Cheating in eSports

How to cheat at virtual cycling using USB hacks

Brad Dixon, Carve Systems



Photo by Markus Spiske on Unsplash

Sweaty eSports



EGVsOG



CVRWorldCup

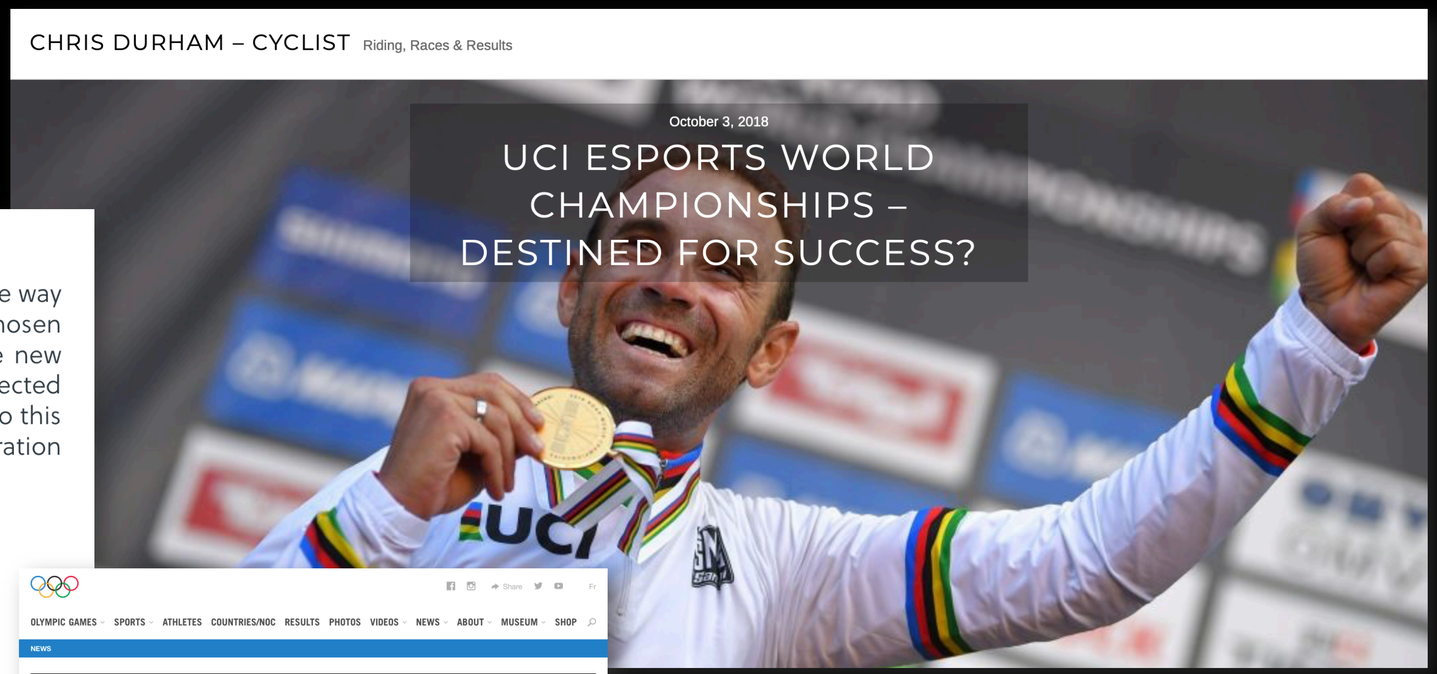
Virtual Cycling is Part of Cycling

UNION CYCLISTE INTERNATIONALE

▶ **Integrate E-Bike as a new discipline**

The ongoing digital revolution has totally transformed our lifestyles, including the way in which we play sport. eSports have made their appearance, and the IOC has chosen to take an interest in this new form of competition, which represents a whole new concept in sport. Cycling has a significant advantage over other sports in that connected cyclists can engage in both real and virtual sport at the same time. In response to this social phenomenon, the UCI must be at the forefront of the International Federation movement and create a structure for E-Bike.

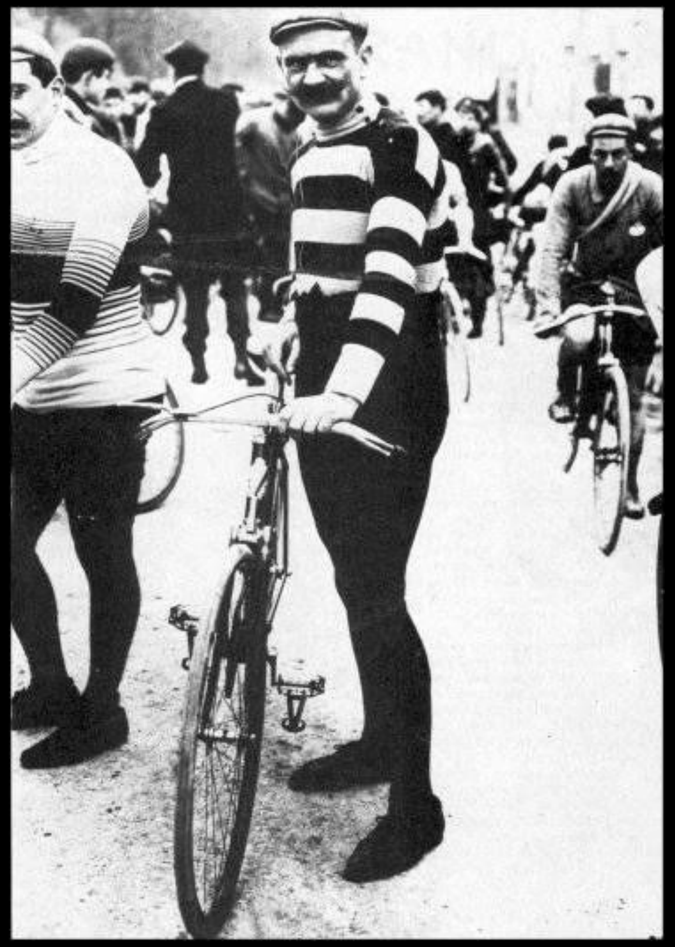
- Draw conclusions from the process of reflection engaged in by the IOC on the subject of eSports (2019-2022).
- Bring together, structure and organise E-Bike activities (2018-2022).
- Organise the UCI E-Bike World Championships and national and continental championships (2019).
- Promote the inclusion of E-Bike on the Olympic programme (2024).



