UAVCAN Dataset Description

Korea University Graduate School of Information Security Hacking Response Technology Lab (Dongseong Kim, Yuchan Song, Sunhyeon Kwon, Haerin Kim, Jeongdo Yoo, Hwigang Kim)

October 2022

**Abstract**

"We collected attack data from unmanned vehicles using the UAVCAN protocol and prepared a technical document for public release. We established a testbed using drones with PX4 and performed three types of attacks: Flooding, Fuzzy, and Replay attacks. In total, there were 10 attack scenarios conducted. We anticipate that this attack data will be helpful in the development of technologies such as anomaly detection for addressing security threats in drones."

Expect.

Index terms—UAVCAN, dataset, cybersecurity

The study was funded by Institute for Information and communications Technology Promotion (Grant No. 2020-0-00374, Development of Security Primitives for Unmanned Vehicles)

Contents

1 Abbreviations and acronyms 3

2 Introduction 3

3 Background 5

3.1 UAVCAN. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

3.2 UAVCAN Frame. . . . . . . . . . . . . . . . . . . . . . . . . . . 5

3.2.1 UAVCAN Message Frame. . . . . . . . . . . . . . . . . . 6

3.2.2 UAVCAN Anonymous Message Frame. . . . . . . . . . . 6

3.2.3 UAVCAN Service Frame. . . . . . . . . . . . . . . . . . . 7

3.2.4 Single Frame. . . . . . . . . . . . . . . . . . . . . . . . . 8

3.2.5 Multi Frame. . . . . . . . . . . . . . . . . . . . . . . . . 8

3.3 UAVCAN Payload. . . . . . . . . . . . . . . . . . . . . . . . . . 8

3.3.1 CRC. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

3.3.2 Payload. . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

3.3.3 Tail bytes. . . . . . . . . . . . . . . . . . . . . . . . . . . 9

3.4 CAN. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

3.4.1 CAN protocol . . . . . . . . . . . . . . . . . . . . . . . . 10

4 Testbeds 10

4.1 Hardware. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

4.2 Pixhawk4 (PX4). . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

4.3 PX4 Supported ESCs. . . . . . . . . . . . . . . . . . . . . . . . . 11

4.4 UAVCAN Version. . . . . . . . . . . . . . . . . . . . . . . . . . 11

4.5 UAV internal network CAN BUS wiring structure. . . . . . . . . . . . . . 12

4.6 Experimental UAV Internal Network Environment . . . . . . . . . . . . . . . . . . . . 12

5 Methodology 13

5.1 Flood Attack. . . . . . . . . . . . . . . . . . . . . . . . . . . 14

5.2 Fuzzy Attack. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

5.3 Replay Attack. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

Community Verified icon

6 Attack Scenarios 17

6.1 Drone Attack Scenario 1 . . . . . . . . . . . . . . . . . . . . . . . . . 18

6.2 Drone attack scenario 2 . . . . . . . . . . . . . . . . . . . . . . . . . 19

6.3 Drone attack scenario 3 . . . . . . . . . . . . . . . . . . . . . . . . . 20

6.4 Drone Attack Scenario 4 . . . . . . . . . . . . . . . . . . . . . . . . . 20

6.5 Drone Attack Scenario 5 . . . . . . . . . . . . . . . . . . . . . . . . . 21

6.6 Drone Attack Scenario 6 . . . . . . . . . . . . . . . . . . . . . . . . . 22

6.7 Drone attack scenario 7 . . . . . . . . . . . . . . . . . . . . . . . . . 23

6.8 Drone Attack Scenarios 8 . . . . . . . . . . . . . . . . . . . . . . . . . 24

6.9 Drone attack scenarios 9 . . . . . . . . . . . . . . . . . . . . . . . . . 25

6.10 Drone Attack Scenarios 10 . . . . . . . . . . . . . . . . . . . . . . . . 26

2

7 Dataset Information (Metadata) 27

7.1 Dataset Metadata . . . . . . . . . . . . . . . . . . . . . . . . . 27

7.2 Dataset Structure . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28

1 Abbreviations and acronyms

This technical document uses the following abbreviations:

CRC Cyclic Redundancy Check

DDoS Distributed Denial of Service attack

DSDL Data Structure Decryption Language

ESC Electronic Stability Control

GCS Ground Control System

MAVLink Micro Air Vehicle Link

UAV Unmanned Aerial Vehicle

UAVCAN Uncomplicated Application-level Vehicular Computing And Network

**Introduction**

|  |
| --- |
|  |
| Figure 1: Security threats in unmanned vehicles [? ] |

"Unmanned Aerial Vehicles (UAVs) are vehicles that can operate stably and perform missions without human intervention. They are widely used in various fields such as driving, delivery, and reconnaissance. Ground Control Systems (GCS) support decision-making by exchanging data with UAVs, checking their status, and injecting missions. UAVs and GCS use various communication protocols such as MAVLink and UAVCAN. However, these protocols do not provide security metrics. Figure 1 illustrates the various attack scenarios that can occur from a network security perspective for UAVs. To ensure safe UAV operations and mission execution, it is essential to address various threats.

Technical methods to counter threats include developing systems that detect anomalies in abnormal states or implementing encryption modules. Jeong et al. proposed an intrusion detection system using deep learning for the MAVLink protocol, and Seo et al. proposed an anomaly detection system based on sequence similarity for the UAVCAN protocol. MAVLink is an external network protocol used for communication between UAVs and GCS, while UAVCAN is an internal network protocol used for actuators in UAVs. Kim implemented block ciphers like AES, ARIA, SEED, and LEA, and applied security modules to analyze threats for unmanned aircraft.

