Exploiting Smart-Phone USB Connectivity For Fun And Profit

Angelos Stavrou & Zhaohui Wang
Department of Computer Science
George Mason University

Talk Outline

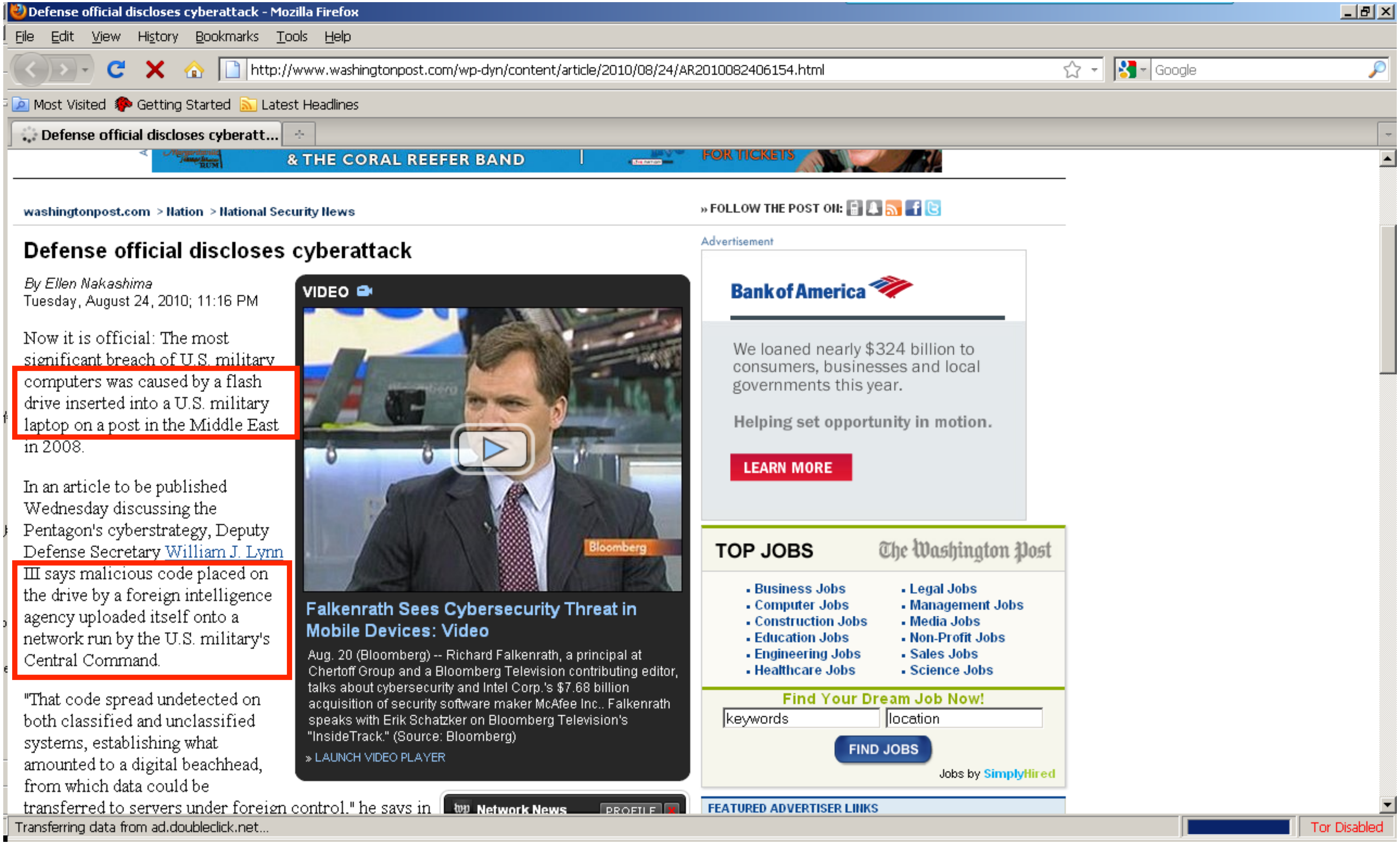
- Background – Why USB attacks? What's new here?
  - New attack vectors, different from simple USB storage
- Phone-to-Computer Attack
- Computer-to-Phone Attack
- Phone-to-Phone Attack
- Demo & Discussion Points
- Defenses & Future Work

USB is Pervasive in Gadgets

- All Smart-Phone devices use USB
  - Google Android Devices (HTC, Motorola, …)
  - Apple iPhone
  - Blackberry
  - Others

- Multi-purpose Usage
  - Charging the Device Battery
  - Data & Media Transfer
  - Control external Devices (new capability)

USB-borne Threats only focused on Auto-Mounting



Defense official discloses cyberattack - Mozilla Firefox
File
Edit
View
History
Bookmarks
Tools
Help
http://www.washingtonpost.com/wp-dyn/content/article/2010/08/24/AR2010082406154.html
Google
Most Visited
Getting Started
Latest Headlines
Defense official discloses cyberatt...

washingtonpost.com > Nation > National Security News

» FOLLOW THE POST ON:

Defense official discloses cyberattack

By Ellen Nakashima
Tuesday, August 24, 2010; 11:16 PM

Now it is official: The most significant breach of U.S. military computers was caused by a flash drive inserted into a U.S. military laptop on a post in the Middle East in 2008.

In an article to be published Wednesday discussing the Pentagon's cyberstrategy, Deputy Defense Secretary William J. Lynn III says malicious code placed on the drive by a foreign intelligence agency uploaded itself onto a network run by the U.S. military's Central Command.

"That code spread undetected on both classified and unclassified systems, establishing what amounted to a digital beachhead, from which data could be transferred to servers under foreign control," he says in

VIDEO

Falkenrath Sees Cybersecurity Threat in Mobile Devices: Video

Aug. 20 (Bloomberg) -- Richard Falkenrath, a principal at Chertoff Group and a Bloomberg Television contributing editor, talks about cybersecurity and Intel Corp.'s $7.68 billion acquisition of security software maker McAfee Inc.. Falkenrath speaks with Erik Schatzker on Bloomberg Television's "InsideTrack." (Source: Bloomberg)

» LAUNCH VIDEO PLAYER

USB-borne Threats are much more complex…

- USB protocol can be (ab)used to connect ***any*** device to a computing platform ***without*** authentication
  - Desktops, Laptops, phones, kiosks, tables (ipad)
- USB Storage is just the tip of the iceberg and it is usually locked-down and scanned by anti-virus and other defenses
- USB Human Interface Devices (HIDs) are one class of devices that are ***much*** more appealing
  - Keyboard/Mouse/??? on your Android Phone
  - Other USB devices?

USB-borne Threats are much more complex…

Many other devices:

- Ethernet/Wireless Network Adapter
  - No password, man in the middle for your network traffic installed as the default “gateway”
- Printer
  - Capture all the documents printed
- Joystic(!)
- Biometric USB Reader
  - Brute force your way into a protected system(?)

Phone-to-Computer Attacks

- Program the Phone with USB Gadget API for Linux
- Pretend to be a USB Human Interface Driver,
  - Dell USB keyboard, VendorID=413C,ProductID=2105
  - Touchpad or Mouse
- Pre-programmed key code.
  - User-lever or System-level attacks
  - Anything you would imagine
- Transparent to Victim Machine
  - No Human Input or approval

HIDs are recognized automatically…

Phone-to-Computer Attacks (Cont)

- Traditional autorun attacks are easy but easily detectable

- Autorun and autoplay are default since Windows XP SP2
  - (MS KB967715) tries to address that

- Flash Autoplay Content exploitation by re-enumeration
  - Exploit different content (PDF, HTML, DOC, MP3)
  - ReMount/unmount MMC card controlled by device

- Exploit Autoplay feature of default Media Programs
  - Selectively prepare attack payload, i.e. Malicious mp3 files targeting MacOSX iTunes, pdf targeting unpatched Adobe Reader
  - Highly robust exploit, works for for a variety of programs

Computer-to-Phone Attacks

- Gaining Root Access to the Smart Phone Device
  - Official: simulate screen tap event to the oem unlock menu on selected devices
  - Universal: linux local root exploit (CVE-2009-1185, RLIMIT_NPROC exhaustion) send via USB

- Insert malicious payload
  - Kernel-level: disassemble boot partition
  - Replace kernel zimage with your own
  - Replace Applications

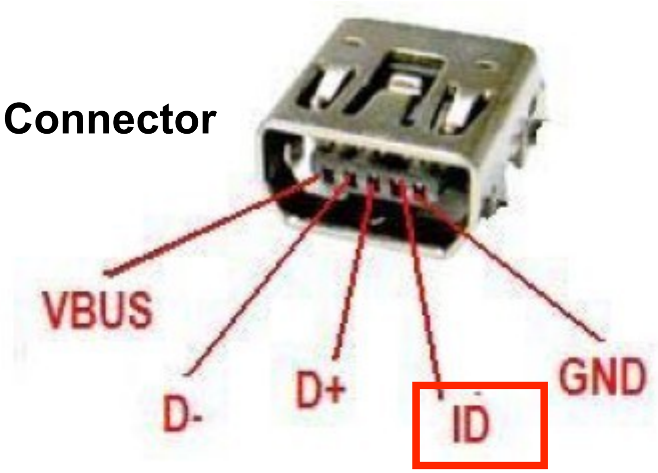
- Remove traces by un-rooting to avoid detection
  - We can quickly cleanup, not need for traces
  - Next reboot, not traces at all
  - Very very difficult to identify, it has to happen before next reboot

Computer-to-Phone Attacks (Cont.)