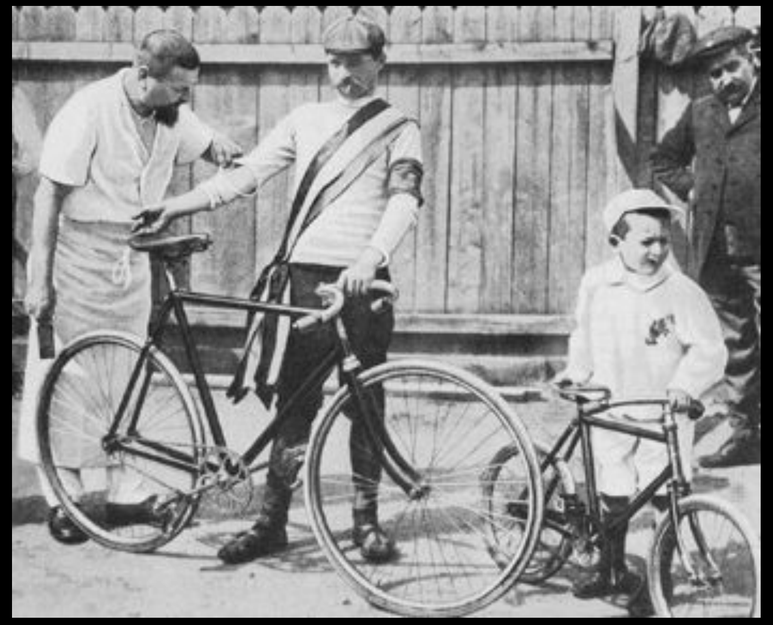


CYCLING: THE **BEST** CHEATERS

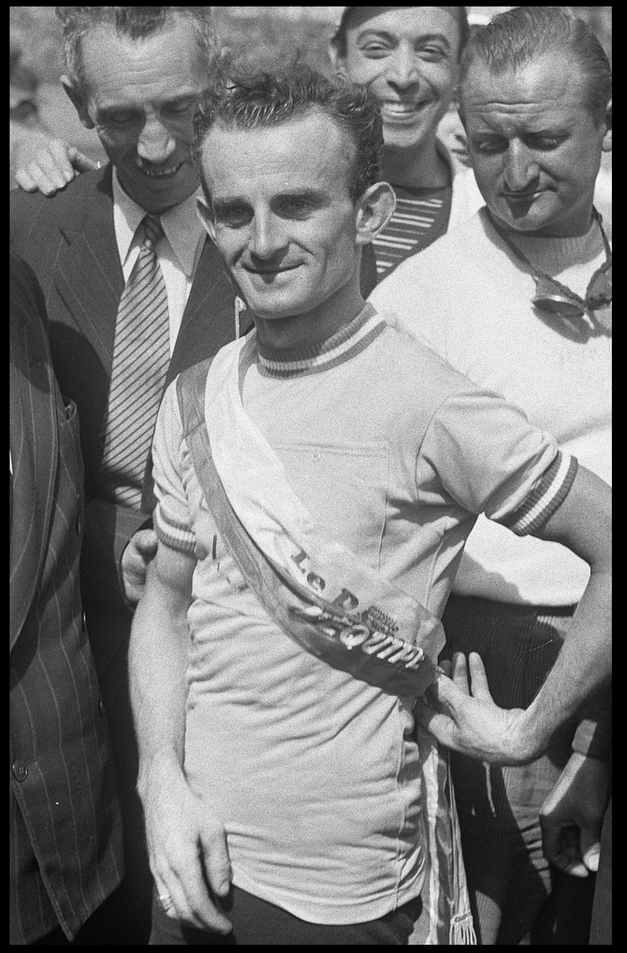
Cycling: Over 100 Years of Cheating Innovation



1903, 1904: Hippolyte Aucouturier



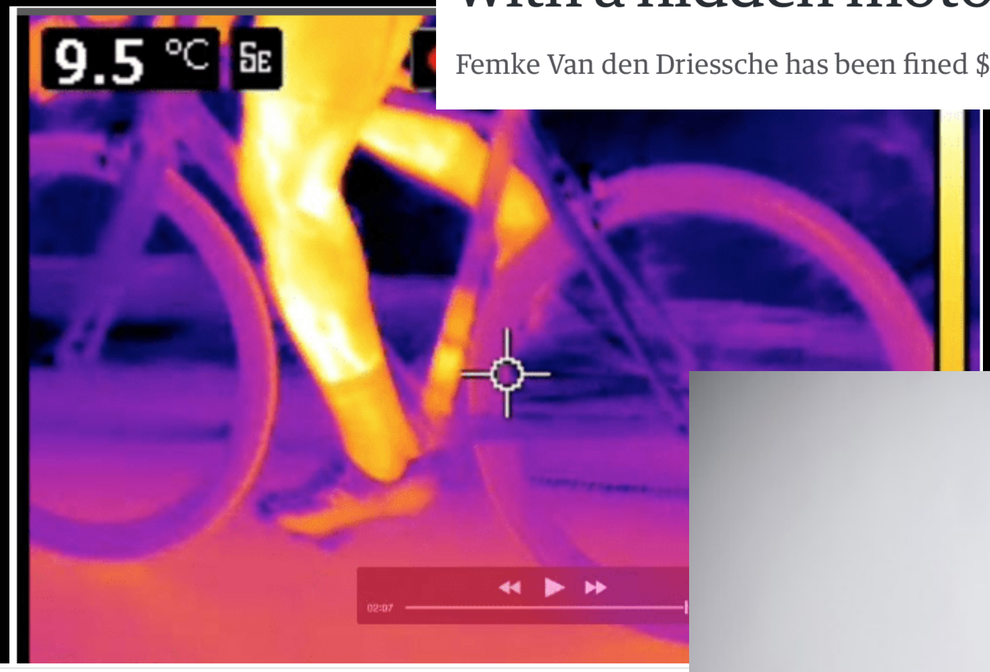
1904: Maurice Garin

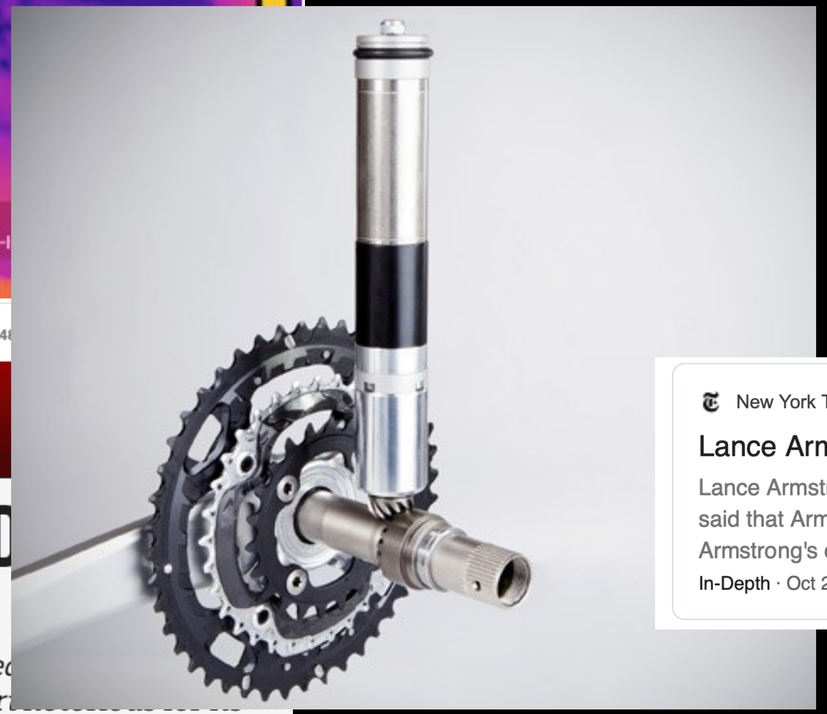


1947: Jean Robic

Cyclist banned for six years after racing with a hidden motor

Femke Van den Driessche has been fined $20,569 too.







CBS News / CBS Evening News / CBS This Morning /

60 MINUTES EPISODES

60 MINUTES INVESTIGATES HIDD MOTORS AND PRO CYCLING

Bill Whitaker investigates whether pro cyclists have used motors to win races -- like the Tour de France -- in a sport culture of cheating

