# USB: Camera to Driver

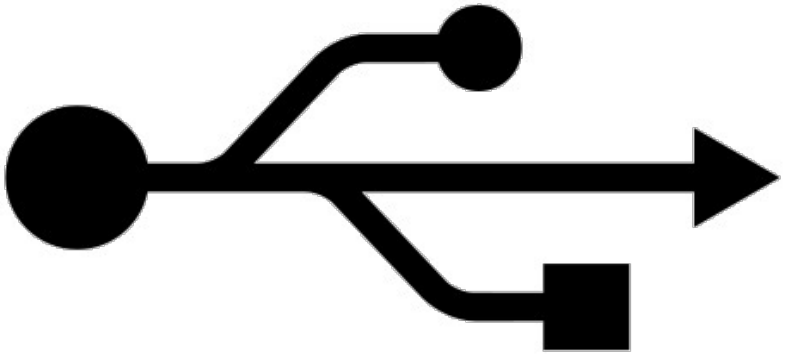

John McMaster
JohnDMcMaster@gmail.com

# What

- Goal: Camera + Windowz app => Linux app
- Why: PoC to create Linux drivers
- Small introduction to USB
- Capturing packets with Wireshark
- Replaying packets with uvusbrply
- Making a simple app

# USB101

- Low to high speed packetized communications
- Many uses: hard drive, rocket launcher, etc
- Standard driver interfaces but not always used
  - Ex: webcam standard is rarely used



# USB 101: endpoints

- Communicate with “endpoints”
- Two endpoint types of interest:
  - Control: small data transfer
  - Bulk: large data transfer
- Some initialization stuff we can glaze over

# USB 101: misc

- URB: Linux packet identifier
- USB version (ie 2 vs 3) only affects PHY

Image source: http://www.hackolog.com/content-img-01/wiimote-used-for-usb-missile-launcher-auto-aiming-feature-1.jpg

# USB 101: sample control packet

- Transfer type: URB_CONTROL (0x02)
- Endpoint: 0x40, Direction: OUT
- URB setup
  - bmRequestType: 0x40
  - brequest: 1
  - wValue: 0x0001
  - wIndex: 15
  - wLength: 0

# USB 101: sample bulk packet

- Transfer type: URB_BULK (0x03)
- Endpoint: 0x82, Direction: IN
- URB length: 16384
- Device responds with 16384 or less bytes

# Wireshark: capturing USB

- Prereqs:
  - Wireshark installed.  I'm using 1.6.7
  - Linux USB monitoring enabled (sudo modprobe usbmon)
  - A way to run the app (ie VM)
  - USB port w/ device attached to it
- Demo Windows app + Wireshark

# Introduction to pylibusb

- Python bindings for libusb
- Demo: small pylibusb program


Image source: http://en.wikipedia.org/wiki/Python_%28programming_language%29#mediaviewer/File:Python_logo_and_wordmark.svg

# usbrply

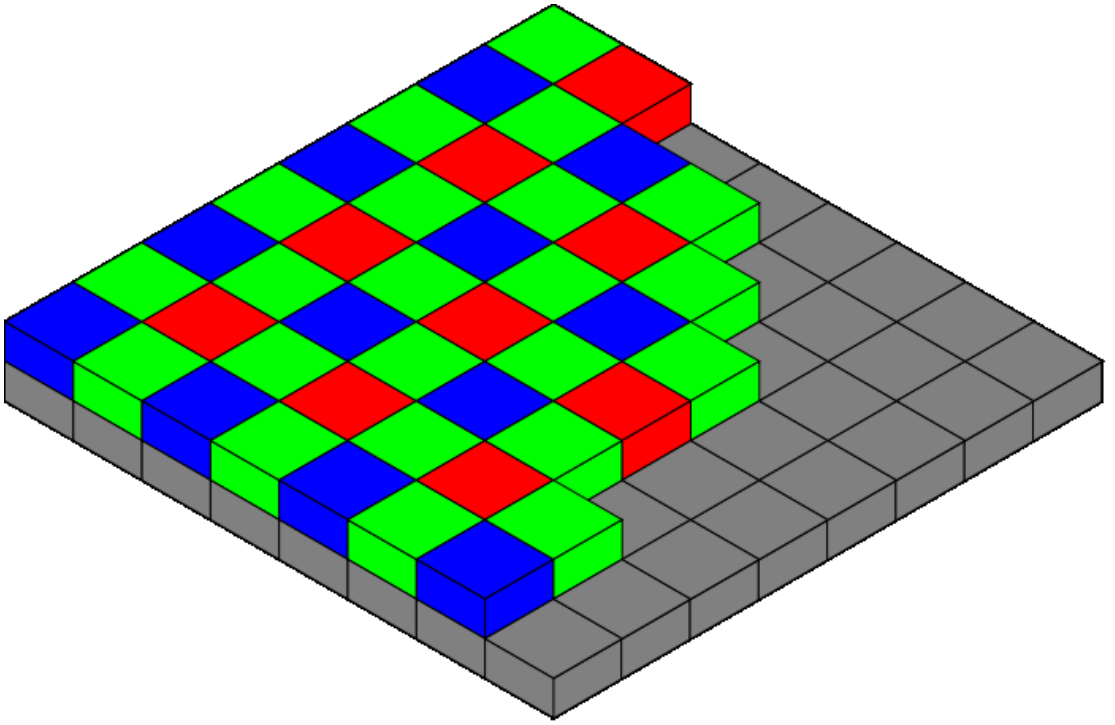
- Converts wireshark packet capture to code
  - Python: we'll focus on this
  - C (libusb)
  - C (Linux kernel)
- Written in Python
- Demo: replaying our capture

# Making a simple capture program

- Use pylibusb to take uvusbrply output and capture camera data
- We must setup bulk capture ourself
- Writes some images to a file

# Decoding the image

- How do we know that the image we captured really works?
- Bayer pattern safe guess
- Demo!



# Extras: understanding the output

- Why: learn to tweak settings
- Capture with setting at one value, then another
- Diff packet captures
- Camera may conform to SMIA
- Protocol may be obfuscated (ex: MU800)

# Extras: frame sync

- Look at Linux drivers for clues on how others do it
- Common sync algorithms:
  - Fixed/magic pattern in frame
  - Checksum in frame
- MU800: short bulk read

# Extras: making a Linux module

- Minor function differences over libusb
- Copy another driver as a template
- Demo: MU800 driver overview
  - I have an MD1800 driver but its not very polished

Image source: http://en.wikipedia.org/wiki/Linux#mediaviewer/File:Tux.svg

# Thanks for listening!

- Questions?  Interested?
  - JohnDMcMaster@gmail.com
- All content CC BY unless otherwise noted
- usbrply: https://github.com/JohnDMcMaster/usbrply/blob/master/main.py