- Kernel manipulation
  - Rootkits
  - Traffic Redirection to a known proxy
  - Data Exfiltration
- Native ARM ELF binary
  - bypasses Android framework permissions and checks
- A complete phone provisioning process fully automated with evil payload
  - No application-level traces

Phone-to-Phone Attacks - OTG

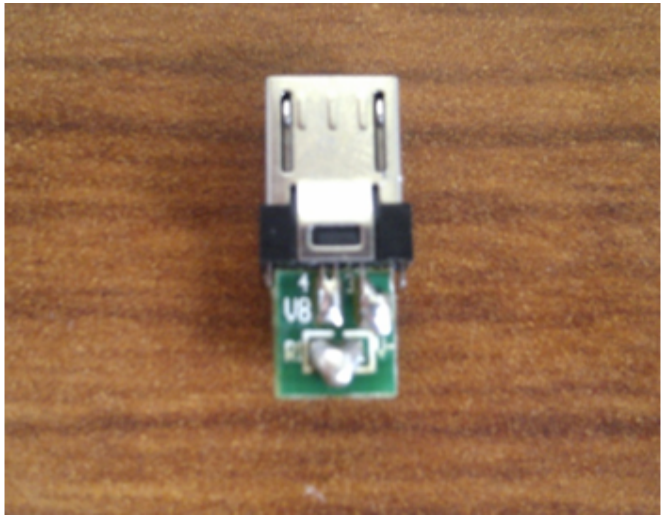
USB(Mini) OTG Connector

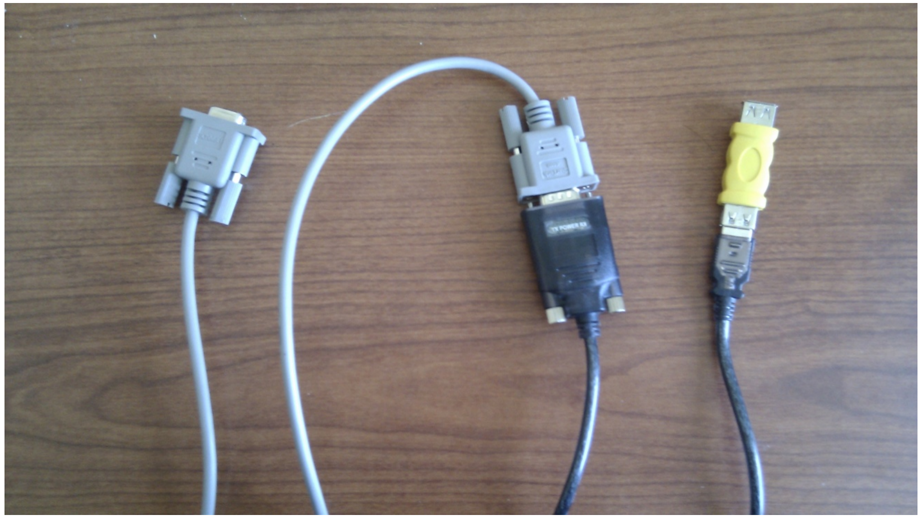


- USB OTG (On-the-Go) controller
  - Capability to switch the controller and become a host or a gadget
- Smart Phones are shipped with such OTG capable chipset
  - Qualcomm QSD8250, Texas Instruments OMAP 3430
- The 5th pin (ID) pin identifies the function of the controller host or gadget
  - floating ID denotes gadget, grounded ID denotes host

Smart Phone as a Host Controller

- Specially shorted USB mini-B dongle to signal the OTG controller behave as a host

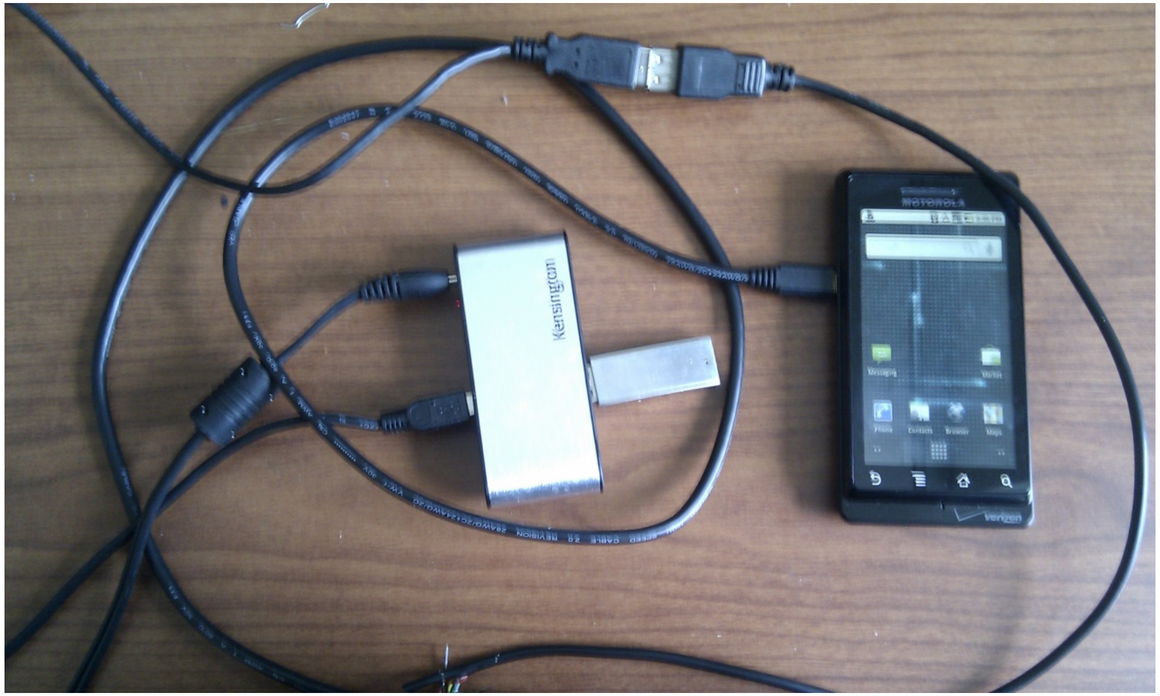




- USB transgender or USB micro-A to Standard-A Female cable.(out-of-box cable is micro-B to Standard-A Male)

Smart Phone as a Host Controller (Cont.)

- Power hub, for additional power supply



- Host side software stack, UHCI/EHCI HCD driver, device driver, userland programs

USB Hacking 101

Crucial Steps for USB Hacking:

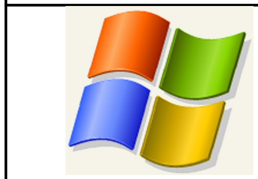
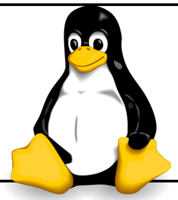
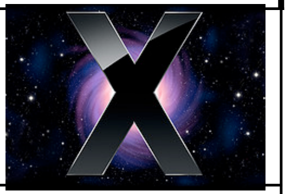
- Understand the USB Background (coming up)
  - Low-level “USB Hubs” VS device driver
- Good tools to help debugging (Demo
  - Some tools are helpful but have flaws as we will show
  - Combination of tools much better
- (Some) Hardware hacking
  - Craft cables to put the phone in “Master” mode
  - Use the phone to connect and hack Other Phones
- Patience!

USB Reconnaissance

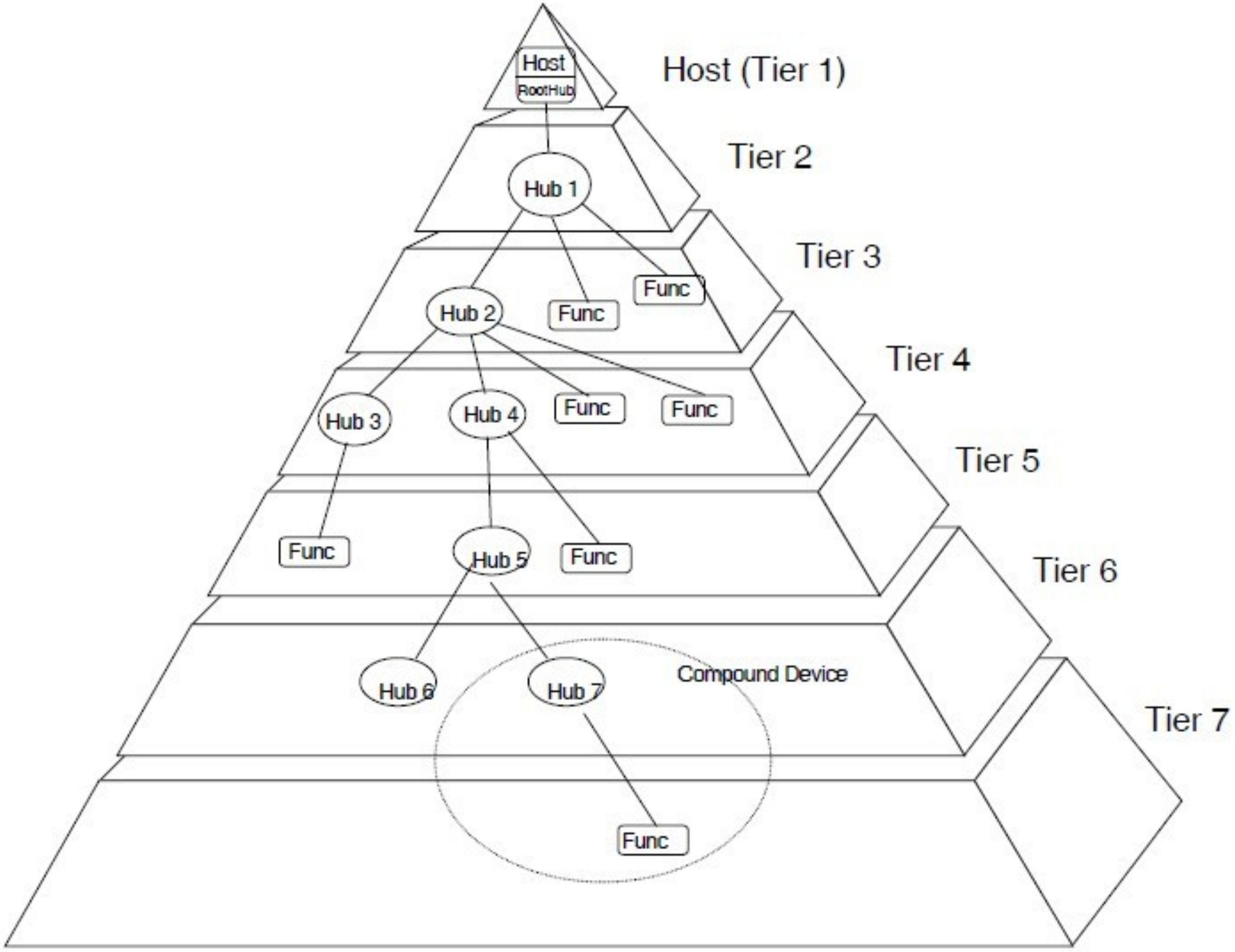
Operating System Fingerprinting using USB:

