Making USB Great Again with USBFILTER

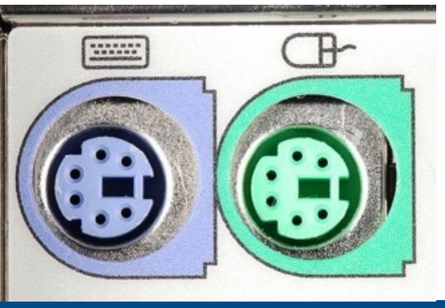
Dave Tian*, Nolen Scaife*, Adam Bates**, Kevin Butler*, Patrick Traynor*
*University of Florida, Gainesville, FL
**University of Illinois, Urbana-Champaign, IL

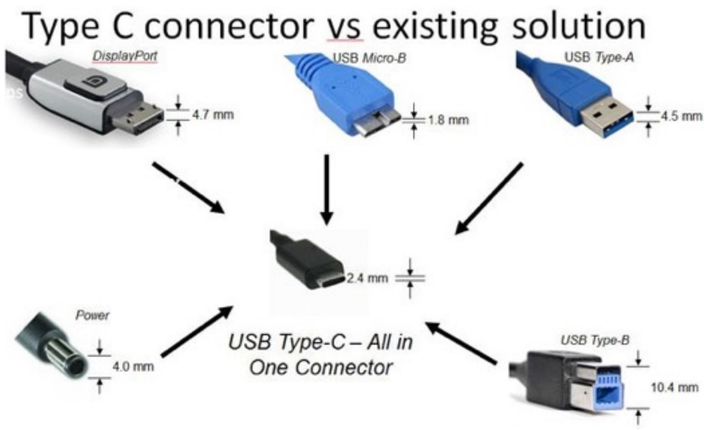
USENIX Security'16, Austin, TX
Aug 11, 2016

