Hacking and securing the AR.Drone 2.0 quadcopter - Investigations for improving the security of a toy

Conference Paper *in* Proceedings of SPIE - The International Society for Optical Engineering · January 2014

DOI: 10.1117/12.2044868

Johann Pleban
Brandenburg University of Applied Sciences
14 PUBLICATIONS **31** CITATIONS
SEE PROFILE

Reiner Creutzburg
Brandenburg University of Applied Sciences
460 PUBLICATIONS **331** CITATIONS
SEE PROFILE

Hacking and securing the AR.Drone 2.0 quadcopter - Investigations for improving the security of a toy

Johann-Sebastian Pleban, Ricardo Band, Reiner Creutzburg

Brandenburg University of Applied Sciences, Department of Informatics and Media
P.O.Box 2132, D-14737 Brandenburg, Germany

ABSTRACT

In this article we describe the security problems of the Parrot AR.Drone 2.0 quadcopter. Due to the fact that it is promoted as a toy with low acquisition costs, it may end up being used by many individuals which makes it a target for harmful attacks. In addition, the videostream of the drone could be of interest for a potential attacker due to its ability of revealing confidential information. Therefore, we will perform a security threat analysis on this particular drone. We will set the focus mainly on obvious security vulnerabilities like the unencrypted Wi-Fi connection or the user management of the GNU/Linux operating system which runs on the drone. We will show how the drone can be hacked in order to hijack the AR.Drone 2.0. Our aim is to sensitize the end-user of AR.Drones by describing the security vulnerabilities and to show how the AR.Drone 2.0 could be secured from unauthorized access. We will provide instructions to secure the drones Wi-Fi connection and its operation with the official smartphone app or third party PC software.

Keywords: drones, security vulnerabilities, quadcopter, AR.Drone 2.0, UAV

1. INTRODUCTION

The Parrot AR.Drone 2.0 is a remote controlled quadrocopter from the French company Parrot SA. The drone is controlled by the user via an app which is currently available for iOS and Android. Unofficial software is available for bada, Symbian and Windows Phone [1]. The second version of the AR.Drone was unveiled at CES Las Vegas in 2012. This drone offers an upgraded camera and more sensors. It was first presented at the International Consumer Electronics Show in Las Vegas in the year 2010 (winning the CES Innovations Award for Electronic Gaming Hardware). The drone consists of plastic and foam material and measures about 30 cm. The connection to the drone is realized using Wi-Fi and the drone is then controllable from smartphones and tablets using iOS or Android operating systems, which allows the transmission of the video stream from both cameras. The drone features include object tracking and the compatibility for AR-Game applications [2].

Quadrocopter and other remote controlled drones gain more and more in popularity and are getting cheaper in price, for this reason the AR.Drone 2.0 which is a popular device at low cost is used in this investigation and tested for its security vulnerabilities. In this paper the focus will be set on how to hijack the Parrot AR.Drones and how to secure them.

Even though the AR.Drone is not a professional drone, its features turn it into an interesting object for research and gaming purposes. It is easily controlled through the mobile application where the user gains access to the various sensors. The available API and documentation allows to create own applications and projects. Bonus features include the AR.Drone academy [3] for sharing flights, movies and pictures and augmented reality games (which are only supported under iOS). After powering up the drone, it will set up an open Wi-Fi Access-Point.

This paper is structured as follows: Section 2 introduces the Parrot AR.Drone 2.0 and its technical aspects. In Section 3 the security vulnerabilities of the AR.Drone 2.0 using different attacking scenarios are described. A way of securing the drone is explained in Section 4. Section 5 will introduce possible future work and finally, in Section 6 a summary is given.

Further author information:
Johann-Sebastian Pleban: pleban@fh-brandenburg.de
Ricardo Band: band@fh-brandenburg.de
Reiner Creutzburg: creutzburg@fh-brandenburg.de

2. TECHNICAL SPECIFICATIONS OF THE AR.DRONE 2.0

The AR.Drone 2.0 uses an OMAP 3630 CPU. This processor is based upon a 32 bit ARM Cortex A8 and runs with 1 GHz, it also uses a PowerVR SGX530 GPU with a frequency of 800 MHz on the System on a Chip (SoC) constructed by Texas Instruments. The drones memory include 1 GB DDR2-RAM, running with 200 MHz and 128 MB NAND-Flash. The drone runs the Kernel Linux uclibc 2.6.32.9-gbb4d210 using uclibc as c-library. This allows a lightweight and fast reacting system. The quadrocopter supports the WLAN standards b/g/n for Video and flight data. For navigation, an Inertial Navigation System (INS) is used. It contains a 3 axis accelerometer (with +-50 mg precision), a 3 axis gyroscope (2000°/second precision) and a 3 axis magnetometer (6°precision). For flight height, a pressure sensor (+- 10 Pa precision) and below 6 meters, ultrasound sensors for ground altitude measurement are used. The vertical QVGA (320x240) camera is used for ground speed measurement and videostream.
Expanded polypropylene (EPP) is used for the hull of the drone. The motors are mounted onto carbon fiber tubes, which leads to a total weight of 420 g (Indoor) or 380 g (Outdoor). For batteries, lithium polymer accumulators with 1000 mAh at 11.1 V are used in the standard configuration. 1500 mAh batteries are also available. The quadrocopter uses 4 brushless inrunner motors running 28.500 revolutions per minute (RPM) which are controlled by a 8 MIPS AVR CPU on each motor controller [4].
The front camera offers 720p at 30 fps with a wide angle lens (92° diagonal). The H.264 video stream can be stored onto a connected USB stick or the control device. Recorded videos and pictures can be shared to YouTube and Picasa or in the AR.Drone Academy, a social network from Parrot in the app.

