4-2024

DronLomaly: Runtime log-based anomaly detector for DJI drones

Wei MINN
Singapore Management University, wei.minn.2023@phdcs.smu.edu.sg

Naing Tun YAN
Singapore Management University, yannaingtun@smu.edu.sg

Lwin Khin SHAR
Singapore Management University, lkshar@smu.edu.sg

Lingxiao JIANG
Singapore Management University, lxjiang@smu.edu.sg

DronLomaly: Runtime Log-based Anomaly Detector for DJI Drones

Wei Minn
wei.minn.2023@phdcs.smu.edu.sg
Singapore Management University
Singapore

Yan Naing Tun
yannaingtun@smu.edu.sg
Singapore Management University
Singapore

Lwin Khin Shar
lkshar@smu.edu.sg
Singapore Management University
Singapore

Lingxiao Jiang
lxjiang@smu.edu.sg
Singapore Management University
Singapore

Abstract

We present an automated tool for realtime detection of anomalous behaviors while a DJI drone is executing a flight mission. The tool takes sensor data logged by drone at fixed time intervals and performs anomaly detection using a Bi-LSTM model. The model is trained on baseline flight logs from a successful mission physically or via a simulator. The tool has two modules — the first module is responsible for sending the log data to the remote controller station, and the second module is run as a service in the remote controller station powered by a Bi-LSTM model, which receives the log data and produces visual graphs showing the realtime flight anomaly statuses with respect to various sensor readings on a dashboard. We have successfully evaluated the tool on three datasets including industrial test scenarios. *DronLomaly* is released as an open-source tool on GitHub [10], and the demo video can be found at [17].

CCS Concepts: • **Security and privacy → Intrusion/anomaly detection and malware mitigation**; • **Computer systems organization → Embedded and cyber-physical systems**.

Keywords: Drone security, anomaly detection, log analysis, deep learning

ACM Reference Format:
Wei Minn, Yan Naing Tun, Lwin Khin Shar, and Lingxiao Jiang. 2024. DronLomaly: Runtime Log-based Anomaly Detector for DJI Drones. In *2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion '24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3639478.3640042

ICSE-Companion '24, April 14–20, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0502-1/24/04
https://doi.org/10.1145/3639478.3640042

1 Introduction

Drones have been increasingly used in missions such as military operations, surveillance, infrastructure inspection, package delivery, emergency response, etc. As more and more drones are flying in public airspace, safety concerns are also rising. While conducting a flight mission, several factors such as input validation vulnerabilities in the drone controller program [5, 6], harsh environmental conditions causing the drone to deviate significantly from the planned flight path [13], a remote cyber-attack that intentionally produces noises to confuse the sensors may affect the drone physical stability [16], etc. Hence, conducting flight missions in populated areas poses serious risks, as also highlighted by several incidents [2]. Therefore, it is important to monitor drone behaviors and detect anomalies while it is executing safety-critical missions so that appropriate measures, such as evasive maneuver or mission abort, can be taken in time.

Drones, especially industrial and military grade drones, usually record flight data at runtime. Given a baseline data, these runtime log data can be leveraged to detect anomalies during the drone flight. Therefore, in our previous work [14], we proposed an approach for runtime detection of anomalous drone Behaviors via log analysis and deep learning.

In this demonstration paper, we present *DronLomaly*, a tool that implements our runtime anomaly detection approach. The design and implementation of *DronLomaly* focuses on DJI drones. Alternatively, we could have targeted other drone platforms like Ardupilot and PX4. As these platforms differ significantly in terms of the SDK, the communication protocol, and the log format, the implementation has to be platform-specific. We chose to support DJI drones because DJI holds 76% of the global drone market according to Statista [15] and to our knowledge, none of the existing approaches provides such an anomaly detection tool for DJI drones.

DronLomaly takes as input a list of time series flight log data of a good drone which successfully executed a given flight mission repeatedly, which is used for learning a baseline model. Each log entry contains a timestamp, flight status

(e.g., flight mode), and state units (e.g., GPS and gyroscopic readings). During the runtime, *DronLomaly* produces a graph that shows the deviation of the predicted drone state with respect to the baseline model, from the current state. *DronLomaly* comprises of 2 components: 1) *Telemetry Subscriber* that listens a set of Telemetry topics (velocity, angular pose, etc.) measured by drone's sensors; 2) *Anomaly Detector* listens to the sensor readings retrieved by the Telemetry Subscriber and predicts the next sensor readings using the baseline model.