To address attacks on UAVs' confidentiality, integrity, availability, and authentication, it is essential to propose security systems and conduct experiments and analyses. Collecting dataset in a direct attack state during experiments can be costly, and in such cases, publicly available attack datasets can be very useful. Keipour et al. published datasets for various control fault scenarios for fixed-wing UAVs and expected their dataset to contribute to fault detection and anomaly detection research. Similarly, we collected and published a dataset for attack states in the UAVCAN protocol of the Pixhawk4 quadcopter. This technical document provides detailed descriptions of the dataset we made available.

Section 3 covers the basics of UAVCAN and CAN. Section 4 explains how we built the experimental environment for dataset creation. Section 5 describes the attack methods used for each type of attack in the dataset. In Section 6, we outline the attack scenarios designed for the dataset. Finally, Section 7 provides detailed information about the collected dataset, such as capacity and packet numbers."

|  |
| --- |
|  |
| Figure 2: Unmanned Vehicle Network Protocol |

**3 Background**

**3.1 UAVCAN**

"UAVCAN is designed as a lightweight protocol that provides high-reliability communication over the CAN Bus. It operates in the Application Layer of the CAN communication and has been developed for next-generation intelligent mobile platforms such as manned and unmanned aircraft, spacecraft, robots, and automobiles. UAVCAN uses the Data Structure Description Language (DSDL) as its design specification. DSDL defines data types used for communication and is embedded in the nodes (Electronic Speed Controllers - ESCs) firmware. Each node interprets the predefined data types, ensuring data integrity with CRC calculated from the data definition.

The key features of UAVCAN are as follows:

* Designed for real-time vehicle computing systems.
* Provides cost-free rich interface abstraction and service-oriented design.
* Lightweight.
* Peer-to-Peer Network: No bus master required.
* Modular redundancy.
* Supports various transport layer protocols.
* Open-source.

**3.2 UAVCAN Frame**

In UAVCAN, frames are classified into three types based on the structure of their ID, as shown in Figure 3:

• Message Frame

• Anonymous Message Frame

• Service Frame

Additionally, frames can be further categorized into two types based on the number of messages they contain:

• Single Frame

• Multi Frame"

|  |
| --- |
|  |
| Figure 3: UVACAN Message Frame |

|  |  |  |  |
| --- | --- | --- | --- |
| **Fields** | **Bits** | **Allowed Values** | **Description** |
| Priority | 5 | Any | 0-31, 0 highest priority 31 the lowest priority |
| Message type ID | 16 | Any | 1 of the encoded message |
| Service not message | 1 | 0 | Always 0 |
| Source node ID | 7 | 1-127 |  |

**3.2.1 UAVCAN Message Frame**

This is the most basic frame type in UAVCAN. It is used to command actions to Electronic Speed Controllers (ESCs), and upon receiving such a frame, the ESC executes the specified actions. The roles of each field in the ID are as follows:

* Priority: Determines the priority of the message.
* Message type ID: Determines the role of the message (e.g., motor voltage command, LED voltage command, etc.).
* Service not message: Indicates whether the message is a service frame or not. In the Message Frame, this field is always fixed at 0.
* Source node ID: Represents the ID of the node that transmitted this message.

**3.2.2 UAVCAN Anonymous Message Frame**

|  |
| --- |
|  |
|  |

Apart from the Message type ID, the functioning of the Message Frame and the Anonymous Message Frame is almost identical. The Anonymous Message Frame is used for messages from nodes that have not been assigned an ID, and such nodes are typically assigned dynamic IDs. UAVCAN does not allow different messages to have the same ID, which the Anonymous Message Frame violates since its transmission ID is always 0. To prevent this, messages are distinguished using Discriminator. Additionally, only a single message frame is allowed. The roles of each field in the ID are as follows:

* Priority: Determines the priority of the message.
* Discriminator: Serves the role of distinguishing the Anonymous Message Frame.
* Lower bits of message type ID: Determines the role of the message.
* Service not message: Indicates whether the message is a service frame or not. In the Anonymous Message Frame, this field is always fixed at 0.
* Source node ID: Represents the ID of the node that transmitted this message. In the Anonymous Message Frame, this field is always fixed at 0.

**3.2.3 UAVCAN Service Frame**

|  |
| --- |
|  |
|  |

This is a message that represents a specific service request or response between ESCs. If the value of the Request not response field is 0, it indicates a request message, and if it is 1, it represents a response message. Additionally, the Service not message field is always set to 0. The roles of each field in the ID are as follows:

• Priority: Determines the priority of the message.

• Service type ID: Specifies the service of the message.

• Request not response: Distinguishes whether the service frame is a request or a response. • Destination node ID: Represents the ID of the target node.

• Service not message: Indicates whether the message is a service frame or not. In the service frame, this field is always fixed at 1.

• Source node ID: Represents the ID of the node that transmitted this message.

**3.2.4 Single Frame**

**A screen shot of a computer

Description automatically generated**

This is a message consisting of a single frame. It does not have a CRC field, and the last byte is the Tail byte. The CRC and Tail byte will be explained in the UAVCAN Payload section.

**3.2.5 Multi Frame**

**A screenshot of a multi frame transfer

Description automatically generated**

This is a message consisting of multiple frames. The first 2 bytes of the first message are the CRC field, and the last byte of each message is the Tail byte. The CRC and Tail byte will be explained in the UAVCAN Payload section.

**3.3 UAVCAN Payload**

The Payload field of UAVCAN can include the following three components:

1. CRC: It verifies the integrity of the message.
2. Payload: It contains the direct meaning of the message, such as commands sent to the ESC.
3. Tail byte: It marks the start and end of the frame and represents one complete message.

