Evaluating Security Aspects of the Universal Serial Bus

Moritz Jodeit
University of Hamburg

2009/01/13

Agenda

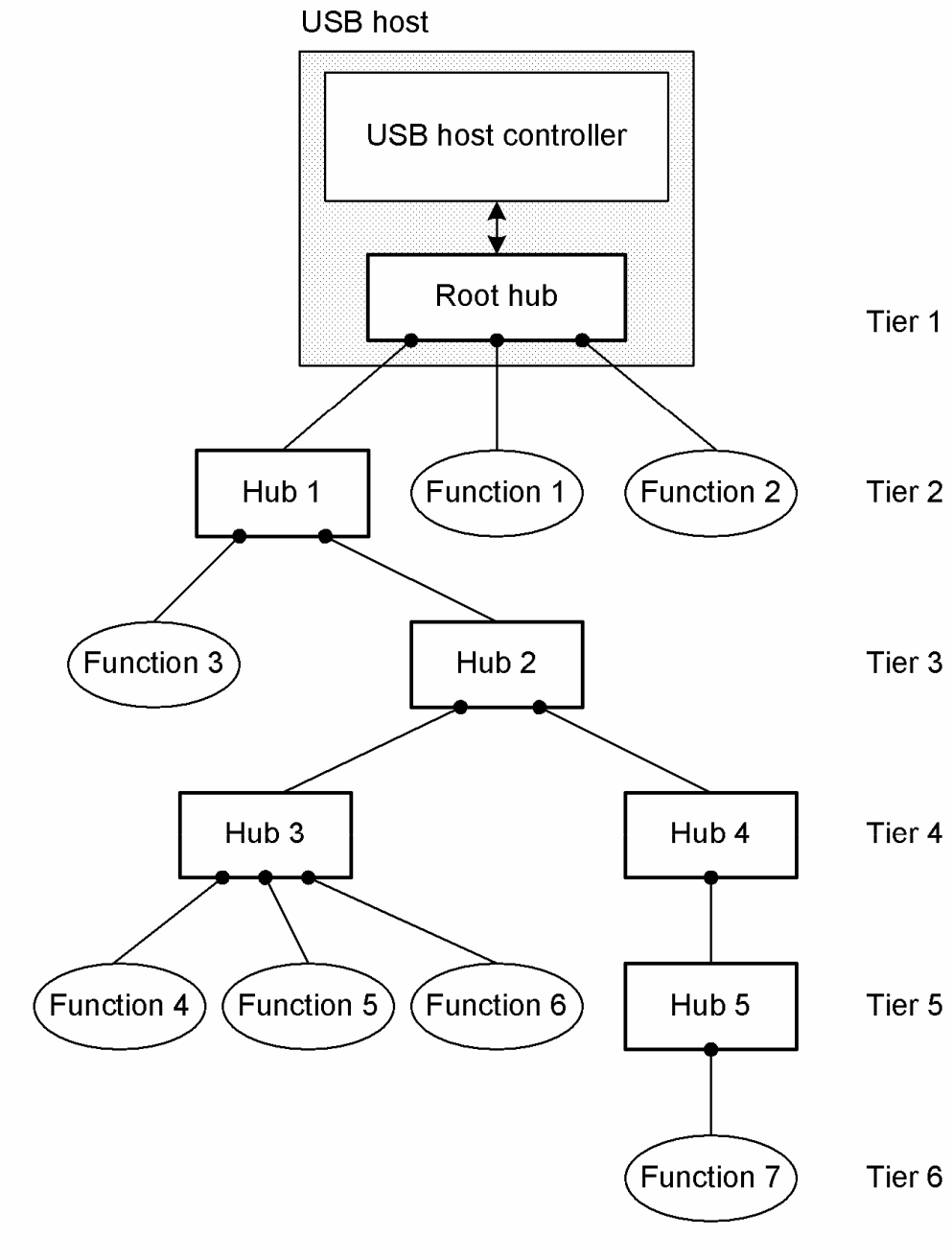
- Introduction to USB
- Attack scenarios
- Attacks
  - Logical attacks
  - Application-level attacks
  - Stack and device driver attacks
  - Kernel subsystem attacks
- Implementation of a USB fuzzer
- Results
- Future work

USB Intro

- Serial cable bus
- Wide range of peripherals
- Hot swapping
- Power can be provided by the host
- Standardized by the USB-IF
  - HP, Intel, LSI, NEC, Microsoft
- Latest spec is USB 3.0
  - Research based on USB 2.0