- Not all USB implementations are the same
  - Windows vs Linux vs Mac OSX
  - Flavors of Windows
- The protocol is the same but not the implementation
- USB devices in “slave”/ gadget mode can identify the OS upon connection
- Smart (i.e. programmable USB devices) can do so much more as we will see.

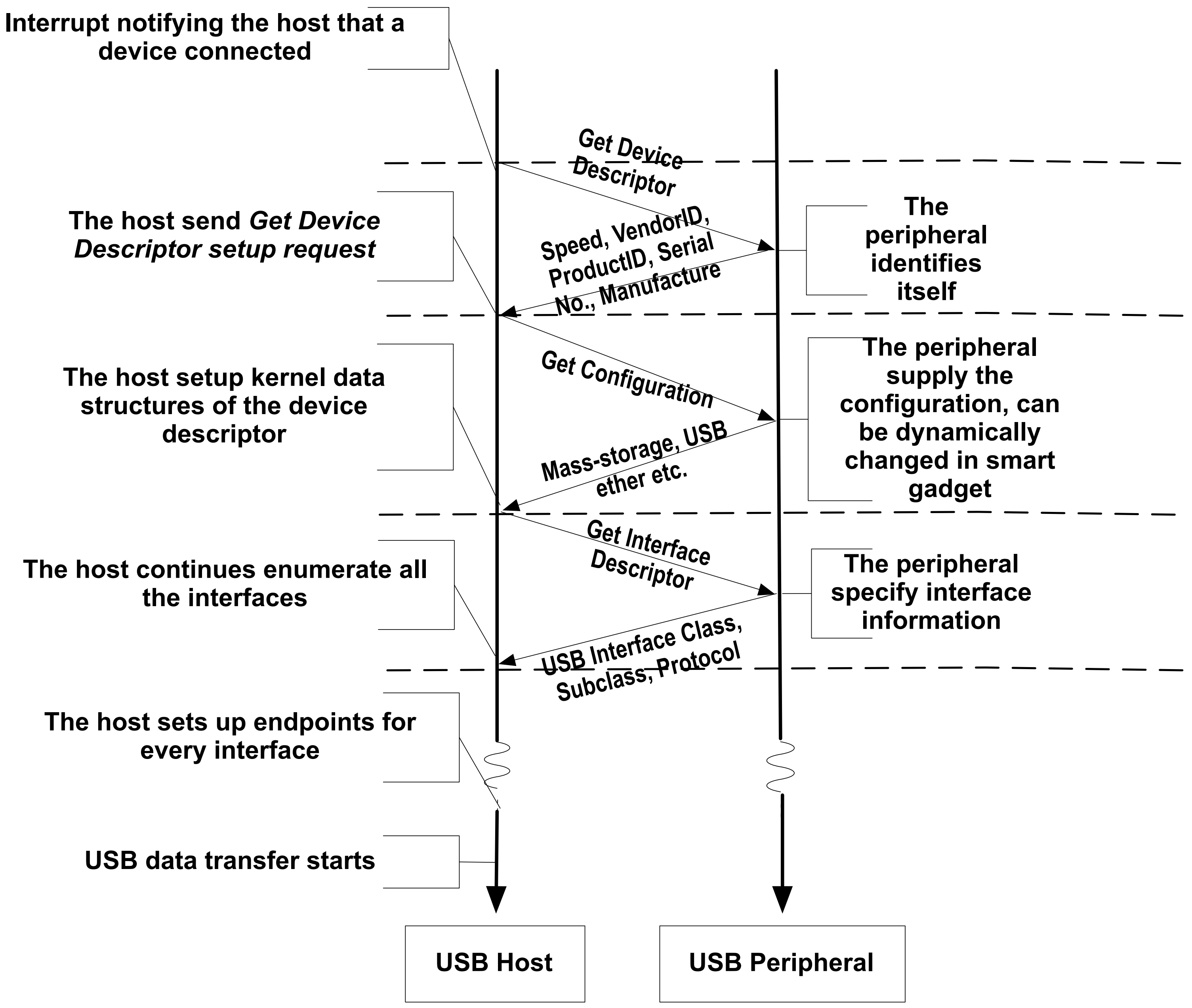
USB Reconnaissance

USB Gadget Observations	Operating System		
			
Full function probe	✓	✗	✗
Bare device w/o configuration retries	6	12	1
Device alive probe	✓	✗	✓
Single adb/umass interface bus reset	✓	✓	✗

USB Background: Hierarchical Topology

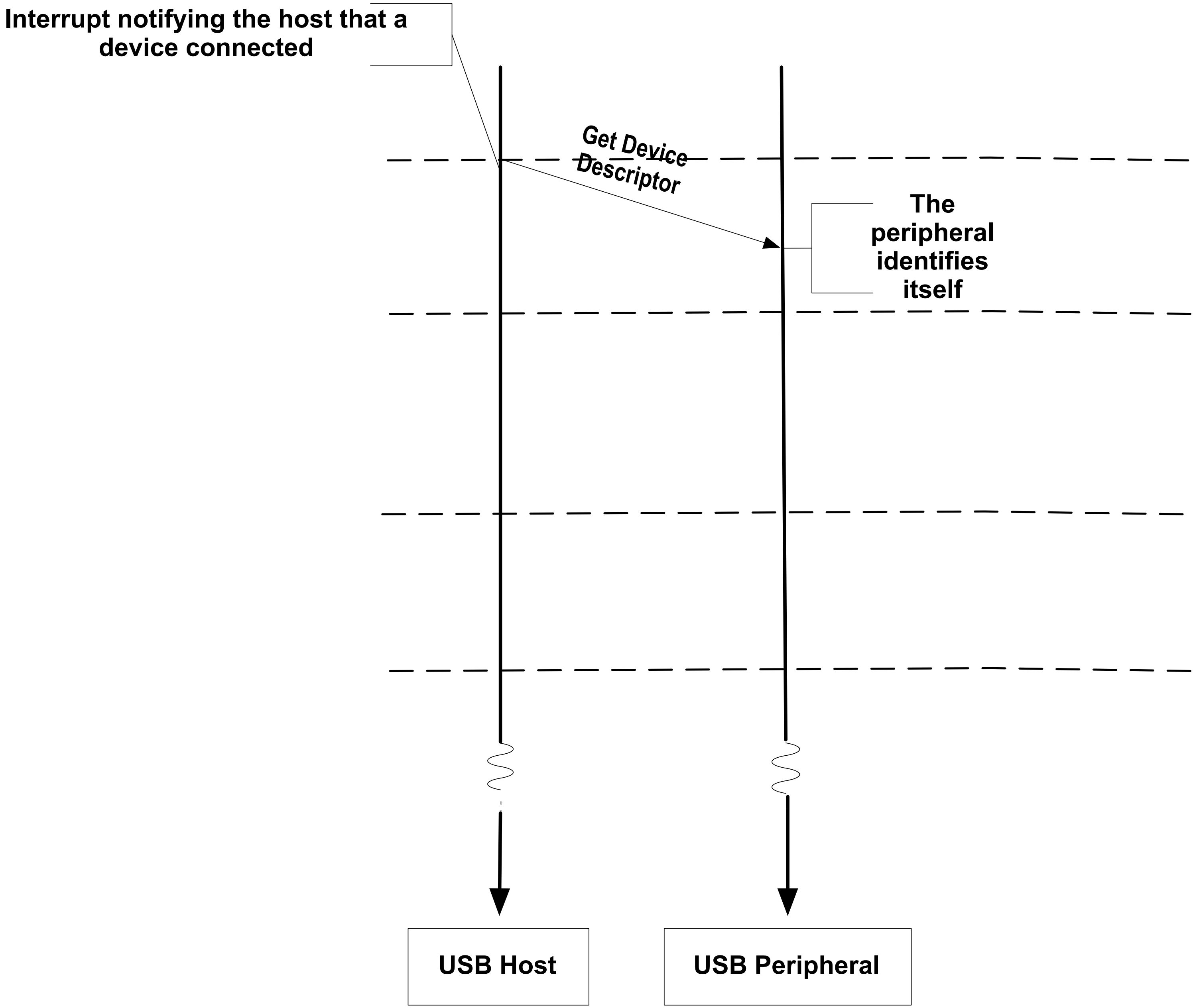


USB: Series of Events (Overview)


