SkyShield: Game-Theoretic Defense for Autonomous Drones

Jackson McCullough

November 2, 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

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)(5k total)**: At this moment, we will also be trying 5k 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)**: 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 (X)**: This strategy is just taking in the attack and waiting it out. This could lead to possible indefinite disconnect of the device depending

on number of packets or time interval, possible full reboot of the drones system, prolonged delay where the drone cannot be moved, could be prolonged until the battery is low enough to land. This is the most cost effective route when it comes to cost, time, effort, but could be outweighed completely by the scale of attack
- **5 Meters Added(Y)**: Added 5 meters (approx. 16.4 feet) to the distance between the drone and the adapter. This will add problems with the antenna gain, and this specific adapter may not be strong enough to be a persistent threat with distance added between the drone and the adapter.
- **10 Meters Added (Y)**: Double the space and see the outcome of accepted packets. See how crucial distance plays with the adapter being used. This could allow us to interpret if there is some point between the drone and adapter where the adapter is useless, and where that point exactly is.

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 result. We will have a notation of P_t =Total Packets Sent
The attacker would want to know an ideal distance to attack, to be far enough away, but still be able to attack. In an open space, signal drop off can be represented by **Inverse Square Law** for free space path loss(FSPL), which is $1/d^2$, where d is the distance (meters) from the device (drone). If there were barriers and interferences, the attacker would need to use the FSPL added by the Obstruction loss which is the sum of **Wall Attenuation Factors(WAF)** in this case.
Therefore, finding the **Total Path Loss (TPL)** would be represented as:

$$TPL = \frac{1}{d^2} + \sum_{i=1}^{n} WAF_i(dB)$$

3.2 S_{ρ_2} Logic

S_{ρ_2} is going to be based on the amount of packets they take in and the disconnection time. It would be difficult to fully prevent a DoS attack on a phone-drone pair, so the important factor here is going to be the time it takes for the drone to get back up and running fully operational. We will notate **FRT** = Full Reconnection Time, and Time Between Connections as **TBC**.The difference between the two is that FRT will be the time from first deauthentication, until the time where the drone is fully operational again. TBC will represent any time that the drone is able to reconnect even for a short period of time for various uses, this will be seen in later strategies.
Reconnection time can be found by:
(Last AR - First Deauth = FRT), where AR=Association Response packet, and deauth is the deauthentication packet.
If desired, ρ_2 could use a logistic probability model as a defender and calculate the probability of a successful attack with:

$$p_\gamma(d, P_t) = \frac{P_t}{P_t + k_d d + k_0}$$

Where

- P_t = Total packets sent
- d = distance
- k_d = Distance Scaling Constant
- k_0 = Baseline Resistance

The attacker strength here would be P_t, where the defensive strength would be the solution of $k_d d + k_0$. Depending on the strength of the drone, k_d and k_0 will change. A higher distance constant means that distance plays a larger effect to your system, and a lower baseline resistance would mean that your drone is more vulnerable, having less defensive strength. The value of p_γ will change depending on scale values

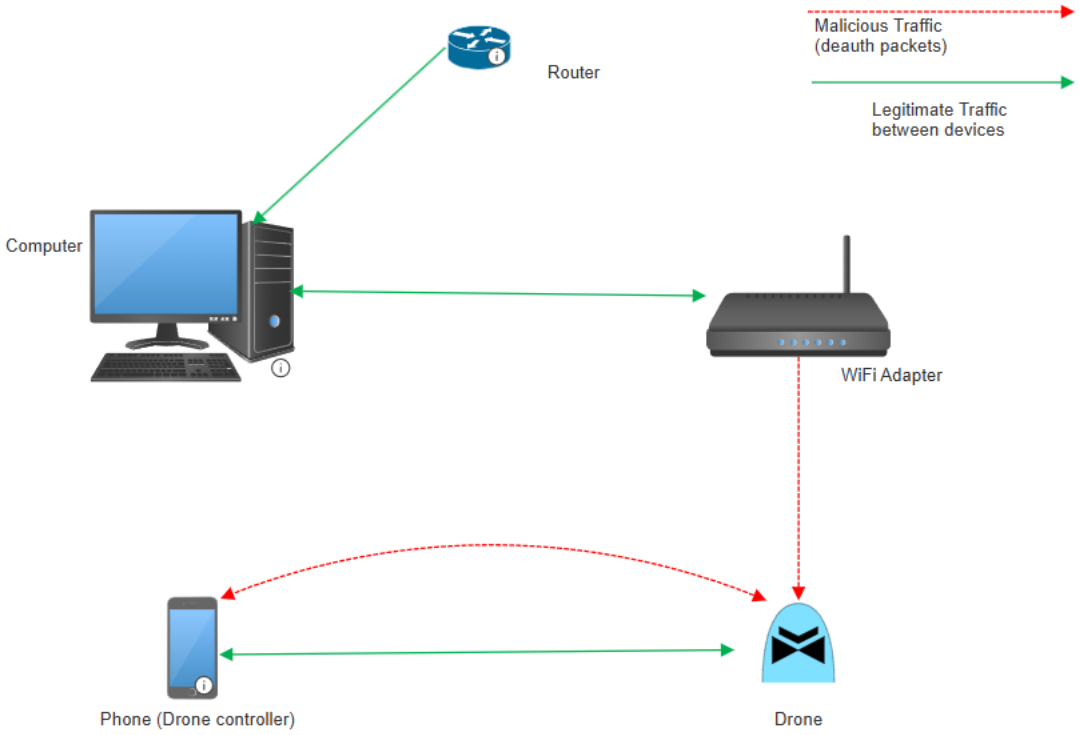
3.3 Payoffs

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*

4 Simulations

Now going through each strategy at a time and running ten simulations per strategy. For reference, this is what the traffic flow looks like during the simulations, just to know how each device is communicating with each other.