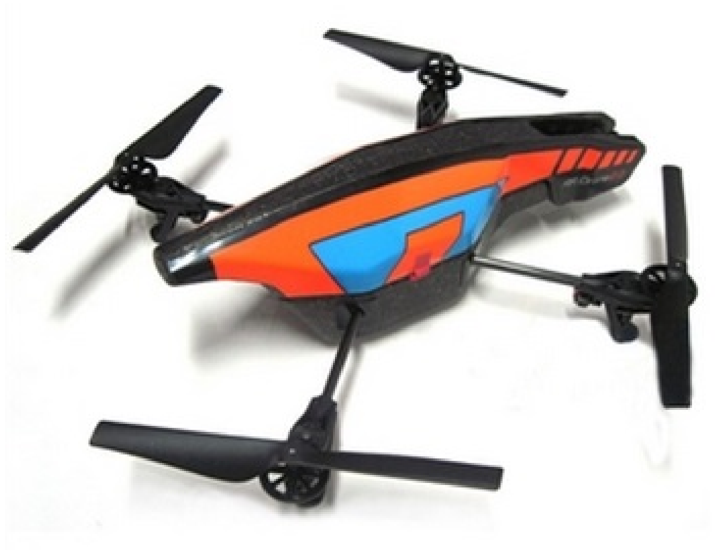


Figure 1. The Parrot AR.Drone 2.0

A flight recorder is to be released by Parrot, offering added GPS support, improved stability in flight and flights after set courses, a coming-home safe return function, review flights in 3D and sharing flights with in-flight data with other AR.Drone Academy users [5].

If one part of the drone gets broken, every single part of the drone can also be bought from the Parrot AR.Drone store and the user can then replace them individually.

The app offers an easy to use interface, which makes the control of the drone relatively easy, compared to more professional RC controlled drones. Most of the controlling aspects are handled automatically, like the start- and landing phases, calibration and position holding. That way, the user only needs to issue 'high level' commands, and can totally forget about the complex handling required by other drones (see Figures 2 and 3).

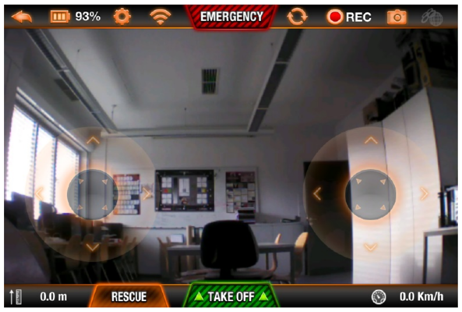


Figure 2. Parrot AR.FreeFlight control interface with two control buttons and a take off button for starting or landing the drone

Figure 3. Exemplary remote control for drones

2.1 Sensors and Inertial Navigation System (INS)

The AR.Drone 2.0 is equipped with multiple sensors, which are placed under the hull. This includes a miniaturized Inertial Navigation System (INS) which measures in six degrees of freedom. This allows the drone to perform yaw, pitch and roll movement. Other sensors perform flight stabilisation. The height control is realized through the usage of a pressure sensor and ultrasonic sensors.

2.2 Augmented reality games

Parrot offers two augmented reality games, AR.Race 2 and AR.Rescue 2. AR.Race 2 is a racing game allowing single and multiplayer games where the users race against each other to finish with the fastest time. AR.Rescue 2 is an augmented reality single player game, after placing the starting point (the AR.Drone target) the player must collect virtual scraps placed in the environment and fight against virtual aliens in 40 levels [6]. Additionally, Parrot released the ARdrone.org Open API to support studios and individuals in producing own software for the drone.

2.3 Expansion capabilities

The SDK [7] offered from Parrot allows developers to create apps that are even more feature-rich than the standard ones provided by Parrot. Multiple sensors on the drone and low acquisition cost make the drone an interesting object even for research [8] [9] [10] and education [11] [12] purposes. Already created apps include autonomous flight and to fulfill different tasks [13] [14] [15] [16]. Additionally, the drone can be connected to a third party receiver to increase the operation radius significantly (see next section). The internal USB connector allows to connect several hardware such as LTE modems.

