EXPLORING SECURITY VULNERABILITIES OF UNMANNED AERIAL VEHICLES

MASTER THESIS

NILS RODDAY

Supervisors at UTwente:
Prof. dr. ir. Aiko Pras
Dr. Ricardo de Oliveira Schmidt

Supervisors at KPMG:
Matthieu Paques
Ruud Verbij

Faculty for Electrical Engineering, Mathematics and Computer Science
Department of Computer Science & Electrical Engineering
DACS Research Group

Amsterdam
July 2015
To my parents.
ABSTRACT

We are currently observing a significant increase in popularity of Unmanned Aerial Vehicles (UAVs). This is not only the case for consumer UAVs, but also for more sophisticated UAVs used for professional services, whereby the cost can easily be magnitudes higher. Given that professional UAVs are being widely deployed in sensitive missions such as monitoring of critical infrastructures and police operations, security aspects of such UAVs should be questioned. In this research we investigated the level of security applied to the communication channels of a professional UAV. Furthermore, we explored security vulnerabilities that were identified, performed a Man-in-the-Middle attack and injected control commands to interact with the UAV. In addition, appropriate countermeasures to improve the current level of security were suggested. Our findings raise awareness within (i) the general public that use and trust UAVs, (ii) the scientific community by showing that further investigation is needed in this area, and (iii) the manufacturers by showing the importance of implementing a higher level of security in their devices.
ACKNOWLEDGMENTS

This thesis would not have been possible without the guidance and support of many people.

First of all, I would like to thank my first supervisor, Dr. Ricardo de Oliveira Schmidt, as well as my second supervisor, Prof. dr. ir. Aiko Pras, at the University of Twente, for providing me assistance and support throughout the time of the thesis.

Secondly, I am thankful that Matthieu Paques and Ruud Verbij from KPMG Netherlands took the time to supervise and support me within KPMG on a weekly basis. Additionally, other colleagues gave me the chance for discussions and I valued their input very much.

Thirdly, the Startup providing the UAV was for obvious reasons very important to my research. Their help and openness for cooperation impressed me and clearly had a positive impact on my own attitude towards future work.

Apart from the people directly involved in the research I performed, there were many other people who helped me in different fields. I would like to thank Atul Kumar for spending time with me reverse-engineering Android applications, Geert Jan Laanstra for providing technical support and equipment from the University of Twente and Annika Dahms for reading and correcting my thesis multiple times.

Additionally, I am very grateful to have such a supportive family. I would like to thank my parents, Gudula and Frank Rodday, for encouraging me to take up higher studies, together with my sister Lara Rodday and my brother Lars Rodday for their unconditional love and support.

Finally, I am very glad that I had the chance to study within the Security & Privacy track of the EIT Digital - Master School. The program provided me with two exceptional years of study in foreign countries strengthening both my professional as well as social skills. I certainly made friends for life.

Nils Rodday
Amsterdam
June 25th, 2015
# CONTENTS

## 1 INTRODUCTION

1.1 Motivation ............................................. 1  
1.2 Research Goal, Research Questions and Their Approach 2  
1.3 Research Contributions ............................. 3  
1.4 Organization of the Thesis .......................... 4  

## 2 RELATED RESEARCH

2.1 Recent Drone Hacks .................................. 5  
2.2 Warflying ............................................. 6  
2.3 GPS Spoofing .......................................... 6  
2.4 WiFi & Bluetooth Cracking ............................ 8  
2.5 Zigbee & XBee .......................................... 8  
2.6 KillerBee .............................................. 9  

## 3 UNMANNED AERIAL VEHICLES

3.1 Definition ........................................... 11  
3.2 Classification of UAVs ............................... 13  
3.2.1 EUROUVS .......................................... 13  
3.2.2 Austin ............................................. 14  
3.2.3 Comparison ......................................... 15  
3.2.4 Consumer & Professional UAV .................. 16  
3.3 Control Channel ..................................... 16  
3.3.1 Infrared ........................................... 16  
3.3.2 Radio Waves ....................................... 17  
3.4 Attack Vectors ....................................... 19  
3.4.1 Physical ........................................... 19  
3.4.2 Logical ............................................ 20  

## 4 THE INVESTIGATED UAV

4.1 Setup .................................................. 25  
4.2 WiFi Link .............................................. 26  
4.3 Telemetry Link ........................................ 26  
4.3.1 XBee Experiments Setup ......................... 27  
4.3.2 XBee PCB .......................................... 28  
4.3.3 Spectral Analysis ................................ 29  
4.3.4 Overview .......................................... 30  
4.3.5 Brute-force ....................................... 31  
4.3.6 Software-Defined Radio ......................... 34  
4.3.7 XBee API-mode and Broadcast .................. 36  
4.4 Manual Remote Control Link ....................... 40  
4.5 Video Link ............................................. 41  
4.6 GPS .................................................... 41  
4.7 Concluding Remarks .................................. 42  

## 5 ATTACK SCENARIOS & COUNTERMEASURES

5.1 Investigated UAV - WiFi Communication .......... 43
5.1.1 Attacks on the WiFi Communication
5.1.2 WiFi Cracking
5.1.3 Attack Countermeasures
5.2 Investigated UAV - XBee Communication
5.2.1 Attacks on the XBee Communication
5.2.2 Popularity of XBee
5.2.3 Reconstruction of Commands
5.2.4 Emulation of Android Tablet
5.2.5 Proof-of-Concept
5.2.6 Attack Countermeasures
5.3 Concluding Remarks

6 CONCLUSION AND FUTURE WORK
6.1 Conclusion
6.2 Future Work

A APPENDIX A
B APPENDIX B
C APPENDIX C
D APPENDIX D
E APPENDIX E
F APPENDIX F

BIBLIOGRAPHY
LIST OF FIGURES

Figure 3.1 Definition of terms (based on [31]) ............................................. 12
Figure 3.2 Interceptor MP200 ................................................................. 20
Figure 3.3 Rapere Drone ........................................................................... 20
Figure 4.1 Communication links of the UAV .............................................. 26
Figure 4.2 Setup for experiments ............................................................... 27
Figure 4.3 XBee chip on proprietary PCB ............................................... 28
Figure 4.4 XBee chip on proprietary PCB - removed cover ..................... 29
Figure 4.5 Spectral analysis with full range .............................................. 30
Figure 4.6 Spectral analysis with zoom ..................................................... 30
Figure 4.7 XBee packet composition in transparent mode ......................... 37
Figure 4.8 XBee packet composition in API mode .................................... 37
Figure 5.1 Attack on the WiFi link of the UAS ......................................... 43
Figure 5.2 Take-over of WiFi connection ................................................ 44
Figure 5.3 Display all available networks - airmon-ng ................................ 45
Figure 5.4 Fake authentication - aireplay-ng .......................................... 46
Figure 5.5 ARP replay - aireplay-ng ......................................................... 47
Figure 5.6 Successful key cracking - aircrack-ng ..................................... 47
Figure 5.7 Attack on the XBee link of the UAS ......................................... 48
Figure 5.8 DoS & Man-in-the-Middle attack scenario ............................... 50
Figure 5.9 Android App decompilation - file tree .................................... 52
Figure 5.10 Error Message - Unsupported device ..................................... 55
Figure 5.11 DoS & Man-in-the-Middle attack scenario ............................... 58
Figure 5.12 XCTU Security ..................................................................... 59
Figure 5.13 Schematic XBee standard communication ............................. 60
Figure 5.14 XBee & Dedicated hardware encryption ................................. 60
Figure B.1 XBee Remote AT Command Frame, from [64] ....................... 68

LIST OF TABLES

Table 3.1 UAV Classification - EUROUVS .............................................. 13
Table 3.2 UAV Classification - Austin (based on [31]) .............................. 15
Table 3.3 Standard radar frequency letter band nomenclature ..................... 18
Table 4.1 Brute-force calculation for XBee link ........................................ 32
Table 4.2 Brute-force calculation for XBee link with address patterns .......... 33
Table 4.3 XBee throughput rate with or without encryption ........................ 40
Table D.1 Commands for flight computer ............................................... 70
LISTINGS

Listing 5.1    Android APK - Command construction . . .  . . .  53
Listing 5.2    Model Check in Android APK ................  56
Listing C.1    Android APK - Command construction 2 . . .  69
Listing F.1    Proof-of-Concept - Source Code ..............  73

ACRONYMS

AES         Advanced Encryption Standard
AFA         Adaptive Frequency Agility
AP          Access Point
API         Application Programming Interface
APK         Android Application Package
ARP         Address Resolution Protocol
AVD         Android Virtual Device
BLE         Bluetooth Low Energy
CA          Certificate Authority
cm          Centimeter
CR          Close Range
DEC         Decoy
DH          Device High
DL          Device Low
DoS         Denial of Service
DPA         Differential Power Analysis
ECC         European Communications Committee
EEPROM      Electrically Erasable Programmable Read-Only Memory
EN          Endurance
EU          European Union
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXO</td>
<td>Exo-Stratospheric</td>
</tr>
<tr>
<td>FCC</td>
<td>Federal Communications Commission</td>
</tr>
<tr>
<td>FHSS</td>
<td>Frequency Hopping Spread Spectrum</td>
</tr>
<tr>
<td>FTDI</td>
<td>Future Technology Devices International</td>
</tr>
<tr>
<td>GHz</td>
<td>Gigahertz</td>
</tr>
<tr>
<td>GND</td>
<td>Ground</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>GSM</td>
<td>Global System for Mobile Communications</td>
</tr>
<tr>
<td>GTAM</td>
<td>Ground to Air Missile</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HALE</td>
<td>High Altitude Long Endurance</td>
</tr>
<tr>
<td>HAXM</td>
<td>Hardware Accelerated Execution Manager</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineers</td>
</tr>
<tr>
<td>INS</td>
<td>Inertial Navigation System</td>
</tr>
<tr>
<td>IV</td>
<td>Initialization Vector</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>IR</td>
<td>Infrared</td>
</tr>
<tr>
<td>Kbps</td>
<td>Kilobits per second</td>
</tr>
<tr>
<td>km</td>
<td>kilometer</td>
</tr>
<tr>
<td>LBT</td>
<td>Listen Before Talk</td>
</tr>
<tr>
<td>LET</td>
<td>Lethal</td>
</tr>
<tr>
<td>LR</td>
<td>Long Range</td>
</tr>
<tr>
<td>MAC</td>
<td>Media Access Control</td>
</tr>
<tr>
<td>MALE</td>
<td>Medium Altitude Long Endurance</td>
</tr>
<tr>
<td>MAV</td>
<td>Micro Aerial Vehicle</td>
</tr>
<tr>
<td>MHz</td>
<td>Megahertz</td>
</tr>
<tr>
<td>MitM</td>
<td>Man-in-the-Middle</td>
</tr>
<tr>
<td>MR</td>
<td>Medium Range</td>
</tr>
<tr>
<td>MUAV</td>
<td>Mini Unmanned Aerial Vehicle</td>
</tr>
<tr>
<td>Acronym</td>
<td>Definition</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>NAV</td>
<td>Nano Aerial Vehicle</td>
</tr>
<tr>
<td>NDA</td>
<td>Non-Disclosure Agreement</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PCB</td>
<td>Printed Circuit Board</td>
</tr>
<tr>
<td>PoC</td>
<td>Proof-of-Concept</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>RC</td>
<td>remote control</td>
</tr>
<tr>
<td>ROI</td>
<td>Return on Investment</td>
</tr>
<tr>
<td>Rx</td>
<td>Receiver</td>
</tr>
<tr>
<td>SAS</td>
<td>Signature Augmentation System</td>
</tr>
<tr>
<td>SDR</td>
<td>Software-Defined Radio</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kit</td>
</tr>
<tr>
<td>SR</td>
<td>Short Range</td>
</tr>
<tr>
<td>SRAM</td>
<td>Static Random Access Memory</td>
</tr>
<tr>
<td>Strato</td>
<td>Stratospheric</td>
</tr>
<tr>
<td>TUAV</td>
<td>Tactical UAV</td>
</tr>
<tr>
<td>TV</td>
<td>Television</td>
</tr>
<tr>
<td>Tx</td>
<td>Transmitter</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned Aerial Vehicle</td>
</tr>
<tr>
<td>UAS</td>
<td>Unmanned Aerial System</td>
</tr>
<tr>
<td>USA</td>
<td>United States of America</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>USD</td>
<td>United States Dollar</td>
</tr>
<tr>
<td>USRP</td>
<td>Universal Software Radio Peripheral</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
<tr>
<td>WEP</td>
<td>Wired Equivalent Privacy</td>
</tr>
<tr>
<td>WPA</td>
<td>WiFi Protected Access</td>
</tr>
<tr>
<td>WPA2</td>
<td>WiFi Protected Access 2</td>
</tr>
<tr>
<td>WR</td>
<td>Write</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
INTRODUCTION

As we have seen recently, UAVs are gaining more and more attention from the public. This can be explained by the increasing affordability of the technology for hobbyists and enthusiasts which allows the creation of innovative business models, as well as the occurrence of several severe incidents involving both military and civil UAVs [1]. These were reported by nearly every major news channel around the world, making UAVs appear almost every week in the news. Famous examples of incidents are the capture of a Lockheed Martin RQ-170 Sentinel by the Iranian forces and the live capture of a video feed sent by a U.S. Predator drone in Iraq [2] [3]. Also in the civil sector UAVs are used by criminals to smuggle drugs or simply by hobbyists who do not respect flight safety regulations and no-fly areas [4] [5] [6]. All these cases involve the security of UAVs. It is questionable whether the three key principles of IT Security: Availability, Integrity and Confidentiality, are maintained with regard to the UAV itself and the data related to it.

1.1 MOTIVATION

Due to several security issues revealed in recent investigations of cheap consumer UAVs it became clear that these cannot be considered secure [7]. However, in a professional environment, these cheap UAVs are rarely used and therefore the aforementioned studies are of little relevance to large enterprises and governmental bodies. The capabilities of UAVs used in the professional environment are significantly more advanced and accordingly the cost can be magnitudes higher. Moreover, as such professional UAVs are used for sensitive missions like surveillance and search and rescue, their security is highly relevant. What if an attacker could gain control over a police UAV during a surveillance mission, or capture an expensive UAV during a power line inspection? Obviously there is financial damage done to the enterprise but, more importantly, the mission is compromised. As no study has been conducted so far on the security of these professional UAVs, they are considered secure and used for the previously-mentioned use cases. However, it has not yet been proven that this hypothesis is true.

This Master Thesis will give an insight into the security of a professional UAV and state the security-related findings including a Proof-of-Concept (PoC) for a successful Man-in-the-Middle (MitM) attack.
and command injection under real world circumstances. Being also aware of security weaknesses in expensive professional UAVs may cause enterprises and governmental bodies to reconsider using such devices for their scenarios, as well as encourage manufacturers to put more effort into securing their products.

1.2 RESEARCH GOAL, RESEARCH QUESTIONS AND THEIR APPROACH

This section provides information on the research goal and the research questions to be answered during the Master Thesis. Additionally, the research approach for each of the raised questions is explained.

Given the wide-ranging literature review and extensive study on Unmanned Aerial System (UAS) technologies, the following research goal and research questions have been defined:

Exploit security vulnerabilities in the communication channels of a professional Unmanned Aerial System in order to raise awareness amongst manufacturers and the general public about existing vulnerabilities of UAS that are used for daily critical operations.

Research questions to achieve this goal are:

RQ-1 What is the current state-of-the-art with regard to the security of professional UAS?

A literature analysis and study of current projects can deliver the answer to this question. Therefore existing projects and research results need to be analyzed and their importance in relation to this research judged.

RQ-2 Which technologies are used within UAS to implement control channels and which attack vectors on those control channels can be identified?

Information about multiple UAS has to be aggregated and the common technologies have to be identified. Based on the underlying technologies, attack vectors can be listed.

RQ-3 What kind of security implementations are applied to the communication channels of the investigated UAS in order to protect them?

To answer this research question, the communication channels of the investigated UAS have to be determined and investigated in detail. Firstly, the technologies used need to be identified and secondly its security implementations pointed out.
RQ-4 Do the used technologies have security vulnerabilities and can those be exploited?

As research question three identified the technologies and their security implementations, we can now continue to find new vulnerabilities or exploit existing and known ones. As weaknesses can only be found once the system is understood properly, this research question involves studying the used technologies and their security implementations in great detail. During the different stages of research, several approaches are suggested and their effectiveness discussed. A PoC is created to prove the feasibility of exploitation.

RQ-5 What would be appropriate countermeasures to close the discovered vulnerabilities?

