DEGREE PROJECT IN COMPUTER ENGINEERING,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2020

Penetration testing a civilian drone

Reverse engineering software in search for security vulnerabilities

VIKTOR EDSTRÖM

ELDAR ZEYNALLI

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Penetration testing a civilian drone

VIKTOR EDSTRÖM, ELDAR ZEYNALLI

Bachelor in Computer Science
Date: June 8, 2020
Supervisor: Robert Lagerström
Examiner: Pawel Herman
School of Electrical Engineering and Computer Science
Swedish title: Penetrationstestning på civil drönare

Abstract

Civilian drones have become more popular in recent years. As an IoT (Internet of Things) device full of state of the art technologies, its level of security is highly relevant. That is why we chose to take a look at the previous research done in the field to see how the attack surface of civilian drones looks. This revealed that drones are especially vulnerable to GPS and WiFi-based attacks. Furthermore, there have also been vulnerabilities discovered in the applications used by different civilian drones. We took a closer look at a certain drone model, DJI Mavic 2 Zoom, by analyzing its internals and reverse engineering certain parts of its software architecture to see what attacks it could be vulnerable to. Our research revealed that the drone uses a proprietary protocol dubbed Ocusync. This means it is not vulnerable to the same attacks as drone models that use WiFi. However, the drone could still be as vulnerable to GPS spoofing attacks. Through reverse engineering, we also discovered a vulnerability in the software of the drone, which has been reported to the manufacturer.

Sammanfattning

Civila drönare har under de senaste åren blivit mer populära. Som en IoT-enhet (Internet of Things) full av modern teknik är dess säkerhetsnivå mycket relevant. Det är därför vi valde att titta på den tidigare forskningen som har gjorts på området för att se hur attackytan på civila drönare ser ut. Detta avslöjade att de är särskilt sårbara för GPS- och WiFi-baserade attacker. Dessutom har det också upptäckts sårbarheter i applikationerna som används av olika civila drönare. Vi tittade närmare på en viss drönarmodell, DJI Mavic 2 Zoom, genom att analysera dess intern arkitektur och dekompilera vissa delar av programvaruarkitekturen för att se vilka attacker den kan vara sårbar för. Vår forskning avslöjade att drönaren använder ett proprietärt protokoll som kallas Ocusync. Det här innebär att drönaren inte är sårbart för samma attacker som modeller som använder WiFi. Däremot, kan drönaren vara lika sårbar för GPS-spoofing attacker. Med hjälp av dekompilering, upptäckte vi också en sårbarhet i drönarens mjukvara, som har rapporterats till tillverkaren.

Acknowledgements

We want to thank our supervisor Robert Lagerström as well as MSAB for providing us with various equipment.

Contents

Chapter 1

Introduction

IoT devices are becoming increasingly popular with every passing day. One such device is drones, more specifically civilian drones. As more technological advances have been made, drones have, in recent years, become widely available and purchasable by anyone. The civilian drone market is a rapidly growing market that is expected to have a very major role in future [1].

Drones are essentially highly sophisticated computers with flight capabilities that can be controlled remotely. The fact that drones are easily accessible to consumers despite being a relatively new technology brings their safety into question. There is a lot to be researched in the field, to widen our understanding of how the device works as well as increase the sphere of knowledge around it.

The largest manufacturer of civilian drones is DJI, and they have released multiple series such as Mavic and Phantom, each series further having different models. That is why we focused on investigating a DJI-manufactured drone - DJI Mavic 2 Zoom.

1.1 Problem statement

In our thesis, we aimed to research how the attack surface on civilian drones generally look, and what attacks they are vulnerable to by doing a review of previous attacks conducted on civilian drones and vulnerabilities discovered by security researchers. We tried to investigate what entry points malicious attackers could use to attack civilian drones.

We also aimed to take a closer look at a certain high-end drone model – DJI Mavic 2 Zoom through reverse engineering, to gather information on its internal architecture to form an idea of its attack surface. We also searched for vulnerabilities in the applications used by the drone.

1.2 Scope

Due to the limited time frame of this research, we chose to focus primarily on a general literature review of previous drone attacks and reverse engineering parts of a certain drone's (DJI Mavic 2 Zoom) software architecture.

We did not have the necessary tools or the time to confirm whether any vulnerability we have discovered through reverse engineering in the software system of the drone is also present in other similar DJI drone models. Similarly we have not verified any of the vulnerabilities we discovered through literature review.

Chapter 2

Background

2.1 Drones – Unmanned Aerial Vehicles

In order to get a sense of how an attack on a civilian drone would look, we need to understand how drones work and what software and hardware technologies they use.