2.4 Monitoring

Occupy-Wall-Street-protester Tim Pool configured one of the first generation AR.Drone, which he described as Occucopter, for the purpose of monitoring the police by the citizens. He tried to maintain a stable livefeed, allowing 50 persons to keep control over it. As soon as the police notice the controller of the drone, they might interrupt it but the control automatically switches to the next person [17] [18]. Using a 3rd or 4th generation mobile communication standard (like UMTS/LTE) it is also possible to control such drones from a much larger distance than just the normal Wi-Fi range. A proof of concept has already been shown by Alcatel-Lucent Bell Labs Acceleration Program and Parrot R&D [19], piloting the AR.Drone from a 1000 m (3280 ft) distance with a smartphone. This increase in control distance will increase the potential privacy threat.

3. FINDINGS - SECURITY VULNERABILITIES OF THE AR.DRONE 2.0

We investigated the security vulnerabilities of the AR.Drone 2.0 running in the standard out of the box configuration using firmware version 2.2.9. The following interesting details regarding the security have been found.

3.1 Port scan

After connecting the battery to the drone, it automatically boots up and is ready to use within seconds. It will check the motors and set up an unencrypted Wi-Fi hotspot named ardrone2_ followed by a random number with 6 digits. After connecting to this Wi-Fi hotspot, a port scan using the Nmap software on the drones IP has been performed. This scan already shows two interesting open well-known system ports, port 21 (FTP) and port 23 (Telnet). The other ports are related to the drone operation. Table 1 will show the usage of all ports.

Table 1. Open ports running on the AR.Drone 2.0.

Port	Explanation
21 (TCP)	FTP Server which serves video and image files recorded by the drone
23 (TCP)	Telnet Server offering a root shell
5551 (TCP)	FTP access to the update folder for the purpose of firmware updates
5553 (TCP)	VIDEO: The H264-720p frames of the camera are available here if the phone application is recording
5554 (UDP)	NAVDATA: Current telemetry data (status, speed, rotor speed) is sent to the client here (15 cmds/s demomode, 200 cmds/s full/debug mode).
5555 (TCP)	VIDEO: The video stream of the drone is available to clients here
5556 (UDP)	ATCMD: The drone is controlled in the form of AT commands. These control commands are sent periodically to the drone (30 cmds/s).
5559 (TCP)	CONTROL port: Some critical data, such as configurations are transferred here.

3.2 Explanation

3.2.1 Port 21 - FTP

The connection to the FTP service automatically running is not password protected, allowing anonymous access to the /data/video/ subdirectory of the drone. After connecting a USB thumb drive to the drone, it is automounted to /data/video/usb/. This allows direct access to the connected USB device and possibly gain access to confidential data or place malicious files on it.

3.2.2 Port 23 - Telnet

In the next step one can connect to the Telnet port which leads to a root shell. The root account is not password protected which gives an attacker free access to the entire drone operating system. This will allow an attacker to perform malicious actions like changing important config files or in the worst case scenario, wipe the filesystem to make the drone unusable. After investigating the filesystem, the following interesting shell scripts have been found:

Table 2. Interesting shellscripts and Linux files on the AR.Drone 2.0.

Filepath	Explanation
/bin/check_update.sh	Update script
/bin/init_gpios.sh	Initialisation of GPIO ports used for connecting the navigation board to the SoC
/bin/mount_usb.sh	Mounting of USB devices
/bin/pairing_setup.sh	Shell script for pairing using the Smartphone app
/bin/parallel-stream.sh	Camera streams
/bin/reset_config.sh	Reset config.ini while keeping total flighttime value
/bin/umount_usb.sh	Unmounting USB devices
/bin/Wifi_setup.sh	Start of Wi-Fi connection and other services
/sbin/udevd.sh	Start udevd with udev_init launcher
/lib/udev/rndis.sh	Hook script called by udhcpc on rndis interfaces-related events
/usr/sbin/loadAR6000.sh	Additional Wi-Fi settings
/etc/inetd.conf	Superserver for FTP (/update and /data/video)
/etc/udhcpd.conf	DHCP server configuration for Wi-Fi network
/data/config.ini	Main config file

3.2.3 Filesystem backdoor

Using these files, an attacker has even more options to cause harm. This could include automatic moving of malicious files to a connected USB device every time a device is connected, changing the configuration of the reset-script or prevent pairing with the drone. The shellscript /bin/reset_config.sh shows that, after using the reset button below the batteries, only the config file /data/config.ini will be reset. All other files remain untouched. This allows an attacker to always regain access to the drone, even after the user might use the reset button if he notice something is wrong with his drone.

Figure 4. Parrot AR Drone 2.0 in university lab

3.2.4 AT commands

The fact that the port 5556 (ATCMD) uses UDP and is therefore not a stable connection like TCP, a system with ascending sequence numbers has been selected for the commands. This prevents older commands with lower sequence numbers incoming later (due to transmission errors) from executing. The internal sequence counter can be set back to 1 in two different ways. One way this happens is if - within 2 seconds - no commands were sent to the ATCMD port. The other way is sending a new command with the sequence number 1. This offers another attack to take over the drone. This can be realized by either always sending the malicious command using the sequence number 1 or by a man-in-the-middle attack with a sequence number which is always higher than the one being sent from the legitimate user.

