1. The recorded MAC addresses.

```
tap0      Link encap:Ethernet  HWaddr f2:e7:a6:f3:85:46
          inet addr:10.0.0.3  Bcast:10.0.0.255  Mask:255.255.255.0
```

```
ruiyang@nslabu:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
knoxville.nslab          ether   08:00:27:7b:11:4c   C                     tap0
paiute.nslab                     (incomplete)                              tap0
192.168.254.2            ether   52:54:00:12:35:02   C                     eth0
chicago.nslab            ether   00:50:56:9f:5a:33   C                     tap0
ruiyang@nslabu:~$ arp -a
knoxville.nslab (10.0.0.113) at 08:00:27:7b:11:4c [ether] on tap0
paiute.nslab (10.0.0.254) at 00:50:56:9f:3d:d9 [ether] on tap0
? (192.168.254.2) at 52:54:00:12:35:02 [ether] on eth0
chicago.nslab (10.0.0.124) at 00:50:56:9f:5a:33 [ether] on tap0
ruiyang@nslabu:~$
```

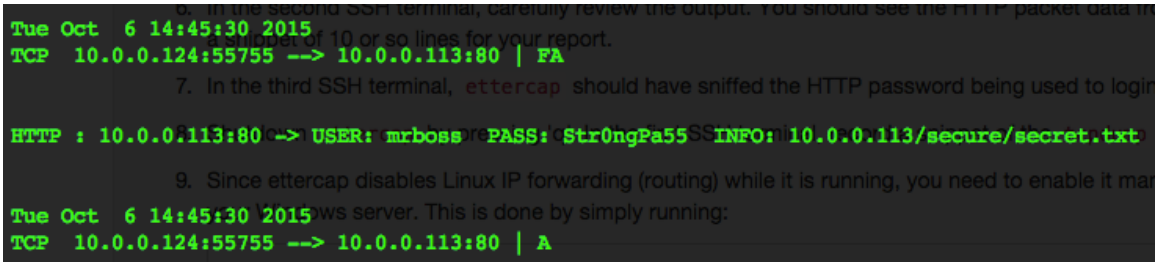
2. A snippet of the ARP data right after the ARP poisoning, showing at least one correct ARP reply and a few spoofed ARP replies.

```
ruiyang@nslabu:~$ sudo tcpdump -n -i tap0 arp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on tap0, link-type EN10MB (Ethernet), capture size 65535 bytes
14:27:47.731373 ARP, Request who-has 10.0.0.254 tell 10.0.0.3, length 28
14:27:48.732812 ARP, Request who-has 10.0.0.254 tell 10.0.0.3, length 28
14:27:49.731387 ARP, Request who-has 10.0.0.254 tell 10.0.0.3, length 28
14:29:30.740664 ARP, Request who-has 10.0.0.124 tell 10.0.0.113, length 46
14:33:30.772650 ARP, Request who-has 10.0.0.124 tell 10.0.0.113, length 46
14:37:30.804530 ARP, Request who-has 10.0.0.124 tell 10.0.0.113, length 46
14:37:44.142250 ARP, Request who-has 10.0.0.124 tell 10.0.0.3, length 28
14:37:44.148672 ARP, Reply 10.0.0.124 is-at 00:50:56:9f:5a:33, length 28
14:37:44.156274 ARP, Request who-has 10.0.0.113 tell 10.0.0.3, length 28
14:37:44.159841 ARP, Reply 10.0.0.113 is-at 08:00:27:7b:11:4c, length 46
14:37:45.194566 ARP, Reply 10.0.0.113 is-at f2:e7:a6:f3:85:46, length 28
14:37:46.206442 ARP, Reply 10.0.0.113 is-at f2:e7:a6:f3:85:46, length 28
14:37:47.218388 ARP, Reply 10.0.0.113 is-at f2:e7:a6:f3:85:46, length 28
14:37:48.229195 ARP, Reply 10.0.0.113 is-at f2:e7:a6:f3:85:46, length 28
14:37:49.240358 ARP, Reply 10.0.0.113 is-at f2:e7:a6:f3:85:46, length 28
```