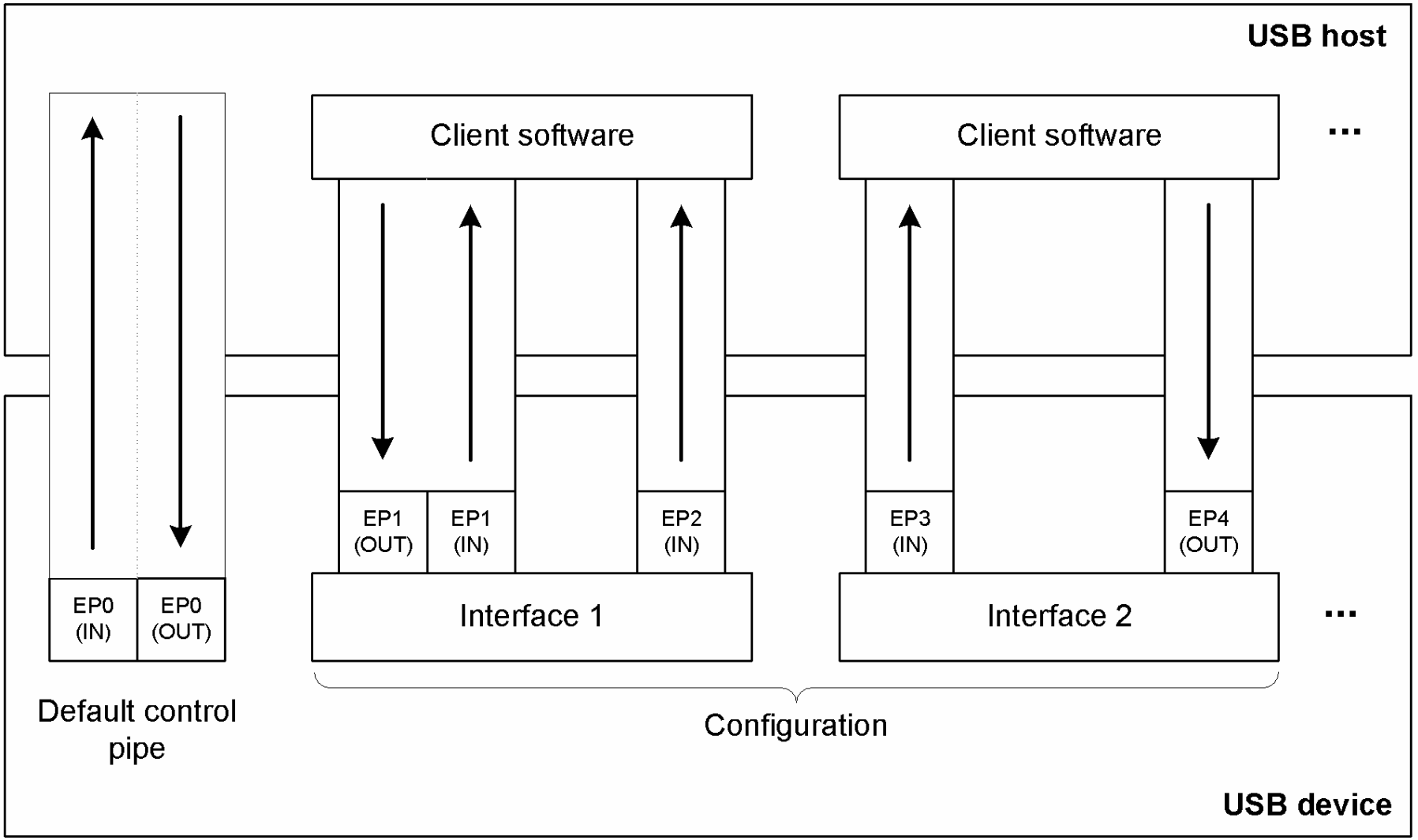
USB Architecture

- USB devices
  - Hubs
  - Functions
- USB host
  - Controls the communication on the bus
    - Detection of device attachment/removal
    - Configuration of new devices
  - Only a single host per bus
  - All transfers initiated by the host
- USB interconnect
  - Tiered star topology


USB host
USB host controller
Root hub
Tier 1
Hub 1
Function 1
Function 2
Tier 2
Function 3
Hub 2
Tier 3
Hub 3
Hub 4
Tier 4
Function 4
Function 5
Function 6
Hub 5
Tier 5
Function 7
Tier 6

