Further adventures with USB

13/01/2012

Andy Davis

Research Director

Telephone: +44 (0) 208 401 0070
e-mail: andy.davis@ngssecure.com

NCC Group Plc, Manchester Technology Centre, Oxford Road, Manchester M1 7EF **www.nccgroup.com**

ngssecure
an ncc group company

Agenda

- **USB Primer**
- Previous work
- USB vulnerability classes
- Phase one – Fuzzbox
- Phase two – Commercial test equipment
- Phase three – Frisbee
- Phase four – Frisbee Lite
- Security impact
- Conclusion

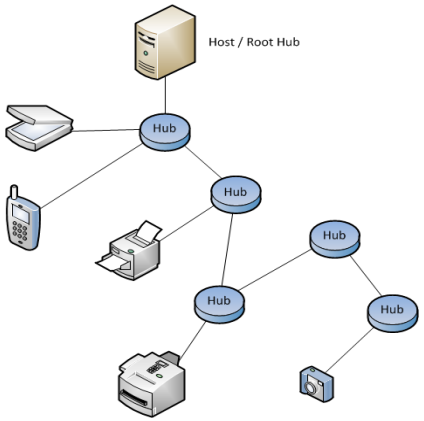
ngssecure
an ncc group company

USB – it's everywhere!

- PCs
- Tablets
- Cameras
- TVs
- Cable / Satellite set-top-boxes
- Telephones
- Microwave ovens!

USB Primer - Architecture

- The aim of USB was to find a solution to the mixture of connection methods to the PC



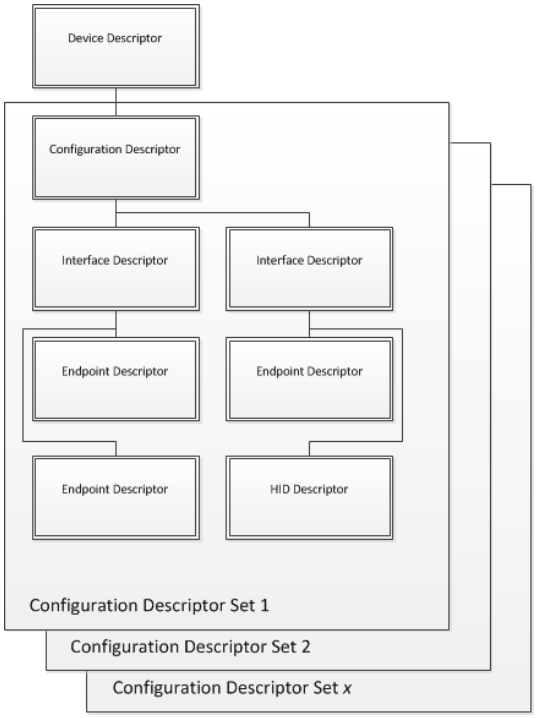
- Currently three speeds of data transfer (USB 2.0):
  - Low – 1.5Mbps (keyboard, mouse etc.)
  - Full – 12Mbps (originally for all other devices)
  - High – 480Mbps (developed in response to FireWire)
- Architecture is a tiered star topology:
  - Single host controller and up to 127 slave devices
  - A device can be plugged into a hub, and that hub can be plugged into another hub and so on. The maximum number of tiers permitted is six
- Host is Master - all communications on this bus are initiated by the host
- Devices cannot communicate directly with other devices (except for USB On-The-Go protocol)

USB Primer - Terminology

- The USB bus
  - When the host is transmitting a packet of data, it is sent to every device connected to an enabled port. It travels downwards via each hub in the chain which resynchronises the data transitions as it relays it. Only one device, the addressed one, actually accepts the data
- Endpoints
  - Each USB device has a number of endpoints. Each endpoint is a source or sink of data. A device can have up to 16 OUT and 16 IN endpoints.
  - OUT always means from host to device.
  - IN always means from device to host.
  - Endpoint 0 is a special case which is a combination of endpoint 0 OUT and endpoint 0 IN, and is used for controlling the device.
- Pipe
  - A logical data connection between the host and a particular endpoint, in which we ignore the lower level mechanisms for actually achieving the data transfers.