The source code of *DronLomaly* is available at GitHub [10].

2 Tool Design

DronLomaly is the tool supporting our approach described in our research paper [14]. It is designed and implemented as a service at the remote station. It can also be deployed in the drone itself by embedding the anomaly detector in a Raspberry Pi device. Figure 1 shows the overall architecture of *DronLomaly*.

The tool consists of two main components — (1) Telemetry Subscriber and (2) Anomaly Detector. During runtime, the telemetry subscriber fetches the telemetry data from the DJI drone simultaneously as the drone is conducting a mission. The telemetry data is a time series data that characterize the physical states of the drone. At any given time, the telemetry subscriber stores a sequence of log data corresponding to h+1 time units, where h is the configured time window. This log buffer is fed to a pre-trained bidirectional Long Short-Term Memory (Bi-LSTM) for anomaly detection . Given a sequence h of feature vectors corresponding to h time units, the model predicts the next possible feature vector at $h + 1$ time unit. More specifically, each feature vector x in h consists of flight status f and state units a such that

$$x = \{\text{flight mode}, \text{gain}, \text{velocity}_x, \text{velocity}_y, \text{velocity}_z\}. \quad (1)$$

Flight status refers to flight status information such as flight mode and gain configuration value; state units refers to sensor readings such as velocity at X-axis, Y-axis, and Z-axis. In short, we shall refer to this vector as $x = \{f, a\}$.

If the L1 (Manhattan) distance of the ground truth feature vector (representing the physical state of the drone) differs from that of the predicted feature vector by a threshold of 3 standard deviations, an anomaly is reported, together with visualized graphs showing the anomalous sensor readings.

3 Implementation

3.1 Telemetry Subscriber

DJI Onboard SDK's Telemetry module [3] provides various kinds of log data. We wrote a C/C++ script to collect the log data through this SDK during runtime. For the sake of our experiments, we collected GPS coordinates, velocity, acceleration, angular position and angular rate and ran this tool as a service at the remote station. Our telemetry subscriber tool

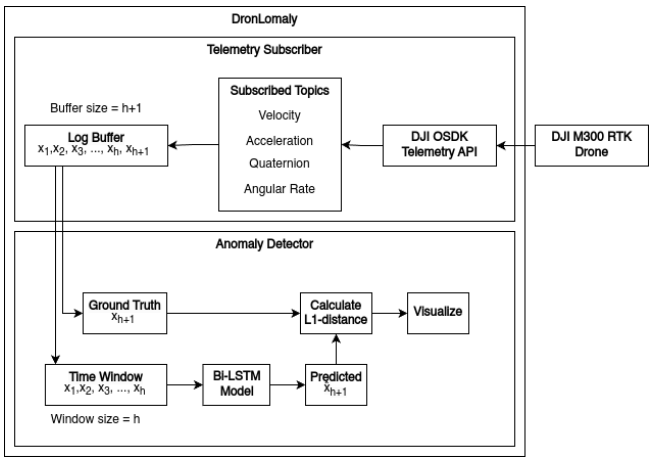

Figure 1. DronLomaly Design.

can also be run in an embedded device such as Raspberry Pi which could already be attached to a Physical DJI drone for autonomous operations. In our proof-of-concept, we installed the tool in Raspberry Pi 4 Model B with 8 GB memory and report its runtime performance results (Figure 5).

3.1.1 Baseline log extraction. For a given mission, we setup the drone to conduct the mission three times under different but normal operating conditions. We use the telemetry subscriber tool to collect flight logs, which essentially capture the baseline behavior of the drone for a given mission.

3.2 Anomaly Detector Model Training

We train a Bi-LSTM model on the baseline log data collected by telemetry subscriber. The following explains how the model is implemented.

3.2.1 Input. Let $\Omega = \{(f_1, a_1), (f_2, a_2), \ldots (f_n, a_n)\})$ be the whole set of feature vectors, extracted from a given log of a flight mission. The sequence for detection is a sliding window of the h most recent feature vectors. This is the input to our prediction model. As an example, Figure 2 shows the sequential patterns of certain flight states, where $h = 2$. As we can observe from Figure 2, our data is a multi-variate time series data where (bidirection) LSTM model can be applied. We normalize the values in each vector by the average and the standard deviation of all values from the same parameter position from the training data.

3.2.2 Model. The model has one bidirectional LSTM layer with a drop out of 0.1, one fully-connected linear layer with 128 neurons and 'ReLU' activation function, and one output layer whose size (# of neurons) is the number of drone sensor values to be predicted.