**3.3.1 CRC**

The CRC is composed of the first 2 bytes of the first frame in a Multi Frame message. It ensures the integrity and validity of the message, checking if it conforms to the pre-defined DSDL format for the given Message type ID. The process of calculating the CRC is as follows:

1. Verify the Message type ID of the message.
2. Normalize the pre-defined data definition for the corresponding Message type ID.
3. Use the normalized value as the key and pass it to the signature function.
4. Pass the signature value and the Payload field of the message to the CRC function.
5. Use the result of the CRC function as the CRC.

**3.3.2 Payload**

**A diagram of a number

Description automatically generated**

The ESC directly manages and exchanges information with UAVCAN. The Payload is pre-defined for each UAVCAN ID, and by checking it, one can determine the meaning of the message. An example like Figure 8 shows a message that allows for checking ESC information. It provides details such as ESC voltage, current, motor rotation speed, temperature, and more.

**3.3.3 Tail byte**

"In Multi Frame, the beginning and end of each frame are identified to distinguish individual messages. In Single Frame, there is no explicit start and end; it exists as a single entity. Therefore, the Start of transfer and End of transfer fields are always fixed to 1, and the Toggle field is fixed to 0.

In Multi Frame, there are multiple frames with distinct beginnings and endings, including frames positioned in between. Hence, the Start of transfer field for the first frame in Multi Frame is set to 1, while for subsequent frames, it is set to 0. The End of transfer field for the last frame is set to 1, but for non-final frames, it is set to 0. The Toggle field alternates between 0 and 1 for intermediate frames to identify them as part of a single message.

Additionally, the Transfer ID field's value ranges from 0 to 31 in repetition, allowing the sequencing of the messages to be determined."

**A screenshot of a computer

Description automatically generated**

3.4 CAN

"3.4.1 CAN Protocol

In small unmanned aerial vehicles (UAVs), the UAVCAN protocol operates on top of the CAN protocol. CAN operates at the 1st and 2nd layers of the OSI model and possesses characteristics such as serial, multimaster, and multicast.

Specifically, when the bus is idle, any node can send a message, and all nodes can receive the transmitted messages. CAN messages can have a maximum length of 8 bytes. CAN was introduced in 1986, and in 1993, CAN 2.0A was released, followed by CAN 2.0B in 1995.

UAVCAN operates on CAN 2.0B, which differs from CAN 2.0A in its support for two identifier lengths: 11 bits and 29 bits. The payload size in CAN is limited to a maximum of 8 bytes."

4 Testbed

4.1 Hardware

The system architecture of the unmanned mobile platform used in this research is as follows:

• Autopilot system (Pixhawk 4) • Actuator, Motor, Transmission • Battery

Speaker, LED • Companion computer • Inertial Measurement Unit (IMU): Accelerometers, gyroscopes, magnetometers • GPS: GPS, GNSS, GLONASS, Galileo, BeiDou, QZSS • Wireless telemetry: Wi-Fi, Cellular, RF, Satellite communication • Wired external modules: RS232 serial, RAW, CAN (UAVCAN protocol)

**4.2 Pixhawk 4 (PX4)**

Pixhawk 4 (PX4) is developed as a subproject of ArduPilot, an open-source autopilot system for remote-controlled drones and autonomous aircraft. PX4 offers support for various protocols, including ISO 11898-2 CAN 2.0A/B.

4.3 PX4 Supported ESC (Electronic Speed Controllers)

In the context of PX4, the Holybro Kotleta20 ESC is utilized, which supports UAVCAN v0. However, as of now, there are no modules available that support UAVCAN v1 among the ESCs used in PX4.

**A screenshot of a computer

Description automatically generated**

UAVCAN was derived from CAN and has two versions: UAVCAN v0 and UAVCAN v1. Currently, UAVCAN v0 is being developed and maintained under the name "DroneCAN," while UAVCAN v1 is being developed and maintained as "Cyphal."

For this research, the target version is UAVCAN v0, specifically focusing on "DroneCAN."

|  |
| --- |
|  |
| Internal Network Wiring Structure." |

As Figure 11 shows, PX4 is connected to four ESCs (Electronic Speed Controllers) that control the motors via a serial CAN Bus. Each node in this setup receives power from a power board connected to an 11.1V battery.

At the termination point of the CAN Bus, there exists a "CAN terminator" represented by a 120Ω resistor. The CAN terminator serves to indicate the end of the CAN Bus and helps in preventing signal reflections and maintaining signal integrity.

Figure 12 represents the UAV (Unmanned Aerial Vehicle) environment used in the experiment. In this experiment, the traditional "CAN Terminator" in the UAVCAN's CAN BUS was replaced with a "CAN Shield." The CAN Shield module is connected to a Raspberry Pi 4, enabling the reading and analysis of CAN messages and allowing the transmission of desired messages. This setup was utilized to analyze both normal messages and attack messages in UAVCAN, creating attack scenarios as a part of the research.

The final experimental environment, as depicted in Figure 13, shows that the attacker's PC communicates with the target UAV by establishing an SSH connection with the Raspberry Pi connected to the UAV.

**A diagram of a computer

Description automatically generated**

**A computer and a drone

Description automatically generated**

**5 Methodology**

**A diagram of a drone

Description automatically generated**

We performed three types of injection attack methods on the UAVCAN protocol of the internal network of unmanned vehicles. The attacker injects malicious packets into the CAN Shield connected to the ESC (Electronic Speed Controller) as shown in Figure 14. In this case, the CAN ID of the injected packet is..."