2017 JAN 29 | CORRESPONDENT BILL WHITAKER | FACEBOOK | TWITTER | REDDIT | FLIPBOARD

New York Times

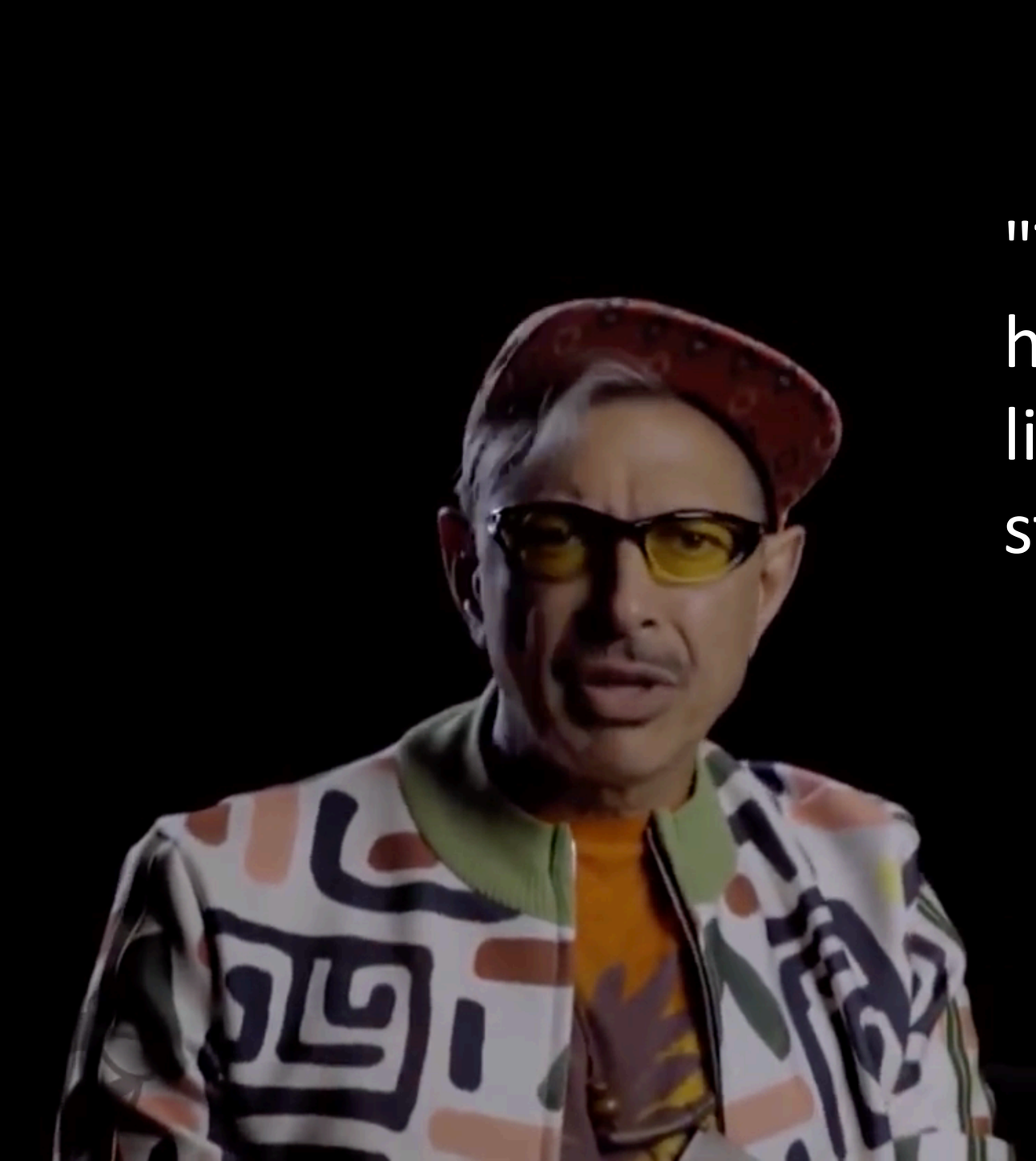
Lance Armstrong Stripped of 7 Tour de France Titles

Lance Armstrong Is Stripped of His 7 Tour de France Titles ... McQuaid said that Armstrong's teams had a “win at all costs” attitude fueled by ... Armstrong's case and will probably strip him of the bronze medal he won

In-Depth · Oct 22, 2012



Will people cheat at virtual cycling, too?



"This is a sport with literally hundreds of dollars on the line, and dozens of fans...the stakes are medium!"

Marty Hass -- Tour de Pharmacy, 2017, HBO

No… Marty Hass is not a real person. Don’t you recognize Jeff Goldblum? It is a silly mockumentary. Have a laugh.

Virtual Cycling: How does this work?

- Just like any MMPOG plus
- Sensors to measure real world performance
- App-controlled resistance

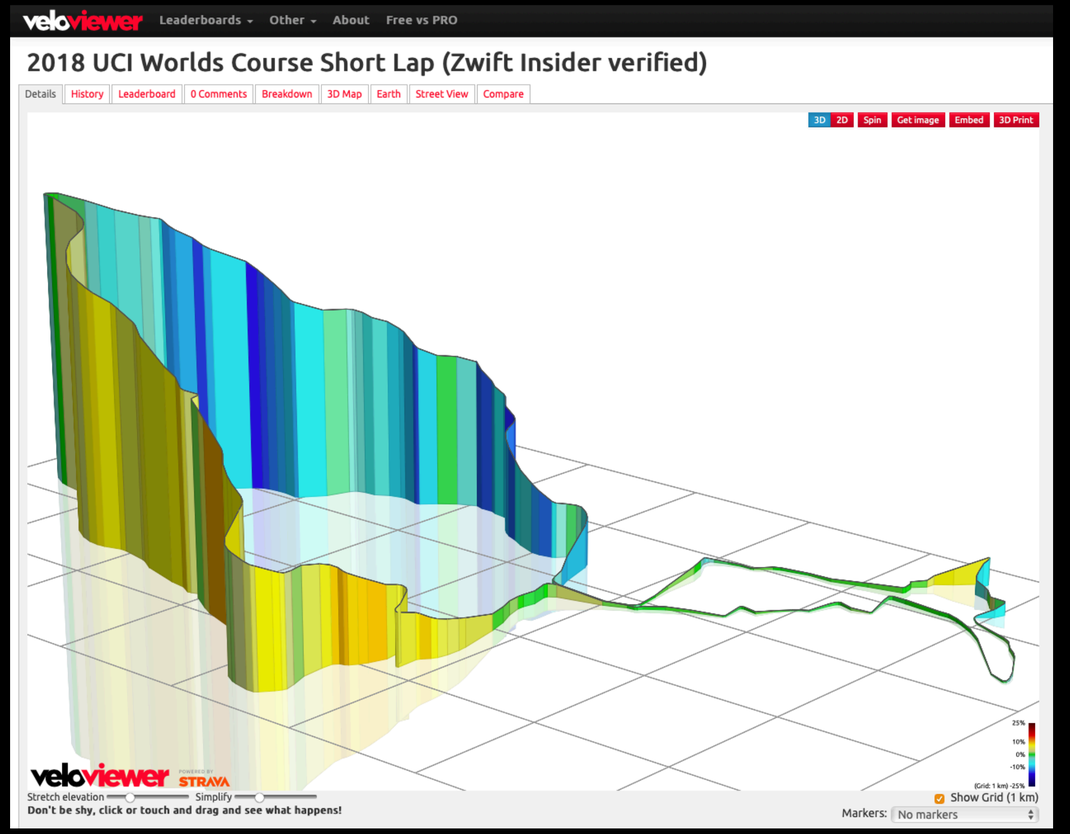




Bike Radar: Best Cycling Smart Trainers - 10-Way Mega-Test

Speed Estimation

- Course terrain model
- Power
- Rider mass
- Drafting model?

