This presentation

- Existing USB HID attacks / tools
- What's wrong with them?
- Available protections / mitigations
- My implant PoC
- Putting it into an attack scenario
- Demo-time...

About me

Niels van Dijkhuizen
Existing USB HID attacks / tools

KeyGhost USB keylogger

2005

PHUKD - Irongeek
HAK5 USB Rubber Ducky

Keylogger/PHUKD Hybrid

2010

BadUSB - Karsten Nohl
USBDriveby - Samy Kamkar

BadUSB 2.0
David Kierznowski

2014

2016

HAK5 Bash Bunny
Cactus WHID injector

2017
What's wrong with it?

Kind of in-your-face!
What's wrong with it?

Requires either:

- An unlocked and unattended computer
- Very good social engineering skills

- Many payloads require direct internet access
- Protection available
### Available protection mechanisms

<table>
<thead>
<tr>
<th>Protection Mechanism</th>
<th>Developer</th>
</tr>
</thead>
<tbody>
<tr>
<td>USG</td>
<td>Robert Fisk</td>
</tr>
<tr>
<td>USB Proxy</td>
<td>Dominic Spill</td>
</tr>
<tr>
<td>USB Guard</td>
<td>Daniel Kopeček</td>
</tr>
<tr>
<td>GoodDOG</td>
<td>Tony DiCola</td>
</tr>
<tr>
<td>Beamgun</td>
<td>Josh Lospinoso</td>
</tr>
<tr>
<td>USB keyboard guard</td>
<td>G Data</td>
</tr>
<tr>
<td>Linux patches</td>
<td>GRSecurity</td>
</tr>
<tr>
<td>Duckhunt</td>
<td>Pedro M. Sosa</td>
</tr>
</tbody>
</table>
A new implant?

a HID attack that works with locked machines and bypasses known protection mechanisms
Design requirements

1. The implant should be in-line with the keyboard and the host.

2. The implant should have notion of real-time.
and spice it up a bit...

3. The implant could use an over the air communication channel.
Hardware diagram

Teensy 2.0: Arduino-based MC
- Maxim MAX3421E
- Maxim DS3132
- 433Mhz UART
- EEPROM (nvram)
Keynterceptor PoC HW

Could have been smaller™
Keynterceptor PoC HW
**Bypassing protections: Device cloning**

USB Standard Descriptor & USB HID Report Descriptors

---

### Interface Descriptor:

<table>
<thead>
<tr>
<th>bLength</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>bDescriptorType</td>
<td>4</td>
</tr>
<tr>
<td>bInterfaceNumber</td>
<td>1</td>
</tr>
<tr>
<td>bAlternateSetting</td>
<td>0</td>
</tr>
<tr>
<td>bNumEndpoints</td>
<td>1</td>
</tr>
</tbody>
</table>

### Interface Class:

<table>
<thead>
<tr>
<th>bLength</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>bDescriptorType</td>
<td>4</td>
</tr>
<tr>
<td>bInterfaceNumber</td>
<td>1</td>
</tr>
<tr>
<td>bAlternateSetting</td>
<td>0</td>
</tr>
<tr>
<td>bNumEndpoints</td>
<td>1</td>
</tr>
</tbody>
</table>
Bypassing protections: Human emulation

//Add random delays to avoid detection
int r = rand() % 111;

r += 8;
delay(r);
void setup() {
  Keyboard.begin();
  delay(1000);
  Keyboard.print("This is a typical super human typing speed!!!");
}

void loop() {  
}
# BOM / Costs

<table>
<thead>
<tr>
<th>Component</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teensy 2.0</td>
<td>$16,00</td>
</tr>
<tr>
<td>433 MHz module</td>
<td>$4,00</td>
</tr>
<tr>
<td>USB Host module</td>
<td>$8,00</td>
</tr>
<tr>
<td>DS3231 RTC</td>
<td>$4,00</td>
</tr>
<tr>
<td>MCP1825S regulator</td>
<td>$1,00</td>
</tr>
<tr>
<td>Exp. print / LEDs / resistors</td>
<td>$2,00</td>
</tr>
<tr>
<td><strong>Total in US Dollars:</strong></td>
<td><strong>$35,00</strong></td>
</tr>
<tr>
<td><strong>Total in Euro's:</strong></td>
<td><strong>€30,00</strong></td>
</tr>
</tbody>
</table>
Use-Cases

a. Control keyboard remotely OTA
b. Autologin with captured creds
c. Inject keystrokes after inactivity with chosen time-frame
d. Block user input with RF kill-switch (for a take-down)
e. <<insert scenario here>>
Add-on for a full attack scenario

Keynterceptor-Companion:
- Nanopi Neo
- 433 MHz
- 4G dongle

Niels van Dijkhuizen

Keynterceptor | NebulaH 2018
DEMO-Time...
New mitigations?

a. Multi-factor or challenge-response (like captcha’s) with every unlocking action ✓

b. Profiling / monitoring power consumption per device
   *(HW support is problematic)*
USBGuard + power profiling PoC
USBGuard + power profiling PoC

```
$ cat usbg-powermon.conf
--- # USBGuard Power Monitor configuration
monitorPort: /dev/ttyACM0
thresholds:
  16c0:0483: 20.0  # Teensyduino Serial [monitor port]
  03f0:034a: 48.5  # Hewlett-Packard Elite Keyboard
  046d:c018: 40.0  # Logitech, Inc. Optical Wheel Mouse

$ ./usbg-powermon.py
Measuring [16c0:0483] USB Serial: [--] port not assigned
Measuring [03f0:034a] HP Elite USB Keyboard: [OK]
```

USB Device Inserted
USB ID: 03f0:034a
Name: HP Elite USB Keyboard
Port: 7-1
USBGuard + power profiling PoC
USBGuard + power profiling PoC
Keynterceptor attack feasible?

- 430 lines of C code
- 85 lines of Python code
- 301 lines of Perl code
- Some development euro's

Future work?
- Fit it inside real hardware
- Have automatic descriptor cloning
- Encrypt covert OTA communication channel