This research question is based on the findings of the previous research question. For the discovered security vulnerabilities, suggestions for appropriate countermeasures to avoid exploitation are given.

While conducting the research it transpired that some of the research questions had to be discarded as they would have led the focus of research away from the security of the UAV towards the security of the computer running the flight planning software. It can be assumed that, after successful infiltration of the Personal Computer (PC) running the flight programming software, including the possession of administrator privileges, relevant data can be changed as the attacker desires. However, the infiltration of PCs should not be the focus of this research, which is why these questions were set aside. To provide completeness, those discarded research questions and their approach, together with additional ideas for this thesis in connection with UAVs, can be found in Appendix A.

1.3 RESEARCH CONTRIBUTIONS

The following contributions are made during this Thesis:

1. We demonstrate the feasibility of a Man-in-the-Middle attack on XBee 868LP chips, which are widely used in all kinds of applications.

2. We prove that the professional UAS used in this thesis has vulnerabilities and is susceptible to command injection attacks.

3. By demonstrating the feasibility of such attacks, we raise awareness within the general public, the scientific community and UAS manufacturers that professional UAS should incorporate a higher level of security.
To the best of our knowledge these contributions are novel and have not previously been achieved. Related work will be presented during this thesis, as well as the boundaries of such.

1.4 ORGANIZATION OF THE THESIS

The following chapters are organized as follows: Chapter 2 provides an overview about related research and answers RQ-1. There has not yet been a lot of research on the security of professional UAVs, which further highlights the importance of this work. Chapter 3 gives a general introduction to the different terms and classifications. Moreover, RQ-2 is answered during this chapter by listing possibilities for control channels and investigating attack vectors of UAS. By introducing the investigated UAV and its communication channels in chapter 4, the answers to RQ-3 are given. Additionally, RQ-4 is answered as the vulnerabilities of each channel are explained, but is even further developed in chapter 5. This chapter allows insights into the actual exploitation of vulnerabilities, including a PoC, and additionally answers RQ-5 by suggesting countermeasures to avoid such malicious use in the end. Finally, this thesis closes with a conclusion and suggestions for future work in chapter 6.
RELATED RESEARCH

This chapter gives an overview on related research and provides background information for further reading. It therefore answers RQ-1.

RQ-1: What is the current state-of-the-art with regard to the security of professional UAS?

Firstly, recent drone hacks and warflying are described in section 2.1 and 2.2, respectively. Secondly, GPS Spoofing in section 2.3, WiFi & Bluetooth cracking in section 2.4, ZigBee and XBee in section 2.5 as well as KillerBee in section 2.6 are introduced to provide background knowledge for further reading.

2.1 RECENT DRONE HACKS

Recently there have been research projects targeted on the security of consumer UAVs. ProtoX, a 5x5 cm sized quadrocopter, has successfully been reverse engineered. Moreover, SkyJack and MalDrone obtained quite a lot of media attention.

It was recently shown that a small quadrocopter (ProtoX) could be reverse engineered with little technical equipment (just a development board and a logical analyzer) [8]. To communicate with the small model aircraft the original control was used. But instead of using the joysticks on the remote control and letting the signals be processed by the micro-controller of the remote control, the author desoldered the original micro-controller from the remote control and connected the development board to the output pins. By attaching a logical analyzer to the development board, which was connected to the PC, he was able to write a script sending control signals through the development board to the aircraft, bypassing the original microprocessor. Therefore it was proven that it is possible to reverse engineer the ProtoX drone protocol. The project can be found in [9].

SkyJack exploits the fact that an unencrypted WiFi communication channel between the smart phone, which is used to control the UAV, and the Parrot AR.Drone 2 is used. Due to the design of frames in the Institute of Electrical and Electronics Engineers (IEEE) 802.11 specification it is possible for an attacker to inject de-authentication packets to the network and disconnect the legitimate operator from the UAV. Afterwards it is possible to connect as an attacker and gain control
over the device. As there is no encryption in place, this attack is feasible and easy to perform. However, the author is not only disconnecting users from their UAVs and connecting his own control device, but also uses his own drone equipped with an additional WiFi card to forward control information to hacked UAVs [10], thereby extending the range to hijack somebody else’s UAV.

Another project is called MalDrone, which is described as payload rather than an exploit by the author and requires a connection to the targeted UAV. This is only the case if the channel is not encrypted or can be compromised in any other manner. If an encrypted channel is in place, the malware cannot be installed on the device. A further restriction is that the malware is only applicable to a Parrot AR.Drone 2. If the attack were successful and could be placed on the flight computer of the UAV, it would establish a connection back to the attacker and allow him to control the device [11].

2.2 WarFlying

Presented at the DefCon 19 conference, the "Rabbit-Hole" is an air surveillance platform which allows the attacker to spy on people beneath. Contrary to the previously-presented projects, the UAV itself is used as a medium for security sensitive tasks in this scenario rather than being investigated itself.

It is possible to attach many kinds of wireless communication technology to the UAV, which is why it can also spy on and compromise those channels. Examples are WiFi, Bluetooth and Global System for Mobile Communications (GSM) communication (IMSI-Catcher) [12]. Moreover, cameras and microphones could be attached to perform further privacy invasion.

2.3 GPS Spoofing

As most UAVs rely on longitude and altitude, as well as accurate time information provided by the Global Positioning System (GPS), GPS Spoofing poses an immense threat to UAVs. It has been shown in [13] that UAVs and other vehicles relying on GPS can be tricked into believing that they are at a different position [14] [15] [16]. There exist several GPS projects, e.g. from the United States of America (USA), European Union (EU), Russia and China, but all of them follow the same separation between two types:

- Civilian GPS
- Military GPS
While the military GPS system uses authenticated and encrypted signals to transfer data from the satellites towards the receivers, civilian GPS does not. It was never intended to be used for critical services which would require such security measures and is therefore not designed to be run in such applications. However, as GPS became more popular to use for all kinds of services, applications with security sensitive requirements also began to use civilian GPS to determine the position as well as to get accurate time information [17]. Since no security measures are in place in a civilian GPS system, the receiver might receive forged data and has no ability to prove whether the payload is correct and was sent by an authorized entity. Unfortunately it is not possible for a civilian entity to use the military GPS. As a result, civilian applications need to use the civilian GPS, which is vulnerable. The requirements of GPS spoofing are investigated in [18]. The authors state that:

"The attacker can delay signals or send them prematurely [...] He can modify the content of received GPS signals or arbitrarily generate the spoofing signals [...] using the public GPS parameters (e. g., by using a GPS signal generator)" ([18])

This refers to the civilian GPS system. Encryption and authentication are implemented for the military GPS, hence no modification of payload is possible. However, delays can still be introduced by an attacker. The attack on military GPS is therefore known as selective-delay attack but can be considered as having a lower impact than GPS Spoofing. The content of military GPS packets cannot be modified and therefore the GPS receivers cannot be tricked into believing they are at a wrong position.

It has been proven by the previously-mentioned research that most civilian GPS receivers do not check whether the content of messages makes sense or not. For instance, a system could become suspicious if a previous GPS lock was performed far away from the current GPS position and the assumption that the newly acquired GPS signal is spoofed could be raised. To recognize such malicious behavior several countermeasures were proposed in [19]. These include:

- Monitor the absolute GPS signal strength
- Monitor the relative GPS signal strength
- Monitor the signal strength of each received satellite signal
- Monitor satellite identification codes and number of satellite signals
- Check the time intervals
• Do a time comparison
• Perform a sanity check

Obviously, all of these countermeasures require logic on the receiver side. This implies not only additional implementation work for the manufacturers, but might also affect the performance and therefore the internal hardware might become more expensive.

A major incident, which is related to GPS Spoofing, was the capture of a Lockheed Martin RQ-170 Sentinel by the Iranian forces in December 2011 [2]. Although the United States asked for a return of the UAV, the Iranian forces denied the request and used the captured UAV instead for reverse engineering purposes [20]. This incident shows how important it is to consider the security of UAVs when deploying them. If third parties can fool the flight computer of a UAV, the integrity of the entire mission is compromised.

2.4 WIFI & BLUETOOTH CRACKING

When first introduced in 1997, wireless communication used an encryption standard called Wired Equivalent Privacy (WEP) [21]. However, this encryption standard has vulnerabilities which can be exploited to expose the encryption key of the algorithm [22]. Newer encryption standards like WiFi Protected Access (WPA) and WiFi Protected Access 2 (WPA2) impose higher levels of security, but can still be cracked if certain conditions are met [23].

During this research, WiFi cracking has been used to gain access to one of the communication channels of the investigated UAV. An in-depth description of the performed attack, mentioning also the specific requirements in this scenario, can be found in section 5.1.2.

Additionally, Bluetooth can be used to exchange control signals between Model Aircraft, Drone, Micro UAV and remote controls (RCs) (mostly smart phones). As has been shown several times, Bluetooth has security vulnerabilities and is susceptible to multiple attacks [24]. Especially the newer standard Bluetooth Low Energy (BLE) 4.0 is used in combination with a Model Aircraft and Drone. However, also here security vulnerabilities and their related attack vectors reoccur [25].

2.5 ZIGBEE & XBEE

ZigBee is based on the international standard 802.15.4 and was developed by the ZigBee Alliance [26]. It is a suite of protocols to allow communication through low-power and low-cost devices. To extend
the transmission range, ZigBee is adding mesh networking functionality on top of the 802.15.4 standard, whereby single messages are forwarded through the network to its destination node. Depending on the frequency band used, transmission rates can vary. Typically it ranges from 20kbit/s to 250kbit/s. As the devices require low power they are mostly used as embedded devices.

On the other hand, XBee is a brand name and product line developed by Digi International. They also use the 802.15.4 standard underneath, but build their own protocol suite on top. XBee devices can also be flashed with ZigBee-compliant firmware, thereby losing the advantages of XBee implementations but achieving connectivity with other ZigBee-compliant devices. Digi describes in several documents the use of proprietary modulation, as well as proprietary multipoint and digimesh protocols [27]. The range of XBee 868LP devices, using the proprietary multipoint protocol, is up to 40 km, according to the manufacturer [28]. By using XBee devices there is no need to have a Coordinator and End-Device configured in the network. Moreover, the XBee 868LP chip is using Listen Before Talk (LBT) and Adaptive Frequency Agility (AFA) to determine if a channel is free and data can be sent. This allows multiple different networks to exist, as data transfer is done dynamically.

2.6 KillerBee

KillerBee is a penetration testing framework for ZigBee and 802.15.4-based networks. It has several features, e.g. Denial of Service (DoS) attacks on nodes by establishing recurring connections, sniffing and dumping traffic, replay previously recorded packets, etc. However, specific hardware is required to use the framework. It currently works with three different Universal Serial Bus (USB) dongles, all of them equipped with a 2.4 Gigahertz (GHz) radio chip, which makes it unsuitable for the 868 Megahertz (MHz) band. Since the firmware to use the framework is only available for these three dongles, it was not possible to test if this framework would also be able to interfere with XBee 868LP devices [29] [30].

Killerbee can also be combined with the GoodFET hardware debug interface tool. According to [29], GoodFET exploits a common vulnerability in Ember and Texas Instrument chips to extract the Random Access Memory (RAM) even when the chip is locked. This allows the user of both devices to first sniff an encrypted packet and then try to apply the correct key to decrypt the packet by trying all possible values contained in the RAM.
This chapter provides a definition and classification of UAVs. Moreover, RQ-2 is answered by explaining the different possibilities of control channels and attack vectors.

RQ-2: Which technologies are used within UAS to implement control channels and which attack vectors on those control channels can be identified?

The definition is introduced in section 3.1 and a classification is given in section 3.2. Afterwards, the control channels are explained in section 3.3 and the attack vectors are described in section 3.4. During this research two logical attacks are performed on two different communication channels. The theory behind logical attacks is described in detail in section 3.4.2.

3.1 Definition

UAS, UAV and drone are terms which are sometimes used interchangeably to talk about the same thing. To assure that everything is understood properly a definition is required. For this purpose the definition of Reg Austin is used throughout this work:

"The system [UAS] comprises a number of sub-systems which include the aircraft (often referred to as a UAV or unmanned air vehicle), its payloads, the control station(s) (and, often, other remote stations), aircraft launch and recovery sub-systems where applicable, support sub-systems, communication sub-systems, transport sub-systems, etc." [31, p. 3]

The main difference between a UAS and normal manned systems is that no aircrew is required. This does not mean that there is no pilot at all, as the piloting can (usually) be done from a remote location (depending on the communication equipment on board). All the other elements and components of the system find their equivalent in both manned and unmanned systems. According to the definition above, the UAV is (the flying) part of the UAS.

A model aircraft is an aircraft which is only allowed to stay within the sight of the operator and is directly controlled using a remote control. Although this is also the case for a drone, the additional feature
of having a flight computer is relevant. This flight computer allows the operator to pre-program flight paths such that the drone is able to perform an autonomous flight afterwards. Having this definition in mind, a model aircraft which is upgraded with a flight computer can be considered as a drone.

But how can a line be drawn between a drone and a UAV, as the UAV also takes pre-programmed flight paths and can be controlled remotely within the distance of the remote control? According to Austin, a UAV possesses some kind of "intelligence", while a drone is somehow dumb, meaning it cannot make any decisions based on changing environmental factors (e.g. no automated coming home feature when the battery level is falling below a certain threshold). The following figure was produced based on Austin’s definition and illustrates the boundaries of each term. It is evident that the complexity increases as the terms encompass more technology and refer to increased intelligence of the device.

However, this definition is unsatisfactory as it is hard to draw a clear line between a drone and a UAV. Even very simple systems nowadays include a flight computer and also possess some kind of intelligence (which would be a UAV according to Austin). There is currently extensive research performed on autonomous vehicles and UAVs which can adapt to environment variables (e.g. avoidance of moving objects or altering flight paths due to obstacles) [32] [33] [34]. These devices have capabilities which are far more intelligent than reacting to a predefined set of possible situations with an appropriate action. The central problem that relates to the separation within the definition is described as: What can be considered intelligent?
There are several types of UAVs in existence which are used for different purposes. To gain an overview over the available devices and their attributes, two classifications will be compared in this section.

### 3.2 EUROUVS

EUROUVS (a lobby organization of UAV manufacturers with mainly military purpose) introduced a classification in 2006, dividing UAVs in four main categories [35], originally taken from UVS International [36]. The division was made according to attributes which characterize each group based on "Maximum Take Off Weight (kg)", "Maximum Flight Altitude (m)", "Endurance (hours)" and "Data Link Range (km)". It can be inferred that the attributes are in correlation with each other. A larger aircraft can carry more equipment and fuel, which leads to extension in range as well as endurance. Table 3.1, from [35], shows the classification.