The message frame with a Message type ID of 1030 and CAN ID 0x05040601 is the RawCommand packet. RawCommand is a packet used to control the Electronic Speed Controllers (ESC) connected to the motors in Pixhawk4. The payload of the RawCommand consists of six 14-bit integers, totaling 12 bytes of data. Including the 2-byte CRC and 2-byte Tail byte, the entire packet is composed of 15 bytes. Therefore, the attacker injects RawCommand packets into the drone to perform Flooding, Fuzzy, and Replay attacks.

"5.1 Flooding Attack

A Flooding attack is aimed at consuming the available server resources to the maximum extent, preventing legitimate users from accessing the service. It utilizes commands that halt the system, and it falls under the category of Distributed Denial of Service (DDoS) attacks. In a DDoS attack, a specific server or network device is targeted with a massive amount of data injection, depleting its resources and rendering it unable to function as intended.

When a Flooding attack is carried out on an unmanned vehicle, it can incapacitate the aircraft's systems, making stable flight and mission execution impossible."

**A screenshot of a computer program

Description automatically generated**

The Flooding attack we performed on the UAV is represented by Code 1. It involves setting all bits in the RawCommand message to 0, excluding the CRC and Tail byte, and transmitting it at regular intervals. The detailed explanation of the code is as follows:

* Lines 2 to 3 generate data with all bits set to 0.
* Lines 7 to 8 and 17 modify the Transfer ID of the Tail Byte to be an incremented value from the previous packet's Transfer ID.
* Lines 9 to 15 use the Python 'can' library to inject the packet into the CAN Shield.
* Line 16 transmits the packet and sets the time to wait before generating the next packet.
* Lines 18 to 19 contain code to stop packet transmission at a specific point, determined by the function's parameters.

**A screen shot of a computer

Description automatically generated**

The screen showing the execution of the Flooding attack is depicted in Figure 15. The left side of Figure 15 represents the portion where Code 1 was executed, and the right side shows the dumped results of UAVCAN's packet values during the execution of Code 1. When the attacker does not inject packets, only the normal packets sent by Pixhawk4 are displayed. However, when the attacker starts injecting packets, multiple packets with CAN ID 0x05040601 and filled with zeros are shown.

**"5.2 Fuzzy Attack**

A computer screen shot of a computer code

Description automatically generated

The Fuzzy attack is aimed at discovering potential vulnerabilities in software by injecting random values to induce abnormal behavior. In the context of unmanned vehicles, the Fuzzy attack on the UAVCAN protocol involves injecting random values into the CAN ID of the packets. While the other fields, excluding the CAN ID field, follow minimal rules for CRC calculation and Tail byte input, generating the data.

To ensure that the injected packets appear as anomalous but still get accepted by the unmanned vehicle, meaningful CAN ID values, collected beforehand, are used. The Fuzzy attack can lead to critical attacks on the drone, potentially causing it to stop functioning.

The Fuzzy attack performed on the UAV is depicted in Code 2. In this attack, the data, excluding CRC and Tail byte, is set to random values and transmitted at regular intervals. Unlike the Flooding attack, where the data is the same for every packet, the Fuzzy attack requires CRC calculation for each packet due to varying data. The 'sign' value in line 2 is used to verify whether the DSDL (Data Structure Description Language) between the sending and receiving nodes is the same.

Since the CAN ID is fixed at 0x05040601, the 'sign' value is also fixed. Lines 8 to 9 involve filling the data field with random values. Lines 4 and 10 contain code to calculate CRC using data and 'sign.' 'transCRC' and 'data\_gen' in lines 13 and 19 are functions written by the researcher to facilitate CRC calculation. Lines 22 to 23 are code to stop packet transmission at a specific point."

5.3 Replay Attack

A Replay attack is a type of attack where valid data transmission is maliciously repeated. By copying and resending protocol messages, attackers deceive the system into thinking that the data is from an authorized user, thus launching an attack. In the context of unmanned vehicles, the Replay attack involves collecting previously controlled values of the directional keys and then resending the intended or random direction values to execute the attack.

The Replay attack we performed on the UAV is depicted in Code 3. It involves transmitting stored frames one by one at predetermined times. The frames list contains the time to be sent at index 0 and the data to be transmitted at index 1. In line 6, the code checks the time at each moment and transmits frames that should have been sent before the current time. "time.time() - replayStart" represents the elapsed time since the attack started, and "frames[idx][0]-frames[0][0]" denotes the time between the next frame to be transmitted and the first frame.

Lines 7 to 11 use the Python 'can' library to inject packets into the CAN Shield. Lines 13 to 14 are code to end the transmission when all stored frames have been sent, and lines 15 to 16 are code to stop packet transmission at a specific point.

**A screen shot of a computer code

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

6 Attack Scenarios

In this section, we describe the expected effects when an actual drone is attacked and the method of collecting data through these attacks. Therefore, we have devised scenarios where the actual drone undergoes attacks, and we collect the generated data to build the UAVCAN Intrusion Dataset. The UAVCAN Intrusion Dataset is constructed by selecting various situations to predict possible occurrences in real drones. This dataset will be used in the development of IDS (Intrusion Detection Systems) and is expected to significantly contribute to UAVCAN protocol security. The drone attack scenarios are constructed using the attack methods described in Section 4: Flooding Attack, Fuzzy Attack, and Replay Attack. A total of 10 scenarios are composed, and they are structured to take place between 180 seconds and 270 seconds.

