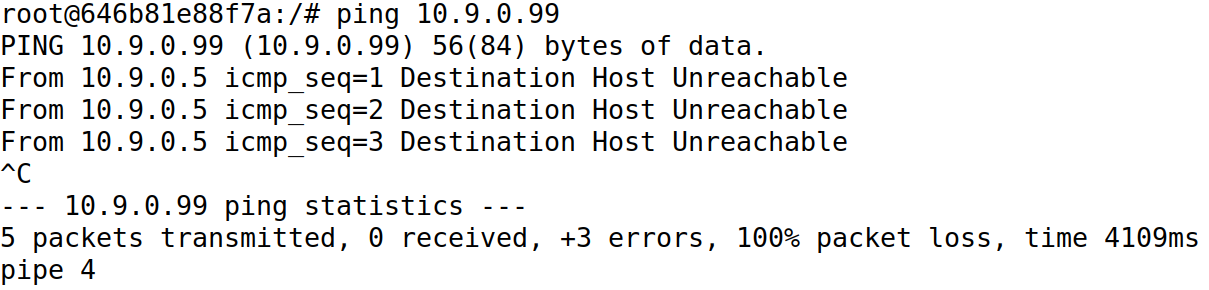
/\* During this phase my Virtual Machine shut down and I had to restart it so the network interface name of the victim machine is different from that in Task 1 & 2 \*/



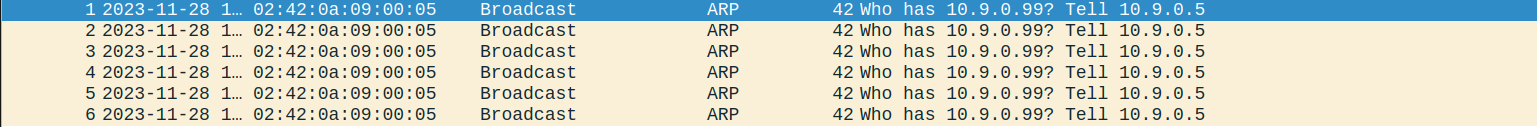




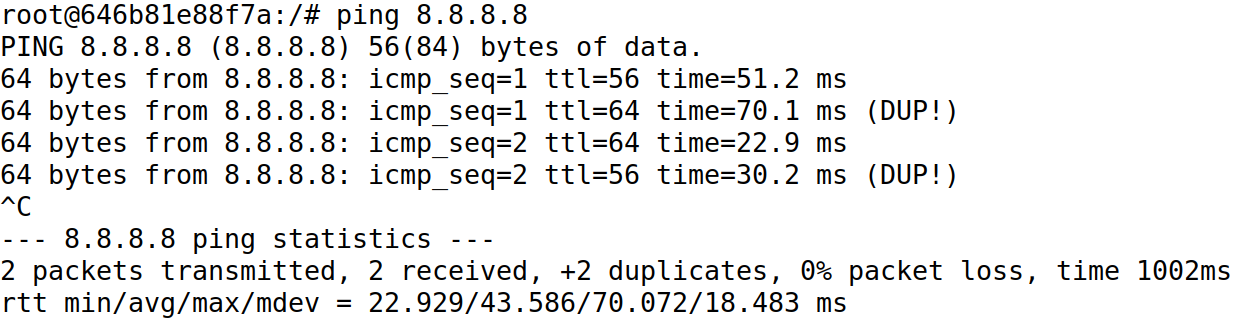
The sniff-and-then-spoof attack was successful, the victim was tricked to "believe" that it has received ICMP echo replies from the non-existing host 1.2.3.4.

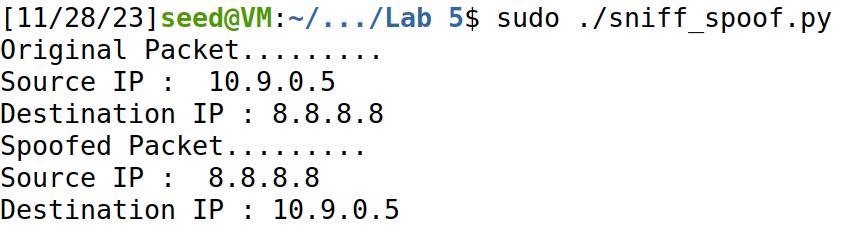


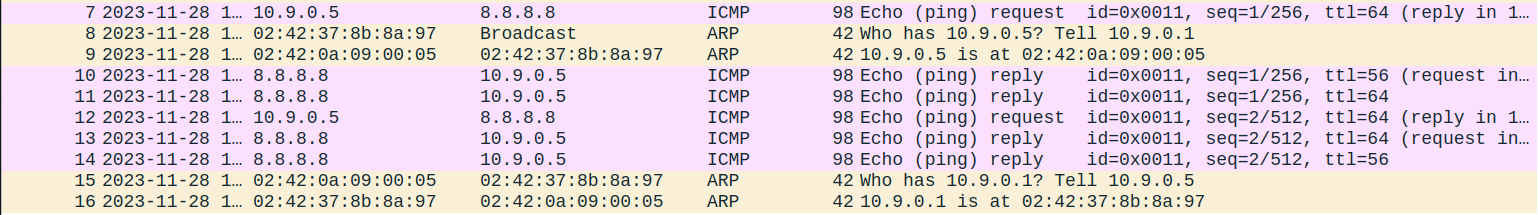




The sniff-and-then-spoof was unsuccessful, the victim kept sending ARP broadcasts but did not send any ICMP requests, the attacker did not sniff any echo request, and had no chance to forge a reply. The victim decides that destination host is unreachable.