Interrupt notifying the host that a device connected
Get Device Descriptor
The host send *Get Device Descriptor setup request*
Speed, VendorID, ProductID, Serial No., Manufacture
The peripheral identifies itself
Get Configuration
The host setup kernel data structures of the device descriptor
Mass-storage, USB ether etc.
The peripheral supply the configuration, can be dynamically changed in smart gadget
Get Interface Descriptor
The host continues enumerate all the interfaces
USB Interface Class, Subclass, Protocol
The peripheral specify interface information
The host sets up endpoints for every interface
USB data transfer starts
USB Host
USB Peripheral

Standard USB Handshake

USB: Series of Events



Interrupt notifying the host that a device connected

Get Device Descriptor

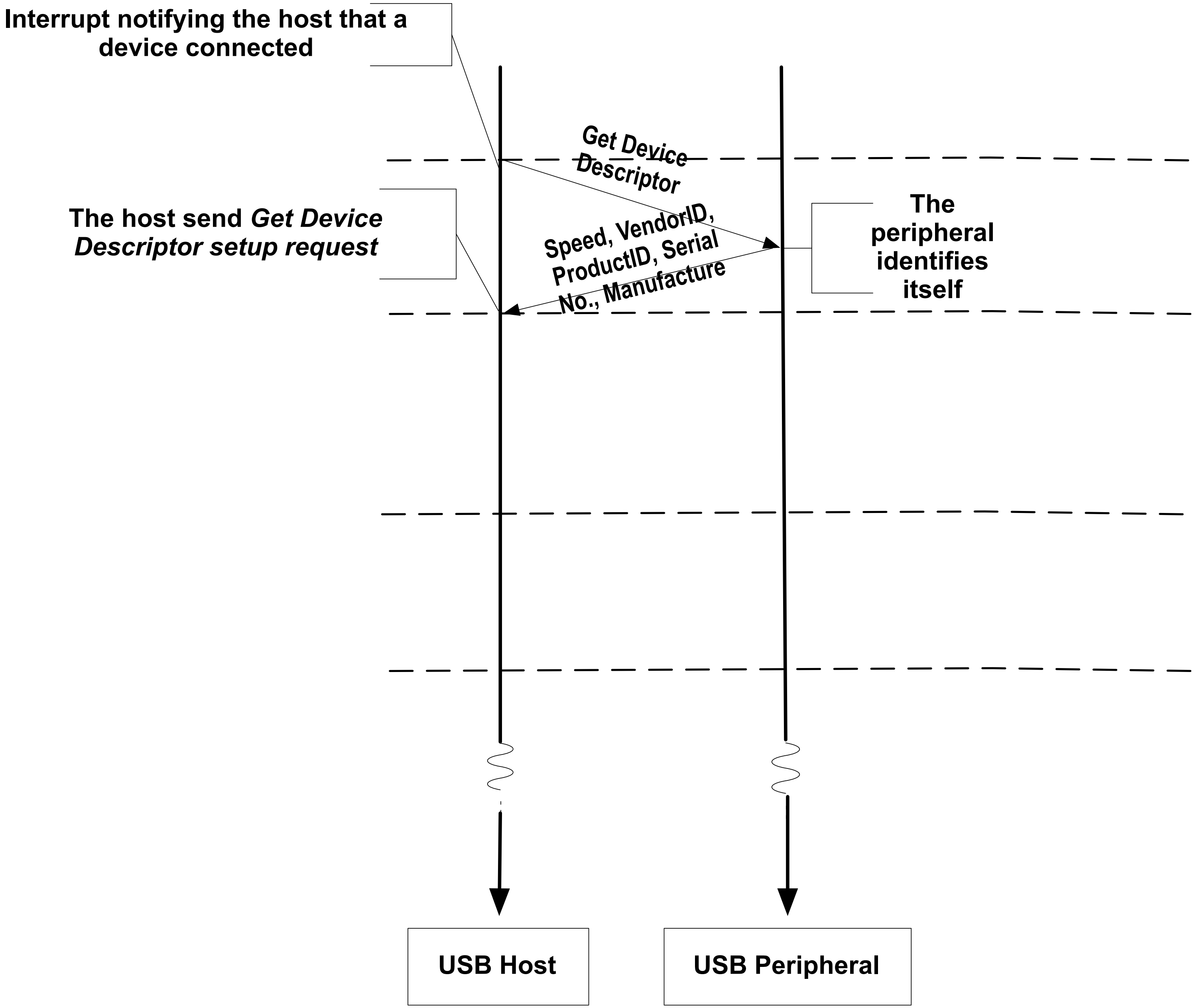
The peripheral identifies itself

USB Host

USB Peripheral

Standard USB Handshake

USB: Series of Events



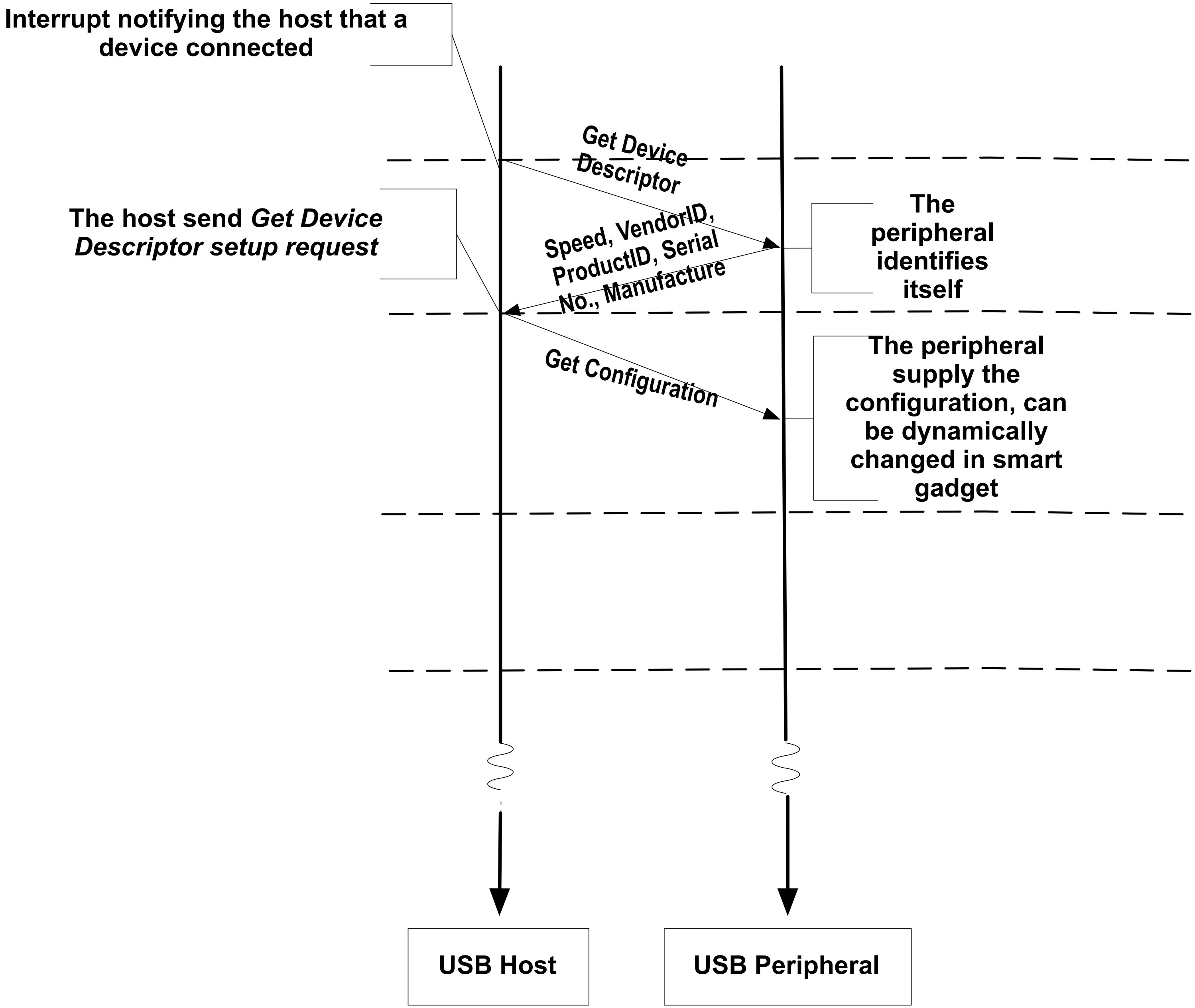
Interrupt notifying the host that a device connected

