Fuzzing the USB in your devices
or “How to root your USB-stick”

Olle Segerdahl
olle@nxs.se
whoami

- Technical IT-sec background

- Currently in Information Assurance
  - When you're sure it does what it's specified to ...
  - ... how sure are you “it doesn't do anything else”? 
Motivation

“Security will not get better until tools for practical exploration of the attack surface are made available.”

Joshua Wright – willhackforsushi.com
Motivation

• Explore USB attack surface
  … of devices!
  – Mobile Devices
  – “Secure” USB Drives
  – “PinPad” Card Readers
  – and more…
Intro to USB

- Host-controlled “bus”
  - Initiator / Responder - “Host” / “Function”
  - IN / OUT
Intro to USB

• Devices carry “descriptors”
  – Hosts “enumerate” them
  – Configurations, Interfaces, Endpoints
Intro to USB

• Device Classes
  – Indicated in Device descriptor ...
    ... or in Interface descriptors

<table>
<thead>
<tr>
<th>Device Classes</th>
<th>CDC Sub Classes</th>
<th>CDC Interface Protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>01h Audio</td>
<td>01h Direct Line</td>
<td>01h ITU-T V.250</td>
</tr>
<tr>
<td>02h CDC</td>
<td>02h Abstract</td>
<td>02h PCCA-101</td>
</tr>
<tr>
<td>03h HID</td>
<td>03h Telephone</td>
<td>03h PCCA-101 + Annex O</td>
</tr>
<tr>
<td>05h Physical</td>
<td>04h Multi-Channel</td>
<td>04h GSM 7.07</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Intro to USB

• Transfer types
  – Control
  – Bulk
  – “Interrupt”
  – Isochronous
## Intro to USB

### Transfer types

- **Control**

- **Endpoint 0 (EP0), “default” endpoint**

<table>
<thead>
<tr>
<th>Field</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bmRequestType</td>
<td>8 bits</td>
<td>Direction, Type, Recipient</td>
</tr>
<tr>
<td>bRequest</td>
<td>8 bits</td>
<td>Specific request value</td>
</tr>
<tr>
<td>wValue</td>
<td>16 bits</td>
<td>Request specific parameter</td>
</tr>
<tr>
<td>wIndex</td>
<td>16 bits</td>
<td>Request specific parameter</td>
</tr>
<tr>
<td>wLength</td>
<td>16 bits</td>
<td>Bytes to transfer (if any)</td>
</tr>
</tbody>
</table>

**Standard Device Requests**

<table>
<thead>
<tr>
<th>Request Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00h</td>
<td>GET_STATUS</td>
</tr>
<tr>
<td>01h</td>
<td>CLEAR_FEATURE</td>
</tr>
<tr>
<td>03h</td>
<td>SET_FEATURE</td>
</tr>
<tr>
<td>05h</td>
<td>SET_ADDRESS</td>
</tr>
<tr>
<td>06h</td>
<td>GET_DESCRIPTOR</td>
</tr>
<tr>
<td>07h</td>
<td>SET_DESCRIPTOR</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Intro to USB

• Transfer types
  - Bulk
    • Asynchronous ("bursty")
    • Use available bandwidth ("laggy")
    • 2 endpoints (IN/OUT) make a "Pipe"
Intro to USB

• Transfer types
  – Control
  – Bulk
  – “Interrupt”
  – Isochronous
Fuzzing USB Hosts

- Darrin Barrall, David Dewey (2005)
- Moritz Jodeit, Martin Johns (2009)
- Rafael Dominguez Vega (2009)
- Tobias Müller (2010)
- Travis Goodspeed's Facedancer
Fuzzing USB Devices

- Prior work
  - Pod2g, posixninja in 2010
  - Andy Davis @ BHUSA 2011

- Facedancer20

- libusb!
Fuzzing with libusb

• libusb
  – Library for developing userland drivers
  – Works on Linux, Windows, MacOS
  – Nice introduction by Peter Stuge @ 27C3

• Limitations
  – Not expecting some “invalid” input
    • Tends to crash instead of error out
  – Linux kernel performs sanity checks
Building a simple fuzzer

- PyUSB – python interface to libusb
- Let's target Control Transfers
- Simple iterative loops around ctrl_transfer()