#### Table 3.1: UAV Classification - EUROUVS

<table>
<thead>
<tr>
<th>Category (Average)</th>
<th>Maximum Take Off Weight (kg)</th>
<th>Maximum Flight Altitude (m)</th>
<th>Endurance (hours)</th>
<th>Data Link Range (km)</th>
<th>Missions</th>
<th>Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Micro/Mini UAVs</td>
<td>1.10</td>
<td>250</td>
<td>1</td>
<td>&lt; 10</td>
<td>Scouting, NBC sampling, surveillance, remote buildings</td>
<td>Black Widow, MicroStar, Microbat, FireCaptor, QuantumCopter, Mosquito, Insect, MZB</td>
</tr>
<tr>
<td>Mini</td>
<td>&lt; 30</td>
<td>150–300</td>
<td>&lt; 2</td>
<td>&lt; 10</td>
<td>Film and broadcast industries, agriculture, pollution measurements, surveillance, remote buildings, communication relay, and EW</td>
<td>Mikado, Albatros, Tracker, Dragonfly, Raven, Hawker 8, X-300, Caddell F1502, epoxy, R–Max and R–15, Robocopter, HeliOscop, 19–2008</td>
</tr>
<tr>
<td>Sectoral UAVs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Close Range (SR)</td>
<td>150</td>
<td>3,600</td>
<td>2–4</td>
<td>30–80</td>
<td>KS1, mine detection, search &amp; rescue, EW</td>
<td>Phantom, Phantom, Copter A, Mikado, RobotCopter 250, Hawker, CopterOpt, Aerial and Agricultural Mike</td>
</tr>
<tr>
<td>Short Range (SR)</td>
<td>200</td>
<td>3,600</td>
<td>3–6</td>
<td>20–20</td>
<td>B3A, B3E, E1, mine detection, X-ray, EW</td>
<td>Scan-E (50), Luna, Silverfox, Ethereal, F-droid, R–Max Agri/Phot, Hawker, Hawker, phantom, Goldwind 500, flight, Aeronaut</td>
</tr>
<tr>
<td>Medium Range (MR)</td>
<td>150–500</td>
<td>3,000–5,000</td>
<td>5–10</td>
<td>70–200</td>
<td>B3A, B3E, E1, mine detection, EW</td>
<td>Hunter, B–31, Predator, Sniper, Vulture, Arrow 95, Street 500, R–300, Eagle Eye, Alpha, Predator, Shadow 200–600</td>
</tr>
<tr>
<td>Long Range (LR)</td>
<td>150–2500</td>
<td>5,000–8,000</td>
<td>15</td>
<td>200–500</td>
<td>B3O, B3D, communications relay</td>
<td>Hunter, Vigilant, 802, Shadow 200</td>
</tr>
<tr>
<td>Endurance (EN)</td>
<td>500–1,500</td>
<td>5,000–8,000</td>
<td>22–24</td>
<td>&gt; 500</td>
<td>B3A, B3E, E1, communications relay, EW</td>
<td>Aerosonde, Cachalot I, Shadow 200, Shadow 1, Hermes 450/450T/700</td>
</tr>
<tr>
<td>Medium Altitude, Long Endurance (MALE)</td>
<td>3,000–1,500</td>
<td>5,000–8,000</td>
<td>24–48</td>
<td>&gt; 1,000</td>
<td>B3A, B3E, EW weapons delivery, communication relay, EW</td>
<td>Shadow 30, Hermes 1500, Hermes 7, B–31, Predator, Predator A, Shadow 1–2, Darkstar, E–Hunter, Orenda, Globmar, Sunbird, Silverware, Reaper, ScanEagle, Shadow 200</td>
</tr>
<tr>
<td>Strategic UAVs</td>
<td>2,500–12,500</td>
<td>15,000–20,000</td>
<td>24–48</td>
<td>&gt; 2,000</td>
<td>B3A, B3E, EW communications relay, EW</td>
<td>Global Hawk, Reaper, Centaur, Boeing, Hermes, Predator B, B–31, SilverEagle, EuroHawk, Harpy, ScanEagle, Global Observer, Predator A, Shadow 200, Global Hawk, Shadow 200</td>
</tr>
<tr>
<td>Special Task UAVs</td>
<td>250</td>
<td>3,000–8,000</td>
<td>3–4</td>
<td>300</td>
<td>Anti-radar, anti-spy, anti-aircraft, anti-structure</td>
<td>MAL, Hycon, Loral, Mercedes</td>
</tr>
<tr>
<td>Energy (ECC)</td>
<td>25s</td>
<td>50–1,000</td>
<td>&lt; 4</td>
<td>0–500</td>
<td>Aerial andusal daughter</td>
<td>Flyte, MALD, Rakitka, V11L, Chukar</td>
</tr>
<tr>
<td>Stratigraphic (TMD)</td>
<td>200</td>
<td>20,000–50,000</td>
<td>&gt; 48</td>
<td>&gt; 2,000</td>
<td>-</td>
<td>Neptune</td>
</tr>
<tr>
<td>Low-stratospheric (EOS)</td>
<td>750</td>
<td>&gt; 34,000</td>
<td>TBD</td>
<td>TBD</td>
<td>-</td>
<td>Marev 500, MAX–1</td>
</tr>
</tbody>
</table>

*Source [35]*
The first category consists of Micro/Mini UAVs which are small and lightweight. Micro UAVs are typically used for scouting and surveillance missions in highly dense areas and inside buildings. A typical example is the 'PD-100 Black Hornet’ from Prox Dynamics [37]. However, this classification was proposed before individuals and companies discovered UAVs for a variety of use cases, which is why this segment seems to be rather briefly covered. As prices for these UAVs are getting lower, this category is already, and will be in future, most challenging for governmental bodies, since regulations are required. UAVs in other categories are usually not available for purchase to consumers or even companies, but only for military and governmental use. This is due to the fact that most of them are considered as weapons [38]. Additionally the costs are usually extremely high, making them only affordable for military use [39].

The second category describes Tactical UAVs with a weight between 150-1500 kilograms and an altitude between 3000-8000 meters. Endurance and data-link connection are increased compared to the preceding group. More specified subcategories are Close Range (CR), Short Range (SR), Medium Range (MR), Long Range (LR), Endurance (EN) and Medium Altitude Long Endurance (MALE). As the naming already suggests, the difference is defined by looking at the endurance of these UAVs. Examples are the RoboCopter 300 [40] as a CR UAV and the Dominator [41] as a MALE UAV.

The third category lists Strategic UAVs. Those are characterized by an extreme increase in weight, altitude, endurance and data link range. A famous example is the Global Hawk [42]. One unit had an average price of USD 222.69 M in 2012 [43].

The fourth category is called Special Task UAVs. It contains the subgroups of Lethal (LET) and Decoys (DECs) as well as Stratospheric (Strato) and Exo-Stratospheric (EXO) UAVs. An example of a DEC UAV is the Miniature Air Launched Decoy (MALD) missile. The missile carries a Signature Augmentation System (SAS), which allows the missile to imitate any aircraft and therefore confuse the enemy [44].

3.2.2 Austin

Austin offers a slightly different categorization of UAVs [31]. A summary can be found in Table 3.2.
### Table 3.2: UAV Classification - Austin (based on [31])

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nano Aerial Vehicles (NAVs)</td>
<td>Size of sycamore seeds. Used in swarms to confuse radar or eventually for ultra-short range surveillance, if sub systems can be designed that small.</td>
</tr>
<tr>
<td>Micro Aerial Vehicle (MAV)</td>
<td>Wing size: &lt;150 mm. Mainly used in urban environments. Especially useful within buildings.</td>
</tr>
<tr>
<td>Mini Unmanned Aerial Vehicle (MUAV)</td>
<td>Weight: &lt;20 kg Range: &lt;30 km</td>
</tr>
<tr>
<td>Close Range (CR) UAV</td>
<td>Range: &lt;100 km Used for military as well as civilian purposes. Examples are crop spraying and power line inspections.</td>
</tr>
<tr>
<td>Tactical UAV (TUAV)</td>
<td>Range: 100-300 km. Typically operated by land and naval forces.</td>
</tr>
<tr>
<td>Medium Altitude Long Endurance (MALE)</td>
<td>Altitude: 5,000-15,000 m Endurance: &lt;24 hours Usually operated from fixed bases.</td>
</tr>
<tr>
<td>High Altitude Long Endurance (HALE)</td>
<td>Altitude: &gt;15,000 m Endurance: &gt;24 hr.</td>
</tr>
</tbody>
</table>

### 3.2.3 Comparison

As can be seen, both definitions mainly correspond. However, there are minor differences. One difference between the two classifications lies in the fact that EUROUVS specifies the Technical UAVs in greater details, listing also SR, MR, LR and EN UAVs. A second difference can be seen in the definition of Special Tasks UAVs in the EUROUVS paper, while Austin adds a Nano UVS category, which is not taken into consideration by EUROUVS.

Looking at the actual market situation and available models for purchase, it can be concluded that consumer and professional UAVs
are mostly in the range of Micro, Mini and Close-Range UAVs. UAVs with wider range or the option to carry heavier payload are usually utilized by the military (also because the costs are much higher). The listed classifications offer the possibility to sort UAVs based on their criteria. However, there might still be a difference within certain categories. Moreover, some UAVs are not easily classified and allocated to a category. Depending on the purpose of the device, it might have a very long range, while still being quite light (solar glider), making a classification within the above-mentioned schemes difficult or even impossible. A solution could be to assign a UAV to multiple groups.

3.2.4 Consumer & Professional UAV

Throughout this work, the terms consumer UAV and professional UAV are used to differentiate between two “virtual” classes of UAVs within the Mini UAV section, mainly referring to the targeted audience which is supposed to use the devices. Characteristics of a consumer UAV are a relatively cheap price, low lifting power and limited range. The additional equipment is also more targeted towards hobbies and self-building projects. On the other hand, a professional UAV usually has a much higher price, improved endurance and lifting power. Moreover, additional equipment for such UAVs is made for specific use cases. To give an example, consumer UAVs are mostly used by hobbyists to take amateur videos and fly for fun, while professional UAVs are used by companies or governmental bodies to generate business value (e.g. power line inspection). During this research a professional UAV is investigated.

3.3 Control Channel

To control a UAV, technically, all different kinds of communication technologies can be used. Essentially, it is data which needs to be transferred from the RC to the UAV (and eventually back). Different forms of electromagnetic radiation can be used to serve this purpose. An extensive overview can be found in [45, p. 314] while an overview of the most relevant ones is given here.

3.3.1 Infrared

Infrared (IR) is a technology typically used for remote controls of TVs and other peripheral devices in a household, as it is easy to use and cheap. Due to the physical limitations of the technology, “[t]here must be a line of sight between the transmitter (lightsource) and the receiver (lightdetector). Any obstacles between transmitter and receiver will prevent from [sic] correct reception.” ([46]) This limits the ability to use Infrared light powered remote controls for the purpose of
controlling a UAV tremendously. A UAV is usually operated with a greater distance than 10 meters from the RC and also obstacles in between the RC and the UAV cannot always be avoided. As even rain and sunlight can be a possible problem for IR communication to work, this technology is not recommended to be used in a typical UAS setting outside buildings.

3.3.2 Radio Waves

Radio waves are another type of electromagnetic radiation with longer wavelength than Infrared light. The frequency bands are grouped due to their wavelength characteristics. The usage of those bands is regulated by the Federal Communications Commission (FCC) in the United States and the European Communications Committee (ECC) in the EU. Table 3.3 shows the frequency bands as defined by the IEEE. For this work, the most important bands are the UHF band (especially the frequencies around 868 MHz) and the S band (especially the frequencies around 2.4 GHz).

As the frequency bands are regulated, the usage of specific bands is prohibited. Additionally, some frequencies are licensed to private entities, which allows them to use the granted frequencies exclusively [47]. However, there are free to use frequency spectra available which can also be used for remote controls in UAVs. In [48], the authors provide an overview of the different frequency bands and the applications used on them in the European Union.
Table 3.3: Standard radar frequency letter band nomenclature

<table>
<thead>
<tr>
<th>Band designation</th>
<th>Nominal frequency range</th>
<th>Specific frequency ranges for radar based on ITU assignments (see Notes 1, 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Region 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Region 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Region 3</td>
</tr>
<tr>
<td>HF</td>
<td>3–30 MHz</td>
<td>(Note 3)</td>
</tr>
<tr>
<td>VHF</td>
<td>30–300 MHz</td>
<td>None</td>
</tr>
<tr>
<td>UHF</td>
<td>300–1000 MHz (Note 5)</td>
<td>420–450 MHz (Note 4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>890–942 MHz (Note 6)</td>
</tr>
<tr>
<td>L</td>
<td>1–2 GHz</td>
<td>1215–1400 MHz</td>
</tr>
<tr>
<td>S</td>
<td>2–4 GHz</td>
<td>2300–2500 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2700–3600 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2700–3700 MHz</td>
</tr>
<tr>
<td>C</td>
<td>4–8 GHz</td>
<td>4200–4400 MHz (Note 7)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5250–5850 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5250–5925 MHz</td>
</tr>
<tr>
<td>X</td>
<td>8–12 GHz</td>
<td>8.5–10.68 GHz</td>
</tr>
<tr>
<td>Ku</td>
<td>12–18 GHz</td>
<td>13.4–14 GHz</td>
</tr>
<tr>
<td>K</td>
<td>18–27 GHz</td>
<td>24.05–24.25 GHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>24.05–24.25 GHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>24.65–24.75 GHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Note 8)</td>
</tr>
<tr>
<td>Ka</td>
<td>27–40 GHz</td>
<td>24.05–24.25 GHz</td>
</tr>
<tr>
<td>V</td>
<td>40–75 GHz</td>
<td>33.4–36 GHz</td>
</tr>
<tr>
<td>W</td>
<td>75–110 GHz</td>
<td>59–64 GHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>76–81 GHz</td>
</tr>
<tr>
<td>mm (Note 9)</td>
<td>110–300 GHz</td>
<td>92–100 GHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>126–142 GHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>144–149 GHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>231–235 GHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>235–248 GHz</td>
</tr>
</tbody>
</table>

Source: [49]

Apart from using proprietary protocols on these radio frequencies, standardized technology like WiFi or Bluetooth is used to control UAVs (which work on the same, unregulated frequencies). Since the range of these technologies is limited due to their wavelength, they are mostly used to connect a computer (PC, smart phone or tablet) to the UAV in order to pre-program the flight computer of the UAV conveniently. For controlling purposes, these technologies are only used on smaller UAVs with a very limited range (Micro/Mini UAV). An obvious advantage is the low cost approach as the smart phone of the customer can be used as a remote control. Therefore no additional RC needs to be shipped, which, together with the use of standardized protocols, increases the profit margin of the manufacturer. A typical example of a widely distributed UAV using this combination is the Parrot AR.Drone 2 [50].
3.4 ATTACK VECTORS

There are multiple ways to gain control over a UAV which is originally controlled by somebody else. To give a better overview, this section is grouped in physical and logical attacks. While physical attacks require physical contact with the UAV itself, the logical attacks are targeted on the data used by the UAV to figure out its position, perform steering, initiate emergency landing sequences, etc. This section focuses on Micro/Mini UAVs, like the ones a company or an individual would be able to purchase.

3.4.1 Physical

Shooting a UAV out of the sky might seem the most obvious and easiest way to take it down (Ground to Air Missile (GTAM)). As can be seen in history, shooting down enemy aerial vehicles (manned or unmanned) is the most used tactic so far. As manned aircraft are steered by the pilot, logical attacks might irritate (manipulating the displayed data) but not actually cause a crash if the pilot does not rely solely on them but also on his human abilities and manual control to steer the aircraft. However, while this attack might work for war zones and open fields, it is not possible to use missiles in crowded areas or for consumer and professional UAVs (in a civil context). The cost-benefit ratio would also prevent institutions employing such systems, as missiles and their launch pads are usually quite expensive, while consumer and professional UAVs are mostly cheaper. Moreover, it lies in the nature of UAVs that they are moving objects, which makes them a hard target to shoot down with non-following systems (like a rifle).

Another approach to take down a UAV could be to use a second UAV ("Interceptor Drone"). Flying into the invading UAV would have the same impact as a GTAM (crash) which can be used to protect airfields or other areas where the damage to third parties can be excluded, but it is not acceptable for areas where innocent third parties could be harmed. Instead the second UAV could carry some kind of tool which allows it to bring the invading UAV down safely (causing harm to its functionality but without involving third parties). Ideas can be to equip the second UAV with a harpoon to shoot at the invading UAV, throwing a net over it [51] or releasing a wire into its propellers [52] in order to drag it away afterwards. This is illustrated in Figure 3.2 and 3.3.

Other physical attack vectors, such as wind, magnetism or high-powered microwave beams can also be considered. As high-powered microwave beams require sufficient funds for the equipment as well
as knowledge to develop, they might not be applicable for the private sector (also considering the legislation issues to use them). Instead they are used for military purposes [53]. However, leaving aside the financial and legislative issues, they seem to fit in the scenario described in this work. A UAV with appropriate power sources and transport capabilities could emit a high-powered microwave to an intruding UAV and destroy the inner electrical circuits. Instead of using a military UAV it might be possible to mount such a weapon under a smaller and lower cost UAV (like an Octocopter). Assuming this scenario as technically possible, this is a very plausible way to take down an intruding UAV. An obvious disadvantage of this approach is that the intruding UAV will be destroyed and crash, possibly harming innocent parties beneath it.

3.4.2 Logical

UAVs use several technologies to communicate with the operator and to determine their flight parameters like speed, altitude, etc. Since UAVs are not manned, they need solely to rely on those parameters and on the commands sent by the base station. If one could influence one or multiple of these parameters, or even change the data sent through the control link of the UAV, it might react to the attacker’s commands. This is a very serious issue, as total control can be lost due to one of these attacks.

Naturally, the amount of logical attack vectors is limited to the systems used by the UAV. In [54] the authors discuss possible UAV vulnerabilities using a risk assessment approach. One requirement for a legitimate use is that there always needs to be a control channel to command the UAV in case of mission modifications, even when the missions are pre-programmed in the UAV’s flight computer (also stated in the new proposal for UAV regulation in the USA [55]). The control channel can be established through different kind of radio communication technologies, discussed in section 3.3. Moreover, every UAV which is able to steer on its own (using a flight computer
with pre-programmed flight paths) needs to determine its flight parameters. Usually GPS is used to serve this purpose, as GPS provides the UAV with latitude and longitude signals as well as accurate time and altitude information. Additionally on-board navigational systems like the Inertial Navigation System (INS) are commonly used as well [56] [57].

