UAV and UAV Controller Authentication and Communication Security

Abhinav Kumar

2019CS50415

Under the guidance of

Prof. Vireshwar Kumar, IIT Delhi

Submitted in partial satisfaction of the requirements for the degree of

Dual Degree Computer Science & Engineering

School of Computer Science

Indian Institute of Technology Delhi

2021

Acknowledgements

I would like to express my special thanks of gratitude to my supervisor **Prof. Vireshwar Kumar** who gave me the golden opportunity to do this wonderful project on the topic of UAV and UAV Controller Authentication and Communication Security. I came to know about so many new things after completing my research project. This was my first research project so I was a bit scared but after hearing to Prof Vireshwar Kumar in every meeting, I always got the motivate that I can do this. Also, I want to thanks to my friends who helped and encouraged me throughout this project.

Abhinav Kumar

Abstract

We are now living in a world where the use of Unmanned systems are growing very fast. Unmanned aerial vehicles (UAVs), also known as drones are are being used in many real world application like defense, commercial services, search and rescue and monitoring. Even though drones can provide many benefits, they can also pose serious security threats. One of the biggest challenge is to make the command and control communication between the UAV and UAV controller secure. A possible communication protocol for bi-directional communication between the drone and a GCS is the MAVLink protocol. If we want to make drone usable for different applications, then firstly we needs to make sure that the communication protocol by which ground control station and unmanned aerial vehicle communicates is secure. The MAVLink protocol has many vulnerabilities and flaws in communication. Any attacker with basic knowledge of penetration testing and about attacking tools can try to scan the drone services and can disrupt the services of drone. In our research we focussed on the MAVLink protocol because this is trying to become a worldwide standard. Our goal is to carry security analysis of MAVLink protocol and find some security vulnerability. For this, I tried to make an attacker try to communicate with the drone with the help of mavlink protocol at the same time ground control station is also connected with drone and send some irregular messages to the drone while the real user is using the drone for a simple go to mission. It resulted in making drop stopped for some seconds at the location where it is currently flying. Further analysis by more type of attacjer scripts result in stopping the drone for much more seconds and then drone suddenly appearing at different location in the actual path of the mission which is given by actual user. This was done in simulated environment and we can also verify it in real environment.

Table of Contents

List of Figures

Chapter 1

Introduction

Unmanned systems are autonomous platforms that can be easily programmed to perform many operations without the intervention of a pilot. This section introduces the conducted research. We start with the motivation for the research. This will be followed by the related work, our research goals, research questions and our approach.

1.1 Motivation

Unmanned aerial vehicles (UAVs), also known as drones, are vehicles that are controlled remotely from a Ground Control Station (GCS) or autonomously by a programmed mission which are the UAV Controllers. There are two main purposes of drones that can be identified, drones for military usage and drones for commercial usage. There are different types of UAVs based on payload, flying mechanism, range and altitude, speed and flight time and power supply. With the enhanced data rate, the 5G cellular networks are touted to realize the role of UAVs in various applications. All the major standardization bodies like 3GPP have already established dedicated study items and working groups to analyse the specific requirements for enabling reliable communications towards and from flying UAV through current cellular networks. As UAVs are becoming widespread and their demand is growing, they are becoming more prone to security attacks. Although the communication between a UAV and its ground control station (GCS) can potentially be authenticated using the most recent version of the Micro Air Vehicle Link (MAVLink) protocol, it is unclear whether the integration of the UAV network using MAVLink with the cellular network using 5G introduces some new security vulnerabilities In particular, the wireless communication channel opens up the door for several types of remote attacks. For example, adversaries could attempt to obtain sensitive data by eavesdropping the wireless medium, send malicious commands to the drone or even alter its software. If a drone is hijacked it can pose serious threats.

