**CIS 551 – Homework 3, Phase 2**

**Team members**:

- Aditya Deshpande
- Aryaa Gautam
- Aniruddha Rajashekhar
- Di Wu

Modifications that will be required to the Homework2:

Since the specifications of Homework 2, did not say anything about it being used for homework 3, we had developed a system with a very strict access control. This control was put in to make sure that, even the wifi access is denied once the iptable rules are in play. Furthermore, only tcp connection is allowed. Since the ARDrone client used UDP packets for communication (as confirmed using wireshark), an unmodified homework2 will not allow the ARDrone app to control the drone.

In order to successfully demonstrate the attack, after consulting with the TAs, two modifications are necessary to the original submission:

- An iptable rule to allow wifi connection and all access to Dr. Evil's phone.

We are submitting a revised version of the server code to account for the above properties.

**Plan of Attack**:

We have decided to exploit the ARP vulnerability that exists in the network.

Tools used: Scapy library, ARDrone.java, tcprewrite, tcpreplay, wireshark.

The first task is to obtain the IP and the MAC address of Dr. Evil's device. We use wireshark to filter out the command packets (udp 5556). On analyzing the packets, we obtain the source ip address and the source MAC address of Dr. Evil's phone.

Next step is to take control away from Dr. Evil.

ARP poisoning using the Scapy library allows us to alter the ARP cache of the controlling device. We poison the ARP cache of the controlling device (Dr. Evil's phone) such that the IP "192.168.1.1" gets mapped to the attacker's MAC address. As a result, the attacker's laptop starts receiving the packets meant for the ARDrone. We have turned off IP forwarding. This way Dr. Evil has lost control of the drone.

Next objective, is to make the drone crossover the line. Using wireshark, we had recorded the packets that are sent to the drone when a user may want to turn the drone left or right, beforehand.

The “pitch left” and “pitch right” command that we send were captured using the below java commands respectively:

java ARDrone 192.168.1.1 AT*ANIM=1,0,1000 for pitch left

and, java ARDrone 192.168.1.1 AT*ANIM=1,1,1000 for pitch right

Due to the Iptable rules in place, the drone accepts packets only if the source ip and the source mac address of the packets match that of Dr. Evil’s phone. Therefore, before sending the maneuver command packets to the drone, we change the source IP address and the source MAC address to that of Dr. Evils phone using “tcprewrite”.

tcprewrite --fixcsum --infile=roll_left.pcap
--outfile=roll_left_capture4_otherdevice.pcap --srcipmap=0.0.0.0/0:192.168.1.x
--enet-smac=xx:xx:xx:xx:xx:xx

Since the packets that the drone now receives from the attacker’s laptop contains the authenticated MAC and the IP addresses, it responds to the commands by moving left or right depending upon the command sent.

One of the properties of the AR Drone is, it will not accept command packets if, the sequence number in them is smaller than something that the drone has already seen. However, “1” is a special sequence number which resets the drone’s internal counter. By making sure that the sequence number in all the maneuvering packets is set to “1”, we ensure that the drone responds to the commands.

Furthermore, care has to be taken that the checksum is valid. Hence when using the “tcprewrite”

command we use the “—fixcsum” flag to check the checksum.

**Possible Precautions:**

The ARP vulnerability exists because, the ARP cache in the Phone is dynamic and can be modified. Instead, having a static ARP cache will prevent this problem.

Furthermore, using the sequence number “1” to reset the Drone’s counter is a vulnerability. By not giving this capability, it will be difficult for the attacker to inject the packets with the correct sequence numbers in the input stream.