USB Communication Flow

- Endpoints
  - EP number
  - EP direction (IN/OUT)
- Pipes
  - Unidirectional
  - Bidirectional
    - Two EP‘s with same EP number
  - Default control pipe (EP0)
- Interfaces
  - Multiple pipes
  - More than one can be provided at the same time („composite device“)
- Configurations
  - Multiple interfaces
  - Only a single configuration can be active


USB host
Client software
Client software
...
EP1
(OUT)
EP1
(IN)
EP2
(IN)
EP3
(IN)
EP4
(OUT)
EP0
(IN)
EP0
(OUT)
Interface 1
Interface 2
...
Default control
pipe
Configuration
USB device

USB Bus Protocol

- Token-based packet protocol
- Employs polling mechanism
- Data transferred using transactions
- Transaction consists of multiple packets
  - Token packets
    - Initiation of a transaction by the host
    - Contain device address + endpoint number
  - Data packets
    - Actual data transmission
  - Handshake packets
    - Acknowledgement of transaction

USB Bus Protocol

- Packet flow of a transaction
  - Host sends token packets on a scheduled interval
  - Every connected device reads initial token packet
  - Specified EP on device receives packet
  - Depending on token packet type, data packet is send
    - IN token packets request the device to send a data packet to the host
    - OUT token packet request the device to receive the next data packet sent by the host
  - Receiver of data packet acknowledges reception with a handshake packet

USB Enumeration

- Starts when device is attached at a hub
- Hub notifies host about event
- Host requests from hub to enable device
- Device has default address 0
  - Provides default control pipe (EP0)
- Host configures attached device
  - Standard USB requests to EP0
- Host assigns unique device addres
  - Device now only answers to new address

USB Enumeration

- Host requests various descriptors
- Descriptors
  - Data structures provided by device
  - Describe all attributes of device
  - USB spec defines some standard descriptors
  - Vendor specific descriptors are possible

USB Enumeration

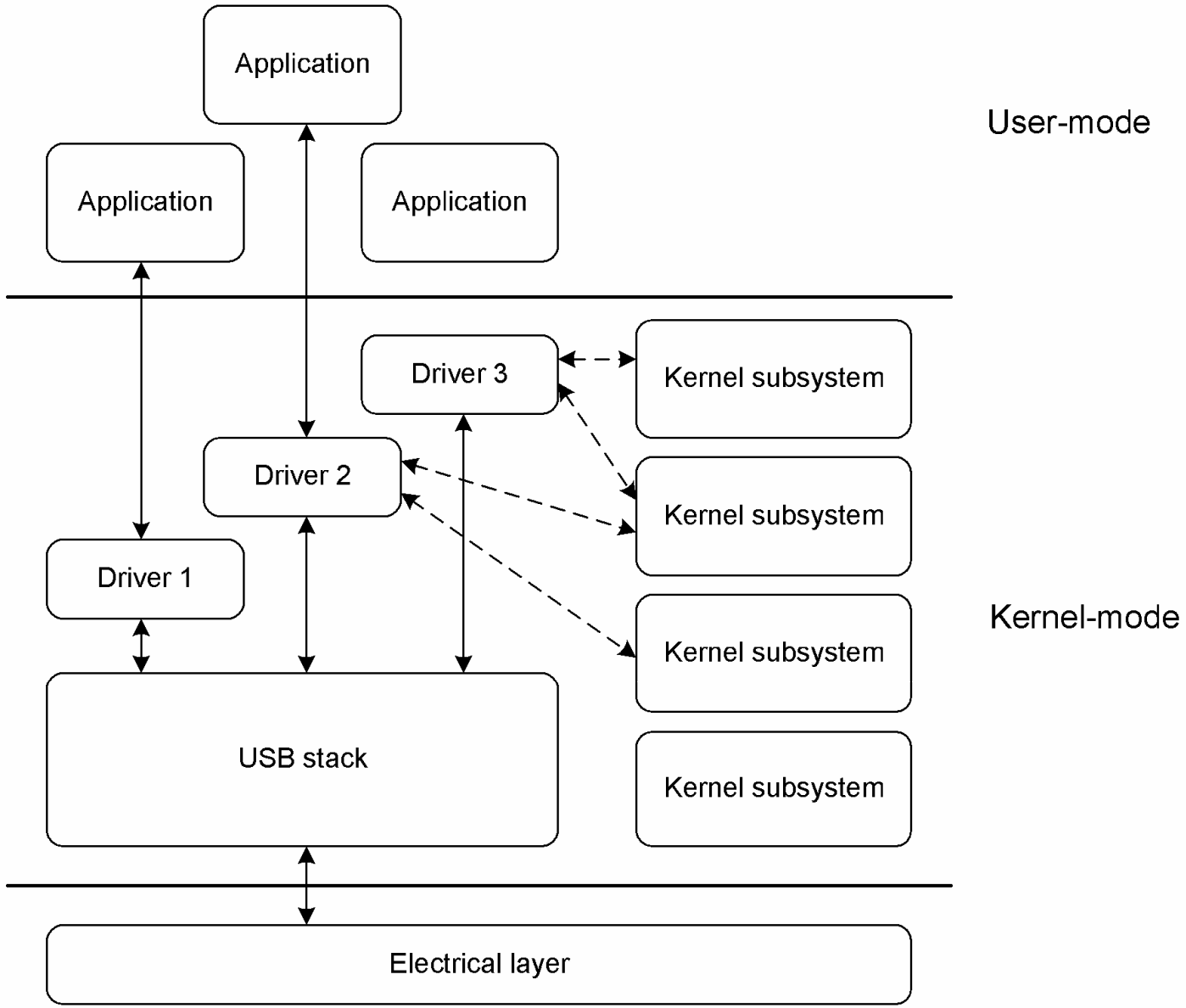
- Host tries to find matching device driver
  - Based on received descriptors
- Loaded device driver selects configuration
- Device provides all interfaces/EP‘s