1.2 Related Work

In the last decade, the amount of commercial drones sold increased enormously. Drones are getting cheaper and people start experimenting with them, resulting in interesting projects. For example, VirusCopter consists of a virus, acting as a worm, capable of infecting AR Drones. Also Facebook Aquila and Google Loon are drone for providing coverage in remote area where either ground wireless infrastructure is not feasible or to improve end user performance[4]. There are a lot of research already done and a lot are ongoing on the security of drone and its controller communication. There are a lot of different type of security vulnerability possible and so there are work going on to find it out. There are also some research done on encrypting the messages which is communicated between drone to drone and drone to its controller and got some good result also.

A possible communication protocol for bi-directional communication between the drone and a GCS is the MAVLink protocol and is attempting to become a worldwide standard. The MAVLink protocol version 1 does not provide any security measures and the MAVLink version 2 provide only signing of messages. Research by J. Marty proposed a methodology to quantify the cost of securing the MAVLink protocol through the measurement of network latency, power consumption, and exploit success[6]. In addition, the research of Thomas M. Dubuisson introduces a light-weight encapsulation format that can be used with MAVLink to protect against forgery, replay attacks, and snooping. This is accomplished by using cryptographic solutions, and adds an additional 16 bytes to each message[7]. Subsequently, N. Butcher et. al studied the possible wireless attacks on MAVLink, and proposed the RC5 encryption algorithm as a countermeasure to these attacks and studied how encryption affects the performance of the drone[8]. We can conclude that adding encryption to the protocol is needed to protect sensitive data, but until today this has not yet been implemented in the official distribution of the MAVLink protocol.

1.3 Our Approach

Our goal is to carry out a security analysis of the MAVLink protocol by illegally connecting to drone and sending message which can make drone not work as usual or maybe stop it or crash it. We will try different type of messages and in different ways of writing it and then will be send to drone by the attacker. For our research, the MAVLink protocol specifications such as the different messages and messages structure have been examined in detail to construct attacker script.

1.4 Organization

Chapter 2 provides an detailed introduction to the MAVLink protocol as it is the main subject of our research. Chapter 3,explains the methodology of the research containing the lab setup and the attacking methodology. Chapter 4, gives the details about the implementation. Chapter 5 discusses the results of the work and We conclude with final remarks and suggestions for future work in chapter 6.