Drones, also referred to as UAVs (Unmanned Aerial Vehicles), are unmanned aircrafts. They are equipped with various tools such as GPS, laser and infrared cameras, depending on what type of application the drone has. One of those applications is the military use of drones, which has also had the most development. In the last years, a lot of the technologies used on military drones have been made available for other applications of drones as well, such as business and consumer/civilian applications [2].

Their business application is only expected to increase with companies such as Amazon, Google, and Facebook, showing interest in using them for purposes like order delivery and providing broadband connectivity [1].

Following the developments in software technologies, the civilian drone market has been rapidly developing in the last years as well. The biggest civilian drone manufacturer is DJI, a China-based company. [3]

DJI Mavic 2 Zoom is one of their latest drone models, and like all drones, it has two main components - the drone itself and the controller to communicate with it. A smartphone can also be mounted to the controller to receive the camera feed from the drone. [4]

2.2 Reverse engineering

Reverse engineering is the process of deconstructing some objects to reveal knowledge about it, such as its architecture and functionality [5].

This process can be applied to software, e.g., when a program's source code is not available. Compilers are able to transform high-level languages into lower-level languages, such as binary machine language. By analysing the compiler's output it is possible to reveal information about the program. This is often refereed to as binary reverse engineering [5].

A disassembler is a program that transforms machine language into assembly language, a more readable language describing the underlying machine language. A decompiler is a similar tool but tries to transform machine language into high-level source-code [5].

Ghidra is an open source software reverse engineering tool that includes a variety of tools for decompiling, disassembling, and more for analysing compiled code [6]. IDA Pro is a similar tool for software analysis that has been developed by Hex Rays, is a commercial product [7].

2.3 Attack surface

A variety of definitions for the term attack surface exist, and researchers have different views on what the term means [8]. A commonly used definition is: *"union of code, interfaces, services, protocols, and practices available to all users, with a strong focus on what is accessible to unauthenticated users"* [9] [8]. We'll be using the term to refer to the sum of entry points – parts of the device through which a malicious attacker could attack the drone and cause damage.

2.4 STRIDE

STRIDE threat modeling was used to categorize the results. STRIDE is a threat model developed by Praerit Garg and Loren Kohnfelder at Microsoft. Threat modeling is a structured way to identify and prioritize potential threats to a system in order to prevent said possible vulnerabilities [10].

STRIDE is an acronym that stands for six different types of threats: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege. These threats violate the desirable properties of a secure system such as: Authenticity, Integrity, Non-repudiability, Confidentiality, Availability. A short description of these can be observed in figure 2.1.

Threat Violated	**Violated Property**	**Description of Attack**
Spoofing	Authenticity	Successfully identified as a legitimate user by falsifying data.
Tampering	Integrity	Tampering with memory, network, disk et. cetera
Repudiation	Non-repudiability	Not claiming responsibility for some action performed.
Information Disclosure	Confidentiality	Giving out information to unauthorized entity.
Denial of Service	Availability	Making a system unavailable by malicious requests.
Elevation of Privilege	Authorization	Allowing unauthorized users elevated privileges, allowing functionalities that are not intended by the system.

Figure 2.1: STRIDE categories [11]

2.5 Terminology

In this section we will explain some of the terms used throughout the paper.

2.5.1 Cyclic redundancy check

Cyclic redundancy check – CRC is a method that calculates a checksum for some piece of data, e.g., data received via transmission or storage. This method can be used for error-detection in case transmission errors occur [12].

2.5.2 Advanced Encryption Standard

The Advanced Encryption Standard – AES is a block cipher, an encryption function for a fixed-size block of data. It supports multiple key-sizes, such as 128-, 256-, 512-bits [13].

2.5.3 Cipher Block Chaining

Cipher Block Chaining – CBC is a widely used block cipher mode, meaning a function that encrypts one or multiple blocks of data via a block-cipher (such as AES). CBC applies a xor with the previous block's ciphertext on each plaintext before applying the block-cipher to said ciphertext. For the first block, an IV – initialization vector is used [13].

2.5.4 Digital signatures

Digital signature is a mathematical scheme for verifying the authenticity of some data via asymmetric cryptography. A signer is able to sign data with a private key, with which a recipient can verify the data's authenticity and integrity [14].

2.5.5 ARM architecture

ARM is a RISC – reduced instruction set computing architecture, it is the most widely used ISA – Instruction Set Architecture with over 160 billion ARM processors produced [15]. The ISA comes in multiple versions, supporting various computer architecture bit widths (e.g. 16-, 32- and 64-bit) depending on the version.

2.5.6 TEE and TrustZone

A TEE – Trusted Execution Environment is a secure area inside a processor, running in parallel with a main operating system. It also guarantees that data and code which have been loaded into the TEE are protected, in respect to confidentiality and integrity [16].