The host send *Get Device Descriptor setup request*

The peripheral identifies itself

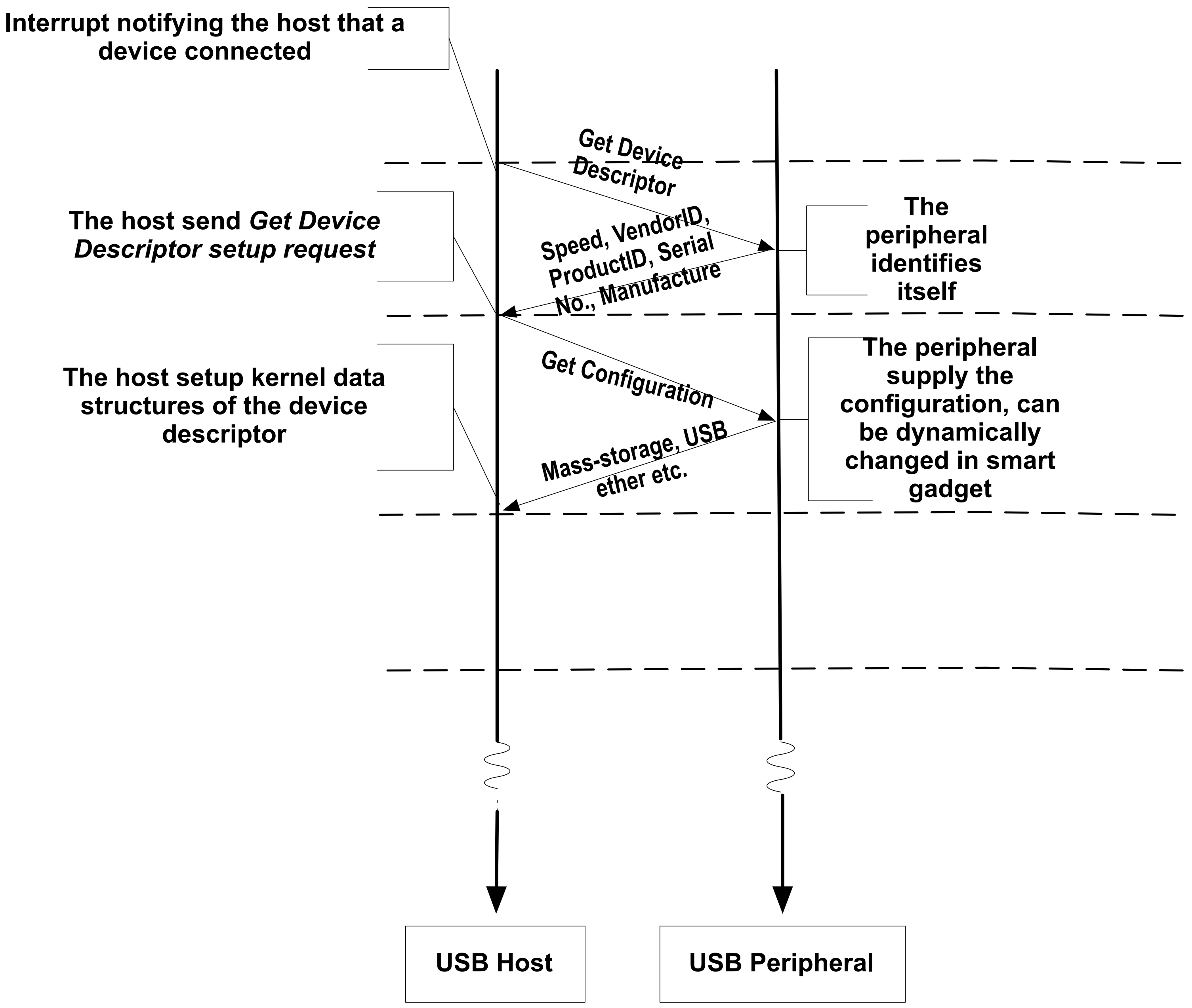
Standard USB Handshake

USB: Series of Events


Interrupt notifying the host that a device connected
Get Device Descriptor
The host send *Get Device Descriptor setup request*
Speed, VendorID, ProductID, Serial No., Manufacture
The peripheral identifies itself
Get Configuration
The peripheral supply the configuration, can be dynamically changed in smart gadget
USB Host
USB Peripheral

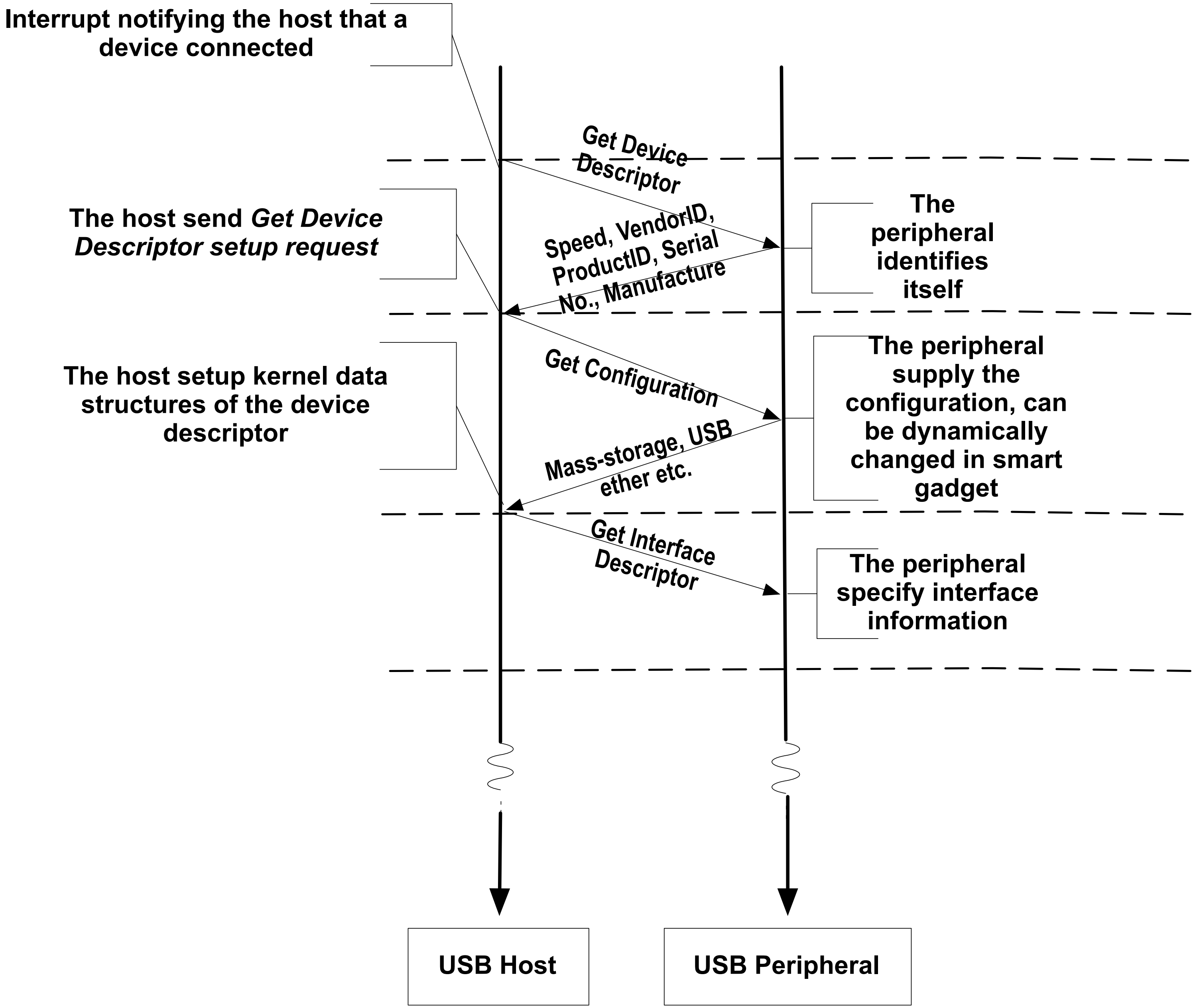
Standard USB Handshake

USB: Series of Events


Interrupt notifying the host that a device connected
Get Device Descriptor
The host send *Get Device Descriptor setup request*
Speed, VendorID, ProductID, Serial No., Manufacture
The peripheral identifies itself
Get Configuration
The host setup kernel data structures of the device descriptor
The peripheral supply the configuration, can be dynamically changed in smart gadget
Mass-storage, USB ether etc.
USB Host
USB Peripheral