Make it real…

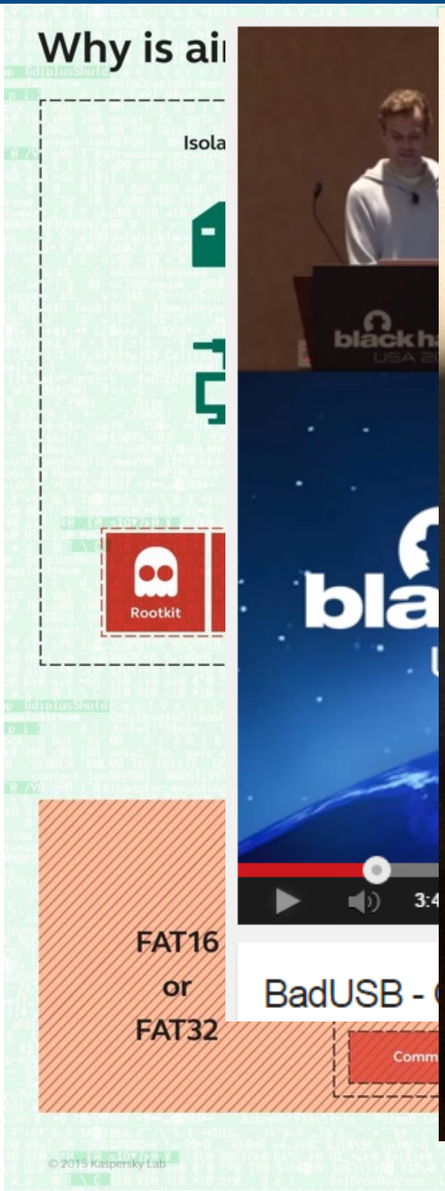
Why USB was great

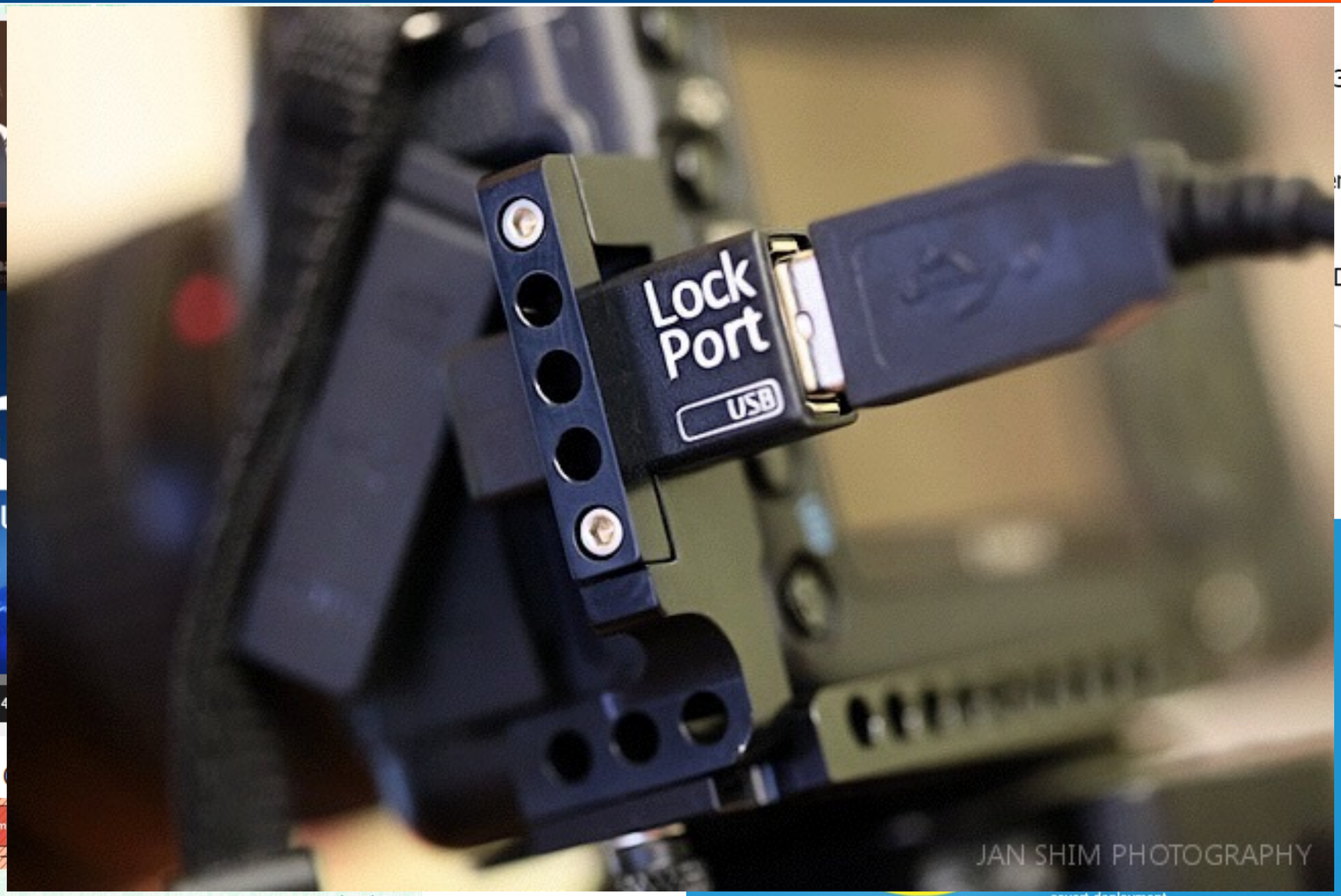
- Universal Serial Bus
  - USB 1.0/2.0/3.0/3.1/Type-C
- Speed
  - 10 gigabits per second
- Ubiquitous