USB Primer – More Terminology

- Configurations, Interfaces, and Endpoints
  - The device contains a number of **descriptors** (as shown to the right) which help to define the device's capabilities
  - A device can have more than one **configuration**, though only one at a time, and to change configuration the whole device would have to stop functioning
  - A device can have one or more interfaces. Each **interface** can have a number of endpoints and represents a functional unit belonging to a particular class
  - Each **endpoint** is a source or sink of data



ngssecure
an ncc group company

USB Primer – plugging in a device

- **Pull-up resistor on data line** – Indicates device was connected (**reset device**)
- **Get Device descriptor** – what's the max packet size? – address 0
- **Reset then Set Address** - for the rest of the communications use this address
- **Get Device descriptor** – What are the device basic capabilities?

HCI Driver

- **Get Configuration descriptor** – What are the configuration details?
  - Interface descriptors
  - Endpoint descriptors
  - HID descriptors
- **Get String descriptors** – Get string language + product name etc.
- **Set Configuration** – Configuration is chosen – the device can be used

USB Bus Driver

- **Class-specific communication** - from this point onwards

USB Device Driver

USB Primer – Example descriptor

A typical Device Descriptor

Field	Value	Meaning
bLength	18	Valid Length
bDescriptorType	1	DEVICE
bcdUSB	0x0200	Spec Version
bDeviceClass	0x00	Class Information in Interface Descriptor
bDeviceSubClass	0x00	Class Information in Interface Descriptor
bDeviceProtocol	0x00	Class Information in Interface Descriptor
bMaxPacketSize0	64	Max EP0 Packet Size
idVendor	0x05AC	Apple Computer
idProduct	0x1297	Unknown
bcdDevice	0x0001	Device Release No
iManufacturer	1	Index to Manufacturer String (Not known)
iProduct	2	Index to Product String "iPhone"
iSerialNumber	3	Index to Serial Number String
bNumConfigurations	4	Number of Possible Configurations

Agenda

- USB Primer
- **Previous work**
- USB vulnerability classes
- Phase one – Fuzzbox
- Phase two – Commercial test equipment
- Phase three – Frisbee
- Phase four – Frisbee Lite
- Security impact
- Conclusion

ngssecure
an ncc group company

Various approaches to interface fuzzing

- **Rafael Dominguez Vega** used a number of different approaches including USB over IP, QEMU and a microcontroller to discover Linux-based USB vulnerabilities in String descriptors
- **David Dewey and Darrin Barrall** used an SL811 USB controller to discover a heap overflow in Windows XP
- **Moritz Jodeit** emulated USB devices in software and with a Netchip NET2280 peripheral controller
- **Tobias Mueller** used virtualisation techniques with QEMU to identify USB vulnerabilities in a number of different operating systems
- **Jon Larimer** investigated USB drive security and associated file system vulnerabilities in Windows and Linux

Difficulties in testing USB

- Black box Testing of drivers for interfaces on embedded systems can be tough:
  - No low-level view of the firmware
  - Hard to debug
  - Hard to hook into firmware
- Testing some interfaces is tough using conventional approaches:
  - Getting a PC to pretend to be a USB device requires some serious driver development – need to turn the root hub into a USB device.

Agenda

- USB Primer
- Previous work
- **USB vulnerability classes**
- Phase one – Fuzzbox
- Phase two – Commercial test equipment
- Phase three – Frisbee
- Phase four – Frisbee Lite
- Security impact
- Conclusion

ngssecure
an ncc group company

USB vulnerability classes

- Stack overflows
  - String descriptors in Linux and Windows XP
  - **Solaris 11 Express hub-class descriptor**
- Integer and Heap overflows
  - Sony PS3 hub-class descriptors
  - Apple iPod Touch jailbreak "0xA1,1" USB control message
  - **Xbox 360 image-class descriptors**
- Null-pointer dereference (exploitable)
  - Apple iPhone jailbreak "0x21,2" USB control message
- Logic errors
  - **Windows 7 HID report descriptors**

ngssecure
an ncc group company

Agenda

- USB Primer
- Previous work
- USB vulnerability classes
- **Phase one – Fuzzbox**
- Phase two – Commercial test equipment
- Phase three – Frisbee
- Phase four – Frisbee Lite
- Security impact
- Conclusion