Demo Time!
Building a simple fuzzer

#!/usr/bin/env python

import sys
import usb.core

def TestCtrlTransfer(device, rt, r, v, i):
    for size in (2, 10, 100, 1000, 4000):
        try:
            res = device.ctrl_transfer(rt&0x80, r, v, i, bytearray().fromhex('u'[::-1]*size))
        except usb.core.USBError as e:
            if (e.backend_error_code != -9): # ignore LIBUSB_ERROR_PIPE
                print('OUT %0.2x %0.2x %0.4x %0.4x err(%i) len(%u)' % (rt, r, v, i, e.backend_error_code, size))
        try:
            res = device.ctrl_transfer(rt|0x80, r, v, i, size)
        except usb.core.USBError as e:
            if (e.backend_error_code != -9): # ignore LIBUSB_ERROR_PIPE
                print('IN  %0.2x %0.2x %0.4x %0.4x err(%i) len(%u)' % (rt, r, v, i, e.backend_error_code, size))

arg = sys.argv[1].split(':')
device = usb.core.find(idVendor=int(arg[0], 16), idProduct=int(arg[1], 16))

for t in range(0, 0x04): # bmRequestType.Type
    for r in range(0, 0x04): # bmRequestType.Recipient
        for q in range(0, 0x100): # bRequest
            for v in range(0, 0x1000): # wValue
                for i in range(0, 0x1000): # wIndex
                    TestCtrlTransfer(device, r|(t<<4), q, v, i)
Building a simple fuzzer

• Adding some target control
  – Monitoring
    • Simple: ctrl_transfer(GET_STATUS)
    • Better: use a class-specific request
  – Resuming
    • Simple: device.reset() to recover device
    • Better: use external hub for power control
Building a simple fuzzer

def is_alive(device):
    res = ""
    try:
        res = device.ctrl_transfer(0x80, 0, 0, 0, 2)
    except usb.core.USBError as e:
        if e.backend_error_code == -4:  # LIBUSB_ERROR_NO_DEVICE
            print "Device not found!"
            sys.exit()

        if e.backend_error_code == -3:  # LIBUSB_ERROR_ACCESS
            print "Access denied to device!"
            sys.exit()

        print "GET_STATUS returned error %i" % e.backend_error_code
        return False

    if len(res) != 2:
        print "GET_STATUS returned %u bytes: %s" % (len(res), binascii.hexlify(res))
        return False

    return True
Building a simple fuzzer

def is_alive(device):
    try:
        device.write(bytearray().fromhex(u'5534243E019EA85000008000A2800000000000010000000000000'))
    except usb.core.USBError as e:
        if e.backend_error_code == -4: # LIBUSB_ERROR_NO_DEVICE
            raise Exception('Function check failed: device not present!')
        elif e.backend_error_code == -6: #LIBUSB_ERROR_BUSY
            raise Exception('Function check failed: function is busy!')
        else:
            print "Function check failed: usb error %i" % e.backend_error_code
            return False

    try:
        res = ep_in.read(readlen)
    except usb.core.USBError as e:
        print "Function check failed: usb error %i" % e.backend_error_code
        return False

    if len(res) != 0x206:
        print "Function check returned %u bytes: %s" % (len(res), binascii.hexlify(res))
        return False

    return True
Extending our reach

- Reach more complex code!
- Device Classes
  - Audio, CDC, HID, Image, Printer, Mass Storage, Hub, Smart Card, Video, Wireless Controller, DFU, Vendor Specific

http://www.usb.org/developers/defined_class/
http://www.usb.org/developers/devclass_docs/
First attempt: Peach

- Very easy to add pyUSB "Publisher"
- Data modelling and test cases in XML
  - Very cumbersome to work with state
- Target control framework: "Agents"
  - Not built for controlling local devices
Second attempt: Scapy

- Scapy for data modelling
  - Abstracts data as “Layers” of “Packets”
  - Keeps everything in python!
  - Easy use of python code for “fixups”

- Easy reuse of code with Facedancer!
  - Travis Goodspeed, Ryan Speers
Demo Time!
Helpful tools

- Total Phase Beagle USB
  - http://www.totalphase.com/protocols/usb/
- Travis Goodspeed's Facedancer
  - http://goodfet.sourceforge.net/
Get the code

https://github.com/ollseg/usb-device-fuzzing

Will gladly accept pull requests...
Examples of bugs found
First bug found

- Atmel AT91SAM7 example USB code
  - Prevalent in devices using Atmel MCUs
- Off-by-one on string descriptor index
  - `ctrl_transfer(0x80, 6, 3<<8 | i+1, 0, len)`
static void GetDescriptor(
    const USBDDriver *pDriver,
    unsigned char type,
    unsigned char index,
    unsigned int length)
{
    // Check the descriptor type
    switch (type) {

    case USBGenericDescriptor_STRING:
        TRACE_INFO_WP("Str%d ", index);

        // Check if descriptor exists
        if (index > numStrings) {
            USBD_Stall(0);
        }
        else {
            pString = pStrings[index];
        }
    
    }
Bugs in Nokia phones

- Random crashes while fuzzing
- Seemed related to “large” Control Transfers
- Looks like a stack buffer overwrite
- Threw together a 5-line python PoC

DEMO TIME!
Bugs in USB-Sticks

SLIDE REDACTED
Exploiting a USB-stick

SLIDE REDACTED
Future work

- Support more protocols
- Reaching deeper into targets
- https://wiki.mozilla.org/WebUSB/
- Travis Goodspeed's Facedancer!
Thank You!

Olle Segerdahl
olle@nxs.se

https://github.com/ollseg/usb-device-fuzzing

Please remember to fill out the feedback forms!