USB arsenal for masses

Krzysztof Opasiak

Samsung R&D Institute Poland
Agenda

USB protocol intro

USB sniffing & modification

USB security testing

Summary

Q & A
USB protocol intro
What USB is about?

It’s about providing services!

- Storage
- Printing
- Ethernet
- Camera
- Any other
Endpoints...

- Device may have up to 31 endpoints (including ep0)
- Each of them gets a unique endpoint address
- Endpoint 0 may transfer data in both directions
- All other endpoints may transfer data in one direction:
  - **IN** Transfer data from device to host
  - **OUT** Transfer data from host to device
Endpoint types

• Control
  • Bi-directional endpoint
  • Used for enumeration
  • Can be used for application

• Bulk
  • Used for large data transfers
  • Used for large, time-insensitive data (Network packets, Mass Storage, etc).
  • Does not reserve bandwidth on bus, uses whatever time is left over
Endpoint types

- **Interrupt**
  - Transfers a small amount of low-latency data
  - Reserves bandwidth on the bus
  - Used for time-sensitive data (HID)

- **Isochronous**
  - Transfers a large amount of time-sensitive data
  - Delivery is not guaranteed (no ACKs are sent)
  - Used for Audio and Video streams
  - Late data is as good as no data
  - Better to drop a frame than to delay and force a re-transmission
USB device
USB bus

- USB is a Host-controlled bus
- Nothing on the bus happens without the host first initiating it.
- Devices cannot initiate any communication.
- The USB is a Polled Bus.
- The Host polls each device, requesting data or sending data.
USB transport (Link Layer)

IN

- Host sends an IN token
- If the device has data:
  - Device sends data
  - Host sends ACK
- else
  - Device sends NAK
  - Host will retry until timeout
USB transport (Link Layer)

**OUT**

- Host sends an OUT token
- Host sends the data (one packet)
- If device accepts data transfer:
  - Device sends an ACK
- else
  - Device sends an NAK
  - Host will retry until success or timeout

* PING, NYET - bandwidth savers
USB transfer vs transaction

- **Transaction**
  - Delivery of data to endpoint
  - Limited by \texttt{wMaxPacketSize}

- **Transfer**
  - One or more transactions
  - May be large or small
  - Completion conditions

Source: [10]
USB Request Block

- Kernel provides hardware independent API for drivers
- URB is a kind of envelope for data
- This API is asynchronous
  - `usb_alloc_urb()`
  - `usb_free_urb()`
  - `usb_submit_urb()`
  - `usb_unlink_urb()`
  - `usb_kill_urb()`

```c
struct urb {
    struct list_head urb_list;
    struct usb_device *dev;
    unsigned int pipe;
    int status;
    unsigned int transfer_flags;
    void *transfer_buffer;
    u32 transfer_buffer_length;
    u32 actual_length;
    unsigned char *setup_packet;
    void *context;
    usb_complete_t complete;
};
```
USB sniffing & modification
USBMon

- Kind of logger for URB related events:
  - submit()
  - complete()
  - submit_error()

- So it’s not going to show you link layer USB tokens!
- Text interface
- Binary Interface
- One instance for each USB bus
Data validity

- Data in URB buffer may is not always valid
- Validity depends on transfer results
- And on endpoint direction:

<table>
<thead>
<tr>
<th></th>
<th>IN</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>submit()</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>complete()</td>
<td>YES</td>
<td>NO</td>
</tr>
</tbody>
</table>
Good old friend Wireshark - DEMO
USBProxy[1]

- Framework for USB MITM
- In theory, works on any SBC with UDC and HCD
- In practice, works only on BBB with custom kernel image
- Uses libusb & GadgetFS
- Can intercept only one device
- Still needs some love…
Just a logic analyzer...

• For Full or Low Speed devices definitely yes!
• High speed bus signaling is 480 Mbit/s
• So you would need to probe with 1GHz frequency
OpenVizsla[8]
OpenVizsla host tools - DEMO

- ovctl.py
- ViewSB
- Wireshark!
USB security testing
FaceDancer[3]

- **Hardware**

- **Software**
  - Python framework for emulating USB devices
BTW 2x Facedancer MITM

Source: [12]
GreatFET[4]

- Hacking platform
- Initially created for Radio Hacking
- NXP LPC4330 MCU
- 1x HS USB
- 1x FS USB
- Compatible with Facedancer software

Source: [6]
GreatFET Rhododendron[5]

- GreatFET neighbor with USB3343 for sniffing
- Unfortunately GreatFET does not have any external RAM memory...

Source: [5]
umap2[13]

- umap2scan
- umap2emulate
- umap2stages
- umap2fuzz (kitty-based)

**Supported backends:**
- Facedancer (and GreatFET)
- Raspdancer
- GadgetFS (partially supported)

- VM-based fuzzing
- Hypervisor specific
- Limited by hypervisor implementation
- Scapy-based fuzzing

Source: [11]
syzcaller-based architecture[7]

- DummyHCD-based
- GadgetFS/Custom module
- Use syzcaller to generate USB traffic
- Require “description” files

![Syzkaller USB Fuzzing Approach Diagram]

Source: [7]
Summary

- You don’t need to spend a lot money to sniff USB traffic
- There is a number of Open Source and Open Hardware USB tools
- There is no perfect architecture for testing USB security
Thank you!

Krzysztof Opasiak
Samsung R&D Institute Poland

+48 605 125 174
k.opasiak@samsung.com
References


References II


[9] *OpenVizsla USB Analyzer - fail0ver article*. URL: https://fail0verflow.com/blog/2014/ov3-hardware/.

References III