3.2.3 Objective function for training. In order to minimize the error between the predicted and observed feature vectors, we adjust the weights of the Bi-LSTM model using

Mean Squared Error (MSE) loss and 'Adam' optimization function. This process is done in an iterative manner, which is determined by the Epoch value. We set *Epoch* to 2000 with *Early Stopping* which is reached when the validation loss does not decrease below 0.001 continuously for 20 iterations.

Timestep	Sequence	Next
2	{ (ATTI, 1.5), (ATTI, 1.5) }	-1.5 if RTH
3	{ (ATTI, 1.5), (RTH, -1.5) }	-1.52 if RTH
4	{ (RTH, -1.5), (RTH, -1.52) }	-1.54 if RTH

Figure 2. Sample sequential patterns of flight log data. ATTI refers to 'Altitude Hold' flight mode. 'RTH' refers to 'Return to Home' mode, whereas the values represent sensor readings of velocity at Z-axis in m/s. Note that the actual feature vectors consists of more than one set of readings with respect to flight status and state units.

3.2.4 Output. The output is a real value vector as a prediction for the next state units, based on a sequence of feature vectors from recent history (Time Window).

We implement the above model using several Python libraries such as Numpy, Pandas, Scikit-learn, and PyTorch. Numpy, Pandas, and Scikit-learn are used for data processing and PyTorch is used for implementing the model. We leverage the yaml library for configurations such as specifying which sensor data to monitor, threshold for determining anomalies, etc. The trained model is saved as `model.pt` file.

3.3 Deployment

We deploy the model `model.pt` as a web service via Visual Studio Code IDE running in a remote controller machine. The web service provides REST API endpoint for Anomaly Detector, which accepts GET and POST requests from Telemetry Subscriber. The endpoint is built on XML-RPC module and is implemented in Python, encompassing the necessary code for API setup, route definition, and request handling and importing various Python modules for data processing, invocation of the Bi-LSTM model for the regression task, and visualization. Each request sent to the endpoint is expected to contain a flight record. The endpoint service processes the flight record to extract the relevant information (e.g., sensor data that is being monitored). It then invokes the Bi-LSTM model providing the extracted sensor values as an input, where the model generates prediction of the next possible state. It is also possible to run Anomaly Detector in the embedded system along with our Telemetry Subscriber. However, the web service approach allows Anomaly Detector to be decoupled from Telemetry Subscriber as seperate applications, so that the anomaly detector can be readily reused with different telemetry subscribers from other drone development platforms that may be implemented in different programming languages.

3.4 Visual Aid Generation

We implement the visualization tool for showing the deviation in terms of L1-distance between predicted and actual sensor values in realtime. We use the Python library Plotly to generate a live graph that simultaneously displays the multiple state values using multiple differently colored lines that are updating in realtime to visualize the realtime deviations. As an example, Figure 3 shows the visualization of predicted state deviation from the actual state. Y-axis value represent the percentage of threshold the current time window's deviation is at. The threshold is represented by the horizontal line that denotes y-axis value of 1. Crossing this value means that the deviation has exceeded 100% of the threshold.

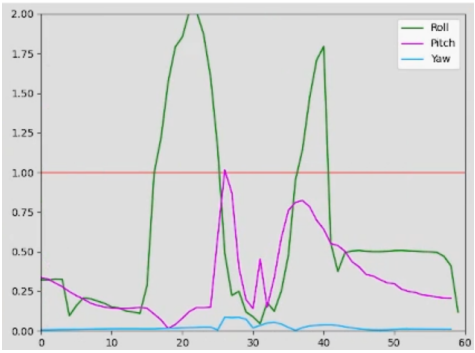

Figure 3. Visualization of deviation of predicted Roll, Pitch, Yaw sensor values from the actual ones for a duration of 60 seconds

4 Usage Scenario

As a proof of concept, we have demonstrated the following usage scenario of detecting anomalous behaviors caused by harsh weather, control program bugs, faulty configurations and safety/privacy regulation infringement to our industry partner:

Figure 4 shows the experiment setup we applied to demonstrate *DronLomaly* to our industry partner. Firstly, we collected DJI logs by flying a physical DJI drone, DJI Matrice 300. We used the Waypoint Mission module provided in DJI OSDK [1] to upload randomly generated waypoints into the drone. During the flight, we used the telemetry program to subscribe to the logs for realtime information on the drone's physical state. Next, we simulated an anomalous flight condition where there is an abnormally strong wind. We use DJI's Assistant 2 software for Hardware-in-the-loop simulation, and for mutation of environmental factors such as wind speed and direction. In Figure 3, *DronLomaly* 's visualization graph reports the anomalies with respect to the 'Roll' sensor readings at certain time periods. This anomaly is due to the drone having to produce a significantly higher Roll to compensate for the heavy wind.