3.4.2.1 Control Channel

Sending forged signals (Open channel/Reverse engineering): It is possible to send forged data to the UAV or to the RC if we consider an open channel (unencrypted WiFi network or simple radio communication) without having an application layer encryption nor authentication mechanisms in place. Since both devices cannot verify the received data, the correctness is assumed. However, this is a critical flaw. Forged signals will also assumed to be correct and the UAV, as well as the RC, will process these signals without differentiating them from the original ones. This leads to gaining control over the UAV or to displaying false information on the RC. If proprietary protocols are used to communicate with a UAV, they should also use proper encryption as well as authentication. Just to alter the standard protocols without making use of these crucial security features is called security by obscurity and can possibly be reverse engineered. A problem with this attack can be that the UAV will still listen to commands sent by the genuine user. This might lead to unexpected behavior of the UAV. Also the RC will still receive data from the UAV, which might lead to data values jumping on the display of the RC from one value to another, depending on the signal received at that moment. The exploitation of this attack vector was done during this work and can be found in section 5.2.1 of this work.

Denial of Service: Since wireless communication is used to control a UAV, it lies in the nature of the technology that it is hard to protect (as everybody who is in range can receive the signals and perform computations). Moreover, wireless communication is based on data sent on specific channels. Both the receiving and the sending device need to use the same channel in order to be able to communicate. If the adversary knows the channels used (standard communication technology or known frequencies for public use), he could possibly flood these channels with data - preventing the legitimate signals from being received by the UAV.

Another possibility within a WiFi network would be to send de-authentication packets in order to prevent a connection. Both attacks would lead to termination of communication and are therefore called Denial of Service attacks.
Replay Attack: Even if data streams are encrypted it can be possible to record signals which were sent before and to replay these towards the target. This would show an effect in situations when the protocol does not use cryptographic nonces to verify the validity of messages. Simply by looking at the reactions of the UAV, the adversary could tell which recorded message matches which command. Collecting all possible commands in this manner would allow him to gain control over the UAV. Similar to the attack of sending forged signals on an open channel, the UAV will also still listen to the genuine user, which might make controlling the device hard.

During this research all of the aforementioned attacks on the Control Channels were tested. Their application can be found in chapter 5.

3.4.2.2 Global Positioning System (GPS)

Denial of Service: GPS Jamming refers to Denial of Service, but on GPS frequencies. The channels are flooded with garbage data to avoid the legitimate signals passing through. Devices having this functionality can easily be bought for around USD 50. Obviously the range of these cheap devices is limited, but professional equipment can reach much further, allowing the attacker to jam huge areas. It should be added, that the use of these devices is illegal in most countries (including the European Union) [58] [59] [60]. The impact of a GPS Jammer and the consequences of its abuse can be seen in [61]. A more general work on the threats of GPS Jamming can be found in [62].

Spoofing: While jamming aims to disrupt all communication, spoofing instead forges signals and introduces them, usually stronger or in higher frequency of appearance, to the channel. As a result, the victim might use the forged signals instead of the original ones as an input for calculations. As GPS is used to determine one’s position, forged GPS signals allow the attacker to trick the victim into believing he needs to correct the steering to stay on the correct route, while in reality the vehicle is following the course of the adversary. Essentially GPS-Spoofed signals are suggesting a wrong position. More details can be found in section 2.3.

3.4.2.3 Video Link

Depending on the link used to transmit video signals to the control station or additional third parties, it might be possible to read the video feed and display it on an attacker’s device. This happened with a live video feed of a U.S. Predator UAV in Iraq [3]. As for the control channel communication, the manufacturer of the UAV can choose between different products using different frequencies (in unregulated bands) to transmit the video signal. Usually video transmission
requires quite a large bandwidth, which is why higher frequencies could be used. However, a higher frequency implies a lower range of the signal which is a downside. Therefore the data might be split and sent through multiple channels at the same time using lower frequencies. Accordingly the receiver needs to be able to listen to all used channels at the same time in order to reconstruct the data.

3.4.2.4 Software for Flight Planning

Most of the UAVs are flown using pre-programmed flight paths instead of a manual remote control. The RC usually serves as a backup (also because it is legally required), but the flights are conducted autonomously by the flight computer inside the UAV. Before a mission starts the data is written from a computer running the flight planning software to the flight computer. The computer can be a common laptop, tablet or smart phone. In some configurations the software is connected to the flight computer the whole time continuously to perform mission changes in real time. If the adversary could introduce malware to the computer running the flight planning software, he might be able to reprogram the flight computer and to control the UAV through the software. In cases when the computer is in possession of an Internet connection the attacker could gain live control over the UAV. If no Internet connection is available during the flight, but at any time before the flight computer is programmed, the attacker could still transmit his flight coordinates to the infected machine beforehand and upload this data upon next connection to the UAV. This attack is difficult to detect as it focuses on a part of the UAS which is not directly involved in a flight. Malware has already been found on base stations controlling UAVs, although it is believed that this was a general key-logger, not specifically targeted on the aforementioned goals [63]. The discarded research questions in Appendix A are focusing on this issue.
THE INVESTIGATED UAV

This chapter reveals the security implementations on the used communication channels and accordingly answers RQ-3. Moreover this chapter introduces different approaches on how to compromise the different communication channels. Since this relates to vulnerabilities it answers RQ-4, while chapter 5 will present the exploitation and therefore further develop this research question.

RQ-3: What kind of security implementations are applied to the communication channels of the investigated UAV in order to protect them?

RQ-4: Do the used technologies have security vulnerabilities and can those be exploited?

The setup is explained in section 4.1 to allow a thorough understanding of the used equipment. Due to a Non-Disclosure Agreement (NDA) the investigated UAV is not identified throughout this thesis. However, the system is sufficiently described to understand the underlying security problems. Afterwards, the WiFi link and the telemetry link are described in section 4.2 and 4.3, respectively. The manual remote control link and the video link follow in section 4.4 and 4.5. Finally, the GPS link is presented in section 4.6 and the chapter closes in section 4.7 with concluding remarks.

4.1 SETUP

To conduct a research on UAVs, a third party company kindly made one of their UAVs available. The UAV has eight rotors on four arms and is therefore an Octocopter. It can carry up to 2.9 kg and has a flight time of approximately 35 minutes. It is sold for around € 25,000 to customers using it for different purposes. Examples are power line inspections, agricultural applications, aerial photography and filming as well as the use of law enforcement entities to perform surveillance.

The investigated system consists of the following parts:

- The investigated UAV
- Remote Control (Graupner SJ MX-20)
- Telemetry Equipment (Monitor, Video receiver, WiFi and XBee Radios)
The UAV can be controlled through two different means. The first option is the manual remote control, employing a 2.4 GHz link to the UAV and running a proprietary protocol. The second option is to connect a tablet with the proprietary flight planner to the WiFi link of the telemetry box. From there on the signal is forwarded to the UAV through an XBee communication channel using the 868 MHz frequency. As radio waves can travel longer with lower frequency, the XBee link reaches much further than the 2.4 GHz manual RC link.

The manufacturer of the XBee chip used in this research claims that the chip has a range of up to 4 km while delivering a bandwidth of 50 Kbps. All communication links are displayed in Figure 4.1.

### 4.2 WIFI LINK

The Wifi link is used to connect a tablet, running the proprietary flight planning software, to the telemetry box, which is physically attached to the RC but not logically connected. The data from the flight planning software is therefore first transmitted through the WiFi link to the telemetry box and from there on forwarded using the telemetry link (XBee). An investigation revealed that the WiFi link uses WEP as encryption technology. It is therefore vulnerable and can be cracked.

Since the password of the WEP network opened by the telemetry box was not known, WEP cracking needed to be performed in order to connect a device and to be able to send data to the flight computer. The attack on the WiFi network and the resulting attack scenarios on the UAS can be found in section 5.1.

### 4.3 TELEMETRY LINK

The telemetry link uses XBee, a chip manufactured by the company Digi International. XBee chips are widely distributed and can be found in all kinds of use cases due to their low price tag.
4.3.1 XBee Experiments Setup

To test the functionality of the XBee chips, multiple experiments have been performed. The setup of these experiments is described in this section.

Required hardware:

- 1x Laptop with Windows 7+ with XCTU installed (Slave)
- 1x Laptop with Windows 7+ with XCTU and Serial Port Monitor installed (Master)
- 2x USB to Serial adapter (+12V/-12V)
- 2x XBee 868LP chips
- 2x 3 jumper cables for Ground (GND), Transmitter (Tx), Receiver (Rx)

As can be seen in Figure 4.2, each USB-to-Serial adapter cable is connected to one laptop. For each D-SUB port, the pins 2(Tx), 3(Rx) and 5(GND) are used to connect to the XBee devices by making use of jumper cables. Therefore one laptop is connected to one XBee chip. To simplify the setup, two separate laptops have been used, as the data sent between the two can directly be seen on the other screen. It would also have been possible to attach both USB-to-Serial adapters to one laptop and perform the tests just with a single workstation.

![Figure 4.2: Setup for experiments](image-url)
It is worth mentioning that, since a proprietary Printed Circuit Board (PCB) is used, the USB-to-Serial adapter needs to work with +12/-0 V and 0/-12 V. The level shifter on the PCB itself lowers the voltage from 12 V to 3 V, which is why a +3.3/0 V and 0/-3.3 V Future Technology Devices International (FTDI) adapter does not work.

4.3.2 XBee PCB

The proprietary PCB serves multiple functions. It can be seen in Figure 4.3 that on the left side a power plug is connected, providing the PCB with 12V. This is required as the power converter next to the power socket scales the voltages from 8-40V down to 3.3V. This is done as the whole UAV is supported by a 12V battery pack and all components are directly connected to this power supply. The data port provides 3 pins to connect to. Reading from the top to the bottom, the pins are Rx, GND, Tx. The serial cable is connected to this port to communicate with the XBee chip. Below the data connector a RS232 chip is mounted to convert the +12V and -12V signals from the serial connection down to +3.3 and -3.3V. If a direct connection to the XBee chip were established instead of using the proprietary PCB, a normal FTDI connector could be used. In the very middle the original XBee chip is placed, while on the right side of the PCB there are spots for LEDs.
Figure 4.4 depicts the same chip but the cover was removed. It can be seen that this XBee chip uses an EFM32 micro-controller inside. The radio chip is located in the upper right half of the XBee PCB. On the upper left is an empty position. The manufacturer provides a PRO version of the same chip, having an additional programmable micro-controller. As this is the normal version of the chip, the additional micro-controller does not exist.

Figure 4.4: XBee chip on proprietary PCB - removed cover

4.3.3 Spectral Analysis

The XBee protocol uses LBT + AFA as well as a proprietary modulation. Figure 4.5 shows the software SDRSharp in combination with a SALCAR SDR R820t DVB-T USB Stick while an XBee device sends data. The channel hopping within the frequency range of the DVB-T Stick is clearly visible. In order to show the size of the channel space used during a transmission, Figure 4.6 was taken while zooming in.
The documentation states that XBee 868LP chips use the frequency range from 863 - 870 MHz [64]. The used DVB-T stick only allows the display of a total bandwidth of 3 MHz. For this reason the figures displayed above are showing only a fraction of the whole transmission range.

4.3.4 Overview

At least two XBee modules are required for communication. The following data is needed to send packets from one XBee in a direct manner to another XBee (Point-to-Point). The parameters can be changed by anybody who has physical access to the device:

- PAN ID
- Channel
- BAUD rate
• Device High (DH) address
• Device Low (DL) address

Digi International uses default values for PAN ID, Channel and BAUD rate and only changes the DH and DL values to allow the customer a very quick use of the devices. The customer only needs to insert the DH and DL addresses of the other party and the chips can communicate. As the manufacturer of the UAV does not change any parameters himself, only the DH and DL parameters vary.

So far, four possibilities are known to gain knowledge of the DH + DL values:

• Reading them from the cover of the chip (physical access needed)
• Reading out from the storage on the chip (physical access needed)
• Brute-force
• Software Defined Radio

The authors in [65] focused on directly reading the connection data from the cover of the chip and were therefore able to send data and make the XBee chip accept the same. Obviously, once the connection parameters are known, forged data can be sent to the UAV and different attacks performed (e.g. replay). However, this would not be possible for an attacker and is a greatly simplified scenario, as the chips are hidden in the hardware of the user (UAV and telemetry box) and the attacker lacks physical access. The very important problem on how to acquire those connection parameters was excluded from their research, while this is a crucial step to compromise the security of the channel. Their argumentation is based on time and hardware constraints. It can be assumed that the authors had the same assumptions which were made in the beginning of this research (DH+DL cannot be acquired easily and cannot be changed remotely) and did not find a way to actually solve this problem.

Reading out the chip with software can also not be considered a realistic attack scenario (given that a cable connection is required and the attacker has no physical access).

Both previously-mentioned approaches are not practical to obtain the needed parameters. Therefore we focused in this research on the two approaches left which are explained in the following sections.

4.3.5 Brute-force

Brute-force is a fairly simple approach to gain knowledge about a secret. By trying all possible combinations the attacker must find the
right one. The downsides of brute-force are the time and resources which are needed to perform the attack. The time can usually be reduced by adding more resources, while more resources increase the cost of the attack. On the other hand, if the cost needs to be kept low, it takes longer to perform the attack.

Table 4.1 presents the level of difficulty for a brute-force attack, stating all possible combinations.

<table>
<thead>
<tr>
<th>Name</th>
<th>Combinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAN ID</td>
<td>65,536 possibilities</td>
</tr>
<tr>
<td></td>
<td>(0x0000 – 0xFFFF)</td>
</tr>
<tr>
<td>Channel</td>
<td>16 possibilities</td>
</tr>
<tr>
<td>BAUD Rate</td>
<td>8 possibilities</td>
</tr>
<tr>
<td>Device High Address (DH)</td>
<td>4,294,967.296 possibilities</td>
</tr>
<tr>
<td></td>
<td>(0x00000000 – 0xFFFFFFFF)</td>
</tr>
<tr>
<td>Device Low Address (DL)</td>
<td>4,294,967.296 possibilities</td>
</tr>
<tr>
<td></td>
<td>(0x00000000 – 0xFFFFFFFF)</td>
</tr>
<tr>
<td>Worst case</td>
<td>$\sim 15.4 \times 10^{25}$ tries</td>
</tr>
<tr>
<td></td>
<td>($65.536 \times 16 \times 8 \times 4.294.967.296^2$)</td>
</tr>
<tr>
<td>Case with default values</td>
<td>$\sim 18 \times 10^{18}$ tries</td>
</tr>
<tr>
<td>for PAN ID, Channel and BAUD rate</td>
<td>($4.294.967.296^2$)</td>
</tr>
</tbody>
</table>

The critical factor is the large address space of DH and DL. Another factor to consider using the brute-force approach is that the XBee chip is using Electrically Erasable Programmable Read-Only Memory (EEPROM) (for persistent storage of the addresses). Rewriting would wear the EEPROM out. With so many combinations the EEPROM would be destroyed while testing (and therefore the XBee chip made unusable). If the attack could be run software based instead of hardware based, it would overcome the physical limitation. Section 4.3.7 presents such a possibility.
During this research, we have observed patterns in the address space of DH and DL, which greatly reduce the address space. According to the manual of the XBee chip, DH is always set to "0013A200" [64]. Additionally, the first two HEX characters of DL are always set to "40". This leaves only 6 characters which are changed upon shipping of the XBee device. As the manufacturer of the UAV confirmed, the addresses from the shipped XBee devices are not changed before they are used in their products. Considering the aforementioned facts, the address space shrinks to $16^6 = 16.777.216$ possibilities, as it can be observed in Table 4.2.