6.1 Drone Attack Scenario 1

Scenario 1 involves an attack while the drone is in the take-off state. The Flooding attack occurs when the drone is taking off, and a total of three attacks happen before take-off is completed. The specific attack method is as follows:

* 0s to 20s: The drone is powered on, and during the 20 seconds, booting and waiting for the basic functions to execute.
* 20s to 50s: Normal drone data occurs during this period as the drone takes off from 20 seconds to 50 seconds.
* 50s to 80s: The first Flooding attack starts at 50 seconds, where attack data is injected at intervals of 0.0015 seconds. This attack occurs for a total of 30 seconds, disrupting the drone's functions. During this time, the drone's flight behavior is disturbed, and the motors stop, maintaining the drone in a halted state until 80 seconds.
* 90s to 120s: After the initial attack, from 80 seconds to 90 seconds, normal drone data occurs, and then additional Flooding attacks occur from 90 seconds onward, repeating the same attack pattern. The attack lasts for 30 seconds, and attack data is injected at intervals of 0.0015 seconds. As a result, the motors stop until 120 seconds.
* 130s to 160s: Normal drone data occurs from 120 seconds to 130 seconds, and then additional Flooding attacks occur, repeating the same pattern as before. The attack lasts for 30 seconds, and attack data is injected at intervals of 0.0015 seconds. The motors stop until 160 seconds.
* 160s onward: After 160 seconds, no additional attacks occur, and normal drone data occurs. The drone starts the landing behavior at 170 seconds and generates data indicating a normal shutdown at 180 seconds.

6.1 Drone Attack Scenario 1

Scenario 1 involves attacking the drone during take-off. The Flooding attack occurs when the drone is taking off, and a total of three attacks happen before the take-off process is completed. The specific attack method is as follows, as shown in Figure 18:

• 0s to 20s: Initially, the drone is powered on, and for the first 20 seconds, it goes through the booting process, waiting for the basic functions to execute.

• 20s to 50s: During this period, from 20 seconds to 50 seconds, the drone is taking off, and normal drone data is generated. Therefore, all the data during this timeframe represents regular drone operation.

• 50s to 80s: Starting from 50 seconds, the first Flooding attack occurs. The injected attack data is generated at an interval of 0.0015 seconds. This attack lasts for a total of 30 seconds, disrupting the drone's functionality. During this time, the drone's flight behavior is disturbed, and the motors stop, keeping the drone in a halted state until 80 seconds.

• 90s to 120s: From 80 seconds to 90 seconds, normal drone data is generated as it resumes regular operation. However, after 90 seconds, another Flooding attack occurs, similar to the previous one, lasting for 30 seconds with 0.0015 seconds interval, causing the motors to stop again until 120 seconds.

• 130s to 160s: From 120 seconds to 130 seconds, normal drone data is generated again, but after 130 seconds, an additional Flooding attack occurs, similar to the previous ones, lasting for 30 seconds with 0.0015 seconds interval, causing the motors to stop until 160 seconds.

• 160s onwards: From 160 seconds, no further attacks occur, and normal drone data is generated. The drone starts its landing behavior at 170 seconds, and at 180 seconds, the drone's operation ends normally.

6.2 Drone Attack Scenario 2

Scenario 2 is also an attack during take-off, and it follows a similar pattern to Scenario 1 with a Flooding attack. However, in this scenario, the attack data is injected at the same frequency as regular data.

• 0s to 20s: The drone is powered on, and for the first 20 seconds, it goes through the booting process, waiting for the basic functions to execute.

• 20s to 50s: During this period, from 20 seconds to 50 seconds, the drone is taking off, and normal drone data is collected, representing regular operation.

• 50s to 80s: Starting from 50 seconds, the first Flooding attack occurs. In Scenario 2, the attack is slower than in Scenario 1, with an attack interval of 0.005 seconds. This attack lasts for 30 seconds, disrupting the drone's functionality, and the drone's flight behavior is disturbed, causing the motors to stop until 80 seconds.

• 90s to 120s: From 80 seconds to 90 seconds, normal drone data is generated as it resumes regular operation. However, after 90 seconds, another Flooding attack occurs, similar to the previous one, lasting for 30 seconds with a 0.005 seconds interval, causing the motors to stop until 120 seconds.

• 130s to 160s: From 120 seconds to 130 seconds, normal drone data is generated again, but after 130 seconds, an additional Flooding attack occurs, similar to the previous ones, lasting for 30 seconds with a 0.005 seconds interval, causing the motors to stop until 160 seconds.

• 160s onwards: From 160 seconds, no further attacks occur, and normal drone data is generated. The drone starts its landing behavior at 170 seconds, and at 180 seconds, the drone's operation ends normally.

**A diagram with red rectangles and black text

Description automatically generated**

6.3 Drone Attack Scenario 3

Scenario 3 involves attacking the drone during take-off with a Fuzzy attack. The Fuzzy attack occurs when the drone is taking off, and a total of three attacks happen before the take-off process is completed. The specific attack method is as follows, as shown in Figure 19:

• 0s to 20s: Initially, the drone is powered on, and for the first 20 seconds, it goes through the booting process, waiting for the basic functions to execute.

• 20s to 50s: During this period, from 20 seconds to 50 seconds, the drone is taking off, and normal drone data is generated. Therefore, all the data during this timeframe represents regular drone operation.

• 50s to 80s: Starting from 50 seconds, the first Fuzzy attack occurs. The injected attack data is generated at an interval of 0.0015 seconds. This attack lasts for a total of 30 seconds, disrupting the drone's functionality. During this time, the drone's flight behavior is disturbed, and critical attacks occasionally occur, causing the motors to stop.

• 90s to 120s: From 80 seconds to 90 seconds, normal drone data is generated as it resumes regular operation. However, after 90 seconds, another Fuzzy attack occurs, similar to the previous one, lasting for 30 seconds with 0.0015 seconds interval. During this attack, critical attacks also occur, leading to motor stoppages.