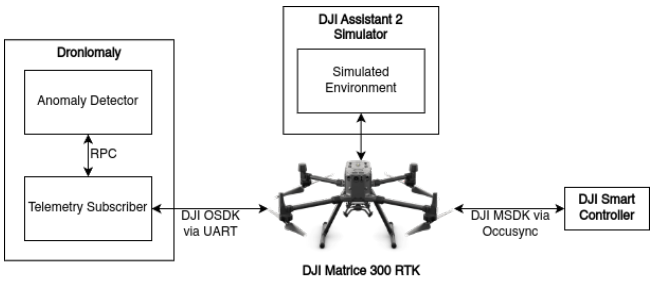

Figure 4. Experiment Setup

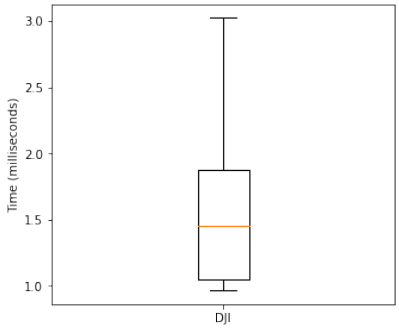

Figure 5. *DronLomaly*'s runtime performance on Raspberry Pi

In our research paper [14], we also evaluated two other scenarios (as required by our industry partner) — sensor failure and communication failure. Sensor failure may occur due to hardware fault or software configuration issue. For example, sensors need to be recalibrated after flying through extreme environmental conditions. Otherwise, it may lead to false sensor values. Communication failure between the remote controller and the drone can also occur due to long distance, encapsulated surroundings, or signal jamming, etc. We simulated such sensor failure and communication failure behaviors by injecting faulty sensor and GPS coordinate values into the DJI flight logs.

The recall, precision, and F-measure of *DronLomaly* for detecting anomalies in these three scenarios is reported in Table 1. On average, it achieves 0.986 recall, 0.948 precision, and 0.967 F-measure. The experiments were also run in a Raspberry Pi device for measuring its practical runtime performance. *DronLomaly* takes about 2 minutes to load a model, but this is a one time cost. Figure 5 shows the boxplot of the time taken for predicting the next state of the drone. The medium time taken is about 1.5 ms and the maximum is 3 ms. The total time taken may be longer due to the network latency because the alert signal needs to be sent to the drone operator. In our experience with flying DJI drones physically, the network latency is in the range of a few milliseconds to 2 seconds when operating within the visual line of sight. We believe that this should still give the operator ample time to take an appropriate action.

Table 1. Results

Anomaly	Recall	Precision	F-measure
DJI-windy	0.972	0.898	0.934
DJI-vel	0.986	0.948	0.967
DJI-com	1.0	0.999	0.999

5 Related Work

There are several anomaly detection approaches in the domain of Cyber Physical systems, for example, for detecting machine wear and predictive maintenance [4] and for detecting anomalies in traffic data [11]. However, to our knowledge, ours is the first work that focuses on runtime anomaly detection approach for DJI drones.

There are drone forensic approaches such as [8, 9, 12] that analyze flight log data to detect anomalies. But these approaches are offline analysis approaches that are typically used after an incident has been occurred. There are also fuzzing approaches for detecting bugs in drone control programs such as RVFuzz [7], PGFuzz [6], and LGDFuzzer [5]. They detect bugs such as input validation and semantic bugs. While fuzzing approaches are certainly useful, they cannot handle unstable physical states in drones when undetected bugs are activated at runtime. Therefore, our runtime anomaly detection approach complements these approaches.