Table 3. AR.Drone 2.0 AT command style

```
AT*REF=<sequence>,<UI>
AT*PCMD=<sequence>,<enable>,<pitch>,<roll>,<gaz>,<yaw>
AT*CONFIG=<sequence>,"<name>","<value>"
```

An AT command begins with the fixed string "AT*", followed by either REF, PCMD or CONFIG. The equals sign is followed by the sequence number and the UI code separated by an comma. REF commands are single commands such as land or takeoff. PCMD commands consists of the sequence number, enable, pitch, roll, gaz and yaw seperated by comma. These commands are used for flight control. CONFIG commands are used for sending new configuration as key-value pairs, separated also by comma and quoted with double quotes.

Below is an excerpt of the simple text based protocol the AR.Drone uses.

Table 4. Excerpt of the ATCMD protocol of the AR.Drone 2.0

```
AT*REF=102,290717696
AT*PCMD=201,1,0,0,0,0
AT*PCMD=301,1,0,0,1036831949,0
AT*PCMD=302,1,0,0,-1110651699,0
AT*PCMD=303,1,1036831949,0,0,0
AT*PCMD=304,1,-1110651699,0,0,0
AT*PCMD=305,1,0,0,0,1036831949
AT*PCMD=306,1,0,0,0,-1110651699
AT*PCMD=307,1,0,1036831949,0,0
AT*PCMD=308,1,0,-1110651699,0,0
```

3.3 Possible attack scenarios

The following attack scenarios are possible:

3.3.1 Telnet

The attacker gains full access to the drone system. Here he can cause any damage. This includes the modification of critical system files and shell scripts of the Parrot software (see Table 2 of interesting shell scripts and other interesting files of the Linux operating system). In the simplest case, the attacker restarts the drone in mid air - resulting in immediately turning off the motors and thus falling onto the ground - just as soon as it is in his visible range. After catching it or picking the drone up from the ground, the drone is easily stolen. Since the drone runs on a Linux operating system, the same attacks which can be used to attack a 'normal' computer can be applied to the drone.

3.3.2 FTP

The FTP server can be used to inject files, or even a (custom) firmware update. There already have been cases where the firmware update did not work correctly and thus the drone was made "unusable". Fixing this requires technical steps the normal user would find difficult and time consuming or even pricey. After connecting a USB stick to the drone, the attacker has access to it and this gives him the possibility to read, modify and delete files on it.

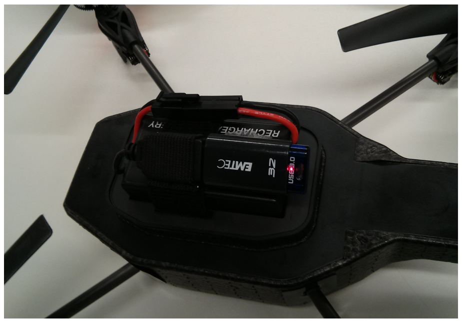


Figure 5. USB stick connected to the AR.Drone 2.0

3.3.3 Control ports

An attacker can interrupt the operation from the user that is using the smartphone app and he can eavesdrop the video stream (see [20] for more details on eavesdropping) or submit new configurations here.

3.3.4 Combination of attacks

A combination of attacks - like the virus-copter [21] or the SkyJack-Hack, can be used by a malicious attacker to takeover the drone entirely. This hack actively searches for hotspots generated by the Parrot AR.Drones. This is done by "seeking out any wireless connections from MAC addresses owned by the Parrot company" (refer to [22]). If the hack finds a drone nearby, it disconnects the real user by deauthenticating him, then initiates a new connection to it while the drone is waiting for the new connection from the real owner. Additional files will be transferred to the drone to control it from the attackers machine.

3.3.5 User accounts

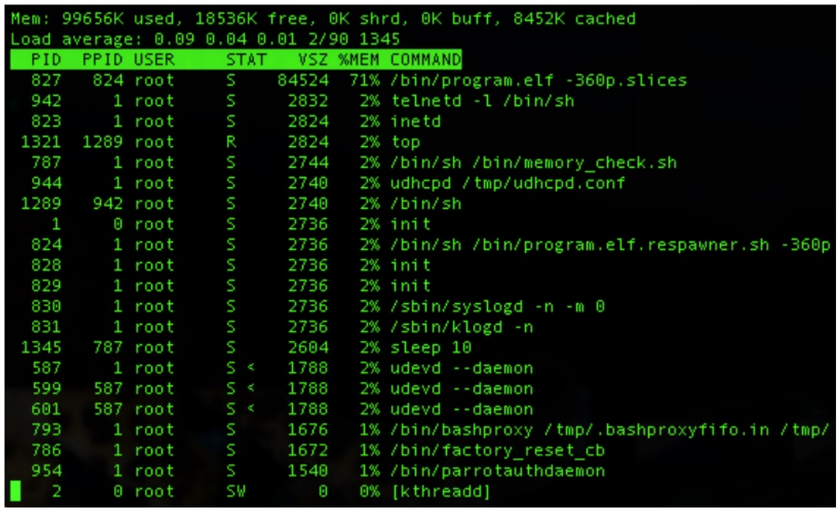
The linux system on the drone is already configured to have multiple users. But these users are not used. All programs run under the root account of the drone. An attacker could use these accounts to run his own code.

