AESS Challenge Simulation and Data Collection

1. Simulation Details

We used the Ardupilot Software in the Loop (SITL) simulator for simulation and data collection. The installation of the SITL simulator can be found at https://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html. We simulated Copter (See more details at https://ardupilot.org/copter/.)

The following parameters have been modified for simulating the attack scenarios of interest.

For GPS parameters, we set GPS_AUTO_SWITCH = 2 (2:Blend); GPS_BLEND_MASK = 7(0:Horz Pos, 1:Vert Pos, 2:Speed); GPS2_TYPE = 14 (14:MAVLink). We simulated wind speed SIM_WIND_SPEED = 0, 6, or 10 (m/s). All the other parameters were set as default. The configuration file is config_waypoint_inject.param.

The map with the waypoints in a considered mission is shown in Figure 1. The mission file is "mission_2_wp_23.waypoints".

Figure 1: The mission plan with 24 waypoints (including the home position).

The procedures of one simulation are as follows: 0) Use sim_vehicle.py -v ArduCopter --console --map to start the simulator; 1) (Optional) Start the Wireshark (https://www.wireshark.org/) software to capture packets of the Copter-GCS (Ground Control Station) network traffic; 2) Load mission (waypoints) from the file mission_2_wp_23.waypoints; 3) Use MAVProxy commands to start the simulation: a) arm throttle; b) mode guided; c) takeoff 10; d) mode auto; e) mode RTL (Return To Launch after the copter reached the last waypoint); 4) Exit the simulator and stop capturing packets (if used) in the Wireshark.

The Dataflash and Telemetry logs will be saved automatically. We can use the software Mission Planner to replay logging, review a log, and export visible csv files for the data of interest (e.g., GPS and IMU). Please refer to https://ardupilot.org/copter/docs/common-downloading-and-analyzing-data-logs-in-mission-planner.html for more details.

The columns of the GPS data include Message ID, Timestamp, Message Type, TimeUS, I, Status, GMS, GWk, NStats, HDop, Lat, Lng, Alt, Spd, GCrs, VZ, Yaw, U.
The columns of the IMU data include Message ID, Timestamp, Message Type, TimeUS, I, GyrX, GyrY, GyrZ, AccX, AccY, AccZ, EG, EA, T, GH, AH, GHz, AHz.

2. Attack Scenario

2.1. Waypoint Injection Attack

Attackers modify or insert new waypoints into the drone's mission plan without operator approval by sending a forged MISSION_ITEM MAVLink message to the flight controller. The message defines a new waypoint with specific latitude, longitude, altitude, and behavior parameters. After a successful waypoint injection, the drone will treat the injected waypoint as part of its mission, even if it was not originally programmed by the GCS. In our simulations, we sent the message via TCP port 5762. Please see the script pymavlink_send_waypoints.py for the implementation of the waypoint injection attack. We can run the script multiple times during the mission. Each run will add one new waypoint. The position of the new waypoint is generated based on the current and the next waypoints of the latest mission.

3. Data Processing for AESS Challenge

3.1. Network Data Extraction

3.1.1. Raw Packet Processing: We extract the data including timestamp, source port, destination port, message length, message ID, and protocol type using the script "packets_analysis_pyshark.py" for each captured packet. The data was saved in a .npy file with the shape of $N \times D$, where N is the number of packets, and $D = 6$ is the dimension of the extracted data.

3.1.2. Data Labeling: We use the filter tcp.port == 5762 to find the time periods when an attack was conducted, since the waypoint was injected via serial port 5762. Then, we label the data with the attack periods. If the timestamp of a packet falls in any attack periods, we will label the packet to be under attack. Similarly, we label the GPS data and IMU data using the attack periods.

The labeled network data, except those related to port 5762 and the labeled GPS and IMU data will be provided for the AESS challenge participants to detect attacks.

2.2. GPS Spoofing Attack

Besides the parameter changes in Section 1, we set EK3_CHECK_SCALE = 200 (default 100) to make it easier for the EKF to accept the injected GPS data. The configuration file is "config_gps_injection.param". In a GPS spoofing attack simulation, we run the gps_injection.py before Step 3) in the procedures of Section 1 to send spoofing GPS input. If successful, we can see GPS2 change from state 1 (0) to state OK6 (12) in the mavproxy console. The Copter will blend the GPS1 and GPS2 for positioning.

2.2.1. The blended GPS data (i.e., the positions used for navigation) the IMU data, the EKF data, and the GPS1 data will be provided to the participants for training. Among the data, GPS1 data are used as groundtruth, which is reliable and unattacked.

2.3. Data Preparation for Federated Learning

Please check the markdown file in the GitHub repository.