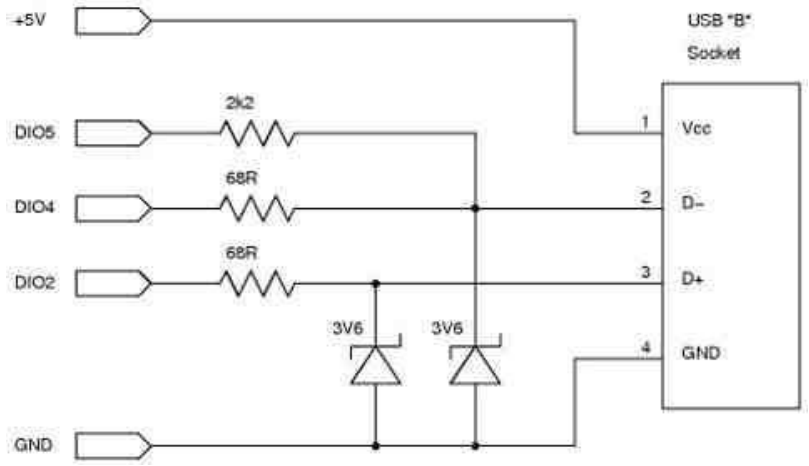
ngssecure
an ncc group company

Phase one - FuzzBox

- Arduino microcontroller
- Fuzzer written in C++
- Only emulates USB HID devices
- Only allows semi-automated fuzzing
- Has found bugs in:
  - Windows 7
  - Windows XP
  - OS X
- Limitations – not really fast enough to emulate most USB devices

FuzzBox details

Simple circuit to interface with an Arduino:



Driver code had already been written to emulate a HID device:
http://code.google.com/p/vusb-for-arduino/

Agenda

- USB Primer
- Previous work
- USB vulnerability classes
- Phase one – Fuzzbox
- **Phase two – Commercial test equipment**
- Phase three – Frisbee
- Phase four – Frisbee Lite
- Security impact
- Conclusion