3.4 Security by Parrot

As a safety precaution against non-authorized connections to the drones access point, the smartphone app offers coupling with the drone. The coupling of the users MAC address is realized using iptables. The pairing provides the easiest, but also most effective protection against non-professional attackers. As a result, any non-authorized traffic is blocked. Even though the coupling using a MAC address can easily be spoofed, should an attack occur before pairing, this is the first point an attacker can take advantage of by editing the pairing script and adding his own MAC to lock out the legitimate user.

3.5 Summary

After investigating the Parrot AR.Drone 2.0 some remarkably attacking scenarios for a malicious attacker were found. Gaining full access to the drone using the root shell over the unencrypted Wi-Fi network is not hard to do. Here someone who wants to steal the drone might just place himself close to the drone, send the reboot command using the root shell and escapes with a new drone. Private data stored on a connected USB device is insecure while connected to the drone. Due to the automount, its data is visible for anyone connected to the FTP server.



```
Mem: 99656K used, 18536K free, 0K shrd, 0K buff, 8452K cached
Load average: 0.09 0.04 0.01 2/90 1345
  PID  PPID USER     STAT   VSZ %MEM COMMAND
  827   824 root     S    84524  71% /bin/program.elf -360p.slices
  942     1 root     S     2832   2% telnetd -l /bin/sh
  823     1 root     S     2824   2% inetd
 1321  1289 root     R     2824   2% top
  787     1 root     S     2744   2% /bin/sh /bin/memory_check.sh
  944     1 root     S     2740   2% udhcpd /tmp/udhcpd.conf
 1289   942 root     S     2740   2% /bin/sh
    1     0 root     S     2736   2% init
  824     1 root     S     2736   2% /bin/sh /bin/program.elf.respawner.sh -360p
  828     1 root     S     2736   2% init
  829     1 root     S     2736   2% init
  830     1 root     S     2736   2% /sbin/syslogd -n -m 0
  831     1 root     S     2736   2% /sbin/klogd -n
 1345   787 root     S     2604   2% sleep 10
  587     1 root     S <   1788   2% udevd --daemon
  599   587 root     S <   1788   2% udevd --daemon
  601   587 root     S <   1788   2% udevd --daemon
  793     1 root     S     1676   1% /bin/bashproxy /tmp/.bashproxyfifo.in /tmp/
  786     1 root     S     1672   1% /bin/factory_reset_cb
  954     1 root     S     1540   1% /bin/parrotauthdaemon
    2     0 root     SW       0   0% [kthreadd]
```

Figure 6. Running tasks on the AR.Drone 2.0

4. SECURING THE AR.DRONE

Secure operation of drones in general is an important factor [23] [24]. The only security feature currently active is the MAC address filter. This system can easily be tricked by spoofing the MAC address of the owners device when connecting to the drone. To get the needed address the attacker simply monitors and reads out the network traffic while the victim is operating the drone.

We will show how one can secure the AR.Drone in such a way that an attack is almost not possible. Our method follows a similar way as in [25] and [26]. This is realized by crosscompiling the needed programs.

4.1 Securing the Wi-Fi connection

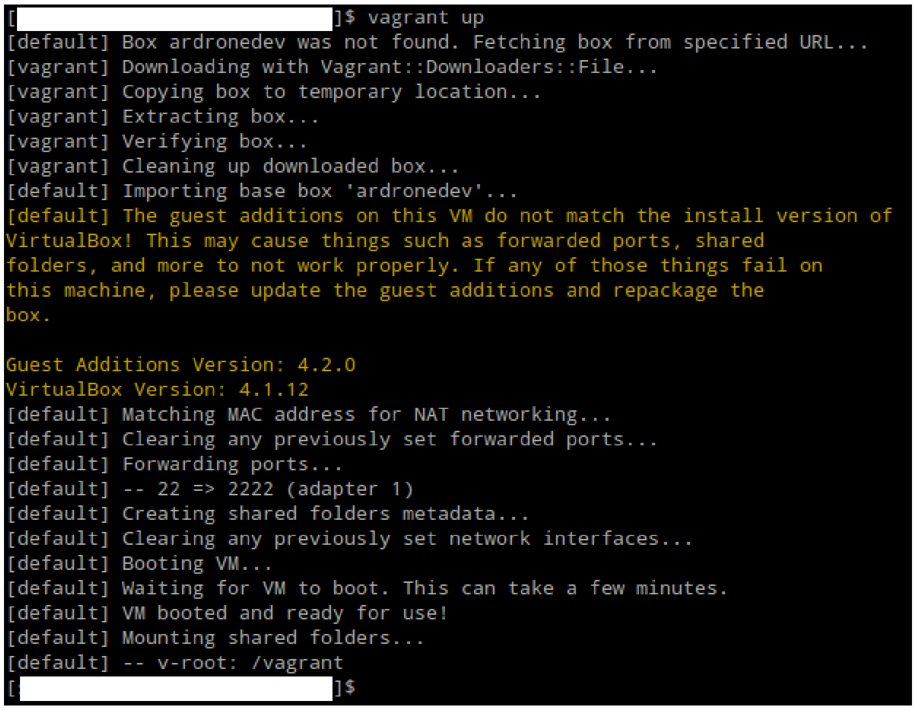
As previous investigation of the first version of the AR.Drone has shown, it is possible to eavesdrop the videosteam of the drone (refer to [20]). This is possible due to the unencrypted Wi-Fi network set up by the drone on start. In this chapter, a way of encrypting this network connection is shown. Our approach to a more secure concept that can be used to control the drone is that the drone is just a client and the control device acts as the access point. We will show how the wpa_supplicant can be used on the AR.Drone 2.0 to let it connect to an encrypted Wi-Fi network. This tool is a WPA Supplicant for Linux, Mac OS X, and Windows systems and supports WPA and WPA2 (IEEE 802.1X [27]). To use the wpa_supplicant on the drone, we first have to cross-compile it to run on the ARM architecture.