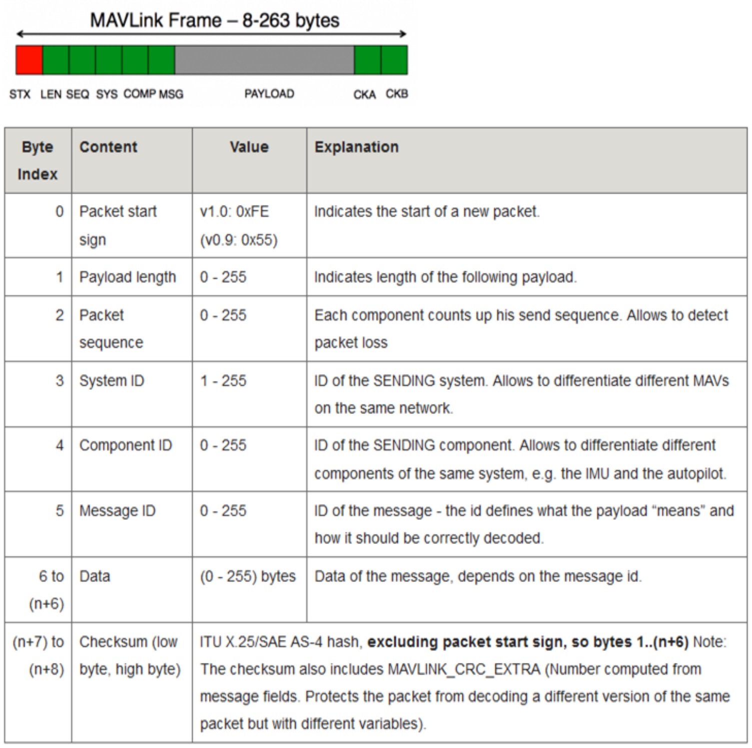
Chapter 2

MAVLink Protocol

The MAVLink is a very lightweight messaging protocol allows entities to communicate over a wireless channel[3]. When used in drones, it is used for the bidirectional communications between the UAV and the GCS. The GCS sends commands and controls to the drone whereas the drone sends telemetry and status information. MAVLink was first released in 2009 under the GNU Lesser General Public License and is now part of the DroneCode project, governed by the Linux Foundation. Currently, the DroneCode project has thousands of developers and over a hundred thousand of users. MAVLink is used in many different Autopilot systems like ArduPilotMega, pxIMU Autopilot, SLUGS Autopilot, etc. It is also used in software packages like iDoneCtrl(IOS), QGroundControl (Windows/Mac/Linux), APM Planner, MAVProxy, etc. Finally, there are many projects using MAVLink, these include ArduPilotMega, MatrixPilot, PIXHAWK, etc[2].

MAVLink messages are defined within XML files. Each XML file defines the message set supported by a particular MAVLink system, also referred to as a "dialect". The reference message set that is implemented by most ground control stations and autopilots is defined in common.xml (most dialects build on top of this definition).

A MAVLink message is sent bytewise over the communication channel, followed by a checksum for error correction. The mavlink frame is given in fig 2.1 below. If the checksum does not match, then it means that the message is corrupted and will be discarded. The protocol defines a large set of messages. The messages are not guaranteed to be delivered which means ground stations or companion computers must often check the state of the vehicle to determine if a command has been executed. MAVLink message types are identified by the ID field on the packet and the payload contains the appropriate data. Once a connection is opened each device (aka “System”) sends the HEARTBEAT message at 1Hz. MAVLink 2 allows 24 bit message ID and adds support for message signing, which allows a MAVLink system to verify that messages originate from a trusted source.



Byte Index	Content	Value	Explanation
0	Packet start sign	v1.0: 0xFE (v0.9: 0x55)	Indicates the start of a new packet.
1	Payload length	0 - 255	Indicates length of the following payload.
2	Packet sequence	0 - 255	Each component counts up his send sequence. Allows to detect packet loss
3	System ID	1 - 255	ID of the SENDING system. Allows to differentiate different MAVs on the same network.
4	Component ID	0 - 255	ID of the SENDING component. Allows to differentiate different components of the same system, e.g. the IMU and the autopilot.
5	Message ID	0 - 255	ID of the message - the id defines what the payload “means” and how it should be correctly decoded.
6 to (n+6)	Data	(0 - 255) bytes	Data of the message, depends on the message id.
(n+7) to (n+8)	Checksum (low byte, high byte)	ITU X.25/SAE AS-4 hash, **excluding packet start sign, so bytes 1..(n+6)** Note: The checksum also includes MAVLINK_CRC_EXTRA (Number computed from message fields. Protects the packet from decoding a different version of the same packet but with different variables).	