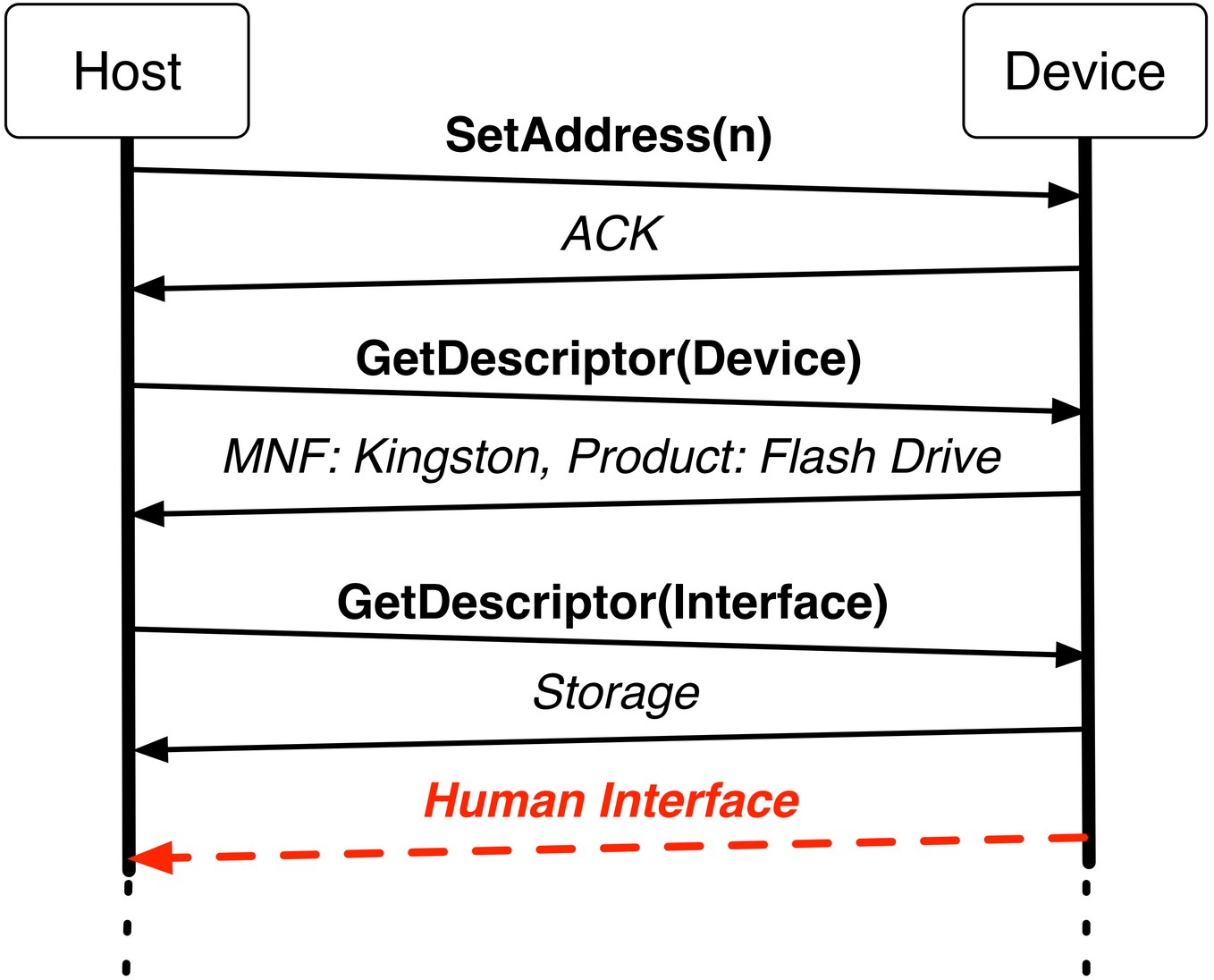


Why USB is not great anymore

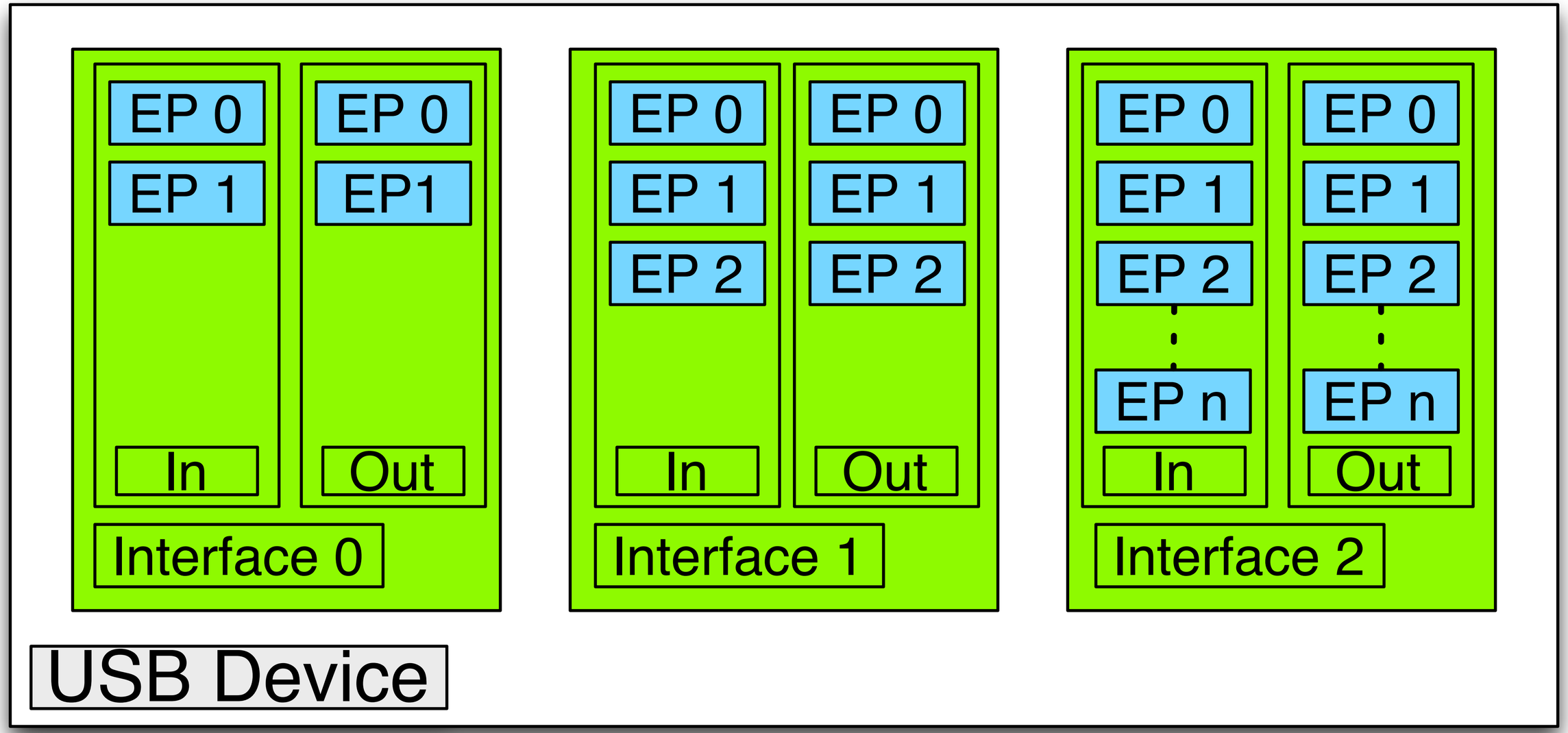




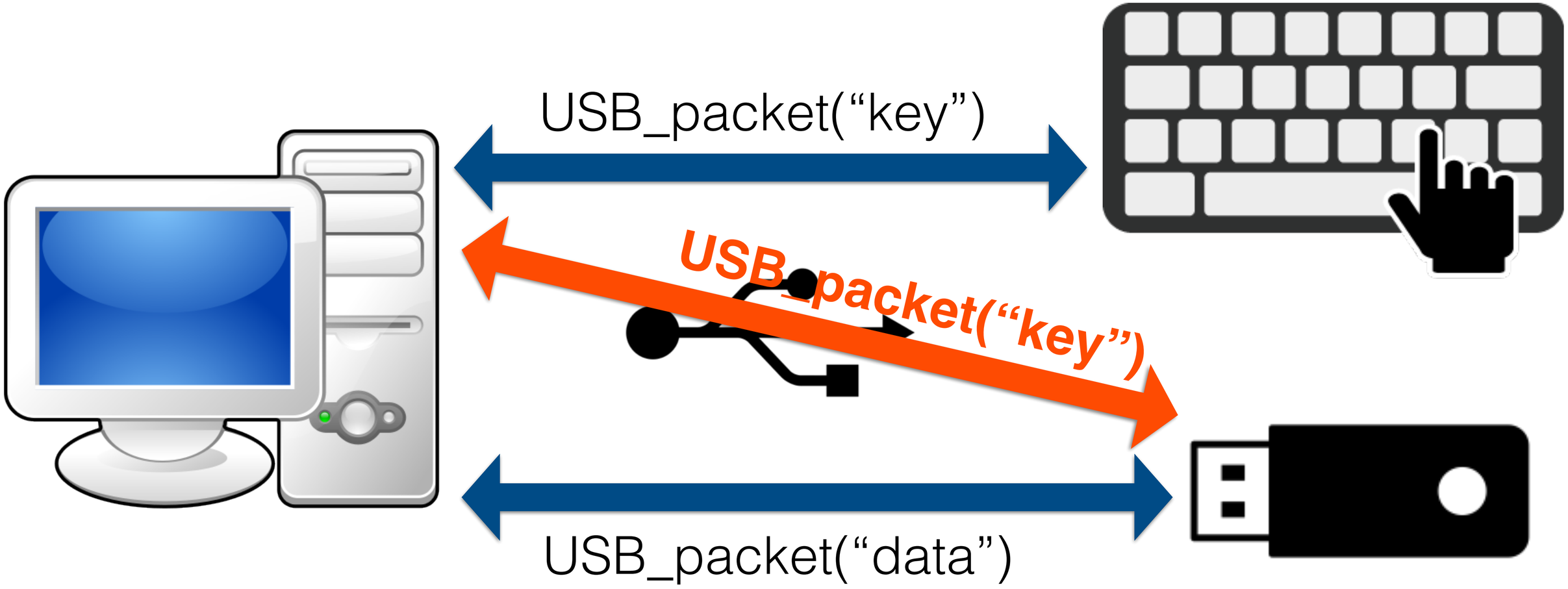
USB enumeration



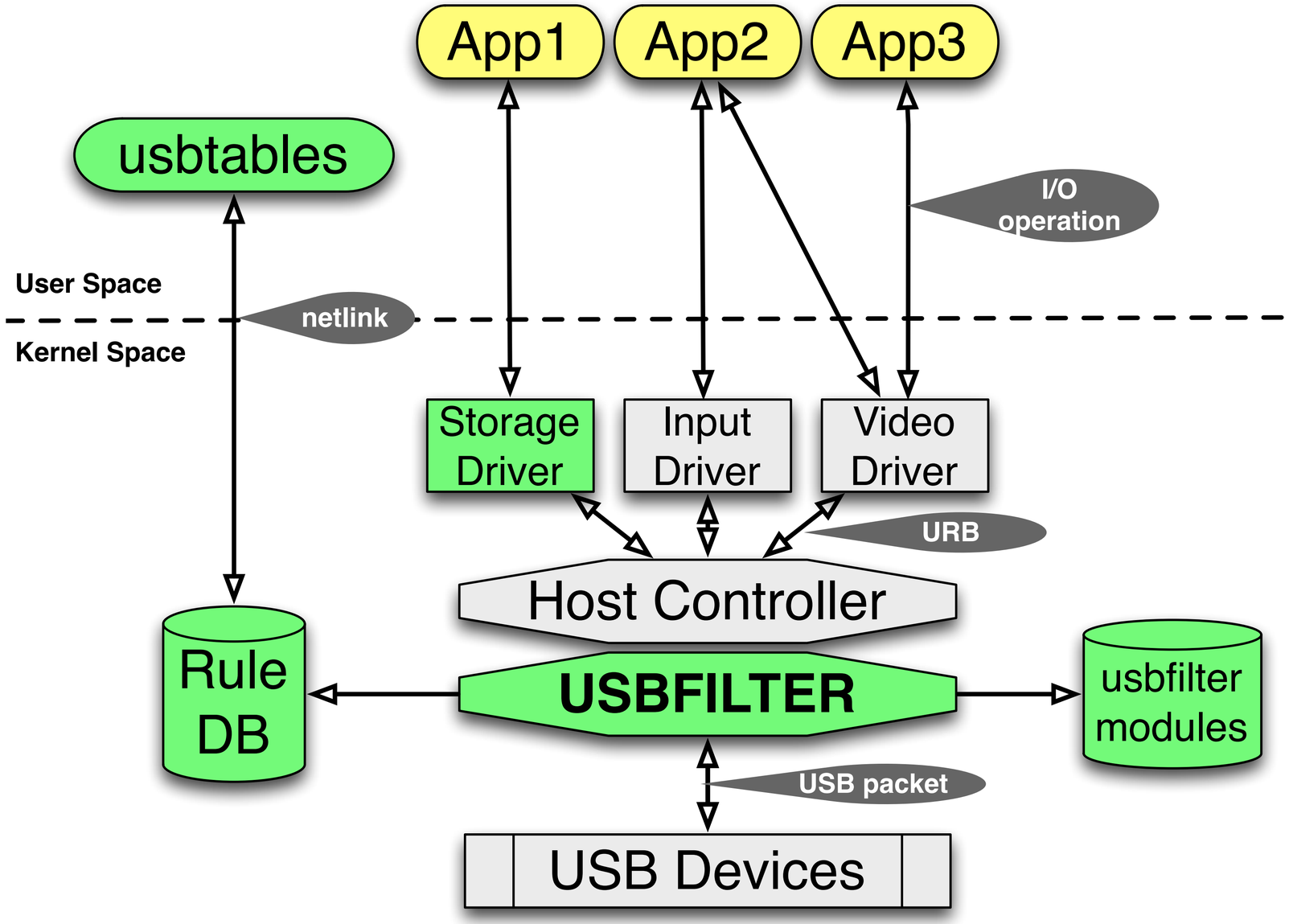
USB device


EP 0
EP 1
In
EP 0
EP1
Out
Interface 0
EP 0
EP 1
EP 2
In
EP 0
EP 1
EP 2
Out
Interface 1
EP 0
EP 1
EP 2
EP n
In
EP 0
EP 1
EP 2
EP n
Out
Interface 2
USB Device

USB packet



USBFILTER

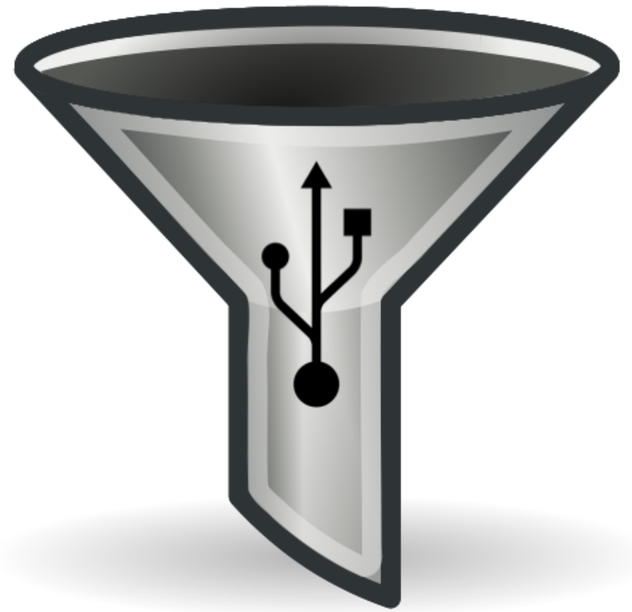