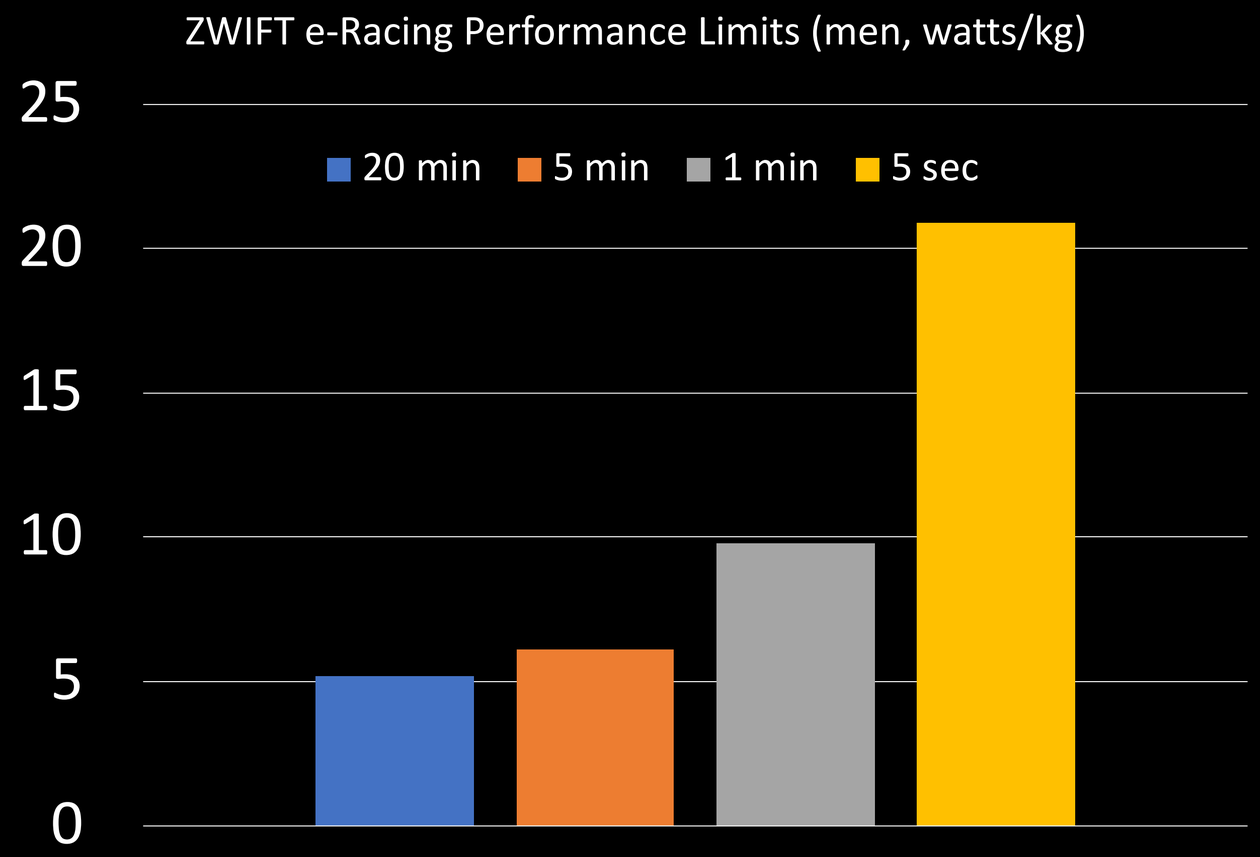


The Easy Way to Cheat at Virtual Cycling

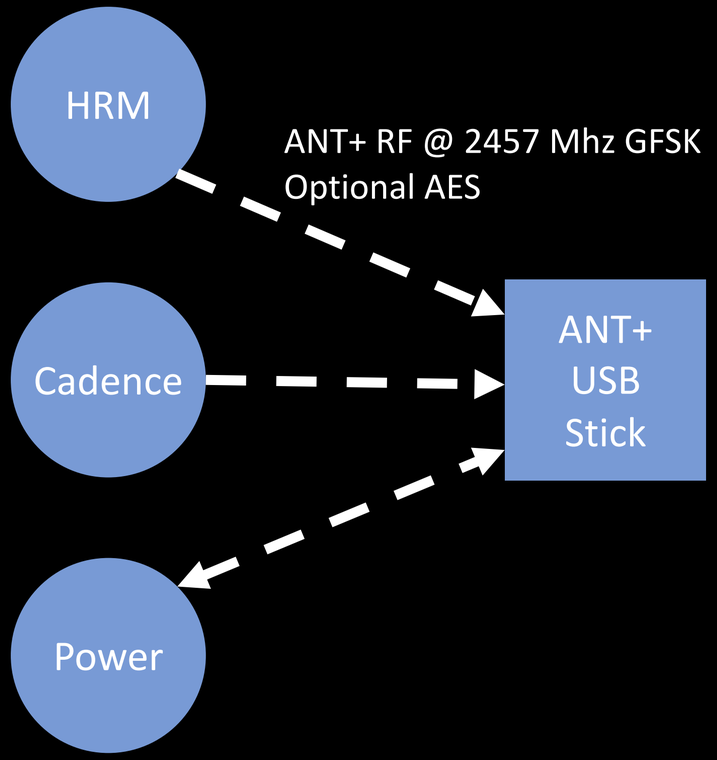
given the same power…

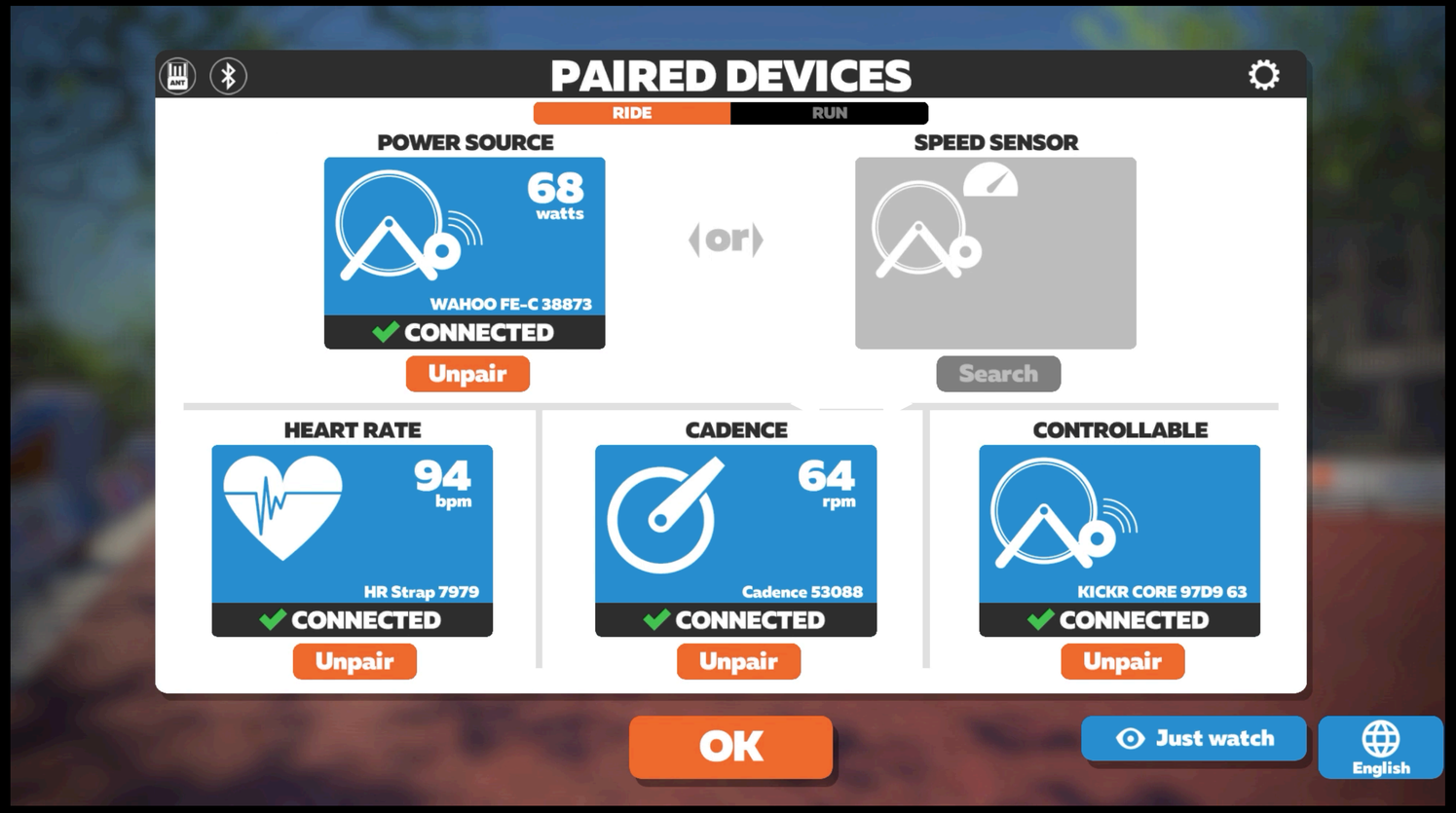
- Lighter riders *go faster*
- Shorter riders *draft better*

…there are limits!