Standard USB Handshake

USB: Series of Events



Interrupt notifying the host that a device connected

The host send *Get Device Descriptor setup request*

The host setup kernel data structures of the device descriptor

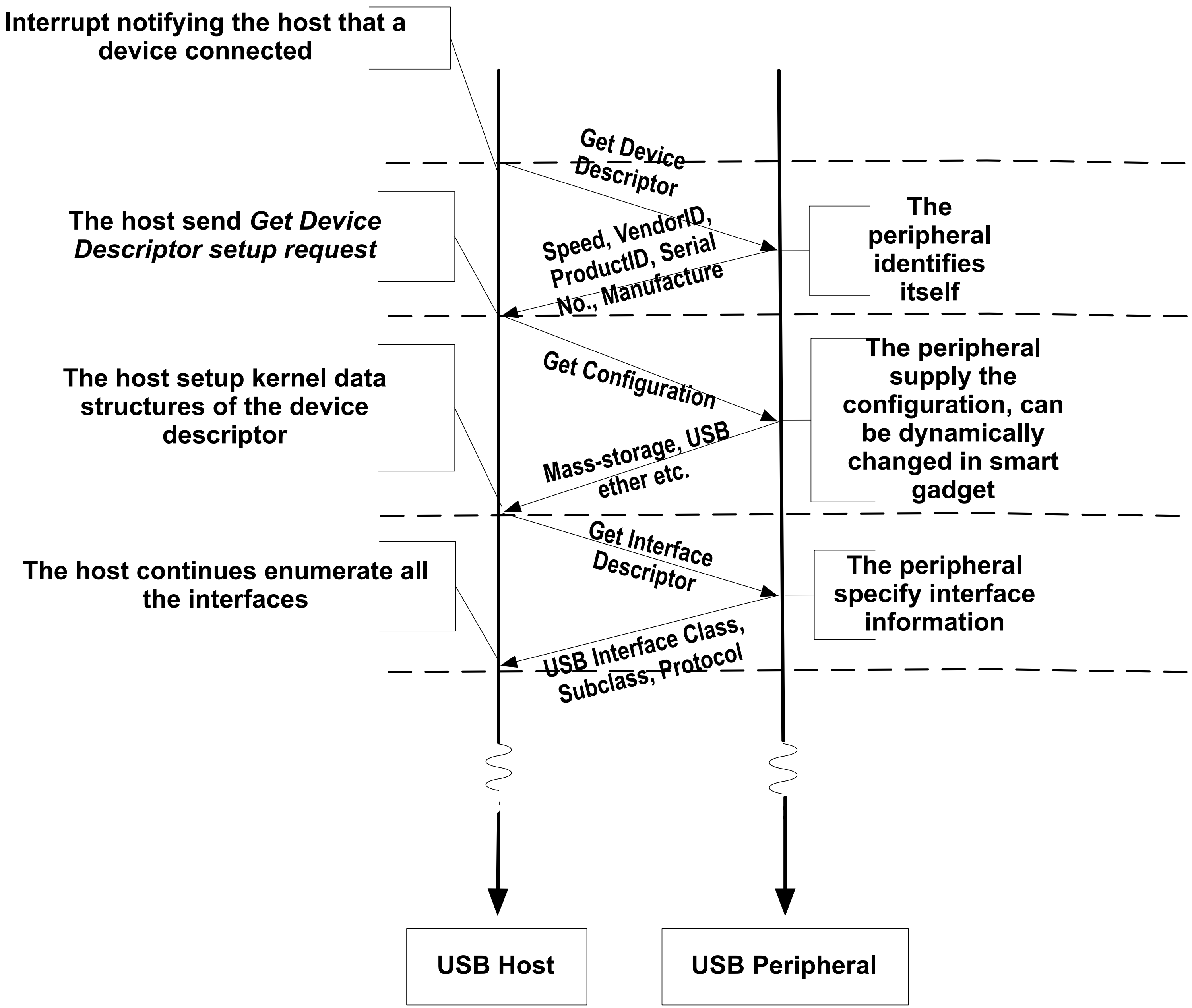
The peripheral identifies itself

The peripheral supply the configuration, can be dynamically changed in smart gadget

The peripheral specify interface information

Standard USB Handshake

USB: Series of Events



Interrupt notifying the host that a device connected

Get Device Descriptor

The host send *Get Device Descriptor setup request*

Speed, VendorID, ProductID, Serial No., Manufacture

The peripheral identifies itself

Get Configuration

The host setup kernel data structures of the device descriptor

Mass-storage, USB ether etc.

The peripheral supply the configuration, can be dynamically changed in smart gadget

Get Interface Descriptor

The host continues enumerate all the interfaces

USB Interface Class, Subclass, Protocol

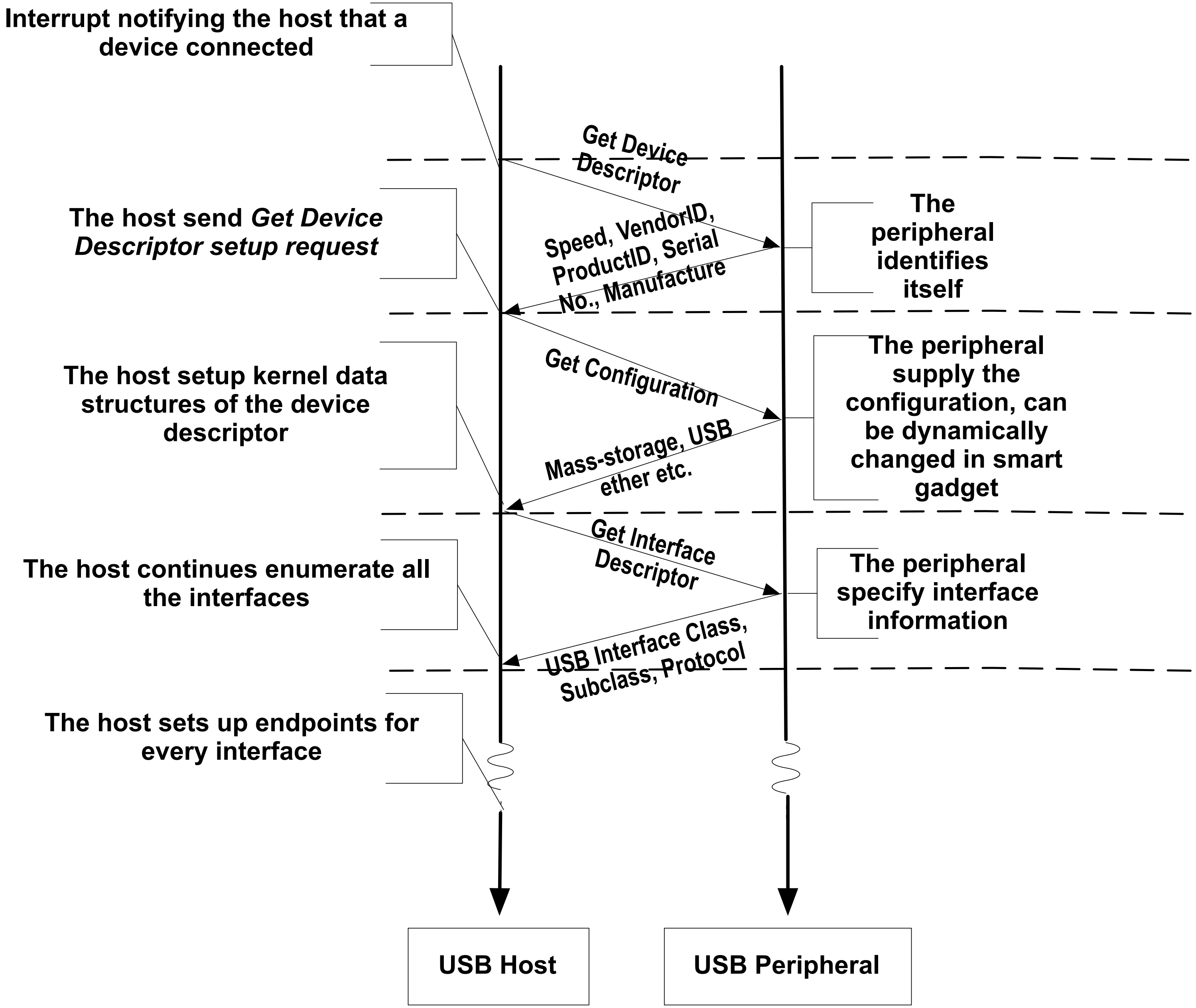
The peripheral specify interface information

USB Host

USB Peripheral

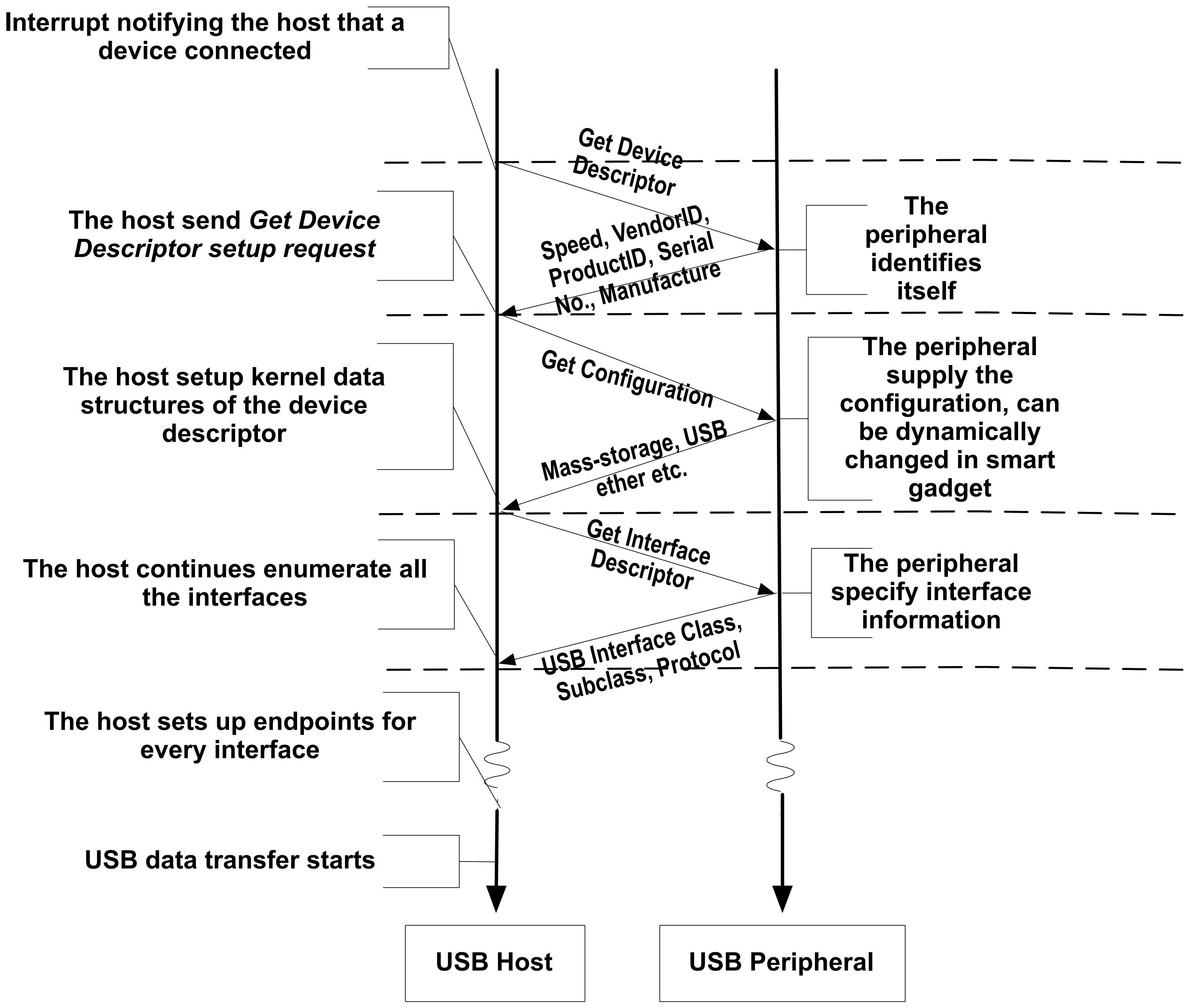
Standard USB Handshake

USB: Series of Events


Interrupt notifying the host that a device connected
Get Device Descriptor
The host send *Get Device Descriptor setup request*
Speed, VendorID, ProductID, Serial No., Manufacture
The peripheral identifies itself
Get Configuration
The host setup kernel data structures of the device descriptor
Mass-storage, USB ether etc.
The peripheral supply the configuration, can be dynamically changed in smart gadget
Get Interface Descriptor
The host continues enumerate all the interfaces
USB Interface Class, Subclass, Protocol
The peripheral specify interface information
The host sets up endpoints for every interface
USB Host
USB Peripheral