**Table 4.2: Brute-force calculation for XBee link with address patterns**

<table>
<thead>
<tr>
<th>Name</th>
<th>Combinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAN ID</td>
<td>65.536 possibilities</td>
</tr>
<tr>
<td></td>
<td>$(0x0000 – 0xFFFF)$</td>
</tr>
<tr>
<td>Channel</td>
<td>16 possibilities</td>
</tr>
<tr>
<td>BAUD Rate</td>
<td>8 possibilities</td>
</tr>
<tr>
<td>Device High Address (DH)</td>
<td>1 possibility</td>
</tr>
<tr>
<td></td>
<td>(always set to $0x0013A200$)</td>
</tr>
<tr>
<td>Device Low Address (DL)</td>
<td>$16.777.216$ possibilities</td>
</tr>
<tr>
<td></td>
<td>$(0x40000000 – 0x40FFFFFF)$</td>
</tr>
<tr>
<td>Worst case</td>
<td>$\sim 14.1 \times 10^{10}$ tries</td>
</tr>
<tr>
<td></td>
<td>$(65.536 \times 16 \times 8 \times 1 \times 16.777.216)$</td>
</tr>
<tr>
<td>Case with default values for PAN ID, Channel and BAUD rate</td>
<td>$\sim 16.777.216$ tries</td>
</tr>
<tr>
<td></td>
<td>$(1 \times 16.777.216)$</td>
</tr>
</tbody>
</table>

Unfortunately, the amount of possibilities is still too large for a hardware based brute-force attack, as several chips would be destroyed during the rewriting process. Section 4.3.7.1 introduces another approach which removes the hardware constraints and makes the brute-force attack very feasible.
4.3.6 Software-Defined Radio

Another approach to compromise the XBee channel is to use a Software-Defined Radio (SDR). By tuning into the right frequency it should be possible to observe data sent between two XBee devices. The XBee chips apply channel hopping, which would require the attacker to follow either the same pattern or to listen on all possible channels in order to recover the full transmission. As LBT + AFA are used in the chip, the receiver does not know himself where the next transmission will be (as no pattern is exchanged) but hops through all channels and listens for transmissions. If a preamble is discovered the rest of the transmission is recorded.

In the situation when a full recovery of the radio wave is assumed, the attacker needs to find out the modulation in order to decode the wave into 1s and 0s. Only when one can get hold of the modulation scheme, by decrypting the firmware of the original XBee chip or reverse engineering the captured wave forms in a proper manner, can the actual data exchanged between these two devices be read. The problem lies in the fact that every modulation returns 1s and 0s and afterwards the meaning of these bits still needs to be discovered (packet structure, preamble, payload). Therefore it will be hard for a reverse engineer to figure out which modulation reveals the right sequence of 1s and 0s, in case it is not a standard modulation.

Reading out the firmware of the chip would also reveal the desired information. Since the chip knows the channel hopping algorithm and also the modulation used to decode the received radio signals this would compromise the whole system. Unfortunately, it is only possible to download the firmware in an encrypted format. Therefore a key is needed to decrypt the firmware and read the content. The encryption used is AES 256, which is considered secure up to the date of writing this paper, hence it is not possible yet to break the algorithm \[66\]. However, the following approaches can be considered to gain knowledge about the key.

Once uploaded to the XBee device, which uses an EFM32 microcontroller inside, the firmware will be decrypted and flashed \[67\]. Therefore the key must be stored on the device itself. Also the key must be the same for all shipped devices, otherwise the decryption would not work on all of them. Hence, a compromised key has a very high security impact. In order to gain knowledge of the key to decrypt the firmware, the following approaches can be considered:

1. The chip provides the developer with a debug interface. This debug interface allows access to the whole chip storage and should therefore be disabled before shipping. If this is not the
case, the key can be recovered and the firmware can be de-
crypted [68].

2. Even if the debug interface was disabled there is the possibil-
ity to activate it again. Unfortunately for an attacker, the chip
storage will be erased for security reasons and data will be lost.
According to the data sheet there are three different types of
storage on the chip: Flash, SRAM and User Data Page [68]. It is
recommended not to place the key in the User Data Page stor-
age, as this is the only storage which will not be wiped upon
restoring the debug interface. If the key is placed in this area,
the debug interface can be reactivated and it is possible to read
out the key.

3. In situations where the debug interface was deactivated before
shipping and the key was not stored in the User Data Page
area, there is no deliberate way to read out the key. But, as it
is very hard not to leak information during a computation, a
side channel attack might still be successful to recover the key
[69] [70]. Differential Power Analysis (DPA) measures the power
consumption of a device. Upon decryption the power consump-
tion is higher than usually. Therefore intermediate states can be
guessed and possibly the key reconstructed. Since the decryp-
tion needs to be performed by the device when the firmware is
flashed, it should be possible to record power traces of the AES
decryption.

This attack requires a great amount of time and resources. De-
pending on the level of noise, more or fewer power traces are
needed, which means that the decryption needs to be triggered
the same amount of times (automatically or manually). Also the
equipment is quite expensive as well as the software needed to
analyze the generated power traces. The chip uses a very low en-
ergy level compared to other models, which makes it even more
difficult to recover the key as the noise level is much higher. A
DPA analysis on this chip has already been performed without
success in [71] using the hamming weight model. No DPA re-
search on this chip could be found which used the hamming
distance model as a power model.

If an attacker can successfully recover the key and decrypt the
firmware, the functionality of the XBee chip could be rebuild in a
SDR.

Apart from using a SDR to eavesdrop on the radio communica-
tion and reverse engineer the modulation, it can also be used as an
instance to acknowledge that the brute-force attack was successful. If
an XBee device receives a packet containing a different destination address then its own, the packet will be discarded. But if the destination address matches the address of the XBee chip, an acknowledgment will be generated and sent. In this case the SDR can be used as an instance to prove a right DH and DL guess for an XBee chip. However, it can be hard to visually differentiate between the initial packet and the acknowledgement on the screen as the time in between those two packets would be very short and therefore the observer might not be able to tell if the observed data sent on the channels is in fact an acknowledgement.

4.3.7 XBee API-mode and Broadcast

During this research an easier, more convenient and sophisticated solution has been found to gain knowledge of the needed DH + DL values. It extends therefore the four possibilities listed in section 4.3.4.

Firstly, the different XBee modes are explained in section 4.3.7.1. For an attacker, the XBee API mode is crucial for success as it opens up many possibilities. Secondly, XBee broadcast is introduced in section 4.3.7.2. XBee chips are responding to every broadcast packet sent within their network with an acknowledgement. By parsing and interpreting the acknowledgement (only forwarded in API mode), it is possible to determine the sender’s address. The sender of the broadcast packet gains knowledge about the answering chips within his network and is therefore able to send directly targeted packets towards these addresses. A broadcast packet is only sent within an XBee network and the default values for preamble and network ID, which are defining a network, can be changed. To discover non-default networks, section 4.3.7.3 describes an approach using brute-force. With this approach practically every XBee network can be discovered. Lastly, Remote AT Commands are explained in section 4.3.7.4.

Every section presents an important ingredient for a successful attack. Their practical implementation within a Proof-of-Concept is explained in section 5.2.1.

4.3.7.1 XBee modes

There exist two different working modes for the XBee modules. The one used in the UAS setup is called transparent mode. In this mode, all data received by the chip through the serial interface is considered payload and wrapped into a packet, whereby the chip adds the stored DH + DL values together with preamble and network ID from its memory. A change of the stored DH and DL value is required, in order to communicate with another chip. Figure 4.7 presents the
composition of a packet in a simplified manner for the transparent mode.

\[
\begin{array}{|c|c|c|}
\hline
\text{Preamble} & \text{Network ID} & \text{Destination Address} \\
\hline
\text{Added by chip} & \text{Received from serial port} \\
\hline
\end{array}
\]

Figure 4.7: XBee packet composition in transparent mode

However, there exists another mode called Application Programming Interface (API) mode. Using this mode, the chip does not only expect payload, but it expects a so-called API frame in which the payload is included. The destination address can also be specified in one field of the API frame along with other parameters. This eliminates the need of changing the DH and DL addresses in the hardware memory of the chip as simply other API frames can be generated and handed over to the chip. Regardless of the chosen mode, the chip adds the preamble and network ID from its storage to the payload in transparent mode, as well as to the API frame in API mode. Figure 4.8 depicts the simplified composition of a packet for the API mode. The structure of an API frame can be found in Appendix B.

\[
\begin{array}{|c|c|c|}
\hline
\text{Preamble} & \text{Network ID} & \text{Destination Address} \\
\hline
\text{Added by chip} & \text{Received from serial port} \\
\hline
\end{array}
\]

Figure 4.8: XBee packet composition in API mode

As only the DH and DL are changed by the manufacturer of the investigated UAV, a connection can be established using a brute-force approach in API mode for the DH and DL addresses. The address space of DH and DL together has in the worst case \(18 \times 10^{18}\) combinations. Therefore a brute-force attack seemed unfeasible. As explained in section 4.3.5, patterns could be recognized which reduce the address space significantly, leaving only \(16^6 = 16,777,216\) possibilities. With this amount of combinations and API mode enabled, a brute-force attack is possible. Fortunately a calculation of the potentially
needed time for this attack was not needed as the interpretation of acknowledgement packets offers a far more convenient and sophisticated solution to gain knowledge about the desired connection parameters.

4.3.7.2 Broadcast

The chip discards all packets which are received but contain another address than its own in the destination address field. However, there is another option which was discovered during more detailed experiments. The used XBee devices also allow for sending and receiving of broadcast packages. Moreover, every received packet which is considered valid by the chip (broadcast packet or directly addressed to this chip), independent of the sender’s address, will be acknowledged by the receiving chip. Even if the receiving chip has a different destination address stored in its memory, it will return an acknowledgement to the sender of the packet. This feature is used in the software XCTU for a tool called “Node Discovery”. It allows the user of one XBee chip to discover other chips in range which are using the same preamble and network ID. This functionality can be easily abused for malicious purposes as the acknowledgment reveals the address of the responding chip. Therefore brute-force of DH and DL addresses became superfluous.

The only measure to complicate detection of the used network is to change preamble and network ID. The parameters are checked in the firmware of the chip itself upon reception of a packet. If the values of the preamble or network ID are not matching to the ones stored on the chip, the frame is not forwarded to the serial connection, but directly discarded. Presumably this was originally implemented to avoid interference with other networks in range and overloading the serial output of the chip with data from other networks. Since the chip is proprietary and only official firmware can be installed, it is not possible to force the chip to forward such packets. If this were possible, the preamble and network ID would also not contribute additional obscurity. Section 4.3.7.3 provides insights on how to circumvent this limitation.

4.3.7.3 Approach to Detect other Networks

Approach: As explained in section 4.3.4, in a generic scenario the attacker needs to know the correct preamble and network ID in order to send packets which will be accepted by the recipient (for the investigated UAV they are always the same). Additionally, DH and DL need to be known, but broadcast packets (section 4.3.7.2) are substituting this need. As the attacker can choose the preamble and network ID for the chip under his control, a brute-force attack can be
mounted. The preamble needs to be a value between 0 and 9. Therefore 10 possible combinations exist. The network ID needs to be a value between 0x0000 and 0xFFFF. Therefore 65,536 possibilities exist. Combined, 655,360 possibilities occur. In every combination theoretically just one broadcast packet is needed to conclude if XBees are receiving packets on this network or not. If an acknowledgement is sent back, the right combination was chosen.

**Cost:** The values of preamble and network ID are always taken from the storage of the chip and added to the packet. Therefore we can only implement the brute-force attack by changing the values inside the hardware. The data sheet of the internally used EFM32G230 microprocessor states that only 20,000 write cycles are possible until a failure of the memory occurs [72]. Using a brute-force approach, 32 XBee chips would be destroyed in the worst case. On average, the right combination would be found by destroying 16 XBee chips. At a cost of around USD 20, the average cost would be USD 320 to find a network. The maximum cost for a successful attack would be USD 660, including also the last device which stays intact. This seems reasonable for high value targets like a UAV for 25,000 € or even more critical devices.

**Impact:** As a result of the previously presented findings, it can be assumed that a network cannot be hidden from an attacker. If an attacker is always (with limited effort) able to communicate with the devices within the network, the transferred data needs to be validated to differentiate between valid and malicious data. The given XBee chip provides an encryption feature, which would prevent receiving and forwarding malicious packets to the serial interface. The encryption is disabled by default and is also not enabled by the manufacturer of the UAV. The possibility to turn on the encryption is known, however the manufacturer of the UAV claims that the transmission rate becomes too slow for the used application. Table 4.3 shows the measured transmission rates by Digi International. It can be seen that the official measurements indicate an insignificant reduction of the throughput rate. This leads to the conclusion that either these measurements were not really accurately performed by the manufacturer and higher values are stated than can be achieved in reality or that the manufacturer of the UAV eventually had an error in his setup.
Table 4.3: XBee throughput rate with or without encryption

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Data Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point to point unicast, Encryption Disabled</td>
<td>8.4 kbps</td>
</tr>
<tr>
<td>Point to point unicast, Encryption Enabled</td>
<td>8.3 kbps</td>
</tr>
</tbody>
</table>

Source: [67]

4.3.7.4 Remote AT Commands

XBee provides a possibility to remotely change parameters. By sending the right API frame containing the DH (or DL) parameter with a new value the remote chip will temporarily use the newly received address. To persist the change a second API frame with the Write (WR) command is needed. As no integrity checks are performed, an attacker is also able to remotely change the DH and DL addresses of a any given XBee chip and can therefore redirect the whole data transfer. Once this is done, the attacker has the entire control over the channel and interference with the original user can be avoid, if desired. This will be exploited in an attack proposed in section 5.2.1.

4.4 Manual Remote Control Link

The manual remote control communication uses on the remote control side a Graupner SJ MX-20 and is realized on the UAV side through a Graupner SJ HOTT GR 24 2.4 GHz receiver. It uses Frequency Hopping Spread Spectrum (FHSS) as channel hopping technology, with a pseudo random hopping pattern through 79 different channels with 1 MHz spacing. Therefore possibly making use of 79 MHz bandwidth.

FHSS is not an encryption technology (although some resources are suggesting this [73]). It is merely a way to split data through a lot of different channels within a given bandwidth to allow coexistence of multiple devices within the same area [74]. Therefore it is very resistant to jamming (up to 26 networks can coexist with a high throughput rate). However, if an attacker could capture all traffic sent through all 79 channels simultaneously, he is able to reconstruct the payload. In order to do so, 79 MHz of bandwidth need to be monitored permanently and the received traffic concatenated. Multiple Universal Software Radio Peripheral (USRP) 1 devices would be needed to cover
the whole bandwidth \cite{75}. Therefore, the equipment to perform such monitoring is quite expensive and was not available for this research.

There has been an approach called Hop Hedy Attack presented at ShmooCon 2011 to record the pattern of a link and predict the upcoming sequences using FHSS within 24 hours \cite{76}. Unfortunately, as the attack takes 24 hours and the regular flight time of a UAV is not more than one hour, the attack seems unfeasible.

For the aforementioned reasons this research does not focus further on the 2.4 GHz manual RC channel.

4.5 Video Link

The telemetry box possesses the ability to receive video signals from the UAV and can display those on the mounted screen. However, the UAV was not delivered with a camera. Also the transmitter for the video signal on the UAV is missing, hence no analysis of this link can be performed.

Although there was no analysis itself possible, one observation could be made during the analysis of the other channels. Since the telemetry box has a common power source, the video receiver turns on when the WiFi and XBee modules are powered. During a visit to the manufacturer’s site, the video screen received signals from another video transmitter used in production. Therefore it can be assumed that there is no encryption applied on this link and every receiver tuned into the right frequency can receive the video signals sent by the UAV.

For surveillance missions this can have an impact as the interceptor would have the same knowledge about the observed situation as the legitimate operator.

4.6 GPS

If the UAV is pre-programmed, it uses civil GPS signals to determine its position. Therefore GPS-specific attacks can be applied, which have been explained in section 2.3. Unfortunately, GPS Spoofing requires additional equipment which is expensive. GPS Jamming can be done easily. It is most likely that the UAV would rely on its Inertial Navigation System (INS) in this case, which makes flying unstable and inaccurate. The INS gathers data from several sensors attached to the UAV. If these can be confused or fed with false information, the UAV would act accordingly. The problem is that these sensors are attached to the UAV (inside and outside), hence it is hard for an
attacker to perform manipulation as he has no physical access to the UAV during a flight.

4.7 Concluding remarks

This chapter listed the different communication channels used within the investigated UAS. For every channel, the relevant security features were pointed out and vulnerabilities explained. Moreover, approaches to exploit the previously-explained vulnerabilities have been proposed.

It can be concluded that RQ-3 has been answered. The WiFi communication channel makes use of WEP encryption, which is considered weak. The XBee communication channel uses LBT + AFA as channel hopping algorithm. No link layer encryption is applied. Also, no application layer encryption is applied anywhere. As for the remote control, FHSS is used as a hopping pattern, which makes it quite hard for an attacker to sniff the whole bandwidth of 79 MHz. Again, no encryption is applied and once the data stream can be captured this channel can be compromised. The video signal is not encrypted and can be sniffed easily by any receiver tuned into the right frequency. Furthermore the UAV uses civilian GPS, which is why GPS Spoofing attacks can be applied. RQ-4 has also been answered, as vulnerabilities have been pointed out. It will be further developed in chapter 5, containing the exploits of the presented vulnerabilities.
ATTACK SCENARIOS & COUNTERMEASURES

The following chapter provides information about the attack scenarios developed during the research period as well as their countermeasures. It therefore further develops RQ-4 and answers RQ-5.