Attack Scenarios

- Hardware security tokens
  - Lot‘s of them based on USB
  - Implemented for higher needs of security
- Kiosk print systems
  - Unattended and not actively monitored
- Employees (insiders)
  - Good knowledge of internal processes

Attack Scenarios

- Bribery
  - Ask your friendly janitor ;)
- Trick people with legitimate access
  - A few well-placed USB devices in front of a corporate building
  - Send shiny new devices by mail as a present
    - Who would reject a brand new iPhone?
  - Electronic voting systems using digital pens


Application
Application
Application
User-mode
Driver 3
Kernel subsystem
Driver 2
Kernel subsystem
Driver 1
Kernel subsystem
Kernel-mode
USB stack
Kernel subsystem
Electrical layer

Attacks

- Logic attacks
- Application-level attacks
- USB stack and device driver attacks
- Kernel subsystem attacks

Logic Attacks

- Malicious HID devices
  - HID class driver provided by any OS
  - HID device can act as a mouse/keyboard
  - Can perform anything a user could do
  - Could even be remotely controlled
    - Certified Wireless USB (CWUSB)

Logic Attacks

- Windows AutoRun
  - Executable can be launched
  - Disabled for USB storage devices on XP/Vista
  - CD-ROM drives can still make use of it on XP
  - We can easily build a USB CD-ROM device
    - U3 flash drives provide mass storage + CD-ROM
    - ISO filesystem can be modified

Logic Attacks

- USB packet sniffer
  - Token-based packet protocol
  - Every connected device can see all packets send by the host
    - Token packets
    - Data OUT packets
  - Device could capture ALL received packets
    - Files transfered from host to flash drive
    - Documents printed on USB printer

Logic Attacks

- USB packet sniffer
  - USB is also used to connect internal devices
    - IEEE 802.11
    - Bluetooth
    - …
  - Device can sniff outgoing wireless traffic
    - Encryption on the wireless link is bypassed
  - This presumes, that the same bus is used

Application-Level Attacks

- Apple iTunes and iPods
  - USB connector since 3rd generation
  - Attach as mass storage devices
  - Filesystem contains iTunes control data
  - iTunesHelper applications detects iPods
    - Launches iTunes when iPod is connected
  - iTunes parsers could be attacked

Application-Level Attacks

- OS X Quick Look
  - Used to display thumbnails / preview images
    - Used by Finder / Spotlight
  - Applications request previews of specific files
  - Quick Look daemon
    - Running in background
    - Receives requests from user applications
    - Supports various file formats
  - Lot‘s of parsers with lot‘s of code
    - Fuzzing already showed some promising results ;)
  - Opening mounted USB flash drive using the Finder