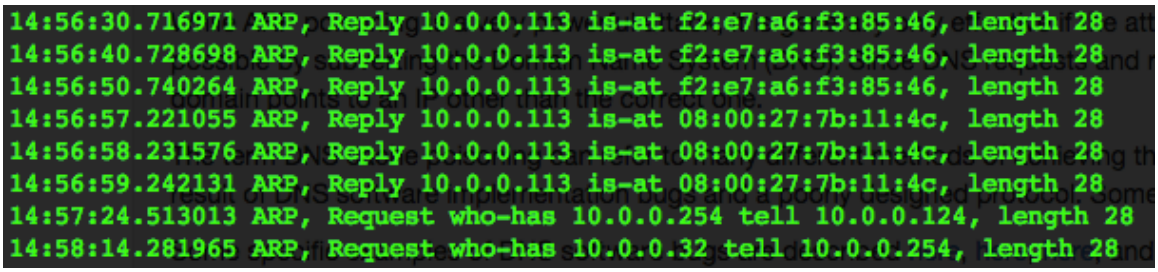
3. A snippet of 10 or so lines of HTTP data between `LIN` and `WIN`.

```
ruiyang@nslabu:~$ sudo tcpdump -n -i tap0 port 80 and host 10.0.0.124
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on tap0, link-type EN10MB (Ethernet), capture size 65535 bytes
14:41:30.733924 IP 10.0.0.124.55753 > 10.0.0.113.80: Flags [S], seq 1656722159, win 5840, options [mss 1336,sackOK,TS val 716103554 ecr 0,nop,wscale 4], length 0
14:41:30.738444 IP 10.0.0.124.55753 > 10.0.0.113.80: Flags [S], seq 1656722159, win 5840, options [mss 1336,sackOK,TS val 716103554 ecr 0,nop,wscale 4], length 0
14:41:30.742914 IP 10.0.0.124.55753 > 10.0.0.113.80: Flags [.], ack 3353125227, win 365, options [nop,nop,TS val 716103558 ecr 0], length 0
14:41:30.743089 IP 10.0.0.124.55753 > 10.0.0.113.80: Flags [.], ack 1, win 365, options [nop,nop,TS val 716103558 ecr 0], length 0
14:41:30.744754 IP 10.0.0.124.55753 > 10.0.0.113.80: Flags [P.], seq 0:125, ack 1, win 365, options [nop,nop,TS val 716103558 ecr 0], length 125
14:41:30.744928 IP 10.0.0.124.55753 > 10.0.0.113.80: Flags [P.], seq 0:125, ack 1, win 365, options [nop,nop,TS val 716103558 ecr 0], length 125
14:41:30.752342 IP 10.0.0.124.55753 > 10.0.0.113.80: Flags [.], ack 1325, win 546, options [nop,nop,TS val 716103560 ecr 28643107], length 0
14:41:30.752371 IP 10.0.0.124.55753 > 10.0.0.113.80: Flags [.], ack 1862, win 712, options [nop,nop,TS val 716103560 ecr 28643107], length 0
14:41:30.752627 IP 10.0.0.124.55753 > 10.0.0.113.80: Flags [.], ack 1325, win 546, options [nop,nop,TS val 716103560 ecr 28643107], length 0
14:41:30.752800 IP 10.0.0.124.55753 > 10.0.0.113.80: Flags [.], ack 1862, win 712, options [nop,nop,TS val 716103560 ecr 28643107], length 0
14:41:30.754715 IP 10.0.0.124.55753 > 10.0.0.113.80: Flags [P.], seq 125:297, ack 1862, win 712, options [nop,nop,TS val 716103561 ecr 28643107], length 172
14:41:30.754962 IP 10.0.0.124.55753 > 10.0.0.113.80: Flags [P.], seq 125:297, ack 1862, win 712, options [nop,nop,TS val 716103561 ecr 28643107], length 172
14:41:30.760678 IP 10.0.0.124.55753 > 10.0.0.113.80: Flags [F.], seq 297, ack 2201, win 877, options [nop,nop,TS val 716103563 ecr 28643107], length 0
14:41:30.761081 IP 10.0.0.124.55753 > 10.0.0.113.80: Flags [F.], seq 297, ack 2201, win 877, options [nop,nop,TS val 716103563 ecr 28643107], length 0
```

4. A password line from ettercap showing the HTTP username and password.



5. A snippet of the ARP data right after the ARP poisoner deactivated, showing the correct MAC address for the `LIN` machine.



6. **What is Mallory's probability of success in the DNS cache poisoning attack against Alice?**

Given that we already knew the source and destination IP info and port info, "the transaction ID is the sole form of authentication for a DNS reply" (DNS Cache Poisoning - The Next Generation). So what Mallory will do is actually trying to guess the correct transaction ID. We know that the length of transaction ID is 16 bits (hence 65535 possibilities). So the probability of a success attack is 1/65535.

7. **In the DNS cache poisoning section, you studied a specific flaw in some DNS software, which could allow an attacker to easily poison it. State which flaw you studied and describe the *specific* scenario(s) in which an attacker could exploit it**.

I studied Dan's Shenanigans and found out the main flaw in the early design of DNS is a predictable Query ID and Port Number. This flaw makes the DNS reply packets easy to be forged. Via a flood of forged packets within a proper time interval, the target DNS could be poisoned with the forged data and ignore the real ones.

8. **Suppose it was your task to design a simple heuristic to detect ARP poisoning attacks. What kinds of abnormalities could a passive sniffer look for that would be strongly indicative of this kind of MitM attack?**

One can check the ARP table, and look for hosts with different IP have the same MAC address, that is a strong indicator that the ARP table has been polluted.

9. **Submit a link to a tool that you can install on your Linux machine to detect ARP poisoning.**

http://www.xarp.net/

10. **In the `tcpdump` output of HTTP packets, why are only packets from `LIN` to `WIN` shown?**

Because of the "oneway" option, it forced ettercap to poison only from LIN to WIN. That means only communication from LIN to WIN has been captured and redirected to my host machine port 80. After all, those packets will be forward to the target WIN.