6 Conclusion

This demonstration presents *DronLomaly*, a runtime anomaly detection tool for DJI drones. The tool has two main components — one for fetching realtime flight log data and the other for detecting anomalies reflected in the log. *DronLomaly* leverages DJI's onboard SDK to fetch realtime flight logs. It also leverages normal flight logs of successful drone missions to train a Bi-LSTM model. The model is then used to detect deviations from the learnt state values, as the new log entries are fetched at runtime. We demonstrated that this tool can be practically used in scenarios such as strong wind condition, sensor failure, and communication failure. In terms of accuracy, *DronLomaly* achieved 0.967 F-measure. In terms of efficiency, *DronLomaly*, when deployed on a Raspberry Pi device that can actually be used to control a drone, were able to generate a prediction within a few milliseconds. In future, we plan to extend this work by developing a mobile app leveraging DJI's mobile SDK so that *DronLomaly* can be deployed in the mobile phone, for more practical runtime monitoring purposes. We also plan to look into other kinds of scenarios, such as remote cyber-attacks and infringement of safety or privacy regulations, which were not considered in this work.

References

[1] Accessed 2022. DJI Onboard SDK (OSDK) 4.1.0. https://github.com/dji-sdk/Onboard-SDK.

[2] Accessed 2022. List of UAV-related incidents. https://en.wikipedia.org/wiki/List_of_UAV-related_incidents.

[3] Accessed 2023. Telemetry Topics. https://developer.dji.com/onboard-api-reference/group__telem.html.

[4] Nagdev Amruthnath and Tarun Gupta. 2018. A research study on unsupervised machine learning algorithms for early fault detection in predictive maintenance. In *2018 5th international conference on industrial engineering and applications (ICIEA)*. IEEE, 355–361.

[5] Ruidong Han, Chao Yang, Siqi Ma, JiangFeng Ma, Cong Sun, Juanru Li, and Elisa Bertino. 2022. Control Parameters Considered Harmful: Detecting Range Specification Bugs in Drone Configuration Modules via Learning-Guided Search. In *International Conference on Software Engineering (ICSE 22)*.

[6] Hyungsub Kim, Muslum Ozgur Ozmen, Antonio Bianchi, Z. Berkay Celik, and Dongyan Xu. 2021. PGFUZZ: Policy-Guided Fuzzing for Robotic Vehicles. In *Network and Distributed Systems Security (NDSS) Symposium*.

[7] Taegyu Kim, Chung Hwan Kim, Junghwan Rhee, Fan Fei, Zhan Tu, Gregory Walkup, Xiangyu Zhang, Xinyan Deng, and Dongyan Xu. 2019. RVFuzzer: Finding Input Validation Bugs in Robotic Vehicles through Control-Guided Testing. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, 425–442.

[8] Ravin Kumar and Animesh Kumar Agrawal. 2021. Drone GPS data analysis for flight path reconstruction: A study on DJI, Parrot & Yuneec make drones. *Forensic Science International: Digital Investigation* 38 (2021), 301182.

[9] Sri Harsha Mekala and Zubair Baig. 2019. Digital forensics for drone data–intelligent clustering using self organising maps. In *International Conference on Future Network Systems and Security*. Springer, 172–189.

[10] Wei Minn. 2023. DronLomaly: Runtime Detecting of Anomalous Drone Behaviors. https://github.com/weiminn/DronLomaly.

[11] Gerhard Münz, Sa Li, and Georg Carle. 2007. Traffic anomaly detection using k-means clustering. In *Gi/itg workshop mmbnet*, Vol. 7.

[12] Ankit LP S Renduchintala, Abdulsahib Albehadili, and Ahmad Y Javaid. 2017. Drone forensics: digital flight log examination framework for micro drones. In *2017 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 91–96.

[13] Stephen Shankland. 2016. Facebook drone investigation: Wind gust led to broken wing. https://www.cnet.com/tech/services-and-software/facebook-drone-investigation-wind-gust-led-to-broken-wing.

[14] Lwin Khin Shar, Wei Minn, Nguyen Binh Duong Ta, Jiani Fan, Lingxiao Jiang, and Daniel Lim Wai Kiat. 2022. DronLomaly: runtime detection of anomalous drone behaviors via log analysis and deep learning. In *2022 29th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 119–128.

[15] Daniel Slotta. June 2022. Leading global drone manufacturers 2021, by share of sales volume. https://www.statista.com/statistics/1254982/global-market-share-of-drone-manufacturers/.

[16] Yunmok Son, Hocheol Shin, Dongkwan Kim, Youngseok Park, Juhwan Noh, Kibum Choi, Jungwoo Choi, and Yongdae Kim. 2015. Rocking drones with intentional sound noise on gyroscopic sensors. In *24th USENIX Security Symposium*. 881–896.

[17] Yan Naing Tun. 2023. DronLomaly Demo. https://www.youtube.com/watch?v=LLHbhqEhLCA.

Received 23 October 2023; accepted 21 December 2023