App1
App2
App3
usbtables
I/O operation
User Space
netlink
Kernel Space
Storage Driver
Input Driver
Video Driver
URB
Host Controller
Rule DB
USBFILTER
usbfilter modules
USB packet
USB Devices

Goals

- Complete mediation
- Tamperproof
- Verifiability

Reference Monitor

- Granularity
- Extensibility



Rule constructions

Process	pid,ppid,pgid,uid,euid,gid,egid,comm
Device	bus#,dev#,port#,if#,devpath,manufacturer,product,serial
Packet	type,direction,endpoint,address
LUM	name

Rule consistency

- General conflict

$$general\_conflict(R_a, R_b) \leftarrow$$
$$\forall C_i \ni \mathscr{C} :$$
$$(\exists C_i^a \ni R_a \wedge \exists C_i^b \ni R_b \wedge value(C_i^a) \neq value(C_i^b)) \vee$$
$$(\exists C_i^a \ni R_a \wedge \nexists C_i^b \ni R_b) \vee$$
$$(\nexists C_i^a \ni R_a \wedge \nexists C_i^b \ni R_b).$$

- Weak conflict

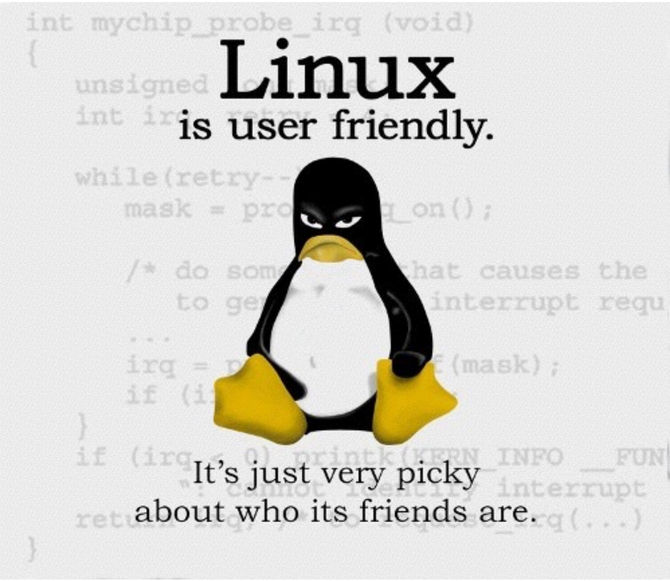
$$weak\_conflict(R_a, R_b) \leftarrow$$
$$general\_conflict(R_a, R_b) \wedge action(R_a) = action(R_b).$$

- Strong conflict

$$strong\_conflict(R_a, R_b) \leftarrow$$
$$general\_conflict(R_a, R_b) \wedge action(R_a) \neq action(R_b).$$

Linux USBFILTER Module (LUM)

- User-defined extension for USBFILTER
  - <linux/usbfilter.h>
- Rule construction unit
  - writing new rules with LUM
- Looking into the USB packet
  - SCSI commands, IP packets, HID packets, and etc.



