SkyShield: Game-Theoretic Defense for Autonomous Drones

Jackson McCullough

April 2025

1 Players

This game is going to be represented by 2 players
Player one is represented as ρ_1 which is the attacker and the deauthentication packets.
Player two is going to be represented as ρ_2 which is the defender(operator) and the drone

2 Strategies (Game 1)

Multiple strategies will take place for each player, represented by S, and will have a letter next to the strategy for later use:

2.1 S_{ρ_1} Strategies

-**No attack** : nothing sent from the attacker
- **Low-rate attack (A)(1k total)**: a low-rate attack will be more stealthy, and will most likely disconnect the drone, but may not be effective when it comes to a flooding attack, may not work at all in that instance
-**Mid-Rate Attack(B)**: At this moment, we will also be trying 5-6k packets to see if the optimal amount may be somewhere in here, and see how the drone reacts to this amount
- **high-rate attack (C) (10k total)**: this is the ideal rate for a flooding attack, ideally shooting for 10k+ total packets sent over some duration of time, aiming for a maximum number per second. With 10k+ packets, the time for the drone being inoperable is significantly lengthened, which in turn will cause greater problems the longer the drone is inoperable.

2.2 S_{ρ_2} Strategies

- **Do nothing (W)**: Possible indefinite disconnect of the device, full reboot of the drones system, prolonged delay where the drone cannot be moved

- **Firewall proxy (X)**: This would be to setup an adapter as a proxy firewall for the connection of the Tello drone and the connected station. Doing this will allow for packet filtering and determining which packets will go through, acting as a middle man to the AP(Tello drone) and station (phone), essentially giving it to the firewall proxy and determining whether it should drop/modify the packet or send it to the AP.
-**Specified Packet Rejection Via Proxy (Y)**: In this scenario, we would use the firewall proxy created as before, but will specify certain packets to reject. In this case, deauthentication packets (subtype hex: 0x0c, binary: 1100)

3 Quantification, Payoffs, & Game Creation

This part is going to go over what is desired to be measured and quantified, and how games can be created based on what is collected.

We are wanting to demonstrate the amount of packets that are sent by anything of S_{ρ_1} in each attack.

3.1 S_{ρ_1} Logic

Ideally, the attacker is going to want a high-volume attack. It can be much more noticeable, but the ideal packet sum for a flood is very high, high enough to where ρ_1 can successfully packet flood the client before action is taken. This will take more power consumption, but have a greater solution. The payoff may be equivalent to that of a low-rate attack, depending on the calculated payoff.

3.2 S_{ρ_2} Logic

S_{ρ_2} is going to be based solely on the effective packets taken in by the client and AP. We would use different tools to measure and capture the acceptance rate of all packets sent. This would be based on a string of calculations. Declaring variables first as: P_a = Packets accepted; P_d = packets declined; P_e = packets effective; and P_s = packets sent

calculations would be performed as follows:
$P_d = P_s - P_a$
$P_e = (P_s)(PDR)$ where PDR = Packet Delivery Ratio
$PDR = P_a/P_s$
and P_e will give the effectiveness of the attack P_s will be predetermined by ρ_1 and collected by whatever tool ρ_2 is using to analyze packets

3.3 Payoffs

Once all calculations are then performed after trials, the next step will then be calculating and creating a payoff matrix, and creating games to the standard of the research. As of now, the game is being simulated (ρ_1 is [A,B,C]. as to where ρ_2 is [W,X,Y]) as shown:

Strategies	W	X	Y	Max ρ_2
A				
B				
C				
Max ρ_1				*Solution*

In this game, there is seemingly no zero-sum game.
$\therefore$ There is no reason to calculate any minimax or maximin.

4 Simulation(Do Nothing vs Packets)

The introductory step would be to create and gather everything needed to simulate each game, starting with everything the attacker needs

4.1 Attacker(1000 packets)

The list of tools that the attacker is using will be **Aircrack-ng, Scapy, and a Wifi Adapter**. Tools for observation and data analysis will be **Scapy(with python) and Wireshark** to sniff out the packets.

First, preparation takes place for the attacker. The attacker will have to configure the WiFi adapter correctly to be able to send 0x0c packets and monitor all activity on one specific channel, whichever the drone is on. This is done by using the command `airodump-ng wlan0` to pull surrounding WiFi networks and their MAC addresses, channels, and other information. From there, the adapters channel is changed by using the command `iwconfig wlan0 channel x` (x being whatever channel the drone is on).
The deauth attack will be launched using a program made with python and the scapy library, the iteration for the packets is:

```
deauth_attack(target, gateway, count=1000, delay=0.001)
```

There will be different iterations and sums for each strategy. For the Total packets of 1000 with some margin for more or less. We have a .001 delay for each packet, but the iteration may be capped, all is really based on the computing power. We will notate this as:

$$\left\{ (x_1, x_2, \ldots, x_n) : \forall j \; x_j \in [1, 39] \text{ and } \sum_{j=1}^{n} x_j = [1002, 1031] \right\}$$

This shows that we have some, x_j, that are in between [1,39] each iteration, until it reaches the total sum after some amount of seconds (n) that we found is in the range of [1002,1031], the time of disconnection was also in the range of t(seconds) $\in [23.2, 59.2]$
This gives averages (over 10 simulations) of:

- Total packets: 1012.9
- Maximum packets/second: 31.3
- Minimum packets/second: 7.2
- Reconnection time(seconds): 42.39

Packet filtering was applied in wireshark to find deauthentication, connection, and reconnection, using the filter:
`wlan.fc.type_subtype == 0x0c || wlan.fc.type_subtype == 0x01`

This will filter deauthentication packets, and association responses between the AP and Station (drone and phone). This makes it possible to measure the time it took to reconnect, by taking the **Last Association Response(AR) packet minus the first Deauthentication(deauth) packet**. Therefore,
last AR - first deauth = time disconnected
After ten simulations, this would conclude S_{ρ_1}(A) strategy, a low-rate attack.

4.2 Attacker (5000 packets)

The same process and tools are being used that were used by the one thousand packet simulation. The only change made is the count, moving from 1000 to 5000. After ten simulations, the ranges fit with each simulation held to:

$$\left\{ (x_1, x_2, \ldots, x_n) : \forall j \; x_j \in [1, 70] \text{ and } \sum_{j=1}^{n} x_j = [4988, 5150] \right\}$$

As well as reconnection time, each time to reconnect to the drone completely held a range of: t(seconds) $\in [176.8, 248]$
Giving averages (per 10 simulations) of:

- Total packets: 5026.9
- Maximum packets/second: 36.5

- Minimum packets/second: 3.6
- Reconnection time(seconds): 207.34

4.3 Defender(Do Nothing)

For the first strategy of the defender, the defense would be to do nothing. The tools the defender would have is the tello drone, and the phone. The defender will fly the drone and attempt manual reconnection. Some observations during the simulations show that this is possible for a small period of time. There were instances where the phone could connect back to the drone just for second, that could allow the defender to click the "land button" but could not manually move the drone. This allows the defender to safely land the drone, avoiding damages, but this is not effective or possible 100% of the time, and still delays the drones flight time to wherever its destination is. After ten simulations, this would conclude $S_{\rho 2}$(W) strategy, do nothing. The same thing will be repeated for five thousand packets and ten thousand packets.