Cheating in eSports

How to cheat at virtual cycling using USB hacks

Brad Dixon, Carve Systems
Virtual Cycling is Part of Cycling

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).
- 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.
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
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!*

![ZWIFT e-Racing Performance Limits (men, watts/kg)](image-url)
Vulnerable Sensor Network

- HRM
- Cadence
- Power

- ANT+ RF @ 2457 Mhz GFSK
- Optional AES

ANT+ USB Stick

Paired Devices

- Power Source
  - Power: 68 watts
  - Watts Per Pound: 9.4

- Speed Sensor
  - Cadence: 64 rpm

Options:
- Unpair
- OK
- Just watch
- English
Cheat the Hard Way with USBQ

HRM
- ANT+ RF @ 2457 MHz GFSK
- Optional AES

Cadence

Power

ANT+ USB Stick

USBQ

USB Host

Zwift App

Zwift API

Zwift

USBQ

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.
Stuff Brad Knows

*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!*
USBi-quitous by Benoît Camredon

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

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

**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/rdixon/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
What is next for USBQ?

• Release: Visit [usbq.org](http://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
eSports
Leet
Automatic
Network
Cheating
Enhancement
EPO Mode

*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.

---

**Table B-25, Specific Trainer Data Format**

<table>
<thead>
<tr>
<th>Byte</th>
<th>Description</th>
<th>Length</th>
<th>Value</th>
<th>Units</th>
<th>Range or Rollover</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Data Page Number</td>
<td>1 byte</td>
<td>0x19 – Page 25</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>1</td>
<td>Update Event Count</td>
<td>1 byte</td>
<td>Event counter increments with each update</td>
<td>N/A</td>
<td>256</td>
</tr>
</tbody>
</table>
| 2    | Instantaneous Cadence        | 1 byte | Crank Cadence – if available
Otherwise: 0xff indicates invalid
RPM 0-254rpm | N/A   | 256               |
| 3    | Accumulated Power USB        | 2 bytes| Accumulated power 1-Watt resolution       | 1 Watt| 65536W            |
| 4    | Accumulated Power MSG        | 2 bytes| Accumulated power 1-Watt resolution       | 1 Watt| 65536W            |
| 5    | Instantaneous Power LSB      | 1 byte | Instantaneous power
0xff indicates both the instantaneous and accumulated power fields are invalid
1 Watt 0 - 4604W   | N/A   | N/A               |
| 6    | Instantaneous Power MSN      | 1 byte | Instantaneous power
0xff indicates both the instantaneous and accumulated power fields are invalid
1 Watt 0 - 4604W   | 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

USBQ

- Modify power
- Modify heart rate
- Modify cadence
- Set grade to “flat”

USB Host
Zwift App
Zwift API
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.
Online Racing*

- 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:
  - 2\textsuperscript{nd} power monitor / IRL power monitor
  - Either real height + weight or fake videos
  - Bribe / dodge / fake 3\textsuperscript{rd} party verification lab

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

- 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.

* 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

edope.bike