4.1 Attacker: 1k packets (Strategy(A))

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. Python was used to create a deauth script to launch the attack, and a count script that counts the number of deauthentication packets within a packet capture file.
The count script looks for **Management Frames** of type 0, then looks for hexadecimal subtypes of **0x0C**, which is the deauthentication packet. If the program sees this type/subtype, it counts that packet as a deauthentication packet and add its to a total to make P_t. This script also counts the number of packets per second (P_s) by:

```
for packet in packets:
    if packet.haslayer(Dot11Deauth):
        timestamp = int(packet.time)
        deauth_per_sec[timestamp] += 1
```

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. The adapter has to be configured to monitor mode, make sure that no other processes are interfering with the adapter, stop network tasks, then switch channels. 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. Script is launched by using the built in Pycharm terminal executing `sudo python3 launch_deauth.py`
The iteration for the packets is:
`deauth_attack(target, gateway, count=1000, delay=0.001)`
This means we are sending a total of 1000 packets every .001 seconds, or every millisecond(could be limited on computing powers, so .001 seconds may not be reached every time).
After the ten simulations of this, we have accumulated a minimum and maximum P_s, and a range of P_t being different, but close and within margin for the 10 simulations given by:

$$\left\{ (p_1, p_2, \ldots, p_n) : \forall s \ P_s \in [1, 39] \text{ and } \sum_{s=1}^{n} P_s = [1002, 1031] = P_t \right\}$$

This shows that we have some P_s, 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], FRT was also in the range of $n(\text{seconds}) \in [23.2, 59.2]$
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 doing the arithmetic as stated before, to get the FRT. After ten simulations, this would conclude $S_{\rho_1}(\text{A})$ strategy, a low-rate attack vs $S_{\rho_2}(\text{X})$ strategy, Do Nothing .

4.2 Attacker: 5k packets (Strategy(B))

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. The attack function now being used would be:
`deauth_attack(target, gateway, count=5000, delay=0.001)`
Same time iteration being used, same MAC addresses, just different P_t.

After ten simulations of this, the data gathered is represented by,

$$\left\{(p_1, p_2, \ldots, p_n) : \forall s \ P_s \in [1, 70] \text{ and } \sum_{s=1}^{n} P_s = [4988, 5150]\right\}$$

As well as reconnection time, FRT held a range of: $n(\text{seconds}) \in [176.8, 248]$
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 Attacker: 10k packets (Strategy(C))

Repeating for the final round of ρ_1 strategies A, B, C, versus the ρ_2 strategy X. Now, we have the count to be 10000.

After all simulations were ran for $S_{\rho_1}(\mathrm{C})$, we have:

$$\left\{(p_1, p_2, \ldots, p_n) : \forall s \ P_s \in [7, 36] \text{ and } \sum_{s=1}^{n} P_s = [10016, 10098]\right\}$$

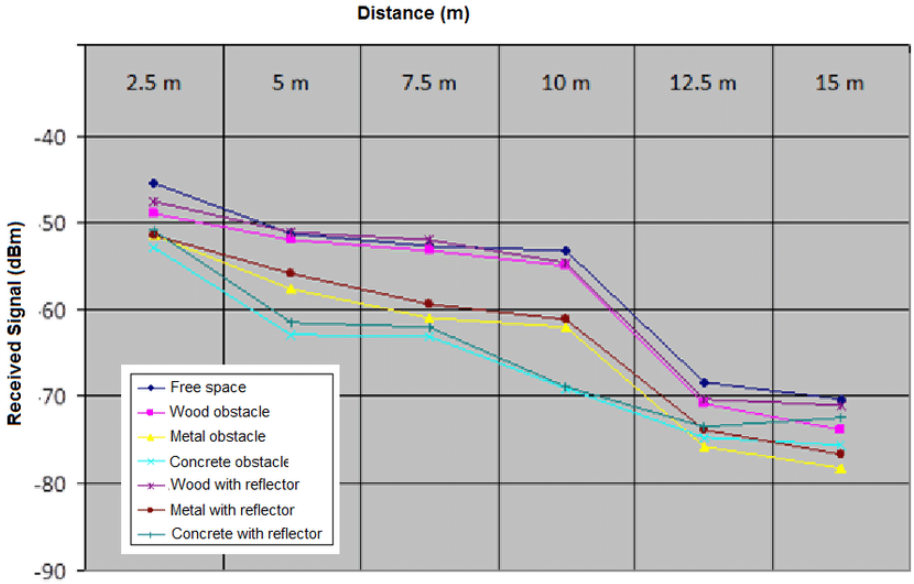
FRT: $n \in [430.9, 447.6]$
See that the 10k packets were closer to the target of 10k, may have something to do with the speed of packets being sent. Shorter sum may cause packet burst.

4.4 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 periodically attempt manual reconnection to see if the attack has passed. In a scenario such as 1000 packets, this may not be such a bad strategy, especially since they do not last very long. If the drone is not sensitive to time, waiting for the attack to pass may be the best option when it comes to resource consumption from putting in preventative or detective security controls. After ten simulations each against $S_{\rho_1}(\mathrm{A,B,C})$, this would conclude $S_{\rho_2}(\mathrm{X})$ strategy, do nothing.

4.5 5 Meter Distance (Strategy Y)

Adapters weaknesses lie greatly in the distance & obstacles between the adapter and the drone. This chart shows the drop off of strength between certain materials:



Adding this distance, in an enclosed area with walls to go through, will lessen the strength of waves sent to the drone.