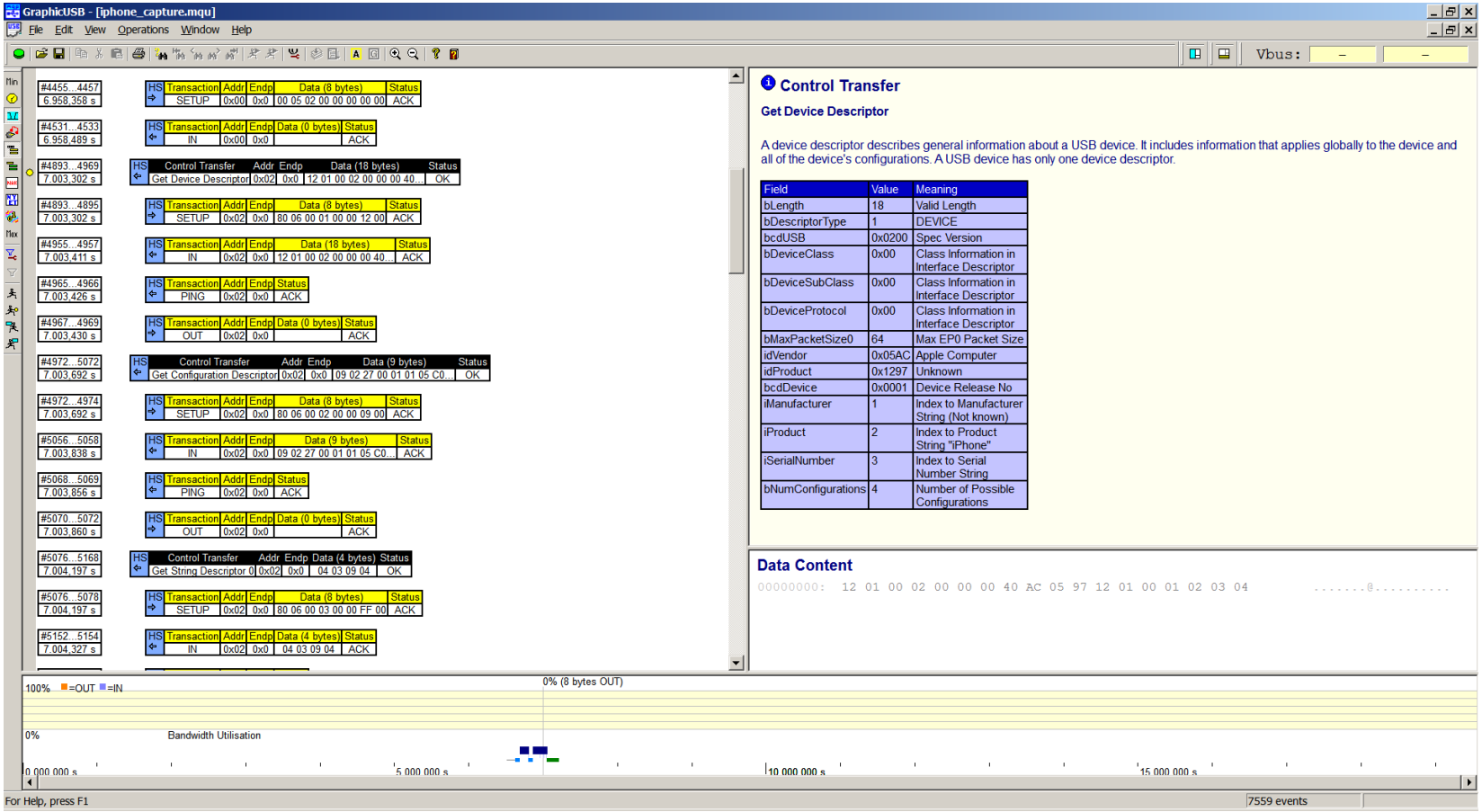
ngssecure
an ncc group company

Packet Master USB500 AG

- Dedicated USB test equipment hardware
- **USB capture and playback**
- **Emulates any USB host or device**
- Understands and analyses the different USB device classes
- **Uses a scripting language to generate USB traffic**
- Costs approx. US$1200 (plus specific class analysis options)
- **Limitations – doesn't currently have a software API to control it**