4.2 Preparing the virtual machine

The cross-compilation is made possible using vagrant as a virtual machine and compiling the source code of wpa_supplicant. We begin with installing virtualbox and vagrant on our development computer (this subsection is skipable, but this allows us to have a defined machine for every cross-compilation process).

The following steps are meant to setup, start and connect to the virtual machine that uses Ubuntu.

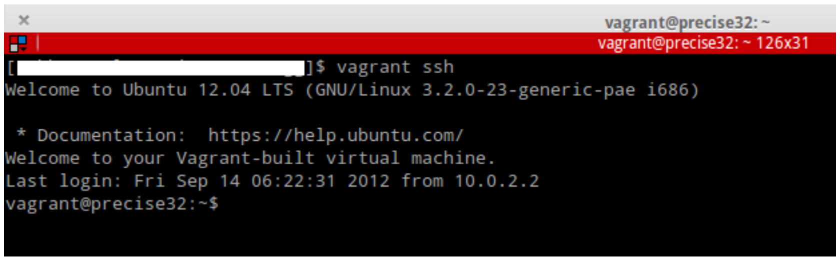
```
$ vagrant init ardronedev http://files.vagrantup.com/precise32.box
$ vagrant up
$ vagrant ssh
```



```
[[redacted]]$ vagrant up
[default] Box ardronedev was not found. Fetching box from specified URL...
[vagrant] Downloading with Vagrant::Downloaders::File...
[vagrant] Copying box to temporary location...
[vagrant] Extracting box...
[vagrant] Verifying box...
[vagrant] Cleaning up downloaded box...
[default] Importing base box 'ardronedev'...
[default] The guest additions on this VM do not match the install version of
VirtualBox! This may cause things such as forwarded ports, shared
folders, and more to not work properly. If any of those things fail on
this machine, please update the guest additions and repackage the
box.

Guest Additions Version: 4.2.0
VirtualBox Version: 4.1.12
[default] Matching MAC address for NAT networking...
[default] Clearing any previously set forwarded ports...
[default] Forwarding ports...
[default] -- 22 => 2222 (adapter 1)
[default] Creating shared folders metadata...
[default] Clearing any previously set network interfaces...
[default] Booting VM...
[default] Waiting for VM to boot. This can take a few minutes.
[default] VM booted and ready for use!
[default] Mounting shared folders...
[default] -- v-root: /vagrant
[[redacted]]$
```

Figure 7. Booting the Vagrant virtual machine



```
[[redacted]]$ vagrant ssh
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.2.0-23-generic-pae i686)

 * Documentation:  https://help.ubuntu.com/
Welcome to your Vagrant-built virtual machine.
Last login: Fri Sep 14 06:22:31 2012 from 10.0.2.2
vagrant@precise32:~$
```

Figure 8. SSH into the virtual machine

4.3 Preparation and crosscompilation of wpa_supplicant using a virtual machine

Now that we are connected to the virtual machine, we can continue to install our ARM toolchain and other needed packages:

```
$ apt-get install build-essential curl
$ curl -OL http://www.codesourcery.com/public/gnu_toolchain/arm-none-linux
-gnueabi/arm-2013.05-24-arm-none-linux-gnueabi-i686-pc-linux-gnu.tar.bz2
```

Extract the ARM toolchain package, change owner to the (vagrant) user, and remove the downloaded tar to save disk space.

```
$ tar -xf arm-2013.05-24-arm-none-linux-gnueabi-i686-pc-linux-gnu.tar.bz2
$ chown -R vagrant:vagrant arm-2013.05
$ rm -rf arm-2013.05-24-arm-none-linux-gnueabi-i686-pc-linux-gnu.tar.bz2
```

Now the environment is set to compile our needed tools. We begin with cross-compiling wpa_supplicant. To do so, we first download the source code of wpa_supplicant which can be obtained from the official page and then extract it to the local home folder:

```
$ curl -OL http://hostap.epitest.fi/releases/wpa_supplicant-2.0.tar.gz
$ tar -zxf wpa\_supplicant-2.0.tar.gz
```