Stack and Device Driver Attacks

- USB stack
  - Handles low-level protocol details
  - Loads matching device drivers
  - Hardened systems with minimal set of device drivers can be attacked
- USB device drivers
  - Significant number of different drivers
  - Lot‘s of drivers developed by 3rd parties
    - Varying code quality can be expected
  - Class drivers available on nearly every system

Kernel Subsystem Attacks

- USB = Universal Serial Bus
- Large number of different devices
- Device drivers make use of other kernel components
  - Disk subsystem
  - Network subsystem
  - Audio/video subsystem
  - Various protocol stacks!
    - IrDA, 802.11, Bluetooth, …
- Does all this code handle malformed data?
  - Answer is „no“, as we‘ll see later ;)

Implementation of a USB Fuzzer

- Prerequisites
  1. Fuzzing should be automatic
  2. Should be able to send malformed data
  3. Implemented in software
  4. Should be able to emulate various devices

Implementation of a USB Fuzzer

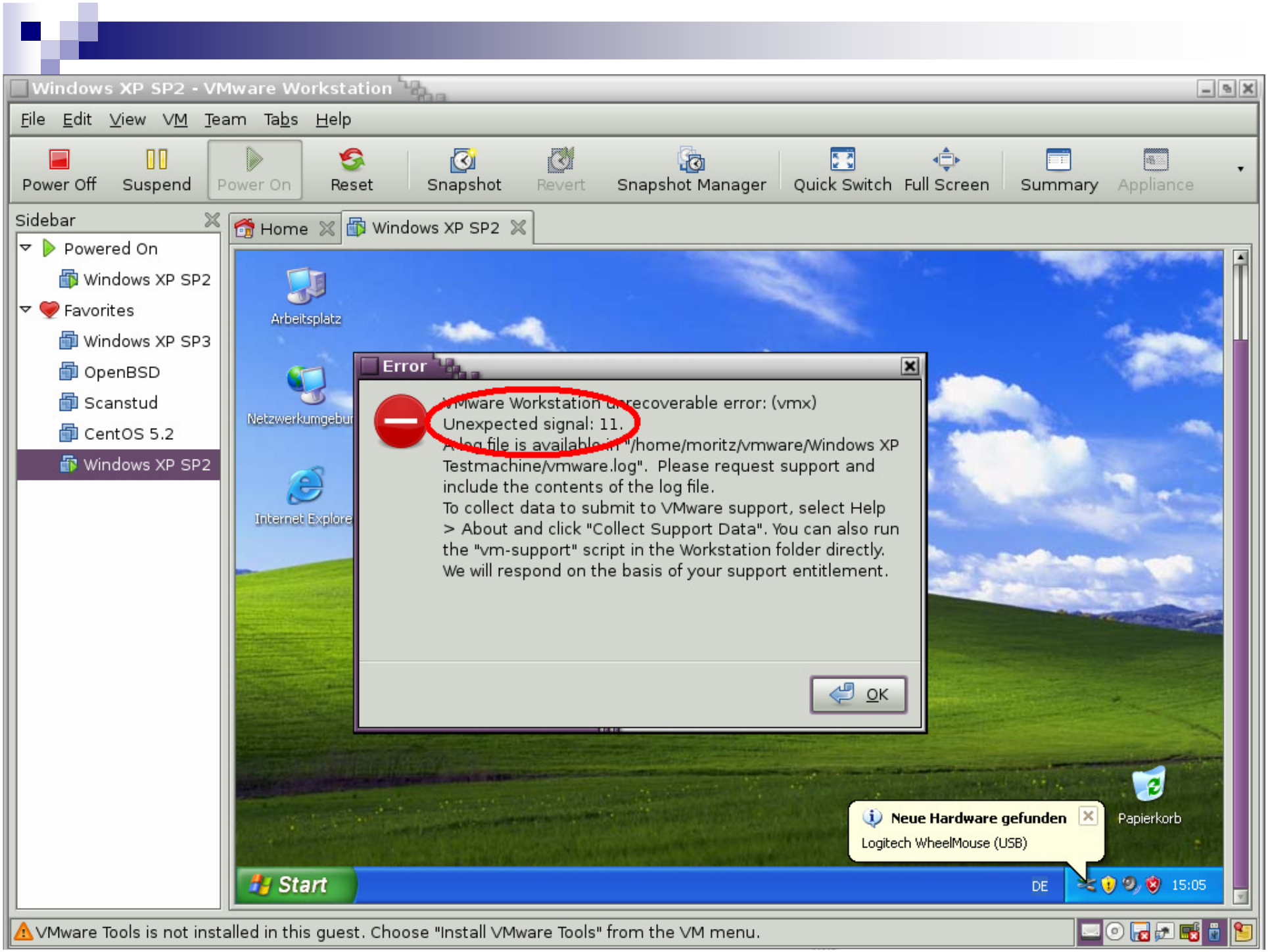
- Linux-USB Gadget API Framework
  - Allows embedded Linux systems to act in the USB device role
  - Peripheral controller + gadget drivers
  - Multiple USB peripheral controller drivers
    - Netchip 228x, AMD5536 UDC, Renesas M665992,…
    - DummyHCD