TrustZone is a security extension offered by ARM, which allows possible TEE implementations. TrustZone achieves this by having a *secure world* and *in-secure world*, allowing for restriction of memory and peripherals only to certain worlds [17]. Switching between the different worlds is done via a specific processor instruction SMC – Secure Monitor Call [18].

2.5.7 SoC

A SoC – System on a chip is an integrated circuit that integrates most if not all components required for an electronic system. This includes a processor, memory and possibly various peripherals all inside a single chip [19].

2.5.8 SDR

A radio is a system that transmits data wirelessly. Many radio implementations rely on circuits with fixed functionality, meaning the functionality of the radio cannot change once the circuits are produced. SDRs – Software Defined Radios aim to resolve this issue, allowing for control of how radio hardware operates based on software, meaning functionality of the radio could be altered via software changes [20].

Chapter 3

Methodology

3.1 Literature review

To examine the attack surface of civilian drones, we did a general literature-review to investigate what attacks have previously been conducted on civilian drones. Databases and search engines that have been used to find articles include Google Scholar and KTH Primo. Terms such as "drone cybersecurity", "civilian drone vulnerabilities" and more were used for searching for relevant articles in the databases.

Grey literature (such as FCC documents) was also valued as it can be used to, among other things, gather information about different companies and their products.

3.2 Penetration-testing DJI Mavic 2 Zoom

We reverse engineered various software components used by the DJI Mavic 2 Zoom, such as binary programs, binary structures, and data over various means of transport to gather information about the internal system of the drone.

When penetration-testing DJI Mavic 2 Zoom's software components, we mainly chose to focus on parts of the system that various types of data could be transmitted through. We hypothesized that those parts of the system were more likely to be an entry point for a malicious attack.

More specifically, we used Ellisys USB Explorer 200, a hardware device capable of sniffing high-speed USB traffic to gather USB traffic sent between the remote controller and a computer. This was done to get an idea of how the USB communication between the devices works.

We used tools such as IDA Pro and Ghidra to dissect the functionality of the binary programs in the device's system, through which we discovered more information about its internal software architecture. By doing so, we could also search for previously-undiscovered vulnerabilities in various parts of the software system. The information gathered through reverse engineering can be verified with the use of similar tooling.

As the code-base for the applications used by the DJI Mavic 2 Zoom was very large, we decided to primarily focus on the communication between the drone and the remote controller as well as between the remote controller and software responsible for firmware upgrades. In order to determine which binary programs were relevant for these tasks, the programs were filtered by searching for text sequences, e.g., to determine which programs handled firmware upgrades, we searched for the file extension used by firmware files.

We also used WiFi networking tools, Aircrack-ng and Airodump-ng, to attempt to capture network packets sent between the controller and the drone to analyse how vulnerable DJI Mavic 2 Zoom is to WiFi attacks. We did this by doing a scan of the wireless network to search for relevant access points. This method has previously been successfully employed by other researchers [1].

We used STRIDE, a popular threat modeling framework, to categorize the vulnerabilities mentioned throughout the thesis to give an overview of what consequences different vulnerabilities could have. STRIDE models are usually made before penetration testing itself to serve as a tool to help identify what parts of the system could be vulnerable, so our way of utilizing it is unusual.

Chapter 4

Results

4.1 Vulnerabilities of civilian drones

4.1.1 WiFi-based drone vulnerabilities

Our research into previous attacks on drones revealed that civilian drones often use the WiFi network for communication between the controller and the drone, which is vulnerable to various malicious attacks [1].

Tools such as Skyjack have been used to wirelessly compromise civilian drones such as various Parrot drones and the DJI Phantom 2 mid-flight by sending deauthentication frames to the access point (remote controller) and then broadcasting itself as the access point for the drone giving the attacker control over the device [21].

Deauthentication attack allows for another primitive – Denial of Service (DoS). By repeatedly sending deauthentication frames to the access point, the drone is no longer able to connect to the controller [21].

4.1.2 GPS spoofing

Civilian drones use GPS for navigation, which exposes another entry point through which GPS spoofing attacks can be performed. As GPS signals are unencrypted, they can be easily spoofed or jammed. If a drone is trying to reach a certain position, by following a certain command, a fake GPS signal can be sent to overwrite the real GPS signals from the satellites, making it possible to get the drone to follow the attacker's desired path. This effectively gives the attacker complete control over the drone [1].

Manipulating the drone's receiver's reported position, velocity and time can be done with even low-cost technology [22]. This has been demonstrated by researchers from University of Texas at Austin, who sent out spoofed GPS signals that aligned with the legitimate GPS signals that the drone received from the satellites. By doing so, they could overpower and overwrite the legitimate GPS signals, getting control of the drone's velocity and position [1].