Figure 2.1: MAVLink frame

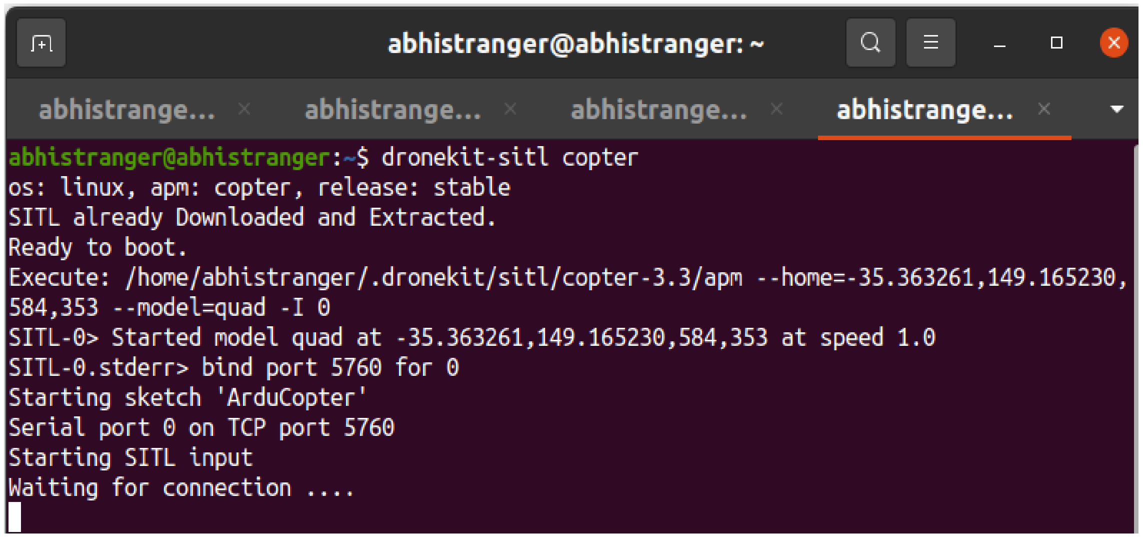
Chapter 3

Methodology

This chapter describes the setup that is used for the experiments. In the real world, the setup consists of a drone and a GCS. In our experiments we use a simulator from the SITL environment, providing a virtual drone with the GCS. First, the whole setup is explained, describing the used equipment, software and how to connect all components together. Afterwards, the attacking methodology is described in detail.

3.1 Setup

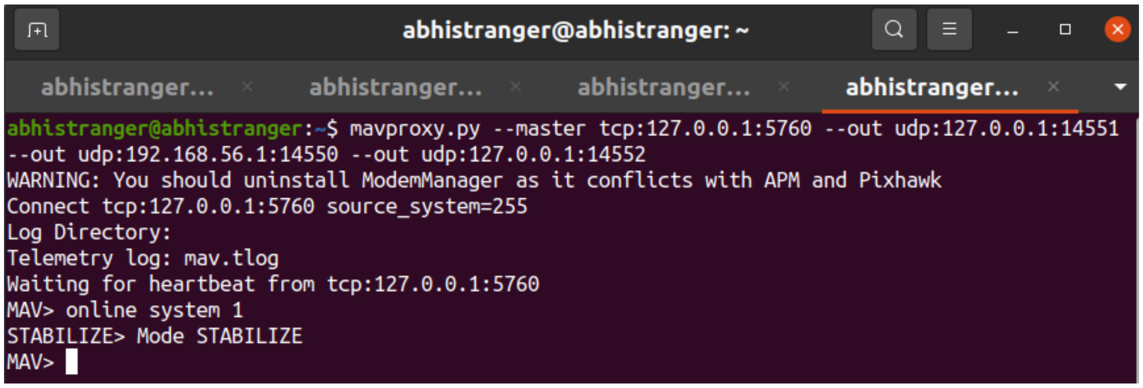
The SITL simulator[1] is run on a Linux virtual machine (Ubuntu 20.04 TLS). The laboratory setup employs the simulator for a multirotor UAV, ArduCopter. The simulator can be run as shown in Figure 3.1.



```
abhistranger@abhistranger:~$ dronekit-sitl copter
os: linux, apm: copter, release: stable
SITL already Downloaded and Extracted.
Ready to boot.
Execute: /home/abhistranger/.dronekit/sitl/copter-3.3/apm --home=-35.363261,149.165230,
584,353 --model=quad -I 0
SITL-0> Started model quad at -35.363261,149.165230,584,353 at speed 1.0
SITL-0.stderr> bind port 5760 for 0
Starting sketch 'ArduCopter'
Serial port 0 on TCP port 5760
Starting SITL input
Waiting for connection ....
```

Figure 3.1: Starting the Copter

This starts the ArduCopter simulation for a QuadCopter, at a home location with coordinates -35.363261,149.165230,584,353, all of the internal memory wiped and faster operation. The output shows that after execution of the command, a connection to the SITL can be established using TCP/IP at the network address of the machine running STIL at port 5760. In the normal setup, SITL is used in combination with a GCS, like MAVProxy. Now we will use mavproxy to connect to the drone and provide a way through UDP port to communicate to the drone by connecting to the UDP port. The connection of mavproxy to tcp port 5760 is shown in Figure 3.2.