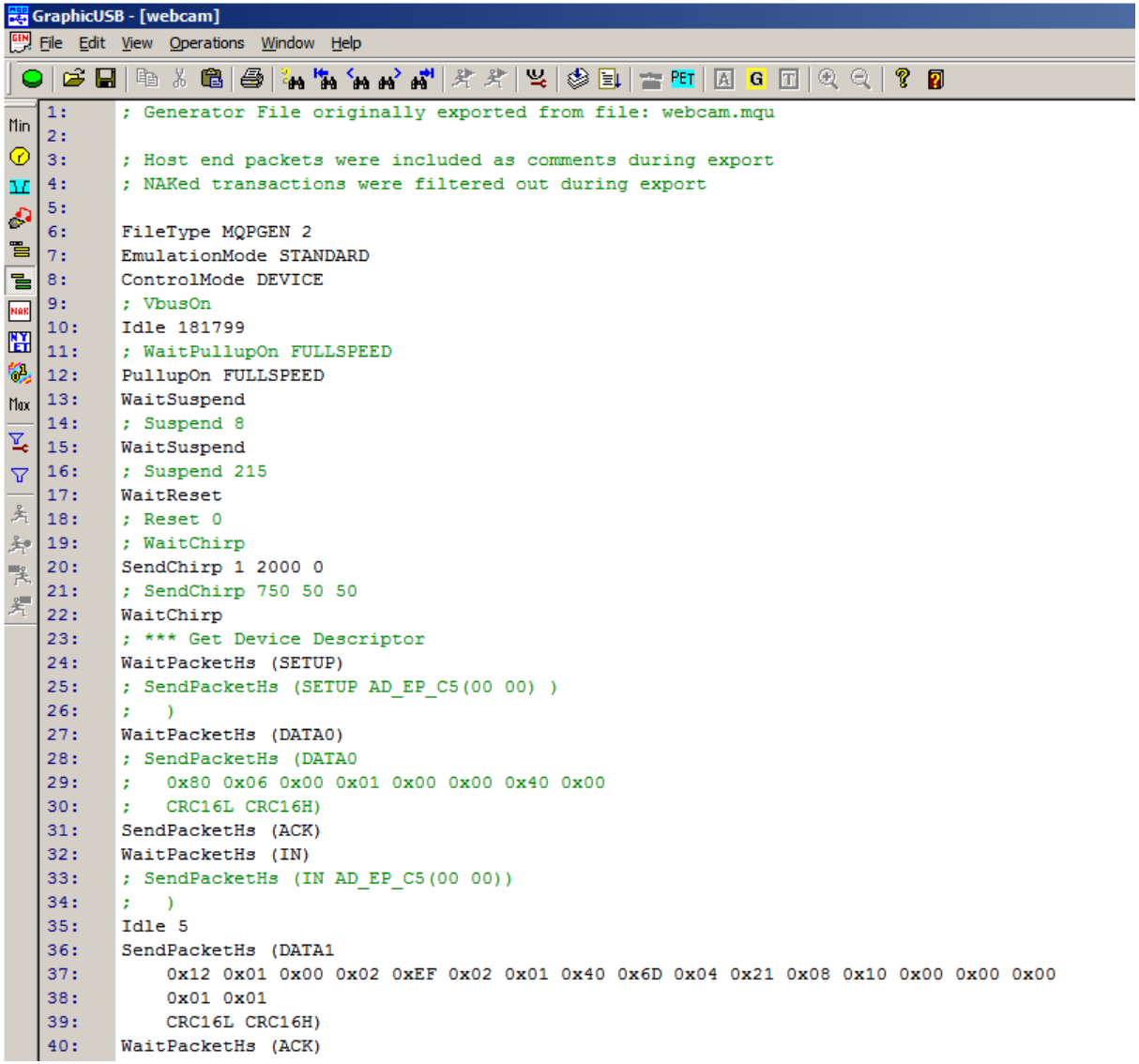
ngssecure
an ncc group company

GraphicUSB – Capture and Analysis



ngssecure
an ncc group company

GraphicUSB – Generator Scripts




```
1:  ; Generator File originally exported from file: webcam.mqu
2:
3:  ; Host end packets were included as comments during export
4:  ; NAKed transactions were filtered out during export
5:
6:  FileType MQPGEN 2
7:  EmulationMode STANDARD
8:  ControlMode DEVICE
9:  ; VbusOn
10: Idle 181799
11: ; WaitPullupOn FULLSPEED
12: PullupOn FULLSPEED
13: WaitSuspend
14: ; Suspend 8
15: WaitSuspend
16: ; Suspend 215
17: WaitReset
18: ; Reset 0
19: ; WaitChirp
20: SendChirp 1 2000 0
21: ; SendChirp 750 50 50
22: WaitChirp
23: ; *** Get Device Descriptor
24: WaitPacketHs (SETUP)
25: ; SendPacketHs (SETUP AD_EP_C5(00 00) )
26: ;   )
27: WaitPacketHs (DATA0)
28: ; SendPacketHs (DATA0
29: ;   0x80 0x06 0x00 0x01 0x00 0x00 0x40 0x00
30: ;   CRC16L CRC16H)
31: SendPacketHs (ACK)
32: WaitPacketHs (IN)
33: ; SendPacketHs (IN AD_EP_C5(00 00))
34: ;   )
35: Idle 5
36: SendPacketHs (DATA1
37:     0x12 0x01 0x00 0x02 0xEF 0x02 0x01 0x40 0x6D 0x04 0x21 0x08 0x10 0x00 0x00 0x00
38:     0x01 0x01
39:     CRC16L CRC16H)
40: WaitPacketHs (ACK)
```

ngssecure
an ncc group company

So what can we fuzz?

USB hosts:

- The Device Descriptor (HCI driver)
- String Descriptors (HCI driver / USB device drivers)
- The Configuration Descriptor: Interface Descriptor (USB device driver)
- The Configuration Descriptor: Endpoint Descriptor (USB device driver)
- Class-specific descriptors e.g. HID Report Descriptor (USB device driver)
- Class-specific communication (USB device driver and client software)

USB devices:

- Host -> device USB control messages

Why the new approach is different

Device and Platform Independent:

- Previous approaches have mainly focussed on either testing or emulating specific devices or hosts
- We don't care (for the most part) what the host or device is
- If we understand USB and where vulnerabilities may be present then we can use a black-box fuzzing approach
- It certainly produces results…

ngssecure
an ncc group company

Agenda

- USB Primer
- Previous work
- USB vulnerability classes
- Phase one – Fuzzbox
- Phase two – Commercial test equipment
- **Phase three – Frisbee**
- Phase four – Frisbee Lite
- Security impact
- Conclusion

ngssecure
an ncc group company

Frisbee – the USB fuzzer

- GraphicUSB currently has no API for controlling it
- Frisbee modifies the GraphicUSB generator script created from a capture
- The generator script must be loaded, compiled then executed to generate USB traffic
- Frisbee uses the *SendKeys* Python library to inject Windows events into GraphicUSB
- Although this is slow, the HCI driver needs time to identify a device, which has ostensibly just been plugged in
- The host under test is connected to the same Ethernet network as the host running Frisbee and is periodically pinged to identify if the host has crashed

Demos

ngssecure
an ncc group company

What have we found so far?