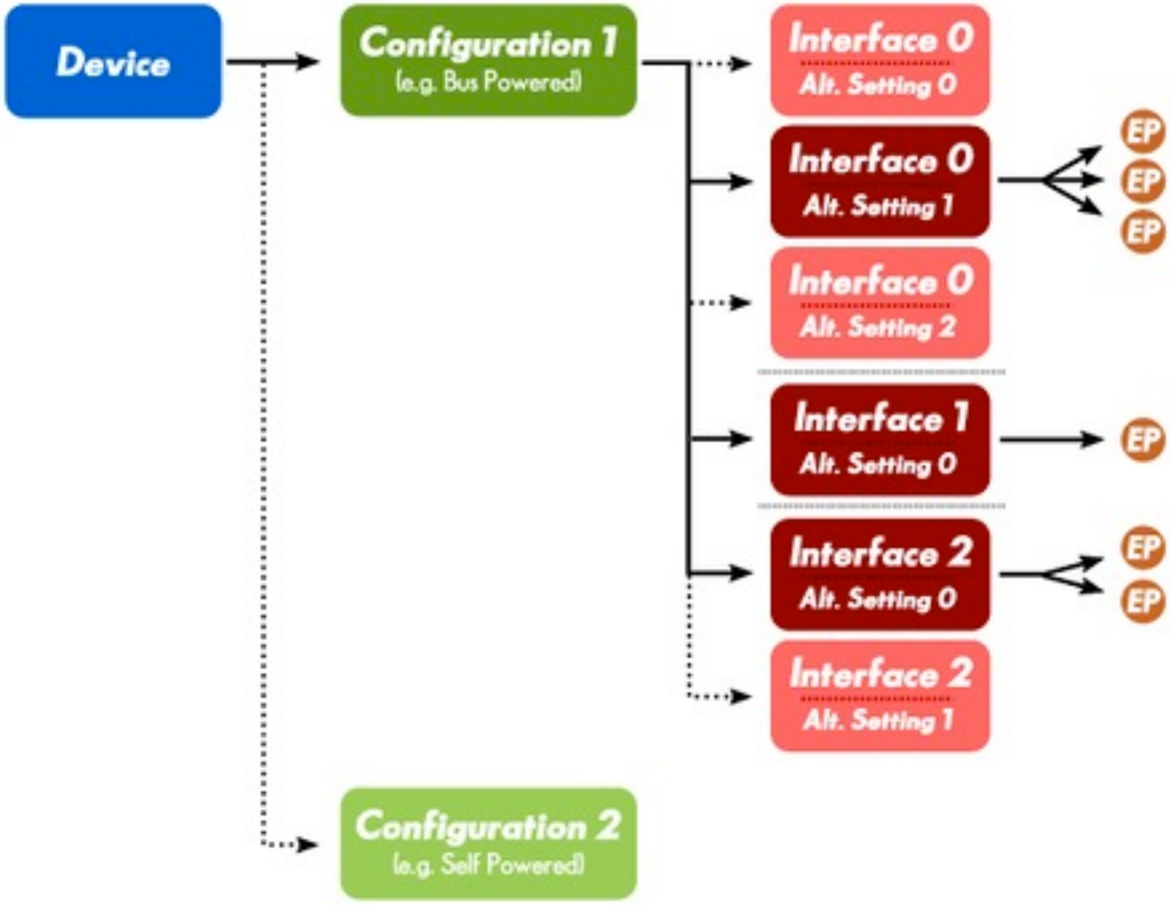
Standard USB Handshake

USB: Series of Events (Overview)


Interrupt notifying the host that a device connected
Get Device Descriptor
The host send *Get Device Descriptor setup request*
Speed, VendorID, ProductID, Serial No., Manufacture
The peripheral identifies itself
Get Configuration
The host setup kernel data structures of the device descriptor
Mass-storage, USB ether etc.
The peripheral supply the configuration, can be dynamically changed in smart gadget
Get Interface Descriptor
The host continues enumerate all the interfaces
USB Interface Class, Subclass, Protocol
The peripheral specify interface information
The host sets up endpoints for every interface
USB data transfer starts
USB Host
USB Peripheral

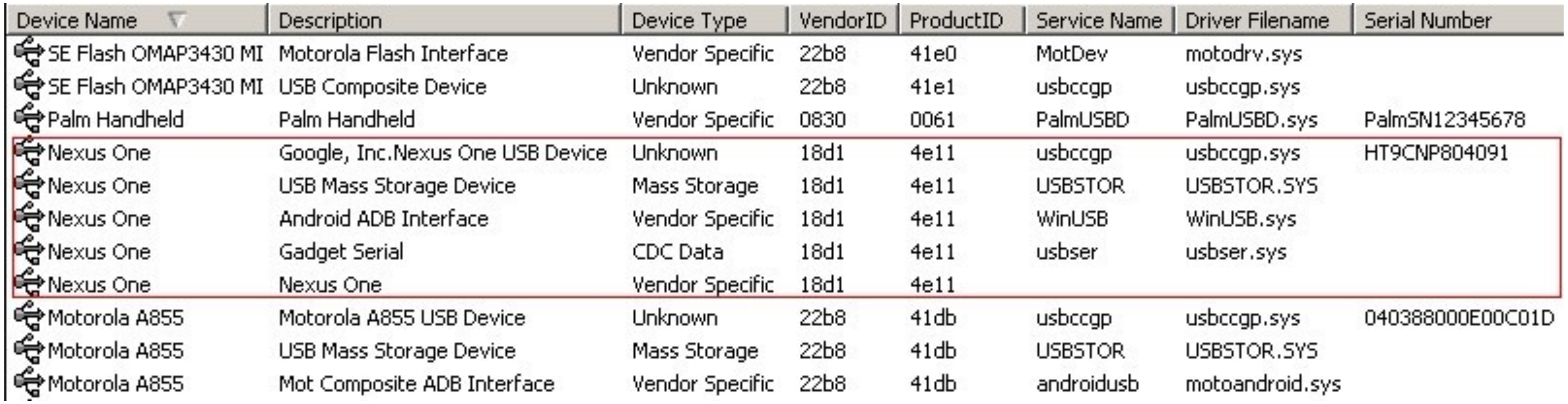
Standard USB Handshake

Device Configuration Map



USB Host Enumeration

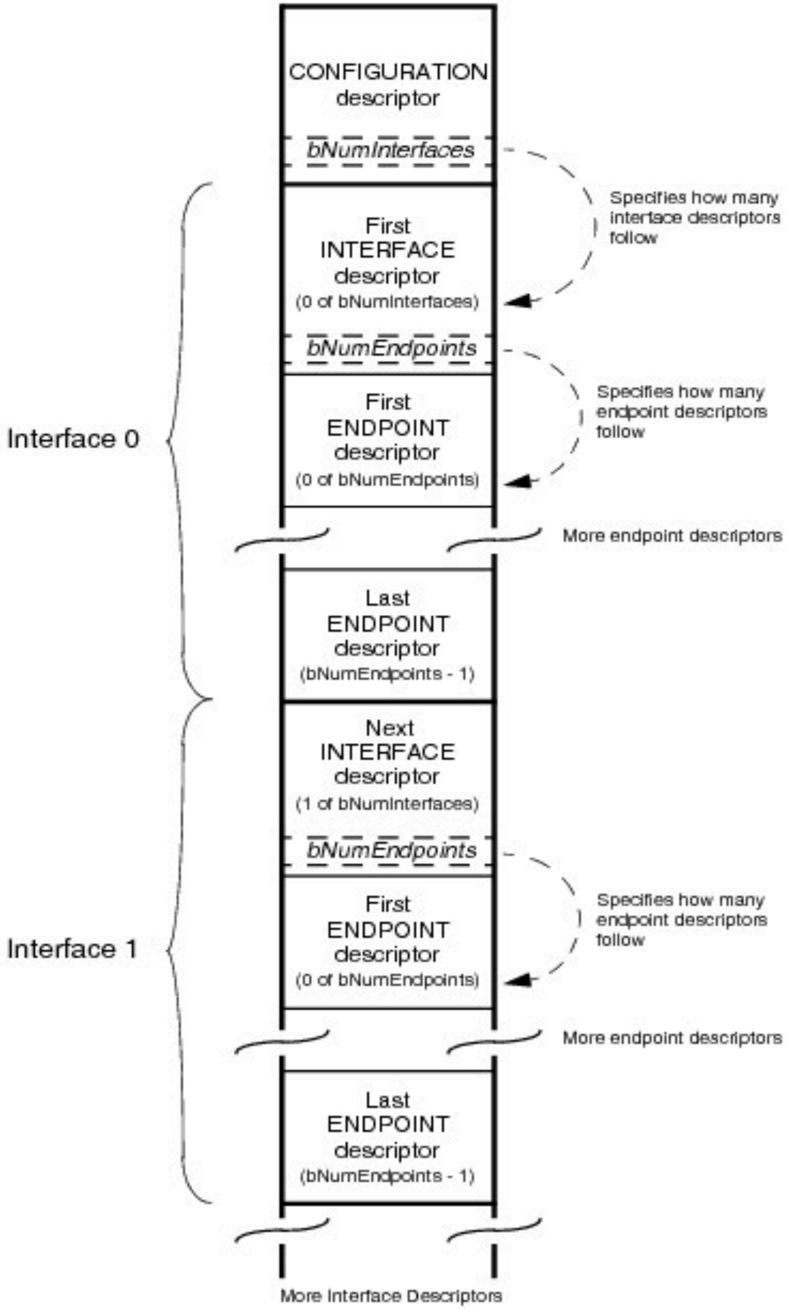
- Enumeration: How the host learns about devices
- All USB devices must support (HW/SW) control transfers, the standard requests, and endpoint zero.
- Smart gadgets are often composite devices
- Enumeration is transparent and automatic