• 130s to 160s: From 120 seconds to 130 seconds, normal drone data is generated again, but after 130 seconds, an additional Fuzzy attack occurs, similar to the previous ones, lasting for 30 seconds with 0.0015 seconds interval. If critical attacks happen during this period, the motors stop.

• 160s onwards: From 160 seconds, no further attacks occur, and normal drone data is generated. The drone starts its landing behavior at 170 seconds, and at 180 seconds, the drone's operation ends normally.

6.4 Drone Attack Scenario 4

Scenario 4 is an attack during take-off, and it follows a similar pattern to Scenario 3 with a Fuzzy attack. However, in this scenario, the attack data is injected at the same frequency as regular data.

• 0s to 20s: The drone is powered on, and for the first 20 seconds, it goes through the booting process, waiting for the basic functions to execute.

• 20s to 50s: During this period, from 20 seconds to 50 seconds, the drone is taking off, and normal drone data is collected, representing regular operation.

• 50s to 80s: Starting from 50 seconds, the first Fuzzy attack occurs. In Scenario 4, the attack is slower than in Scenario 3, with an attack interval of 0.005 seconds. This attack lasts for 30 seconds, disrupting the drone's functionality, and the drone's flight behavior is disturbed, causing the motors to stop. This attack continues until 80 seconds.

• 90s to 120s: From 80 seconds to 90 seconds, normal drone data is generated as it resumes regular operation. However, after 90 seconds, another Fuzzy attack occurs, similar to the previous one, lasting for 30 seconds with a 0.005 seconds interval. During this attack, critical attacks also occur, leading to motor stoppages.

• 130s to 160s: From 120 seconds to 130 seconds, normal drone data is generated again, but after 130 seconds, an additional Fuzzy attack occurs, similar to the previous ones, lasting for 30 seconds with a 0.005 seconds interval. If critical attacks happen during this period, the motors stop.

• 160s onwards: From 160 seconds, no further attacks occur, and normal drone data is generated. The drone starts its landing behavior at 170 seconds, and at 180 seconds, the drone's operation ends normally.

6.5 Drone Attack Scenario 5

Scenario 5 involves attacking the drone while it is being piloted (during flight) with a Replay attack. A total of three attacks occur before the take-off process is completed. The specific attack method is as follows, as shown in Figure 20:

• 0s to 30s: Initially, the drone is powered on, and for the first 30 seconds, it goes through the booting process, waiting for the basic functions to execute.

• 30s to 60s: During this period, from 30 seconds to 60 seconds, the drone is taking off, and normal drone data is generated. Therefore, all the data during this timeframe represents regular drone operation.

**A diagram of a graph

Description automatically generated**

**A line with red squares and black text

Description automatically generated**

60s to 100s: Starting from 60 seconds, the first Replay attack occurs. The injected attack data is generated at an interval of 0.005 seconds. This attack lasts for a total of 40 seconds, disrupting the drone's normal movements. During this time, the drone's flight behavior is disturbed, causing it to move in a manner affected by the attack. Consequently, the drone continuously moves in the left direction, and normal control of its behavior becomes impossible. This attack continues until 100 seconds.

110s to 140s: From 100 seconds to 110 seconds, normal drone data is generated as it resumes regular operation. However, after 110 seconds, another Replay attack occurs, similar to the previous one, lasting for 30 seconds with a 0.005 seconds interval. During this attack, the drone's behavior continues to be affected, causing it to move leftwards uncontrollably.

160s to 200s: From 140 seconds to 160 seconds, normal drone data is generated again, but after 160 seconds, an additional Replay attack occurs, similar to the previous ones, lasting for 40 seconds with a 0.005 seconds interval. The drone continues to move in the left direction uncontrollably due to the attack during this period.

200s onwards: From 200 seconds, no further attacks occur, and normal drone data is generated. The drone starts its landing behavior at 210 seconds, and at 280 seconds, the drone's operation ends normally.

6.6 Drone Attack Scenario 6

Scenario 6 involves attacking the drone while it is being piloted (during flight) with a Replay attack. A total of four attacks occur before the take-off process is completed. The specific attack method is as follows, as shown in Figure 21:

• 0s to 30s: Initially, the drone is powered on, and for the first 30 seconds, it goes through the booting process, waiting for the basic functions to execute.

• 30s to 60s: During this period, from 30 seconds to 60 seconds, the drone is taking off, and normal drone data is generated, representing regular operation.

• 60s to 100s: Starting from 60 seconds, the first Replay attack occurs. In this scenario, the attack data is injected at an interval of 0.005 seconds. This attack lasts for a total of 40 seconds, disrupting the drone's normal movements. During this time, the drone's flight behavior is disturbed, causing it to move in a manner affected by the attack. Consequently, the drone continuously moves in the left direction, and normal control of its behavior becomes impossible.

• 110s to 150s: From 100 seconds to 110 seconds, normal drone data is generated as it resumes regular operation. However, after 110 seconds, another Replay attack occurs, similar to the previous one, lasting for 40 seconds with a 0.005 seconds interval. During this attack, the drone's behavior continues to be affected, causing it to move leftwards uncontrollably.

• 170s to 210s: From 150 seconds to 170 seconds, normal drone data is generated again, but after 170 seconds, an additional Replay attack occurs, similar to the previous ones, lasting for 40 seconds with a 0.005 seconds interval. The drone continues to move in the left direction uncontrollably due to the attack during this period.