Implementation of a USB Fuzzer

- Linux-USB Gadget API Framework
  - Some gadget drivers included
    - USB ethernet device
    - Mass storage device
    - Serial device
    - MIDI device
  - GadgetFS module for user mode drivers

Implementation of a USB Fuzzer

- First approach
  - Emulate USB device in software
    - DummyHCD
  - Use virtualization solution to fuzz guest OS
  - Problems encountered
    - Emulated device is handled by host OS and virtualization first!
    - These additional layers can prevent attachment


Windows XP SP2 - VMware Workstation
File Edit View VM Team Tabs Help
Power Off
Suspend
Power On
Reset
Snapshot
Revert
Snapshot Manager
Quick Switch
Full Screen
Summary
Appliance
Sidebar
Powered On
Windows XP SP2
Favorites
Windows XP SP3
OpenBSD
Scanstud
CentOS 5.2
Windows XP SP2
Home
Windows XP SP2
Arbeitsplatz
Netzwerkumgebur
Internet Explore
Error
VMware Workstation unrecoverable error: (vmx)
Unexpected signal: 11.
A log file is available in "/home/moritz/vmware/Windows XP Testmachine/vmware.log". Please request support and include the contents of the log file.
To collect data to submit to VMware support, select Help > About and click "Collect Support Data". You can also run the "vm-support" script in the Workstation folder directly.
We will respond on the basis of your support entitlement.
OK
Neue Hardware gefunden
Logitech WheelMouse (USB)
Papierkorb
Start
DE
15:05
VMware Tools is not installed in this guest. Choose "Install VMware Tools" from the VM menu.

Implementation of a USB Fuzzer

- Second approach
  - Emulate USB device in software
  - But use a hardware solution to connect to the host to be tested
  - Netchip NET2280 peripheral controller
    - PCI evaluation board