4.2 Wifi penetration attemps

We tried using wireless network tools to capture raw packets sent between the DJI Mavic 2 Zoom remote controller and the drone. We could not find the access point by scanning the wireless network, which made us realize DJI Mavic 2 Zoom might not even be using WiFi. This got us searching, and we found out from various sources that high-end DJI drones do, in fact, not use WiFi for transmission. The radio communication between the remote controller and the drone in the system is instead managed by an in-house developed SDR implementation dubbed Ocusync [23] [4].

4.3 Reverse engineering DJI Mavic 2 Zoom

4.3.1 DJI Mavic 2 Zoom internals

We discovered that both the DJI Mavic 2 Zoom remote controller and the drone run on 32-bit ARM processors through looking at FCC documents.

We also found out that the drone and remote controller both house a Leadcore L1860C processor based on the ARM Cortex-A7 core. In addition to the Leadcore processor the drone also features a Atmel ATSAME70N19 processor [24] [25].

Reverse engineering the firmware upgrade process revealed that both respective systems and chips utilize ARMs TEE – Trusted Execution Environment, TrustZone.

The implementation of Trust-Zone DJI uses in the Mavic 2 Zoom is OP-TEE – Open Portable Trusted Execution Environment, initially developed by ST-Ericsson and later on open-sourced. We found out that one use-case of OP-TEE in the DJI system is firmware upgrades. Firmware files are validated, decrypted, and flashed from the TEE.

```
int dji_fw_load(const char *a1, int a2, int a3) {
  /* ... */
  v6 = TEEC_InitializeContext(0, v30);
  if ( !v6 )
  {
    v8 = TEEC_OpenSession(v30, &v20, &v26, 0, 0, 0, &v21);
    if ( v8 )
    {
      printf("TEEC_Opensession failed with code 0x%x origin 0x%x",
             v8, v21);
      TEEC_FinalizeContext(v30);
      return v8;
    }
   /* ... */
}
```

Figure 4.1: DJI firmware function utilizing TEE

Reverse engineering build files revealed that the Mavic 2 Zoom remote controller and the drone both run on a version of Android 4.4.4 featuring various custom written binary programs for communication and interaction with hardware. Both devices use U-Boot, a commonly used boot loader among embedded devices.

Given the DJI Mavic 2 Zooms feature list containing a return-to-home functionality, we expected that the device includes GPS functionality. We confirmed this hypothesis by looking at data and services inside the device.

```
/dev/mpu                    u:object_r:gps_device:s0
/dev/mpuirq                 u:object_r:gps_device:s0
/system/vendor/bin/gpsd     u:object_r:gpsd_exec:s0
/vendor/bin/gpsd            u:object_r:gpsd_exec:s0
/data/gps(/.*)?             u:object_r:gps_data_file:s0
```

Figure 4.2: String search on the file system revealing gpsd – a GPS service daemon

4.3.2 DUML protocol

Through reverse engineering packets sent between the USB traffic, we discovered DUML – DJI Universal Markup Language, a proprietary protocol is used throughout the DJI system. By auditing the system, we discovered DUML could be transmitted through other means, such as Ocusync. It was also discovered that the protocol is used for various tasks such as firmware upgrades, the transmission of video, control of aircraft et cetera. It is also able to send data in encrypted form.

We used Ellisys USB Explorer 200 to analyse the packets, which revealed that every DUML packet starts with an identifier – the byte value '0x55' in hexadecimal, followed with two bytes containing the length of the package being sent and the version of the DUML packet. With every packet having a length limit of 0x3ff bytes. If the length of the packet being sent is more than 255 bytes, the version_msb is being used to store additional data by setting its two lowest bits.

A sequence number is in place to assert that packets are sent in sequential order. This number increments for every package sent. CRC checksums are in place to assert that the data has not been tampered during transmission. Before being sent, the packet is packed in little endianness.

```
struct duml_packet {
  uint8_t identifier;
  // LSB of length
  uint8_t len;
  /*
  * MSB of length and DUML version
  * ((version << 2) | (length >> 8) & 0x3
  */
  uint8_t version_msb;
  uint8_t crc8;
  uint8_t src;
  uint8_t target;
  /*
  * sequence number, increments
  * per packet sent
  */
  uint16_t seq_num;
  uint8_t cmd_type;
  uint8_t cmd_set;
  uint8_t cmd_id;
  uint8_t data[N];
  uint16_t crc16;
  /* ... */
};
```

Figure 4.3: Psuedocode for DUML packet structure

4.3.3 DUMLRacer

Further research into DUML protocol also led us to the finding a previous vulnerability discovered by Justin Case, and published on Github. This vulnerability, called DUMLRacer, was a vulnerability affecting various DJI Drones and remote controllers such as DJI Mavic Pro, Phantom 4 and Inspire 2 up to and including version 01.04.0200. This vulnerability does not affect the DJI Mavic 2 family of drones.