• 220s to 260s: From 210 seconds to 220 seconds, normal drone data is generated again, but after 220 seconds, another Replay attack occurs, similar to the previous ones, lasting for 40 seconds with a 0.005 seconds interval. The drone continues to move in the left direction uncontrollably due to the attack during this period.

• 270s onwards: From 270 seconds, no further attacks occur, and normal drone data is generated. The drone starts its landing behavior at 280 seconds, and at 280 seconds, the drone's operation ends normally.

**A diagram of a graph

Description automatically generated**

6.7 Drone Attack Scenario 7

Scenario 7 involves attacking the drone while it is being piloted (during flight) with both Flooding and Fuzzy attacks, alternating between them. A total of four attacks occur before the take-off process is completed. The specific attack method is as follows, as shown in Figure 22:

• 0s to 30s: Initially, the drone is powered on, and for the first 30 seconds, it goes through the booting process, waiting for the basic functions to execute.

• 30s to 50s: During this period, from 30 seconds to 50 seconds, the drone is taking off, and normal drone data is generated, representing regular operation.

• 50s to 90s: Starting from 50 seconds, the first Flooding attack occurs. In this attack, the attack data is injected at an interval of 0.005 seconds. This attack lasts for a total of 40 seconds, disrupting the drone's normal movements. During this time, the drone's flight behavior is disturbed, and the drone may experience moments of motor shutdown and inability to maintain normal behavior or control.

• 100s to 130s: From 90 seconds to 100 seconds, normal drone data is generated as it resumes regular operation. However, after 100 seconds, the first Fuzzy attack occurs. This attack lasts for 30 seconds with a 0.005 seconds interval. During this attack, the drone's behavior is affected, causing it to experience disturbances in flight behavior and motor shutdown during critical attack moments.

• 140s to 180s: From 130 seconds to 140 seconds, normal drone data is generated again, but after 140 seconds, the second Flooding attack occurs. This attack is similar to the previous one, lasting for 40 seconds with a 0.005 seconds interval. During this time, the drone's flight behavior is again disturbed, and the drone may experience moments of motor shutdown and inability to maintain normal behavior or control. This attack continues until 180 seconds.

• 190s to 220s: From 180 seconds to 190 seconds, normal drone data is generated again, but after 190 seconds, the second Fuzzy attack occurs. This attack lasts for 30 seconds with a 0.005 seconds interval. Similar to the previous Fuzzy attack, the drone's behavior is affected, and it may experience disturbances in flight behavior and motor shutdown during critical attack moments. This attack continues until 220 seconds.

• 220s onwards: From 220 seconds, no further attacks occur, and normal drone data is generated. The drone starts its landing behavior at 240 seconds, and at 240 seconds, the drone's operation ends normally.

**A close up of a diagram

Description automatically generated**

6.8 Drone Attack Scenario 8

Scenario 8 involves attacking the drone while it is being piloted (during flight) with both Fuzzy and Replay attacks, alternating between them. A total of four attacks occur before the take-off process is completed. The specific attack method is as follows, as shown in Figure 23:

• 0s to 30s: Initially, the drone is powered on, and for the first 30 seconds, it goes through the booting process, waiting for the basic functions to execute.

• 30s to 60s: During this period, from 30 seconds to 60 seconds, the drone is taking off, and normal drone data is generated, representing regular operation.

• 60s to 100s: Starting from 60 seconds, the first Fuzzy attack occurs. In this attack, the attack data is injected at an interval of 0.005 seconds. This attack lasts for a total of 40 seconds, disrupting the drone's normal movements up to 100 seconds. During this time, the drone's flight behavior is disturbed, and it may experience moments of motor shutdown during critical attack moments.

• 110s to 140s: From 100 seconds to 110 seconds, normal drone data is generated as it resumes regular operation. However, after 110 seconds, the first Replay attack occurs. This attack lasts for 30 seconds with a 0.005 seconds interval. The first Replay attack continues until 140 seconds, during which the drone experiences disturbances in flight behavior and moves continuously in the left direction, making it impossible to maintain normal behavior or control.

• 150s to 190s: From 140 seconds to 150 seconds, normal drone data is generated again, but after 150 seconds, the second Fuzzy attack occurs. Similarly, this attack lasts for a total of 40 seconds, disrupting the drone's normal movements. During this time, the drone's flight behavior is disturbed, and it may experience moments of motor shutdown during critical attack moments. This attack continues until 190 seconds.

• 200s to 230s: From 190 seconds to 200 seconds, normal drone data is generated again, but after 200 seconds, the second Replay attack occurs. This attack lasts for 30 seconds with a 0.005 seconds interval. The second Replay attack continues until 230 seconds, during which the drone's flight behavior is disturbed, and it continuously moves in the left direction, making it impossible to maintain normal behavior or control.

• 230s onwards: From 230 seconds, no further attacks occur, and normal drone data is generated. The drone starts its landing behavior at 250 seconds, and at 250 seconds, the drone's operation ends normally.

**A diagram of a graph

Description automatically generated**

6.9 Drone Attack Scenario 9

Scenario 9 involves attacking the drone while it is taking off, with both Flooding and Replay attacks occurring alternately until a total of four attacks are executed before the drone completes its take-off. The specific attack procedure is as follows, as shown in Figure 24:

• 0s to 30s: Initially, the drone is powered on, and for the first 30 seconds, it goes through the booting process and waits for basic functionalities to be executed.

• 30s to 60s: From 30 seconds to 60 seconds, data is generated while the drone is taking off. Therefore, during this period, all the data is normal drone data.