Vulnerable Sensor Network





Cheat the Hard Way with USBQ



Hack’in USB ain’t new

- Facedancer: *excellent!*
  - Travis Goodspeed (@travisgoodspeed)
  - Sergey Bratus (@sergeybratus)
  - Kate Temkin (@ktemkin)
  - Dominic Spill (@dominicgs)
  - Michael Ossmann (@michaelossmann)
- Hardware Village USB Links: Andrey Konovalov
- USB Reverse Engineering: Down the Rabbit Hole: Grant “devalias” Glenn

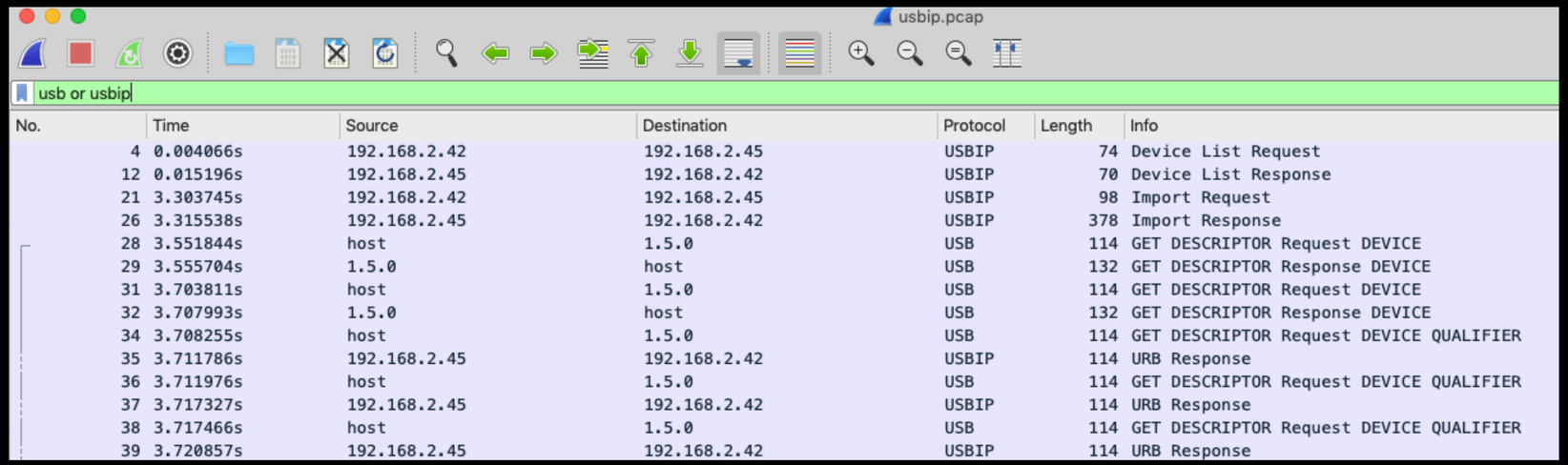
Just want to observe USB?

tcpdump + Wireshark

- Requires Linux and the `usbmon` module.
- Capture with `tcpdump -i usbmon0 …`
- Wireshark is great!

usbip + Wireshark

- Linux `usbip` module can export USB devices over TCP.
- Capture TCP, observe in Wireshark.



No.	Time	Source	Destination	Protocol	Length	Info
4	0.004066s	192.168.2.42	192.168.2.45	USBIP	74	Device List Request
12	0.015196s	192.168.2.45	192.168.2.42	USBIP	70	Device List Response
21	3.303745s	192.168.2.42	192.168.2.45	USBIP	98	Import Request
26	3.315538s	192.168.2.45	192.168.2.42	USBIP	378	Import Response
28	3.551844s	host	1.5.0	USB	114	GET DESCRIPTOR Request DEVICE
29	3.555704s	1.5.0	host	USB	132	GET DESCRIPTOR Response DEVICE
31	3.703811s	host	1.5.0	USB	114	GET DESCRIPTOR Request DEVICE
32	3.707993s	1.5.0	host	USB	132	GET DESCRIPTOR Response DEVICE
34	3.708255s	host	1.5.0	USB	114	GET DESCRIPTOR Request DEVICE QUALIFIER
35	3.711786s	192.168.2.45	192.168.2.42	USBIP	114	URB Response
36	3.711976s	host	1.5.0	USB	114	GET DESCRIPTOR Request DEVICE QUALIFIER
37	3.717327s	192.168.2.45	192.168.2.42	USBIP	114	URB Response
38	3.717466s	host	1.5.0	USB	114	GET DESCRIPTOR Request DEVICE QUALIFIER
39	3.720857s	192.168.2.45	192.168.2.42	USBIP	114	URB Response

Stuff Brad Knows

JAN 2019 EDITION

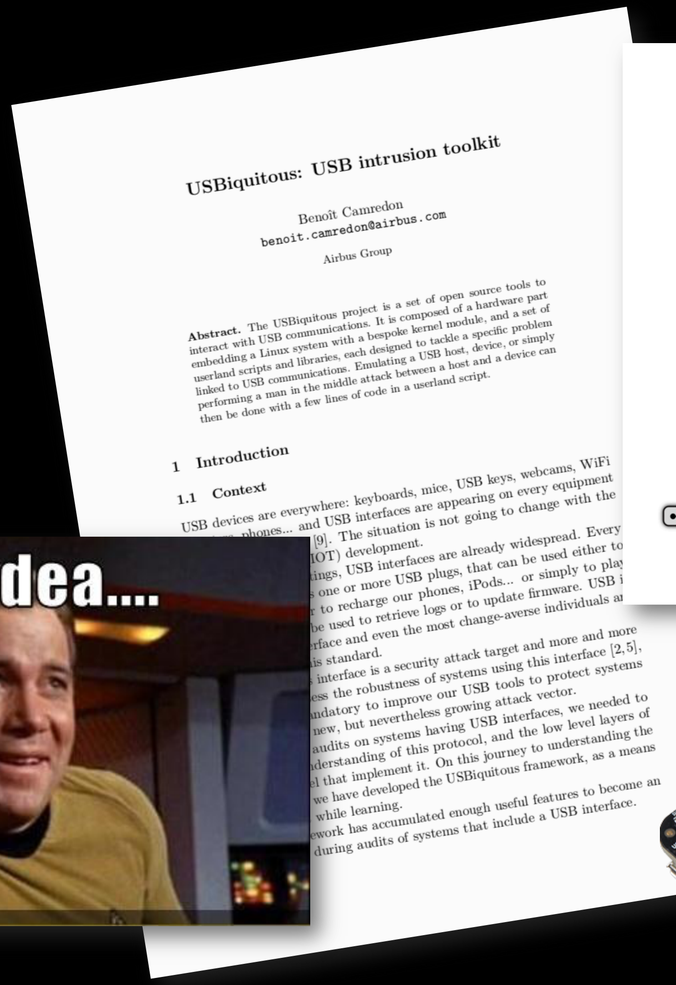
USB Device Drivers and Kernel Code

- Emulate USB host or device functions at the lowest level.
- Behave badly and deviate from the expectations of USB drivers.
- Use GoodFET-based board and *Facedancer!*



USBiquitous by Benoît Camredon

- USB 2.0 MITM using loadable kernel module
- Beaglebone Black
- Python 2 userspace
- usbq_core
- usbq_userland



USBiquitous: USB intrusion toolkit

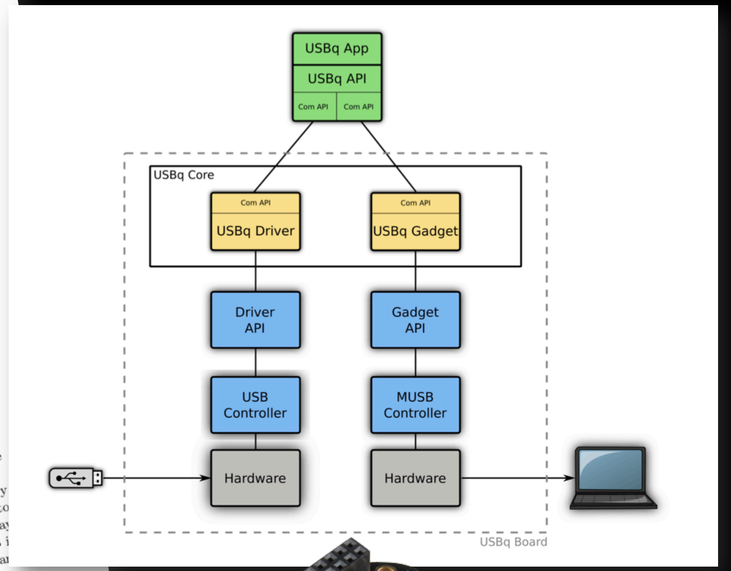
Benoît Camredon
benoit.camredon@airbus.com
Airbus Group