Device Name	Description	Device Type	VendorID	ProductID	Service Name	Driver Filename	Serial Number
SE Flash OMAP3430 MI	Motorola Flash Interface	Vendor Specific	22b8	41e0	MotDev	motodrv.sys	
SE Flash OMAP3430 MI	USB Composite Device	Unknown	22b8	41e1	usbccgp	usbccgp.sys	
Palm Handheld	Palm Handheld	Vendor Specific	0830	0061	PalmUSBD	PalmUSBD.sys	PalmSN12345678
Nexus One	Google, Inc.Nexus One USB Device	Unknown	18d1	4e11	usbccgp	usbccgp.sys	HT9CNP804091
Nexus One	USB Mass Storage Device	Mass Storage	18d1	4e11	USBSTOR	USBSTOR.SYS	
Nexus One	Android ADB Interface	Vendor Specific	18d1	4e11	WinUSB	WinUSB.sys	
Nexus One	Gadget Serial	CDC Data	18d1	4e11	usbser	usbser.sys	
Nexus One	Nexus One	Vendor Specific	18d1	4e11			
Motorola A855	Motorola A855 USB Device	Unknown	22b8	41db	usbccgp	usbccgp.sys	040388000E00C01D
Motorola A855	USB Mass Storage Device	Mass Storage	22b8	41db	USBSTOR	USBSTOR.SYS	
Motorola A855	Mot Composite ADB Interface	Vendor Specific	22b8	41db	androidusb	motoandroid.sys	

USB Enumeration Hierarchy

- Device
  - Configuration
    - Interface
      - Endpoint


CONFIGURATION descriptor
bNumInterfaces
Specifies how many interface descriptors follow
First INTERFACE descriptor (0 of bNumInterfaces)
bNumEndpoints
Specifies how many endpoint descriptors follow
First ENDPOINT descriptor (0 of bNumEndpoints)
Interface 0
More endpoint descriptors
Last ENDPOINT descriptor (bNumEndpoints - 1)
Next INTERFACE descriptor (1 of bNumInterfaces)
bNumEndpoints
Specifies how many endpoint descriptors follow
First ENDPOINT descriptor (0 of bNumEndpoints)
Interface 1
More endpoint descriptors
Last ENDPOINT descriptor (bNumEndpoints - 1)
More Interface Descriptors

- Configuration changes the ProductID
  - USB debugging will Change N1's ProductID from 4e11 to 4e12)

Demo Demo Demo

- Show Exploitation of Computer using the phone as Keyboard

- Android based

  but *any* smart phone device with modern USB controller can perform the attack

- *Any* operating system is vulnerable, core functionality not just a hack

- We can lunch, reboot, redirect, …

Discussion

- USB connections are unprotected in current USB 1.1/2.0/3.0 protocol

- USB is the new venue for emerging exploits due to trust in physical proximity

- Smart gadget can cause more damages than traditional passive USB devices.

- Mutual USB authentication

- Revise the USB protocol for security features

Phone-to-Computer Defenses

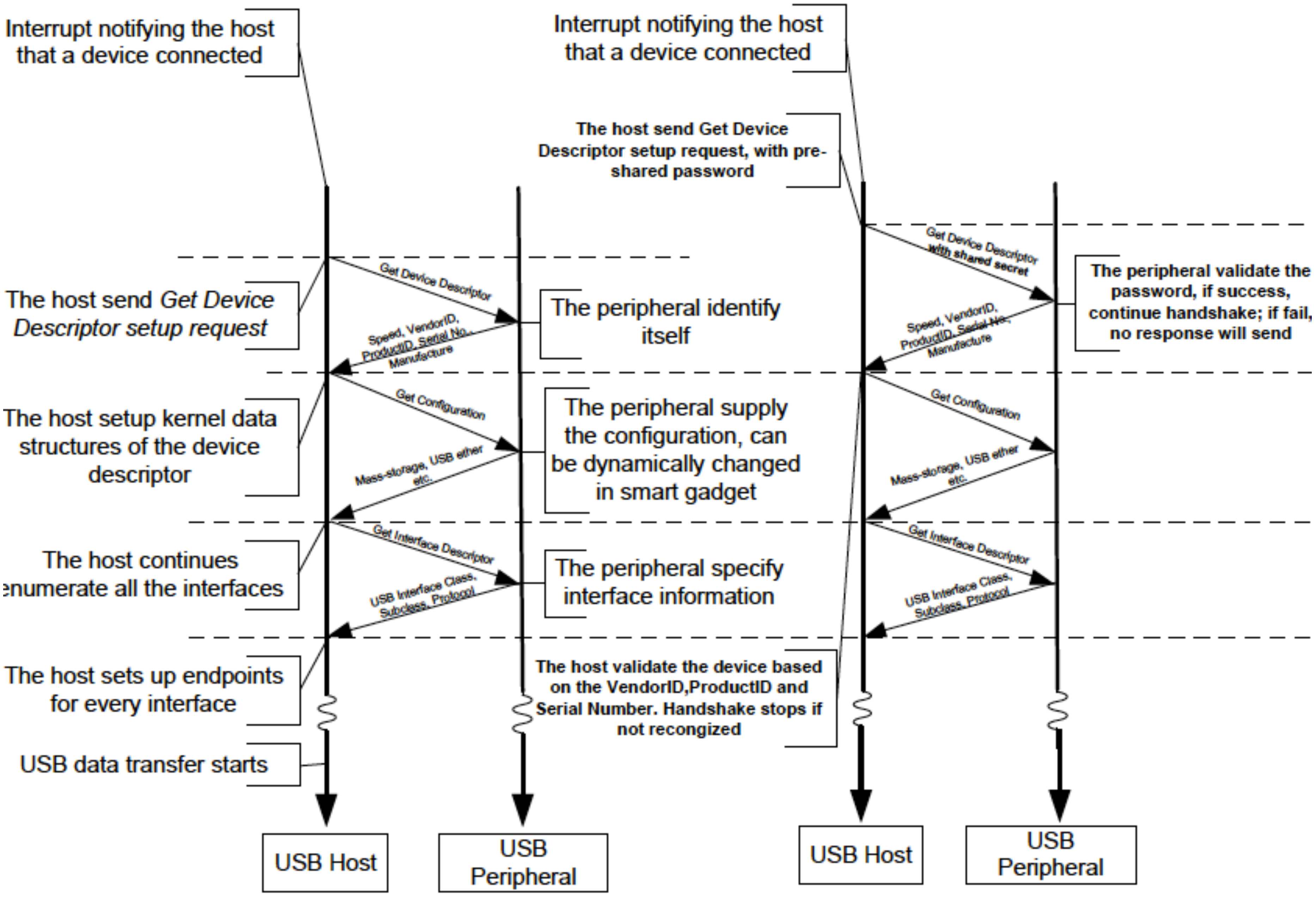
Potential Defense Strategies

- Disable autorun on USB storage device
  - MS KB971029, non mandatory

- Disable all USB storage devices from automatically attaching
  - MS KB823732

- Validate the Autenticity of the USB Devices once upon connect
  - Bluetooth devices
  - Does not prevent attacks from corrupted devices

Discussion – Defenses?


Get Device Descriptor
Speed, VendorID, ProductID, Serial No., Manufacture
Get Configuration
Mass-storage, USB ether etc.
Get Interface Descriptor
USB Interface Class, Subclass, Protocol
USB Host
USB Peripheral
Get Device Descriptor with shared secret
Speed, VendorID, ProductID, Serial No, Manufacture
Get Configuration
Mass-storage, USB ether etc.
Get Interface Descriptor
USB Interface Class, Subclass, Protocol
USB Host
USB Peripheral

Interrupt notifying the host that a device connected

The host send *Get Device Descriptor setup request*

The peripheral identify itself

The host setup kernel data structures of the device descriptor

The peripheral supply the configuration, can be dynamically changed in smart gadget

The host continues enumerate all the interfaces

The peripheral specify interface information

The host sets up endpoints for every interface

USB data transfer starts

Interrupt notifying the host that a device connected

The host send Get Device Descriptor setup request, with pre-shared password

The peripheral validate the password, if success, continue handshake; if fail, no response will send

The host validate the device based on the VendorID,ProductID and Serial Number. Handshake stops if not recongized

a) Standard USB Handshake

b) USBSec I Handshake

Discussion– Defenses?

- Adding static token authentication is not enough
  - Guessable
  - Easy to bypass (wait for the USB device to get authenticated, swap to another device)
  - Data Exfiltration
- Mutual Dynamic Authentication is good but…
  - Passive and Dumb devices cannot cope with
  - Many devices support partially the protocols
  - Windows USB-Hub subsystem a problem…

Discussion– Defenses?

- Getting the Human in the loop
  - Bluetooth has tried that
  - It works but only to validate the device it cannot prevent a device which is "approved" but compromized from corrupting and taking over the other end.
- The Solution requires Human to verify both Type of Device and restrict its permissions
  - Very very difficult given the current user body
  - Can only be applied to enterprise settings
  - Disabling the USB not an option (Why? Recharging…)