• 60s to 110s: Starting from 60 seconds, the first Flooding attack occurs with attack data being injected at 0.005-second intervals. This attack disrupts the drone's normal movement for 50 seconds until 110 seconds. During this time, the drone experiences disturbances in its flight behavior and may come to a complete stop. Normal control over the drone is also impossible.

• 120s to 150s: From 110 seconds to 120 seconds, the drone generates normal flight data, and then from 120 seconds, the first Replay attack occurs. This attack lasts for 30 seconds, with attack data being injected at 0.005-second intervals. Until 150 seconds, the drone is disturbed in its flight behavior, moving continuously in the left direction, and normal control becomes impossible.

• 160s to 200s: From 150 seconds to 160 seconds, the drone generates normal flight data, and then from 160 seconds, the second Flooding attack occurs. Similar to the previous Flooding attack, this one disrupts the drone's normal movement for 40 seconds until 200 seconds. During this time, the drone experiences disturbances in its flight behavior and may come to a complete stop. Normal control over the drone is also impossible.

• 210s to 250s: From 200 seconds to 210 seconds, the drone generates normal flight data, and then from 210 seconds, the second Replay attack occurs. This attack lasts for 40 seconds, with attack data being injected at 0.005-second intervals. Until 250 seconds, the drone is disturbed in its flight behavior, moving continuously in the left direction, and normal control becomes impossible.

• 250s to end: From 250 seconds onwards, no further attacks occur, and the drone generates normal flight data. The drone then initiates the landing process, and at 270 seconds, the data shows the drone's normal termination.

(Note: The mentioned times are in seconds, representing the duration of each attack phase during the scenario.)

**A line of text and a red rectangle

Description automatically generated**

6.10 Drone Attack Scenario 10

Scenario 10 involves attacking the drone while it is taking off, with Flooding, Fuzzy, and Replay attacks occurring alternately, resulting in a total of three attacks before the drone completes its take-off. The specific attack procedure is as follows, as shown in Figure 25:

• 0s to 30s: Initially, the drone is powered on, and for the first 30 seconds, it goes through the booting process and waits for basic functionalities to be executed.

• 30s to 60s: From 30 seconds to 60 seconds, data is generated while the drone is taking off. Therefore, during this period, all the data is normal drone data.

• 60s to 110s: Starting from 60 seconds, the first Flooding attack occurs with attack data being injected at 0.005-second intervals. This attack disrupts the drone's normal movement for 50 seconds until 110 seconds. During this time, the drone experiences disturbances in its flight behavior and may come to a complete stop. Normal control over the drone is also impossible.

• 120s to 160s: From 110 seconds to 120 seconds, the drone generates normal flight data, and then from 120 seconds, the first Fuzzy attack occurs. This attack lasts for 40 seconds, with attack data being injected at 0.005-second intervals. Until 160 seconds, the drone is disturbed in its flight behavior, with critical attacks causing the motors to stop at times.

• 170s to 200s: From 160 seconds to 170 seconds, the drone generates normal flight data, and then from 170 seconds, the second Replay attack occurs. Similar to the previous Replay attack, this one disrupts the drone's normal movement for 30 seconds until 200 seconds. During this time, the drone experiences disturbances in its flight behavior, moving continuously in the left direction, and normal control becomes impossible.

• 200s to end: From 200 seconds onwards, no further attacks occur, and the drone generates normal flight data. The drone then initiates the landing process, and at 220 seconds, the data shows the drone's normal termination.

(Note: The mentioned times are in seconds, representing the duration of each attack phase during the scenario.)

**A black and red rectangle with black text

Description automatically generated**

7 Dataset Information (Metadata)

In this section, we collect the normal data and attack data generated after conducting the attack scenarios in Section 6 to construct the dataset.

7.1 Dataset Metadata

The dataset consists of a total of 10 sets. Table 1 describes the characteristics of each dataset, and the explanations for each title are as follows:

• Interval: The duration during which the attacker remains idle after sending a packet before generating the next packet. The actual time interval between attack packets may be longer due to factors such as packet generation time and the limitations of the CAN bus. The unit of measurement is seconds.

• Total Time: The collection time of frames included in the dataset. The unit of measurement is seconds.

• DataFrame(N/A): This represents the number of frames in the CAN protocol, which is a sub-protocol of DroneCAN (UAVCAN v0). "N" denotes the number of normal frames generated by nodes connected to the CAN bus, and "A" represents the number of attack frames generated by the attacker.

**A table with text and numbers

Description automatically generated**

7.2 Dataset Structure

The dataset is created based on the attack scenarios and is generated from the data produced by UAVCAN. The attack data and normal data are labeled, allowing them to be used in research that involves distinguishing between attack and normal data.

The structure of the dataset is as shown in Figure 26. It consists of the following columns: Label, Timestamp, Interface, CAN ID, DLC, and Data. Firstly, the Label column has two possible values: "Normal" when drone's normal data is collected and "Attack" when attack data is collected. Next is the Timestamp, which represents the relative time starting from 0 when the drone begins and collects data. The relative time is recorded in seconds. The Interface column records the name of the device that transmits and receives UAVCAN data. The CAN ID is recorded according to the rules for generating UAVCAN IDs explained in Section 3. The DLC (Data Length Code) indicates the length of the collected data. Finally, the Data column contains the recorded values of the data generated by the drone.

Except for the Label column, the rest of the columns match the output of Linux's CAN protocol utility, SocketCAN. The labeling process involves separately storing attack frames during dataset creation and later comparing them with frames stored using SocketCAN to determine whether a frame is an attack frame. Therefore, during scenarios, when attacks occur within specific time intervals, the corresponding attack data is labeled as "Attack," while all other normal data is labeled as "Normal."

Top of Form

**A close-up of a number

Description automatically generated**