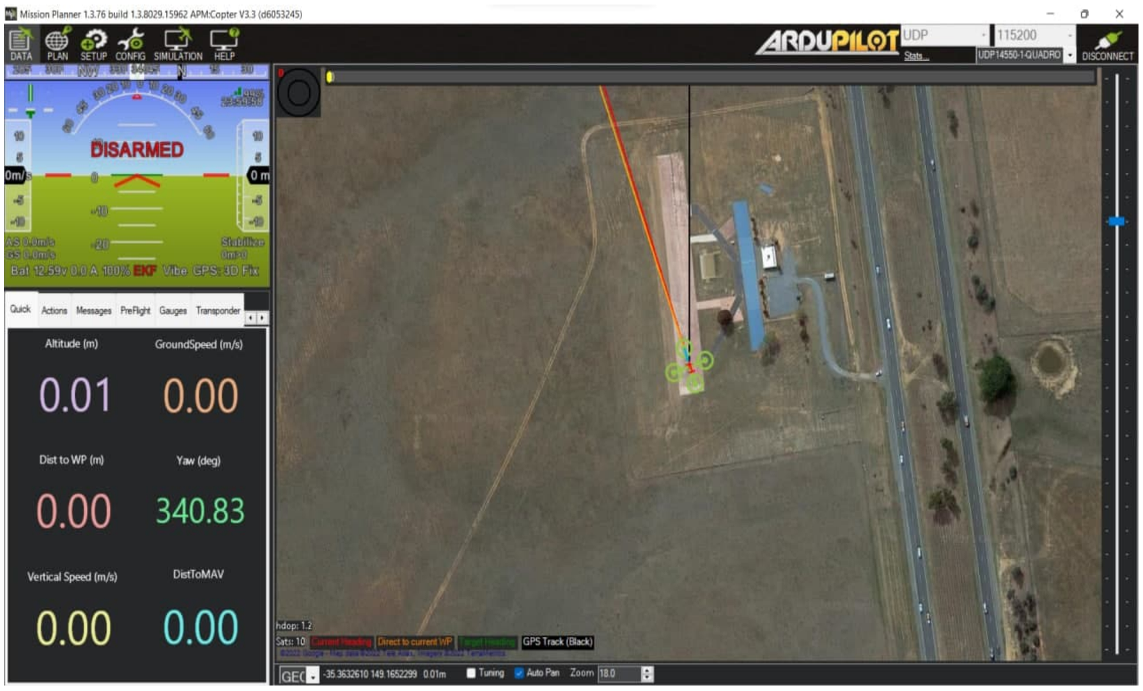


```
abhistranger@abhistranger:~$ mavproxy.py --master tcp:127.0.0.1:5760 --out udp:127.0.0.1:14551
--out udp:192.168.56.1:14550 --out udp:127.0.0.1:14552
WARNING: You should uninstall ModemManager as it conflicts with APM and Pixhawk
Connect tcp:127.0.0.1:5760 source_system=255
Log Directory:
Telemetry log: mav.tlog
Waiting for heartbeat from tcp:127.0.0.1:5760
MAV> online system 1
STABILIZE> Mode STABILIZE
MAV>
```

Figure 3.2: Connection of mavproxy to the drone

It will now provide three UDP port where one is for the mission planner gui, other is for the real user and third is for the attacker. Now we will connect mission planner gui for the graphics and connect via UDP port 14550 and it is running in my Windows 11 pc. After connecting the mission planner will look like as shown in figure 3.3 where the drove is at home location.

Now We will have a normal user connected to the drone and We will make another user which will work as attacker which will also be connected to the drone. Also when we run these file we will start wireshark to capture these packets to know about the messages sent.