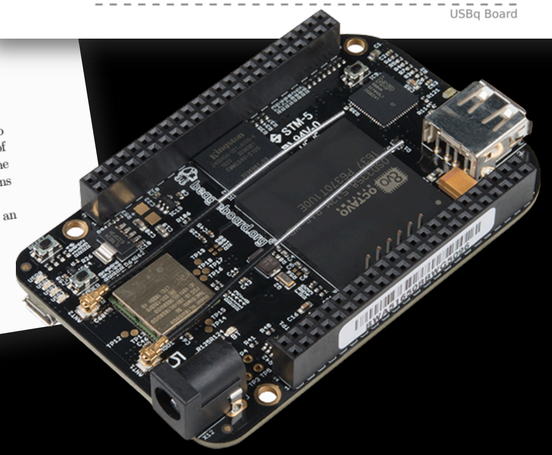
Abstract. The USBiquitous project is a set of open source tools to interact with USB communications. It is composed of a hardware part embedding a Linux system with a bespoke kernel module, and a set of userland scripts and libraries, each designed to tackle a specific problem linked to USB communications. Emulating a USB host, device, or simply performing a man in the middle attack between a host and a device can then be done with a few lines of code in a userland script.

1 Introduction

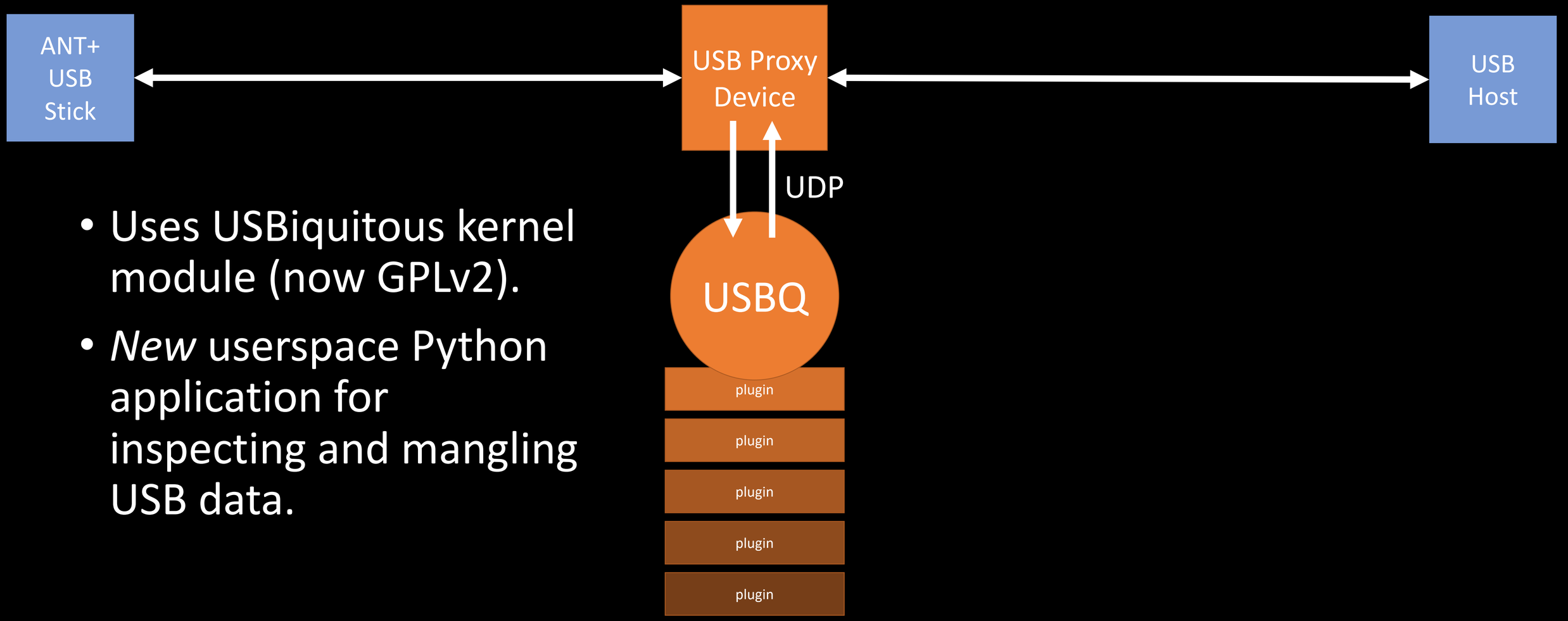
1.1 Context

USB devices are everywhere: keyboards, mice, USB keys, webcams, WiFi ... phones... and USB interfaces are appearing on every equipment [9]. The situation is not going to change with the IOT) development.
tings, USB interfaces are already widespread. Every s one or more USB plugs, that can be used either to r to recharge our phones, iPods... or simply to play be used to retrieve logs or to update firmware. USB i rface and even the most change-averse individuals ar is standard.
interface is a security attack target and more and more ess the robustness of systems using this interface [2, 5], ndatory to improve our USB tools to protect systems new, but nevertheless growing attack vector.
audits on systems having USB interfaces, we needed to derstanding of this protocol, and the low level layers of el that implement it. On this journey to understanding the we have developed the USBiquitous framework, as a means while learning.
ework has accumulated enough useful features to become an during audits of systems that include a USB interface.





USBQ Architecture



- Uses USBiquitous kernel module (now GPLv2).
- *New* userspace Python application for inspecting and mangling USB data.

Stuff Brad Knows

AUG 2019 EDITION

USB Device Drivers and Kernel Code

- Emulate USB host or device functions at the lowest level.
- Behave badly and deviate from the expectations of USB drivers.
- Consider: *GreatFET One and Facedancer!*



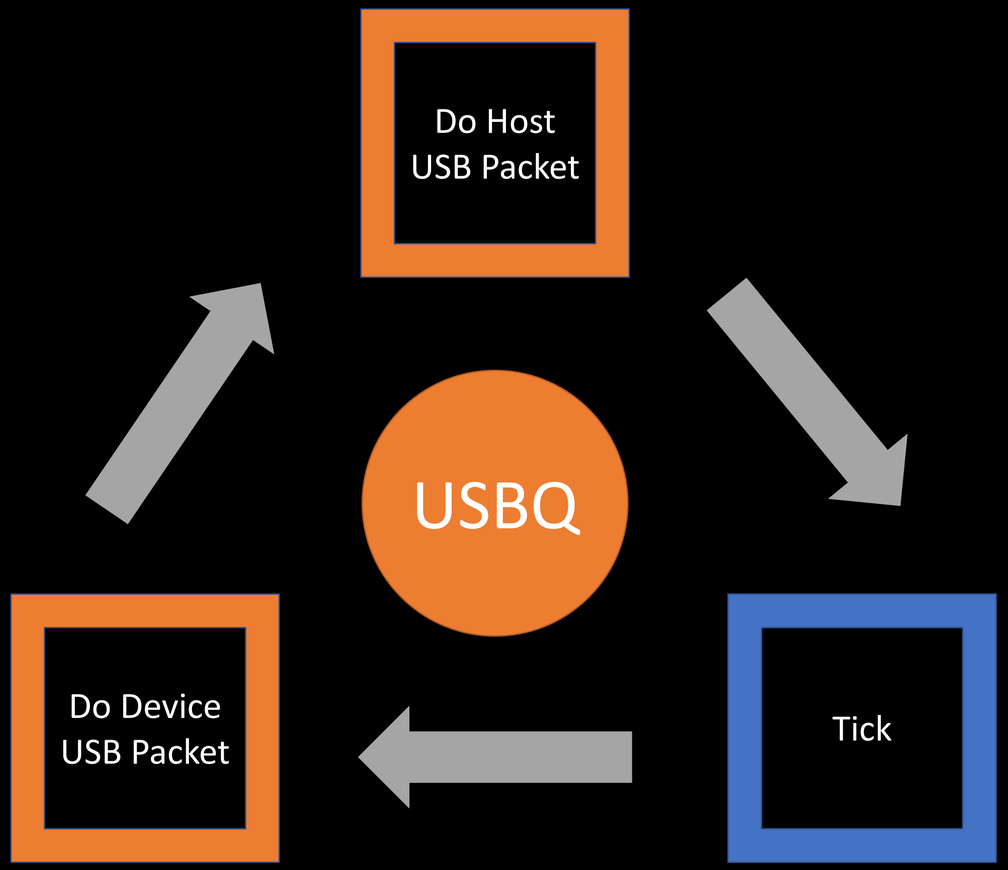
Applications Using USB Peripherals

- Inspect and mangle application-specific payloads transported across a USB bus.
- Use commodity hardware for USB hacking.
- Consider: USBQ