- HID class descriptor memory corruption in **Windows 7**
- Hub class descriptor stack overflow in **Solaris 11Express**
- Printer class communication memory corruption in **Solaris 11Express**
- Image class communication memory corruption in **Solaris 11Express**
- Image class communication integer overflows in **Microsoft Xbox 360**
- Image class descriptor memory corruption in **Apple OS X**

Reusing the Arduino approach

Small, portable, programmable USB device to trigger USB vulnerabilities:

Why are USB vulnerabilities still prevalent?

- Some vendors don't seem to view USB vulnerabilities as a security issue

Quote from vendor x:

"Thank you for sending this to us. This is something that I will definitely pass on, however since this requires physical access its not something that we will fix in a security update"

- The knowledge of how to emulate USB devices is not widespread

Quote from vendor y:

"We think we've fixed this issue, but we'll need to get you to test it as we don't have the ability to replicate your attack"

ngssecure
an ncc group company

Agenda

- USB Primer
- Previous work
- USB vulnerability classes
- Phase one – Fuzzbox
- Phase two – Commercial test equipment
- Phase three – Frisbee
- **Phase four – Frisbee Lite**
- Security impact
- Conclusion

ngssecure
an ncc group company

Frisbee Lite – The USB device fuzzer

A bit more USB primer:

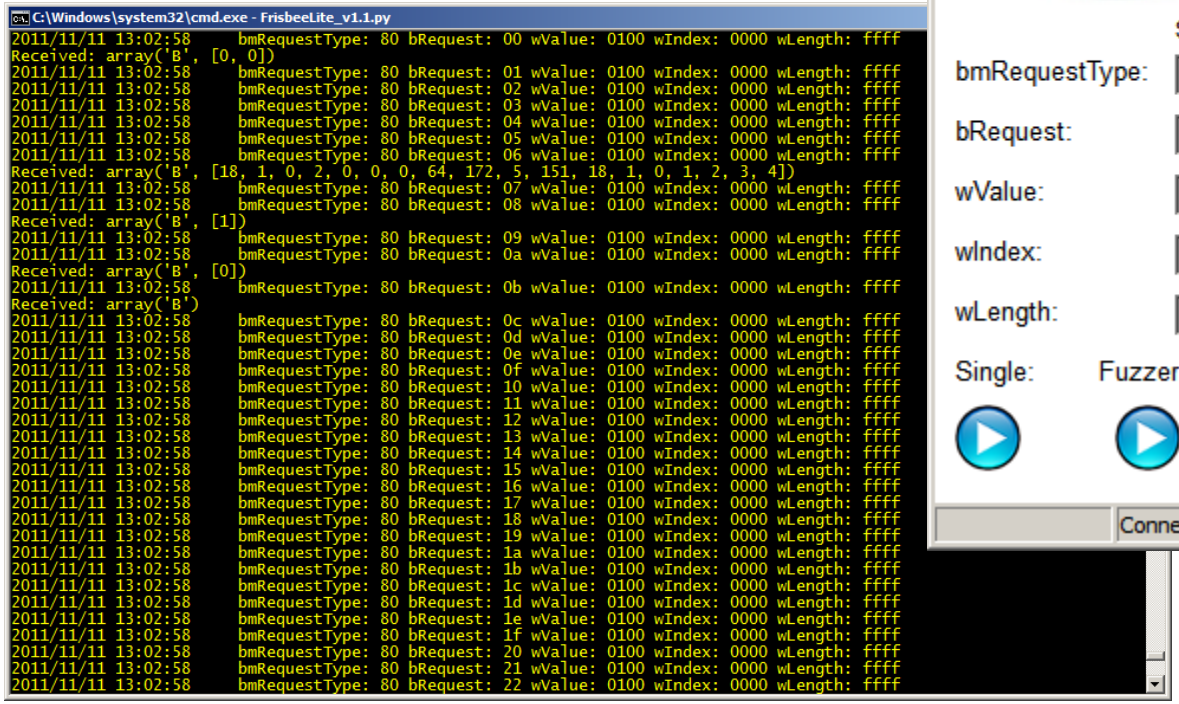
- bmRequestType
  - This is a bitmapped field that describes the characteristics of the request.
- bRequest
  - This is the actual request that is being sent
- wValue
  - The contents of this field are request-specific.
- wIndex
  - The contents of this field are request-specific. However, it is often used to specify an endpoint or interface.
- wLength
  - This specifies the length of the data transferred during the second phase of the control transfer. If this field is zero, there is no second (data transfer) phase.