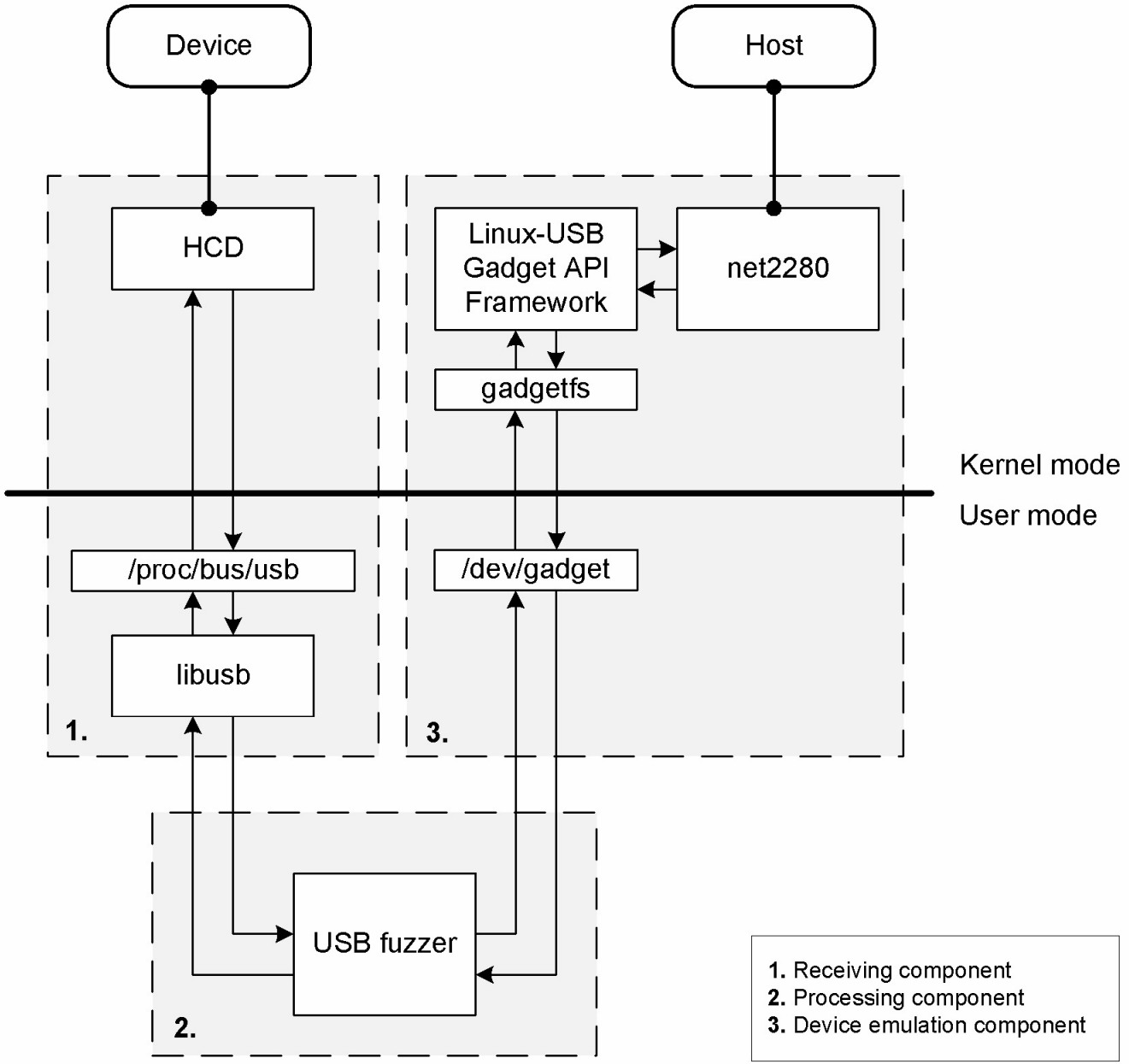
0x1e

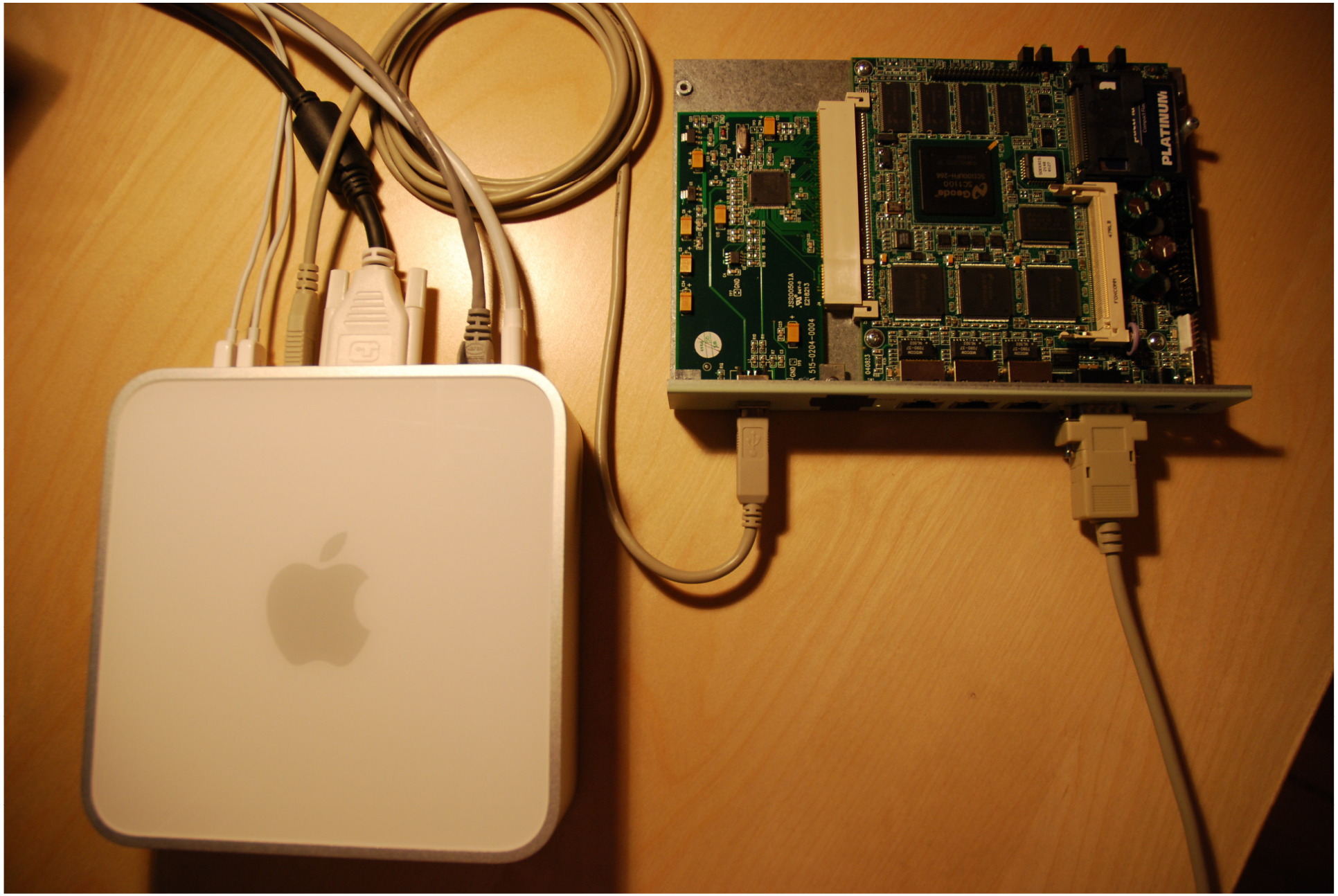
Implementation of a USB Fuzzer

- PoC fuzzer implemented in the peripheral controller driver of the NET2280
  - Every loaded gadget driver is fuzzed
  - Restriction to available gadget drivers
  - Time-consuming to add new gadget drivers
- Most bugs found using this PoC
  - But we wanted a more universal solution…

Implementation of a USB Fuzzer

- Final design
  - Man-in-the-middle mutation-based fuzzer
  - Still in development
- Components
  - Receiving component
    - libusb
  - Processing component
    - Fuzzing + replaying
  - Device emulation component
    - Gadget framework


Device
Host
HCD
Linux-USB
Gadget API
Framework
net2280
gadgetfs
Kernel mode
User mode
/proc/bus/usb
/dev/gadget
libusb
1.
3.
USB fuzzer
2.
1. Receiving component
2. Processing component
3. Device emulation component


PLATINUM

Results

- Fuzzing USB mass storage driver
  - Repeatedly attached mass storage device
  - Randomly fuzzed all IN transactions on all pipes
    - Randomly replaced bytes with random ones
    - Most significant bit set more often
  - Crashes where reproduced using the packet number
    - Led to some non-reproducible crashes
  - Tested: Windows XP/Vista, OS X, Linux, OpenBSD

Results

- Windows XP (SP2)
  - Multiple crashes encountered
    - Not all of them could be reproduced
  - Double-free of kernel pool memory in usbstor.sys