USBQ + Hardware

```
usbq version 0.1.0

Usage: usbq [OPTIONS] COMMAND [ARGS]...

  USBQ: Python programming framework for monitoring and modifying USB
  communications.

Options:
  --debug                 Enable usbq debug logging.
  --logfile FILE          Logfile for --debug output
  --trace                 Trace plugins.
  --dump                  Dump USBQ packets to console.
  --disable-plugin TEXT   Disable plugin
  --enable-plugin TEXT    Enable plugin
  --config FILE           Read configuration from FILE.
  --help                  Show this message and exit.

Commands:
  mitm  Man-in-the-Middle USB device to host communications.

Available plugins:

- decode: Decode raw USBQ driver packets to Scapy representation.
- encode: Encode raw USBQ driver packets to Scapy representation.
- hexdump: Display USBQ packet and hexdump of USB payload.
- ipython: Start an IPython session so that USBQ can be updated on the fly.
- lookfor: look for a specific USB device to appear
- pcap: Write a PCAP file containing USB communications.
- proxy: Send and receive USB packets from a USBQ proxy device using the usbq_core module.
- reload: Monitor usbq_hooks.py file and reload if changed.
- usbq_hooks: Optional user-provided hook implementations automatically loaded from from ./usbq_hooks.py

Default config file: /Users/rbdixon/Library/Application Support/usbq/usbq.cfg
```

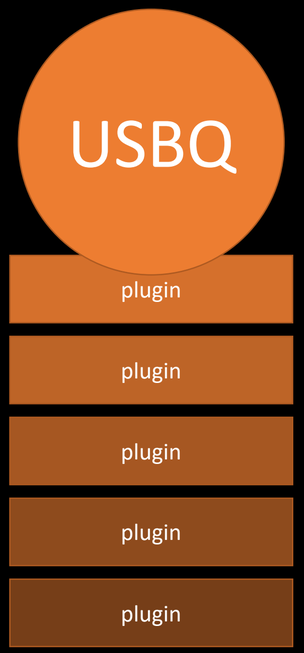
USBQ Main Loop



***DO* Host/Device Packet**

1. Wait for a packet
2. Get the packet
3. Decode the packet
4. Log the packet
5. Modify the packet
6. Encode the packet
7. Send the packet out

USBQ Plugins – Built with Pluggy



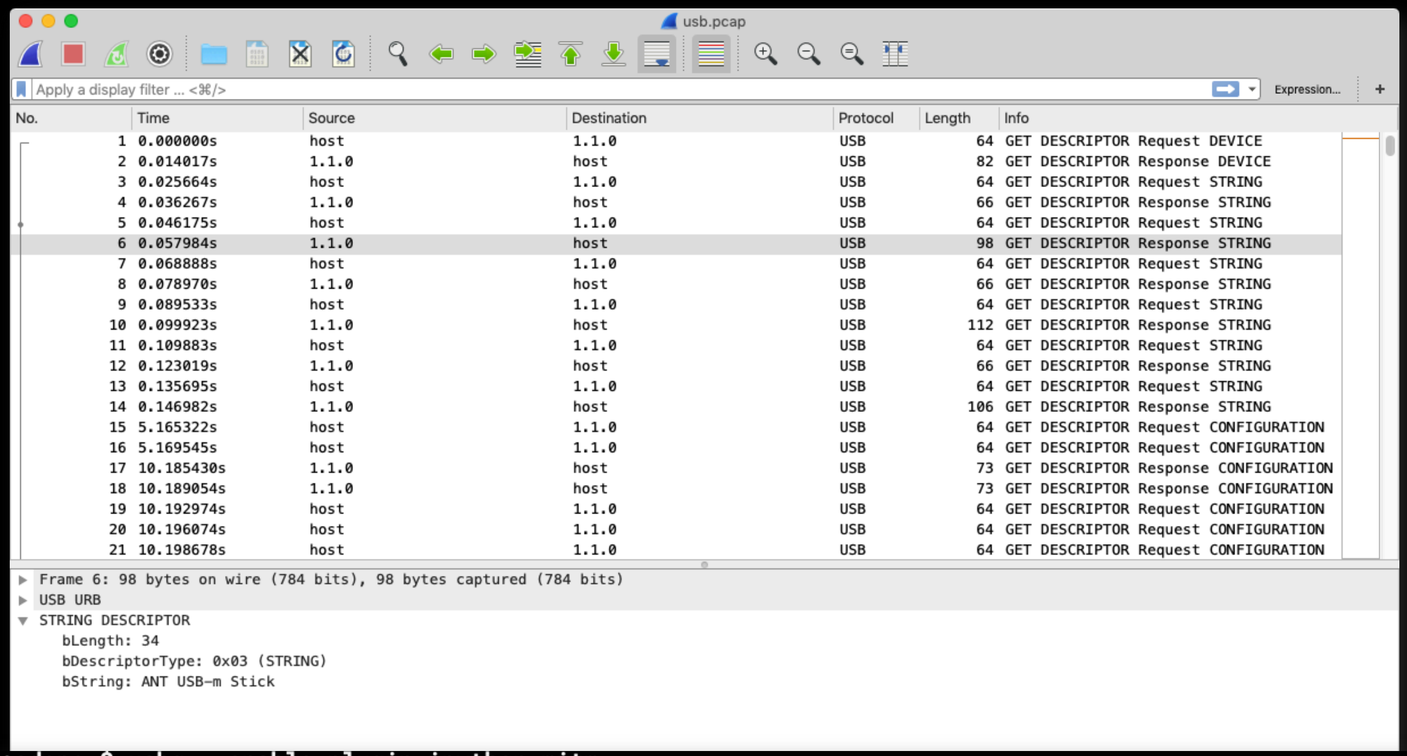
- Defined extension points for plugins to use.
- Plugins can stack and modify the results of plugins lower-down the stack. LIFO-call order.
- Plugins can be distributed as independent Python packages.

included:

- Get and Send USB packets using the proxy kernel module
- Decode/Encode packets to a more useful representation
- Implement convenience features for development

Get Hack’in

- Inspect PCAP
- Modify plugins on-the-fly
- IPython console




```python
class USBQHooks:
    @hookimpl
    def usbq_log_pkt(self, pkt):
        if is_ant(pkt):
            for ant in ant_each(pkt.content.data):
                if FitnessPayload in ant:
                    log.info(repr(ant))
```

```
0|mac:edope $ usbq --enable-plugin ipython mitm
                usbq.pm]: Loading plugins: reload, proxy, pcap, decode, encode, lookfor, ipython, usbq_hooks
usbq.plugins.reload]: Monitoring usbq_hooks.py for changes.
 usbq.plugins.proxy]: Device listen to 0.0.0.0:64240
 usbq.plugins.proxy]: Host send to 10.0.10.90:64241
  usbq.plugins.pcap]: Logging packets to PCAP file usb.pcap
usbq.plugins.lookfor]: Searching for USB device 0fcf:1009
INFO [        usbq.engine]: Starting USB processing engine with IPython UI.
INFO [    keyring.backend]: Loading KWallet
INFO [    keyring.backend]: Loading SecretService
INFO [    keyring.backend]: Loading Windows
INFO [    keyring.backend]: Loading chainer
INFO [    keyring.backend]: Loading macOS
Python 3.7.0 (default, Aug 22 2018, 15:22:33)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.3.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]:
```

What is next for USBQ?