RQ-4: Do the used technologies have security vulnerabilities and can those be exploited?

RQ-5: What would be appropriate countermeasures to close the discovered vulnerabilities?

Firstly, an attack on the WiFi link is performed in section 5.1. Afterwards, attacks on the XBee link are described and implemented in section 5.2. Every section includes a description of the actual attacks as well as countermeasures to improve the current situation.

5.1 INVESTIGATED UAV - WIFI COMMUNICATION

This section will provide insights in how to exploit security weaknesses of the WiFi communication channel of the investigated UAV and will demonstrate the feasibility of such attacks. Moreover, different attack scenarios in conjunction with their applicability to other UAVs on the market will be explained. Figure 5.1 illustrates the attacked communication channel within the whole setup.

![Figure 5.1: Attack on the WiFi link of the UAS](image)

5.1.1 Attacks on the WiFi Communication

**Denial of Service:** As with every other WiFi network, it is possible to send de-authentication packets to the client, making the client believe that the Access Point (AP) lost the connection and requires a re-authentication. If these packets are sent over and over again, it essentially makes a stable connection impossible. This attack works with every WiFi network and is not specifically tailored to UAVs. However,
as WiFi is used it is applicable. Using this attack an attacker can prevent a legitimate user from controlling his UAV.

**Taking over the communication:** The investigated UAV uses a WEP-encrypted network to connect the tablet to the telemetry box. With the attack detailed in the next section it is possible to crack the passphrase for this network. Being at this stage, the attacker is able to connect his own device to the telemetry box.

One problem which will occur is that Internet Protocol (IP) addresses are hard-coded in the source code of the flight planning software. Since the communication will only work using these static IPs, the attacker needs to disconnect the legitimate user and prevent him from reconnecting, otherwise IP address conflicts will occur, making a take-over unstable. Sending continuous de-authentication packets to the Media Access Control (MAC) address of the legitimate client will solve this problem. As soon as the legitimate user has been disconnected, the attacker can connect his own tablet with the flight planning software and take-over control. Figure 5.2 depicts the attack scenario.

![Attack scenario](image)

It should be noted that this take-over scenario is not only applicable to this specific UAV but was used for several UAVs in the past. If the WiFi network is an open network, it is even easier for an attacker to perform the attack as there is no need to crack the channel password in order to gain access. This scenario applies to the SkyJack project explained in section 2.1.

A major disadvantage of this attack is its limitation of applicability. The attacker needs to be in close range to the AP. Hence, the attack might be not possible (in cases where the area is closed). Besides,
since such a close vicinity to the telemetry box is required (within WiFi range) the attacker might be discovered.

5.1.2 WiFi Cracking

As explained in section 4.2, the telemetry box of the UAV uses a WEP-protected network to connect to the tablet, which is running the flight planning software. Since WEP encryption is insufficient as an encryption algorithm and known to be weak, this channel has been attacked first.

The network name is "[****]-12536", where [****] was used in this paper instead of the UAV name. In order to recover the password to connect to the network, WiFi cracking is required. In this case, the attack was performed with a Kali Linux 64bit and the network was available at the time of attack. The cracking process has several steps:

1. List the available interfaces and create a monitor

   The command "airmon-ng" lists all available interfaces on the attacker’s machine. The interface wlan0 is now displayed and the command `ifconfig wlan0 down` is used to disable the WiFi interface on the computer in order to avoid complications. The former makes it impossible for Graphical User Interface (GUI) applications to connect to WiFi networks. Afterwards a monitor is generated by using the command `airmon-ng start wlan0`. The generated monitor can be addressed as mon0.

   As it is shown in Figure 5.3 the traffic on this monitor can be observed using the "airodump-ng mon0" command.

   ![Display all available networks - airmon-ng](image)

   **Figure 5.3:** Display all available networks - airmon-ng

Since all available network traffic is now captured, the attacker might refine the filters of the traffic for further analysis. This can easily be done by adding the matching parameters for the targeted WiFi network to the command. The command to show traffic for just the attacked network is:

   # airodump-ng -c 11 --bssid 00:0E:C6:00:AC:63 mon0

   ![Filter for desired network](image)

   **Filter for desired network**
46 attack scenarios & countermeasures

The parameter `-c 11` specifies the channel, while `-bssid 00:0E:AC:63` is filtering only traffic coming from the AP we wish to attack.

2. Association of MAC address with the Access Point

As there is no associated client in the attacked network, fake-authentication is required to authenticate and associate the attacker’s computer with the network. If the attacker is not associated with the AP before injecting packets, the AP will just discard the received packets and no additional Initialization Vectors (IVs) are generated. The association itself does not generate IVs, but instead it makes the AP accept injected packets for the future. The command used to associate the interface of the attacker’s computer with the AP is:

```
# aireplay-ng -1 6000 -o 1 -q 10 -e 12536 -a 00:0E:AC:63 -h 8C:A9:82:C3:60:74 mon0
```

The parameter `-1 6000` triggers a fake-reauthentication every 6000 seconds, `-o 1` makes the tool only send one packet at a time, `-q 10` sends a keep-alive packet every 10 seconds, `-e 12536` specifies the network and `-a`, respectively `-h`, are announcing the MAC addresses of the interface and the AP. Figure 5.4 depicts a successful fake authentication and keep-alive packets.

Figure 5.4: Fake authentication - aireplay-ng

3. Capture and generate traffic for the desired network

To capture all network traffic sent, the previously-defined filter is used and the output written to a file. This file can be later used for analysis with aircrack-ng. The command used is:

```
# airodump-ng -c 11 --bssid 00:0E:C6:00:AC:63 -w capture mon0
```

The only difference to the previously-explained filter is the switch `-w capture`, which writes the output of the command to a file called `capture-01.cap`.

The generation of packets is important to create traffic on the network and to provoke the AP to reply with packets containing
IVs. To achieve this goal, Address Resolution Protocol (ARP)-packets were recorded and broadcasted back to the network, making the AP reply. In Figure 5.5 shows this process.

Figure 5.5: ARP replay - aireplay-ng

4. Crack password

By simply starting aircrack-ng with the output file of the previous step as a parameter we are able to start the cracking process. Figure 5.6 displays a successful attack on the targeted network. It can be seen that the key is 2012201212.

The tool stops and automatically restarts if the file contains insufficient IVs for the cracking process. Once the amount of IVs contains enough information to crack the password, the tool reports a success message including the desired key. The capturing and cracking process took 18 minutes and needed 15,000 IVs, which can be considered slow. The main reason is that the fake-authentication and association, as well as sending fake packets to generate IVs, were performed after starting the capturing and cracking process. Therefore the timer was already running.

Figure 5.6: Successful key cracking - aircrack-ng

From now on a tablet can be connected to the telemetry box in order to program the flight computer. Although it is known that WEP provides insufficient protection, it is still used in many (legacy) applications. The use of such a weak encryption within a very new and expensive device, which is also used for sensitive tasks, is surprising.
5.1.3 Attack Countermeasures

The appropriate countermeasure for the exploited vulnerability is quite straightforward in this case. As an old encryption mechanism is used, it needs to be updated. The manufacturer should replace the used hardware modules where they do not allow newer encryption methods than WEP. Appropriate WiFi modules are not expensive, yet old ones are still in use and an easy target for attackers. At the time of writing this thesis, WPA2 encryption is state-of-the-art and still considered secure. Therefore the change towards this newer encryption technology is proposed.

Another important point is that the pass-phrase to authenticate towards the WiFi AP should be changed for every customer. As it was confirmed by the company, so far the cracked combination 2012201212 is used for all devices. The use of default passwords is a major flaw and very dangerous as the encryption becomes superfluous. If the attacker knows the pass-phrase for one UAS, he is able to hijack every UAS ever sold. Moreover, chosen passwords should not only contain digits, but also upper and lowercase characters as well as special characters [77] [78].

In cases where the hardware does not support newer encryption methods, the substitution of already-shipped equipment is the only way to mitigate the risk revealed by this research. As the attacker does not need to understand the underlying technology nor develop a custom exploit, but instead is able to reuse existing resources available on the Internet, the exploitation of WEP can be done rather easily and does not require very advanced hacking skills. Since the investigated UAS is used by police forces for different purposes, including critical missions, the security of the device should have a high priority.

5.2 Investigated UAV - XBee Communication

This section will give insights in the attack scenarios specifically applied to the XBee chip used in the investigated UAV. Figure 5.7 illustrates the attacked communication channel within the whole setup.

Figure 5.7: Attack on the XBee link of the UAS
5.2.1 *Attacks on the XBee Communication*

**Denial of Service:** To perform a DoS attack, different options can be chosen. Either the connection can be jammed, simply overloading the used channels, or a more sophisticated attack exploits the behavior and abilities of the technology and protocol in use.

The second option will be explained in greater detail and is simultaneously also needed to perform further attacks. Firstly, the attacker performs the aforementioned brute-force attack on all possible preamble and network ID combinations (if the values are not default). By sending broadcast packages using all possible combinations, the attacker will find the active devices. Secondly, the attacker uses a so-called "Remote AT Command" packet, introduced in section 4.3.7.4, to change the destination address of the attacked XBee chip (which is considered a feature of the protocol but can be abused for a malicious purpose in this case). The payload of this request consists of the hexadecimal encoding of the command, followed by the new parameter in the next field. This packet alone will change the address of the attacked XBee chip temporarily, as the new address will not overwrite the existing value but still be used as long as the power supply continues. Upon restart of the device, the old address is in use again. To persist the changes in the memory of the attacked chip, a second "Remote AT Command" is needed. The second packet includes the hexadecimal encoding for the "WR" (write) instruction. From now on the chip will always use the new destination address, even after a reboot.

By simply setting the destination address of the telemetry box as well as the UAV chip to a random value, the communication will be interrupted. "Remote AT Command" instructions can, as every other packet, be sent as broadcast packets, removing the need to address every chip separately. Figure 5.8 depicts the setup and the flow of the attack as it can also be used for DoS. In step 3 the addresses should be set to a random value.

**Man-in-the-Middle:** There are two things needed in order to interfere with the flight computer of the UAV. Access to the communication channel and the right payload.

The first requirement can be fulfilled using the DoS approach explained in the subsection above, but instead of using a random value, the destination addresses of the attacked chips should be set to the attacker’s address. The attacker is at this point able to listen to all communication that is exchanged between the attacked chips and forward the communication between those to allow normal behavior.
The performed attack is called Man-in-the-Middle.

The second requirement can be fulfilled by reverse engineering the original data which is sent through the channel. The attacker can at this point listen to the communication, therefore he might be able to infer the meaning of the payload and reconstruct commands. Even if it is not possible to make sense out of the payload, simply replaying previously recorded packets is possible (assuming that no sequence numbers, etc., are used). However, the manufacturer of the UAV provides its clients with a tablet as well as an installed application to control the UAV. As the application is used to program the flight computer of the UAV, the method and the used commands must be implemented in the software. Therefore reverse engineering of the Android Application Package (APK) might reveal this information and enable the attacker to use it as a payload.

5.2.2 Popularity of XBee

Several UAVs are using the investigated XBee chip in their setup as a low-cost and easy-to-configure solution and are therefore vulnerable to the attack proposed in section 5.2.1. Although most manufacturers are not publicly announcing which internal hardware they use it was possible to determine this information for some UAVs. Examples of professional UAVs apart from the investigated one are the AscTec Pil-
ican and the Draganfly [79] [80].

For consumer UAVs the usage might be even higher, as construction kits and tutorials for XBee are widely available [81]. An example of a consumer UAV which is sold with a ready-to-use XBee setup is the Plexidrone [82]. Additionally, XBee is a low cost device and can for this reason be easily purchased by hobbyists and enthusiasts.

It should be added that the mentioned UAVs are using the same XBee technology but on a different frequency. Probably due to regulations and data transmission rates the manufacturers chose to use a 2.4 GHz link (2.4 GHz can be used worldwide while 868 MHz is only permitted in Europe). This should not have an impact on the proposed attack, as the same underlying technology is used just on a different frequency band. However, although this hypothesis can be considered quite strong, it has not been proven.

Apart from UAVs, the investigated XBee chips are also in use for all kinds of other applications. Since the chip itself is susceptible to the attack, all appliances using it without an activated encryption feature are at risk. The mitigation through XBee encryption is further explained in section 5.2.6.

5.2.3 Reconstruction of Commands

In order to gain knowledge about the payload sent by the Android App to the flight computer, the Android APK was analyzed.

To install software on Android, an APK is needed. It is an executable file, similar to an "exe" file on Windows Operating System (OS). The APK consists of compiled Java source code. It has a specific structure and mandatory files.

To retrieve readable code the following procedure has to be applied:

1. Rename the .apk file to .zip
2. Extract the .zip file with a common archive manager
3. Apply Dex2Jar to transform from classes.dex to .jar archive
4. Utilize JD-GUI to visualize the .jar archive

First, the .apk file is renamed to a .zip file. This step just changes the extension and allows the archive manager to recognize the file as a valid archive. Afterwards the archive manager is used to extract the archive to a normal folder. This folder contains several sub-folders...
with the raw data (icons, XML and style files) and the main source file with the name classes.dex. The extension derives from the Dalvik virtual machine environment, in which every APK is executed once it is installed on an Android device. To transform this classes.dex file into readable code, we need to apply a tool called Dex2Jar. As the name already suggests, the tool will decompile the classes.dex file and create a .jar file. This extension represents a Java archive. Finally this file can be opened with JD-GUI to read the content.

Due to performance optimization and eventually obfuscation applied during the compilation of the original source code, names inside the Java files are only partially reconstructed. Figure 5.9 depicts the file tree within the .jar archive opened in JD-GUI. Sensitive data had to be removed from this figure according to the NDA.

An Internet search of the package names reveals the original Android APK of the flight computer. The application can be downloaded from the Android App store, but does not work in combination with the flight computer inside the UAV.

Figure 5.9: Android App decompilation - file tree
As the manufacturer of the flight computer provides user documentation on the up-to-date model they are offering online, it could be confirmed that the flight computer build in the investigated UAV looks similar and is most probably a related release. As the flight computer case inside the UAV is not labeled, the exact version could not be determined. To gain further knowledge, the Android APK available on Google Play has also been decompiled and compared. The package names are the same but the source code of the Android APK available through Google Play has been obfuscated before compilation. Class names are named from a.class to z.class making it impossible to gain information from the name. However, as the majority of the source code and the names of the packages are the same, we believe that the manufacturer of the investigated UAV is in possession of the original source code and modified it according to his needs. Also the APK downloaded through Google Play contains many more classes, which allows the assumption that the manufacturer removed certain unnecessary features and added new functionality.

While analyzing the different class files within the APK of the investigated UAV, the file CommApBase.class raised attention. It contains several methods which seem to construct commands. Also the function names "SendDataCodecmd()" seem to fit this assumption. Listing 5.1 illustrates how such a command is assembled.

Listing 5.1: Android APK - Command construction

```java
public void SendDataCodecmd(byte paramByte)
{
    byte[] arrayOfByte = new byte[30];
    for (int i = 0; i < 30; i++)
    {
        if (i >= 30)
        {
            arrayOfByte[0] = 36;
            arrayOfByte[1] = 87;
            arrayOfByte[2] = 73;
            arrayOfByte[3] = 70;
            arrayOfByte[4] = 73;
            arrayOfByte[5] = paramByte;
            arrayOfByte[6] = paramByte;
            arrayOfByte[7] = paramByte;
            arrayOfByte[8] = 0;
            arrayOfByte[9] = 0;
            SendbyteData(arrayOfByte);
            return;
        }
        arrayOfByte[i] = 0;
    }
}
```
As can be seen, an array of bytes is being filled with decimal values. Further investigation revealed that the data within this bytewa-
ray is transformed into a hexadecimal representation before forwarding it to the Android Network Service. The meaning of these values
themselves cannot be inferred from reading the code, but suggestions according to the naming of the functions can be made. Some func-
tions, like the one presented in Listing 5.1, are generic and made to transfer different code values in the paramByte variable, while others only serve one single purpose and are therefore hard-coded (see Ap-
pendix C for an example).

The Android APK, compared to a previous release for which the same decompilation procedure was applied, does not contain certain buttons anymore. The functionality inside the code for these buttons also seems to have been removed. Nonetheless, the flight computer might still accept those commands. One example is the Unlock En-
gines command. The button was removed from the interface, but the engines can still be started by injecting a packet with the matching payload.

Other researchers already worked on a similar flight computer, however, this research reveals the commands internally used and ex-
tends the previously performed research accordingly (not all refer-
ces can be given here due to the NDA) [65].