The attacker did sniff the ICMP echo requests and sent back forged replies, but the real host also sent its replies, so the victim received duplicate replies (DUP!).

1. Answer the following two questions in Task 3.

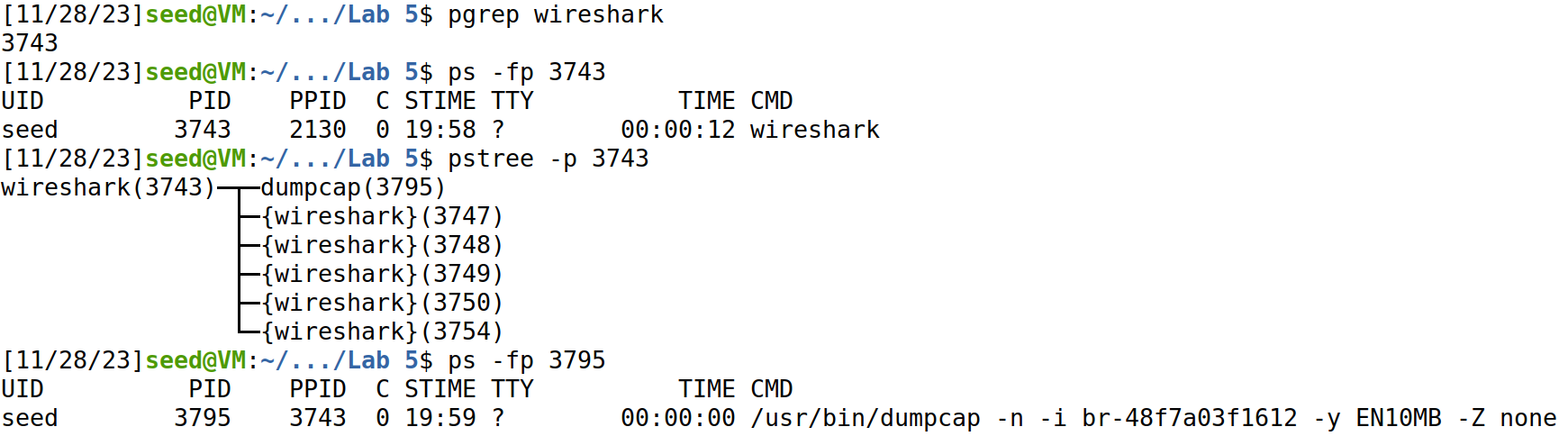
In the sniffing and then spoofing experiment in the lab, why can you get the echo reply from 1.2.3.4 which does not exist on the Internet?

Type 'ip route get 1.2.3.4' in the victim's terminal we see '1.2.3.4 via 10.9.0.1…', so the router our victim uses for 1.2.3.4 is actually the attacker! When the victim attempts to pin '1.2.3.4' from the Internet, an ICMP packet is sent using the attacker's MAC address as destination. (If this MAC is not in its ARP cache yet, it also sends its ARP broadcast to find out the MAC of the "router", as this "router" does exist, ICMP request will be sent eventually). The attacker sniffs the packet and spoofs a forged echo reply.

In the above experiment, why can’t you get the echo reply from 10.9.0.99?

Type 'ip route get 10.9.0.99' in the victim's terminal we see '10.9.0.99…', that is to say as 10.9.0.99 and our victim are (supposed to be) in the same LAN, the victim does not use a router to forward packets to this destination. When it attempts to ping the IP, the ARP broadcasts are sent to find 10.9.0.99's MAC address, which it will never know for the host simply does not exist. As the victim is always broadcasting and waiting for the ARP reply, an ICMP request was never sent; we cannot reply to an "unsent" request.

An investigation to explain why Wireshark is still able to sniff packets:

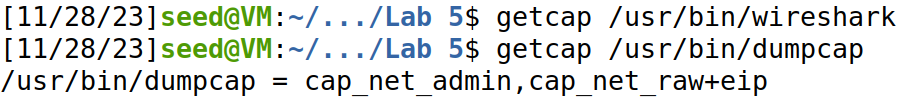


First, we use "pgrep" to find the process ID of the running program Wireshark.

The UID (effective user ID) of the Wireshark process is indeed "seed".

Using "pstree -p", we see a child process dumpcap is launced by Wireshark.

But the UID of the dumpcap process is also "seed"!



Does Wireshark have any special Linux capabilities? No.

What about dumpcap? Yes, it has CAP\_NET\_ADMIN and CAP\_NET\_RAW, etc.

Judging from the name we guess that CAP\_NET\_RAW might be the capability we are more interested in – we want to know why the sniffer is able to use a raw socket without the root privilege. See Linux manual for capabilities, under the CAP\_NET\_RAW entry there is this line "Use RAW and PACKET sockets", just what we are looking for.

Now we see having root privilege is sufficient, not necessary, for a program to be able to use a raw socket (and to sniff packets). Here the UID of dumpcap is still not root, but the program is granted the capability to use raw sockets separately. Note that even this capability is given only to the capture utility, dumpcap, which Wireshark calls as a child process, not the entire program. This corresponds with the principle of least privilege.