- Release: Visit **usbq.org**
- Need help with / working on:
  - **USBIP support**: Native Linux kernel system for remote USB
    - Device emulation with Function FS
    - Replace USBiquitous kernel module? Need Linux kernel USBIP + Multipoint USB Highspeed Dual-Role Controller (MUSB)
  - **GreatFET One:** Looks awesome… need to fiddle with it!
  - More plugins and tools



Photo by Simon Connellan on Unsplash



eSports

Leet

Automatic

Network

Cheating

Enhancement



EPO Mode



Tour de Pharmacy

Sustain performance with less effort and more guilt!

- Boost your power with a multiplier
- Make the world flat



Slacker Mode

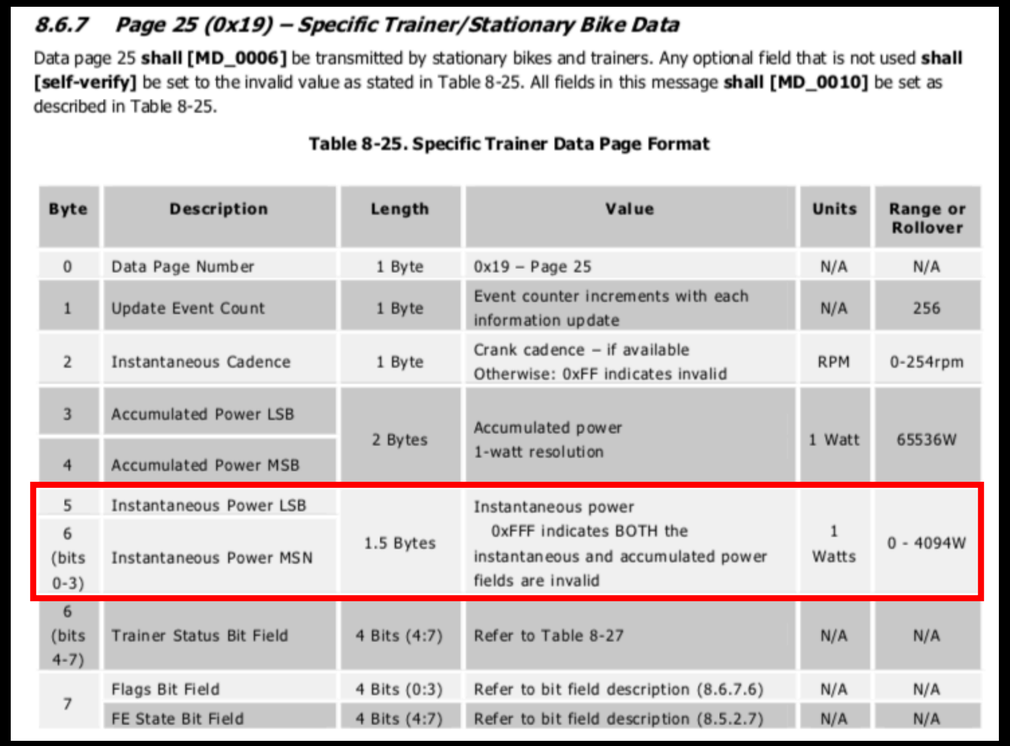
Why even risk sweating a little?



- Automatic pedal POWER
- Cruise control with random jitter
- Terrain-sensitive heart rate and cadence data generation

ELANCE Plugins for USBQ

- Decode ANT+ USB Payload.
- Decode three different ANT+ Payload types: fitness, HRM, and cadence.



8.6.7 Page 25 (0x19) – Specific Trainer/Stationary Bike Data

Data page 25 **shall [MD_0006]** be transmitted by stationary bikes and trainers. Any optional field that is not used **shall [self-verify]** be set to the invalid value as stated in Table 8-25. All fields in this message **shall [MD_0010]** be set as described in Table 8-25.

Table 8-25. Specific Trainer Data Page Format

Byte	Description	Length	Value	Units	Range or Rollover
0	Data Page Number	1 Byte	0x19 – Page 25	N/A	N/A
1	Update Event Count	1 Byte	Event counter increments with each information update	N/A	256
2	Instantaneous Cadence	1 Byte	Crank cadence – if available Otherwise: 0xFF indicates invalid	RPM	0-254rpm
3	Accumulated Power LSB	2 Bytes	Accumulated power 1-watt resolution	1 Watt	65536W
4	Accumulated Power MSB				
5	Instantaneous Power LSB	1.5 Bytes	Instantaneous power 0xFFF indicates BOTH the instantaneous and accumulated power fields are invalid	1 Watts	0 - 4094W
6 (bits 0-3)	Instantaneous Power MSN				
6 (bits 4-7)	Trainer Status Bit Field	4 Bits (4:7)	Refer to Table 8-27	N/A	N/A
7	Flags Bit Field	4 Bits (0:3)	Refer to bit field description (8.6.7.6)	N/A	N/A
	FE State Bit Field	4 Bits (4:7)	Refer to bit field description (8.5.2.7)	N/A	N/A

ANT+ Profile Pages

ANT+

USB Host or Device

USBQ Host, Device, or Management

Cheat the Hard Way with USBQ





HRM

Cadence

Power

ANT+ RF @ 2457 Mhz GFSK
Optional AES

ANT+ USB Stick

USBQ

USB Host

Zwift App

Zwift API

- Modify power
- Modify heart rate
- Modify cadence

- Set grade to “flat”


Terminal Shell Edit View Window Help
100% Tue 1:23 PM Brad Dixon
Terminal — -bash — 137×8
(python-3.7.0) 0|mac:edope $ edope slacker
Backup
ExternalSSD
Cleanup



Photo by Simon Connellan on Unsplash

Could it work?

1. Workouts
2. Online racing
3. Live event racing

Workouts

- Yeah, go ahead and cheat yourself.
- You'll need to use sensible limits.

CONFIRMED





Online Racing*

PLAUSIBLE

- Plausible to *stretch* a mediocre rider into a competitor.
- Use multiple accounts to establish the actual performance limits for verification.
- Build an IRL riding record and a public Strava profile.
- Verification cheats:
  - 2nd power monitor / IRL power monitor
  - Either real height + weight or fake videos
  - Bribe / dodge / fake 3rd party verification lab



* Never actually tried to cheat in an online race nor applied the techniques listed above.

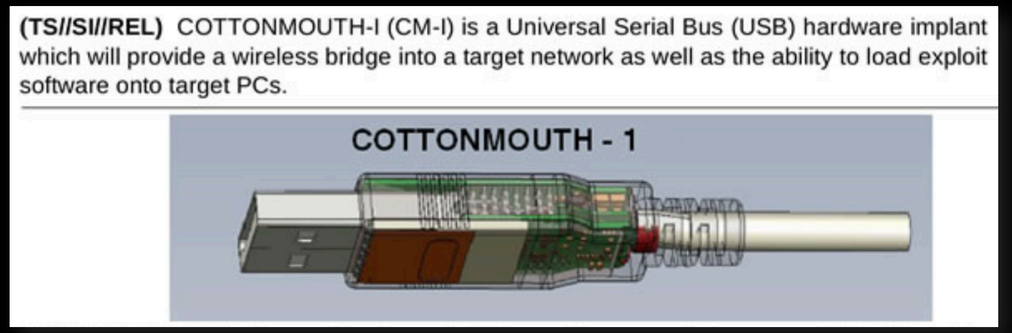
Live Event Racing*

PLAUSIBLE

- This is harder but live events are rare.
- High-stakes events use equipment provided by race.
- Probably can’t fake weigh-in.
- Infiltrate a NSA COTTONMOUTH-I style hacked cable?
- Working on some other techniques, too.



CVRWorldCup



* Never tried this, either. There is no way anyone would believe I’m an elite cyclist. Not even for a second.

Wrap up

1. Overall system not designed for high-integrity competition.
2. Insecure sensor networks and untrusted hardware are not a good foundation for security.
3. Electronics and software are part of cycling. New domains for cheaters to exploit.



Photo by Troy Oldham on Unsplash



Winners never cheat. Cheaters never win.
Hackers sometimes cheat for fun.

edope.bike