The vulnerability is a race-condition vulnerability in the firmware-upgrade process, where firmware-files could be tampered with mid-upgrade. This allows an attacker to locally achieve code-execution on the vulnerable devices [26].

4.3.4 DJI Firmware structure

Managing the firmware for the DJI Mavic 2 Zoom drone and remote controller is possible via the desktop application ’DJI Assisant 2 For Mavic’ available for Windows / Mac OS or the mobile application ’DJI GO 4’ available on iOS / Android. There is also functionally for the remote controller to upgrade the drone wirelessly via Ocusync.

By reverse engineering files responsible for firmware upgrades, using tools such as Ghidra and IDA Pro, we discovered that during the upgrade a signature file gets sent over to the drone/remote controller which contains an XML file containing e.g., filename and hashes for every firmware file which is about to be sent. Both the signature and firmware files are digitally signed by DJI and verified on respective devices. Some of these files can be encrypted.

On various DJI drones and controllers, the firmware is both encrypted and digitally signed. This is to prevent unauthorized firmware files being flashed onto the SoC – system on a chip.

There have been successful attempts at desoldering the underlying Rockchip ARM SoC on the DJI Phantom 4 Pro and unpacking its firmware [21].

Information about the binary structure for firmware files was discovered through reverse engineering procedures that handle and decrypt firmware files. Both the signature and firmware files are packed in a binary structure developed by DJI, with the files starting with the magic constant ’IM*H’.

The encryption key that is used is determined by the entry ’enc_key’. It is also possible for the structure not to be encrypted at all, which is the case for the initial signature file.

If the data is encrypted, a key is derived by decrypting the data inside ’scram_key’ with the key specified in ’enc_key’. The decryption is performed by applying the symmetric encryption scheme 128-bit AES with CBC as its block cipher and having a zero initialization vector.

```
struct fw_header{
    char magic[4];
    uint32_t version;
    uint32_t len;
    uint32_t idk;
    uint32_t header_len;
    /* ... */
    char auth_key[4];
    char enc_key [4];
    uint8_t scram_key[16];
    uint32_t chunk_num;
    /* ... */
};
```

Figure 4.4: Psuedocode for firmware header structure

The firmware data is divided into different chunks, with the number of chunks being specified in ’chunk_num’. A header for every chunk is stored after the firmware header containing an offset into the file where its data is stored (after header and signature data) and the chunk’s size.

```
struct fw_chunk {
    char name[4];
    uint32_t offset;
    uint32_t size;
    uint32_t attrib;
    uint64_t address;
    uint32_t reserved;
};
```

Figure 4.5: Psuedocode for firmware chunk

By using the previously derived key (from the 'scrum_key') it is possible to decrypt the chunks of data with 128-bit AES with a zero initialization vector. With the firmware files decrypted, it is possible to reverse engineer the programs that will be run on the drone/remote controller as well as the SDR firmware.

The decryption and verification of signatures of firmware files is performed inside the TEE, which keeps the integrity of the operating system inside the insecure-world. Therefore it is only necessary to store possible cryptographical keys inside the secure-world. However, this practice is not consistent across the different processors used across the system; the remote controller stores the keys necessary for firmware decryption in such a way that they are accessible to the insecure-world.

```
int read_key_material(int offset, unsigned int sz) {
  int len;
  int fd;
  void *addr;
  len = sz;
  if ( sz > 0x1000 )
    return -1;
  if ( !sz )
    len = 4096;
  fd = open("/dev/mem", 0x101000);
  addr = (void *)mmap(0, len, 1, 1, fd, offset);
  if ( addr == (void *)-1 ) {
    printf("failed to mmap fd: %d to %p\n", fd, offset);
  } else {
    memcpy_chk(dst, addr, len, 4096);
    munmap(addr, len);
  }
  close(fd);
  ...
}
```

Figure 4.6: Material used to derive encryption key is accessible via /dev/mem

4.4 Categorizing vulnerabilities

In this section, we categorize various vulnerabilities mentioned throughout the thesis based on the type of threat they are; and the amount of damage an attacker could inflict through the vulnerability.

	S	**T**	**R**	**I**	**D**	**E**
GPS spoofing	**x**					
Wifi deauthentication	**x**				**x**	
DUMLRacer						**x**

Figure 4.7: STRIDE categories [11]

Spoofing the legitimate signals from satellites gives the attacker complete access to the drone's velocity and position. This is a spoofing threat, where the attacker pretends to be the legitimate GPS transmitter and tricks the device.