USB standard requests

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00000000B 00000001B 00000010B	CLEAR_FEATURE	Feature Selector	Zero Interface Endpoint	Zero	None
10000000B	GET_CONFIGURATION	Zero	Zero	One	Configuration Value
10000000B	GET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero or Language ID	Descriptor Length	Descriptor
10000001B	GET_INTERFACE	Zero	Interface	One	Alternate Interface
10000000B 10000001B 10000010B	GET_STATUS	Zero	Zero Interface Endpoint	Two	Device, Interface, or Endpoint Status
00000000B	SET_ADDRESS	Device Address	Zero	Zero	None
00000000B	SET_CONFIGURATION	Configuration Value	Zero	Zero	None
00000000B	SET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero or Language ID	Descriptor Length	Descriptor
00000000B 00000001B 00000010B	SET_FEATURE	Feature Selector	Zero Interface Endpoint	Zero	None
00000001B	SET_INTERFACE	Alternate Setting	Interface	Zero	None
10000010B	SYNCH_FRAME	Zero	Endpoint	Two	Frame Number

Frisbee Lite – The USB device fuzzer

The user interface is simple and intuitive:



ngssecure
an ncc group company

Frisbee Lite – The USB device fuzzer

- Written in wxPython
- Uses LibUSB to provide low level access to USB on Windows
- Uses pyUSB to send raw USB setup request packets
- Download from: http://www.ngssecure.com/research/research-overview/Public-Tools.aspx
- Send all bugs / feedback / feature requests to me

ngssecure
an ncc group company

Bugs discovered by Frisbee Lite

- A memory corruption vulnerability in iOS 5 (iPhone 4) causes a kernel panic – potentially exploitable
- A memory corruption vulnerability in iOS 5 (iPhone 4) causes the USB stack to stop responding – exploitability currently being investigated

ngssecure
an ncc group company

Agenda

- USB Primer
- Previous work
- USB vulnerability classes
- Phase one – Fuzzbox
- Phase two – Commercial test equipment
- Phase three – Frisbee
- **Security impact**
- Conclusion

ngssecure
an ncc group company

The impact of exploiting USB

Typical scenarios of exploiting USB vulnerabilities:

- "Jailbreaking" embedded devices
- Unlocking a locked workstation
- Installing malware in seconds
- Exfiltrating sensitive data in seconds
- What about Endpoint Protection Systems?...

ngssecure
an ncc group company

Endpoint Protection software

Endpoint protection software:

- Lumension (formerly Sanctuary) - Device Control
- CoSoSys – Endpoint Protector
- DeviceLock - Endpoint DLP Suite

They work higher up the USB stack than many USB driver vulnerabilities exist

Exploitable USB driver vulnerabilities will often circumvent them

Some of the vulnerabilities we have discovered, when exploited will circumvent Endpoint Protection Software

Agenda

- USB Primer
- Previous work
- USB vulnerability classes
- Phase one – Fuzzbox
- Phase two – Commercial test equipment
- Phase three – Frisbee
- Security impact
- **Conclusion**

ngssecure
an ncc group company

Conclusion

- There are still plenty of USB vulnerabilities out there – many of which will be exploitable
- Even though limited physical access is required, USB vulnerabilities still constitute a serious security risk
- Endpoint Protection software is unlikely to protect you from many of these attacks
- If you really don't want people to exploit USB on your PCs, disable USB in the BIOS
- For ultimate protection, fill the USB sockets with epoxy resin!

ngssecure
an ncc group company

Questions?

Andy Davis

Research Director

Telephone: +44 (0) 208 401 0070
e-mail: andy.davis@ngssecure.com

NCC Group Plc, Manchester Technology Centre, Oxford Road, Manchester M1 7EF **www.nccgroup.com**

ngssecure
an ncc group company