To infer the commands, a serial monitor was placed on the XBee chip of the UAV as well as the telemetry box (only Rx can be mon-
itored when the UAV is used). Then the buttons were used sequen-
tially in both apps and the output of the serial monitor correlated. Appendix D shows a summary of the commands of the flight com-
puter which were inferred by using reverse engineering.

The following commands are examples and their functionality is shown in the PoC:

\[
\begin{align*}
24 & 57 49 46 49 89 89 89 & \text{Start-Engines} \\
24 & 57 49 46 49 58 58 58 & \text{Auto-Takeoff}
\end{align*}
\]

Start-Engines initiates the same. Auto-Takeoff also initiates the en-
gines and increases the throttle to start a take-off.

Other commands trigger unexpected and undetermined behavior:

\[
\begin{align*}
24 & 57 49 46 49 75 75 75 \\
24 & 57 49 46 49 00 00 00 \\
24 & 57 49 46 49 46 46 46
\end{align*}
\]
For some of these commands, the throttle is increased to a maximum, then reduced or the engines suddenly stop working. Also the values for the joystick of the RC show undetermined behavior, therefore making it presumably hard to steer. For a flying UAV, this situation would be fatal.

It is important to note that no application layer encryption is applied. All commands can be sent direct to the flight computer. Therefore this setup is very vulnerable to a diversity of injection attacks. Examples are setting or deleting waypoints, changing the home location, reading and rewriting parameters on the flight computer, etc. Moreover, the UAV can be crashed by inserting the previously mentioned junk commands.

The primary goal of the reverse engineering was to find out the commands to interact with the flight computer. Although not used, also the command structure of the communication from the UAV back to the tablet has been analyzed. This structure can be found in Appendix E.

5.2.4 Emulation of Android Tablet

Since a tablet was not included in the equipment, the Android application had to be emulated in an Android emulator to make the setup work. To serve this purpose, the Genymotion emulator was used first [83]. Upon the start of several different emulator images, the same error message always appeared. Figure 5.10 depicts the error message.

![Figure 5.10: Error Message - Unsupported device](image)

Unfortunately, the error message did not include which devices are supported. To get this information, additional work was required. The Android application was disassembled (as described in section 5.2.3) and the source files were searched. A file called "check.class" attracted attention. It can be seen in Listing 5.2 that in the source code of this file, the string of "ModelNumber" is compared with the string "SM-T230". This string is specific for a Galaxy Tab 4. If the check succeeds the application starts, otherwise the earlier mentioned error message is displayed.
Listing 5.2: Model Check in Android APK

```java
if (this.ModelNumber.equalsIgnoreCase("SM-T230"))
{
    new Handler().postDelayed(new Runnable()
    {
        public void run()
        {
            check.this.finish();
            Intent localIntent = new Intent(check.this, start.class);
            check.this.startActivity(localIntent);
        }, 0);
    return;
}
```

Unfortunately Genymotion only offers very limited images and the desired device was not available. The software also did not offer any possibility to alter the "ModelNumber" string of the emulator.

As a result, the Eclipse platform and the Android Software Development Kit (SDK) Manager were installed. By using the Android Virtual Device (AVD) Manager multiple emulators can be created. All of these emulators are created using an image file called "system.img". This file is located in a Windows operating system under "C:\Users\*UserName*\AppData\Local\Android\android-sdk\system-images\*APIVersion*\default\x86".

When opened in a HEX-Editor like XVI32, the mentioned "system.img" file can be read in the displayed ASCII window. By searching for the string "ro.product.model", the required position can be found and the desired HEX value inserted to the file. Since the inserted string had more characters than the default one, the right amount of "00" characters at the end of the file had to be removed. Otherwise the emulator was not booting after creation. This leads to the conclusion that the size of the file is checked before using the image file, but not the content.

For compatibility issues the RAM of the Emulator Virtual Machine (VM) had to be set to 768 instead of 1536 and the Hardware Accelerated Execution Manager (HAXM) had to be installed. Choosing this configuration, hardware acceleration made the emulator much faster.

After changing the configuration as described, the start up of the Android APK was successful. The next step was to connect the host machine of the emulator to the WiFi network of the telemetry box in order to establish a connection between the App and the flight computer. It appeared that the IP-Addresses to connect to the flight computer were hard coded in the application. Since an Android emu-
lator is using a virtual router (the emulator cannot see the host), direct access to a specific network is not possible if it is not in the same address range as provided by the virtual router (10.x.x.x). Therefore the emulator approach proved to be a dead end.

Another option was to remove or alter the check functionality inside the application and to recompile and install it on real hardware. This was considered the last resort, as a PoC should preferably be conducted with the original setup. However, as no other option was left to make the app work and to connect it to the flight computer, the APK was decompiled using apktool as described in section 5.2.3 and the string inside the check method was altered to contain "ZP980". Afterwards, the app was recompiled with the apktool. As the Android OS does not allow the installation of unsigned APKs, although the option "Unknown sources" within the Android settings is active, we had to sign the APK first. The signing was done with a tool called SignAPK. It signs the APK with a non trusted certificate. This is enough to install the app when the previously mentioned option is activated (as only the presence of a signature is required).

Once the app was transferred and installed on a ZOPO 980 phone, the app was started. After launching the app it crashed immediately, which is why the source code was altered another time to "GT-I9505", a Samsung Galaxy S4 phone, including recompiling and signing. Here the application started up as planned and the phone was connected to the telemetry box WiFi network. For some reason, as soon as the connection established and data was exchanged between the app and the flight computer, a crash occurred. As a result, the altering and recompiling method also cannot be taken as a working solution.

Since some time was already spent on these activities, the final decision was made to use the app in the environment in which it is supposed to be installed. This is an original Galaxy Tab 4 tablet. As the manufacturer of the UAV was not able to provide a tablet dedicated for this research, the PoC was tried out on-site using a production tablet. When using the app with the supported device it works as expected and does not show the behavior experienced earlier, which is why the PoC could be performed as planned.

5.2.5 Proof-of-Concept

After the setup was running as expected (normal behavior) the attack was performed as illustrated in Figure 5.11. However, only steps 1-4 are needed to perform the attack in a temporary manner. As soon as
the devices are switched off and on again (UAV and telemetry box) the chips reuse the old addresses instead of the ones transferred in the Remote AT Command. This is an advantage for the attacker, as the attack has two different modes: temporary and persistent. Where a persistent change is desired by the attacker, step 5 can simply be executed as well and the remote XBees will persist changes to their memory. As persistent changes would leave evidence that an attack happened, a temporary change is even better. Considering a situation where the original user is somehow able to regain control over the UAV and performs an investigation into why the communication between UAV and telemetry box (and therefore tablet) was not working as expected, he would not be able to determine the reason as no log files are generated or traces left. Where the persistent version is chosen by the attacker and the addresses are therefore changed permanently, the user would discover changed destination addresses upon investigating (and this investigation would be necessary to reuse the UAV with the original telemetry box).

For this PoC the temporary alternative was chosen. It can be seen in the video in the attachment to this thesis how the attack is performed. The source code can be found in Appendix F.

After the MitM attack was performed, packets were injected containing the commands obtained in section 5.2.3. As the UAV reacted
based upon these commands, it was proven that no application layer encryption is applied on this channel.

5.2.6 Attack Countermeasures

This section will propose three different possibilities to mitigate the risks introduced in the previous section. All mentioned approaches with regard to encryption are using symmetric cryptography and the manufacturer is therefore required to exchange the key for every UAS.

The previously-proposed attacks are possible, because the recipient does not check the authenticity of any message (packet) which arrives. Moreover the whole communication is sent in clear text. As long as the arriving data is syntactically and semantically correct, the data is forwarded to the application. Another important point is that XBee allows a remote change of internally stored values, which is a result of its functionality and the intention for it to be used in mesh networks.

5.2.6.1 XBee on-board encryption

To ensure confidentiality of information, encryption should be used on the channel. As mentioned earlier, XBee provides such an encryption feature. This would be the first option to use, as it does not produce much overhead to implement. Additionally it takes away the functionality to modify internally stored values remotely without knowing the right encryption key. It can be seen in Figure 5.12 that the user can use the XCTU software to store an encryption key in the storage of the XBee chip to allow encrypted communication. Obviously, the encryption key needs to be the same on both devices, otherwise packets will be silently discarded. Since both sides store the same key, and encryption as well as decryption are done using AES-128, the encryption happens symmetrically. As the chip needs time to encrypt and decrypt the payload of each packet which is sent and received, performance decreases. Therefore link layer encryption realized by XBee would be a theoretical solution to the problem, but cannot be applied in this scenario.

![Figure 5.12: XCTU Security](image.png)
5.2.6.2 **Dedicated hardware encryption**

As the build-in encryption of XBee is too slow, a second option is additional hardware to take care of the encryption on both sides. This hardware would be placed directly in front of the XBee chip, allowing encryption and decryption of the serial data passing through. Figure 5.13 depicts the normal data transfer, while Figure 5.14 shows the setup with dedicated hardware encryption.

Since XBee is used in transparent mode within the original setup, it just forwards received data and does not need to interpret anything. In API mode such a solution would not be possible, as part of the data, which is handed over to the XBee chip, needs to be interpreted by the chip itself. If the data were encrypted, the chip would not be able to determine the transmission parameters (DH and DL) inside the encrypted data stream. This solution offers the same functionality as the XBee encryption but has a dedicated functionality, which is why no issues in performance are expected. The downside of this solution is additional equipment that needs to be located in every unit, consequently adding weight to it, perhaps making this solution unfeasible.

---

![Figure 5.13: Schematic XBee standard communication](image1)

![Figure 5.14: XBee & Dedicated hardware encryption](image2)
5.2.6.3 Application layer encryption

Application layer encryption is the third possibility. If the transferred data is already encrypted upon arrival at the XBee serial interface, no further encryption needs to be applied. Moreover, no performance issues on the XBee chips are expected, as the amount of data stays the same. Application layer encryption provides a logical layer over the used physical layers. For this reason it would mitigate confidentiality risks created not only by one weak link but all weak links. The firmware of the flight computer together with the Android App need to be modified for this purpose.

A further decision to make is to choose between symmetric or asymmetric encryption. The advantages of symmetric communication are simplicity and performance. Asymmetric encryption, on the other hand, allows encryption between two parties that have not met before. This use case is a typical example for symmetric encryption, as the keys can be placed in both devices before the first use. Additionally, more complex problems like certificates to ensure the integrity of the exchanged public keys and Certificate Authorities (CAs) can be avoided. Also, due to handshakes and the exchange of certificates, asymmetric cryptography is costly. As a result, only symmetric encryption is a feasible solution.

Data should be encrypted before it leaves the flight computer and decrypted in the flight planning software (Android App), and the other way round. The mentioned scenario implies that the encryption keys are stored in these devices and the computing power is sufficient. The implementation of the encryption requires changes on both sides of the communication channel and does therefore require development time. Implementation of application layer encryption could be expensive, especially if a change of hardware is required to deliver the sufficient performance for the cryptographic processes.

5.2.6.4 Further suggestions

As mentioned earlier, if the on-board encryption feature of XBee is not used, the proposed MitM attack in section 5.2.1, including the change of the DH and DL addresses, will still be possible. The difference is, that the attacker will not be able to read the content of the data. Since the change of addresses is still possible, also the DoS attack can still be performed. There is no mitigation to this end, apart from activating the XBee on-board encryption, which is not possible in this scenario. Hence, to avoid DoS attacks, a substitute has to be found which provides the same functionality and bandwidth as XBee, but does not allow changes of internal parameters. Another option would be to duplicate the channel by using two XBee communication chan-
nels in parallel with enabled encryption. The chips themselves would not be prone to the proposed attacks anymore, but additional logic to split up the communication and reassemble it has to be implemented.

Encryption ensures confidentiality not integrity of the data. An attacker could still record packets and, without knowing what is in the packets, replay them. If we assume that the attacker recorded the whole data stream and re-transmits all split packets, then the application would recognize a valid data stream and act accordingly. To avoid such behavior, sequence numbers or cryptographic nonces need to be used. If a packet is received twice, it needs to be discarded by the application. The XBee chip itself cannot provide such functionality, as the chip cannot decide if the payload is allowed to be sent again or not (there are legitimate cases). Therefore the application layer protocol needs to take care of this.

5.3 CONCLUDING REMARKS

RQ-4 has been further developed by adding the PoC and exemplifying an exploitation. The WEP encryption of the WiFi link has been broken and a MitM attack on XBee chips performed. Moreover, the Android APK was reverse engineered and commands extracted which were injected into the compromised XBee channel and successfully executed on the flight computer of the UAV.

To counter those vulnerabilities, countermeasures were proposed in the end of this chapter, answering RQ-5. The implementation of one of the suggested countermeasures is feasible but may require better hardware, which leads to increased production cost.

It can be concluded that changes in the setup of the UAS are required and need to be implemented. If the security of the system is not improved it will be left prone to attacks.
Chapter 1 provided an introduction to this thesis and stated the research goal, research questions and contributions. The following chapters answered those research questions subsequently. RQ-1 focused on the current state-of-the-art and was answered in chapter 2, providing also background for further reading. Chapter 3 gave a definition and classification of UAVs. Moreover it related to RQ-2 by elaborating on the communication channels which can be used to control UAVs and giving insights into the different attack vectors. As the general knowledge was provided in previous chapters, RQ-3 and RQ-4 focused on the investigated UAV and were answered in chapter 4. The specific security implementations were analyzed and the vulnerabilities pointed out. Chapter 5 followed with the exploitation of these vulnerabilities, therefore further developing RQ-4. By suggesting countermeasures to avoid exploitation, the attempt was made to improve the current situation, according to RQ-5.

6.1 CONCLUSION

During this Master Thesis the hypothesis that expensive, professional UAVs can be considered secure has been disproven. Although the costs for professional UAVs are extensively higher compared to consumer UAVs, the security of the investigated model can be judged insufficient. It was possible to perform a MitM attack on the XBee communication channel. As no encryption and authentication are applied anywhere, packets were successfully injected into the compromised channel, making the UAV react to the attacker’s commands. Due to the fact that multiple UAV manufacturers are using the investigated technology, the impact of this research is high.

This research will be shared with the manufacturers who are known to implement the investigated solutions and made publicly available. However, there are presumably many more manufacturers using the vulnerable setup without revealing their hardware components to the public, leaving their setup prone to attacks. To encounter this issue, security awareness within the community of UAV manufacturers is important.
6.2 **Future Work**

Firstly, there needs to be more studies on professional UAVs in general. Therefore, the presented research should be applied to several models of UAVs. This is a time-consuming activity and requires appropriate funding. However, it can be combined with the development of a product and leverage the expected Return on Investment (ROI). The Dutch government is working in conjunction with UAV manufacturers on a solution for the issue of invading UAVs under the control of an attacker [84]. A solution could be the proposed "framework for remote exploits" which can be found in Appendix A. Combating unlawfully used UAVs by performing logical attacks requires a database of exploits. Therefore for each of the targeted UAVs, a research similar to the presented one has to be performed.

Secondly, the research on this specific UAV can be extended. One possibility is to find out the proprietary XBee modulation in order to emulate XBee behavior in a SDR. A detailed explanation of the benefits and the approach can be found in section 4.3.6. Where XBee encryption is enabled the approaches explained in the end of the same section should be taken into consideration.

A list of control commands for the flight computer was provided in Appendix D. By extending this list, new combinations of commands might reveal other attack possibilities. In order to inject commands, access to the communication channel must be assumed.

In cases where necessary equipment and funding can be assumed, the 2.4 GHz manual RC link to the UAV should be analyzed. As FHSS is not an encryption technology, one should be able to capture traffic on the whole bandwidth and infer the transferred data. This link provides direct control and a compromise would be fatal.

GPS Spoofing was not performed in this research due to its complexity and the fact that it is illegal in Europe. Therefore, future research could also investigate the susceptibility of the UAV to GPS Spoofing.

Finally, the discarded research question in Appendix A can be a valuable target for further investigation. The scope of this research did not allow to include them, but it could be beneficial for further research with a wider scope to take them into consideration.
APPENDIX A

Previous Research Questions and Ideas:

RQ-6 Is it possible to write malicious code which interferes with the flight planning software in such a manner that manipulated coordinates can be sent to the flight computer?

Research question RQ-6 can be answered by investigating on available malware and how to exploit the targeted host of the flight planning software (e.g. a tablet running Android). Therefore the host OS should be discovered. Once the payload has been placed on the host, privilege escalation might be needed to access the data owned by the flight planning software. Either the data can be modified or the malware could directly communicate with the connected flight computer without using the original software (if no authentication is involved).