Wifi deauthentication attacks on drones make the remote controller lose the connection with the drone, which is a Denial of Service threat. By doing so, the attacker can also trick the drone into thinking that the attacker is the original access point. This is a spoofing threat, as by doing so, the attacker can have complete control of the device. To sum it up, depending on how the attack is utilized, different outcomes can be achieved.

DUMLRacer vulnerability affecting various DJI drones enabled tampering of firmware files, which gives an attacker the access to local code-execution on devices affected by this vulnerability. This is an Elevation of Privilege threat.

Chapter 5

Discussion

5.1 WiFi

Previous research in the field clearly demonstrates that GPS and WiFi communications are some of the biggest sources of vulnerability in civilian drones. A lot of civilian drones use WiFi, and there are many ways WiFi network can be exploited. This makes civilian drones quite vulnerable. There are publicly available tools that can be used to compromise different civilian drones. The STRIDE modeling shows that these types of attacks can be devastating as they can be used to achieve several critically dangerous outcomes as WiFi vulnerabilities are both Denial of Service and Spoofing threats. This tells us that WiFi is a big part of the attack surface on civilian drones, being that it is a vulnerable entry point that can be exploited relatively easily.

This piece of information was what prompted us to try the same methods used by other researchers to penetration test DJI Mavic 2 Zoom through that entry point. But as we discovered, DJI Mavic 2 Zoom, did not use any WiFi protocol for the communication between the drone and remote controller. Using an in-house developed SDR implementation, in this case, Ocusync, means the drone is not vulnerable to the attacks civilian drones using WiFi technology are. The entry point no longer exists, and can therefore not be exploited. This could mean the drone is less vulnerable to deauthentication threats. The fact that an in-house system is used in and of itself makes the level of entry for attacks higher. Once we found out DJI Mavic 2 Zoom used an SDR, we were eager to examine it but were unable to do so, because it required hardware, SDR receiver and transmitter, that we did not have access to either. Similarly, a malicious attacker trying to exploit that entry point would need hardware that might not be readily available. On the other hand, having a proprietary solution could mean fewer individuals are actively auditing the SDR implementation, and it might be less prone to security checks. For the most part, we think in-house developed SDR implementations might prove to be a valuable replacement for WiFi in civilian drones to improve their level of security.

5.2 GPS

Another vulnerable entry point and a part of the attack surface, is GPS. GPS signals are unencrypted, and that can be exploited by malicious attackers. As has been demonstrated by the University of Texas at Austin on a civilian drone, it is possible to execute a GPS spoofing attack and ultimately get full control of the drone's position and velocity.

We were fairly certain DJI Mavic 2 Zoom used GPS system as well, but we looked at the services running inside the device to be sure. It was no surprise that GPS services were running on the device. This meant DJI Mavic 2 Zoom was just as vulnerable to GPS spoofing attacks as other civilian drones. We did not have the time or resources to confirm it on DJI Mavic 2 Zoom, but as the source of the vulnerability is the GPS system itself, we think it is fair to assume that the attack would be successful. Better, encrypted, alternatives to GPS could be used to ensure the civilian drones' security from such threats.

5.3 Attack surface via DUML

As we discovered through reverse engineering, the proprietary communication protocol – DUML is absolutely central to the DJI Mavic 2 Zoom's system. Being that it is transmitted during Ocusync communication, USB traffic, firmware upgrades, and more, it is safe to say that DUML constitutes a part of the attack surface of not only DJI Mavic 2 Zoom, but also other DJI drones using the protocol. Understanding said protocol requires reverse engineering traffic as well as the programs which parse the data. If a vulnerability in the protocol is found, it could be absolutely critical for the device. A malicious attacker could conduct a number of attacks on the device with the help of the vulnerability.

5.4 Software vulnerabilities on DJI products

Applications using DUML protocol onboard DJI drones have been shown to be vulnerable to possible attacks, like the DUMLRacer vulnerability. It is an

Elevation of Privilege threat, but vulnerabilities affecting drones which are only accessible via physical access, such as DUMLRacer, could be seen as less valuable compared to one triggerable remotely. Executing such an attack would require physical access to the drone.

Nevertheless, having a device capable of running arbitrary code also allows for further tampering of the device. A researcher could use such a setup to debug various tasks on the drone, possibly making vulnerability research tasks simpler by being able to debug pieces of code and interacting with hardware. One such use case could be potentially using the onboard DJI SDR instead of opting to use expensive hardware. Vulnerabilities resulting in local privilege escalation could also be used to remove drone restrictions set by the manufacturer, such as no-fly zones.