Altitude (m) 0.01
GroundSpeed (m/s) 0.00
Dist to WP (m) 0.00
Yaw (deg) 340.83
Vertical Speed (m/s) 0.00
DistToMAV 0.00

-35.3632610 149.1652299 0.01m
Zoom 18.0

Figure 3.3: Mission Planner

Chapter 4

Implementation

This chapter describe the details about the implementation part of the research. As we are doing everything in python, there are two ways to communicate with drone that is by either using pymavlink or dronekit. As now we have two user one is real and other is attacker so we have two scripts for sending the message to the drone. It can be written in any of the two ways. The real user file is real.py and the attacker file is attacker1.py and there are more attacker file which you can see on the github repo here: .

The first thing for any user is to connect to the drone. To connect with the drone, connection with the drone is established which we can be established in two ways:

1. By pymavlink: For setting up communication links and receiving and decoding messages, running periodic tasks etc mavutil utility functions can be used. Mavutil utility is imported to use its functions by creating a connection as follows:

```
from pymavlink import mavutil
master = mavutil.mavlink_connection('service protocol :IP
 ↪  ADDRESS:PORT')
```

This connection object will be used to send and receive message from drone.

2. By Dronekit: connect function from Dronkit library is imported which is used to make a connection with drone as follows :

```
from Dronekit import connect
vehicle = connect('IP_ADDRESS:PORT_NUMBER', wait_ready=True)
```

this vehicle object is used communicate with drone.

The user.py which is the real user script contains a mission of a drone going from home location to other location and then coming back to its home location with a air speed of 1m/s and this is the real user giving these commands to the drone.

The code for the attacker is pretty simple to just get connected with the drone and send some messages while actual user is also sending commands to the drone.

attacker1.py is a simple attacker file by using dronkit which just send a message to change speed from 1 to 10 while drone is in air moving with speed 1:

```
import time
from dronekit import connect, VehicleMode, LocationGlobalRelative
connection_string='udp:127.0.0.1:14552'
print('Connecting to vehicle on: %s' % connection_string)
vehicle = connect(connection_string, wait_ready=True)
vehicle.airspeed = 10
print("Close vehicle object")
vehicle.close()
```

attacker1_pymav.py which is same as attacker1.py but its written by using pymavlink:

```
from pymavlink.dialects.v20 import common as mavlink2
from pymavlink.dialects.v10 import ardupilotmega as mavlink1
from pymavlink import mavutil
import sys
```

```
the_connection = mavutil.mavlink_connection('udp:127.0.0.1:14552')
the_connection.wait_heartbeat()
print("Heartbeat from system (system %u component %u)" %
↪  (the_connection.target_system, the_connection.target_system))
#sending heartbeat
the_connection.mav.heartbeat_send(mavutil.mavlink.MAV_TYPE_GCS,
↪  mavutil.mavlink.MAV_AUTOPILOT_INVALID, 0, 0, 0)
#COMMAND_INT
the_connection.mav.command_long_send(the_connection.target_system,
↪  the_connection.target_component,
↪  mavutil.mavlink.MAV_CMD_DO_CHANGE_SPEED, 0, 0, 100, -1, 0, 0, 0,
↪  float('NaN'))
print("Airspeed set to 10m/s")
try:
    Res=the_connection.messages['COMMAND_ACK'].result
    timestamp=the_connection.time_since('COMMAND_ACK')
except:
    print('No COMMAND_ACK message received')
```

There are also other attacker files which we used for the test. You can find all the code and on github repo here: https://github.com/abhistranger/UAV-and-UAV-Controller-Authentication-and-Communication-Security

Chapter 5

Results

After running the real user script which is user.py and trying different attacker script, I was able to identify possible security flaw. The results of the different attacker scripts used for our research will be analysed in this chapter. By running attacker1.py, I was able to stop the drone for some second and was able to change the speed. That is if we connect an attacker and send command for changing the speed while the drone is flying at different speed which is given by real user, then the drone stooped for some sec and also changed the speed which in real life drone will make the drone stopped working or may be it would crash or it may fly but with a different speed. By running attacker2.py, I was able to stop the drop for a long time and when it again come back it was on a different location which was far from the location at which it stopped, in the simulation. And this is very strange behaviour which might lead to crash in actual environment. There are also another attacker file which result in a similar way like this. I also experimented with changing speed to different values and also running the different scripts suring one complete mission of the real user. The result was similar and this shows that we can send a command without being a legitimate user and so we can actually take control of the drone which is very big security threat.