Copy the default .config file and edit the new .config file

```
$ cp wpa\_supplicant-2.0/wpa\_supplicant/defconfig wpa\_
supplicant-2.0/wpa\_supplicant/.config
$ cd wpa\_supplicant-2.0/wpa\_supplicant
$ nano .config
```

Comment out the line CONFIG_DRIVER_NL80211=y so the corresponding lines look like

```
# Driver interface for Linux drivers using the nl80211 kernel interface
#CONFIG\_DRIVER\_NL80211=y
```

Also, add the following to the bottom of your .config file

```
export SOURCERY=/home/vagrant/arm-2013.05
export TOOL_PREFIX="${SOURCERY}/bin/arm-none-linux-gnueabi"
export CXX="${TOOL_PREFIX}-g++"
export AR="${TOOL_PREFIX}-ar"
export RANLIB="${TOOL_PREFIX}-ranlib"
export CC="${TOOL_PREFIX}-gcc"
export LINK="${CXX}"

export CCFLAGS="-march=armv7-a -mtune=cortex-a8 -mfpu=vfp"
export ARM_TARGET_LIB="${SOURCERY}/arm-none-linux-gnueabi/libc"

CONFIG_TLS=internal
CONFIG_INTERNAL_LIBTOMMATH=y
CONFIG_INTERNAL_LIBTOMMATH_FAST=y
```

Close and save the file, and finally run:

```
$ make
```

Resulting in the files wpa_supplicant, wpa_cli and wpa_passphrase being generated in the current folder.

4.4 Installation of wpa_supplicant, wpa_cli and wpa_passphrase on the drone

To move the compiled files onto the drone and use the wpa_supplicant for connecting to our secure network, these files can be copied onto an USB stick and after connecting it, simply moving them to the /bin/ subdirectory using the rootshell from Telnet is enough to install them on the drone. Or simply uploading them using the FTP sever and moving them afterwards, which is described here:

```
$ curl -T bin/wpa_cli "ftp://192.168.1.1"
$ curl -T bin/wpa_passphrase "ftp://192.168.1.1"
$ curl -T bin/wpa_supplicant "ftp://192.168.1.1"

$ telnet 192.168.1.1
$ mv /data/video/wpa_* /bin
$ chmod +x /bin/wpa_*
$ exit
```

4.5 Using the secure network

As stated before, we are going to use an external (secured) access point in order to use encrypted Wi-Fi. This hotspot can be generated by the control device as current mobile phones and tablets offer the posibility to create such hotspots. To keep the app running, the drone must maintain its original IP address (192.168.1.1). This is the only IP address that must remain free if the user does not want to use a third party app to control the drone. The following method of starting the secure connection requires a third device to initiate the connection from the drone to the secured network. First, the drone should start up normally after connecting the battery. Now the secure hotspot can be started on the control device that will be later used to control the drone. The third device must now connect to the unsecure drone network which is called ardrone2_ followed by a random number. Still on the third device a telnet connection must be initiated to the drone. After issuing the commands:

```
$ wpa_passphrase $ESSID $PASSWORD > /etc/wpa_supplicant.conf
$ ifconfig ath0 $ADDRESS
$ iwconfig ath0 essid $ESSID
$ wpa_supplicant -B -Dwext -iath0 -c/etc/wpa_supplicant.conf
```

via telnet. $ESSID must be replaced with the name of the secure network and the password must be set. $ADDRESS will be the new address the drone uses in the new network. To keep the original app running, the user should set this to 192.168.1.1. The drone will close the hotspot and connect to the control device on which the secured hotspot is running. By keeping the initiation of the connection to the secure network manual, we ensure the normal operation is still possible after reconnecting the battery, so even in case of a missconfiguration no special operations are required.

4.5.1 Benefits of using an external network

Since the new (secured) network can be connected to the internet, different new posibilities are possible. This includes the operation of the drone over the internet. The videostream could also be transmitted over the internet like a flying webcam which can be controlled from far away. The only problem is the reduced operating time of the drone due to the low battery time.

5. FUTURE WORK

Future work will include adding scripts to the drone or second device to remove the need for a third device to initiate the connection to the network. The security could be improved even more with a randomly generated passphrase which is given to the drone operator by touching a NFC device on the drone with the smartphone app. An app for tablets and smartphones that is allowing the drone to use a custom IP to operate would be of interest too. If this custom app is given, a way of automatically securing the drone by applying our method to it would be the next thing to develop.

6. SUMMARY

In this paper we have shown that the Parrot AR.Drone 2.0 can be secured in the way that it uses encryption instead of an unencrypted Wi-Fi to operate. Several attacking scenarios that a malicious attacker could use to disrupt normal operation have been shown. Using the option to use an external network connection secures the drone from such attacks. Additionally, the risk of showing the contents of a connected USB device is not given anymore.

REFERENCES

[1] Weiss, F., "Quadrocopter Parrot AR.Drone fliegt auch mit Windows Phone (Video)." `http://de.engadget.com/2012/03/04/quadrocopter-parrot-ar-drone-fliegt-auch-mit-windows-phone-vide/` (March 2012).