With the amount of code running on the DJI devices, it is not surprising to find software vulnerabilities and bugs that can serve as an entry point.

During our research, a previously undiscovered vulnerability was discovered affecting the DJI Mavic 2 Zoom. This vulnerability has been sent to DJI's Security Response Center and is being resolved. We were unable to verify if the vulnerability affects other DJI products than the DJI Mavic 2 Zoom, which was the only drone at disposal during this research. One can assume some DJI products share very similar code-base and hardware features to speed up product development, meaning the vulnerability would be present in other drone models as well.

DJI does not only offer civilian drone solutions but enterprise and military solutions as well. These solutions could possibly differ more in software and hardware from the civilian solution, which has been analysed.

It is also important to mention that DJI as a company seems to care about the security of their devices, as they have an active bug bounty program encouraging security researchers to discover and report such vulnerabilities to make their devices more secure. They were also quick to respond to us regarding the vulnerability we discovered, so it is safe to say that they are serious about the security of their devices.

5.5 OP-TEE

Further research could include researching more extensively about the drone's interaction with the Trusted Execution Environment to better get an understanding of the DJI system and how it interacts with the TEE. We are currently unaware of how DJI's TEE implementation differs from the mainline OP-TEE; this would require further research.

Attacking the TEE from the in-secure world could give a malicious attacker the tools to install malicious firmware on a victim's device, thus breaking the integrity of the operating system inside the insecure-world.

5.6 DJI Firmware structure

To conduct our research, we had to form an understanding of how the DJI's firmware files were structured. These files were encrypted, so we were not able to extract programs. Nevertheless, we chose to include our findings on the firmware structure in this paper, to increase the sphere of knowledge and perhaps aid further research, as well as serve as an example of the DJI system interacting with the TEE.

5.7 Categorizing vulnerabilities with STRIDE

STRIDE models are usually used to aid with penetration testing. They are set up to help with finding threats to the system. We opted for doing something unusual, using the model to categorize the vulnerabilities affecting different civilian drones based on the type of threat they exposed the device to. This was great for comparison purposes, but it also meant that we had too many things to investigate. If we had done modeling beforehand, we could've chosen a part of the system or a type of threat to focus on. Instead, we jumped into exploring the system at large with very little guidance and prior knowledge about the system.

We also did our penetration-testing parallel to the literature review, which meant a lot to focus on. In retrospect, it would have been a lot easier on us to focus on a certain contained part of the system, but that would not have gotten us as extensive picture of the attack surface. Focusing on many different parts of a large system means that we may have overlooked some things. However, the software vulnerability we discovered through reverse engineering has been confirmed to exist by DJI.

Chapter 6

Conclusion

The attack surface on civilian drones mainly consists of WiFi entry point. The vulnerabilities a WiFi network solution expose are public and quite easily reachable, making them critical for the security of the devices as an attacker could easily replicate an attack previously done on other hardware with the same or similar WiFi implementation. This makes civilian drones vulnerable to WiFi deauthentication attacks which could give a malicious attacker complete control over the device. Meanwhile, DJI Mavic 2 Zoom uses an in-house SDR implementation instead of a standard WiFi implementation, which has a higher level of entry.

Another entry point and a part of the attack surface is the GPS used by civilian drones, which, being unencrypted, makes the device vulnerable to spoofing threats. DJI Mavic 2 Zoom also uses GPS and could be vulnerable to GPS spoofing as well.

Previous research has shown that there can be critical bugs in the applications used by civilian drones, compromising the device's security. This means that software-level vulnerabilities could serve as an entry point for a malicious attack and are a part of the attack surface. However, these vulnerabilities require the malicious attacker to have physical access to the device.

Our reverse engineering attempts revealed a vulnerability in DJI Mavic 2 Zoom, which has been reported to the manufacturer.

Bibliography

[1] Edwin Vattapparamban et al. “Drones for smart cities: Issues in cybersecurity, privacy, and public safety”. In: *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE. 2016, pp. 216–221.

[2] Dronezone. *How Do Drones Work And What Is Drone Technology*. `https://www.dronezon.com/learn-about-drones-quadcopters/what-is-drone-technology-or-how-does-drone-technology-work/`. [Online; accessed 8/6/2020].

[3] Technode. *World’s top drone seller DJI made $2.7 billion in 2017*. `https://technode.com/2018/01/03/worlds-top-drone-seller-dji-made-2-7-billion-2017/`. [Online; accessed 8/6/2020]. 2018.

[4] DJI. *Mavic 2*. `https://www.dji.com/se/mavic-2`. [Online; accessed 8/6/2020].

[5] Eldad Eilam. *Reversing: Secrets of Reverse Engineering*. eng. 1st ed.. 2005. ISBN: 9780764597688.