Chapter 6

Conclusions and Future Works

As drones are getting more common in today's life, the security of drones is becoming a hot topic for research. This work aims to identify security vulnerabilities, by using the technique of fuzzing. We focused our research on the MAVLink protocol. MAVLink is used as a communication protocol between a drone and a ground control station. The protocol is actively developed by the community and aims to become one of the drone communication standards. Our aim is to contribute to the identification of the security flaws of the protocol and to help the development community to mitigate these flaws. To conduct our research, we have studies the specifications of the MAVLink protocol in detail.

After analyzing the results of our research, we have found that there is vulnerability in the protocol. It stopped the drone for sometime in the air. We also did the whole attack with using both dronkit and pymavlink. But we are unable to use mavlink version 2 so we used mavlink version 1.

Future work can start with doing this entire attack by using mavlink 2 so that we can be sure of it in mavlink 2 also as I was unable to do this. There are also a wide range of messages which we can try to send and see what happens as only a few range of message are tested. There are also some messages which are under development so we can also try it and maybe we can get another type of security vulnerability. We can also improve the attacker script by adding more commands to send in one go.

The research can be repeated on a real drone so that we can be so sure of the current vulnerability. We can then try to find out solutions to tackle the vulnerability and there is a lot we can try where we can use cryptography or machine learning or even something else.

References

[1] Ardupilot. Software in the loop. URL:https://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html.

[2] Karel Domin (2016). "Security Analysis of the Drone Communication Protocol: Fuzzing the MAVLink Protocol".

[3] MAVLink Protocol. URL:https://mavlink.io/en/.

[4] Azade Fotouhi, Ming Ding, Lorenzo Galati Giordano and Jinhong Yuan (2019). "Survey on UAV Cellular Communications: Practical Aspects, Standardization Advancements, Regulation, and Security Challenges".

[5] 3rd Generation Partnership Project. "Technical Specification Group Services and System Aspects; Study on security aspects of Unmanned Aerial Systems (UAS)". Release: 17, 3GPP TR 33.854 V0.6.0 (2021-05).

[6] J. A. Marty. "Vulnerability analysis of the mavlink protocol for command and control of unmanned aircraft". Master's thesis, Air Force Institute of Technology, 2014.

[7] T. M. DuBuisson. "Smaccmpilot secure mavlink communications". Technical report, Galois, Inc.1, 2013.

[8] A. Stewart N. Butcher and Dr. S. Biaz. "Securing the mavlink communication protocol for unmanned aircraft systems". Technical report, Appalachian State University, Auburn University, 2013.

[9] Azza Allouch, Omar Cheikhrouhou, Anis Koubâa, Mohamed Khalgui and Tarek Abbes. "MAVSec: Securing the MAVLink Protocol for Ardupilot/PX4 Unmanned Aerial Systems". 2019 15th International Wireless Communications Mobile Computing Conference (IWCMC).

[10] Anis Koubaa, Azza Allouch, Maram Mohamed Alajlan and Mohamed Khalgui"Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey". June 2019.

[11] Seonghoon Jeong, Eunji Park, Kang Uk Seo, Jeong Do Yoo, and Huy Kang Kim. "MUVIDS: False MAVLink Injection Attack Detection in Communication for Unmanned Vehicles". School of Cybersecurity, Korea University, 2021.

[12] 5G Americas. "Security Considerations for the 5G Era".