[2] Parrot, "Discover the NEW AR.Drone 2.0. Fly, Record & Share in High Definition!." `http://blog.parrot.com/2012/01/09/ardrone2_ces2012/` (January 2012).

[3] Demgen, A., “AR.Drone-Academy: Soziales Netzwerk für Drohnen-Flieger verfügbar.” http://www.netzwelt.de/news/93209-ar-drone-academy-soziales-netzwerk-drohnen-flieger-verfuegbar.html (August 2012).

[4] Parrot, “AR.Drone 2.0 specifications.” http://ardrone2.parrot.com/ardrone-2/specifications/.

[5] “AR.Drone 2.0 flight-recorder.” http://ardrone2.parrot.com/apps/flight-recorder/.

[6] Webster, A., “AR.Drone coming to Android, gets new multiplayer games.” http://arstechnica.com/gaming/2011/06/ardrone-coming-to-android-gets-new-multiplayer-games/ (June 2011).

[7] Piskorski, S., Brulez, N., Eline, P., and D’Haeyer, F., “AR.Drone Developer Guide,” (May 2012).

[8] Krajník, T., Vonásek, V., Fier, D., and Faigl, J., “AR-Drone as a Platform for Robotic Research and Education,” in [*Research and Education in Robotics: EUROBOT 2011*], Springer, Heidelberg (2011).

[9] Schön, S., Band, R., Pleban, J., Creutzburg, R., and Fischer, A., “Identifikation von Anwendungsfeldern der Technologie bei autonomen fliegenden Robotern in Projekten zwischen Hochschule und Wirtschaft,” (2013).

[10] Schön, S., Band, R., Pleban, J.-S., Creutzburg, R., and Fischer, A., “Applications of multimedia technology on autonomous flying robots for university technology transfer projects,” in [*SPIE Proceedings Vol. 8667, Multimedia Content and Mobile Devices, 86670Q*], (2013).

[11] Schön, S., Band, R., Pleban, J., Creutzburg, R., and Fischer, A., “Konzeption von praktischen Übungen mit autonom fliegenden Robotern in der Ausbildung von Informatik-Studenten,” (2013).

[12] Band, R., Pleban, J., Schön, S., Creutzburg, R., and Fischer, A., “Concept for practical exercises for studying autonomous flying robots in a university environment: Part i,” in [*SPIE Proceedings Vol. 8667, Multimedia Content and Mobile Devices, 86670P*], (2013).

[13] Bills, C., Chen, J., and Saxena, A., “Autonomous MAV Flight in Indoor Environments using Single Image Perspective Cues,”

[14] Bills, C. and Yosinski, J., “MAV Stabilization using Machine Learning and Onboard Sensors,” (2010).

[15] Faigl, J., Krajník, T., Vonásek, V., and Přeučil, L., “Surveillance Planning with Localization Uncertainty for UAVs,” in [*3rd Israeli Conference on Robotics,*], –, Ariel University Center, Ariel (2010).

[16] Higuchi, K., Shimada, T., and Rekimoto, J., “Flying sports assistant: external visual imagery representation for sports training,” *AH ’11 Proceedings of the 2nd Augmented Human International Conference Article No. 7* (2011).

[17] Sharkey, N. and Knuckey, S., “OWS Fights Back Against Police Surveillance by Launching "Occucopter" Citizen Drone.” http://www.alternet.org/story/153542/ows_fights_back_against_police_surveillance_by_launching_%22occucopter%22_citizen_drone (December 2011).

[18] Wagstaff, K., “Occupy Wall Street’s New Drone: ’The Occucopter’.” http://techland.time.com/2011/12/21/occupy-wall-streets-new-drone-the-occucopter/ (December 2011).

[19] Méchaly, A., “One flew over the cornfield.” http://www2.alcatel-lucent.com/blogs/corporate/2012/10/one-flew-over-the-cornfield/ (October 2012).

[20] Samland, F., Fruth, J., Hildebrandt, M., Hoppe, T., and Dittmann, J., “AR.Drone: security threat analysis and exemplary attack to track persons,” in [*SPIE Proceedings Vol. 8301 Intelligent Robots and Computer Vision XXIX: Algorithms and Techniques, Juha Röning; David P. Casasent (Eds.), 83010G*], (2012).

[21] Ackerman, E., “AR Drone that infects other drones with virus wins DroneGames.” http://spectrum.ieee.org/automaton/robotics/diy/ar-drone-that-infects-other-drones-with-virus-wins-dronegames (December 2012).

[22] Kamkar, S., “SkyJack.” https://github.com/samyk/skyjack (December 2013).

[23] “Hacking Drones And The Dangers It Presents.” http://www.npr.org/2012/07/08/156459939/hacking-drones-and-the-dangers-it-presents (July 2012).

[24] “Drone hack explained: Professor details UAV hijacking.” http://rt.com/usa/news/texas-professor-drone-hacking-249/ (July 2012).

[25] “node-cross-compiler.” https://github.com/felixge/node-cross-compiler/.

[26] “ardrone-wpa2.” https://github.com/daraosn/ardrone-wpa2.

[27] “IEEE_802.11.” http://en.wikipedia.org/wiki/IEEE_802.11.