[6] NSA. *Ghidra*. `https://www.nsa.gov/resources/everyone/ghidra/`. [Online; accessed 8/6/2020].

[7] Hex-Rays. *IDA Pro – Hex Rays*. `https://www.hex-rays.com/products/ida/`. [Online; accessed 8/6/2020].

[8] Christopher Theisen et al. “Attack surface definitions: A systematic literature review”. eng. In: *Information and Software Technology* 104 (2018), pp. 94–103. ISSN: 0950-5849.

[9] Michael Howard. “Attack surface: Mitigate security risks by minimizing the code you expose to untrusted users”. In: *MSDN Magazine (November 2004), http://msdn. microsoft. com/en-us/magazine/cc163882. aspx* (2004).

[10] Wenjun Xiong and Robert Lagerström. “Threat modeling – A systematic literature review”. eng. In: *Computers Security* 84 (2019), pp. 53–69. ISSN: 0167-4048.

[11] Elsa Dahlman and Karin Lagrelius. *A Game of Drones: Cyber Security in UAVs*. eng ; swe. KTH, School of Electrical Engineering and Computer Science (EECS), 2019.

[12] Yanbin Zhang and Qi Yuan. “A multiple bits error correction method based on cyclic redundancy check codes”. eng. In: *2008 9th International Conference on Signal Processing*. IEEE, 2008, pp. 1808–1810. ISBN: 9781424421787.

[13] Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno. *Cryptography Engineering: Design Principles and Practical Applications*. eng. Indianapolis, Indiana: Wiley Publishing, Inc. ISBN: 9780470474242.

[14] Cybersecurity and Infrastructure Security Agency (CISA). *Understanding Digital Signatures*. `https://www.us-cert.gov/ncas/tips/ST04-018`. [Online; accessed 8/6/2020].

[15] ARM. *Arm Processors for the Widest Range of Devices—from Sensors to Servers*. `https://www.arm.com/products/silicon-ip-cpu`. [Online; accessed 8/6/2020].

[16] Quarklab. *Introduction to Trusted Execution Environment: ARM’s TrustZone*. `https://blog.quarkslab.com/introduction-to-trusted-execution-environment-arms-trustzone.html`. [Online; accessed 8/6/2020].

[17] Sandro Pinto and Nuno Santos. “Demystifying Arm TrustZone: A Comprehensive Survey”. eng. In: *ACM Computing Surveys* 51.6 (2019). ISSN: 03600300. URL: `http://search.proquest.com/docview/2241331758/`.

[18] ARM. *SMC CALLING CONVENTION System Software on ARM® Platforms*. `http://infocenter.arm.com/help/topic/com.arm.doc.den0028b/ARM_DEN0028B_SMC_Calling_Convention.pdf`. [Online; accessed 8/6/2020].

[19] Steve Furber. *ARM system-on-chip architecture*. eng. 2. ed.. Harlow: Addison-Wesley, 2000. ISBN: 0-201-67519-6.

[20] Anxo Tato. “Software Defined Radio: A Brief Introduction”. eng. In: *Proceedings* 2.18 (2018), p. 1196. ISSN: Proceedings. URL: `https://doaj.org/article/2910cd9685e6442bada1c6bd7a5a1bf7`.

[21] Vishal Dey et al. “Security Vulnerabilities of Unmanned Aerial Vehicles and Countermeasures: An Experimental Study”. eng. In: *2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID)*. Vol. 2018-. IEEE, 2018, pp. 398–403. ISBN: 9781538636923.

[22] Andrew J Kerns et al. “Unmanned aircraft capture and control via GPS spoofing”. In: *Journal of Field Robotics* 31.4 (2014), pp. 617–636.

[23] David Atkinson. *DJI TRANSMISSION SYSTEMS – WI-FI, OCUSYNC LIGHTBRIDGE*. `https://www.heliguy.com/blog/2018/08/20/dji-transmission-systems-wi-fi-ocusync-lightbridge/`. [Online; accessed 8/6/2020]. 2018.

[24] FCC. *FCC ID SS3-L1Z1805*. `https://fccid.io/SS3-L1Z1805/Internal-Photos/Internal-Photos-3974376.pdf`. [Online; accessed 8/6/2020]. 2018.

[25] FCC. *FCC ID SS3-RC1A1805*. `https://fccid.io/SS3-RC1A1805/Internal-Photos/Internal-Photos-3974453.pdf`. [Online; accessed 8/6/2020]. 2018.

[26] Justin Case. *DUMLRacer*. `https://github.com/CunningLogic/DUMLRacer`. [Online; accessed 8/6/2020].

TRITA-EECS-EX-2020:353