LUM: detect the SCSI write cmd

```
20 int lbsw_filter_urb(struct urb *urb)
21 {
22     char opcode;
23
24     /* Has to be an OUT packet */
25     if (usb_pipein(urb->pipe))
26         return 0;
27
28     /* Make sure the packet is large enough */
29     if (urb->transfer_buffer_length <= LUM_SCSI_CMD_IDX)
30         return 0;
31
32     /* Make sure the packet is not empty */
33     if (!urb->transfer_buffer)
34         return 0;
35
36     /* Get the SCSI cmd opcode */
37     opcode = ((char *)urb->transfer_buffer)[LUM_SCSI_CMD_IDX];
38
39     /* Current only handle WRITE_10 for Kingston */
40     switch (opcode) {
41     case WRITE_10:
42         return 1;
43     default:
44         break;
45     }
46
47     return 0;
48 }
```

Overview

- USBFILTER - 27 kernel source files
    - 4 new files, 23 modified files
    - Across USB, SCSI, Block, and Networking subsystems
- USBTABLES
    - Internal Prolog engine
    - 21 rule constructions

Stop BadUSB attacks

For my keyboard/mouse:

```
usbtables -a mymouse -v busnum=1,devnum=4,portnum=2,
      devpath=1.2,product="USB Optical Mouse",
      manufacturer=PixArt -k types=1 -t allow
```

```
usbtables -a mykeyboard -v busnum=1,devnum=3,
      portnum=1,devpath=1.1,
      product="Dell USB Entry Keyboard",
      manufacturer=DELL -k types=1 -t allow
```

```
usbtables -a noducky -k types=1 -t drop
```

Pin Skype to webcam

For Logitech webcam C310:

```
usbtables -a skype -o uid=1001,comm=skype -v
        serial=B4482A20 -t allow
```

```
usbtables -a nowebcam -v serial=B4482A20 -t drop
```

Stop data exfiltration

For any USB storage devices:

```
usbtables -a nodataexfil4
        -l name=block_scsi_write -t drop
```

Just speaker, no microphone

For Logitech USB headset:

```
usbtables -a logitech-headset -v ifnum=2,product=
      "Logitech USB Headset",manufacturer=Logitech -k
      direction=1 -t drop
```

Charge safe

For Nexus 4:

```
usbtables -a n4-charger -v product="Nexus 4" -t drop
```

For any phone:

```
usbtables -a charger -v busnum=1,portnum=4 -t drop
```

Scalability

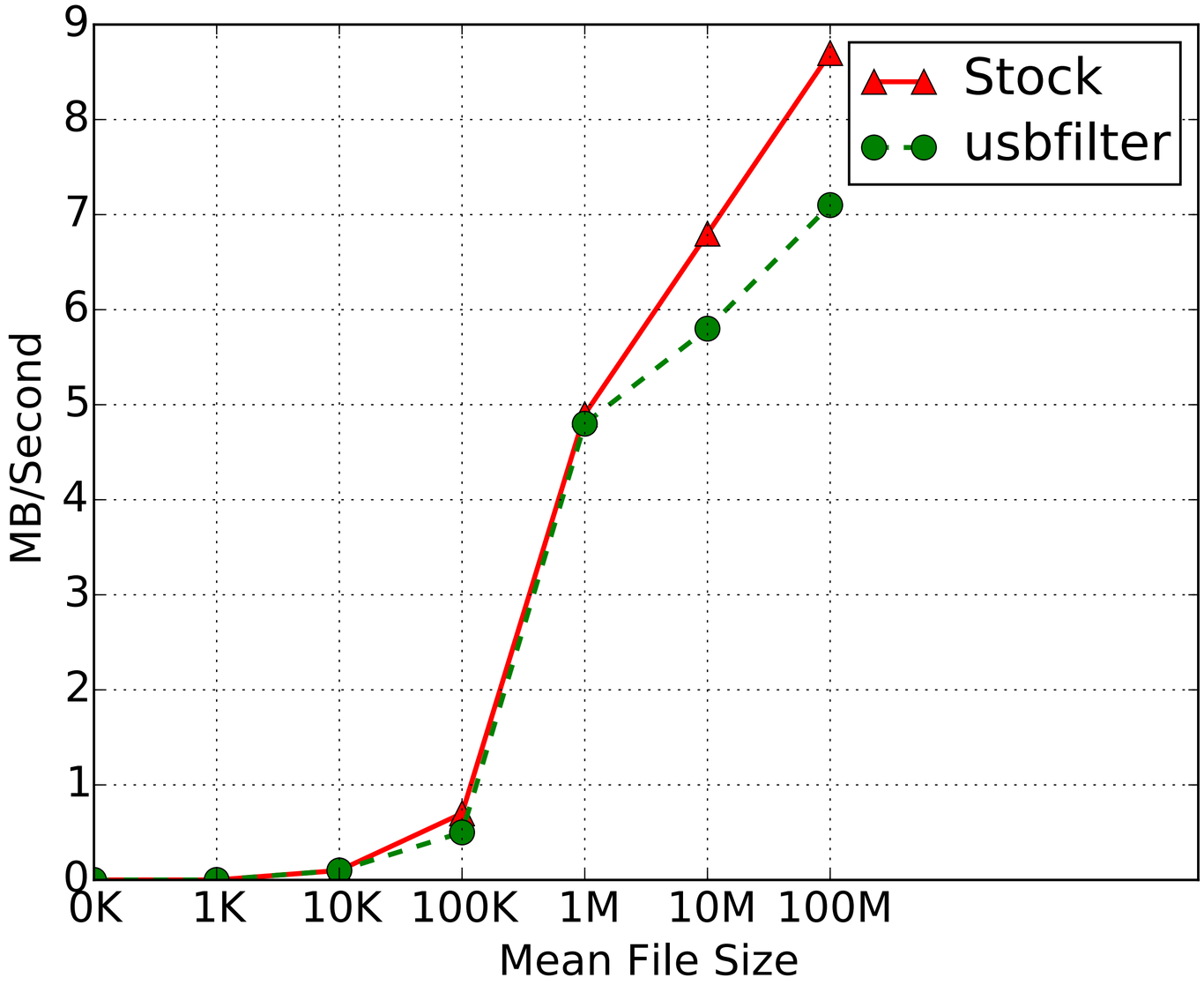
USBTABLES:

Adding a new rule	Avg (ms)
20 Base Rules	5.9
100 Base Rules	5.9

USBFILTER:

Packet filtering	Avg (µs)
20 Base Rules	2.6
100 Base Rules	9.7

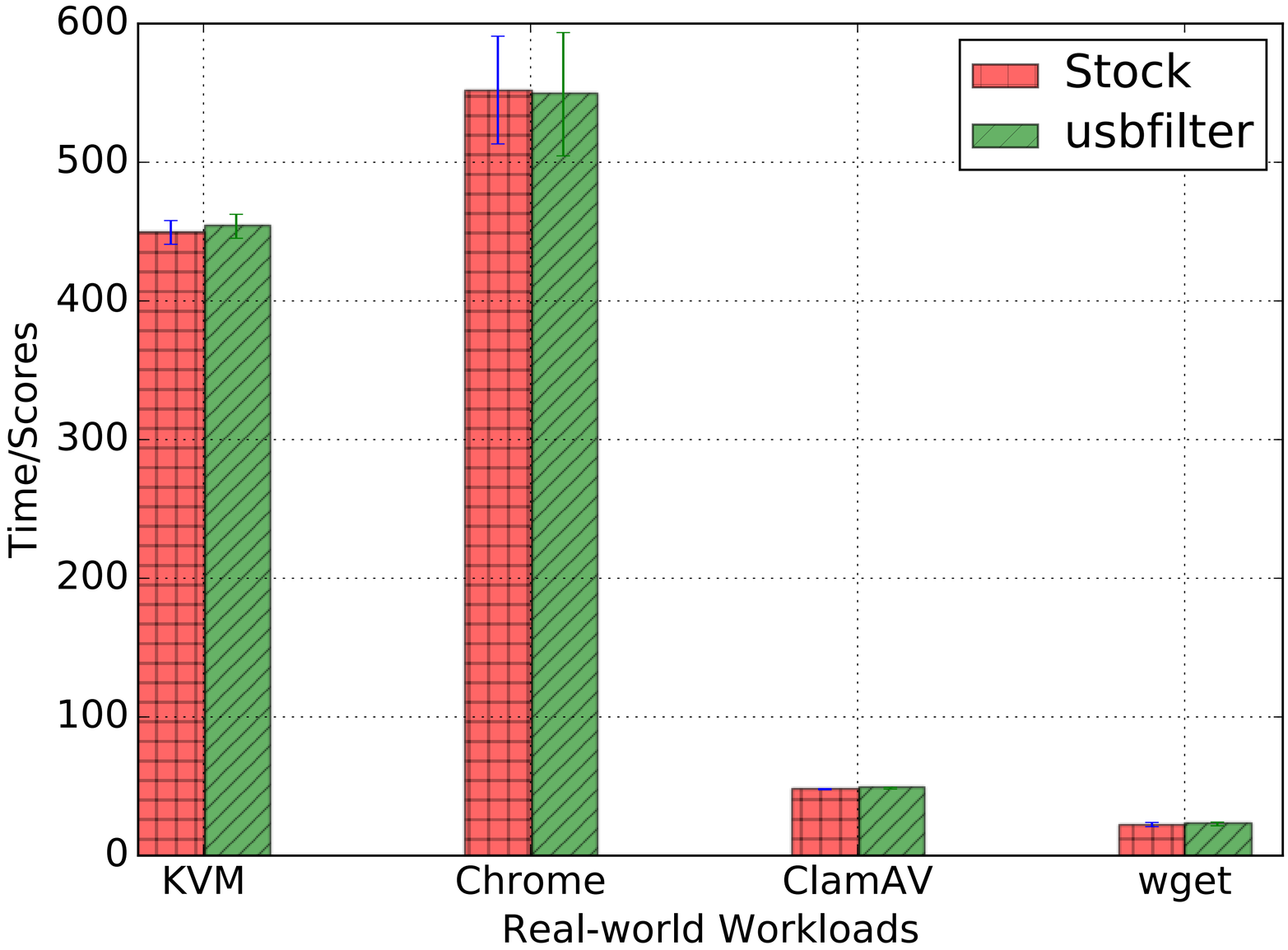
Throughput


MB/Second
0
1
2
3
4
5
6
7
8
9
0K
1K
10K
100K
1M
10M
100M
Mean File Size
Stock
usbfilter

Latency

Latency (µs)	1 KB	10 KB	100 KB	1 MB	10 MB	100 MB
Stock	97.6	98.1	99.2	105.5	741.7	5177.7
USBFILTER	97.7	98.2	99.6	106.3	851.5	6088.4
Overhead	0.1%	0.1%	0.4%	0.8%	14.8%	17.6%

Performance in real world



Limitations & Future Work

- IRQ contexts
- Vendor-specific drivers
- Response-path filtering
- Making it faster - BPF
- More useful LUMs
- Usability - targeting administrators

Conclusion

- USBFILTER
  - A USB layer firewall in the Linux kernel
- USBTABLES
  - A user-space tool to manage policies/rules
- Controlling USB device behaviors
  - Defending against BadUSB attacks
  - Limiting USB device functionalities
- Introducing minimum overhead

Q&A

Get USBFILTER now:
https://github.com/daveti/usbfilter

All bugs are introduced by:
root@davejingtian.org

Thanks!

USBTABLES -h

```
-d|--debug    enable debug mode
-c|--config   path to configuration file (TBD)
-h|--help     display this help message
-p|--dump     dump all the rules
-a|--add      add a new rule
-r|--remove   remove an existing rule
-s|--sync     synchronize rules with kernel
-e|--enable   enable usbfilter
-q|--disable  disable usbfilter
-b|--behave   change the default behavior
-o|--proc     process table rule
-v|--dev      device table rule
-k|--pkt      packet table rule
-l|--lum      LUM table rule
-t|--act      table rule action
-------------------------------
proc: pid,ppid,pgid,uid,euid,gid,egid,comm
dev: busnum,devnum,portnum,ifnum,devpath,product,
     manufacturer,serial
pkt: types,direction,endpoint,address
lum: name
behavior/action: allow|drop
```

A LUM written by dtrump

```c
/*
 * lbsw - A LUM kernel module
 * used to  block SCSI write command within USB packets
 */
#include <linux/module.h>
#include <linux/usbfilter.h>
#include <scsi/scsi.h>

#define LUM_NAME        "block_scsi_write"
#define LUM_SCSI_CMD_IDX    15

static struct usbfilter_lum lbsw;
static int lum_registered;

/*
 * Define the filter function
 * Return 1 if this is the target packet
 * Otherwise 0
 */
int lbsw_filter_urb(struct urb *urb)
{
    char opcode;

    /* Has to be an OUT packet */
    if (usb_pipein(urb->pipe))
        return 0;

    /* Make sure the packet is large enough */
    if (urb->transfer_buffer_length <= LUM_SCSI_CMD_IDX)
        return 0;

    /* Make sure the packet is not empty */
    if (!urb->transfer_buffer)
        return 0;

    /* Get the SCSI cmd opcode */
    opcode = ((char *)urb->transfer_buffer)[LUM_SCSI_CMD_IDX];

    /* Current only handle WRITE_10 for Kingston */
    switch (opcode) {
    case WRITE_10:
        return 1;
    default:
        break;
    }

    return 0;
}
```

```c
static int __init lbsw_init(void)
{
    pr_info("lbsw: Entering: %s\n", __func__);
    snprintf(lbsw.name, USBFILTER_LUM_NAME_LEN, "%s", LUM_NAME);
    lbsw.lum_filter_urb = lbsw_filter_urb;

    /* Register this lum */
    if (usbfilter_register_lum(&lbsw))
        pr_err("lbsw: registering lum failed\n");
    else
        lum_registered = 1;

    return 0;
}

static void __exit lbsw_exit(void)
{
    pr_info("exiting lbsw module\n");
    if (lum_registered)
        usbfilter_deregister_lum(&lbsw);
}

module_init(lbsw_init);
module_exit(lbsw_exit);

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("lbsw module");
MODULE_AUTHOR("dtrump");
```

Just read, seriously

For Kingston USB flash drive:

```
usbtables -a nodataexfil -v manufacturer=Kingston
      -l name=block_scsi_write -t drop
usbtables -a nodataexfil2 -o uid=1001
      -v manufacturer=Kingston
      -l name=block_scsi_write -t drop
usbtables -a nodataexfil3 -o comm=vim
      -v manufacturer=Kingston
      -l name=block_scsi_write -t drop
```

What is wrong with USB

- Unlimited capabilities
- No authentication
- BadUSB attacks