RQ-7 Where an Internet connection for the device running the flight planning software is assumed: Is it possible to extend the before mentioned software implementation in such a manner that the attacker can take control over the UAV in real time using the Internet?

RQ-7 is an extension of the sixth one. If the malware can be placed on the host and it works properly, the answer to this question would provide remote control over the UAV by using the Internet. To accomplish this goal an investigation into firewalls is required to open an unsuspicious connection to the attacker.

To give an insight in the process of how ideas for this thesis were developed, previous ideas will be presented in the following paragraphs.

Framework for remote exploits: The first idea was to build a framework to automatically detect UAVs in range and eventually apply the right exploit to them from an "exploit database". The requirements for this approach would be that exploits for certain UAVs can be created, as it otherwise makes no sense to create a framework which contains such. If a Proof-of-Concept could be done with two UAVs from different manufacturers this might be a reasonable approach. The problem with this approach is that the technology used for different UAVs is
not the same. This would require the computer running the exploitation framework to have all kind of different input interfaces (different UAVs are using different frequencies which require different antennas). Moreover, the exploit database of the framework needs to be filled to automatically apply the right exploits to the UAV in use (which means in turn that the UAVs need to have vulnerabilities). Where this research provides a possibility to compromise a control channel of a UAV, then the already existing approach (e.g. SkyJack) could be added to the framework to proof the feasibility of such a framework. Making this project Open Source would allow different security researchers to investigate their UAVs and add exploits to the framework (making the framework more valuable to the police). Before the decision to investigate the feasibility for such a framework is made, the research on the investigated UAV should proof fruitful.

**Framework for software exploits:** This idea relates to the discarded research questions, as it has been considered as the main topic for investigation before the focus was changed to the current research questions. Commercial UAVs are mostly pre-programmed and not steered by hand any longer. A remote control is usually available as a backup, but is not used in most cases, as even start and landing can be performed fully autonomously. This makes the software used to program the flight computer a valuable target. If a person were able to install malware on the device used for programming the flight computer, full control might be gained over the UAV by the attacker. However, there are multiple flight planning software vendors. An approach could be to develop malware targeted on modifying flight information for one vendor (Open Source preferably, as the code can be reviewed) and then gradually extending this Proof-of-Concept to a framework containing multiple software vendors. Once this level is reached the malware could be used to compromise several UAVs of different manufacturers. A disadvantage of this idea is that it does not provide a solution to an intruding UAV which was not previously targeted by the malware. The malware has always to be installed before in order to be able to take over control.

**Manipulated RC firmware:** Another approach which would allow the attacker to cause damage is a manipulated firmware of the remote control. The remote control is usually connected as a backup device to the UAV although it is not actively used. If the person responsible for maintaining the devices can be tricked into installing a new firmware on the device (using social engineering techniques) the attacker can cause damage to the UAV. Unfortunately, the attack is very limited, because the attacker is not actively involved in the controlling process. The firmware could be a logic bomb, sending descending signals to
the UAV once a predefined threshold is reached.

However, as the work should focus mainly on the UAS, without spending too much time on secondary goals, these approaches were not considered for further study. Instead this thesis has a heavy focus on the UAV itself, especially highlighting its communication channels in order to investigate, compromise and fix vulnerabilities.
APPENDIX B

The following figure depicts the structure of a Remote AT Command defined by Digi International. XBee Remote AT Command packets have to be disassembled and assembled according to this pattern. Other commands can be found accordingly in the documentation [64].

Remote AT Command

Frame Type: 0x17

Used to query or set module parameters on a remote device. For parameter changes on the remote device to take effect, changes must be applied, either by setting the apply changes options bit, or by sending an AC command to the remote.

<table>
<thead>
<tr>
<th>Frame Fields</th>
<th>Offset</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start Delimiter</td>
<td>0</td>
<td>0x7E</td>
<td></td>
</tr>
<tr>
<td>Length</td>
<td>1</td>
<td>0x60</td>
<td>Number of bytes between the length and the checksum</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0x10</td>
<td></td>
</tr>
<tr>
<td>Frame Type</td>
<td>3</td>
<td>0x17</td>
<td></td>
</tr>
<tr>
<td>Frame ID</td>
<td>4</td>
<td>0x01</td>
<td>Identifies this command for correlation to a later response frame (0x07) to this command. If set to 0, no response frame will be sent.</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0x13</td>
<td></td>
</tr>
<tr>
<td>64-bit Destination Address</td>
<td>7</td>
<td>0x02</td>
<td>Set to the 64-bit address of the destination device. The following address is also supported:</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0x00</td>
<td>0x000000000000FFFF - Broadcast address</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>0x40</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>0x11</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>0x22</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>13</td>
<td>0xFF</td>
<td>Set to 0xFFFF.</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>0x7E</td>
<td></td>
</tr>
<tr>
<td>Remote Command Options</td>
<td>15</td>
<td>0x02 (apply changes)</td>
<td>0x02: Apply changes on remote. (If not set, AC command must be sent before changes will take effect.) All other bits must be set to 0.</td>
</tr>
<tr>
<td>AT Command</td>
<td>16</td>
<td>0x02 (h)</td>
<td>Name of the command</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>Command Parameter</td>
<td>18</td>
<td>0x01</td>
<td>If present, indicates the requested parameter value to set the given register. If no characters present, the register is queried.</td>
</tr>
<tr>
<td>Checksum</td>
<td>19</td>
<td>0x05</td>
<td>0xFF - the 8 bit sum of bytes from offset 3 to this byte.</td>
</tr>
</tbody>
</table>

Example: The above example sends a remote command to change the broadcast hops register on a remote device to 1 (broadcasts go to 1-hop neighbors only), and apply changes so the new configuration value immediately takes effect. In this example, the 64-bit address of the remote is 0x0013A200 44401122.

Figure B.1: XBee Remote AT Command Frame, from [64]
The representation of a hard-coded command in the Android APK. The name suggests that this command requests a download of the available waypoints within the storage of the flight computer.

Listing C.1: Android APK - Command construction 2

```java
public void SendDataDownloadWP()
{
    byte[] arrayOfByte = new byte[30];
    for (int i = 0; i < 30; i++)
    {
        if (i >= 30)
        {
            arrayOfByte[0] = 36;
            arrayOfByte[1] = 87;
            arrayOfByte[2] = 73;
            arrayOfByte[3] = 70;
            arrayOfByte[4] = 73;
            arrayOfByte[5] = 82;
            arrayOfByte[6] = 82;
            arrayOfByte[7] = 82;
            SendbyteData(arrayOfByte);
            return;
        }
        arrayOfByte[i] = 0;
    }
}
```
This section shows all commands which have been inferred through reverse engineering.

Table D.1: Commands for flight computer

<table>
<thead>
<tr>
<th>First entry</th>
<th>Second entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>24 57 49 46 49 91 91 91</td>
<td>Parachute Close Position</td>
</tr>
<tr>
<td>24 57 49 46 49 92 92 92</td>
<td>Parachute Open Position</td>
</tr>
<tr>
<td>24 57 49 46 49 75 75 75</td>
<td>Stick Calibration</td>
</tr>
<tr>
<td>24 57 49 46 49 69 69 09 57 09 57</td>
<td>Magnetic Compass - Horizontal Alignment</td>
</tr>
<tr>
<td>24 57 49 46 49 69 69 09 58 09 58</td>
<td>Magnetic Compass - Vertical Alignment</td>
</tr>
<tr>
<td>24 57 49 46 49 93 93 93</td>
<td>Download Trigger Points</td>
</tr>
<tr>
<td>24 57 49 46 49 53</td>
<td>Magnetic Compass - Horizontal Alignment</td>
</tr>
<tr>
<td>01 6D 0C 63 42 80 21 8B BF 01 6D</td>
<td>Upload Waypoint Data (Repeats waypoint until it gets a confirmation that the upload is completed)</td>
</tr>
<tr>
<td>24 57 49 46 49 52 52 52</td>
<td>Verify Waypoint data</td>
</tr>
<tr>
<td>24 57 49 46 49 54 xx xx xx</td>
<td>Waypoint upload confirmation (When upload is finished xx is the amount of waypoints)</td>
</tr>
<tr>
<td>24 57 49 46 49 57 57 57</td>
<td>Auto Landing</td>
</tr>
<tr>
<td>24 57 49 46 49 58 58 58</td>
<td>Auto Takeoff</td>
</tr>
<tr>
<td>24 57 49 46 49 97 97 97</td>
<td>Enable Flightpath (Full Flightpath)</td>
</tr>
<tr>
<td>24 57 49 46 49 98 98 98</td>
<td>Enable Flightpath (One step at a time)</td>
</tr>
<tr>
<td>24 57 49 46 49 7B 7B 7B</td>
<td>Disable Flightpath</td>
</tr>
<tr>
<td>24 57 49 46 49 67 67 XX xx</td>
<td>Target (xx is number of target and repeated once)</td>
</tr>
<tr>
<td>24 57 49 46 49 6A 6A XX XX XX XX XX XX</td>
<td>Change altitude (While X is the overall number of the new altitude)</td>
</tr>
<tr>
<td>24 57 49 46 49 C9 C9 C9</td>
<td>Read one minute of data</td>
</tr>
</tbody>
</table>

Continued on next page
Table D.1 – Continued from previous page

<table>
<thead>
<tr>
<th>First entry</th>
<th>Second entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>24 57 49 46 49 66 66 66</td>
<td>Capture transmitter center point</td>
</tr>
<tr>
<td>24 57 49 46 49 78 78 78</td>
<td>Init Setup</td>
</tr>
<tr>
<td>24 57 49 46 49 79 79 79</td>
<td>Quit Setup</td>
</tr>
<tr>
<td>24 57 49 46 49 94 94 94</td>
<td>Snapshot</td>
</tr>
<tr>
<td>24 57 49 46 49 64 64 64</td>
<td>Zero Gyro</td>
</tr>
<tr>
<td>24 57 49 46 49 68 68 68</td>
<td>Get Params</td>
</tr>
<tr>
<td>24 48 46 4D 52</td>
<td>Params default (+ Get Params)</td>
</tr>
<tr>
<td>24 57 49 46 49 73 73 (50 32 2D 50 40 04 02 41 08 50 2D 14 14 96 5F 1E 32 08 64 83 Fo B1)</td>
<td>Send Params</td>
</tr>
<tr>
<td>24 57 49 46 49 51</td>
<td>POI Fly to target</td>
</tr>
<tr>
<td>24 57 49 46 49 5C</td>
<td>Target Lock</td>
</tr>
<tr>
<td>24 57 49 46 49 55 55 55</td>
<td>Quit target lock</td>
</tr>
<tr>
<td>24 57 49 46 49 56 56 56</td>
<td>Set Home Location</td>
</tr>
<tr>
<td>24 57 49 46 49 8E 8E 8E</td>
<td>Get Mixing Define</td>
</tr>
<tr>
<td>24 57 49 46 49 8A 8A 8A 24 57 49 46 49 8A 8A 8A 24 57 49 46 49 8A 8A 8A 00 00 00 00 00 00</td>
<td>Send Mixing Define</td>
</tr>
<tr>
<td>24 57 49 46 49 8B 8B 8B 24 57 49 46 49 8B 8B 8B 00 00 00 00 00 00</td>
<td>Disable Remote Control</td>
</tr>
<tr>
<td>24 57 49 46 49 8C 8C 8C 00 00 00 00 00 00</td>
<td>Enable Remote Control</td>
</tr>
<tr>
<td>00 00 00 00 00 24 00 00 00 00 00 00 00 00</td>
<td>Unlock Motors</td>
</tr>
<tr>
<td>00 00 00 00 00 00 00 00 00 00 00 00 00 00</td>
<td>Preset PTZLock</td>
</tr>
<tr>
<td>24 57 49 46 49 8D 8D 8D 57 49 46 49 8D 8D 8D FA 00 00 00 00 00 00</td>
<td>Video Recording Start/Stop</td>
</tr>
<tr>
<td>24 57 49 46 49 8A 8A 8A 24 57 49 46 49 8A 8A 8A 00 00 00 00 00 00</td>
<td>Stadicam Alignment</td>
</tr>
<tr>
<td>24 57 49 46 49 5A 5A 5A</td>
<td>Captpure Roll</td>
</tr>
<tr>
<td>24 57 49 46 49 59 59 59</td>
<td>Capture Pitch</td>
</tr>
</tbody>
</table>
APPENDIX E

This section shows the structure of data from the flight computer towards the tablet running the flight planning software. The initial command starts with 24 53 54 50, afterwards follows data.

```
24 53 54 50 63 25 51 42 F6 7D 9B 40 00 00 00 00 00 00 00 00 00 00 00 00 03 0F 06 0E 12 18 15 01 6E 88 6E B8 8A 93 91 C1 00 00 DE 00 3F B8 00 0B F8 FF 9B 00 00 00 00 FF C8 26 09 00 C3 FF F8 FF FF FF FF FF FF FF FF FF FF DF 0C 87 01 00 00 16 00 00 00 96 96 96 00 32 00 04 F5 FF 17 00 00 00 00 7B 02 3A 9E 71 00 00 00 00 00 00 00 00
```

This data stream includes the position of the joysticks on the remote control and other parameters to display within the App.
APPENDIX F

This section shows the source code of the developed Proof-of-Concept.

Listing F.1: Proof-of-Concept - Source Code

```python
# coding: utf-8

from xbee import ZigBee
import serial
import binascii

ser = serial.Serial('/dev/ttyUSB0', 115200)
xbee = ZigBee(ser)

broadcast = '\x00\x00\x00\x00\x00\x00\xFF\xFF'
addr_of_first_device = ''
short_addr_of_first_device = ''
addr_of_second_device = ''
short_addr_of_second_device = ''

# Get my own XBee address
xbee.at(command='SH')
response = xbee.wait_read_frame()
SH = response['parameter']
print(SH)

xbee.at(command='SL')
response = xbee.wait_read_frame()
SL = response['parameter']
print(SL)

# Change Destination addresses
xbee.remote_at(dest_addr_long=broadcast, command='TF', parameter=SH)
xbee.remote_at(dest_addr_long=broadcast, command='TL', parameter=SL)

# Persist changes

# First we need to remember the addresses of both chips
while (addr_of_first_device=='' or addr_of_second_device==' '):
    print('Entered device packets')
    response = xbee.wait_read_frame()
    print(response)

# Program crashes if rf.data is not filled, therefore check first for right frame
data = response['id']
if data=='rx':
```

73
data = response['rf_data']
print 'RF data'
print data

# safe addresses of sending XBee devices
if (addr_of_first_device==''):
    addr_of_first_device = response['source_addr_long']
    short_addr_of_first_device = response['source_addr']
elif (addr_of_second_device=='' and addr_of_first_device!=
    response['source_addr_long']):
    addr_of_second_device = response['source_addr_long']
    short_addr_of_second_device = response['source_addr']

print '--------------------------------------'
print '--------------------------------------'
print 'Finished initialization'
print 'Address of First device: '
print addr_of_first_device
print 'Address of Second device: '
print addr_of_second_device
print '--------------------------------------'
print '--------------------------------------'

# Continuously read and print packets
while True:
    try:
        print '--------------------------------------'

        response = xbee.wait_read_frame()
        print response

        # Program crashes if rf_data is not filled, therefore check first
        # for right frame
        data = response['id']
        if data=='rx':
            data = response['rf_data']
            print 'RF data'
            print data

            # Forward data to the other connected XBee
            if (response['source_addr_long']==addr_of_first_device):
                print 'Adress'
                print addr_of_second_device
                xbee.tx(dest_addr_long=addr_of_second_device, dest_addr=
                        short_addr_of_second_device, data=data)
            else:
                xbee.tx(dest_addr_long=addr_of_first_device, dest_addr=
                        short_addr_of_first_device, data=data)

            # Wait for answer of the receiving XBee
            response = xbee.wait_read_frame()
            print 'Answer of receiving XBee: '
            print response
    except KeyboardInterrupt:
        break
ser.close()
BIBLIOGRAPHY


[48] Electronic Communications Committee (ECC). The european table of frequency allocations and aplications in the frequency range 8.3 kHz to 3000 GHz (ECA TABLE). 2014.


[58] Electronic Communications Committee (ECC). *ECC RECOMMENDATION (03)04. With regard to forbidding the placing on the market and use of GSM Jammers in the CEPT member countries*.


[81] Build Your Own Drone Ltd. Xbee Telemetry kit 2.4 GHz. 2015. URL: http://www.buildyourowndrone.co.uk/xbee-telemetry-kit-2-4-ghz.html (Retrieved 05/18/2015).