  - Kernel pool memory corruption in disk.sys while reading the partition table
  - Fuzzing on EP0 disabled the whole USB functionality until a reboot

Results

- Mac OS X
  - Complete lockup of the system
    - When time between re-attachments was too small
    - No kernel panic screen shown
    - Reset of the system restored behaviour
  - One other kernel panic was produced
    - Unfortunately was not reproducible
- OpenBSD
  - Kernel panic in SCSI subsystem

Results

- Windows Vista & Linux
  - No crashes encountered
  - We did only scratch the surface
    - Only a single class driver tested
    - Dumb random-based fuzzing
- Accidental finding…

Results


Device Monitoring Studio main executable
Device Monitoring Studio main executable hat ein Problem festgestellt und muss beendet werden.
Falls Sie Ihre Arbeit noch nicht gespeichert hatten, können Daten möglicherweise verloren gegangen sein.
Dieses Problem bitte auch an Microsoft berichten.
Ein Problembericht, den Sie uns senden können, wurde erstellt. Wir werden diesen Bericht vertraulich und anonym bearbeiten.
Um zu sehen, welche Daten Ihr Bericht enthält, klicken Sie hier.
Debug
Problembericht senden
Nicht senden

Future Work

- Method used to reproduce crashes
  - New implementation allows replaying
- Finish the MITM approach to test 3rd party drivers
  - Re-implement device emulation component?
- PoC of the USB sniffing attack
  - Only theoretical for now
- Certified Wireless USB (CWUSB)
  - Authentication and encryption

Questions?